12 Modeling Computation

Alphabets and Languages

Alphabet = Finite set of symbols which cannot be further decomposed into other symbols.

Ex. Binary alphabet: $\{0, 1\}$

String (Word) Ex. Montana

Length of a string: $|\text{Montana}| = 7$, $|\lambda| = 0$

Σ^* : Set of all strings over an alphabet (including λ)
Σ^+ : $\Sigma^* - \{\lambda\}$

Language : $L \subseteq \Sigma^*$

Concatenation

Ex. 1 $A = \{0,1\}$, $B = \{1, 10, 110\}$
$AB = \{01, 010, 0110, 11, 110, 1110\}$

Kleene Closure

$$A^* = \bigcup_{k=0}^{\infty} A^k$$

Ex. 3 $B = \{0,1\}$, $C = \{11\}$
$B^* = \{\lambda, 0, 1, 00, 01, 10, 11, \ldots\}$
$C^* = \{\lambda, 11, 1111, 111111, \ldots\}$

12.1 Grammar

Def. G = (N, T, S, P)

N : nonterminal (symbols) // A, B, C, … //
T : terminal (symbols) // a, b, c, … //
S : start symbol
P : productions

Ex. 1 G_1: (N, T, P, S)

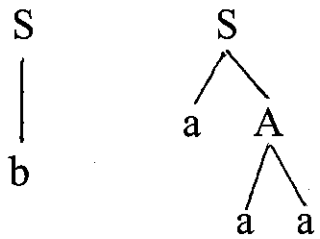
N = {A, B, S}
T = {a, b}
P: S → ABa
A → BB
B → ab
AB → b

Def. $L(G) = \{w \in T^* \mid S \overset{*}{\Rightarrow} w\}$

Ex. 3 G_3: (N, T, P, S)

N = {S, A}
T = {a, b}
P: S → aA | b
A → aa

L(G3) = {b, aaa}



Derivation tree
(Parse tree)

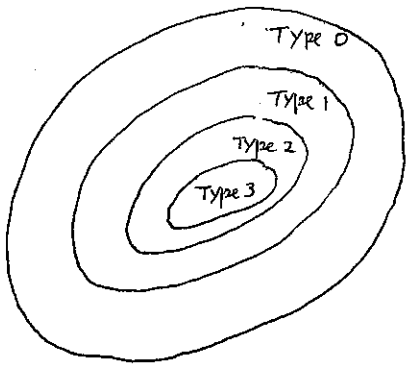
Ex. 5 $L = \{0^n 1^n \mid n = 0, 1, 2, \ldots\}$

G: $S \rightarrow 0S1$
$S \rightarrow \lambda$

Types of Grammars (Chomsky Hierarchy)

Type		Productions
0	(phrase-structure)	no restriction
1	(context-sensitive)	$\alpha \rightarrow \beta$ where $\lvert\alpha\rvert \leq \lvert\beta\rvert$
2	(context-free)	$A \rightarrow \beta$
3	(regular)	$A \rightarrow aB$ or $A \rightarrow a$

Note. $A, B \in N$, $\alpha, \beta \in (N \cup T)^*$



BNF (Backus-Naur Form)
- context-free grammar

Ex. 13 Identifier in Algol 60

<id> ::= <letter> | <id> <letter> | <id> <digit>
<letter> ::= a | b | c | ... | z
<digit> ::= 0 | 1 | 2 | ... | 9

12.3 Finite State Automata (FSA)

Finite State Automaton $M = (S, I, f, s_0, F)$

- S: finite set of states
- I: input alphabet
- f: transition function $(f: S \times I \to S)$
- s_0: start state
- F: final (accepting) states $(F \subseteq S)$

Note. There are two ways to define FSA.

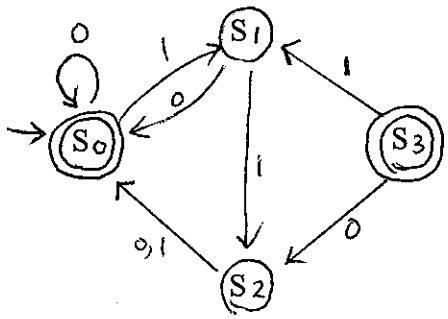
Ex. 4

(1) $S = \{s_0, s_1, s_2, s_3\}$
$I = \{0, 1\}$
$F = \{s_0, s_3\}$
f:

state	Next State 0	1
So	So	S1
S1	So	S2
S2	So	So
S3	S2	S1

Transition Table

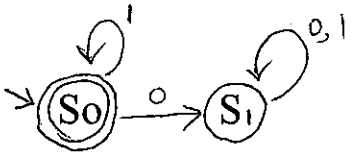
(2)



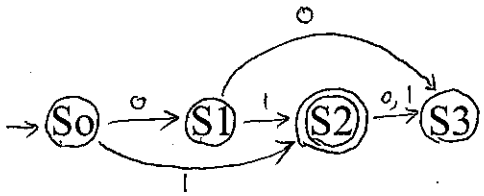
State Diagram (Transition Graph)

Language Recognition

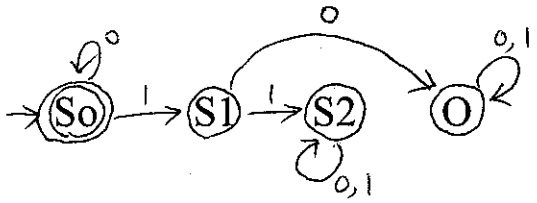
Ex. 5

M1 

$\{\lambda, 1, 11, 111, \ldots\}$ or $\{1^n \mid n = 0, 1, 2, \ldots\}$

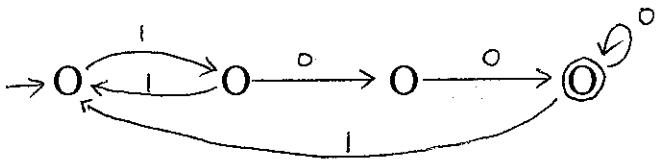
M2 

$\{1, 01\}$

M3 

$\{0^n + 0^n 10(0+1)^*\}$

Ex. 7 odd number of 1's and end with at least two consecutive 0's



Note. FSA recognizes regular language (Type 3).
PDA context-free (Type 2)
LBA context-sensitive (Type 1)
TM phrase-structure (Type 0)

Ex. Even-parity checker