Automata Theory : Theory Assignment 23/08/2022

Aaryan Ajay Sharma - 2022121001

01

1. (a) We see that there is no incoming edge from any state to state 5, therefore State 5 is unreachable. And though state 4 has incoming edge, the incoming edge is of state 5, which is unreachable. Every other state is reachable from the initial state 1. ∴ state 5 and 4 are spurious.

(b) We observe that once we reach state 3 or state 5, there is no way to reach the final state (6), as we are stuck in a loop of 3 and 5.

∴ State 3 and State 5 are dead states.
⇒ State 3 and State 5 are spurious.

2. We say that (p, q) states are equivalent if

$$\delta(p,w) \in F \implies \delta(q,w) \in F$$

AND

$$\delta(p,w) \notin F \implies \delta(q,w) \notin F.$$

where w is any string.

If $|w| = 0$, (p, q) are 0-equivalent

If $|w| = 1$, (p, q) are 1-equivalent

In general, If $|w| = n$, (p, q) are n-equivalent.

In case, $\forall n \in \mathbb{N}$, the condition holds, then the states are equivalent and we can make them as one state.

Now, first we check if there are any unreachable state in our diagram.

→ If there are, we remove them
→ If not, then we do nothing.

By inspection, we see no state is unreachable, therefore we need not remove any states.

Now, draw the State transition diagram for the DFA.

	0	1
→ 0	1	8
1	2	4
2	8	3
3	2	8
4	5	6*
5	8	6*
6	7	8
7	7	7
8	8	8

__/__/____

Now, we iteratively find the n-equivalent state from (n-1) - equivalent states until we get both of them as the same. Once we do, we stop.

0-equivalent states:

[0 1 2 3 4 5 7 8] [6]

(partitioning initial & final states).

1-equivalent states:

[0 1 2 3 7 8] [4 5] [6]

4 & 5 go to different partitions as

$\delta(4,1) = 6$ and $\delta(5,1) = 6$

and since 6 is in different partition, we partition both.

2-equivalent states:

[0 2 3 7 8] [1] [4] [5] [6]

1 is partitioned as $\delta(1,0) = 2$ & $\delta(1,1) = 4$

and since 2 & 4 don't belong to the same state, we partition.

5 is also partitioned ~~into~~ differently as

~~&~~ $\delta(5,0) = 8$ and $\delta(5,1) = 6$

and since $\delta(4,0) = 5$ which is not in the same partition as 8, it needs to be partitioned.

3 - equivalent states:

[0] [2 3 7 8] [1] [4] [5] [6].

Here, 0 is partitioned differently as

$\delta(0,0) = 1$ and $\delta(2,0) = 8$.

Since 1 & 8 are not in the same partition, we again seperate them.

4 - equivalent:

[0] [2 3 7 8] [1] [4] [5] [6]

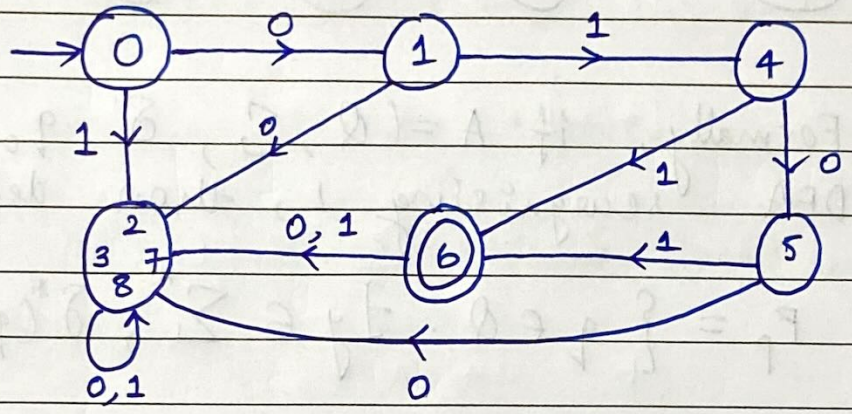
We observe that 2, 3, 7 & 8 cannot be partitioned more as all land on the same state/partition on transition.

and since we have 3 - equivalent = 4 - equivalent, we stop.

So, we minimized our number of states from 9 to 6.

05

The following is the minimized DFA :-

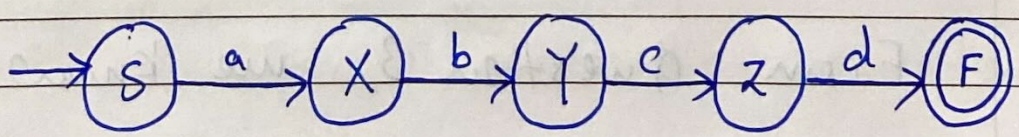


3. To accept all prefixes of a string of the form xy

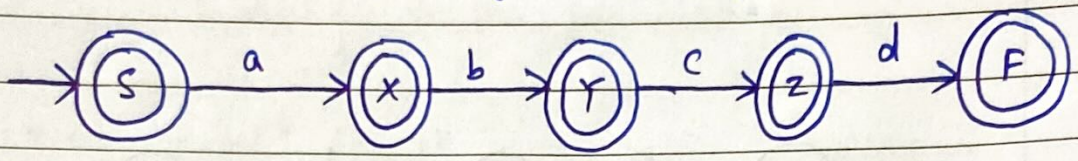
To accept all strings of the form x, where $x \in L$, we can make all states that lie on the path leading to final state as final states.

By doing this, we see that the modified FA would accept any arbitrary x, where $xy \in L$ including the cases where $x = \varepsilon$ or $x = y$.

Example: NFA accepting abcd



NFA accepting any prefix of abcd :-



Formally, if $A = (Q, \Sigma, \delta, q_0, F)$ is a DFA recognising L, then define F_p by

$$F_p = \{ q \in Q : \exists y \in \Sigma^*, \delta^*(q, y) \in F \}$$

Then $A_p = (Q, \Sigma, \delta, q_0, F_p)$ recognises Pre(L)

as for $x \in \Sigma^*$ we have

$$x \in L(A_p) \iff \delta^*(q_0, x) \in F_p$$

$$\iff \exists y : \delta^*(\delta^*(q_0, x), y) \in F,$$

$$\iff \exists y : \delta^*(q_0, xy) \in F$$

$$\iff \exists y : xy \in L.$$

$$\iff x \in Pre(L).$$

4. From Question 3, we know that Regular Languages (RL) are closed under

prefixes. i.e. L is RL $\Rightarrow$ Pre(L) is RL.

Now, define

$$Suff(L) = Pre(L^R)$$

where L^R is the reversal of L.

since RL are closed under Reversal,

$\therefore$ $Suff(L) = Pre(L^R)$ is also regular.

$Pre(L^R) = \{ x \in \Sigma^* \mid \exists y \in \Sigma^* \text{ such that } xy \in L^R \}$.

Now, if $xy \in L^R \Rightarrow yx \in L$.

$\therefore$ $Suff(L) = \{ x \in \Sigma^* \mid \exists y \in \Sigma^* \text{ such that } yx \in L \}$.

$\therefore$ The suffix closure of L is Regular.

By careful observation, we notice that

$$DROP\text{-}OUT^*(A) = Pre(A) \cdot Suff(A)$$

where $\cdot$ is the concatenation operation.

Now, from the above argument, it follows that

DROPOUT*(A) is Regular. when A is regular. □

5. We define,

$$[a\text{-}z] = \{a, b, c, \ldots, z\}$$

$$\Sigma^+ = \Sigma^* - \{\epsilon\}$$

$+$ = union

∴ The Regular expression of the IIIT email address is:

$$[a\text{-}z]^+ (\epsilon + .[a\text{-}z]^+) @ (\epsilon + research.+ students.)$$
iiit.ac.in.

6. (a) Regular expression into Right Linear Grammar.

First we need to ~~convert~~ convert RE to FA.

Step 1: Make a transition diagram for given using NFA with ϵ-transitions.

Step 2: Then, convert the obtained NFA

07

to equivalent DFA

Now, we need to convert Finite Automata to Right Linear Grammar

Step -1: Begin the process from the start state

Step-2: Repeat the process for each state

Step-3: Write the production as the output followed by the state on which the transition is going.

Step-4: And at last, add ∈ (epsilon) to end the derivation.

b) Right Linear Grammar to Left Linear Grammar

A right Linear Grammar can be converted to left Linear Grammar as follows:

1. Construct a finite automata from the right Linear Grammar
2. Reverse the transition directions for each transition
3. Make final states to non-final and non-final to final state.

4. derive a left linear grammar from this automata.

7. a) suppose that the set of all primes is regular in Binary representation.

Let there be m states in it's DFA.

Now, let p be a prime whose binary expansion ends with m zeros and a 1, so that

$$p = \underbrace{\cdots \cdots \cdots}_{n \text{ in binary}} \underbrace{00\ldots001}_{m \text{ zeros}} = 2^{m+1} n +$$

for some n. (Dirichlet's Theorem).

The pumping lemma says that there is a non-empty substring of the m-consecutive zero which can be "pumped". That is, there exists s such that $0 < s < m$ and

$$\underbrace{\cdots \cdots \cdots}_{n \text{ in binary}} \underbrace{00\ldots00\ldots001}_{m+Ks \text{ zeros}}$$

is prime $\forall K$. But

$$q = (p-1)\, 2^{Ks} + 1$$

$\Rightarrow \quad q = 2^{ks} p - (2^{ks} - 1).$

Now, let $k = p - 1$.

then by fermat's little theorem $2^k \equiv 1 \bmod p$.

or $p \mid 2^{ks} - 1$, or $p \mid q$.

$\therefore$ q is not a prime $\Rightarrow$ Contradiction.

$\therefore$ set of primes in binary representation is not regular

b) Assume $L = \{ a^{n!} \mid n \geq 0, n \in \mathbb{N} \}$ is regular.

let its DFA have m states.

let $w \in L$, $w = a^{m!} = a^m a^{m! - m}$.

since $|w| \geq m$

$x = a^t$, $y = a^k$, $z = a^j a^{m! - m}$.

$t + k + j = m$ and $1 \leq k \leq m$

From pumping lemma,

$w' = a^t a^{k \cdot i} a^j a^{m! - m} \in L. \; \forall i, j, k \neq t.$

12

let $i=2$

$\therefore w' = a^t a^k a^k a^j a^{m!-m} = a^{m+k} a^{m!-m} = a^{m!+k}$.

Now, $1 \le k \le m \Rightarrow m!+1 \le m!+k \le m!+m$.

since $m > 1$

$$m! < m!+m \le 2m! < (m+1)!$$

But for $m=1 \Rightarrow k=1$ and for $i=3$ we get

$$2! < |w'| = m! + 2k = 3 < 3!$$

$\therefore w' \notin L$ as every $w \in L$ has factorial length, which is a contradiction.

8. The grammar for palindrome is known to be as follows:

$$S \rightarrow \varepsilon \mid a \mid b \mid aSa \mid bSb$$

We create the grammar of $L = \{a,b\}^* - \{\text{palindrome}\}$ by observing the following:

- If x is the element of L then:
  - axa is an element of L
  - bxb is an element of L
- If y is an element of $\{a,b\}^*$ then:
  - ayb is an element of L
  - bya is an element of L.

__/__/____

Observing the above information, ~~the~~ the CFG of L would be

- $S \rightarrow aSa \mid bSb \mid aAb \mid bAa$
- $A \rightarrow \varepsilon \mid Aa \mid Ab$.

Chomsky Normal form:

1) Remove epsilon:

$S_0 \rightarrow S$
$S \rightarrow aSa \mid bSb \mid aAb \mid bAa \mid ab \mid ba$
$A \rightarrow Aa \mid Ab \mid a \mid b$

2) Remove Duplicate key Value

$S_0 \rightarrow S$
$S \rightarrow aSa \mid bSb \mid aAb \mid bAa \mid ab \mid ba$
$A \rightarrow Aa \mid Ab \mid a \mid b$

3) Remove Single Variable in Every Production

$S_0 \rightarrow aSa \mid bSb \mid aAb \mid bAa \mid ab \mid ba$
$S \rightarrow aSa \mid bSb \mid aAb \mid bAa \mid ab \mid ba$
$A \rightarrow Aa \mid Ab \mid a \mid b$

4) Assign new Variable for two non-terminal or one terminal.

$S_0 \rightarrow$ ~~aSb~~ $aSa \mid bSb \mid aAb \mid bAa \mid ab \mid ba$

$S \rightarrow aSa \mid bSb \mid aAb \mid bAa \mid ab \mid ba$

$A \rightarrow Aa \mid Ab \mid a \mid b$

$G \rightarrow Sa$

$H \rightarrow Ab$

$J \rightarrow Aa$

$K \rightarrow a$

$L \rightarrow b$

5) Replace two terminal with new variable

$S_0 \rightarrow KG \mid LH \mid KI \mid LJ \mid KL \mid LK$

$S \rightarrow KG \mid LH \mid KI \mid LJ \mid KL \mid LK$

$A \rightarrow AK \mid AL \mid a \mid b$

$G \rightarrow SK$

$H \rightarrow SL$

$I \rightarrow AL$

$J \rightarrow AK$

$K \rightarrow a$

$L \rightarrow b$ $\rightarrow$ **CNF**

9. $L = \{ a^n b^n \cup b^n a^n \mid n \geq 0 \}$.

$S \rightarrow aXa \mid bXb \mid a \mid b$

$X \rightarrow aX \mid bX \mid a \mid b \mid \varepsilon$

let the language described by the above CFG

be L'.

Now, L' : 1) contain a & b
2) start & end with different letter but in between the string ∉ L.

Since we can only generate strings going from
$S \rightarrow aSb$ or bSa.
which will never terminate.

∴ only strings of the forms $a^n b^n$ or $b^n a^n$ will never be accepted by L'

∴ L' is the complement of L.

10. let $DFA_A \notin = (Q_A, \Sigma, \delta_A, S_A, F_A)$ & $DFA_B \notin (Q_B, \Sigma, \delta_B, S_B, F_B)$ be the DFAs that recognize A & B respectively.

$DFA_{mix} = (Q, \Sigma, \delta, S, F) = mix(A, B)$.

The DFA for mix (A, B) ~~from A to B~~ switches from DFA_A to DFA_B after each character is read and tracks the current state of DFA_A & DFA_B. i.e., $a_i, b_i \in \Sigma$. For each character read, DFA_{mix} makes moves in the corresponding DFA (either DFA_A or DFA_B)

After the whole string is read, if both DFA_A and DFA_B reaches to the final state, then the input string is accepted by DFA_{mix}.

The DFA_{mix} is defined as follows:

- $Q = Q_A \times Q_B \times \{A, B\}$: set of all possible states of DFA_A and DFA_B which should match with DFA_{mix}.

- The input alphabet of DFA_{mix} is Σ.

- $q = (q_A, q_B, A)$: q_A and q_B are the initial states for DFA_A and DFA_B respectively. DFA_{mix} starts with q_A in DFA_A, q_B in DFA_B and the next character should be read from DFA_A.

- $F = F_A \times F_B \times \{A\}$: F_A and F_B are the final states for DFA_A and DFA_B respectively, DFA_{mix} accepts if both DFA_A and DFA_B reaches to the final states and the character should be read from DFA_A.

- The transition function δ is
  1. $\delta((m, n, A), a) = (\delta_A(m, a), n, B)$
  2. $\delta((m, n, B), b) = (m, \delta(n, b), A)$.

Consider, the current state of DFA_A is m and current state of DFA_B is n. Change the current state of A to $\delta_A(m, a)$ if the next character is to be read from DFA_A when a is the next character.

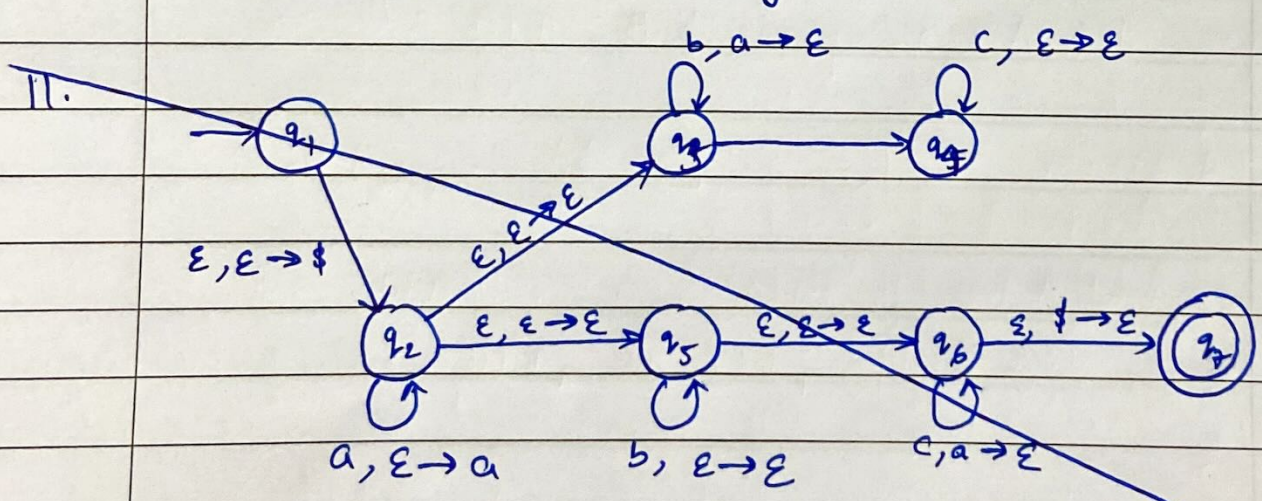
After the character is read, read the next character from DFA_B. Change the current state of B to $\delta_B(n, b)$ if the next character is to be read from DFA_B when b is the next character.

The language L is said to be regular if there exist an FA recognising L.

Here, DFA_{mix} is defined for the language mix(A, B).

∴ mix(A, B) is regular.

11.



PDA recognizing $\{a^i b^j c^k \mid i, j, k \geq 0 \text{ and } i = j \text{ or } j = k\}$

11.

$$L = \{ a^i b^j c^k \mid i, j, k \geq 0,\ i = j \text{ or } j = k \}.$$

PDA for language is: