Regular Expressions and Finite Automata[1]

CS142 Assignment 2, Tanha Kate

[1] **#responsibility:** In this assignment, I believe I have demonstrated #responsibility by starting well ahead of time and finishing major portions of the assignment before the weekend. As a result, I was able to consult with the professor about some confusions while also enjoying my trip. Through this, I showed commitment to my academics and an ability to balance my own plans with Minerva's requirements.

Section 1

Problem A

Consider $\Sigma = \{a, b\}$ $and\, \tau = \{a, b\}$
Let f: $\Sigma \rightarrow \tau^* be\ defined\ by\ f(a) = aa\ and\ f(b) = bb$

We will extend to string by f($a_1...a_n$) = f(a_1)..$f(a_n)$.

Therefore, f(ababa) = aabbaabbaa

Problem B

Consider $\Sigma = \{0, 1\}$ $and\, \tau = \{a, b\}$
Let f: $\Sigma \rightarrow \tau^* be\ defined\ by\ f(0) = ab\ and\ f(1) = \varepsilon$

We will extend to string by f($a_1...a_n$) = f(a_1)..$f(a_n)$.

Therefore, f(01010) = ababab
As another example, f(0*1) = (ab)*

Problem C

The class of regular languages is closed under a homomorphism. In other words, if L is a regular language and f is a homomorphism, then f(L) is also regular.

Proof.

At first, we will define homomorphism as a operation on regular expressions. Consider a regular expression R and let f(R) be the regular expression which replaces each occurence of a $\in \Sigma$ in R by the string f(a).

Formally, we define f(R) as:

$$f(\varnothing) = \varnothing \qquad f(R_1R_2) = f(R_1)f(R_2)$$

$f(\varepsilon) = \varepsilon$ $\quad$ $f(R_1 U R_2) = f(R_1) \, U \, f(R_2)$

f(a) = f(a) $\quad$ f(R*) = (f(R))*

Let L = L(R) for a regular expression R.

Hypothesis. We claim that L(f(R)) = f(L(R)).

Basis. If L is $\varnothing$ *or* ε, *then* $f(R) = R$, *and* $L(f(R)) = L(R) = f(L(R))$

If L is a, then L(R) = {a}, L(f(R)) = L(f(a)) = {f(a)} = f(L(R)).

Induction.

Case 1: $R = R_1 + R_2$.

$$\begin{aligned} Now,\ observe\ that\ L(f(R_1 + R_2)) &= L(f(R_1) + f(R_2)) \\ &= L(f(R_1)) \, U \, L(f(R_2)) \\ &= f(L(R_1)) \, U \, f(L(R_2)) \\ &= L(f(R_1)) \, U \, L(f(R_2)) \\ &= f(L(R_1) \, U \, L(R_2)) \\ &= f(L(R_1 + R_2)) \end{aligned}$$

Case 2: $R = R_1 R_2$.

Now, observe that
$$\begin{aligned} L(f(R_1 R_2)) &= L(f(R_1))L(f(R_2)) \\ &= f(L(R_1))f(L(R_2)) \\ &= f(L(R_1)L(R_2)) \\ &= f(L(R_1 R_2)) \end{aligned}$$

Case 3. $R = R_1{}^*$

Now, observa that
$$\begin{aligned} L(f(R_1{}^*)) &= L(f(R_1)^*) \\ &= L(f(R_1))^* \\ &= f(L(R_1))^* \\ &= f(L(R_1)^*) \\ &= f(L(R^*{}_1)) \end{aligned}$$

Section 2

Question 1

a) The state diagram

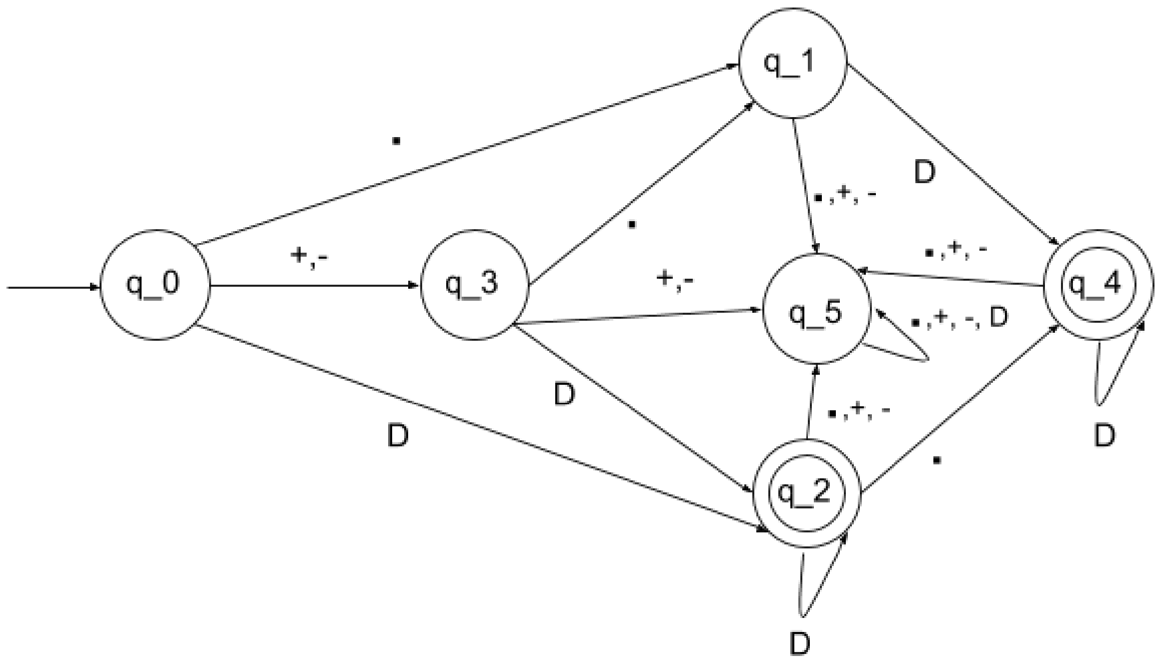

Figure 1. State diagram of M_b1 which recognizes the set of real and integer numbers i.e. all numbers are either integers or reals with zero or more decimal numbers before a the decimal point that can be followed by zero or more decimal digits.

b) Formal mathematical notation

The formal description of the NFA is (Q, Σ, δ, $q_1, F\}$ where

Q = $\{q_0, q_1, q_2, q_3, q_4\}$
$\Sigma = \{+,-, D = \{0,1,2,3,4,5,6,7,8,9\}, .\}$
δ :

	.	D	+	-
q_0	q_1	q_2	q_3	q_3
q_1	q_5	q_4	q_5	q_5
q_2	q_4	q_2	q_5	q_5
q_3	q_1	q_2	q_5	q_5
q_4	q_5	q_4	q_5	q_5

q_5	q_5	q_5	q_5	q_5

$q_1 = q_0$

$F = \{q_2, q_4\}$

Question 2

a) Let $\alpha = \{A \cup a \cup B \cup b \cup ...Z \cup z\}$ and let $\beta = \{< \cup > \cup = \cup >= \cup <=\}$

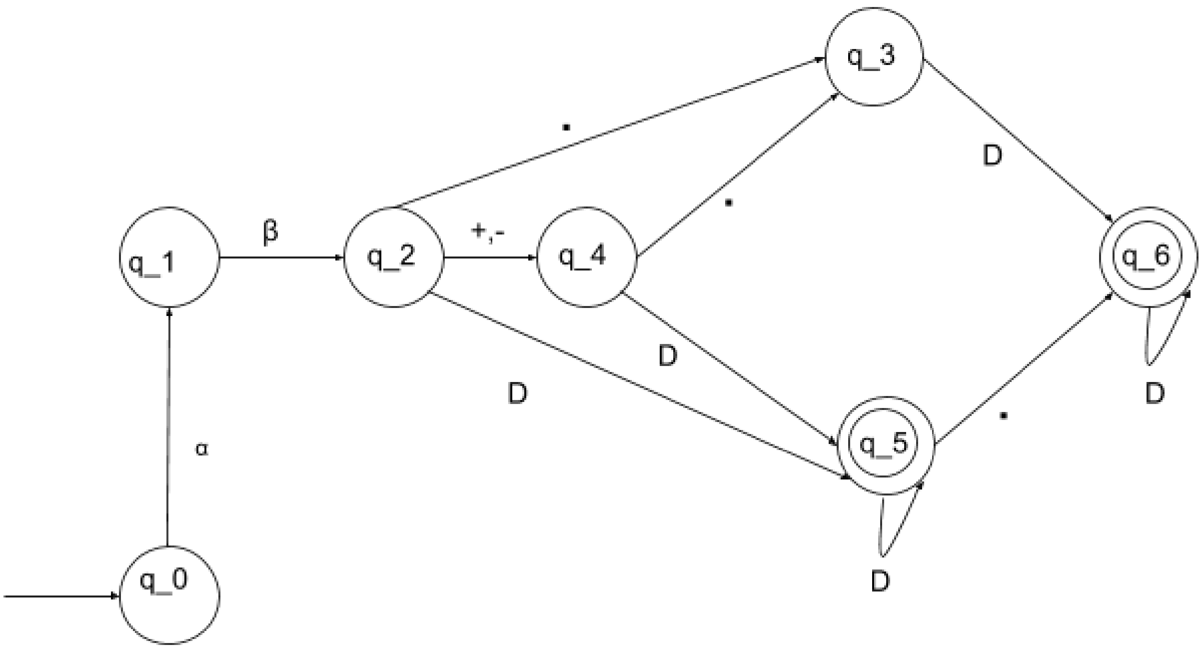

Figure 2. State diagram of M_b2 which recognizes strings of the form <variable>ε<condition>ε<token>ε

b) Let $\alpha = \{A \cup a \cup B \cup b \cup ...Z \cup z\}$ and let $\beta = \{< \cup > \cup = \cup >= \cup <=\}$

c) Formal mathematical notation

The formal description of the NFA is (Q, Σ, δ, $q_1, F\}$ where

Q = $\{q_0, q_1, q_2, q_3, q_4, q_{5,} q_6\}$
$\Sigma = \{+,-, D = \{0,1,2,3,4,5,6,7,8,9\}, ., \alpha, \beta\}$
δ :

	.	D	+	-	α	β
q_0	∅	∅	∅	∅	q_1	∅
q_1	∅	∅	∅	∅	∅	q_2
q_2	q_3	∅	q_4	q_4	∅	∅
q_3	∅	∅	∅	∅	∅	∅
q_4	q_3	∅	∅	∅	∅	∅
q_5	q_6	∅	∅	∅	∅	∅
q_6	∅	q_6	∅	∅	∅	∅

$q_1 = q_0$

$F = \{q_5, q_6\}$

Question 3

a) The state diagram[2]

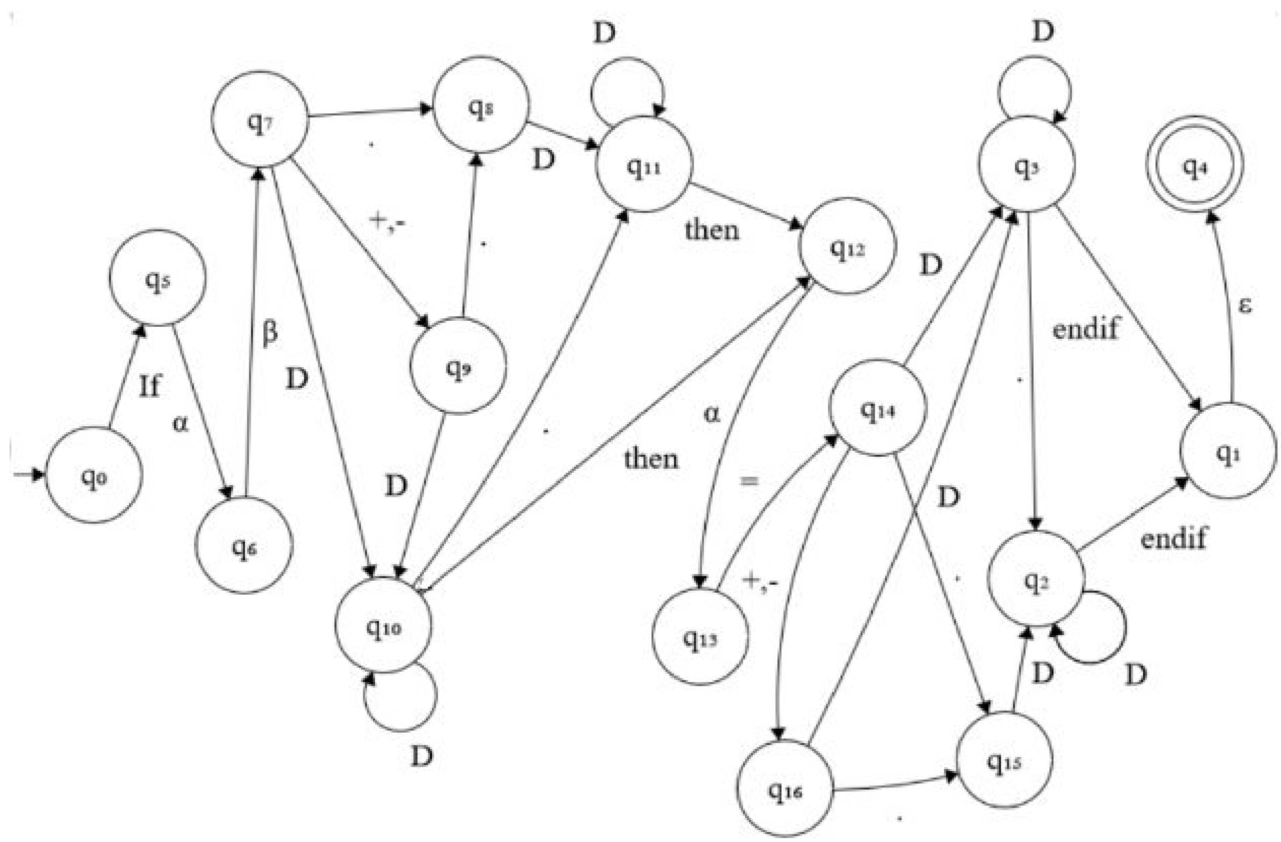

Figure 3. State diagram of M_B which recognizes any string of the form If ε

---

[2] This problem demonstrated **#breakitdown** since the parts of finite automata was made iteratively with the answers of the previous questions and I was able to accurately put together those tractable components to build and simulate the final solution.

<condition_expression> ε then ε <expression> ε endif ε

b) Regular expression

If $(\alpha\ \ \beta\ \ \ (+\cup-\cup\varepsilon)(D^{+}\cup D^{+}.\,D^{*}\cup D^{*}.D^{+}))$ $then\ \alpha\ =\ (+\cup-\cup\varepsilon)(D^{+}\cup D^{+}.\,D^{*}\cup D^{*}.D^{+})\ endif\ \varepsilon$

c) Formal mathematical notation for M_B

The formal description of the NFA is (Q, Σ, δ, $q_1, F\}$ where

Q = $\{q_0, q_1, q_2, q_3, q_4, q_{5,} q_6, q_7, q_8, q_9, q_{10}, q_{11},\ q_{12},\ q_{13},\ q_{14},\ q_{15},\ q_{16}\}$

$\Sigma =\ \{+,-,\ D = \{0,1,2,3,4,5,6,7,8,9\},\ .\ ,\ \alpha,\ \beta,\ \ =,\ if,\ then,\ endif\ ,\ \varepsilon\}$

δ :

	.	**D**	+	-	α	β	=	if	then	endif
q_0	∅	∅	∅	∅	∅	∅	∅	q_5	∅	∅
q_1	∅	∅	∅	∅	∅	∅	∅	∅	∅	∅
q_2	∅	q_2	∅	∅	∅	∅	∅	∅	∅	q_1
q_3	q_2	q_3	∅	∅	∅	∅	∅	∅	∅	q_1
q_4	ε	ε	ε	ε	ε	ε	∅	ε	ε	ε
q_5	∅	∅	∅	∅	q_6	∅	∅	∅	∅	∅
q_6	∅	∅	∅	∅	∅	q_7	∅	∅	∅	∅
q_7	q_8	q_{10}	q_9	q_9	∅	∅	∅	∅	∅	∅
q_8	∅	q_{11}	∅	∅	∅	∅	∅	∅	∅	∅
q_9	q_8	q_{10}	∅	∅	∅	∅	∅	∅	∅	∅
q_{10}	q_{11}	q_{10}	∅	∅	∅	∅	∅	∅	q_{12}	∅
q_{11}	∅	q_{11}	∅	∅	∅	∅	∅	∅	q_{12}	∅
q_{12}	∅	∅	∅	∅	q_{13}	∅	∅	∅	∅	∅
q_{13}	∅	∅	∅	∅	∅	∅	q_{14}	∅	∅	∅

q_{14}	q_{15}	q_3	q_{16}	q_{16}	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
q_{15}	$\varnothing$	q_2	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$
q_{16}	q_{15}	q_3	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$	$\varnothing$

$q_1 = q_0$

$F = \{q_4\}$

L(M_B) = {w | w is of the form: If <variable> <condition> <token> then <variable> = <token> endif, where $< variable >= \{A \cup a \cup B \cup b \cup ...Z \cup z\}$ and $< condition > = \{< \cup > \cup = \cup >= \cup <=\}$ and $< token > = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}\}$

Question 4

A. https://colab.research.google.com/drive/1W5XtXEx6cobULsQzIznkxR7LO7KSTIsH
B. My Python M_B $Simulator$ defines a separate function for each state in Figure 3. The string is passed as a list with each element as a substring. Each state transitions from one to the other if the string at the beginning of the the list corresponds to a transition, and the state passes the remaining elements to the next state.

   There is a single accept state q_4. If the string ends before we reach q_3 (which has an epsilon transition to q_4), the simulator prints a rejection message and returns nothing. Otherwise, if at each state, we don't receive the symbol that corresponds to a transition, the simulator also rejects the string. In addition, if the list is not empty by the time it reaches q_3 i.e. if there are extra symbols after we reach the accept state, the string is rejected.

Question 5

a) CFG using the formal notation

To avoid confusion between the variable D and the symbol for D = $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$, we will recode the latter as $\gamma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

As before, let $\alpha = \{A \cup a \cup B \cup b \cup ...Z \cup z\}$ and let .

Therefore, the formal definition of our CFG is:

V = {A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,S}
$\Sigma = \{+, -, \gamma\ .\ , \alpha, \beta, =, if, then, endif, \varepsilon\}$

The set of rules, R, is

S $\rightarrow$ IfE
A $\rightarrow$ D
B $\rightarrow$ γ B | endifA
C $\rightarrow$ .B | endifA | γ C
D $\rightarrow$ ε
E $\rightarrow$ α F
F $\rightarrow$ βG
G $\rightarrow$ .H | γ J | +I | -I
H $\rightarrow$ γ K
I $\rightarrow$ γ J | .H
J $\rightarrow$ thenL | γ J | .K |
K $\rightarrow$ thenL | γ K |
L $\rightarrow$ α M
M $\rightarrow$ =N
N $\rightarrow$ .O | -P | +P | γ C
O $\rightarrow$ γ B
P $\rightarrow$ γ C | .O

S = S

b) The process for converting a finite automaton to a CFG is is algorithmic: we create a symbol in the grammar for every state in the NFA M_B and a rule for every transition. Particularly, a state q_i corresponds to a symbol L_i (and the language generated by L_i is the suffix language of the state q_i) . A transition $q_i \in \delta(q_h, x)\, for\ x \in \Sigma$ will be translated into the rule

$$L_h \rightarrow xL_i$$

For any accepting state q_h , we also add the rule $L_h \rightarrow \varepsilon$. In some cases, x can be ε. Then the $\varepsilon - transition$ of moving from q_h to q_i is converted into the rule $L_h \rightarrow L_i$).

This method works because if a language is regular, there is a finite automaton which recognizes the language and a CFG which generates the language.

c) Only regular languages can be converted to a finite automaton, and not all CFGs describe regular languages.

As an example, consider the language L = $\{0^n 1^n \;:\; n \geq 1\}$.

We can use the Pumping Lemma to prove that L is not regular.

Assume that L is regular. Let p be the pumping length for L. We will select a string $w \in L$ of length at least p, say, w = $0^p 1^p$ where w = xyz, with |y| > 0 and |xy| $\leq p$

There are three possibilities[3] for our pumpable blocks, note that the red block is y and the blue block before y is x and the blue block after y is z:

w = 000000000..0111111111...1
w = 000000000...0111111111...1
w = 000000000...0111111111...1

The last two cases disqualify because the pumping lemma requires |xy| $\leq$ p. In the first case, pumping on y gives a string not in language L because the number if 0's exceeds the number of 1's.

However, we can define a CFG: {{S}, {0,1}, R, {S}} where the set of rules is as follows:

S $\rightarrow$ 01
S $\rightarrow$ 0S1

[3] **#scienceoflearning:** In this problem, I demonstrated my command over #scienceoflearning by drawing on multiple sections of our curriculum and using them accurately: regular languages, pumping lemma and CFGs. I included an appropriate example to explain how a non-regular language is recognized by CFG and took advantage of spaced repetition to study effectively. In addition, I used color coding to distinguish the pumpable blocks and emphasize why 0^p1^p cannot be pumped.

References

Sipser, M. (2013). *Introduction to the theory of computation*. Boston, MA: Cengage Learning.