1

22/05/21

END SEMEXAM - 4

FA & AT

JUNAID GIRKAR
60004190057
SE - COMPS A-3

Q1 a Differentiate NFA vs DFA

NFA	DFA
• It stands for Nondeterministic Finite Automata	• It stands for Deterministic Finite Automata.
• No need to specify how does the NFA react according to some symbol	• For each symbolic representation of the alphabet, there is only one state transition in DFA
• NFA can use empty string transition	• DFA cannot use empty String transition.
• NFA can be understood as multiple little machines computing at the same time	• DFA can be understood as one machine
• NFA rejects the string in the event of all branches dying or refusing the string.	• DFA rejects the string in case it terminates in a state that is different from the accepting state.
• Time needed for executing an input string is more	• Time needed for executing an input string is less.
• Not all NFA are DFA	• All DFA are NFA
• NFA requires less space than DFA	• DFA requires more state
• E.g: $\delta : Q \times \Sigma \rightarrow 2^Q$ i.e. nent possible state belongs to the power set of Q.	• E.g: $\delta : Q \times \Sigma \rightarrow Q$ i.e next possible state belongs to Q.

6000419005)

Q16

ANS STEP 1 : let m be the DFA

$\therefore m = (Q, \Sigma, \delta, q_0, F)$

where Q = Finite set of all states

Σ = Finite set of input symbols

δ = Transition function

q_0 = Initial state

F = Finite set of final states

STEP 2 : LOGIC

Input alphabets are $\{a, b, c\}$

$\therefore$ starting with a $\Rightarrow$ end with b or c

starting with b $\Rightarrow$ end with a or c

starting with c $\Rightarrow$ end with a or b

let q_0 = initial state

q_1 = string starting with 'a' and end with 'c'

q_2 = string starting with 'b' and end with 'b'

q_3 = string starting with 'c' and end with 'c'

q_4 = string starting with 'a' ends with 'b' or 'c'

q_5 = string starting with 'b' ends with 'a' or 'c'

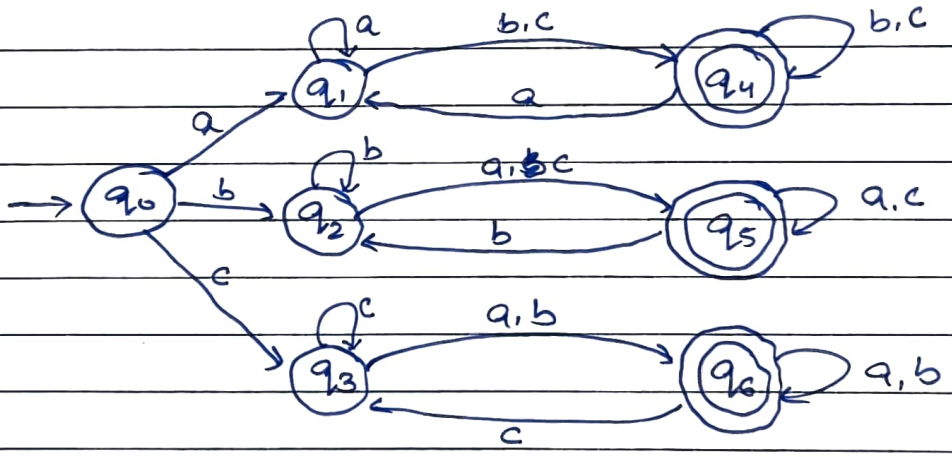
q_6 = string starting with 'c' ends with 'a' or 'b'

6000419005

STEP 3 : TRANSITION TABLE

δ/Σ	a	b	c
→ q_0	q_1	q_2	q_3
q_1	q_1	* q_4	* q_4
q_2	* q_5	q_2	q_5
q_3	q_6	q_6	q_3
* q_4	q_1	* q_4	q_4
* q_5	* q_5	q_2	* q_5
* q_6	q_6	* q_6	q_3

STEP 4 : TRANSITION DIAGRAM.



STEP 5 : SAMPLE INPUT abaa c

$\delta(q_0, abaac)$

$\delta(q_1, baac)$

$\delta(q_4, aac)$

$\delta(q_1, ac)$

$\delta(q_6, c)$

$q_4 \longrightarrow$ Final state accepted.

6000419005 7

Q2 a) Differentiate NPDA vs DPDA.

NPDA	DPDA
• It stands for Nondeterministic Pushdown Automata	• It stands for Deterministic Pushdown Automata
• central symbol is not known in case of NDPA	• In DPDA, the central symbol is known. E.g: abcba. Here c denotes the central symbol and tells after this symbol, pop operation needs to be performed.
• There are multiple moves possible in one situation	• There is only one move allowed in one situation.
• The representation of NDPA is ww^R	• The representation of DPDA is wcw^R
• Every NDPA cannot be simulated by DPDA	• Every DPDA can be simulated by an NDPA

B0004190057

Q6 a church's hypothesis.

ANS Turing machine is defined as an abstract representation of a computing device such as hardware in computers. Alan Turing proposed logical computing Machines (LCM's), i.e. Turing's expressions for Turing Machines. This was done to define algorithms properly. So, church made a mechanical method named as 'M' for manipulation of strings by using logic and mathematics.

The method M must pass the following statements:-

→ Number of instructions in M must be finite
→ output should be produced after performing finite number of steps.
→ It should not be imaginery i.e. can be made in real life
→ It should not require any complex understanding

using these statements church proposed a hypothesis called church's Turing thesis that can be stated as: "The assumption that the intuitive notion of computable functions can be identified with partial recursive functions"

In 1930, this statement was first formulated by Alonzo Church and is usually referred to as church's thesis or the church-Turing thesis. However, this hypothesis cannot be proved.

P.T.O

The recursive functions can be computable after taking the following assumptions :-

→ Each and every function must be computable

→ let 'F' be the computable function and after performing some elementary operations to 'F', it will transform a new function 'G' then this function 'G' automatically becomes the computable function

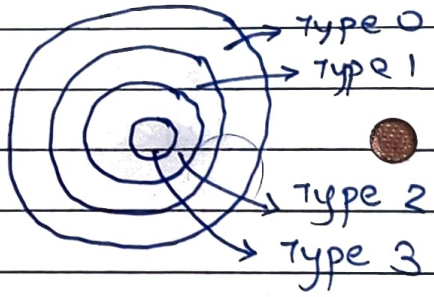
→ If any functions that follow above two assumptions must be stated as computable function.

b Chomsky Hierarchy of languages

ANS: Chomsky Hierarchy represents the class of languages that are accepted by different machines.

The category of language in chomsky's Hierarchy are :-

1) Type 0 known as unrestricted Grammer
2) Type 1 known as sensitive grammer
3) Type 2 known as context free grammer
4) Type 3 known as regular gramma



This is a hierarchy, therefore every language of type 3 is also type 0, 1, 2.

6000419005?

1) TYPE 0 GRAMMER

It is known as unrestricted grammer. There is no restriction on the grammer rules. These languages can be efficiently modeled by Turing machines

e.g: $bAa \rightarrow aa$

$S \rightarrow S$

2) TYPE 1 GRAMMER

It is known as context sensitive grammer. It is used to represent context sensitive languages. It follows the following rules :-

1) The context sensitive grammer may have more than one symbol on the left hand side of their production rules
2) The number of symbols on the LHS must not exceed the number of symbols on the RHS
3) The rule of the form $A \rightarrow \epsilon$ is not allowed unless A is start symbol. It does not occur on RHS of any rule.
4) The type 1 grammer should be type 0. Production is in form on $V \rightarrow T$ where the count of symbol in $V <= T$

e.g: $S \rightarrow AT$

$T \rightarrow xy$

$A \rightarrow a$

3) TYPE 2 GRAMMER

It is known as context free grammer. Context free languages are the languages which can be represented by the CFG. Type 2 should be Type 1. The production rule is of the form $A \rightarrow a$ where A is any single non-terminal and is any combination of terminals and non-terminals.

e.g: $A \rightarrow aBb$

$A \rightarrow b$

$B \rightarrow a$

4) TYPE 3 GRAMMER

It is known as ~~singular~~ regular grammer. Regular languages are those languages which can be described using regular expressions. These languages can be modeled by NFA or DFA. Type 3 is the most restricted form of grammer. It should be in form of $V \rightarrow T^*V \mid T^*$

e.g: $A \rightarrow xy$

5

8000419005 57

Q4

$\Sigma = \{0, 1\}$

$S \rightarrow 0S1 \mid 0A$

$A \rightarrow A0 \mid S0 \mid 0$

ANS let given grammer $G = (V, T, P, S)$

$\therefore G = (\{S, A\}, \{0\}, P, S)$

STEP 1: Simplifying the given grammer.

a) Removing ε productions /Null productions

→ there are no ε productions in Grammer

b) Removing unit productions

→ there are no unit productions in grammer.

c) Removing useless variables.

→ there are no useless variable in grammer.

Hence the given grammer is already simplyfied.

converting to Chomsky Normal Form (CNF)

6000419005

$S \rightarrow OS1$ | $S \rightarrow C_0 S C_1$ | $C_0 \rightarrow 0$

$S \rightarrow C_a C_1$ | $C_1 \rightarrow 1$

$C_a \rightarrow C_0 S$

$S \rightarrow OA$ | $S \rightarrow C_0 A$

$A \rightarrow Ao$ | $A \rightarrow A C_0$

$A \rightarrow So$ | $A \rightarrow S C_0$

$A \rightarrow O$

So the CNF of the given grammer is

$G = (\{S, C_0, C_1, C_a, A\}, \{0,1\}, P', S)$

$\therefore P = \begin{cases} S \rightarrow C_a C_1 / C_0 A \\ A \rightarrow AC_0 / SC_0 / 0 \\ C_0 \rightarrow 0 \\ C_1 \rightarrow 1 \\ C_a \rightarrow C_0 S \end{cases}$

converting the Grammer into Greibach Normal Form (GNF)

STEP 2] Renaming the variables as A_i

$S \quad A$

$A_1 \quad A_2$

$\therefore$ The Productions becomes

$A_1 \rightarrow 0A_1 1 / 0A_2$ .. Invalid

$A_2 \rightarrow A_2 0 / A_1 0 / 0$ .. Invalid

$\therefore A_2 \rightarrow A_2 0 / 0A_1 0 / 0A_2 0 / 0 \longrightarrow (1)$

$\therefore$ left recursion is formed in eqn (1)

Using formula of removal of left recursion

$A \rightarrow A\alpha / \beta$

$\therefore A \rightarrow \beta A' / \beta$

$A' \rightarrow \alpha A' / \alpha$

$\therefore A_2 \rightarrow A_2 0 / 0A_1 0 / 0A_2 0 / 0$ let $C_0 \rightarrow 0$, $C_1 \rightarrow 1$

$\therefore A_2 \rightarrow A_2 C_0 / 0A_1 C_0 / 0A_2 C_0 / 0$

$\therefore A \rightarrow A\alpha / \beta_1 / \beta_2 / \beta_3$

$\therefore A_2 \rightarrow 0A_1 C_0 A' / 0A_1 C_0 / 0A_2 C_0 A' / 0A_2 C_0 / 0A' / 0$

$\therefore A' \rightarrow C_0 A' / C_0$

Now, $A_1 \rightarrow 0A_1 1 / 0A_2$

$A_1 \rightarrow 0A_1 C_1 / 0A_2$ $C_1 \rightarrow 1$

NOW, $A' \longrightarrow C_0 A' / C_0$

$A \longrightarrow 0A' / 0$

Hence, the productions are:

$P' \Rightarrow A_1 \longrightarrow 0A_1C_1 / 0A_2$

$A_2 \longrightarrow 0A_1C_0A' / 0A_1C_0 / 0A_2C_0A' / 0A_2C_0 / 0A' / 0$

$A' \longrightarrow 0A' / 0$

$C_0 \rightarrow 0,$

$C_1 \rightarrow 1$

$\therefore G = (\{A, C_1, A_2, A', C_0\}, \{0,1\}, P', S)$

Q5 a) let P, Q, R be regular expressions on Σ. Then if P does not contain ϵ, the equation

$$R = Q + RP \longrightarrow (1)$$

has unique solution given by

$$R = QP^* \longrightarrow (2)$$

Proof: let us verify whether QP^* is solution of $R = Q + RP$

on substitution of QP^* for R in (1)

$$R = Q + RP = Q + QP^*P$$
$$= Q^*(\epsilon + P^*P)$$
$$= QP^*$$

Thus the equation (1) is satisfied by $R = QP^*$. We still don't know if its unique solution.

To establish uniqueness, we expand

$$R = Q + RP$$
$$= Q + (Q + RP)P$$
$$= Q + QP + RP^2$$
$$= Q + QP + QP^2 + RP^3$$
$$\vdots$$
$$= Q(\epsilon + P + P^2 + P^3) + RP^{i+1} \longrightarrow (3)$$

where i is an arbitrary integer.

let us take a string $w \in R$ | length of $w = k$ substituting k for i in eqn (3)

$$R = Q(\epsilon + P + P^2 + \dots + P^k) + RP^{k+1}$$

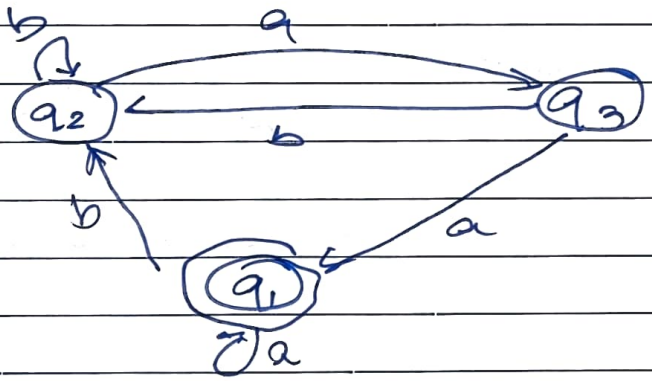
Since P does not contain ε, w is not contained in RP^{K+1} as the shortest string generated by RP^{K+1} will have a length of K+1

since w is contained in R, it must be contained in $Q(\varepsilon + P + P^2 \ldots + P^K)$

Conversely, if w is contained in QP^* then for some integer K it must be in $Q(\varepsilon + P + P^2 \ldots P^K)$ and hence in $R = Q + RP$

Hence verified.

Q5 b



There are no dead states to eliminate.

Equations from diagram:

$q_1 = \varepsilon + q_1 a + q_3 a$ → (1)

$q_2 = q_1 b + q_2 b + q_3 b$ → (2)

$q_3 = q_2 a$ → (3)

substituting (3) in (2)

$q_2 = q_1b + q_2b + q_2ab$

$q_2 = q_1b + q_2(b + ab)$

comparing with $R = Q + RP$ we get

$q_2 = q_1b\,(b+ab)^*$ → (4)

$q_3 = q_1b\,(b+ab)^*a$ → (5)

substituting (5) in (1)

$q_1 = q_1a + q_2aa + \epsilon$

$\therefore q_1 = q_1a + q_1b(b+ab)^*aa + \epsilon$

$= q_1(a + b(b+ab)^*aa) + \epsilon$

$= [a + b(b+ab)^*aa]^*$

R.E : $[a + b(b+ab)^*aa]^*$ //