ZQMS-ARC-REC-002

ZIMBABWE OPEN UNIVERSITY

"Empowerment Through Open Learning"

ASSIGNMENT COVER

REGION: HARARE SEMESTER: 2 YEAR: 3

PROGRAMME: BACHELORS OF SOFTWARE ENGINEERING (HONS) INTAKE: 02

FULL NAME OF STUDENT: TRACEY K. NDORO PIN: P1669806Y

EMAIL ADDRESS: tkndoro@gmail.com

CONTACT TELEPHONE/CELL: 0772498056 ID. NO.: 07-107291-Q-07

COURSE NAME: ELEMENTS OF COMPUTER SCIENCE THEORY COURSE CODE: BITH 311

ASSIGNMENT NO. e.g. 1 or 2: 1 STUDENT'S SIGNATURE T. K. NDORO

DUE DATE: ____________ SUBMISSION DATE: ____________

ASSIGNMENT TITLE: ELEMENTS OF COMPUTER SCIENCE THEORY – BITH 311

Instructions
Marks will be awarded for good presentation and thoroughness in your approach.
NO marks will be awarded for the entire assignment if any part of it is found to be copied directly from printed materials or from another student.
Complete this cover and attach it to your assignment. Insert your scanned signature.

Student declaration
I declare that:

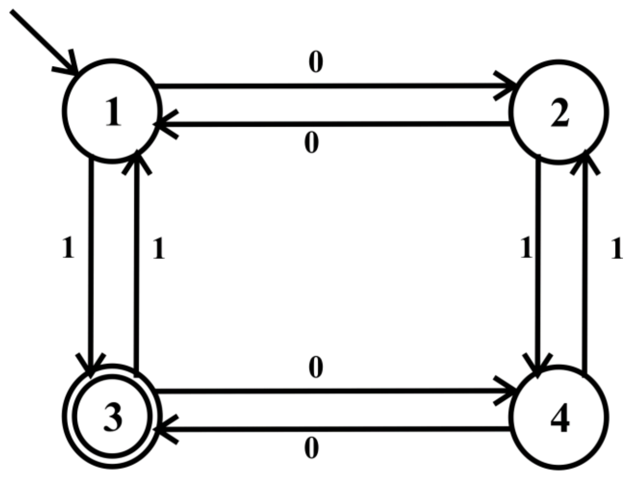
- *I understand what is meant by plagiarism*
- *The implications of plagiarism have been explained to me by the institution*
- *This assignment is all my own work and I have acknowledged any use of the published or unpublished works of other people.*

MARKER'S COMMENTS: ____________

OVERALL MARK: ____________ MARKER'S NAME: ____________
MARKER'S SIGNATURE: ____________ DATE: ____________

BITH 311 ELEMENTS OF COMPUTER SCIENCE THEORY - ASSIGNMENT 1

a) Construct a finite automata that accepts even number of zeros and odd number of ones [6]



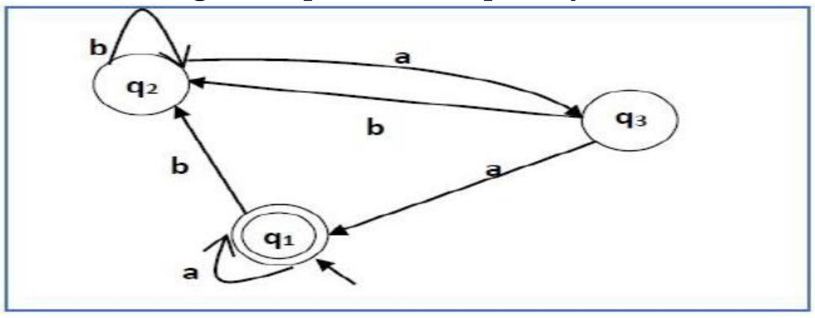
The above finite automata is the required one and the states are as below:

State 1:	even number of 0's and even number of 1's
State 2:	odd number of 0's and even number of 1's
State 3:	even number of 0's and odd number of 1's
State 4:	odd number of 0's and odd number of 1's

b) State Arden's theorem [3]

The Arden's Theorem states that:
"If P and Q are two regular expressions over and if P does not contain, then the following equation in R given by R = Q+RP has a unique solution i.e., R = QP*."

c) Construct a regular expression accepted by the automata below [7]



The regular expression accepted by the automata is as follows:

The automata's initial state and final state id q1.

Equations for the three states q1, q2 and q3 are:

q1= q1a + q3a+ε (ε move is because q1 is the initial state)

q2= q1b + q2b + q3b

q3= q2a

Now, solving these three equations: -
q2= q1b + q2b + q3b
= q1b + q2b + (q2a)b (Substituting value of q3)
= q1b + q2(b+ab)
= q1b + (b+ab)* (Applying Arden’s Theorem)

q1 = q1a + q3a + ε
= q1a + q2aa + ε (Substituting value of q3)
= q1a + q1b(b + ab*)aa + ε (Substituting value of q2)
= q1(a + b(b + ab)*aa) + ε
= ε (a + b(b + ab)*aa)*

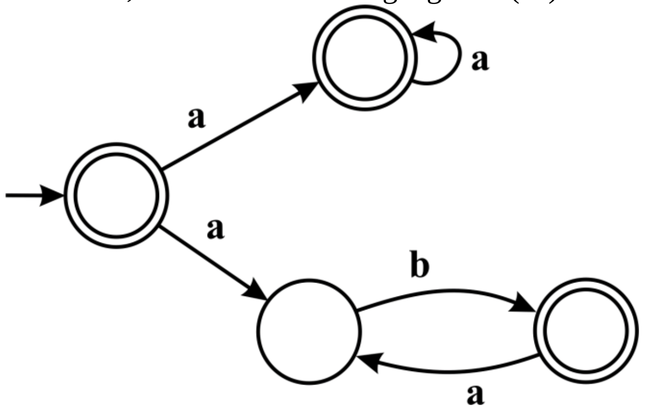
= (a + b(b + ab)*aa)*

Therefore, the Regular expression is (a + b(b + ab)*aa)*

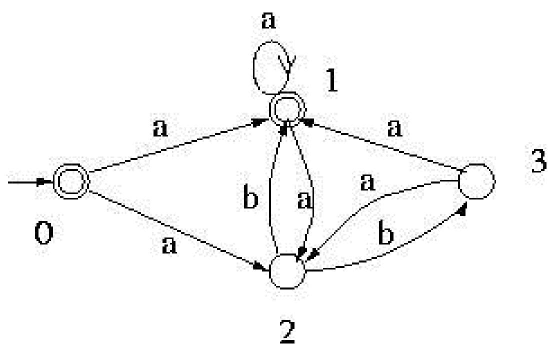
d) Construct an NFA for the language a*+ (ab)* [4]

The Nondeterministic finite automaton (NFA), for each state there can be zero, one, two, or more transitions corresponding to a particular symbol. If NFA gets to state with more than one possible transition corresponding to the input symbol, we say it branches. If NFA gets to a state where there is no valid transition, then that branch dies.

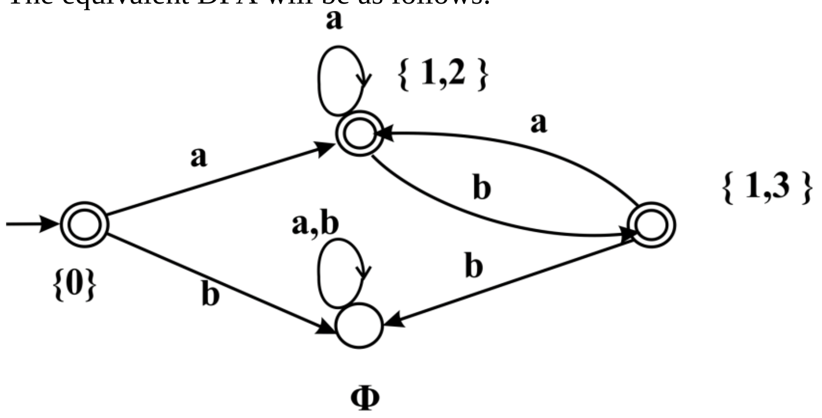
Therefore, the NFA for the language a*+(ab)* is as below:



e) Convert the following NFA to its equivalent DFA [7]



The equivalent DFA will be as follows:



Since states 0 and 1 are the accepting states of the NFA, all the states that contain 0 and/1 are accepting states. Hence {0}, {1,2} and {1,3} are accepting states.

f) State the pumping lemma for regular languages [3]

The theory of pumping lemma means that if a string s is 'pumped', i.e., if s is inserted any number of times, the resultant string still remains in L. Pumping Lemma is used as a proof for irregularity of a language.

The theory is often used to prove that a particular language is non-regular: a proof by contradiction (of the language's regularity) may consist of exhibiting a word (of required length) in the language that lacks the property outlined in the pumping lemma.

g) Use the pumping lemma to show that L ={ $a_nb_nc_n$: n ≥ 0} is not regular [7]

Proving that L= {$a_nb_nc_n$: n $\geq$ 0} is not regular:

Proof: Suppose L $a_nb_nc_n$ is context-free. Let p be the pumping length.

- Consider $z = a^pb^pc^p \in$ L $a_nb_nc_n$
- Since $|z| > p$, there are u,v,w,x,y such that $z = uvwxy$, $|vwx| \leq p$, $|vx| > 0$ and $uv^iwx^iy \in$ L for all $i \geq 0$.
- Since $|vwx| < p$, vwx cannot contain all three of the symbols *a, b, c,* because there are *p bs.*

  So *vwx* either does not have any *as* or does not have any *bs* or does not have any cs. Suppose, (wlong) *vwx* does have any *as*. Then uv^0wx^0y = uwy contains more *as* than either *bs* or *cs*.

  Hence *uwy* $\in \neq$ L.

h) Define a context free grammar (CFG) [3]

A context-free grammar (CFG) is a set of recursive rewriting rules or productions used to generate patterns of strings. A context-free grammar consists of the following components:

- a set of terminal symbols, which are the characters of the alphabet that appear in the strings generated by the grammar.

i) Define Chomsky normal form [2]

The Chomsky normal form is a particular form of writing a Context-free grammar (CFG) which is useful for understanding the CFGs and for proving things about them. Chomsky normal form also makes the parse tree for derivations using this form of the CFG a binary tree.

j) Convert the following CFG into its equivalent CFG in Chomsky normal form [10]

S → ASB
A → aAS |a |ε
B → SbS |A |bb

In converting the above CFG into its equivalent CFG Chomsky normal form, below are the steps of conversion:

STEP 1

Firstly, as start symbol S appears on the RHS, we will create a new production rule $S_0 \rightarrow S$ and the grammar will become:

$S_0 \rightarrow S$
S → ASB
A → aAS | a | ε
B → SbS | A | bb

STEP 2
As the grammar contains null production A → ε, its removal from the grammar yields:

S_0 → S
S → ASB | SB
A → aAS | aS| a
B → SbS | A | ε | bb

Now, it creates null production B→ε, its removal from the grammar yields:

S_0 → S
S → AS| ASB | SB | S
A → aAS | aS | a
B → SbS | A | bb

Now it creates unit production B→A, its removal from the grammar yields:

S_0 → S
S → AS| ASB | SB | S
A → aAS | aS | a
B → SbS | bb | aAS | aS | a

Now, also removal of unit production S_0 → S from grammar yields:

S_0 → AS | ASB | SB | S
S → AS | ASB | SB | S
A → aAS | aS | a
B → SbS | bb | aAS | aS | a

Then, removal of unit production S →S and S_0 →S from grammar yields:

S_0 → AS | ASB | SB
S → AS | ASB | SB
A → aAS | aS | a
B → SbS | bb | aAS | aS | a

STEP 3
The production rule A→ aAS | aS and B → SbS|aAS| aS, terminals a and b exist on RHS with non-terminates. Now, removing them from RHS:

S_0 → AS | ASB | SB
S → AS | ASB | SB
A → XAS | XS | a
B → SYS | bb | XAS | XS | a
X → a
Y → b

Now, $B \rightarrow bb$ can't be part of CNF, removing it from grammar yields:

$S_0 \rightarrow AS \mid ASB \mid SB$
$S \rightarrow AS \mid ASB \mid SB$
$A \rightarrow XAS \mid XS \mid a$
$B \rightarrow SYS \mid VV \mid XAS \mid XS \mid a$
$X \rightarrow a$
$Y \rightarrow b$
$V \rightarrow b$

STEP 4
The production rule $S_0 \rightarrow ASB$, RHS has more than two symbols, removing it from grammar yields:

$S_0 \rightarrow AS \mid PB \mid SB$
$S \rightarrow AS \mid ASB \mid SB$
$A \rightarrow XAS \mid XS \mid a$
$B \rightarrow SYS \mid VV \mid XAS \mid XS \mid a$
$X \rightarrow a$
$Y \rightarrow b$
$V \rightarrow b$
$P \rightarrow AS$

Again, $S \rightarrow ASB$ has more than two symbols, removing it from grammar yields:

$S_0 \rightarrow AS \mid PB \mid SB$
$S \rightarrow AS \mid QB \mid SB$
$A \rightarrow XAS \mid XS \mid a$
$B \rightarrow SYS \mid VV \mid XAS \mid XS \mid a$
$X \rightarrow a$
$Y \rightarrow b$
$V \rightarrow b$
$P \rightarrow AS$
$Q \rightarrow AS$

From above, again $A \rightarrow XAS$ has more than two symbols, removing it from grammar yields:

$S_0 \rightarrow AS \mid PB \mid SB$
$S \rightarrow AS \mid QB \mid SB$
$A \rightarrow RS \mid XS \mid a$
$B \rightarrow SYS \mid VV \mid XAS \mid XS \mid a$
$X \rightarrow a$
$Y \rightarrow b$
$V \rightarrow b$
$P \rightarrow AS$
$Q \rightarrow AS$
$R \rightarrow XA$

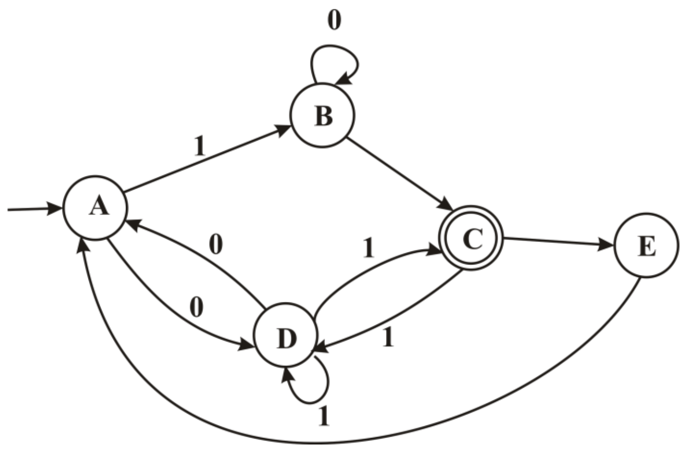
Again, B → SYS has more than two symbols, removing it from grammar will yields:

S_0 → AS | PB | SB
S → AS | QB | SB
A → RS | XS | a
B → TS | VV | XAS | XS | a
X → a
Y → b
V → b
P → AS
Q → AS
R→ XA
T → SY

The same, B →XAX has more than two symbols, removing it from grammar will yields:

S_0 → AS | PB | SB
S → AS | QB | SB
A → RS | XS | a
B → TS | VV | US | XS | a
X → a
Y → b
V → b
P → AS
Q → AS
R→ XA
T → SY
U → XA
Hence, this is the required Chomsky normal form (CNF) for the given grammar.

k) Construct a DFA for the regular expression (10* + (00 + 11*)* [5]



l) Define lexical analysis [3]
The lexical analysis is the first phase of a compiler also known as scanner. It takes the modified source code from language processors that are written in the form of sentences. Then the lexical analyzer breaks these syntaxes into a series of tokens, by removing any whitespace or comments in the source code.

If the process the lexical analyzer finds a token invalid, it generates an error. The lexical analyzer works closely with the syntax analyzer. It reads character streams from the source code, checks for legal tokens, and passes the data to the syntax analyzer when it demands.

REFERENCES:

BITH 311 Elements of Computer Theory MyVista Notes

https://www.courses.cs.washington.edu/courses/cse322/08au/lec14.pdf

https://www.cs.odu.edu/~toida/nerzic/390teched/regular/fa/nfa-2-dfa-html

https://www.geeksforgeeks.org/converting -context-free-grammar-chomsky-normal-form/

https://www.people.cs.clemson.edu/~goddard/texts/theoryOfComputation/3a.pdf

https://www.tutorialspoint.com/automata_theory/ardens_theorem.htm

https://www.youtube.com/watch?v=eOfMcdeyrMU