Department of Computer Science and Engineering
Jahangirnagar University, Savar, Dhaka, Bangladesh

OBTAINED MARK

26

TUTORIAL ANSWER SCRIPT

Student's ID No: 353 Name: Shanjida Alam

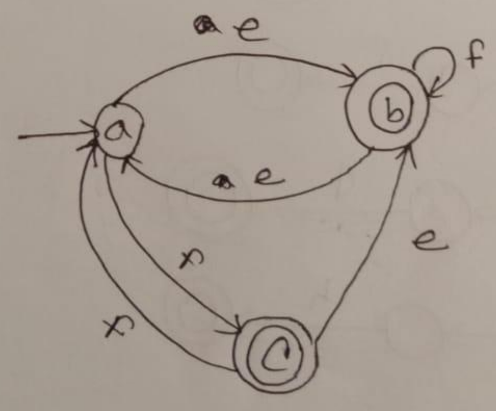
Course Code: CSE-401 Course Title: Theory of Computertion and Compiler Design

Tutorial Examination No: 02 Date: 02/10/24 Signature of Course Teacher:

1) Prove that, if a language is regular then it is describes by a regular ~~language~~ expression.

2) i) Convert the regular expression $(ab \cup cd)^* efgh$ to NFA.

ii) Convert the following 3-state DFA to ~~ar~~ an equivalent regular expression



Ans to the Q. no. → 2(i)

The given regular expression is,

(abucd)* efgh.

Now converting regular expression to NFA,

a

b

c

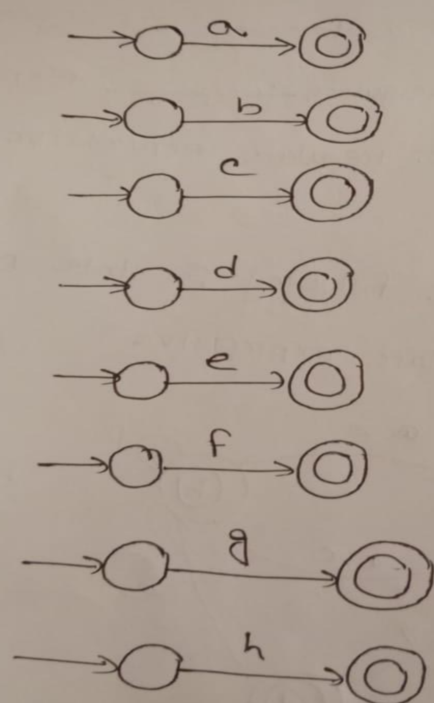
d

e

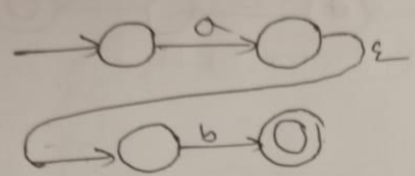
f

g

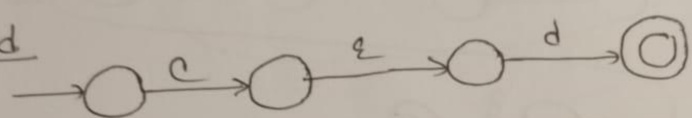
h



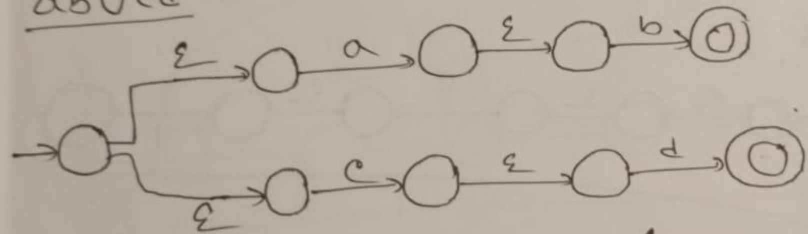
ab



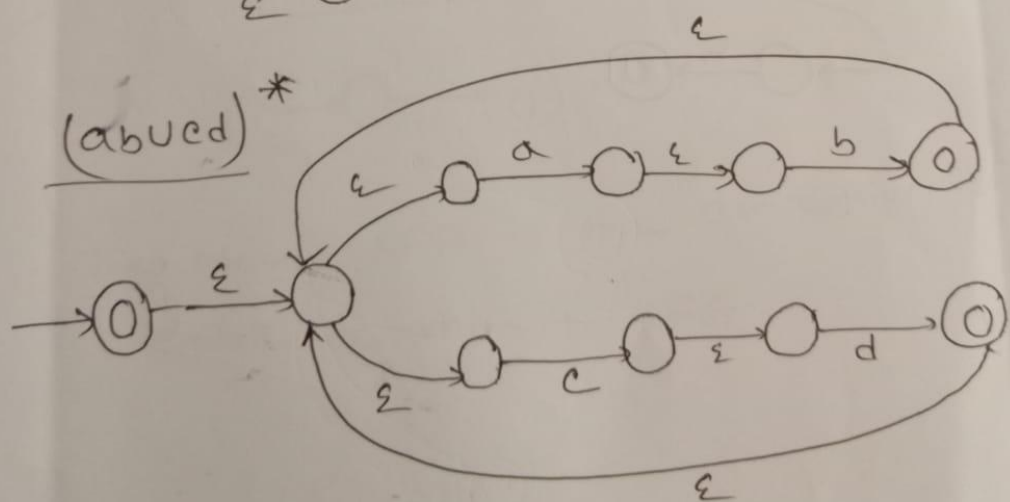
cd



ab∪cd

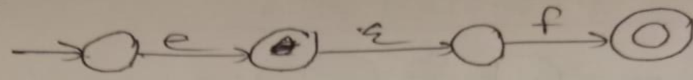


(ab∪cd)*

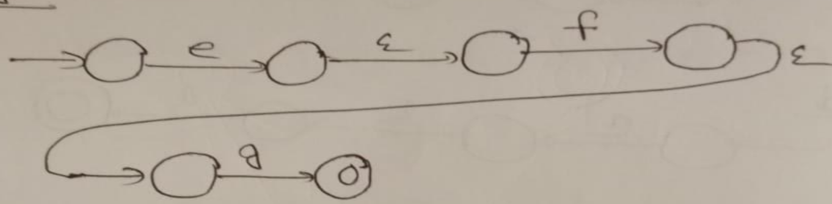


P.T.O

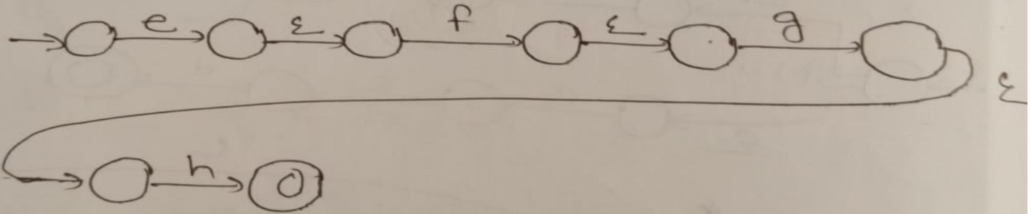
ef



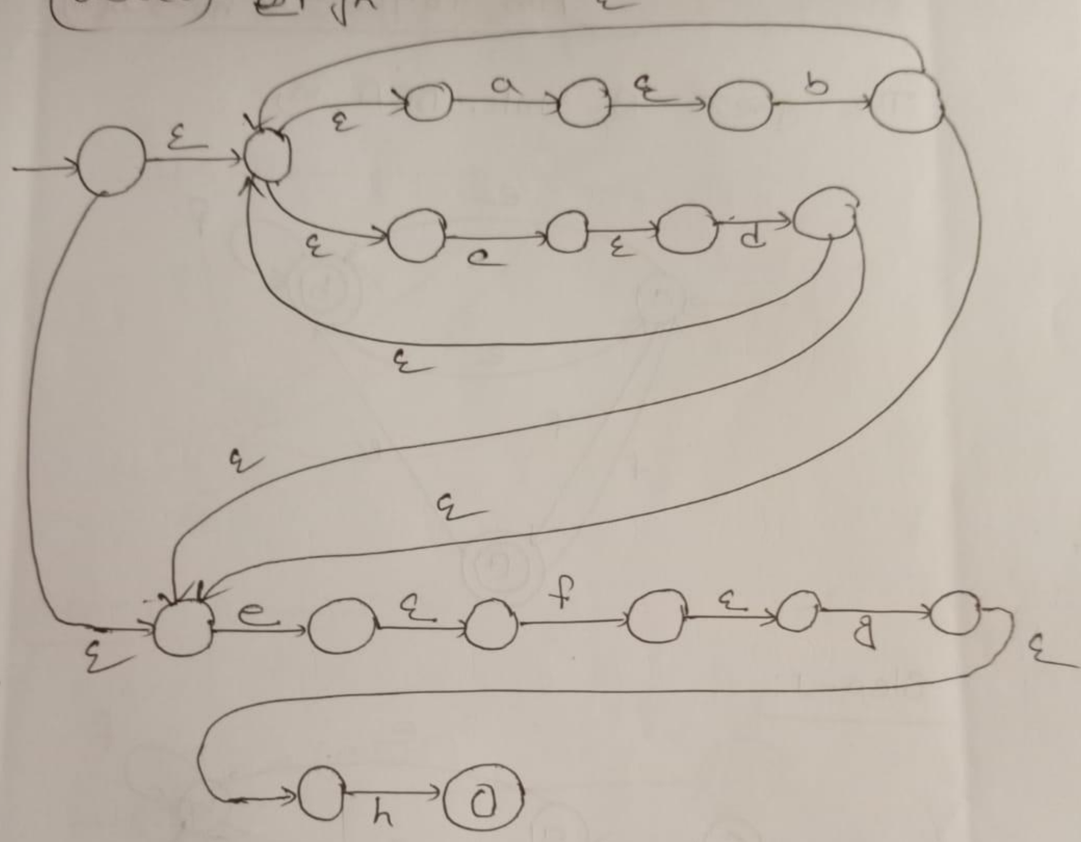
efg



efgh



(abucd)* efgh

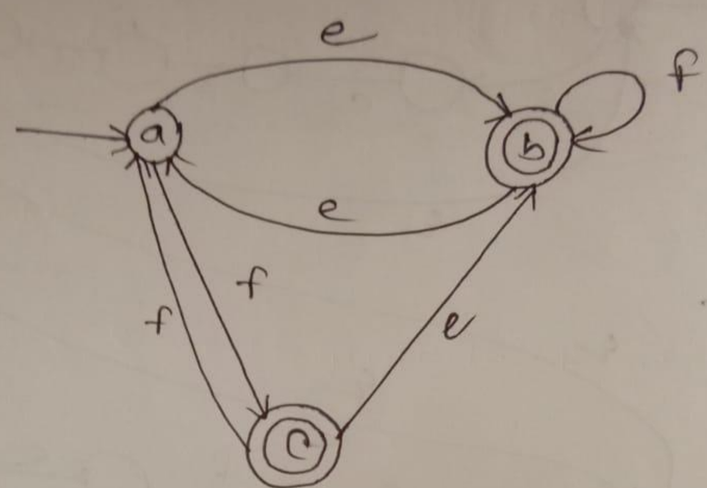


The above NFA is the conversion of

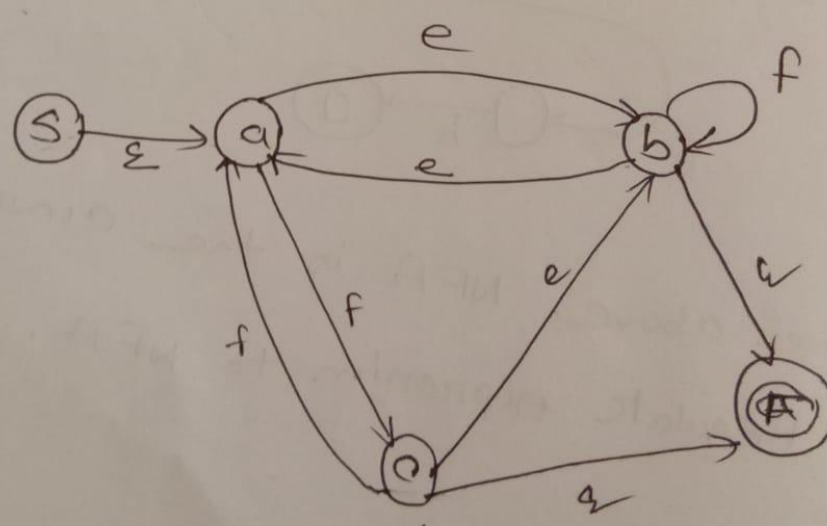
Regular expression to NFA.

Ans to the q. no→2(H)

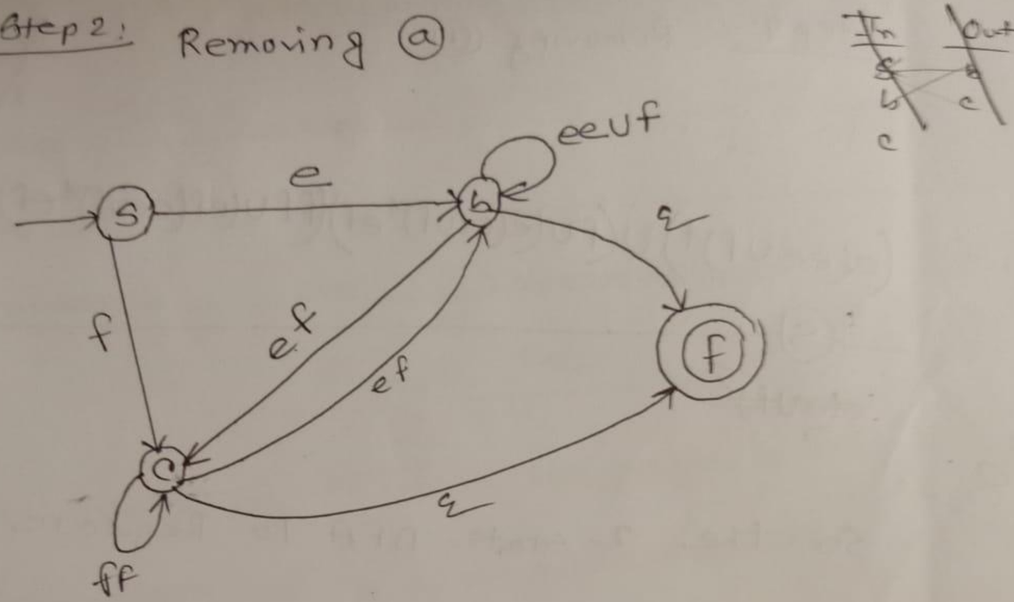
The given 3-state DFA is,



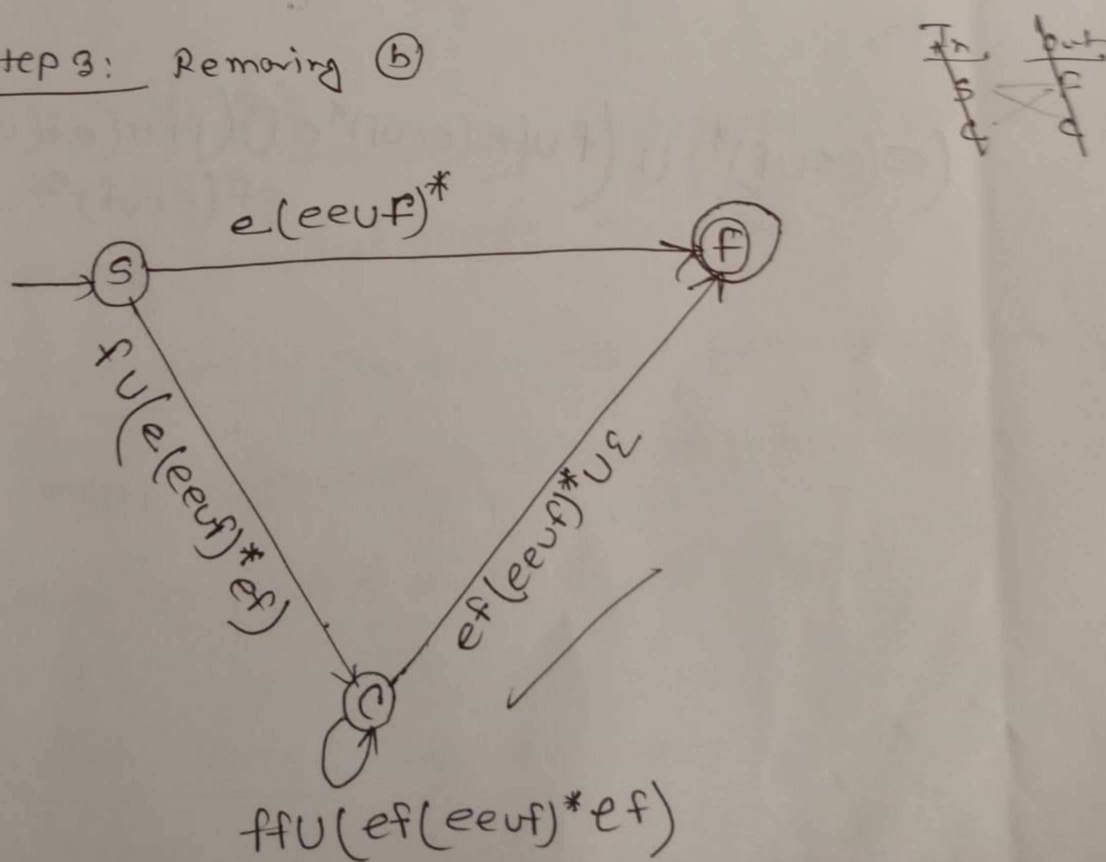
Step-1:



Step 2: Removing ⓐ



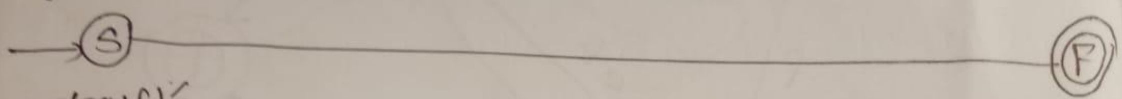
Step 3: Removing ⓑ



Step 4: Removing ⓒ

~~In~~ ~~Out~~
S F

$$(e(ee \cup f)^*) \cup (f \cup (e(ee\cup f)^* ef)(ff \cup (ef(ee\cup f)^* ef)^* ef(ee\cup f)^*)$$



~~e(eeuf)*~~

8/

So the 3-state DFA to Regulate expression is,

$$(e(ee\cup f)^*) \cup (f \cup (e(ee\cup f)^* ef)(ff \cup (ef(ee\cup f)^* ef)^* ef(ee\cup f)^* \cup \varepsilon)$$

Department of Computer Science and Engineering
Jahangirnagar University, Savar, Dhaka, Bangladesh

OBTAINED MARK

TUTORIAL ANSWER SCRIPT

Student's ID No: 358 Name: Shanjida Alam

Course Code: Course Title:

Tutorial Examination No: Date: Signature of Course Teacher:

Ans to the Q. no → 1

We show that; ~~if~~ if a language is regular then it is described by a regular expression.

Prove Idea:

The prove idea of the given theorem should be divided into two parts. They are:

First part: In the first part we basically trying to proves that a regular language must be recognized by any DFA/NFA.

Second part: In the second part we trying to prove that a regular language has an

P.T.O

~~equ~~ regular expression.

Proof of the first part: In this part the proof contains several steps. They are following

1) If any language a, $\Sigma = a$ then $L(R) = \{a\}$ and it recognizes by a NFA

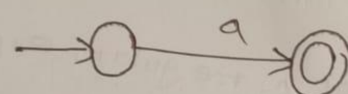


Fig: NFA 'N' recognizing the language $\{a\}$

2) If ~~occurs~~ any language ε, the $L(R) = \{\varepsilon\}$ and it recognizes by a NFA

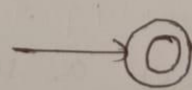


Fig: NFA 'N' recognizing the language $\{\varepsilon\}$

Formally, $N = (q_1, \Sigma, \delta, q_1, q_1)$.

3) If there is coming ~~no lang~~ empty string ϕ, then it recognizes by NFA

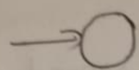


Fig: NFA 'N' recognizing ϕ.

Formally, $N = \{q_1, \Sigma, \delta, q_1, \phi\}$

4) $R_1 \cup R_2$ where, $L\{R_1\} \cup L\{R_2\}$

5) $R_1 \circ R_2$ where, ~~L~~ $L\{R_1\} \circ L\{R_2\}$

6) R_1^* where, $L\{R_1\}^*$

The last three cases are the regular opera-
tion for the regular expression and they are
accepted by the ~~Df~~ NFA that recognizes
them.

So, it is proved that, if a language A is
regulare then it must be recognized by a DFA.

Proof of the 2nd part! In this part we basically prove that how to convert the regular language to the regular expression.

Prove Idea for the second part! At first we confirm about that every regular language is recognized by any DFA. So, we have a DFA/NFA for the given regular language. Then convert the DFA to the generalised non-deterministic finite automata (GNFA). Then removing the intermediate state from the GNFA to calculate the regular expression from the GNFA.

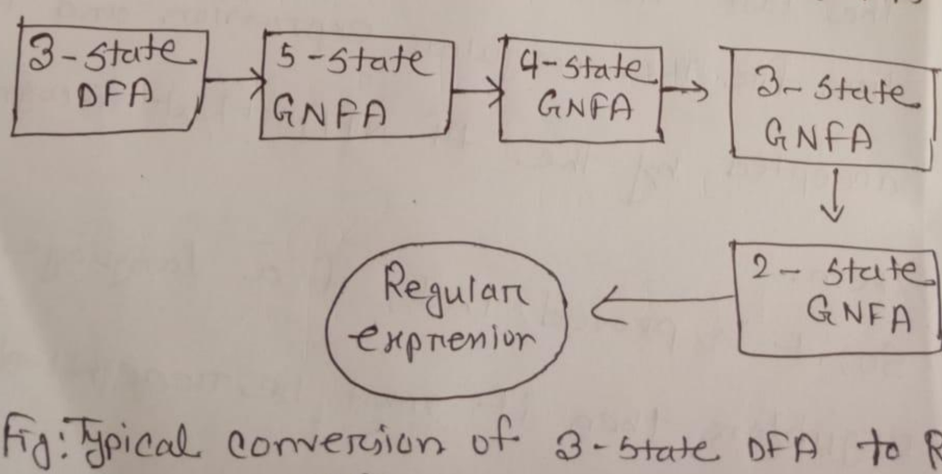


Fig: Typical conversion of 3-state DFA to Regular expression.

Proof:

Step 1: GNFA

From the given DFA it converts to the GNFA.

For this reason GNFA has some properties they are:

1) GNFA has only single start state that means there is no incoming ~~at~~ arrows ~~to~~ to the start state.

2) GNFA has only single ~~end~~/accept state that means there is no outgoing arrows ~~from~~ to the any other state.

3) The labeling of the start state to accept state is considered as regular expression.

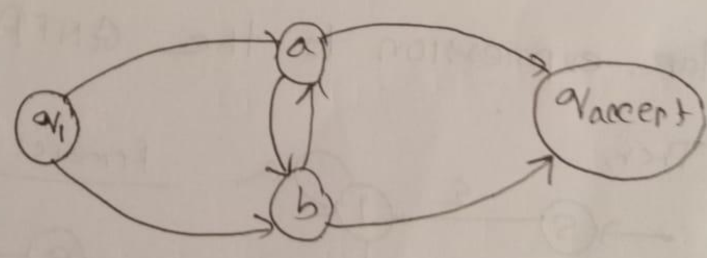


Fig: Example of GNFA.

Step 2: DFA to GNFA

If the given DFA is not maintained the propera of GNFA the modify the DFA for converting it GNFA

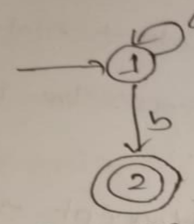


Fig: DFA

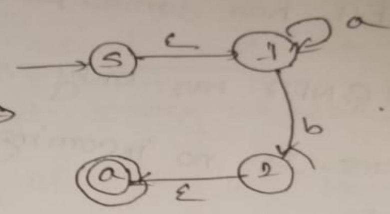
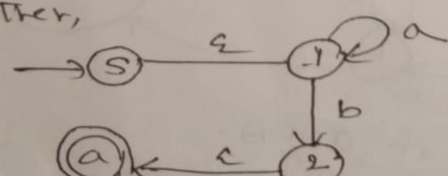


Fig: DFA to GNFA.

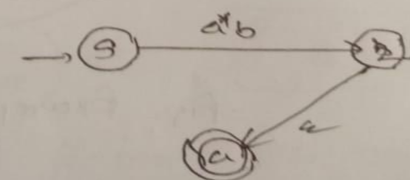
Step 3: GNFA to Regular expression

GNFA only contains the two state accept state and final/accept state. So, the intermediate state should be removed for calculating the regular expression to the GNFA.

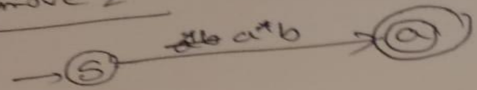
Then,



Remove: 1



Remove 2



The Regular expression is a^*b.

So, it is proved that, if a language is regula
then it is described by a regular expression.

[Proved]

—o—

19

Department of Computer Science and Engineering
Jahangirnagar University, Savar, Dhaka, Bangladesh

OBTAINED MARK

TUTORIAL ANSWER SCRIPT

Student's ID No: 353 **Name:** Shanjida Alam

Course Code: CSE-401 **Course Title:** Theory of computation

Tutorial Examination No: 01 **Date:** 11/09/24 **Signature of Course Teacher:**

Q₁ : Define regular operations.

Prove that the class of regular language is closed under union operation. (DFA)

Q₂ : What do you mean by nondeterminism?

Prove that ~~any~~ every nondeterministic finite automaton (NFA) has an equivalent DFA.

Ans to the Q. no→1

Regular operations: Let A and B be are two languages. The regular operations are Union operation, Concatenation operation and Star operation.

Union operation: $A \cup B = \{(x, y) \mid x \in A \text{ or } y \in B\}$

Concatenation operation:

$$A \circ B = \{(x, y) \mid x \in A \text{ and } y \in B\}$$

Star operation: It is an unary operation. Only works with one language.

$$A^* = \{\varepsilon, x_1, x_2 \ldots x_n\} \text{ where } x_n \in A$$

Theorem: The class of regular language is closed under union operation.

Prove Idea: Let A_1 and A_2 are two languages. A_1 is recognized by the M_1 and A_2 is recognized by the M_2. A_1 and A_2 are regular languages because they are correspondingly recognized by the M_1 and M_2. Now we can prove that, $A_1 \cup A_2$ also regular languages.

In this case, at first we construct the M to recognize that $A_1 \cup A_2$ is regular language. And we also consider that the alphabets are same for both machines M_1 and M_2. States of the machine M are the union of both M_1 and M_2.

Proof:

Let, $A_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ is recognized by M_1.

$A_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ is recognized by M_2.

Now constructing $M = (Q, \Sigma, \delta, q_0, F)$ for $A_1 \cup A_2$.

1) $\cancel{Q = Q_1 \cup Q_2}$ $Q = Q_1 \times Q_2$

The states of the states are the cartesian product of the states Q_1 and Q_2.

2) Σ denotes the alphabet. For the simplicity we assume that the alphabets are same for both machines M_1 and M_2. So, the alphabet is also same for the $A_1 \cup A_2$. If the different alphabets are coming it also be accepted by the machine M.

3) δ is denoted the transition function. In this case δ is described as, r is the state of a,

~~$\delta \in F$~~

$r \in Q$ and $a \in \Sigma$, so,

$$\delta\left((r_1, r_2), a\right) = \delta\left((r_1, a), (r_2, a)\right)$$

4) q_0 denotes the start state of the machine.

So, q_0 describes as, $q_0 = \{q_1, q_2\}$

12

5) F denotes the final state or accept state of the machine. F is described as,

$$F_0 = (F_1 \times Q_2) \cup (F_2 \times Q_1)$$

So, it is proved that the class of regular languag is closed und union operation.

— o —

Ans to the q.no → 2

Nondeterminism : Nondeterminism means that it has no fixed state for the next state transition.

A formal definition of nondeterminism is collection of 5-tuples $(Q, \Sigma, \delta, q_0, F)$ where,

1) Q is the set of all states

2) Σ is the alphabets.

3) $\delta : (Q \times \Sigma) \rightarrow Q_i$ is the transition function.

4) q_0 is the start state.

5) F is the final state or accept state of the machine.

Theorem: ~~Ato~~ Every nondeterministic finite automaton (NFA) has an equivalent DFA.

Prove idea: ~~E~~ Nondeterministic finite automaton means that it has no fixed transition state. It contains the ϵ state. DFA means Deterministic finite ~~A~~ Automata that means it has fixed transition state for accepting the alphabet.

Now we trying to prove that NFA to DFA that means every nondeterministic finite automaton has an equivalent DFA.

7/

Proof:

1) The set of state: At first defines the all states for the NFA to DFA. In this case the set of states are the power set of the given state.

P.T.O

That means,

$$Q = P(Q)$$

2) <u>Alphabets for the machine:</u> Alphabets are defined by the Σ. The alphabets are same for both automaton.

$$\Sigma \rightarrow (\Sigma_1, \Sigma_2)$$

3) <u>The start state!</u>

The start state of the NFA to DFA is,

$$q_0 = q_0$$

That means which states are containing the start state in power set they are counting as start state.

4) <u>Transition function:</u> Let, $a \in \Sigma$

$$\delta(q, a) = \bigcup_{a \in \Sigma_\epsilon} \delta(q_1, a) \cup \delta(q_2, a)$$

5) <u>Final state!</u> After removing the state who does not contain any incoming input. That states are final state.

So, it is proven that every NFA has an equivalent DFA.