Introduction to Languages & Theory of Computation

1 ⇒ Mathimatical induction & Recursive defns
- → The principle of M.I
- → The Strong principle of M.I
- → Recursive definitions of functions with Domain N
- → Recursive definitions of Sets
- → Structural Induction
- → Read chap-2 of John C. Martin for Review

Evening

R ⇒ Regular Languages and Regular Expressions

→ The set R of regular languages over Σ, and the corresponding regular expressions, are defined as follows.

FRIDAY 1 (١٤ محرم)
SUNDAY 3 (١٦ محرم)
SATURDAY 2 (١٥ محرم)

(i)- ϕ is an element of R and the corresponding R.E is ϕ

(ii) $\{\Lambda\}$ is an element of R and the R.E is Λ

(iii) for each $a \in \Sigma$, $\{a\}$ is an element of R, and the corresponding R.E is a.

(iv)- If L_1 and L_2 are any elements of R, and γ_1 and γ_2 are corresponding R.Es

(a). $L_1 \cup L_2 \in R$ and $R.E = (\gamma_1 + \gamma_2)$

(b) $L_1 L_2 \in R$ and its $R.E = (\gamma_1 \gamma_2)$

(c) $L_1^* \in R$ and its $R.E = \gamma_1^*$

Hence Languages that can be obtained using (i) — (iv) are regular languages.

→ Simplified rules for languages & R.E

→ Discard curly brackets { } or replace them with parenthesis

→ Replace ∪ by +

Some examples are as below.

Language	R.E
{1}	1
{0}	0
{001} (i.e. {0}{0}{1})	001
{0,1} (i.e. {0} ∪ {1})	0+1
{1,1}{001}	(1+1)001
{110}*{0,1}	(110)*(0+1)
{0,10}*({11}* ∪ {001, Δ})	(0+10)*((11)* + (001+1))

1*10 is an R.E for strings that consist of substring 10 preceded by any number of 1's

Similarly

$L^2 = LL$

$r^2 = rr$

$r^+ = ((r^*)r)$

→ Precedence (Kleene*, Concatenation, +)
So be Careful in parenthesizing e.g

$$a + b^*c = (a + ((b^*)c))$$

→ Are the following R.Es equal or not over {0,1}

$$1^*(1+\Lambda) = 1^*$$

$$1^*1^* = 1^*$$

$$(0^*1^*)^* = (0+1)^*$$

$$L.H.S = (0^*1^*)^* \Rightarrow \Big((\Lambda,0,00,000,\dots)(\Lambda,1,11,111,\dots)\Big)^*$$

$$\Rightarrow \Big((\Lambda,1,11,111,\dots),(0,01,011,0111,\dots),$$

$$\dots\dots (000,0001,00011,00111,\dots),\ \dots\dots\dots\Big)^*$$

$$\Rightarrow (\Lambda,0,1,01,10,00,11,\dots\dots) = (0+1)^*$$

$$= R.H.S$$

→ Examples

✓(1) $L \subseteq \{0,1\}^*$ be the language of all strings of even Length

$$L = \{00,01,10,11\}^*$$

$$R.E = (00+01+10+11)^* \text{ or } ((0+1)(0+1))^*$$

WEDNESDAY 6 ١٩ محرم
THURSDAY 7 ٢٠ محرم

(ii) Strings with odd number of 1's

R.E = $0^*10^*(10^*10^*)^*$ ✓

OR = $0^*1(0^*10^*1)^*0^*$ ✓

OR = $(0^*10^*1)^*0^*10^*$ ✓

OR = $0^*(10^*10^*)^*1(0^*10^*1)^*0^*$ ✓

Now is it the same R.E

$(10^*10^*)^*10^*$

Strings with 0 in beginning are not producing

another Correct version

$0^*(10^*10^*)^*10^*$ ✓

8 FRIDAY محرم ٢١

9 SATURDAY محرم ٢٢ 10 SUNDAY محرم ٢٣

(iii) Strings of length 6 or less

R.E = $(0+1)(0+1)(0+1)(0+1)(0+1)(0+1)$

or $(0+1)^6$

or $(0+1+\Lambda)^6$ to allow Λ

✓ (IV) Strings ending in 1 and not Containg 00

R.E = $(1+01)^*\cdot(1+01)$

or $(1+01)^+$

Evening

(v) Language for C identifiers

$l = a, b, c, \ldots z, A, B, \ldots Z$

$d = 0, 1, 2, \ldots 9$

$R.E = (l + \_)(l + d + \_)^*$

(vi) – Strings ending with 0 e.g $(0+1)^*0$
(Remembering only the last digit 0, else is not need to be remembered).

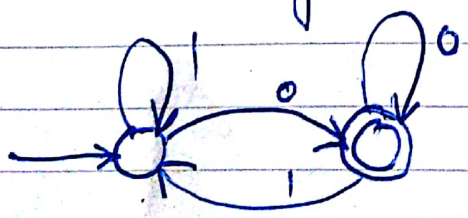
(vii) – Algorithm for strings with next to last symbol 0

MONDAY 11 (٢٤ محرم)
TUESDAY 12 (٢٥ محرم)

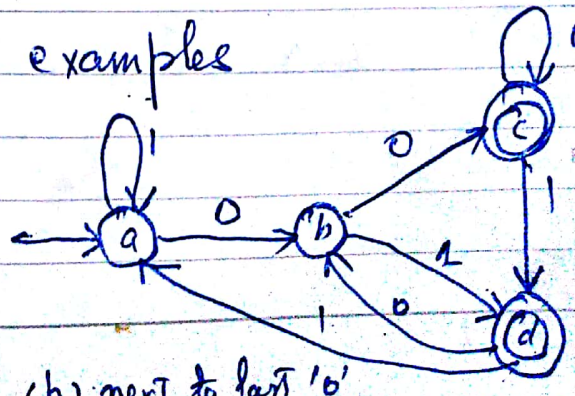
a. The strings is Λ or 1 or end with 11 } not allowed
b. The strings is 0 or ends with 10 }
c. The strings with 00 } allowed
d. The strings end with 01 }

(viii) – Algorithm for (IV) which is ending in '1' and Containg 00 ?

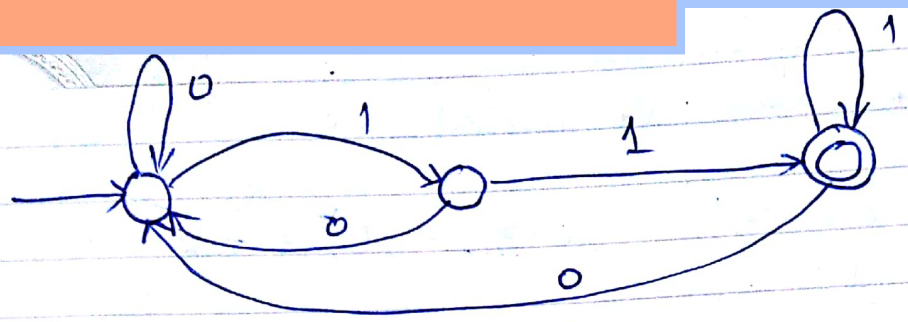
→ Flow Diagrams for examples



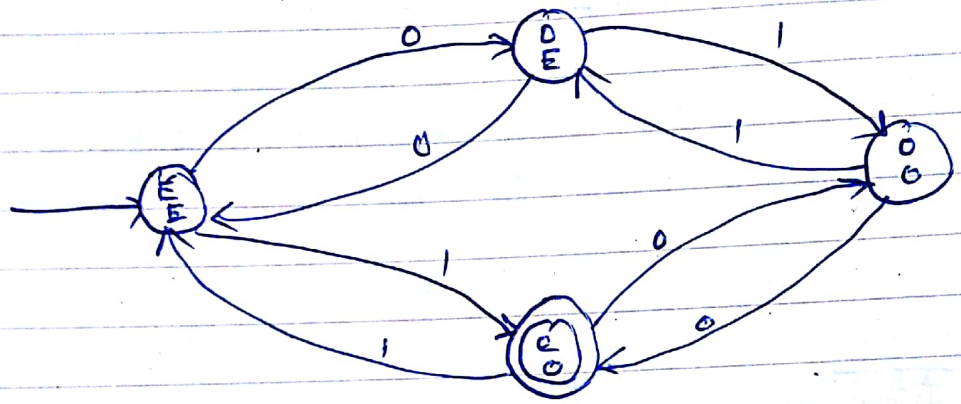
(a) ending in '0'



(b) next to last '0'



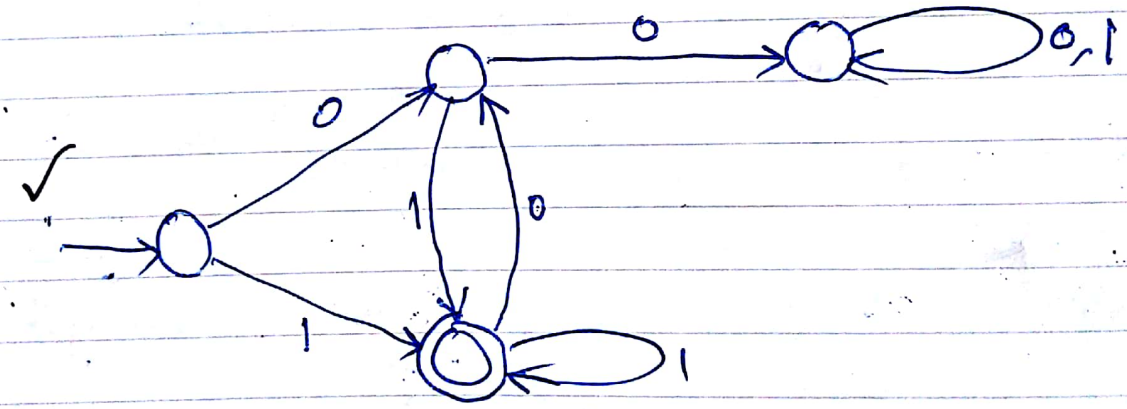
(c) ending with '11'



13 WEDNESDAY ٢٦ محرم

14 THURSDAY ٢٧ محرم

(d) n_0 even and n_1 odd



(e) ending in '1' not containing 00