COMP 335 - Introduction to Theoretical Computer Science
Assignment 2

Ali Hanni - 40157164

Gina Cody School of Computer Science and Software Engineering
Concordia University, Montreal, QC, Canada

Fall 2021

Question 1

[5 pts] Give a regular expression for every string w over $\{0, 1\}$ where the value of w, when interpreted as a binary number, is at least 40.

Answer

In binary, 40 is: 101000.
A binary string of length 6 ranges from 0 to 63. Our regular expression must first cover all the possible values from 40 to 63:

$$0^*1(01+10+11)(0+1)^3$$

Note: 0^* *handles possible leading zeros*
This regular expression only handles binary string with values between 40 and 63 inclusive.

We must also handle number greater than 63. This second part of the problem can be seen as finding a regular expression for all binary strings whose value is greater than or equal to 64.
In binary string, 64 is 1000000.
We must accept all binary strings with any number of leading zeros, a 1 followed by at least 6 characters:

$$0^*1(0+1)^6(0+1)^*$$

Putting the two regular expression together and simplifying, we have:

$$0^*(1(0+1)^6(0+1)^*+1(01+10+11)(0+1)^3)$$

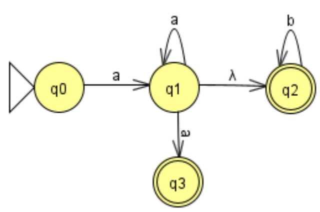
Question 2

[5 pts] Give a finite state automaton for the language denoted by the regular expression $r_1^*(r_2 + r_3)$, where $r_1 = a\emptyset a$, $r_2 = aa^*(a + b^*)$, and $r_3 = (b + ab)^*aa(b + ba)^*$.

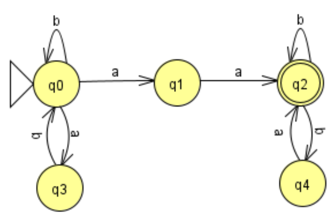
Answer

First, concerning $r_1 = a\emptyset a$, we know that with regular expressions, concatenating a string with an empty set always yields an empty set. Since star closure is applied on that expression, it ends up being an empty character since $\emptyset^* = \lambda$. In the context of this exercise r_1 can simply be ignored.

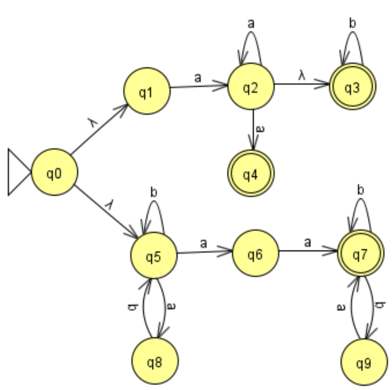
Then we have $r_2 = aa^*(a + b^*)$. The corresponding NFA would be:



Finally we have $r_3 = (b + ab)^*aa(b + ba)^*$:



Now to obtain $(r_1 + r_2)$ we need to put both NFAs as two distinct path in a new NFA. We can obtain the following result:

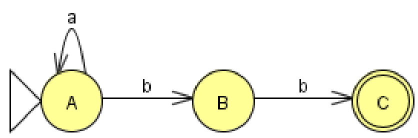


Question 3

[5 pts] Give a regular grammar for $L(a^*bb + ab^*ba)$

Answer

In order to easily solve the problem, we can treat the problem as two distinct regular expressions a^*bb and ab^*ba.
We first search for a regular expression for a^*bb. Constructing a finite automaton for this expression we have:

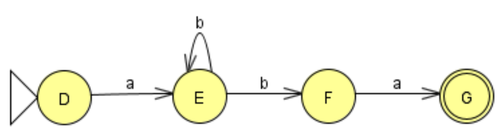


We can transform this finite state automaton into the following grammar using a known construction:

$$A \to aA|bB$$
$$B \to b$$

Which can be simplified to : $A \to aA|bb$

Now we find a finite automaton that corresponds to ab^*ba:



We easily derive the following regular grammar:

$$D \to aE$$
$$E \to bE|bF$$
$$F \to a$$

That we simplify to:

$$D \to aE$$
$$E \to bE|ba$$

Putting these grammars together we obtain:

$$
\begin{aligned}
A &\to aA|bb \\
D &\to aE \\
E &\to bE|ba
\end{aligned}
$$

For simplicity and clarity, we rearrange and rename the symbols:

$$
\begin{aligned}
S &\to A|aB \\
A &\to aA|bb \\
B &\to bB|ba
\end{aligned}
$$

Question 4

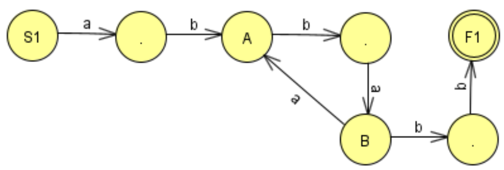
[10 pts] Consider the regular grammar $G_1 = (V_1, \Sigma, S_1, P_1)$, where $V_1 = \{S_1, A, B\}$, $\Sigma = \{a, b\}$, and P_1 consists of the productions $S_1 \to abA$, $A \to baB$, and $B \to aA|bb$. Also consider the grammar $G_2 = (\{S_2, B\}, \Sigma, S_2, P_2)$, where the productions in P_2 are:

$$S_2 \to aaB|\lambda$$
$$B \to bB|ab$$

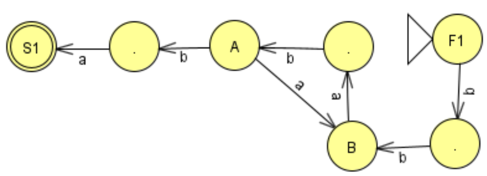
Give a left linear grammar for the language $L(G_1) \cup L(G_2)$.

Answer

First, let's find finite automaton that corresponds to the regular languages described by the regular grammars G_1 and G_2:
For G_1 we have:



We inverse all the edges and swap the initial and final state:



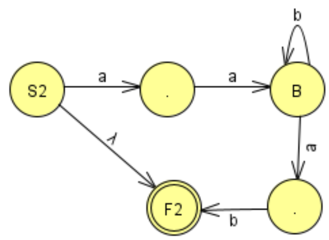
We now have the following grammar:

$$F1 \to bbB$$
$$B \to abA$$
$$A \to aB|ba$$

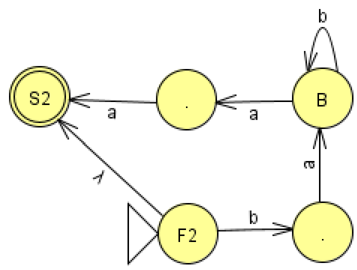
We inverse the symbols to obtain the following left-linear grammar:

$$F1 \to Bbb$$
$$B \to Aba$$
$$A \to Ba|ab$$

We repeat the same process for G_2:
Finite automaton for G_2:



We inverse all the edges and swap the initial and final state:



We now have the following grammar:

$$\begin{aligned} F2 &\rightarrow baB|\lambda \\ B &\rightarrow bB|aa \end{aligned}$$

We inverse the symbols to obtain the following left-linear grammar:

$$\begin{aligned} F2 &\rightarrow Bab|\lambda \\ B &\rightarrow Bb|aa \end{aligned}$$

We finally have to find the union of the found left-linear grammars. Let's call the new grammar G.We rename the non-terminals such that:
left-linear G_1:

$$\begin{aligned} F1 &\rightarrow B_1bb \\ B_1 &\rightarrow A_1ba \\ A_1 &\rightarrow B_1a|ab \end{aligned}$$

And left-linear G_2:

$$\begin{aligned} F2 &\rightarrow B_2ab|\lambda \\ B_2 &\rightarrow B_2b|aa \end{aligned}$$

The left-linear grammar G resulting from the union of left-linear G_1 and left-linear G_2:

$$
\begin{aligned}
S &\rightarrow F1|F2 \\
F1 &\rightarrow B_1bb \\
B_1 &\rightarrow A_1ba \\
A_1 &\rightarrow B_1a|ab \\
F2 &\rightarrow B_2ab|\lambda \\
B_2 &\rightarrow B_2b|aa
\end{aligned}
$$

Question 5

[15 pts] Suppose L is a regular language. Use the closure properties of regular languages to show that each of the following languages is regular as well.

(a) $\{uv : u \in L \text{ and } |v| = 2\}$

(b) $\{uv : u \in L \text{ and } v \in L^R\}$

(c) $\{u : u \in L \text{ and } u^R \in L\}$

Answer

(a) *Proof.* First, we have that $|v| = 2$. v is therefore a finite language, meaning it is necessarily regular. We are also told that $u \in L$ and L is regular, meaning that u is regular as well.
The concatenation closure property states that regular languages are closed under concatenation. Knowing that and since u and v are regular languages, we know that uv is necessarily a regular language. ■

(b) *Proof.* First, we know that u is a regular language since we are told that L is a regular language and that $u \in L$.
Also, since regular languages are closed under reversal, we know that if L is regular, L^R is also a regular language. Hence, since $v \in L^R$, v is also a regular language.
The concatenation closure property states that regular languages are closed under concatenation. Knowing that and since u and v are regular languages, we know that uv is necessarily a regular language. ■

(c) *Proof.* We are told that L is a regular language hence since $u \in L$, u is also a regular language. ■

Question 6

[20 pts] For each of the following languages, prove or disprove if it is regular.

(a) $L_1 = \{a^{2i+5j} : i, j \geq 0\}$

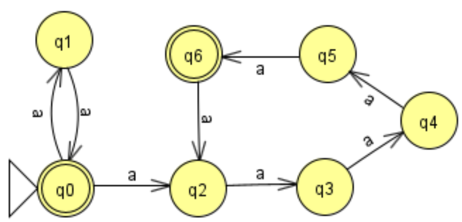
(b) $L_2 = \{w : w \in \{a, b\}^*$ and no two b's in w have odd number of a's between$\}$

(c) $L_3 = \{a^n : n = 3k$, for $k \geq 0\}$

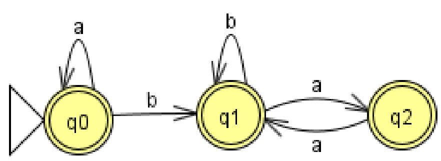
(d) $L_4 = \{a^n b^m : n \geq m\}$

Answer

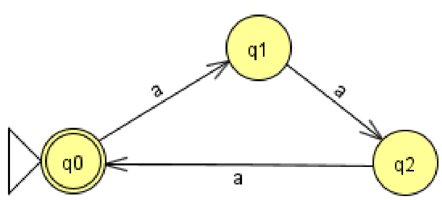
(a) The language is regular since there exists an NFA that corresponds to L_1. The NFA is shown bellow:



(b) The language is regular since there exists an NFA that corresponds to L_2. The NFA is shown bellow:



(c) The language is regular since there exists an NFA that corresponds to L_3. The NFA is shown bellow:



(d) *Proof.* We can prove that L_4 is not a regular language using the pumping lemma. In order to use the pumping lemma to prove that a language is not regular, we have to prove that there exists a string $w \in L_4$ where $|w| \geq m$ for $m > 0$, such that for all possible decomposition of w in xyz where $xyz = w$, $|xy| \leq m$, and $|y| \geq 1$ there exist at least one $i \geq 0$ such that $w = xy^iz \notin L_4$.

Let's assume that L_4 is a regular language.

We know that for all strings w element of L_4 and for any decomposition of w such that $w = xyz$, we have that xy^iz must also be element of L_4 for $i \geq 0$. Now let $k > 0$ and $a^kb^k \in L_4$.

We know that $|a^kb^k| = 2k \geq m$.
Now let $x = a^r$, $y = a^s$ and $z = a^{k-s-r}b^k$ for $r \geq 0$, $s \geq 1$ and $s + r \leq k$ such that:

$$\begin{aligned} w &= xyz \\ |xy| &= s + r \leq m \\ |y| &= s \geq 1 \end{aligned}$$

For $i = 0$:

$$\begin{aligned} xy^0z &= xz \\ xz &= a^ra^{k-s-r}b^k = a^{k-s}b^k \end{aligned}$$

Since $|s| \geq 1$ we know that $k > k - s$. The string $a^{k-s}b^k$ is thus not a valid string of L_4 which contradicts the initial suppositions. L_4 does not respect the pumping lemma and is therefore not a regular language. ■