Morning

Enumerating Binary Strings

If w is a binary string, treat 1w as a binary integer i.

ε,   0, 1, 00, 01, 10, 11, 000, 001, 010, 011, …

1, 10,11,100,101,110,111,1000,1001,1010,1011, …

Coding for Turing machine

Let TM $P = (Q, \{0,1\}, \Gamma, \delta, q_1, B, \{q_2\})$

Where $Q = \{q_1, q_2, \ldots, q_r\}$, $\Gamma = \{X_1, X_2, X_3, \ldots, X_s\}$

$$X_1 : 0,\ X_2 : 1,\ X_3 : B,\ D_1 : \leftarrow,\ D_2 : \rightarrow$$

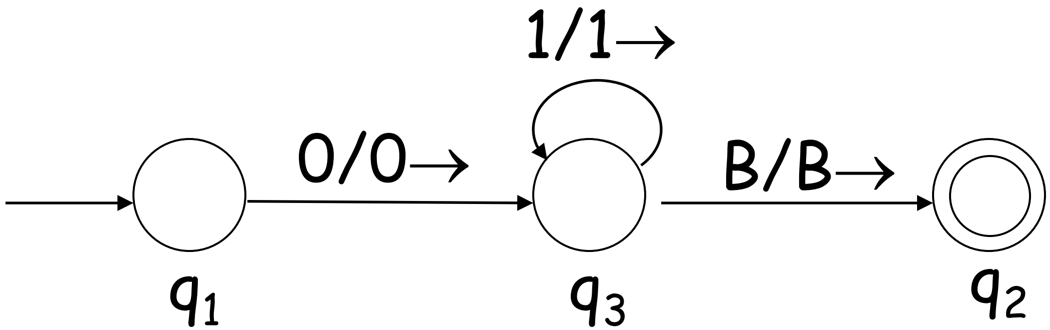
Coding :

$$\delta(q_i, Xj) = (q_k, X_m, D_n)$$

$$\Rightarrow 0^i 1 0^j 1 0^k 1 0^m 1 0^n$$

$$P \Rightarrow C_1 11\ C_2 11\ C_3 11 \ldots C_{n-1} 11\ C_n$$

Example TM for L= {0}{1}*



$\delta(q_1,0) = (q_3,0,\rightarrow) \Rightarrow 0101000101 00$

$\delta(q_3,1) = (q_3,1,\rightarrow) \Rightarrow 00010010001001 00$

$\delta(q_3,B) = (q_2,B,\rightarrow) \Rightarrow 000100010010001 00$

TM $\Rightarrow$ 010100010100 11 0001001000100100 11

00010001001000100

Non-recursively enumerable language

$$L_d = \{\, w_i \mid w_i \notin L(M_i) \,\}$$

	1	2	3	4		...	
1	0	1	1	0		...	
2	1	1	0	0		...	
3	0	0	1	1		...	
4	0	1	0	1		...	
.	.	.	.	.	.		
.	.	.	.	.		.	
.	.	.	.	.			.

L_d is not Recursively Enumerable

Theorem L_d is not a recursively enumerable language. That is there is no TM that accept L_d .

Proof : Suppose L_d were L(M) for some TM M.

$\Rightarrow$ There is at least one code for M, say i, that $M=M_i$

Now, ask if w_i is in L_d .

- w_i is in $L_d \Rightarrow M_i$ accepts $w_i \Rightarrow w_i$ is not in L_d
- w_i is not in $L_d \Rightarrow M_i$ does not accept $w_i \Rightarrow w_i$ is in L_d

Recursive languages

Definition

L is recursive if L=L(M) for some TM M

such that

1. $w \in L \Rightarrow$ M accepts w and halts

2. $w \notin L \Rightarrow$ M eventually halts

Recursive languages

Theorem If L is recursive language, so is $\overline{L}$.

Suppose $L=L(M)$, $M=(Q, \Sigma, \Gamma, \delta, q_0, B, F)$

Let $\overline{M}=(Q \cup \{r\}, \Sigma, \Gamma, \delta, q_0, B, \{r\})$ such that

1. r is a new state which is not in Q
2. if $\delta(q,a) = \phi$ for any $q \in Q-F$ and $a \in \Sigma$

   then $\delta(q,a) = (r, a, \rightarrow)$

Recursive languages

Theorem If both L and its complement $\overline{L}$ are RE, then L is recursive.

Suppose $M_1 = (Q_1, \Sigma, \Gamma, \delta_1, q_1, B, F_1)$

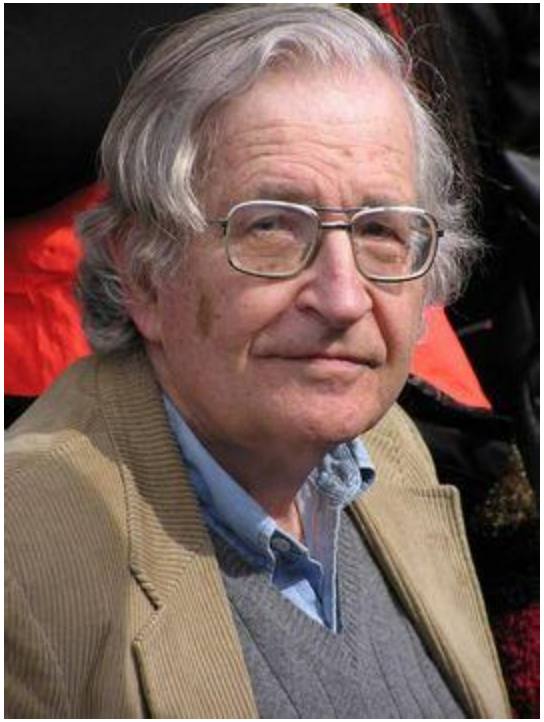
$M_2 = (Q_2, \Sigma, \Gamma, \delta_2, q_2, B, F_2)$

$M = (Q_1 \times Q_2, \Sigma, \Gamma, \delta, (q_1, q_2), B, F_1 \times (Q_2 - F_2))$

$\delta((p,q),(a,b)) = (\delta_1(p,a), \delta_2(q,b))$

Chomsky Grammar

- Noam Chomsky (1928-)
- Chomsky Grammar (1956)
- Syntactic Structures



Chomsky Grammar

Type 0: phrase structure grammar(PSG)

$$\alpha \rightarrow \beta\ ;\ \ \alpha \in (V \cup T)^* V\, (V \cup T)^*,\ \beta \in (V \cup T)^*$$

Type 1: context sensitive grammar(CSG)

$$\alpha A \beta \rightarrow \alpha \omega \beta\ ;\ \ A \in V,\ \alpha, \omega, \beta \in (V \cup T)^*$$

Type 2: context free grammar(CFG)

$$A \rightarrow \omega\ ;\ \ A \in V,\ \omega \in (V \cup T)^*$$

Type 3: regular grammar(RG)

$$A \rightarrow \alpha \mid \alpha B;\ \ A, B \in V,\ \alpha \in T^*$$

Linear Bounded Automata

A linear bounded automata is a nondeterministic Turing machine $M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$

that Σ must contain two special symbols [and], such that $\delta(q_i,[)$ can contain only elements of the $(q_j,[, \rightarrow)$, and $\delta(q_i,])$ can contain only elements of the $(q_j,], \leftarrow)$

Linear Bounded Automata

Example   Consider the language

$$L=\{\ a^n b^n c^n \mid n \geq 1\ \}$$

1. Design a LBA to accept L

2. Give a CSG for L

$\underline{CSG}$ $\underline{W=aaabbbccc}$

$S \longrightarrow aDc$

$D \longrightarrow aDE \mid b$

$bEc \longrightarrow bbcc$

$bEE \longrightarrow bbFE$

$FE \longrightarrow FF$

$FFc \longrightarrow GFc \longrightarrow Gcc$

$FG \longrightarrow GG$

$bGc \longrightarrow bbcc$

$bGG \longrightarrow bbHG$

$HG \longrightarrow HH$

$HHc \longrightarrow EHc \longrightarrow Ecc$

$HE \longrightarrow EE$

S

$a\underline{D}c$

$aa\underline{DE}c$

$aaa\underline{DE}Ec$

$aaa\underline{bEE}c$

$aaabb\underline{FE}c$

$aaabb\underline{FFc}$

$aaabb\underline{GFc}$

$aaab\underline{bGcc}$

$aaabbbccc$

<u>*PSG*</u> <u>*W=aaabbbccc*</u>

$$S \longrightarrow abc \mid aAbc$$
$$Ab \longrightarrow bA$$
$$Ac \longrightarrow Bbcc$$
$$bB \longrightarrow Bb$$
$$aB \longrightarrow aa \mid aaA$$

S

$\underline{aAbc}$

$ab\underline{Ac}$

$a\underline{bBbcc}$

$\underline{aBb}bcc$

$aa\underline{Ab}bcc$

$aab\underline{Ab}cc$

$aabb\underline{Ac}c$

$aab\underline{bBbcc}c$

$aa\underline{bBb}bccc$

$a\underline{aBb}bbccc$

$aaabbbccc$

Right Linear Grammars

A grammar $G = (V, T, S, P)$ is said to be right linear if all productions are of the form

$$A \to xB$$

$$A \to x$$

where $A, B \in V$, and $x \in T^*$

Example $G = (\{S\}, \{a, b\}, S, P)$

$$S \to abS \mid a$$

Left Linear Grammars

A grammar $G = (V, T, S, P)$ is said to be left linear if all productions are of the form

$$A \to Bx$$

$$A \to x$$

where $A, B \in V$, and $x \in T^*$

Example $G = (\{S\}, \{a, b\}, S, P)$

$$S \to Sba \mid a$$

Example Design regular grammars for

(1) $L=\{w \mid w \in \{0, 1\}^* \text{ and ending with } 01\}$

$L = \{0, 1\}^* \{01\}$

$S \rightarrow A\,01$

$A \rightarrow A0 \mid A1 \mid \varepsilon$

ie, $S \Rightarrow A01 \Rightarrow A001 \Rightarrow A0001 \Rightarrow A10001 \Rightarrow 10001$

$G=(\{S, A\}, \{0, 1\}, S, P)$

What is the right linear grammar for L ?

Example Design regular grammars for

(2) $L=\{w \mid w \in \{0, 1\}^* \text{ and } w \text{ contains } 01\}$

$L = \{0, 1\}^* \quad \{01\} \quad \{0, 1\}^*$

$S \to 0S|1S, \quad S \to 01A, \quad A \to 0A|1A|\varepsilon$

$G=(\{S, A\}, \{0, 1\}, S, P)$

P : $S \to 0S|1S|01A, \quad A \to 0A|1A|\varepsilon$

Good good study

day day up !