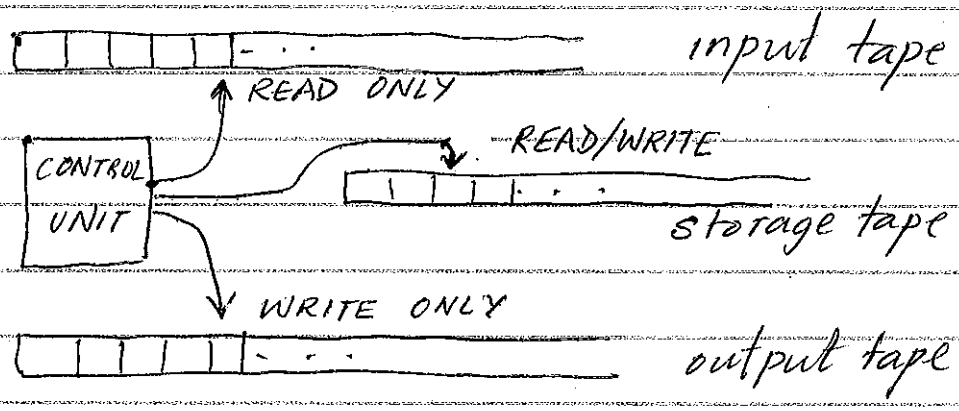
Ch. 3 - Finite State Machines

§1. Deterministic Finite State Acceptors

A Finite State Machine (FSM) is an abstract model of a digital computing machine in which the device can enter into only a finite number of different states. In a finite state machine, there may be input, storage, or output tapes of finite or unlimited capacities. Information can be read from, or written on, these tapes by a control unit (which is also known as the head).



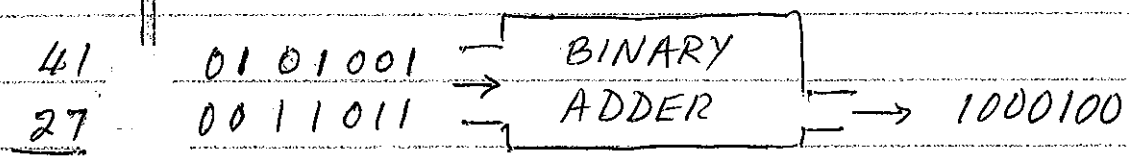
Finite state machines can be classified according to the kinds of tape they have, the kind of access to any storage tape they may have, and the mode of operation.

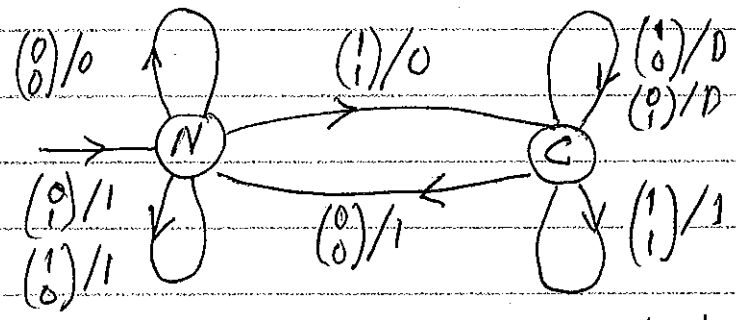
Def. An acceptor is an FSM with an input tape and a YES/NO output tape. (There may or may not be any storage tape.) A yes/no output tape has only one space for a "0" (NO) or a "1" (YES)

Classification of Finite State Machines

TYPE	TAPES, STORAGE PRESENT, MODE OF OPERATION
1(a) DFA Deterministic Finite Acceptor	Infinite input tape, no storage tape, Yes/No output tape, Deterministic mode of operation
1(b) NFA Non-deterministic Finite Acceptor	Infinite input tape, no storage tape, YES/NO output tape, Non-deterministic mode allowed.
2(a) DPDA Deterministic Push Down Acceptor	Infinite input tape, infinite storage tape with stack access only, YES/NO output tape, Deterministic mode of operation.
2(b) PDA Push Down Acceptor	Infinite input tape, infinite storage tape with stack access only, YES/NO output tape, Non-deterministic mode allowed
3. LBM Linear Bounded Machine	Infinite storage tape which also functions as the input & output tape, random access to a portion of the storage tape which is bounded by a constant times the size of input, Non-deterministic mode allowed.
4(a) DTM Deterministic Turing Machine	Infinite storage tape which functions as the input & output tape, random access to the whole of the storage tape, Deterministic mode
4(b) TM Turing Machine	Same as DTM except Non-deterministic mode allowed.

Ex.1 Suppose we want an FSM which does binary addition. How should we design it?





Input : $\binom{0}{0}\binom{0}{1}\binom{1}{0}\binom{1}{1}\binom{0}{0}\binom{0}{1}\binom{1}{1}$ ← start here

Output : 1 0 0 0 1 0 0

state : N C C C N C C N ← starting state

Note: The FSM we got is not an acceptor but its mode of operation is deterministic. So we can call it a DFSM.

Def. A Determinstic Finite Acceptor (DFA) is a 5-tuple $M = \langle Q, T, \delta, q_0, \mathcal{A} \rangle$ where

$Q(M) =$ Q is a finite set of (internal) states

$T(M) =$ T is an alphabet called the input alphabet

$\delta_M =$ $\delta : Q \times T \to Q$ is a function called the transition function

$q_0(M) =$ $q_0 \in Q$ is the designated initial state, and

$\mathcal{A}(M) =$ $\mathcal{A} \subseteq Q$ is the designated set of accepting states.

Ex.1 Let $M_1 = \langle Q, T, \delta, q_0, \mathcal{A} \rangle$ where $Q = \{A, B, C\}$, $T = \{0, 1\}$, $q_0 = A$, $\mathcal{A} = \{B\}$ and $\delta : Q \times T \to Q$ is defined by

$\delta(A,0) = A$ $\quad$ $\delta(A,1) = B$

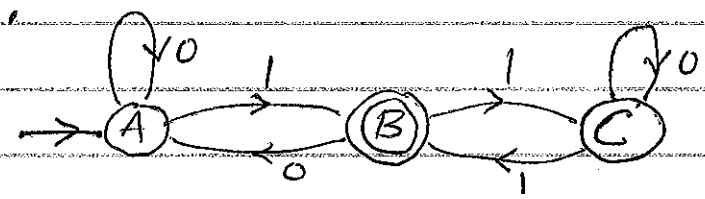
$\delta(B,0) = A$ $\quad$ $\delta(B,1) = C$

$\delta(C,0) = C$ $\quad$ $\delta(C,1) = B$,

We can represent M_1 by the transition table on the right with "$\rightarrow$" designating the initial state & "○" designating accepting states.

Q \ T	0	1
$\rightarrow$ A	A	B
(B)	A	C
C	C	B

We can also represent M_1 by a transition graph with "$\rightarrow$" designating the initial state & ◯ designating the accepting states.



Note:
It is possible for a state to be the initial state as well as an accepting state. If $\delta(B,a) = D$ in M we say that $B \xrightarrow{a} D$ is a transition in M

We can describe the operation of a DFA M_1 by giving the sequence of configurations of M when it is presented with an input $\varphi \in T^*$. A configuration is an ordered pair $\langle q, \varphi_1\varphi_2 \rangle$ where q is the control state and $\varphi_1\varphi_2$ is the input string with the head under the beginning of φ_2.

$\varphi = \varphi_1\varphi_2$

tape: $\overbrace{\boxed{a}\boxed{b}\boxed{a}}^{\varphi_1}\overbrace{\boxed{c}\boxed{b}\boxed{b}\boxed{a}}^{\varphi_2}\boxed{\sqcup}\boxed{\sqcup}\ldots$

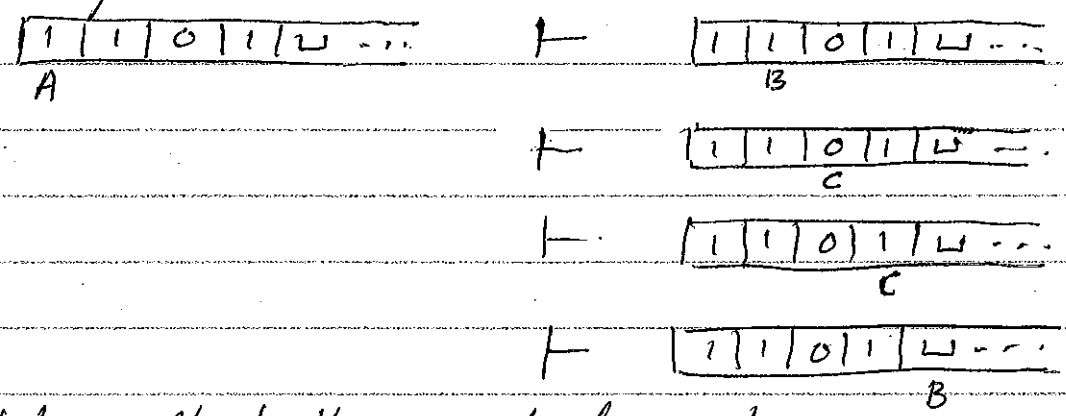
B head in state B in position indicate

configuration: $\langle B, \underbrace{aba}_{\varphi_1}\underbrace{cbba}_{\varphi_2} \rangle$ $\quad$ $\sqcup$ = blank symbol

Let us run the DFA M_1 of Ex. 1 with $\varphi = 1101$ as input. When the head reaches a "$\sqcup$" M_1 will halt.

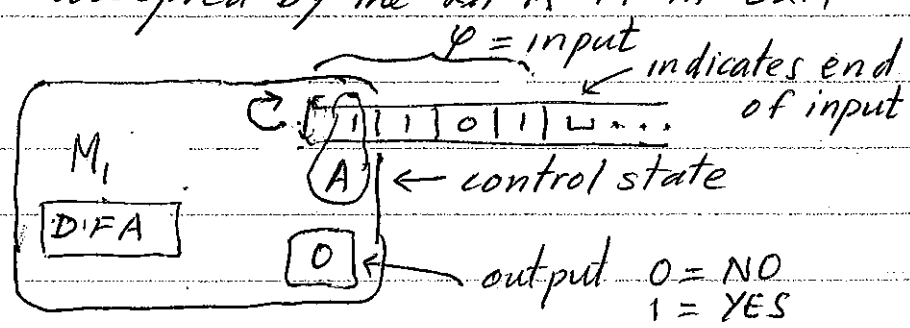
$$\langle A, \underline{1}101\sqcup \rangle \vdash \langle B, 1\underline{1}01\sqcup \rangle$$
$$\vdash \langle C, 11\underline{0}1\sqcup \rangle$$
$$\vdash \langle C, 110\underline{1}\sqcup \rangle$$
$$\vdash \langle B, 1101\underline{\sqcup} \rangle$$

The tapes will look as shown below:



Notice that the input does not change – we simply move into a control state with each character of the input and stop (halt) as soon as we reach the blank symbol "$\sqcup$". If the <u>last control state</u> we reach is an accepting state, we say that the DFA <u>accepts</u> the input; if it is not an accepting state, we say that the DFA <u>rejects</u> the input.

So 1101 is accepted by the DFA M in Ex. 1



<u>Note:</u> In a DFA we are only allowed to input a single string φ from T^*. The "$\sqcup$" cannot be a part of the input alphabet T. Instead of showing the sequence of configurations of the DFSA M_1 on an input φ (or the sequence of tapes) we can just show what happens using two rows as shown below.

<u>input characters</u> :	1	1	0	1	$\sqcup$
<u>control states</u> :	A	B	C	C	B

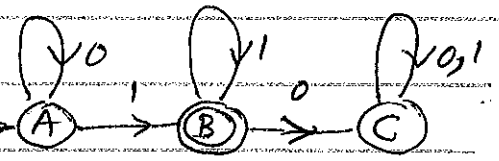
Let us try another input, say $\psi = 100$ on M_1.

input :	1	0	0	$\sqcup$
states :	A	B	A	A

Since A is not an accepting state, 100 is not accepted by M. We will define $L(M_1)$ to be the set of all strings in T^* which are accepted by M_1.

<u>Ex. 2</u> Let M_2 be the DFA shown on the right. Find $L(M_2)$.

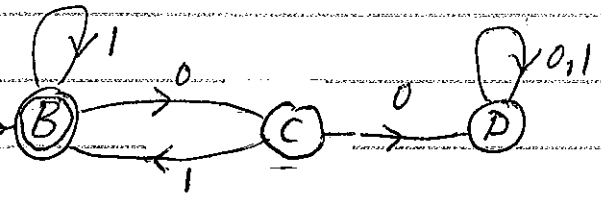
M_2:



$$L(M_2) = \underline{0}^*\underline{1}\,\underline{1}^*$$

<u>Ex. 3</u> Let M_3 be the DFA. show below. Find $L(M_3)$

M_3:



$$L(M_3) = (\underline{1} + \underline{0}\,\underline{1})^*$$

We will now define precisely what it means for a DFA M to accept a string φ and precisely what is $L(M)$.

Def. The <u>extended transition</u> function $\delta^*: Q \times T^* \to Q$ of a DFA is defined recursively as follows.

(a) $\delta^*(q, \lambda) = q$

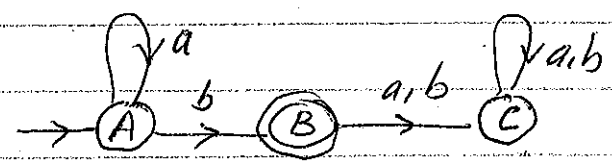
(b) $\delta^*(q, \varphi c) = \delta(\delta^*(q, \varphi), c)$ for each $c \in T$.

Def. The string $\varphi \in T^*$ is said to be <u>accepted</u> by a DFSA M if $\delta^*(q_0, \varphi) \in \mathcal{A}(M)$. The string φ is <u>rejected</u> by M if $\delta^*(q_0, \varphi) \notin \mathcal{A}(M)$.

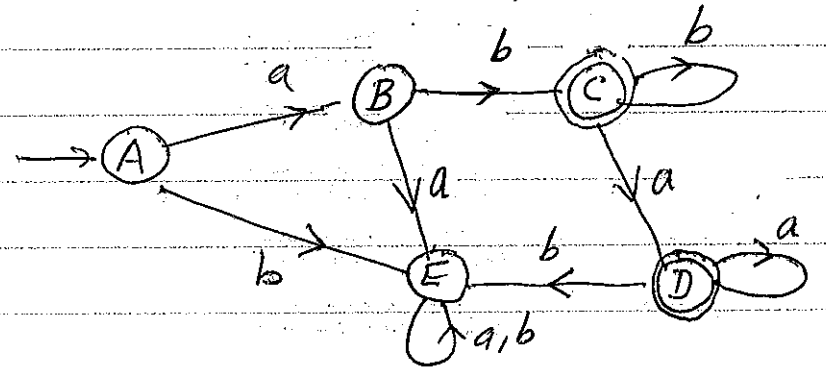
Def. The <u>language recognized</u> by a DFA M is defined by $L(M) = \{\varphi \in T^* : \delta^*(q_0, \varphi) \in \mathcal{A}(M)\}$.

Ex. 1 Find a DFA M_1 which recognizes the language $L_1 = \{a^n b : n \geq 0\}$

M_1:



Ex. 2 Find a DFSA M_2 which recognizes the language $L_2 = \underline{a}\,\underline{b}^*\,\underline{b}\,\underline{a}^*$ <u>Hint</u>: $\underline{a}\,\underline{b}^*\,\underline{b}\,\underline{a}^* \equiv \underline{a}\,\underline{b}\,\underline{b}^*\,\underline{a}^*$



Ex.3 Let $n_a(w)$ = number of a's in w and $n_b(w)$ = the number of b's in w. Find a DFA which recognizes the language

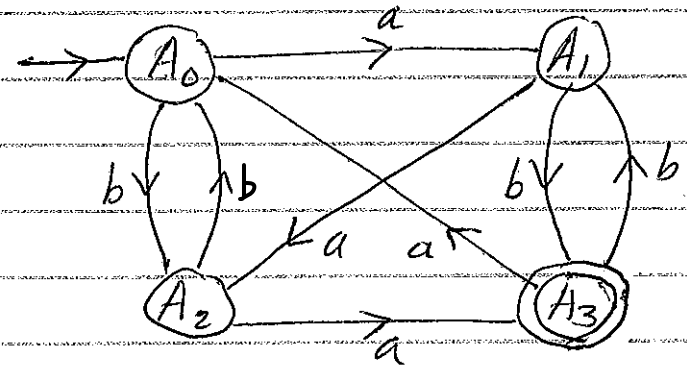
$$L_3 = \{w \in \{a,b\}^* : n_a(w) + 2n_b(w) \equiv 3 \pmod 4\}.$$

Let the states be A_0, A_1, A_2, and A_3 and let $f(w) = n_a(w) + 2n_b(w)$. We will use A_i to keep track of the fact that when the initial part φ of w is processed, the value of $f(\varphi) \equiv i \pmod 4$. Since $f(\lambda) = 0$, the initial state will be A_0. Also when $f(\varphi) \equiv 3 \pmod 4$ φ will be accepted, so A_3 is the only accepting state. Now

$$f(\varphi a) = n_a(\varphi a) + 2n_b(\varphi a) \equiv n_a(\varphi) + 2n_b(\varphi) + 1 = f(\varphi) + 1 \pmod 4$$

& $$f(\varphi b) = n_a(\varphi b) + 2n_b(\varphi b) \equiv n_a(\varphi) + 2n_b(\varphi) + 2 \equiv f(\varphi) + 2 \pmod 4$$

So $M_3 =$



Ex.4 Show what happens when w = ababa is used as the input to the DFA: M_3 and verify your answer using modulo arithmetic.

input: a b a b a ␣

states: A_0 A_1 A_3 A_0 A_2 A_3 So $w \in L(M_3)$

Check: $f(w) = n_a(w) + 2n_b(w) = 3 + 2(2) = 7 \equiv 3 \pmod 4$

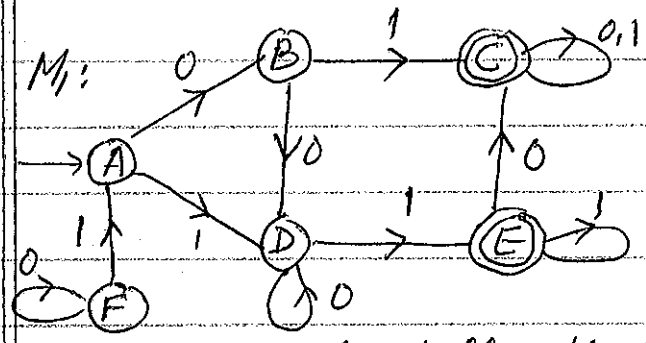
Since $w \in L_3$, w should be accepted by M_3.

§2. The Partition Algorithm & Minimal DFSAs

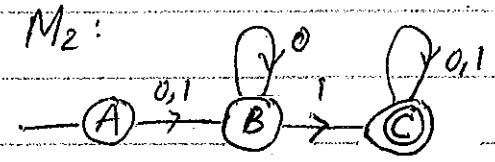
Def. Two DFAs M_1 and M_2 are said to be <u>equivalent</u> if $L(M_1) = L(M_2)$.

Ex.1 Consider the two DFAs M_1 & M_2 below

M_1:



M_2:



It is not difficult to see that

$$L(M_1) = (0+1).0^*.1.(0+1)^* = L(M_2). \quad \text{So } M_1 \equiv M_2.$$

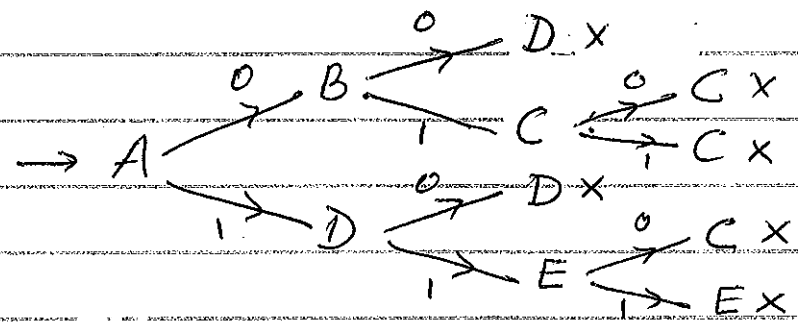
But it is clear that M_2 is much simpler than M_1. We will later see M_2 is the DFSA with the smallest number of states which is equivalent to M_1.

Def. A state q in a DFA M is said to be <u>inaccessible</u> if there is no string $w \in T^*$ such that $\delta^*(q_0, w) = q$.

In otherwords, q is inaccessible if we can never reach it from the initial state. In M_1 the state F is inaccessible because there is no way to get from A to F.

Ex.2 How can we find all the inaccessible states in the DFSA M_1 of Ex.1

Ans: (Ex.2) Start with the initial state of M and keep branching out with each character of T. When a previously accessible state is reached, discontinue that branch. The accessible states of M will be all the states in the accessibility tree.



So the acc. states are A, B, C, D, & E. ∴ F is inacc.

To find the DFA M_R which is equivalent to a given DFA M, we will remove the inaccessible states and get rid of the redundant states.

Def. Two states p & q in a DFA, M, are <u>indistinguishable</u> if for each string $\varphi \in T^*$ we have

$$\delta^*(p,\varphi) \in \mathcal{A}(M) \Leftrightarrow \delta^*(q,\varphi) \in \mathcal{A}(M).$$

So indistinguishable states are in a sense equivalent. Also if $p_1, p_2, \ldots, p_k$ are indistinguishable states which are all accessible in M, then we need only one of them, p_1 say, and the rest will all be redundant.

In M_1, the states B & D are indistinguishable. Also C & E are indistinguishable. So if we keep B and C, then D and E will be redundant & can be removed

In removing a redundant state, we have to make sure that we replace it by an equivalent one.

<u>The Partition Algorithm for DFA</u>

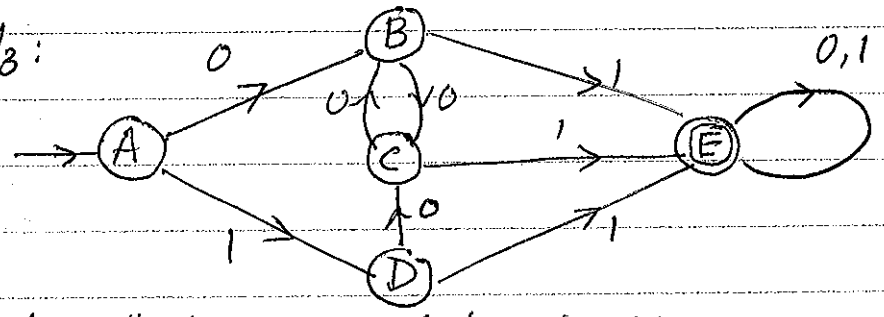
<u>Input</u>: The set of states of a DFA M

<u>Output</u> Blocks of indistinguishable states of M

1. Start with the initial partition $\mathcal{P}_0$ consisting of 2 blocks $\mathcal{P}_0$: $Q - \mathcal{A}$, $\mathcal{A}$. Then let $n \leftarrow 0$.
2. Two states p and q in the same block of $\mathcal{P}_n$ will stay together in the same block of $\mathcal{P}_{n+1}$ if for each $c \in T$, $\delta(p,c)$ & $\delta(q,c)$ belong to the same block of $\mathcal{P}_n$.
   If for at least one $c \in T$, $\delta(p,c)$ & $\delta(q,c)$ belong to different blocks of $\mathcal{P}_n$, then p & q will split and go into different blocks of $\mathcal{P}_{n+1}$.
3. If $\mathcal{P}_{n+1} = \mathcal{P}_n$, then STOP. ($\mathcal{P}_n$ will be the final partition of Q into blocks of indistinguishable states
   Otherwise, let $n \leftarrow n+1$; and go to step. 2

<u>Ex.3</u> Partition the states of the DFA M_3 into blocks of indistinguishable states.

M_3:



Notice that all the states in M_3 are accessible.

1. $\mathcal{P}_0$: $\{A, B, C, D\}$ $\{E\}$

$A \xrightarrow{0} B \in$ 1st block of $\mathcal{P}_0$, $A \xrightarrow{1} D \in$ 1st block of $\mathcal{P}_0$

$B \xrightarrow{0} C \in$ 1st block of $\mathcal{P}_0$, $B \xrightarrow{1} E \in$ 2nd block of $\mathcal{P}_0$

$C \xrightarrow{0} B \in$ 1st block of $\mathcal{P}_0$, $C \xrightarrow{1} E \in$ 2nd block of $\mathcal{P}_0$

$D \xrightarrow{0} C \in$ 1st block of $\mathcal{P}_0$, $D \xrightarrow{1} E \in$ 2nd block of $\mathcal{P}_0$

2. $\mathcal{P}_1$: $\{A\}$ $\{B, C, D\}$ $\{E\}$

$B \xrightarrow{0} C \in$ 2nd block of $\mathcal{P}_1$ $B \xrightarrow{1} E \in$ 3rd block of $\mathcal{P}_1$

$C \xrightarrow{0} B \in$ 2nd block of $\mathcal{P}_1$ $C \xrightarrow{1} E \in$ 3rd block of $\mathcal{P}_1$

$D \xrightarrow{0} C \in$ 2nd block of $\mathcal{P}_1$ $D \xrightarrow{1} E \in$ 3rd block of $\mathcal{P}_1$

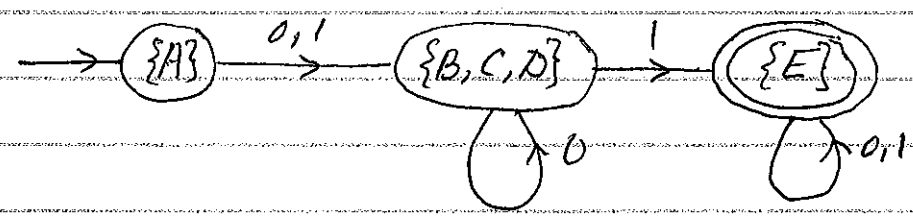
3. $\mathcal{P}_2$: $\{A\}$ $\{B, C, D\}$ $\{E\}$.

Since $\mathcal{P}_2 = \mathcal{P}_1$, $\mathcal{P}_1$ is the final partition of Q into blocks of indistinguishable states.

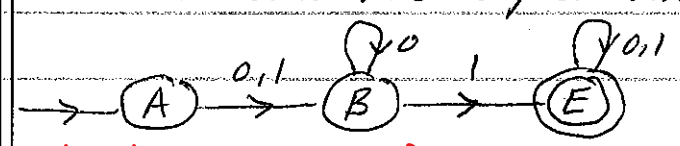
<u>Ex 4</u> Find the minimal (or <u>reduced</u>) DFA, M_R that is equivalent to the DFA: M_3 in Ex. 3.

We will use each block of the final partition as a state in M_R. The initial state will be the one that contains the initial state of M_3. The accepting states will be the blocks which consists only of accepting states

M_R =



Some textbooks like to pick one representative each to get



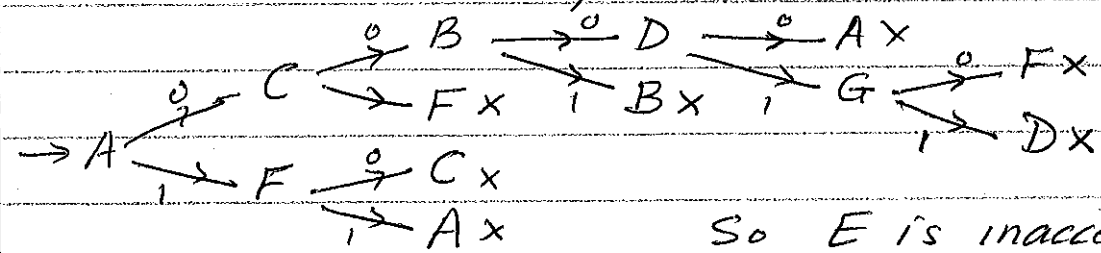
but we prefer using the blocks.

<u>Note</u>: $\mathcal{L}(M_3) = \mathcal{L}(M_R) = (\underline{0}+\underline{1}) \cdot \underline{0}^* \cdot \underline{1} \cdot (\underline{0}+\underline{1})^*$.

Ex. 5 Find the minimal DFA M_R that is equivalent to the DFA M below.

	$\to A$	B	C	D	E	F	G
0	C	D	B	A	A	C	F
1	F	B	F	G	E	A	D

1. First we check for any inaccessible states



So E is inaccessible

2. Now we partition $\{A, B, C, D, F, G\}$ into blocks of indistinguishable states.

$\mathcal{P}_0$: $\{A, B, F\}$ $\{C, D, G\}$
non-accepting states accepting states

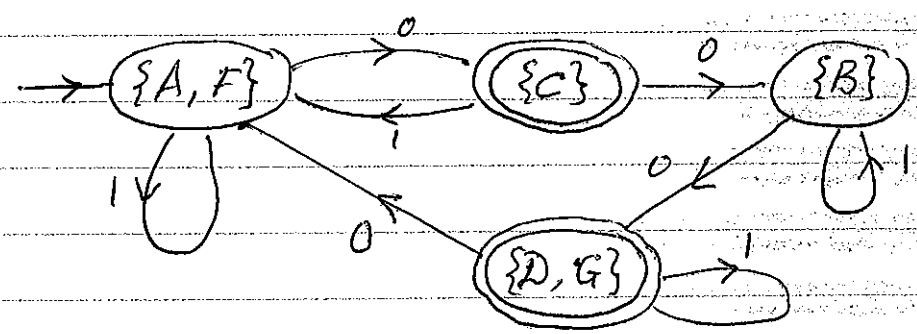
$\mathcal{P}_1$: $\{A, B, F\}$ $\{C\}$ $\{D, G\}$

$\mathcal{P}_2$: $\{A, F\}$ $\{B\}$ $\{C\}$ $\{D, G\}$

$\mathcal{P}_3$: $\{A, F\}$ $\{B\}$ $\{C\}$ $\{D, G\}$ $= \mathcal{P}_2$
= final partition

3. Finally we make the DFSA M_R

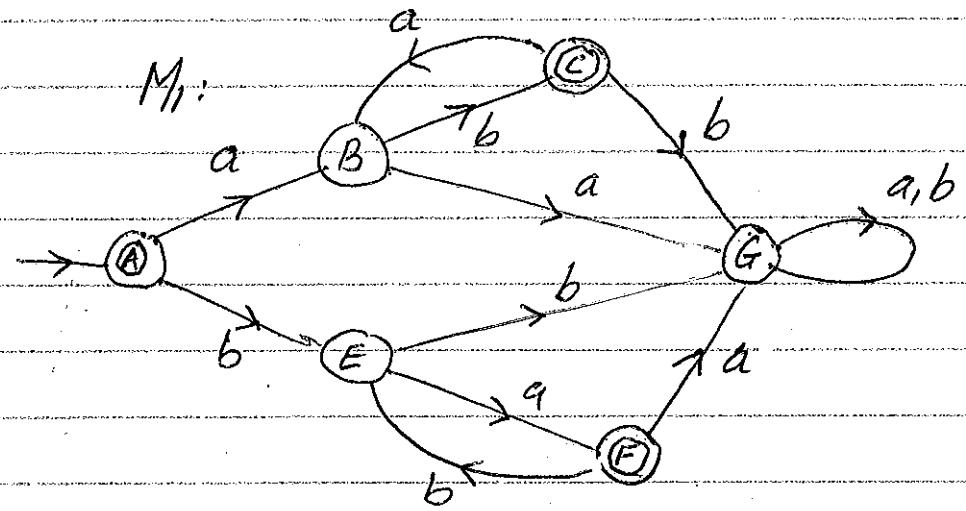
$M_R :=$



§3. Non-deterministic Finite Acceptor

Ex. 1 Find a DFA which recognizes the language $L_1 = (\underline{a}\,\underline{b})^* + (\underline{b}\,\underline{a})^*$

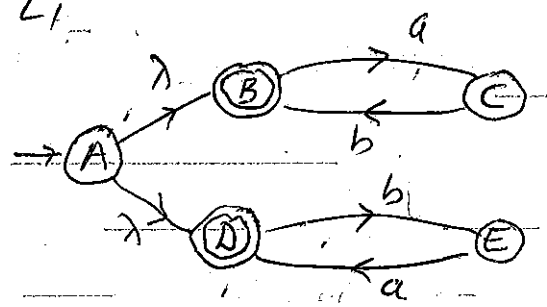
M_1:



Athough L_1 appears to be a rather simpler language, the DFA M_1 is decidedly complex.

If we generalize the concept of a DFA and allow non-deterministic mode of operation, we can get a much simpler acceptor M_1' which recognizes L_1

M_1':



We can jump from A to B with the empty string λ and go around the loop from B to C to recognize $(\underline{a}\,\underline{b})^*$ — but we can't go back to A from B and

try to reach D. We can jump with λ from A to D and go around the loop from D to E to recognize the second part $(ba)^*$ but again we cannot go back to A from D. So $L(M_1') = L_1$

<u>Note:</u> In the NFA M_1', $A \xrightarrow{\lambda} B$ & $A \xrightarrow{\lambda} D$ are transitions which are called <u>lambda-transitions</u> Also if we reach C and the next character is an "a" then we cannot move and we say that the NFA crashes. (The NFA M will accept a string φ only if there is some way to reach the first blank symbol $\sqcup$ after φ and do so in an accepting state). Also for each state X, the transition $X \xrightarrow{\lambda} X$ is present but not shown.

Let us see what happen if baba is the input to the NFA M_1'.

$\rightarrow A \xrightarrow{\lambda} B \xrightarrow{b}$ crashes — rejected by crashing

$\rightarrow A \xrightarrow{\lambda} D \xrightarrow{b} E \xrightarrow{a} D \xrightarrow{b} E \xrightarrow{a} D$ — accepted

Let us also see what happens if aba is the input

$\rightarrow A \xrightarrow{\lambda} B \xrightarrow{a} C \xrightarrow{b} B \xrightarrow{a} C$ — properly rejected

$\rightarrow A \xrightarrow{\lambda} D \xrightarrow{a}$ crashes — rejected by crashing

Finally let us look at the input abb

$\rightarrow A \xrightarrow{\lambda} B \xrightarrow{a} C \xrightarrow{b} B \xrightarrow{b}$ crashes, rejected by crashing

$\rightarrow A \xrightarrow{\lambda} D \xrightarrow{a}$ crashes — rejected by crashing

So M_1' accepts baba and rejects aba & abb.

In otherwords, a string is accepted if there is at least one way of it being accepted. Observe that the string λ, is accepted along two paths & rejected along one path.

$A \xrightarrow{\lambda} A$ rejected because $A \notin \mathcal{A}(M_1')$

$A \xrightarrow{\lambda} B$ accepted because $B \in \mathcal{A}(M_1')$

$A \xrightarrow{\lambda} D$ accepted because $D \in \mathcal{A}(M_1')$.

We will now define precisely what is an NFA and also define what is $L(M)$ for an NFA M.

Def. A <u>Non-deterministic Finite Acceptor</u> (NFA) is a 5-tuple $M = \langle Q, T, \Delta, q_0, \mathcal{A} \rangle$ where Q, T, q_0 and $\mathcal{A}$ are as in a DFA and $\Delta \subseteq [Q \times (T \cup \{\lambda\})] \times Q$ is a <u>binary relation</u> from $Q \times (T \cup \{\lambda\})$ to Q.

We interpret the ordered pair $\langle \langle q_1, c \rangle, q_2 \rangle$ as meaning that c can lead you from q_1 to q_2. We sometimes write the ordered pair $\langle \langle q_1, c \rangle, q_2 \rangle$ as the expression $q_1 \xrightarrow{c} q_2$ and call it <u>a transition</u>. If $c = \lambda$, and $q_1 \neq q_2$, we call $q_1 \xrightarrow{\lambda} q_2$ a <u>λ-transition</u>.

Note also that if the relation Δ turns out to be a total function from $Q \times (T \cup \{\lambda\})$ to Q, then the NFA will actually be a DFA. <u>The extended transition relation</u> Δ^* is the set of all ordered pairs $\langle \langle q_1, \varphi \rangle, q_2 \rangle$ such that φ can lead you from q_1 to q_2.

Def Let $\varphi \in T^*$ and M be an NFA. We say that φ is <u>accepted</u> by M if we can find a sequence of transitions $q_0 \xrightarrow{c_1} q_1 \xrightarrow{c_2} q_2 \xrightarrow{c_3} \cdots \cdots \xrightarrow{c_n} q_n$ such that $c_1 c_2 \cdots c_n = \varphi$ and $q_n \in \mathcal{A}(M)$. Here each $c_i \in T \cup \{\lambda\}$. (In otherwords φ is accepted by M if $\langle\langle q_0, \varphi\rangle, q_F\rangle \in \Delta^*$ for some $q_F \in \mathcal{A}(M)$.)

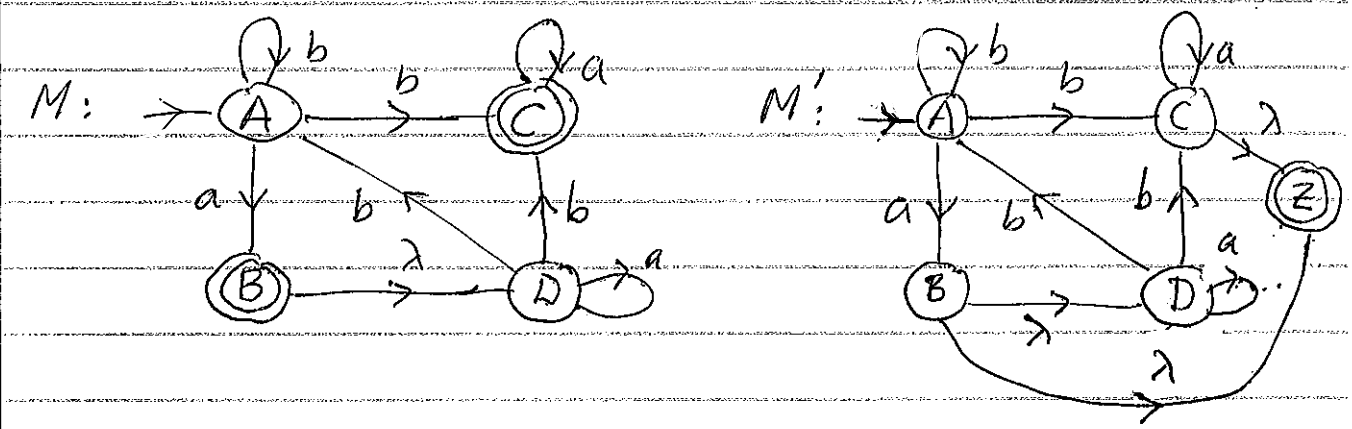
The <u>language recognized</u> by an NFA is defined by $L(M) = \{\varphi \in T^* : \varphi \text{ is accepted by } M\}$.

Def A <u>One-accepting-state NFA</u> (OAS-NFA) is an NFA which has exactly one accepting state which is different from the initial state

Prop. 1 Any NFA M is equivalent to an OAS-NFA M'.

Proof. Let $Q(M') = Q(M) \cup \{Z\}$, where Z is a new state that is not already in $Q(M)$. Then add λ-transitions from each state in $\mathcal{A}(M)$ to Z. Finally let $\mathcal{A}(M') = \{Z\}$. Then it can be shown that $L(M') = L(M)$. So $M' \equiv M$.

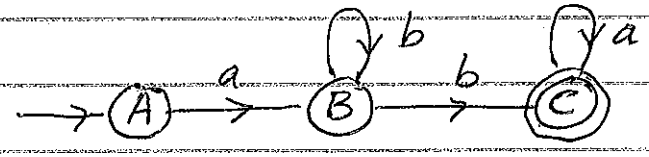
Ex. 2 Find an OAS-NFA M' which is equivalent to the NFA M shown below.



Ex 2: Find an NFA which recognizes the language

$L_2 = \underline{a}\,\underline{b}^*\,\underline{b}\,\underline{a}^*$

M_2:

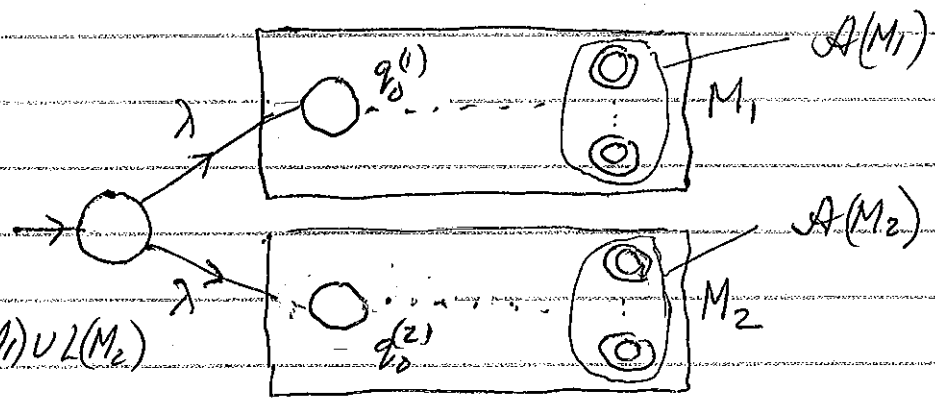


If you try to find a DFA which recognizes L_2, then you will begin to appreciate how "user friendly" NFAs are.

Qu.1 Given NFAs M_1 & M_2, how can we find NFAs which can recognize

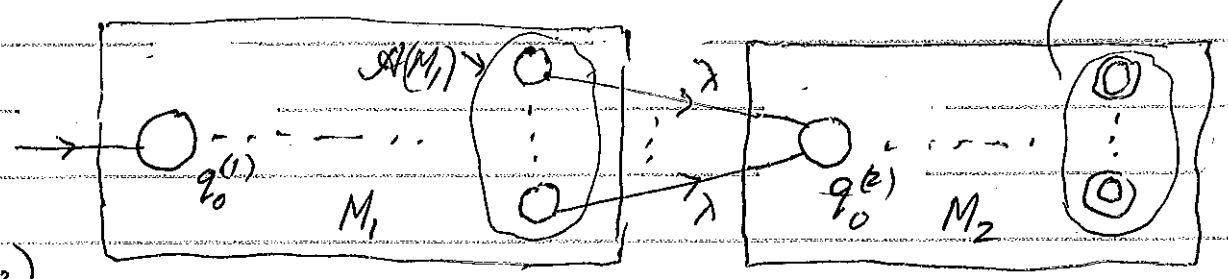
(a) $L(M_1) \cup L(M_2)$ (b) $L(M_1) \cdot L(M_2)$ (c) $L(M_1)^*$ (d) $L(M_1)^R$.

(a)



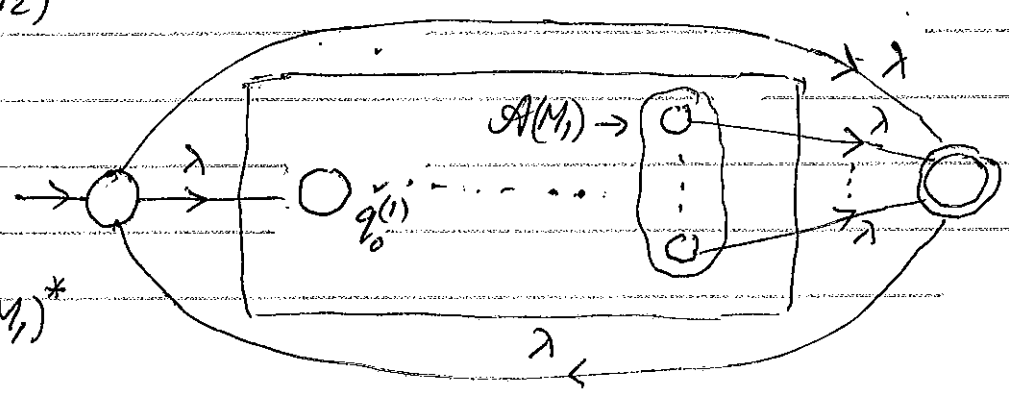
NFA for $L(M_1) \cup L(M_2)$

(b)



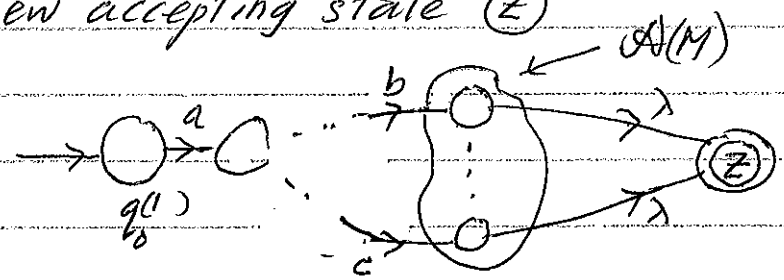
NFA for $L(M_1) \cdot L(M_2)$

(c)



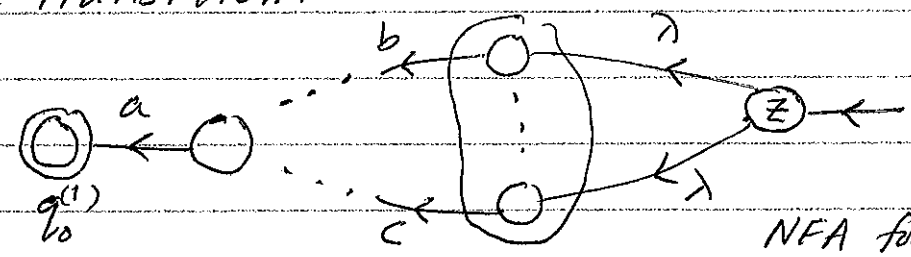
NFA for $L(M_1)^*$

(d) First convert M_1 into an OAS-NFA by adding a new accepting state Z



OAS-NFA

Then make Z into the initial state, $q_0^{(1)}$ into the only accepting state, and reverse the direction of each transition.



NFA for $L(M_1)^R$.

Now if we try to answer Qu.1 with DFAs replacing NFAs, then it will not at all be easy. That is because DFAs are not very easy to combine. But there is one thing that we can do with DFAs that we cannot easily do with NFAs.

<u>Prop. 2</u>: Let $M = \langle Q, T, \delta, q_0, \mathcal{A}(M) \rangle$. Put $M_c = \langle Q, T, \delta, q_0, Q - \mathcal{A}(M) \rangle$. Then $L(M_c) = L(M)^c$

<u>Proof</u>: $w \in L(M_c) \iff \delta^*(q_0, w) \in Q - \mathcal{A}(M)$

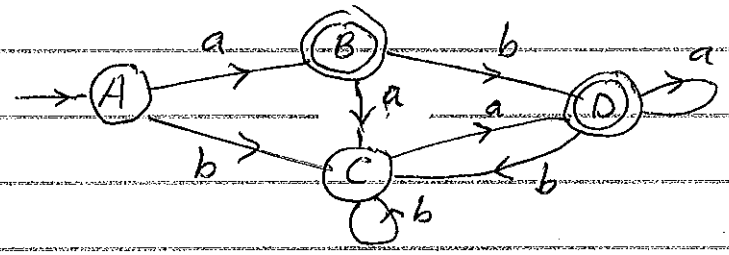
$\iff \delta^*(q_0, w) \notin \mathcal{A}(M)$

$\iff w \notin L(M)$.

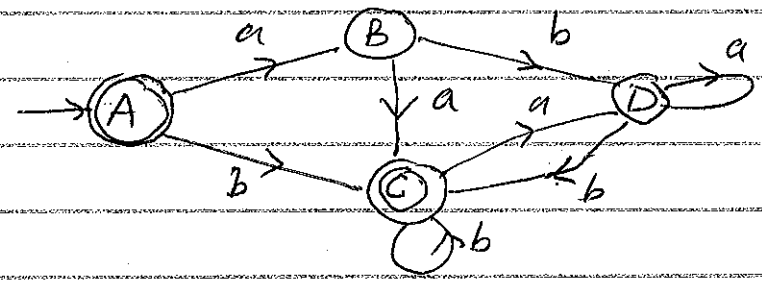
So $L(M_c) = L(M)^c$ and we are done.

Ex. 3 Find a DFA M_c such that $L(M_c) = L(M)^c$ where M is the DFA shown below.

M:



Ans: M_c:

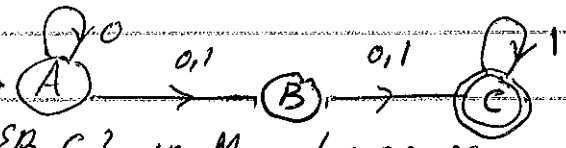


We cannot do this with NFAs because Prop 2 does not hold for NFAs. Fortunately, we can show, in the next section, that every NFA is equivalent to a DFA and this will help us.

§4. Equivalence of NFAs & DFAs.

Def. Let M be an NFA and $\varphi \in T^*$. The reaching set $R(\varphi)$ of φ in M is defined as follows.
$q_n \in R(\varphi)$ if and only if there is a sequence of transitions $q_0 \xrightarrow{c_1} q_1 \xrightarrow{c_2} q_2 \cdots q_{n-1} \xrightarrow{c_n} q_n$ with $c_1 c_2 \cdots c_n = \varphi$ and each $c_i \in T \cup \{\lambda\}$.

Ex. 4 Let $M_4 =$

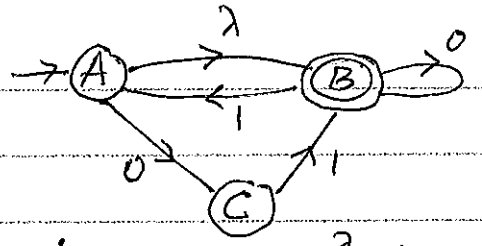


Then $R(01) = \{B, C\}$ in M_4 because

$$\to A \xrightarrow{0} A \xrightarrow{1} B \quad \& \quad \to A \xrightarrow{0} B \xrightarrow{1} C$$

Also $R(00) = \{A, B, C\}$ in M_4 because

$\to A \xrightarrow{0} A \xrightarrow{0} A$, $\to A \xrightarrow{0} A \xrightarrow{0} B$, $\to A \xrightarrow{0} B \xrightarrow{0} C$.

Ex.5 Let $M_5 =$ 

. Then

(a) $R(\lambda) = \{A, B\}$ because $\to A \xrightarrow{\lambda} A$, $\to A \xrightarrow{\lambda} B$

(b) $R(0) = \{B, C\}$ because $\to A \xrightarrow{\lambda} B \xrightarrow{0} B$, $\to A \xrightarrow{0} C$.

(c) $R(00) = \{B\}$ because $\to A \xrightarrow{\lambda} B \xrightarrow{0} B \xrightarrow{0} B$.

<u>The NFA to DFA Algorithm</u>

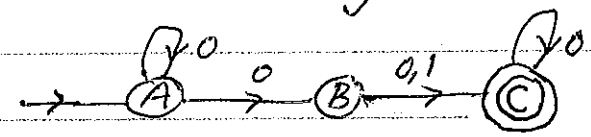
<u>Input</u>: An NFA, $M = \langle Q, T, \Delta, q_0, \mathcal{A} \rangle$

<u>Output</u>: An equivalent DFA, $M_D = \langle Q_D, T, \delta, R(\lambda), \mathcal{A}_D \rangle$

1. First find all possible reaching sets by starting with $R(\lambda)$ and branching out with each character in T. When a previous reaching set is encountered, that branch is discontinued. This process will give us the <u>reaching set tree</u>.

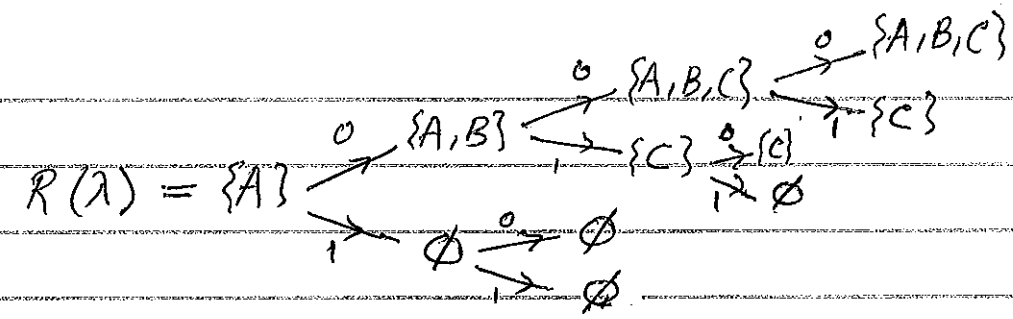
2. Let $Q_D = \{R(\varphi) : \varphi \in T^*\}$ = set of all reaching sets, <u>initial state</u> of $M_D = R(\lambda)$, and $\mathcal{A}_D = \{R(\varphi) : R(\varphi) \cap \mathcal{A}(M) \neq \emptyset\}$ = set of all reaching sets with at least one accepting state of M.

3. The <u>transition function</u> $\delta : Q_D \times T \to Q_D$ is specified by $\delta(R(\varphi), c) = R(\varphi c)$ for each $c \in T$, and can be read off from the reaching set tree.

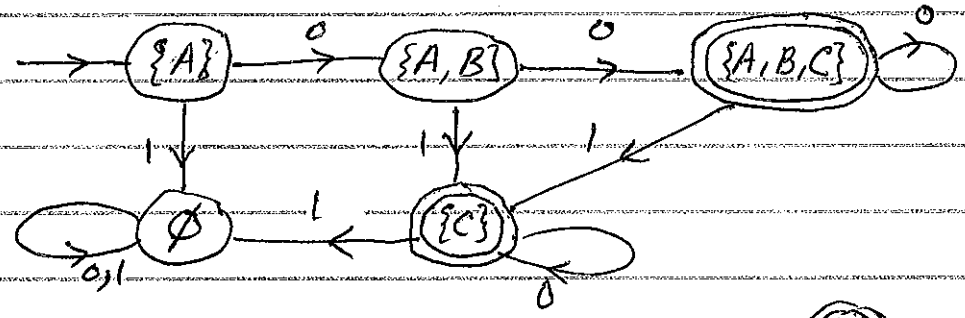
Ex.6 Let M be the NFA 

Find the equivalent DFA, M_D.

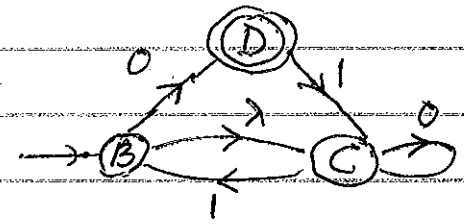
Ex. 6

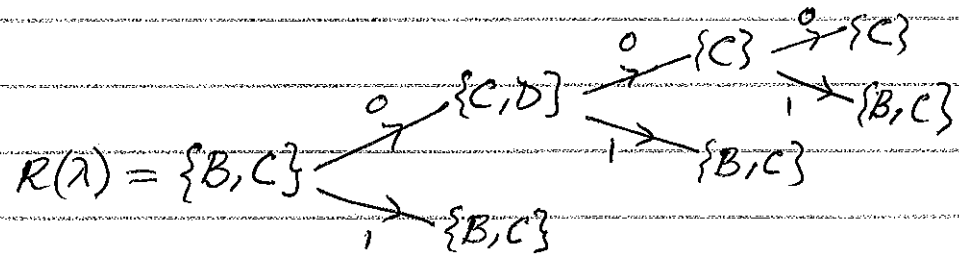


M_D :

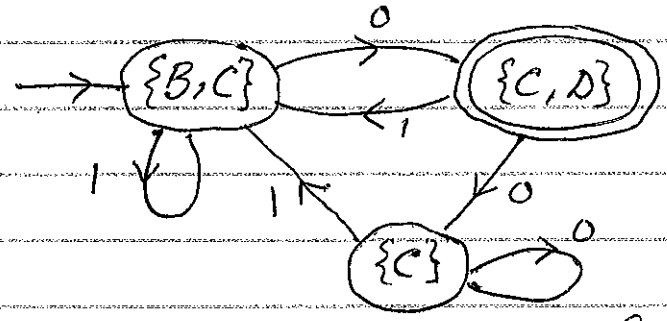


Ex7 Let M be the NFA on the right. Find the equivalent DFA, M_D.





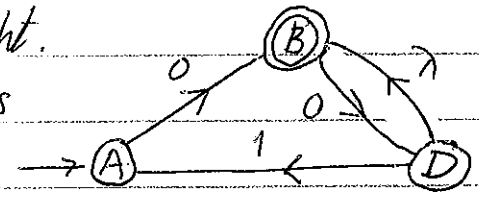
M_D =

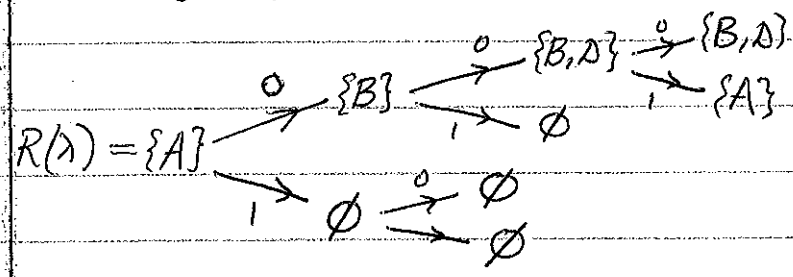


Qu. 2 Given an NFA, M, how can we find an NFA which recognizes the language $L(M)^c$?

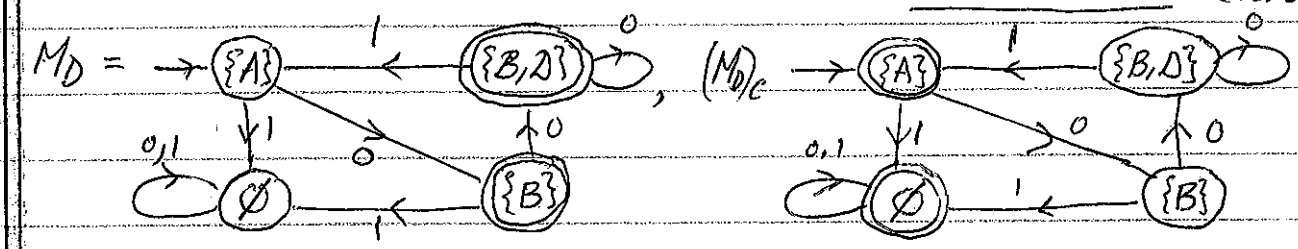
Ans: First convert M into an equivalent DFA M_D. Then switch the accepting & non-accepting states of M_D to get a DFA $(M_D)_c$. This will be your answer because $(M_D)_c$ is also an NFA.

Ex. 8 Let M be the NFA on the right. Find an NFA which recognizes the language $L(M)^c$.





Final answer $= (M_D)_c$



Theorem 3: Let $M = (Q, T, \Delta, q_0, \mathcal{A})$ be any NFA and $M_D = \langle Q_D, T, \delta, R(\lambda), \mathcal{A}_D \rangle$ be the DFA obtained from the NFA to DFA algorithm. Then $L(M_D) = L(M)$.

Proof:

$w \in L(M_D) \iff$ w can lead you from $R(\lambda)$ in M_D to an accepting state R_A in $\mathcal{A}_D$

$\iff$ w can lead you from q_0 in M to an accepting state q_A in $\mathcal{A}$ (because $R(\lambda)$ = all states you can reach from q_0 using λ), and R_A is an accepting state of M_D if & only if R_A contains at least one accepting state q_A of M.

$\iff$ $w \in L(M)$.

So $L(M_D) = L(M)$.

END OF Ch. 3.