Algebraic Laws for RE

Lecture -10

page - 0

06-05-20

Associativity & commutativity

1) $L+M = M+L$
2) $(L+M)+N = L+(M+N)$
3) $(LM)N = L(MN)$
4) $LM \neq ML$

Identities and Annihilators

1) $\phi + L = L$
2) $L + \phi = L$
3) $\in L = L$
4) $\phi L = L\phi = \phi$

set of all linear functionals that map to zero all elements of a given subset of a vector space

↑ True for (•)

Distributive Law

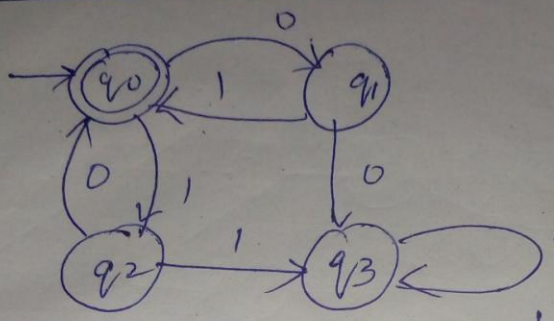
1) $L(M+N) = LM + LN$
2) $(L+M)N = LN + MN$.

Idempotent law

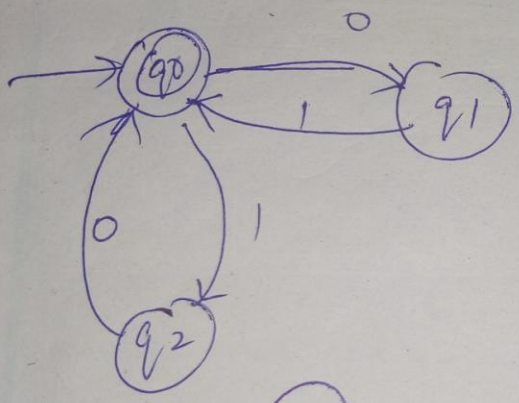
$L + L = L$

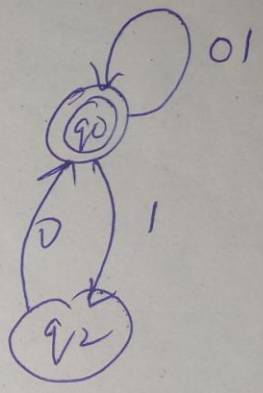
FA to RE

Q/

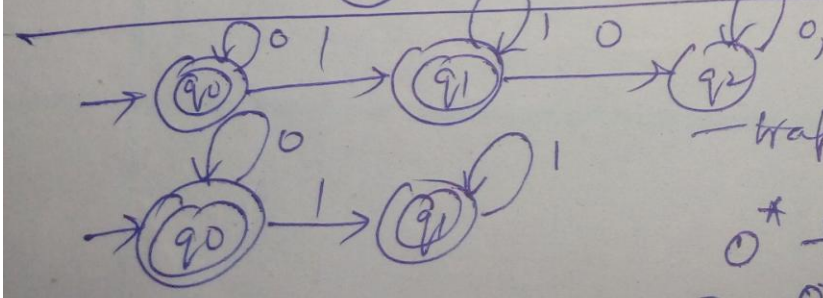


↑ trap / remove it



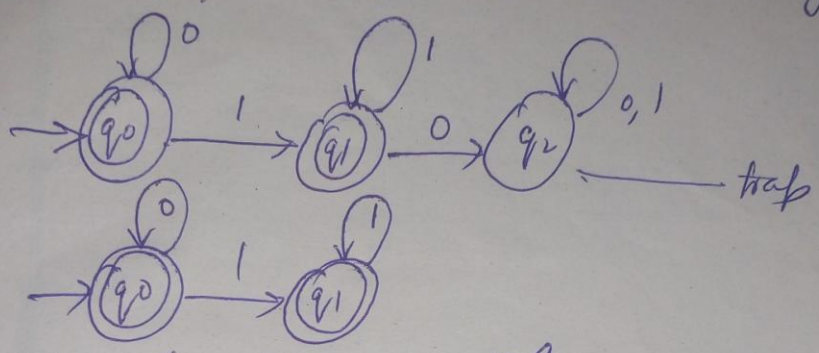


$$RE = (01+10)^*$$



$$0^* + 0^*11^*$$
$$= 0^*(\epsilon + 11^*) = 0^*($$
$$= 0^*1^*$$

What is the language accepted by the FA? (2)



RE = 0^*1^* = 0^*1^*

Language consists of any number of 0's and 1's

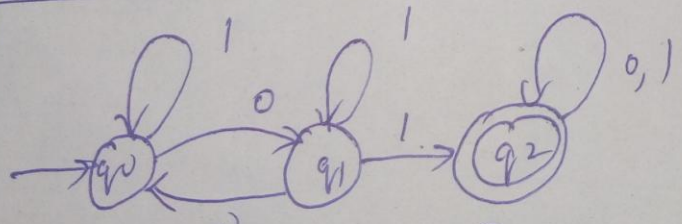
$RE = 0^* + 0^*11^*$

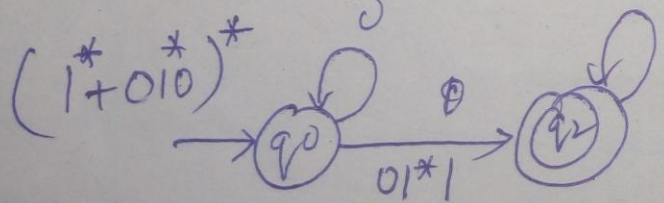
$= 0^*(\epsilon + 11^*)$

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

$= 0^*1^*$

Q3)





$RE = (1^* + 01^*0)^* \cdot 01^*1 \cdot (0+1)^*$

Applications

Pattern matching

↑ refers to a set of objects with some common properties

- searching for strings / identifiers

Lexical analysis

REs are extensively used in the design of a lexical analyser phase.

- This phase scans the source program and recognizes all the tokens which are logically together.

- UNIX

← lex accepts a RE as input command and produces the lexical analyzer generator.

Chomsky Hierarchy.

$G = (V, T, P, S)$

- 4 types of grammar.
- Noam Chomsky (founder of formal language theory) classified the grammars into 4 types.

type 0, 1, 2, 3

Type 0 Grammar (Unrestricted grammar / phase structured grammar)

Def

$G = (V, T, P, S)$ is said to be of type 0 or unrestricted grammar, if all the productions are of the form $\alpha \to \beta$, where

$$\alpha \in (V \cup T)^*$$

$$\beta \in (V \cup T)^*$$

- no restriction on lengths of α and β.
- only restriction is that α cannot be $\in$. i.e, $\in$ cannot be on the LHS of any production, BUT
  → $\in$ can appear on the RHS of the function

- largest family of grammars
- most powerful
- any language can be obtained from this grammar.
- language generated from this grammar is called type 0 lang or recursively enumerable lang.
  (it is a formal language which will enumerate all valid strings of the language)

example: Turing machine.

Eg $S \to aAb \mid \varepsilon$

$aA \to bAA$

$bA \to a$

Type 1 grammar (Context sensitive grammar)

Def $G = (V, T, P, S)$ is said to be type 1 or context sensitive grammar if productions are of the form $\alpha \to \beta$, but there is a restriction on β (length of β must be atleast as much as length of α)

$|\beta| \geq |\alpha|$

α and $\beta \in (V \cup T)^*$

i.e, ϵ cannot appear on the LHS or RHS of any production

- It is ϵ-free grammar.
- Language generated is called type 1 language or context sensitive language.
- Linear bounded automata (LBA) can be constructed to recognize the language generated from this grammar.

$S \rightarrow aAb$

$aA \rightarrow bAA$

$bA \rightarrow aa$