Discrete Maths S.Kalaivani ② ①
Final Maths
Unit - III

FORMAL LANGUAGES AND AUTOMATA

Alphabet. A finite non-empty set Σ of symbols called an alphabet.

Letter: An elt of an alphabet is called letter, character or symbol. A word or string over Σ is a finite sequence.

Language. The set of words over Σ is called a language.

Ex: 1 Let $\Sigma = \{a, b, c\}$ be an alphabet. The sequences abb, c, abc, acbd, bcb are all words over Σ.

We use the symbol λ to denote the empty word.

Consider a collection.

$Q = \{a, b, c, abb, \lambda\}$ is a language over Σ with 5 words

Ex: 2

For $\Sigma = \{0, 1\}$ $Q = \{000, 010, 001, 100, 001, 101, 0010\}$ is a language over Σ with finite words and,

$R = \{001, 000, 100, 110, 101, 1001, \ldots\}$ is a language over Σ with infinite no of words.

Note. The length of a word (or) a string is the number of letters in the word.

Ex $|abc| = 3$, $|bcabab| = 6$.

Concatenation. (Joining two strings)

Let $\Sigma = \{a_1, a_2, \ldots, a_n, b_1, b_2, \ldots, b_m\}$ be an alphabet and $x = a_1a_2 \ldots a_n$ $y = b_1b_2 \ldots b_m$ be any two words.

The concatenation of x & y is defined by

Note

1 The concatenation of empty word with any word x is x itself

i.e. $\lambda x = x\lambda = x$

2 If z is the concatenation of x & y

i.e. $z = xy$. then x is called a prefix of z and

y is called suffix of z

3 $x^n = xx \ldots xx$ (n times)

$(aba)^3 =$ abaabaaba (3 times aba)

$(ab)^5 =$ ababababab (5 times ab

4. The set of all finite words over Σ is denoted by Σ^*

5 The concatenation (on) set of all words including the empty word over an alphabet Σ is denoted by Σ^+

clearly $\Sigma^* = \Sigma^+ \cup \{\lambda\}$.

6. The words $x = a_1a_2 \ldots a_m$ $y = b_1b_2 \ldots b_n$ are said to be equal iff $m=n$ for each i.

$a_i = b_i \ (i = 1, 2, \ldots, n)$

7. If x, y, z be any three words. Then

(i) $x(yz) = (xy)z$ (Associative

(ii) $xy \neq yx$ (not commutative.

<u>Syntax</u>

The specification of the proper construction sentences is called the syntax of the language.

<u>Semantics</u> The specification of the meaning of the sentences is called the semantics of the language.

Free-monoid

The set of strings Σ^* over any alphabet. Σ is a mon under the binary operation concatenation called free monoid over Σ.

The Null string λ is the identity element of this free monoid.

Grammar :

Defn. A phrase structure grammar or simply a grammar G consists of four parts. V_N, V_T, S and P.

where (i) V_N is a finite set, also called vocabulary, whose elements are called variables

(ii) V_T is a finite subset of V_N whose elements are called terminals

(iii) $V_N \cap V_T = \phi$.

(iv) S is a special variable in the set V, called the start symbol that begins the generation of any sentence in the language.

(v) P is a finite set whose elements are ordered pair (α, β) usually written as $\alpha \rightarrow \beta$. Elements of P are called productions or production rules.

∴ A grammar is denoted by $G = (V, \Sigma, S, P)$

otherwise $G = (V_N, V_T, S, P)$

Note : In general an elements (α, β) is written as $\alpha \rightarrow \beta$ (α tending to β) is called a "Production Rule" (or) a Rewriting Rule.

Definition and meaning

1. $V_T = \{a, b, c, \ldots x, y, z, 0, 1, 2, \ldots 9, \ldots\}$

(Small letters - set of terminals) -

Terminal symbols are used to makeup then sentences in the languages

Ex:

The set {a, dog, cat, tree, rose}.

2. $V_N = \{A, B, C, \ldots X, Y, Z\}$ (capital letters - set of non-terminals)

The Non-terminal symbols are intermediate symbols which are used to describe the structure of the sentences.

Ex:

The set {SENTENCE, NOUN, ARTICLE, WORD, ...}

3. P: set of productions:

The productions are grammatical rules that specified how sentences in the formal languages can be made.

A Production is of the form $\alpha \to \beta$ where $\alpha \in (V_N \cup V_T)^*$ and $\beta \in (V_N \cup V_T)^*$.

i.e. α must include atleast one Non-terminal whereas β can consist of any combination of terminals or non-terminals.

A production specifies that string α can be transformed into string β.

4. S: The starting symbol is a special non-terminal that begins the generation of any sentences in the language.

SENTENCE is the starting symbol for $V_N = \{I, LD\}$.

$S = \{I\}$ is the starting symbol

Note

If G is a phrase-structure Grammar, L(G) is the set of strings that can be obtained by starting with S and applying the production rules, a finite no of times until no non-terminal characters remain.

Direct Derivation and Directly Derivable

Let $G = \{V_N, V_T, S, P\}$ be a grammar.

If $\alpha \to \beta$ is a production and $x, y \in (V_N \cup V_T)^*$.

we say that $x\beta y$ is the direct-derivation of $x\alpha y$ (or) $x\beta y$ is directly derivable from $x\alpha y$ (or) $x\alpha y$ directly derives $x\beta y$ in G, and we write

$$x\alpha y \Rightarrow x\beta y \quad \text{if } \alpha_i \in (V_N \cup V_T)^+ \text{ and } \alpha_i \Rightarrow \alpha_{i+1} \text{ for } i = 1, 2, 3, \ldots r.$$

we say that α_i derives α_r in G and write

$$\alpha_1 \Rightarrow \alpha_2 \Rightarrow \cdots \Rightarrow \alpha_{r-1} \Rightarrow \alpha_r \to (1).$$

Thus, α_1 derives α_r in G if α_r is producable from α_1 by employing finite no. of productions from G.

An expression of the form (1) is called a Derivation in G and it can be abbreviated into the expression.

$$\alpha_1 \overset{*}{\Rightarrow} \alpha_r$$ (~~[illegible]~~)

Since, note that the relation $\overset{*}{\Rightarrow}$ is the transitive closure of the relation $\Rightarrow$.

Sentential Form:

A sentential form is any derivative of the unique non-terminals.

Language

The language L generated by a Grammar G is the set of all sentential forms whose symbols are terminals.

i.e. $L(G) = \{\sigma / S \overset{*}{\Rightarrow} \sigma$ and $\sigma \in V_T^*\}$

Ex: 1

(The language $L(G_3) = \{a^n b^n c^n / n \geq 1\}$ is generated by the grammar)

$G_3 = < \{S, B, C\}, \{a, b, c\}, S, \phi >$ where ϕ consists of the productions.

$S \rightarrow aSBC$

$S \rightarrow aBC$

$CB \rightarrow BC$

$aB \rightarrow ab$

$bB \rightarrow bb$

$bC \rightarrow bc$

$cC \rightarrow cc$

The following is a derivation for the string $a^2b^2c^2$

$S \Rightarrow aSBC$

$\Rightarrow aaBCBC$

$\Rightarrow aaBBCC$

$\Rightarrow aabBCC$

$\Rightarrow aabbCC$

$\Rightarrow aabbcC$

$\Rightarrow aabbcc$

In general $L(G_3) = \{a^n b^n c^n / n \geq 1\}$

Problems

1. The language $L(G_4) = \{a^n b a^n / n \geq 1\}$ is generated by the grammar $G_4 = \langle \{S, C\}, \{a, b\}, S, \phi \rangle$ where ϕ is the set of productions.

Proof

$S \to aCa$

$C \to aCa$

$C \to b$.

A derivation for a^2ba^2 consists of the following steps.

$S \Rightarrow aCa$

$\Rightarrow aaCaa$

$\Rightarrow aabaa$

$\Rightarrow a^2ba^2$

In general $L(G) = \{a^n b a^n / n \geq 1\}$.

2. The Language $L(G_5) = \{a^n b a^m / n, m \geq 1\}$ is generated by the grammar.

$G_5 = \langle \{S, A, B, C\}, \{a, b\}, S, \phi \rangle$

where the set of production is,

$S \to aS$

$S \to aB$

$B \to bC$.

$C \to aC$

$C \to a$.

The sentence a^2ba^3 has the following derivation.

$S \Rightarrow aS$

$\Rightarrow aaB$

$\Rightarrow aabC$

$\Rightarrow aabaC$

8

3. Let $G_0 = (\{E, T, F\}, \{a, +, *, (,)\}, P, E)$

where P consists of the productions

$E \rightarrow E + T$

$E \rightarrow T$

$T \rightarrow T * F$

$T \rightarrow F$

$F \rightarrow (E)$

$F \rightarrow a$

Where the variables E, T and F represent the names "expression", "term" and "factor" commonly used in conjunction with arithmetic expressions.

A derivation for the expression $a * a + a$ is

$$
\begin{aligned}
E &\Rightarrow E + T \\
&\Rightarrow T + T \\
&\Rightarrow T * F + T \\
&\Rightarrow F * F + T \\
&\Rightarrow a * F + T \\
&\Rightarrow a * a + T \\
&\Rightarrow a * a + F \\
&\Rightarrow a * a + a.
\end{aligned}
$$

4. Construct the grammar for the language,

$L(G) = \{aaaa, aabb, bbaa, bbbb\}$.

Soln: Since L(G) has a finite number of string.

We can simply list all strings in the language.

Let $V_T = \{a, b\}$. $V_N = \{S\}$ and S be the starting symbol,

P be the set of productions.

i.e. $G = \{V_N, V_T, S, P\}$ is a Grammar with the set of productions. $S \rightarrow aaaa$, $S \rightarrow aabb$, $S \rightarrow bbaa$, $S \rightarrow bbbb$.

construct the grammar for the language L(G) or

$L(G) = \{a^i b^{2i} / i \geq 1\}$

<u>soln</u> :

Let $V_T = \{a, b\}$, $V_N = \{S\}$ and S be the starting symbol and P be the set of productions given by

$S \rightarrow aSbb$, $S \rightarrow abb$.

For ex, if $i = 3$, then we obtain the string

aaa bbbbbb $= a^3 b^6$ as follows.

$S \Rightarrow aSbb \Rightarrow aaSbbbb$

$\Rightarrow aaabbbbbb$

$\Rightarrow a^3 b^6$.

$\therefore G = (\{a, b\}, \{S\}, S, P)$ is the grammar for the given language.

<u>sol</u> : The language L(G) generated by phrase-structure grammar (PSG) is called phrase-structure language (PSL).

<u>Types of Grammars</u> :

Let $V_T = \{a, b\}$ where a, b are arbitrary.

$V_N = \{A, B\}$ where A, B are arbitrary.

and (α, β) are arbitrary strings of terminals and non-terminals

(i) Type-0 grammar :

A phrase-structure Grammar with no restrictions is called a Type-0 Grammar.

~~A language that can~~ be defined a Type-0 Grammar is called "Type-0 language".

(iii) Type-1 Grammar (or) Context sensitive Grammar (CSG)

* A grammar G is context-sensitive or type 1 grammar if each production $\alpha \to \beta$ in G satisfies the condition $|\alpha| \leq |\beta|$ i.e. the length of α is less than or equal to the length of β

i.e. $\alpha \to \beta$ with $|\alpha| \leq |\beta|$

* The restriction on a productions of a context-sensitive grammar can be equivalently stated as follows

α and β in the production $\alpha \to \beta$ can be expressed as $\alpha = \phi_1 A \phi_2$ and $\beta = \phi_1 \psi \phi_2$ (ϕ_1 and/or ϕ_2 are possibly empty)

Where ψ must be non-empty.

* The meaning of "context-sensitive becomes clearer with this reformulation.

* The application of the production $\phi_1 A \phi_2 \to \phi_1 \psi \phi_2$ to a sentential form means that A is rewritten as ψ in the context ϕ_1 and ϕ_2.

* Context-sensitive grammars are said to generate context-sensitive languages.

Ex: 1

$A \to ab$, $A \to aA$, $aAb \to aBcb$

Ex: 2

U.Q

$G_3 = \langle \{S, B, C\}, \{a, b, c\}, S, \phi \rangle$

where ϕ consists of productions.

$S \to aSBC$

$S \to aBC$

$CB \to BC$

$aB \to ab$

$bB \to bb$

$bC \to bc$

$cC \to cc$

(III) Type - 2 Grammar (or) Context free Grammar (CFG)

* A grammar G is context free or type 2 grammar if each production $\alpha \to \beta$ in G satisfies the condition $\alpha \in V_N$ and $|\alpha| \leq |\beta|$

* i, In this Grammar every production is of the form $A \to \alpha$. Inotherwords in any production the left hand string is always a single - non-terminal (capital letter

Ex : The language $L(G) = \{a^k b^k / k \geq 1\}$ is a type - 2 language because it can specified by the type - 2 Grammar.

$A \to aAB$, $A \to ab$, $B \to b$

i, $A \Rightarrow aAB$ (for ex. a^2b^2, $k=2$

$\Rightarrow aabB$

$\Rightarrow aabb$

$\Rightarrow a^2b^2$.

(iv) Type - 3 Grammar (or) Regular Grammar (RG) (or) Right Linear Grammar

* A grammar G is called a regular or type 3 grammar if each production $\alpha \to \beta$ satisfies the condition α is a single non-terminal symbol. i, $\alpha \in V_N$ and β is the single terminal or a terminal followed by a non-terminal.

i, $\beta = aB$ where $a \in V_T$ and $B \in V_N$.

* i, A Grammar is said to be a type - 3 Grammar if all productions in the Grammar are of the forms $A \to a$, $A \to$ Inother words, in any production the lefthand string is always a single non-terminal ~~and the righthand~~ string is either a terminal (or) Non-terminal followed by a terminal.

7. Ex: Construct the Grammar for the language $L(G) = \{a^n b a^m / n, m \geq 1\}$

Soln

Let $V_N = \{S, A, B, C\}$, $V_T = \{a, b\}$ and S be the starting symbol. P be the set of productions given by $S \to aS$, $S \to aB$, $B \to bC$, $C \to aC$, $C \to a$ for a^2ba^3

$$S \Rightarrow aS \Rightarrow aaB \Rightarrow aabC$$
$$\Rightarrow aabaC$$
$$\Rightarrow aabaaC$$
$$\Rightarrow aabaaa$$
$$\Rightarrow a^2ba^3.$$

clearly this Grammar $G = (V_N, V_T, S, \phi)$ is a regular grammar and the language $L(G) = \{a^n b a^m / n, m \geq 1\}$ is a regular language.

Note:

1. Corresponding to different types of Grammars, there are different types of languages. Thus a language is said to be a type-i ($i = 0, 1, 2, 3$) language if it can be generated a type-i Grammar but it cannot be specified by a type $(i+1)$ Grammar.

2. A type-3 Grammar $\subseteq$ type-2 Grammar $\subseteq$ type-1 Grammar $\subseteq$ type-0 Grammar. (or)

type-3 => type-2 => type-1 => type-0 Grammar.

3. A CFL may or may not contain λ. If a CFL does not contain λ then a CFG can be found such that there is no rule of the form $\alpha \to \lambda$. such a CFL is called "a λ-free CFL" (OR) "a pure CFL". On the other hand, if a CFL contains λ then CFG can be found such that the only rule containing λ is $S \to \lambda$.

Q. Finite State Language (or) Regular Set

imp Q. The set of all strings x accepted by M is denoted by

$$T(M) = \{x / f(s_0, x) \in S_f, x \in I^*\}$$

It is called the language accepted or recognized by M and this set of strings x is called a "finite state language" (or) "a regular set".

State Diagram (or) Transition Diagram:

The pictorial method of specifying the finite state machine is called "State Diagram" (or) "Transition Diagram".

Note:

(i) digraph – graph with correct direction.

(ii) $s_1, s_2, \ldots, s_n = S_f \subseteq S$ ($\because s_1 \subseteq S, s_2 \subseteq S, \ldots s_n \subseteq S \Rightarrow S_f \subseteq S$)

Formal Defn.:

Let $M = (S, I, f, S_f, s_0)$ be a finite state automata. (vertices number; edges s_i - s_j; initial state)

The state diagram of M is a digraph G whose vertices are the number of S. An arrow designates the initial state s_0.

A directed edge (s_1, s_2) exists in G if there exists an input i with $f(s_1, i) = s_2$.

Accepting i with $f(s_1, i) = s_2$.

Accepting states (Final states) are marked by double circle.

Ex: 3

Notation and meaning:

1. The meaning of $f(s_0, a) = s_1$ is, when M is in state s_0, it can move one cell to right and goes the state s_1. reading the input symbol a,

2. The meaning of $f(s_0, \wedge) = s_0$ is that when no input symbol read, the machine does not change its state

3. The interpretation of $f(s_i, x) = s_j$ is that the machine starting from state s_i, read the strings x on the input tape from left to right and reaches the state s_j.

Language Accepted (or) Recognised by the automata:

A word (or string or sentence or tape) is said to be accepted if the automata starts from the initial state and enters a final state after reading the word, one letter at a time from left to right. The set of strings accepted by the automata is called the "language accepted" (or) "recognised by the automata".

Defn

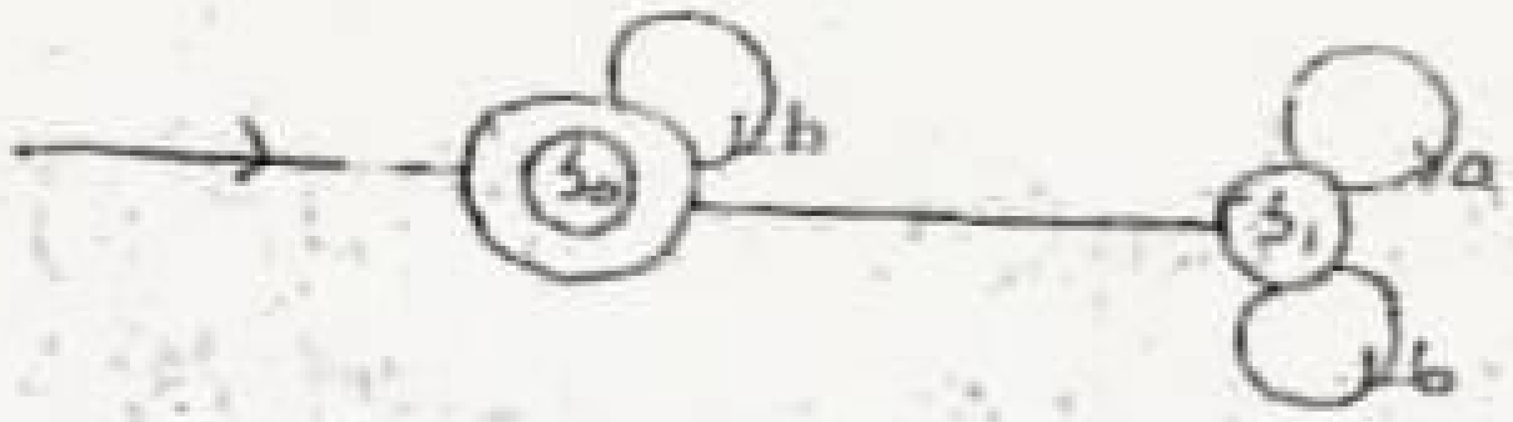
Let $M = \langle S, I, f, s_i, s_0 \rangle$ be a finite state automata. A non-null string $a = x_1 x_2 \ldots x_n$ is said to be accepted by M if

(i) s_0 is the initial state,

(ii) $f(s_{i-1}, x) = s_i \quad i = 1, 2, \ldots, n$.

(iii) s_n be an accepted state.

Here the S_0 is the initial state and the only accepting state. The final state automata is defined by



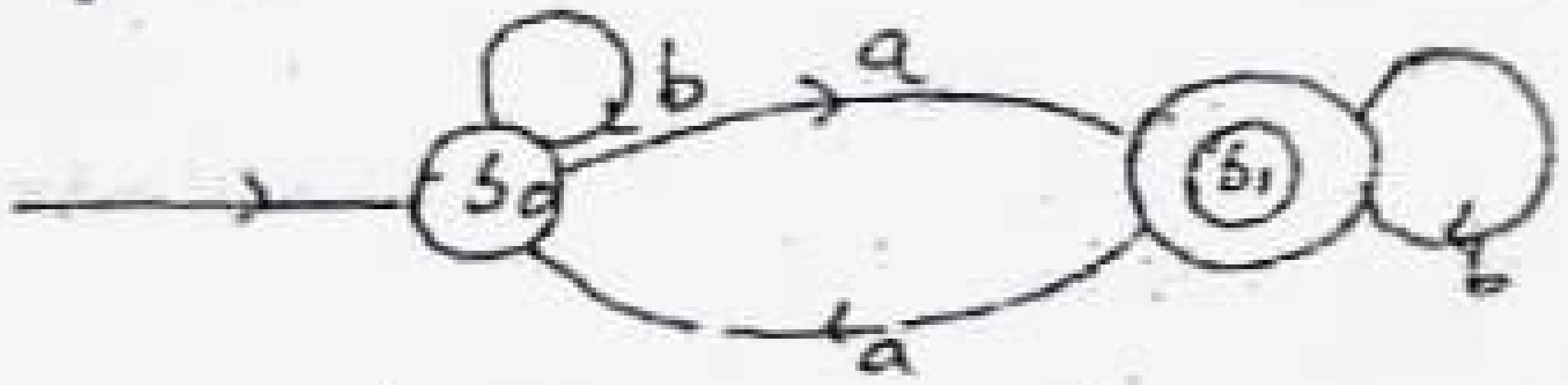
Ex 4

Design a finite state automata that accepts precisely those strings over $\{a, b\}$ that contain an odd no. of a's

Soln:

Let us assume that the state

Here the S_0 is the initial state and S_1 is the accepting state. We obtain the transition diagram as



Defn (Machine congruence):

Let $M = (S, I, f, S_i, S_0)$ be a finite state automat. We say suppose that R is an equivalence relation on S. We say that R is an machine congruence on M if for any $s, t \in S$.

$$sRt \Rightarrow f(x, s) \; R \; f(t, s) \quad (or)$$

$$g_x(s) = R \quad \overline{g_f(s) \text{ for all } x \in I}$$

Note: We know that the set of all finite strings I^* of the alphabet I is a free monoid under the binary operation concatenation with λ as its identity. Also, S^S is a monoid, which consists of all functions from S to S and which has the function composition as the identity in S^S is therefore I_S

as the binary operation. the identity in S^S is

I_S denoted by $I_S(s) = s$ for all s in S.

State Transition Function corresponding to W

If $W = x_1, x_2, \ldots, x_n \in I^*$, we let

$f_W = f_{x_n} \circ f_{x_{n-1}} \circ \ldots \circ f_{x_2} \circ f_{x_1}$; the composition

of the function $f_{x_n}, f_{x_{n-1}}, \ldots, f_{x_1}$.

Also, define $f_\Lambda(s) = I_S(s)$, for all s in S.

In this way we assign an element f_W of S^S to each element of I^*.

If we think of each f_x as the "effect" of the input x on the states of the machine M, then f_W represents the combined effect of all the input letters in the word W received in the sequence specified by W.

We call f_W, "the transition function corresponding to W".

Ex: 5 Let $M = (S, I, F, S_i, S_0)$ be a finite state machine where $S = \{S_0, S_1, S_2\}$. $I = \{0, 1\}$ and f is given by the following state transition table.

State	Inputs 0	Inputs 1
S_0	S_0	S_1
S_1	S_2	S_2
S_2	S_1	S_0

Soln:

$f(0, s_0) = f_0(s_0) = s_0$

$f(1, s_0) = f_1(s_0) = s_1$

$f(0, s_1) = f_0(s_1) = s_2$

$f(1, s_1) = f_1(s_1) = s_2$

$f(0, s_2) = f_0(s_2) = s_1$

$f(1, s_2) = f_1(s_2) = s_0$

Let $W = 011 \in I^*$ then

$$f_W(s_0) = f_{011}(s_0) = (f_1 \circ f_1 \circ f_0)(s_0)$$
$$= (f_1 \circ f_1 \circ f_0(s_0))$$
$$= f_1 \circ f_1 \circ (s_0)$$
$$= f_1 \circ (s_1)$$
$$= s_2.$$

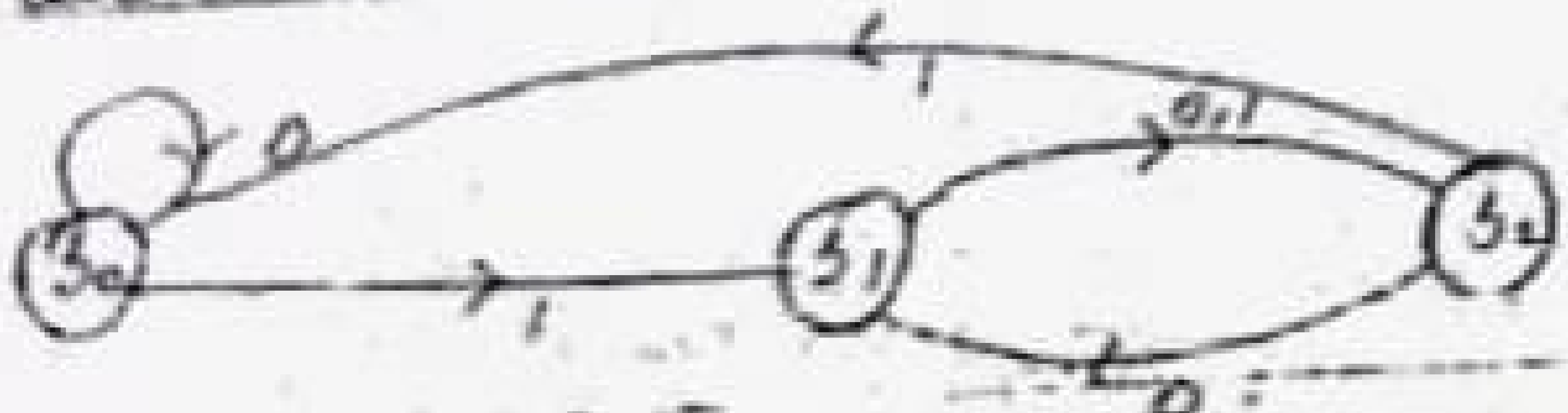
III^ly,

$$f_W(s_1) = (f_1 \circ f_1 \circ f_0(s_1))$$
$$= (f_1 \circ f_1 \circ (s_2))$$
$$= f_1 \circ (s_0)$$
$$= s_1$$

$$f_W(s_2) = f_{011}(s_2) = f_1 \circ f_1 \circ f_0(s_2).$$
$$= f_1 \circ f_1 \circ (s_1)$$
$$= f_1 \circ (s_2)$$
$$= s_0.$$

Ex: 6
UQ Let $M(S, I, f, S_C, s_0)$ be a finite state machine whose transition or state diagram is given by

0101



Scanned by TapScanner

(18)

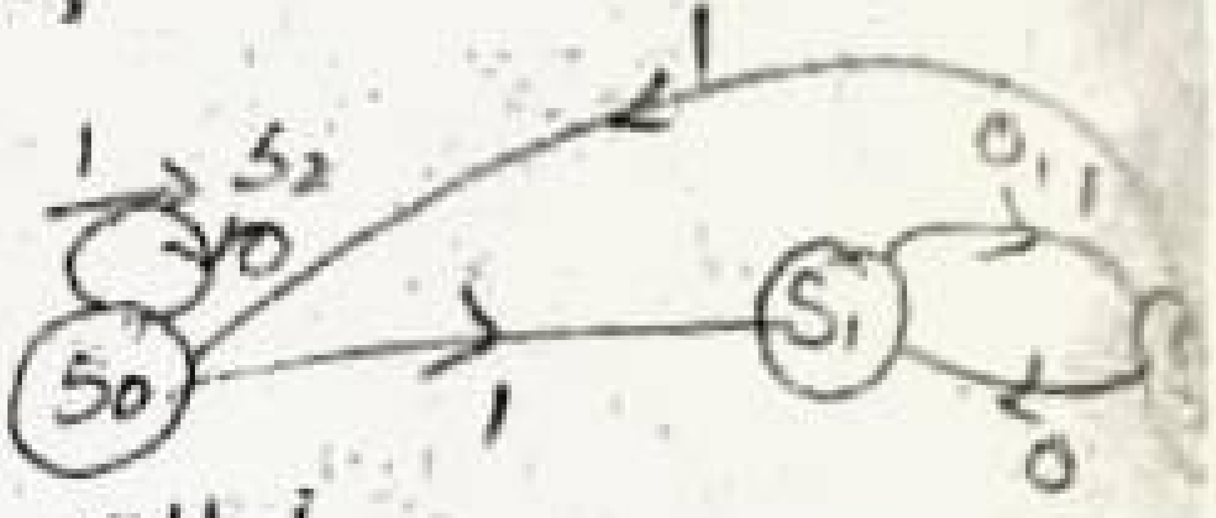
Solu : Let us compute f_w when $w = 01011$

The successive transition of s_0 are

$s_0 \xrightarrow{0} s_0 \xrightarrow{1} s_1 \xrightarrow{0} s_2 \xrightarrow{1} s_0 \xrightarrow{1} s_1$

$\therefore f_w(s_0) = f_{01011}(s_0) = s_1$

Similarly, the transition diagram of s_1 are



$s_1 \xrightarrow{0} s_2 \xrightarrow{1} s_0 \xrightarrow{0} s_0 \xrightarrow{1} s_1 \xrightarrow{1} s_2$

$\therefore f_w(s_1) = s_2$.

The successive transition of s_2 are :

$s_2 \xrightarrow{0} s_1 \xrightarrow{1} s_2 \xrightarrow{0} s_1 \xrightarrow{1} s_2 \xrightarrow{1} s_0$

$f_w(s_1) = s_2$ & $f_w(s_2) = s_0$.

Defn Monoid of the machine M :

Let $M = (S, I, f, s_i, s_0)$ be a FSM.

We define a homomorphism $T : I^* \to S^S$

As I^* and S^S are monoids.

$T(I^*) = K$ is a submonoid of S^S.

This Monoid $T(I^*) = K$ is called the monoid of the machine M.

Note :

1. Every Finite state machine has a monoid associated with it. For any FSM, the element of its associated monoid corresponds to certain input sequences. Because only a finite no of combination of states and inputs is possible for a FSM there is only a finite no of input sequence that summarize the machine.

Q, Any Finite Monoid $(M, *)$ can be represented in the form of a FSM with input and state sets equal to M machine of this type are called "state machine"

Defn

If $(M, *)$ is a finite monoid, then the machine of the monoid $(M, *)$ denoted by $m(M)$ is the state machine with state set M input set M and next state function,

$t : M \times M \to M$ defined by $t(s, x) = s * x$.

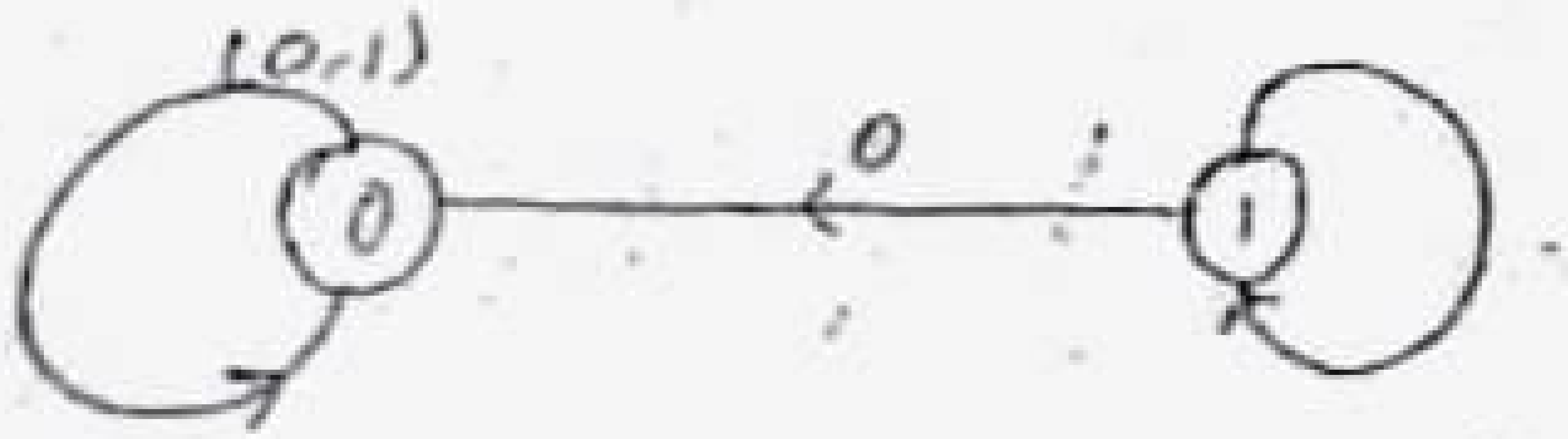
Ex: 7

Let (Z_2, X_2) be a monoid. The machine of this monoid Z_2 denoted by $M(Z_2)$ is the state machine with state set $\{0, 1\}$ input set $\{0, 1\}$ and next state function.

$t : Z_2 \times Z_2 \to Z_2$ defined by $t(s, x) = s \times_2 x$

Soln:

The transition diagram of the machine $M(Z_2)$ is



Non-Deterministic Finite state Automata: (NFA)

A generalization DFA called a non-determini finite state automata (NFA) which despite its super natural behaviour also reputation for pattern recogn prove to be a valuable tool for

A NFA is usually much easier to design than the corresponding DFA. In NFA, one state is distinguished as an initial state, one or more are distinguished as accepting states and there are labelled, directed edges connecting the states.

A string is accepted by the NFA iff there is a path from the initial state to one to the accepting states, such that the edges label on the path generate the string.

Defn :

A Non-Deterministic Finite state Automata : (NFA) M is, $M = (S, I, A, f, s_0)$ where

(i) S : a finite non-empty sets of states.

(ii) I : a finite non-empty sets of input symbol.

(iii) A : a subset of S of accepting states.

(iv) f : a Mapping from $S \times I$ to the finite subset of S

(v) s_0 The initial state and $s_0 \in A$.

Note :

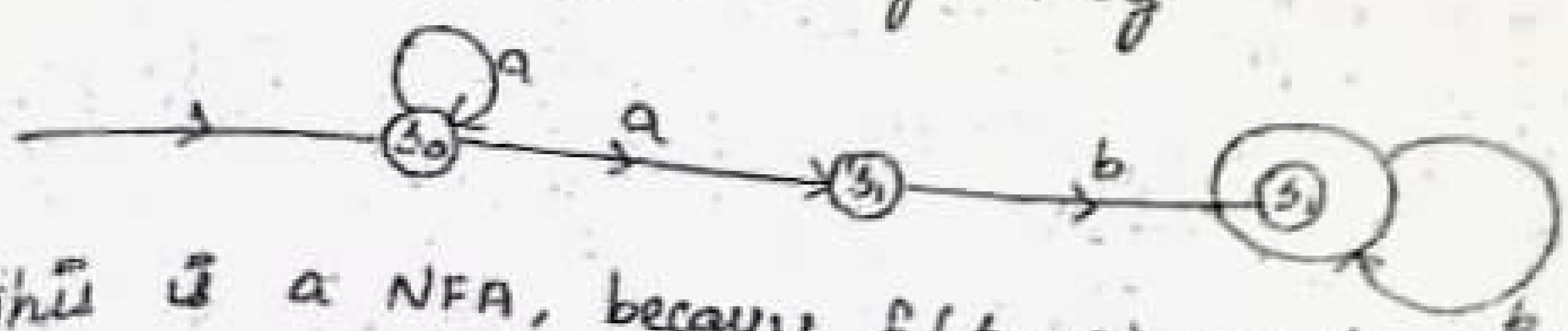
The Main difference between the DFA and NFA is that in the DFA, $f(s_i, a)$ is a single state, while in NFA $f(s_i, a)$ may consists of a sets of state (Possibly empty)

$f(s_i, a) = \{s_1, s_2, \ldots, s_k\}$ means that when M in the state s_i, reading the input "a" on the input tape, it can move to any one of the states $s_1, s_2, \ldots$ as the next state and start reading the input symbol to the input of "a".

Ex - 3

Let $M = (S, I, A, f, S_0)$ be a NFA where $S = \{S_0, S_1, S_2\}$, $I = \{a, b\}$ and S_0 is the initial state. f is defined by. $f(S_0, a) = \{S_0, S_1\}$, $f(S_1, a) = \phi$, $f(S_1, b) = \{S_2\}$; $f(S_2, a) = \phi$; $f(S_2, b) = \{S_2\}$, $f(S_0, b) = \phi$.

Its transition diagram is given by



This is a NFA, because $f(S_0, a)$ can be either S_0 or S_1.

Note :

DFA and NFA represented the type-3 or regular language.

Ex: 9mark

Consider a regular Grammar

$G = \{V_T, V_N, S, P\}$ where $V_N = \{S, A\}$, $V_T = \{a, b\} = I$

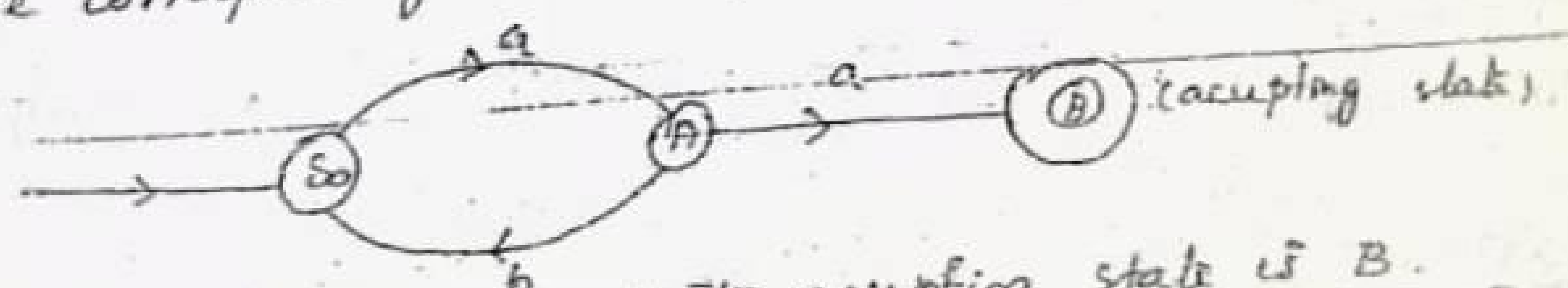
and the set of productions $P = \{S \to aA, A \to aA, A \to bS, A \to a\}$

The corresponding automata is $M = (S, I, A, f, S_0)$

where $S = \{S_0, A, B\}$. f is defined as follows.

$f(S_0, a) = A$, $f(A, a) = (A, B)$, $f(A, b) = S_0$

The corresponding transition diagram is



clearly M is NFA and its accepting state is B.

$L(G) = T(M)$.

Scanned by TapScanner

Note

Two automata M, M' are said to be equal if they accept exactly the same language i.e. if $T(M) = T(M')$.

Ex: 10

(i) Draw the transition diagram of the NFA $M = \{S, I, A, f, S_0\}$ where $I = \{a, b\}$, $S = \{S_0, S_1, S_2\}$ and next state function $A = \{S_1, S_2\}$ with initial state S_0

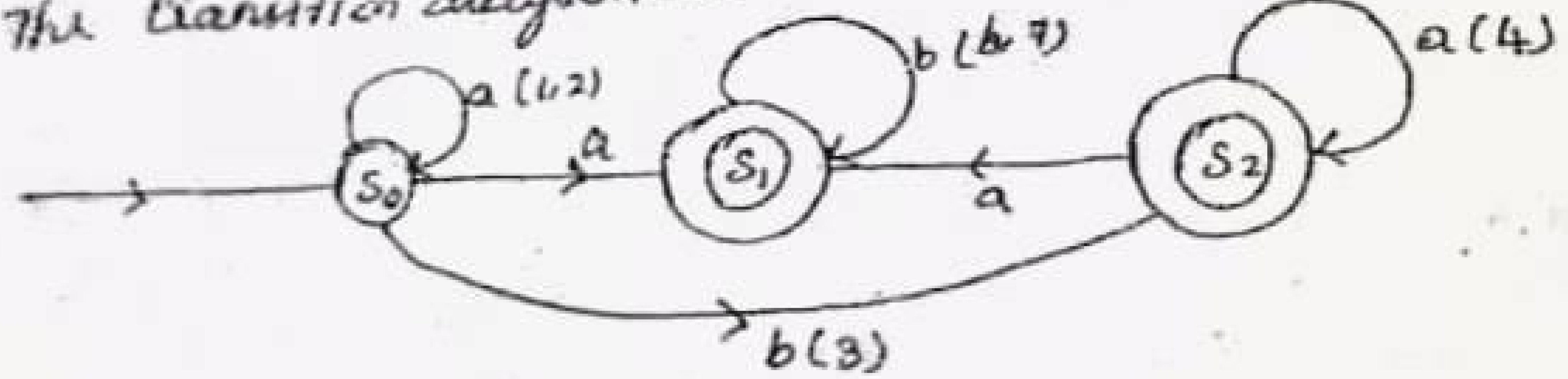
State	Input a	Input b
S_0	$\{S_0, S_1\}$	$\{S_2\}$
S_1	ϕ	$\{S_1\}$
S_2	$\{S_1, S_2\}$	ϕ

(ii) Is the string $\alpha = aabaabb$ accepted by the NFA.

Soln

(i) Here S_0 is the Initial state and S_1, S_2 are accepting states

The transition diagram is



The path $(S_0, S_0, S_0, S_2, S_2, S_1, S_1, S_1)$ represent a string $\alpha = aabaabb$. Since the Final states S_1 an accepting state, the string α is accepted by the above NFA.

Q procedure for converting Non-deterministic finite state automata (NFA) to deterministic finite state automata (DFA)

Given NFA, $M = (S, I, A, f, s_0)$ where

I : set of input symbols

S : a finite set of states

A : a subset of S, consisting the accepting state

s_0 : initial state

f : the next state function.

Construct the corresponding deterministic finite state automata (DFA).

$$M' = (S', I, A', f', s_0')$$

(i) Here the input I is unchanged

(ii) The state S' consisting of all subsets of the original set S. i.e, $S' = (P(S)$ = power set of $S)$.

(iii) The initial state is $s_0' = s_0$.

(iv) The accepting (A') states are all subsets of S that contain an accepting state of the original NFA.

i.e, $A' = \{x \subseteq A \mid x \cap A \neq \phi\}$

(v) The next state function f' is defined by

$$f'(x, n) = \begin{cases} \phi & \text{if } x = \phi \\ \bigcup_{s \in x} f(s, x) & \text{if } x \neq \phi \end{cases}$$

(vi) Draw the transition diagram of M' and by deleting states which can never be reached we can obtain the simplified DFA (M') equivalent to the given NFA.

Theorem:

The above procedure can also stated as "Let L be accepted by a NFA. Then there exists a DFA that accepts L".

Proof:

Sn:10

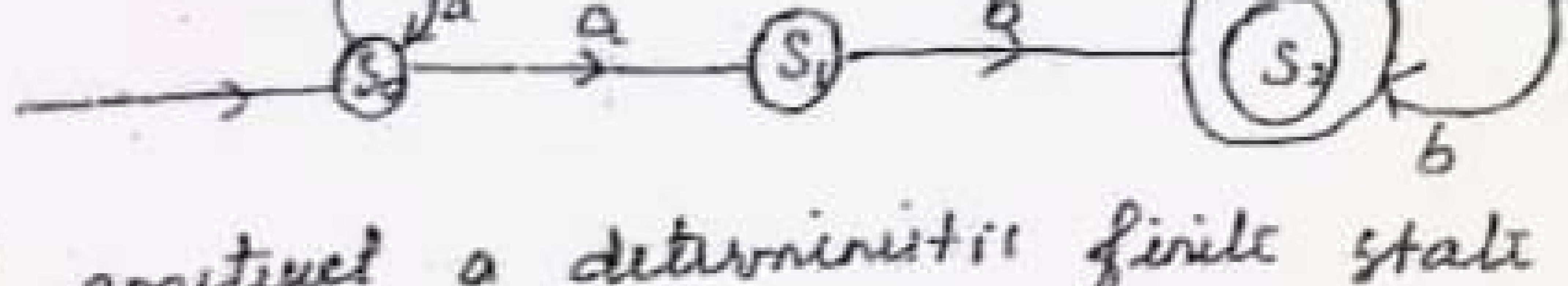
Sn:11

Construct a deterministic finite state automata (DFA) equivalent to a given non-deterministic finite state automata (NFA).

Let $M = (S, I, A, f, s_0)$ is a NFA where $S = \{s_0, s_1, s_2\}$ $I = \{a, b\}$ s_0 is the initial state and s_2 is the accepting state. The next state function f is defined by.

f	a	b
s_0	$\{s_0, s_1\}$	ϕ
s_1	ϕ	$\{s_2\}$
s_2	ϕ	$\{s_2\}$

And its transition diagram is given by



Soln: construct a deterministic finite state automata (DFA)

$M' = (S', I, A', f', s_0')$ where

$S' = \{\{s_0\}, \{s_1\}, \{s_2\}, \{s_0, s_1\}, \{s_0, s_2\}, \{s_1, s_2\},$
$\{s_0, s_1, s_2\}, \phi\}$

= collection of all subsets of S.

$I = \{a, b\}$, $s_0' = \{s_0\}$

$A' = \{\{s_2\}, \{s_0, s_2\}, \{s_1, s_2\}, \{s_0, s_1, s_2\}\}$

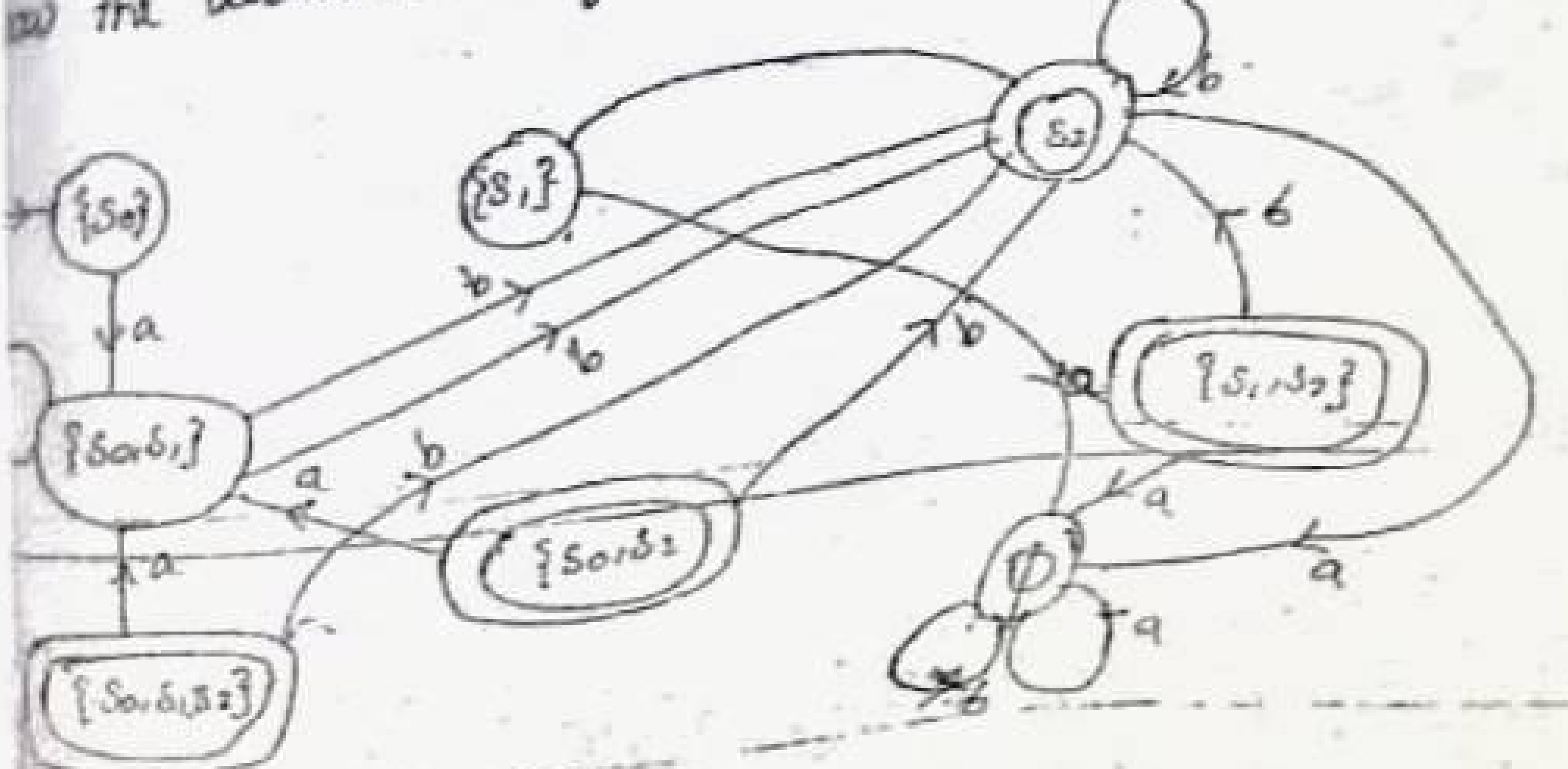
= collection of subsets of A that contains an acc. st. state of the original NFA

The next state function f' is defined by

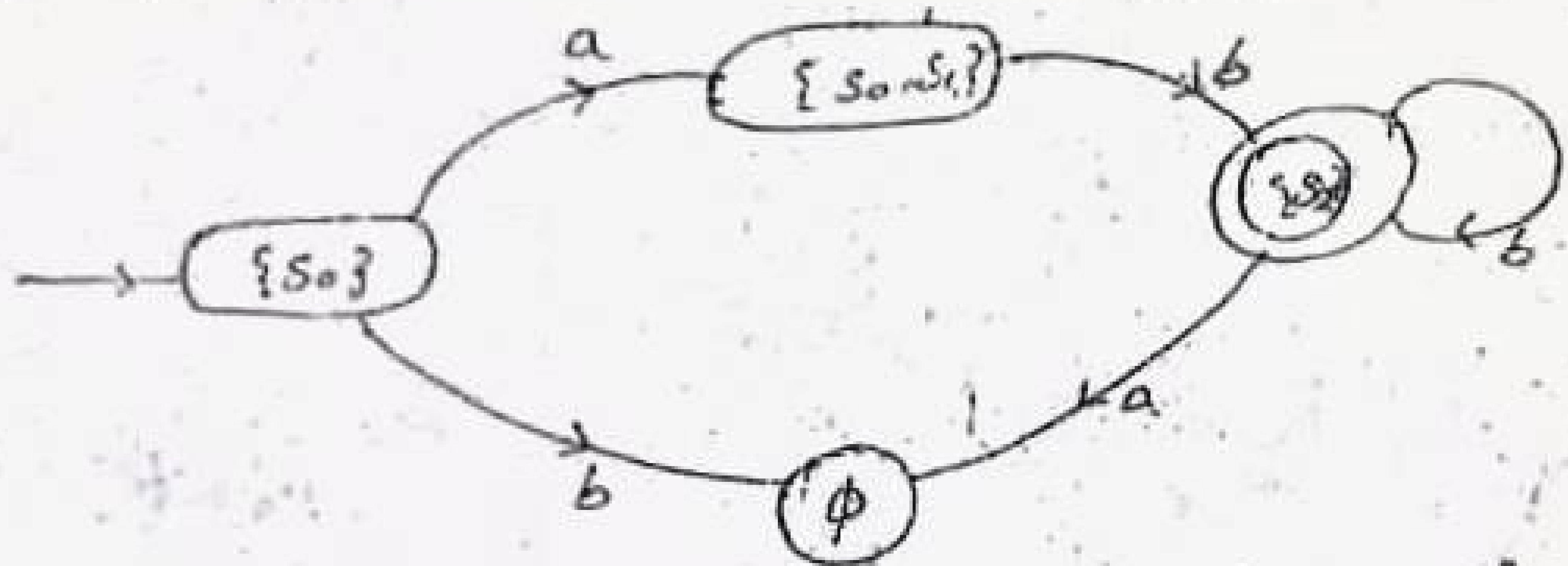
f'	a	b
$\{s_0\}$	$\{s_0, s_1\}$	ϕ
$\{s_1\}$	ϕ	$\{s_2\}$
$\{s_2\}$	ϕ	$\{s_2\}$
$\{s_0, s_1\}$	$\{s_0, s_1\}$	$\{s_2\}$
$\{s_0, s_2\}$	$\{s_0, s_1\}$	$\{s_2\}$
$\{s_1, s_2\}$	ϕ	$\{s_2\}$
$\{s_0, s_1, s_2\}$	$\{s_0, s_1\}$	$\{s_2\}$
ϕ	ϕ	ϕ

$$f(X, x) = \begin{cases} \phi & \text{if } X = \phi \\ \bigcup_{s \in X} f(s, x) & \text{if } X \neq \phi \end{cases}$$

Now the transition diagram is given by,



Here the state $\{S_0, S_2\}$, $\{S_1, S_2\}$, $\{S_0, S_1, S_2\}$, ... can be deleted (except the Initial state) which can never be reached. Then we obtain the simplified equivalent deterministic finite state automata (DFA) M' corresponding to the given non deterministic finite state automata (NFA).



Theorem. 2

Show that every regular set is accepted by a finite state automata M.

Proof

Let $L(G)$ be a regular language generated by a regular Grammar $G = (V_N, V_T, P, S)$. We shall define a finite state automata M to accept $L(G)$.

Assume that there is no rule of the form $A \to B$ where $A, B \in V_N$ in G.

Let $M = (K, I, F, f, S_0)$ where K is a finite non-empty set of states & $F \subseteq K$ is the set of Final states.

Define $K = V_N \cup \{q\}$ $q \notin V_N$, $S_0 = S$.

$$F = \{q\} \cup \{A / A \in V_N \text{ \& } A \to \lambda \text{ is in } P\}$$ and

f as follows.

If $A \to aB$ is a rule in P, $f(A, a)$ contains B.

If $A \to a$ is a rule in P. $f(A, a)$ contains q

Also. G_2 $S \Rightarrow aASb$

$\Rightarrow a\lambda Sb$

$\Rightarrow aSb$

$\Rightarrow a(aSb)b$

$\Rightarrow aaSbb$

$\Rightarrow aa(ab)bb$

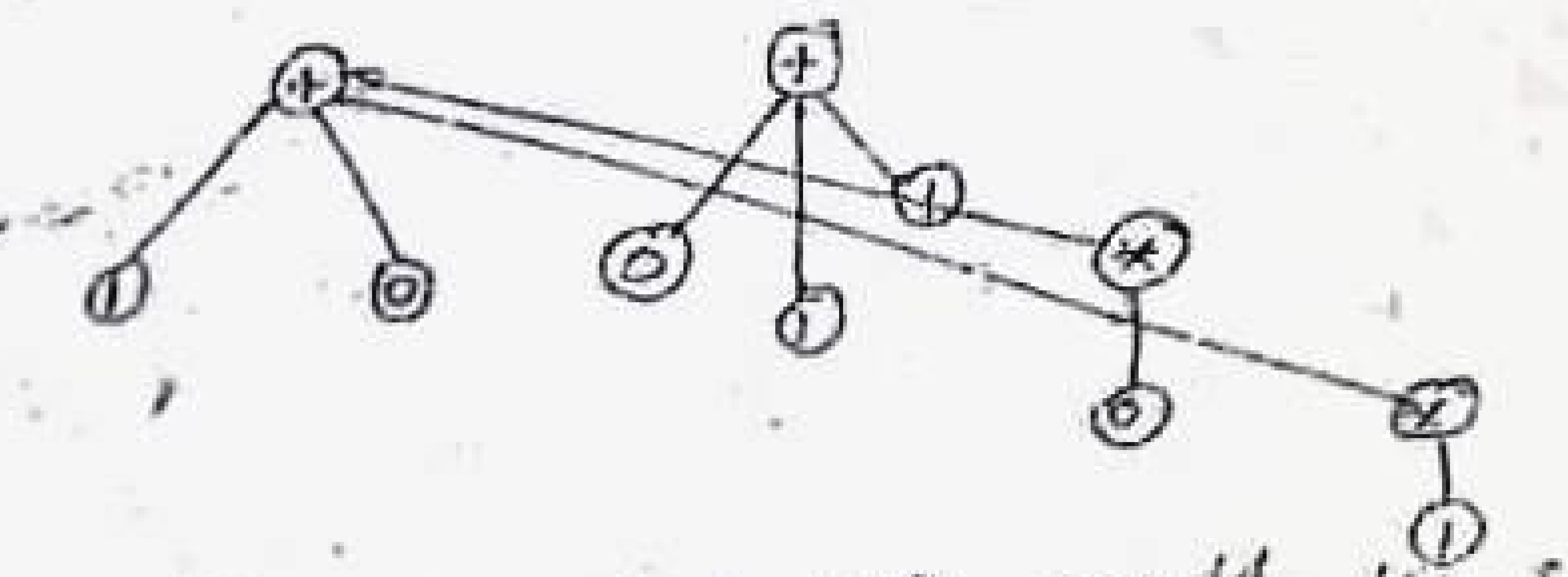
$\Rightarrow aaabbb.$

$\Rightarrow a^3b^3.$

$L(G_1) = L(G_2)$ with $L = \{a^n b^n \mid n \geq 1\}$.

4. Draw the transition Diagram of FSA construct FSA to the following regular expression

$10 + (0+11)0^*1$



5. Design a FSA that will accept the set of natural no. x which are divisible by 3.

Soln

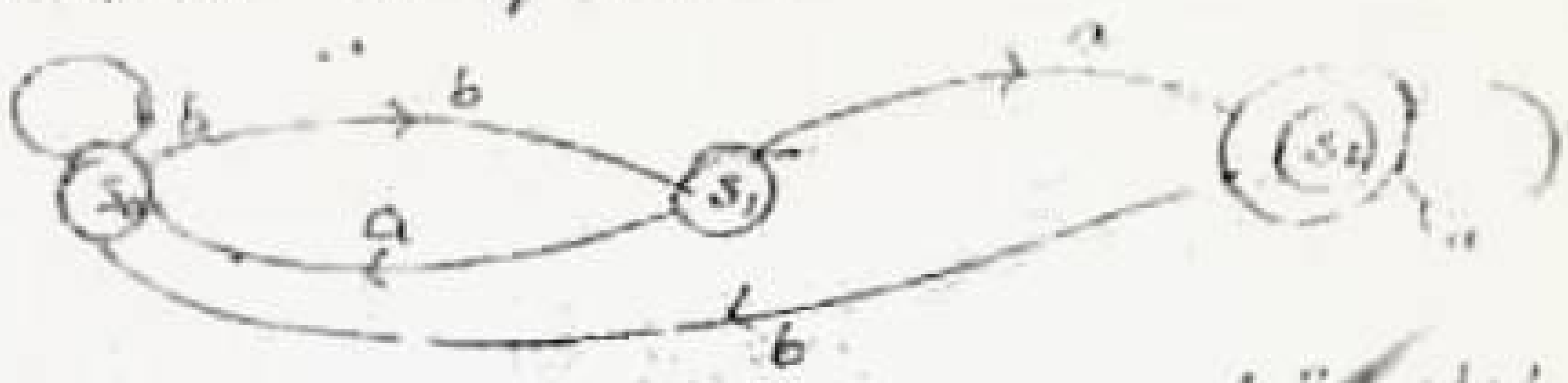
Let $M = (S, I, f, S_f, S_0)$

Let $S = (s_0, s_1, s_2)$

$I = (a, b)$

$s_0 \rightarrow$ Initial state & $s_2 \rightarrow$ Final state

Consider the set of natural no S as



In FSA, the final and the accepting state is s_1

Here, the no. of a's = 3 which is divisible by 3

Hence $n = 3$.

6. NDFSA (i) Defn (ii) Ex with Diagram.

(iii) In NDFSA, $f(S_i, a)$ consist of set of possible sta

and NDFSA is a generalization of DFSA.

(iv) In DFSA, $f(S_i, a)$ a single state.

Theorem

Theorem 3.5.5, 3.5.6

Langrange'. theorem.

Pg. no. 71 (problem)

Theorem - 3.2.8

KalaiSelvi.S

KalaiSelvi.S

KalaiSelvi.S

Sowmiya.C

Sowmiya.C

Selvi.S