Slides 2

Theory Of Computation

Dr. Talha Waheed

Text and Reference Material

1. *Introduction to Computer Theory*, by Daniel I. Cohen, John Wiley and Sons, Inc., 1991, 2nd Edition

2. *Introduction to Languages and Theory of Computation*, by J. C. Martin, McGraw Hill Book Co., Fourth Edition

What does automata mean?

- It is the plural of automaton (or machine), and it means "something that works automatically" (without human intervention)

Introduction to languages

- There are two types of languages

1) Informal (or Natural) Languages, like English, French (Semantic languages)

2) Formal Languages (Syntactic (Strictly Rule oriented) languages)

  - In English we have alphabets, words and sentences

Formal Languages

- Hard to state all rules of English for forming words from Alphabets. E.g.
  - Like only valid single letter words are {a, i}
  - Similarly two letter valid words are {am, an, as, at, ax}
- So, no formal (or precise maths) rule can be made.
- But in Formal languages all the rules are explicitly stated what strings or symbols can occur.
- And we are interested in its syntax (not in semantics)

Alphabets/ Letters

- Definition:

  A finite non-empty set of symbols (letters), is called an alphabet. It is denoted by Σ (Capital sigma).

- Example:

  Σ={a, b}

  Σ={0, 1}

  Σ={i, j, k}

Letters of a Programming Language

- A certain version of (an old programming language) ALGOL has 113 letters

  Σ (alphabet) includes letters, digits and a variety of operators (+, -) including sequential operators such as IF, GOTO

Strings

- Definition:

  Concatenation of finite symbols from the alphabet is called a string.

- Examples:

  If Σ= {a, b} then

  a, abab, aaabb, abababababababababab

  If Σ= {0, 1} then

  0, 0101, 00011, 01010101010101010101

Two strings are same if have same letters in the same order

EMPTY / NULL STRING

- Sometimes a string with no symbol at all is used, denoted by λ or Λ (Small or Capital Lambda), or ϵ (small epsilon) is called an empty or null string.

- The capital lambda will mostly be used to denote the empty string, in further discussion.

- It is most of the time required for mathematical properties, Just like 0 in number theory or empty set in set theory.
- So $a^0 = \Lambda$, $a^1 = a$, $a^2 = aa$, $a^3 = aaa$, …
- for string x = ab, $x^0 = \Lambda$, $x^1 = ab$, $x^2 = ab.ab$, $x^3 = ab.ab.ab$, …

- Language with no Words {} (denoted by Phi, Φ) is different from a Language with a string of no alphabets {Λ}

Words

- Definition:

  Words are strings belonging to some language.

Example:

If Σ= {a} then a language L can be defined as

$L=\{a^n : n=1,2,3,.....\}$ or L={a, aa, aaa,....}

Here a, aa, … are the words of L

Length of Strings

- Definition:

  The length of string s, denoted by |s|, it is the number of letters in the string.

- Examples:

1) $|\Lambda|=0$
2) For $\Sigma=\{a,b\}$, and s=ababa, $|s|=5$

3) For $\Sigma=$ {B, aB, bab, d}, and s=BaBbabBd

   Tokenizing=(B), (aB), (bab), (d) (B),

   |s|=~~4~~ 5

Reverse of a String

- Definition:

  The reverse of a string s denoted by **Rev(s) or s^r**, is obtained by writing letters of s in reverse order.

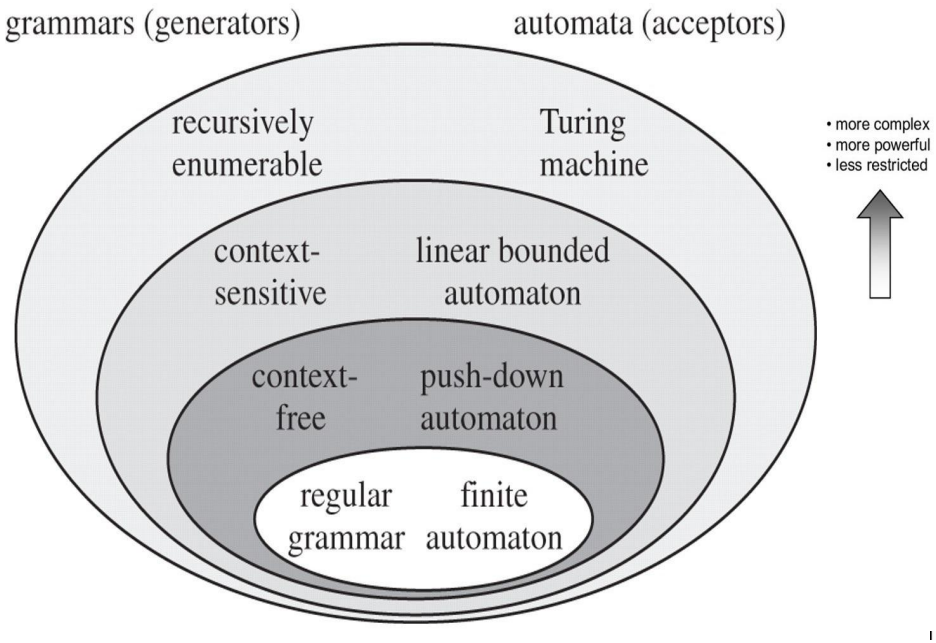
- Examples:

1) If s=abc is a string defined over Σ={a,b,c}
   then Rev(s) or s^r = cba

2) For Σ= {B, aB, bab, d} and s=BBababBd
   Rev(s)=dBbabaBB

Two Issues of All Formal languages

- How to Generate language strings (By Grammars)
- How to Accept/Recognize language strings (By Machines)



The Hierarchy

Class	Grammars	Languages	Automaton
Type-0	Unrestricted	Recursive Enumerable	Turing Machine
Type-1	Context Sensitive	Context Sensitive	Linear-Bound
Type-2	Context Free	Context Free	Pushdown
Type-3	Regular	Regular	Finite

Defining Languages

- Languages can be defined in different ways,
  1) Descriptive definition
  2) Recursive definition
  3) Regular Expressions(RE)
  4) Regular Grammar
  5) Finite Automaton(FA) etc.

Descriptive definition of language:

The language is defined in English, describing the conditions imposed on its words.

Descriptive Definition Examples

- The language L of **strings of odd length**, defined over Σ={a}, can be written as

  L={a, aaa, aaaaa,.....}

- The language L of strings that **does not start with a**, defined over Σ={a,b,c}, can be written as

  L={b, c, ba, bb, bc, ca, cb, cc, ...}

Descriptive Definition Examples

- The language L of **strings of length 2**, defined over Σ={0,1,2}, can be written as
  L={00, 01, 02,10, 11,12,20,21,22}

- The language L of **strings ending in 0**, defined over Σ ={0,1}, can be written as
  L={0,00,10,000,010,100,110,…}

Descriptive Definition Examples

- The language **EQUAL A and B**, of strings with number of a's equal to number of b's, defined over Σ={a,b}, can be written as

  {Λ ,ab,aabb,abab,baba,abba,...}

- The language **EVEN-EVEN**, of strings with even number of a's and even number of b's, defined over Σ={a,b}, can be written as

  {Λ, aa, bb, aaaa,aabb,abab, abba, baab, baba, bbaa, bbbb,...}

Descriptive Definition Examples

Non Regular Languages

- The language {$\mathbf{a^n b^n}$ }, of strings defined over Σ={a,b}, as

  {a^n b^n : n=1,2,3,…}, can be written as

  {ab, aabb, aaabbb,aaaabbbb,…}

- The language {$\mathbf{a^n b^n a^n}$ }, of strings defined over Σ={a,b}, as

  {$a^n b^n a^n$: n=1,2,3,…}, can be written as

  {aba, aabbaa, aaabbbaaa,aaaabbbbaaaa,…}

Descriptive Definition Examples

Non-Regular Language

- The language **factorial**, of strings defined over Σ={1,2,3,4,5,6,7,8,9} *i.e.*

  {1,2,6,24,120,…}

- The language **FACTORIAL**, of strings defined over Σ={a}, as

  $\{a^{n!} : n=1,2,3,\ldots\}$, can be written as

  {a, aa, aaaaaa,…}.

It is to be noted that the language FACTORIAL can be defined over any single letter alphabet.

Descriptive Definition Examples

Non-Regular Languages

- The language **DOUBLEFACTORIAL**, of strings defined over Σ={a, b}, as

  $\{a^{n!}b^{n!} : n=1,2,3,\ldots\}$, can be written as

  {ab, aabb, aaaaaabbbbbb,…}

- The language **SQUARE**, of strings defined over Σ={a}, as

  $\{a^{n^2} : n=1,2,3,\ldots\}$, can be written as

  {a, aaaa, aaaaaaaaa,…}

Descriptive Definition Examples

Non-Regular Languages

- The language **DOUBLESQUARE**, of strings defined over Σ={a,b}, as

  $\{a^{n^2} b^{n^2} : n=1,2,3,\ldots\}$, can be written as

  {ab, aaaabbbb, aaaaaaaaabbbbbbbbb,…}

- The language **PRIME**, of strings defined over Σ={a}, as $\{a^p : p \text{ is prime}\}$, can be written as {aa,aaa,aaaaa,aaaaaaa,aaaaaaaaaaa…}

Descriptive Definition Examples

A palindrome is a word, number, phrase, or other sequence of symbols that reads the same backwards as forwards, such as the words madam or racecar

PALINDROME - An Important language

- The language consisting of Λ and the strings s defined over Σ such that Rev(s)=s.

- Example: For Σ={a,b},

  PALINDROME={Λ , a, b, aa, bb, aaa, aba, bab, bbb, ...}
- Two kinds: Even and Odd Palindromes

- Find some English words that are palindromes

Kleene Star Closure

- Given Σ, then the Kleene Star of the alphabet Σ, denoted by Σ^*, is the collection of all strings defined over Σ, including Λ.
- It is to be noted that Kleene Star can be defined over any set of strings. E.g.
- If Σ = {x}

  Then Σ^* = {Λ, x, xx, xxx, xxxx, ….}

  Kleen star always start with 0. Kleen plus start with 1
- If Σ = {0,1}

  Then Σ^* = {Λ, 0, 1, 00, 01, 10, 11, ….}
- If Σ = {aaB, c}

  Then Σ^* = {Λ, aaB, c, aaBaaB, aaBc, caaB, cc, ….}

Note

Languages generated by Kleene Star Closure of set of strings, are infinite languages.

(By infinite language, it is supposed that the language contains infinite many words, each of finite length).

Questions

Since S* is a subset of T* , so every string belonging to S* also belongs to T*
Also T* is a subset of S* .....

1) Let S={ab, bb} and T={ab, bb, bbbb}
Show that $S^* = T^*$ [Hint $S^* \subset T^*$ and $T^* \subset S^*$]

Do it yourself

subset: Set A is said to be a subset of Set B if all the elements of Set A are also present in Set B. In other words, set A is contained inside Set B.
Example: If set A has {X, Y} and set B has {X, Y, Z}, then A is the subset of B because elements of A are also present in set B.

2) Let S={ab, bb} and T={ab, bb, bbb}
Show that $S^* \neq T^*$ But $S^* \subset T^*$

Solution:

Since $S \subset T$, so
every string belonging to S^*, also belongs to T^*
but bbb is a string belongs to T^*
but does not belong to S^*.

Questions

- 3) Let S={a, bb, bab, abaab} be a set of strings.

  Are abbabaabab and baabbbabbaabb in S^*?

  Does any word in S^* have odd number of b's?

Read it yourself

Solution:

since abbabaabab can be grouped as (a)(bb)(abaab)ab ,

which shows that the last member of the group does not belong to S, so abbabaabab is not in S^*,

while baabbbabbaabb can not be grouped as members of S, hence baabbbabbaabb is not in S^*.

Since each string in S has even number of b's so there is no possibility of any string with odd number of b's to be in S^*.

Kleene PLUS Operation ($^+$)

- Plus Operation is same as Kleene Star Closure except that it does not generate Λ (null string), automatically.

Example:

- If Σ = {0,1}
  Then Σ^+ = {0, 1, 00, 01, 10, 11, ….}

- If Σ = {aab, c}
  Then Σ^+ = {aab, c, aabaab, aabc, caab, cc, ….}

Questions

Read it yourself

Q1)Is there any case when S^+ contains Λ? If yes then justify your answer.

Solution: consider S={Λ,a} then

S^+ ={Λ, a, aa, aaa, …}

Here Λ is in S^+ as member of S.

Thus Λ will be in S^+ , in this case.

Questions

Q2) Prove that for any set of strings S

i. $(S^+)^*=(S^*)^*$

Solution: In general Λ is not in S^+ , while Λ does belong to S^*. Obviously Λ will now be in $(S^+)^*$, while $(S^*)^*$ and S^* generate the same set of strings. Hence $(S^+)^*=(S^*)^*$.

Q2) continued…

ii) $(S^+)^+=S^+$

Solution: since S^+ generates all possible strings that can be obtained by concatenating the strings of S, so $(S^+)^+$ generates all possible strings that can be obtained by concatenating the strings of S^+, will not generate any new string.

Hence $(S^+)^+=S^+$

Q2) continued…

iii) Is $(S^*)^+=(S^+)^*$

Solution: since Λ belongs to S^* ,so Λ will belong to $(S^*)^+$ as member of S^* .Moreover Λ may not belong to S^+, in general, while Λ will automatically belong to $(S^+)^*$.

Hence $(S^*)^+=(S^+)^*$

Remark

- It is to be noted that Kleene Star can also be operated on any string *i.e.* a^* can be considered to be all possible strings defined over {a}, which shows that a^* generates

  Λ, a, aa, aaa, …

It may also be noted that a^+ can be considered to be all possible non empty strings defined over {a}, which shows that a^+ generates

a, aa, aaa, aaaa, …

Defining Languages: 2nd Method

- **Recursive definition of languages**

  The following three steps are used in recursive definition

1. **Base case**: Some basic words are specified in the language.
2. **Rule**: Rules for constructing more words are defined in the language.
3. No strings except those constructed in above, are allowed to be in the language.

Regular Language – All Strings Ending in a

- **Defining the language L, of strings ending in a , defined over Σ = {a,b}**

Step 1:

a is in L

Step 2:

if x is in L then **s(x)** is also in **L,** where s belongs to Σ^*

Step 3:

No strings except those constructed in above, are allowed to be in **L**

Regular Language - Strings Contain Exactly aa

- **Defining the language L, of strings containing exactly aa, defined over Σ={a, b}**

Step 1:

aa is in L

Step 2:

s(aa)s is also in **L,** where s belongs to $\mathbf{b}^*$

Step 3:

No strings except those constructed in above, are allowed to be in **L**

Regular Language – Strings Contain aa or bb

- **Defining the language L, of strings containing aa or bb , defined over Σ={a, b}**

Step 1:

aa and bb are in L

Step 2:

s(aa)s and **s(bb)s** are also in **L,** where s belongs to $\mathbf{\Sigma}^*$

Step 3:

No strings except those constructed in above, are allowed to be in **L**

Regular Language – Begin and end in Same Letter

- **Defining the language L, of strings beginning and ending in same letters, defined over Σ={a, b}**

Step 1:

a and b are in L

Step 2:

(a)s(a) and **(b)s(b)** are also in **L,** where s belongs to Σ^*

Step 3:

No strings except those constructed in above, are allowed to be in **L**

Example – Integer – A Regular Language

- **Defining language of INTEGER**

Step 1 (Base):

1 is in **INTEGER**.

Step 2 (Rule):

If x is in **INTEGER** then **x+1** and **x-1** are also in **INTEGER**.

Step 3:

No strings except those constructed in above, are allowed to be in **INTEGER**.

Example – Even – A Regular Language

- **Defining language of EVEN**

Step 1 (Base):

2 is in **EVEN**.

Step 2 (Rule):

If x is in **EVEN** then x+2 and x-2 are also in **EVEN**.

Step 3:

No strings except those constructed in above, are allowed to be in **EVEN**.

Non-Regular Language - $a^n b^n$

- **Defining the language $\{a^n b^n\}$, n=1,2,3,… , of strings defined over Σ={a,b}**

Step 1:

ab is in $\{a^n b^n\}$

Step 2:

if x is in $\{a^n b^n\}$, then **axb** is in $\{a^n b^n\}$

Step 3:

No strings except those constructed in above, are allowed to be in $\{a^n b^n\}$

Factorial – Non Regular Language

- **Defining the language factorial**

Step 1:

As 0!=1, so 1 is in **factorial**.

Step 2:

n!=n*(n-1)! is in **factorial**.

Step 3:

No strings except those constructed in above, are allowed to be in **factorial**.

Palindrome – A Non-Regular Language

- **Defining the language PALINDROME, defined over** $\Sigma = \{a,b\}$

Step 1:

a and b are in **PALINDROME**

Step 2:

if x is palindrome, then **s(x)Rev(s)** and **xx** will also be palindrome, where s belongs to Σ^*

Step 3:

No strings except those constructed in above, are allowed to be in palindrome

Moving to the 3rd Method of Language Definition- Regular Expression

- As discussed earlier that
- $\mathbf{a}^*$ generates Λ, a, aa, aaa, … and
  $\mathbf{a}^+$ generates a, aa, aaa, aaaa, …,

so the language
L_1 = {Λ, a, aa, aaa, …} can be expressed by a^* and
L_2 = {a, aa, aaa, …} can be expressed by a^+.

a^* and a^+ are called the regular expressions (RE) for L_1 and L_2 respectively.

Note: $\mathbf{a}^+ = \mathbf{aa}^* = \mathbf{a}^*\mathbf{a}$, all generates L_2.

Recursive Definition of Regular Expression

Step 1: Every letter of Σ including Λ is a regular expression.

Step 2: If r_1 and r2 are regular expressions then following are also regular expressions.

1. (r_1)
2. $r_1 r_2$
3. $r_1 + r_2$ and
4. r_1^*

Step 3: Nothing else is a regular expression.

Defining Languages thru Regular Expressions - Method 3

- Consider the language L={Λ, x, xx, xxx,…} of strings, defined over Σ = {x}.

  We can write this language as the Kleene star closure of alphabet Σ or $L=\Sigma^*=\{x\}^*$

  this language can also be expressed by the **regular expression x^***.

- Similarly the language L={x, xx, xxx,…}, defined over Σ = {x}, can be expressed by the **regular expression x^+**.

Regular Expression for Universal Language

- Now consider another language L, consisting of **all possible strings**, defined over Σ = {a, b}.

  expressed by regular expression **$(a + b)^*$**

- **What is the difference** in following

  $r_1 = a^* + b^*$ vs $r_2 = (a+b)^*$

r_1 does not generate any string combination of a and b, while r_2 generates all such strings. So $r_2 \supset r_1$

r1 is a subset of r2

Regular Expression for exactly an aa

- Now consider another language L, of strings having **exactly aa**, defined over Σ = {a, b},

  regular expression = **b^*aab^***

Languages of Even and Odd Lengths

- Now consider another language L, of **even length**, defined over Σ = {a, b},

it's regular expression = **$(aa+ab+ba+bb)^*$**

Or ((a+b)(a+b))*

- Now consider another language L, of **odd length**, defined over Σ = {a, b},

regular expression may be

$(a+b)(aa+ab+ba+bb)^*$ or

Remark

It may be noted that a language may be expressed by more than one regular expressions (as we see in case of Language of even length),

while given a regular expression there exist a unique language generated by that regular expression.

Languages of Atleast one a, start with aa and ends on bb

- Consider the language, defined over Σ={a , b} of words having **at least one a**, expressed by a regular expression

$$\mathbf{(a+b)^*a(a+b)^*}$$

- Consider the language, defined over Σ={a, b}, of words **starting with double a and ending in double b** then its regular expression

$$\mathbf{aa}(a+b)^*\mathbf{bb}$$

Few more Regular Expressions for Languages

- Consider the language, defined over Σ={a, b} of words **starting with a and ending in b**

  OR starting with b and ending in a, then its regular expression

  $$\mathbf{a}(a+b)^*\mathbf{b} + \mathbf{b}(a+b)^*\mathbf{a}$$

- Consider the language, defined over Σ = {a, b} of words having **at least one a and one b**, expressed by a regular expression

$$(a+b)^*\mathbf{a}(a+b)^*\mathbf{b}(a+b)^* + (a+b)^*\mathbf{b}(a+b)^*\mathbf{a}(a+b)^*$$

Questions with Solution

- Determine the RE of the language, defined over Σ={a, b} of **words beginning with a**.

Solution:

The required RE may be **a(a+b)***

- Determine the RE of the language, defined over Σ={a, b} of **words beginning with and ending in same letter.**

Solution:

Required RE may be **a** + **b** + **a**(a+b)***a** + **b**(a+b)***b**

Questions with Solution

- Determine the RE of the language, defined over Σ={a, b} of **words ending in b.**

Solution:

Required RE = $(a+b)^*$**b**

- Determine the RE of the language, defined over Σ={a, b} of **words not ending in a.**

Solution:

Required RE = $(a+b)^*$**b** + Λ

Or

$((a+b)^*b)^*$

Regular Expression for Even-Even

Language of strings, defined over Σ={a, b} having **even number of a's and even number of b's**. *i.e.*

EVEN-EVEN = {Λ, aa, bb, aaaa,aabb,abab, abba, baab, baba, bbaa, bbbb,...} ,

regular expression

$$\mathbf{(aa + bb + ((ab+ba)(aa+bb)^{*}(ab+ba))^{*}}$$

Equivalency of Regular Expressions

- **Definition:**

  Two regular expressions are said to be equivalent if they generate the same language.

  Example:

  Consider the following regular expressions

  $r_1 = (a + b)^* (aa + bb)$

  $r_2 = (a + b)^* aa + (a + b)^* bb$ then

  both regular expressions define the language of strings **ending in aa or bb**.

Note

- It is to be noted that if L_1 and L_2 are expressed by
  r_1 = (aa + bb) and **r_2 = (a + b)**,
- respectively then the language expressed by

1) **r_1 + r_2**, is also the regular language L_1 + L_2 or **L_1 U L_2**

$$r_1+r_2 = (aa + bb) + (a + b)$$

2) **$r_1 r_2$** , is also the regular language L_1L_2, of strings obtained by **prefixing every string of L_1 with every string of L_2**

$$r_1r_2 = (aa + bb)\ (a + b) = (aaa + aab + bba + bbb)$$

3) **r_1^***, is also the regular language L_1^*, of strings obtained by **concatenating the strings of L, including the null string**.

$$(r_1)^* = (aa + bb)^* \text{ and for } (r_2)^* = (a + b)^*$$

Regular Languages formal Definition

The language generated by any regular expression is called a **regular language**.

It is to be noted that if r_1, r_2 are regular expressions, corresponding to the languages L_1 and L_2 then following languages are also regular languages:

Step 1 (Base Cases):

1) Φ (Empty Language)
2) Λ (Empty String)
3) Σ (any single length alphabet)

Step 2 (Rules or Recursive cases):

1) $r_1 + r_2$ (union of two regular languages)
2) $r_1 r_2$ (or $r_2 r_1$) (concatenation of two regular languages)
3) r_1^* (or r_2^*) (Kleene Star of any regular languages)

Step 3: No strings except those constructed in above, are allowed to be in Regular Languages

All Finite Languages are Regular

Consider the language L, defined over Σ={a,b}, of strings of length 2, **starting with a**, then L={aa, ab}, may be expressed by the regular expression **aa + ab**.
Hence L, by definition, is a regular language.

It may be noted that if a language contains even thousand words, its Regular Expression may be expressed, placing ' **+** ' between all the words.

Here any special structure of R.E. is not important.

Consider language of all strings of {a, b} of 3 length

L={aaa, aab, aba, abb, baa, bab, bba, bbb},

that may be expressed by a RE

aaa+aab+aba+abb+baa+bab+bba+bbb,

Or

(a+b)(a+b)(a+b)

Defining Languages Method 5 – Finite Automata (FA) or Finite State Machine (FSM)

Definition:

A Finite automaton (FA), is a collection of the followings

1) Finite number of states,
2) having one initial and
3) some (maybe none) final states.
4) Finite set of input letters (Σ) from which input strings are formed.
5) Finite set of transitions *i.e.* for each state and for each input letter there is a transition showing how to move from one state to another.

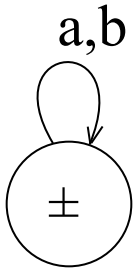
FA for all strings including Λ

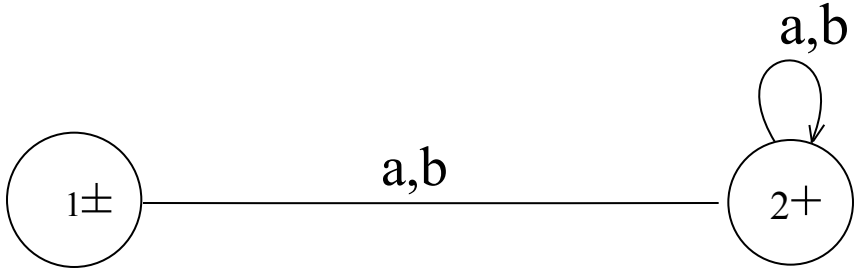
- Consider the Language L, defined over Σ = {a, b} of **all strings including Λ**,
- The language L may be expressed by the following regular expression

$$(a + b)^*$$

- The language L may be accepted by the following FA

- The language L may also be accepted by the following FA





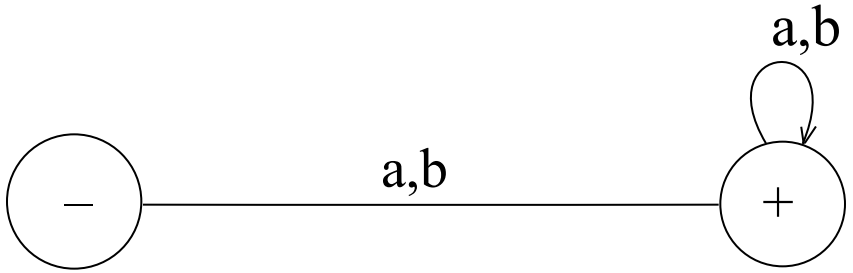
FA for all non empty strings

- Consider the Language L , defined over Σ = {a, b} of **all non empty strings**.

The above language may be expressed by the following regular expression

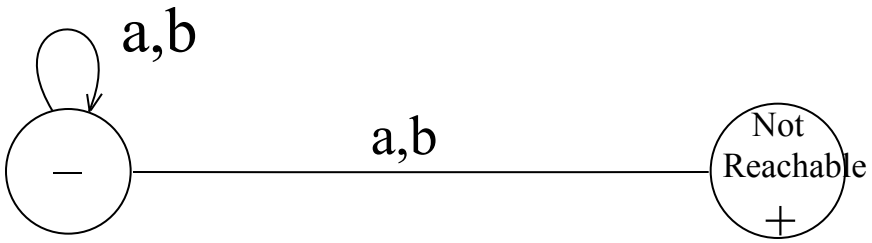
$(a + b)^+$

The language L may be accepted by the following FA



FA that does not accept any string

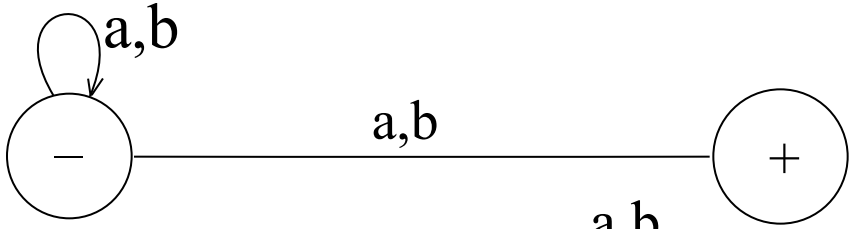
- Consider the following FA, defined over Σ = {a, b}
- Regular Expression = Φ
- Following FA **does not accept any string**. Not Even null string.
- As there is no path starting from initial state and ending in final state.



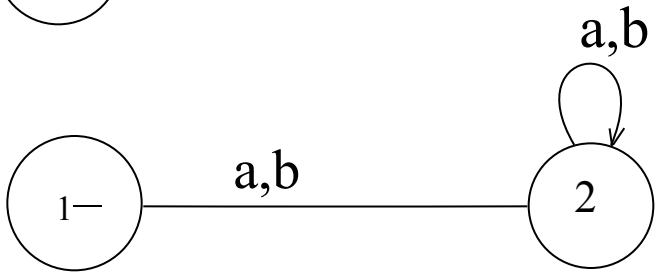
Equivalent FAs

- It is to be noted that two FAs are said to be equivalent, if they accept the same language, as shown in the following FAs.
- FA_1 has already been discussed, while in FA_2, there is no final state and in FA_3, there is a final state but FA_3 is disconnected as the states 2 and 3 are disconnected.
- It may also be noted that the language of strings accepted by FA_1, FA_2 and FA_3 is denoted by the empty set *i.e.* { } OR Ø

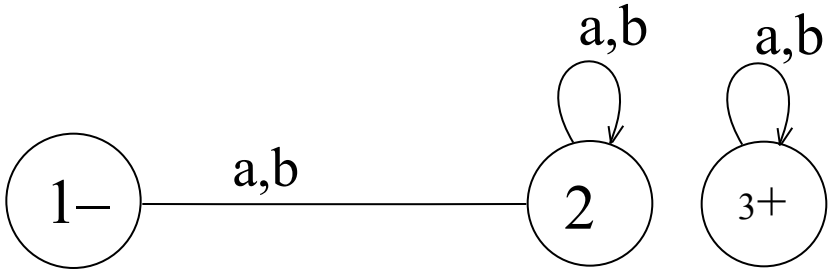
FA_1



FA_2



FA_3



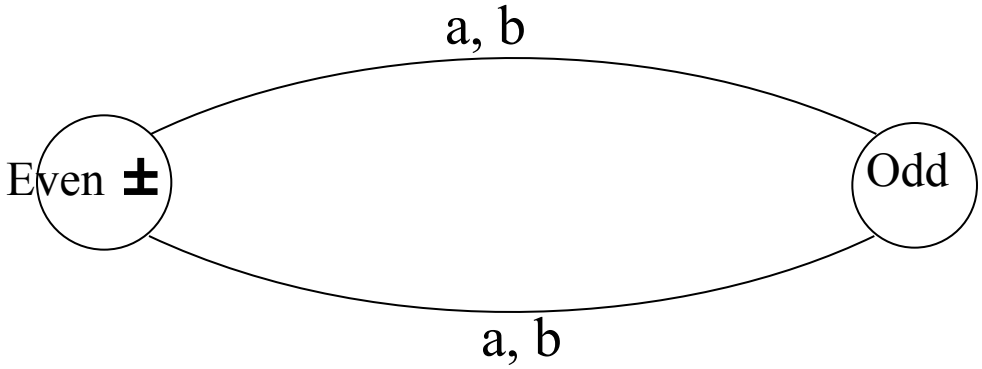
FA for Even Length strings

- All strings of Even Length R. E = **((a+ b) (a + b))***

  States: x, y, where x is both initial and final state.

  Transitions:

  1. At state x reading a or b go to state y.
  2. At state y reading a or b go to state x.



Old States	New States	
	Reading a	Reading b
x ±	y	y
y	x	x

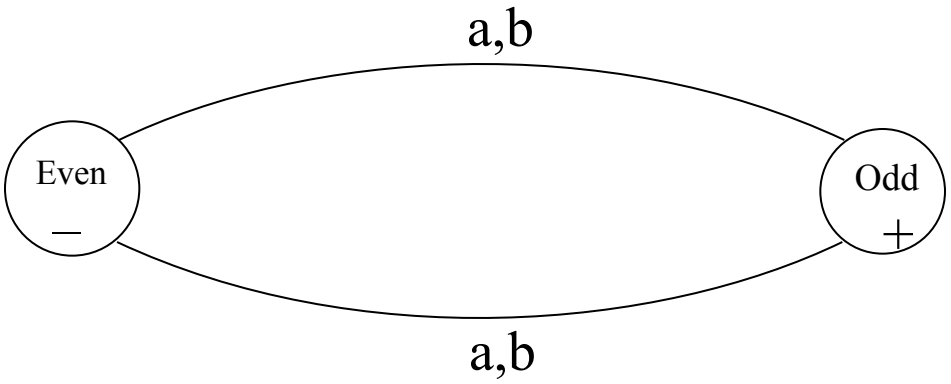
FA for strings of Odd Length

Build an FA for language L of strings, defined over Σ={a, b},
of odd length.

Solution:

The language L may be expressed by RE = **(a+b)((a+b)(a+b))*** or **((a+b)(a+b))*(a+b)**

This language may be accepted by the following FA

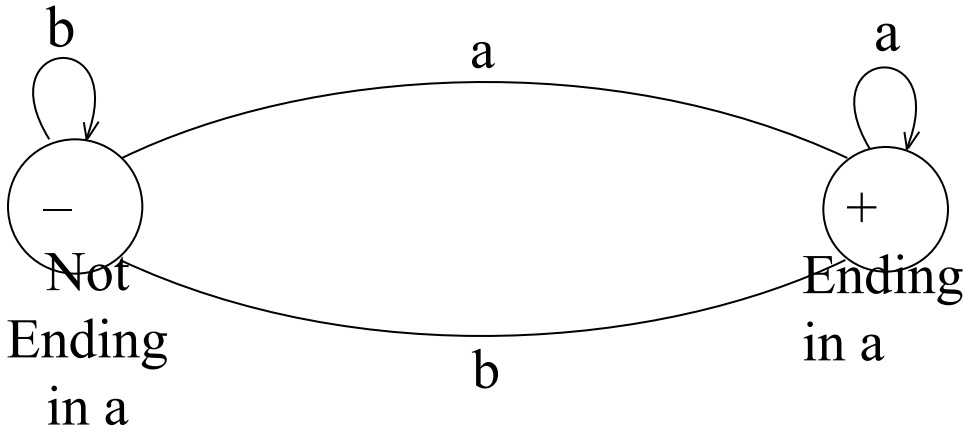


- **Example:**

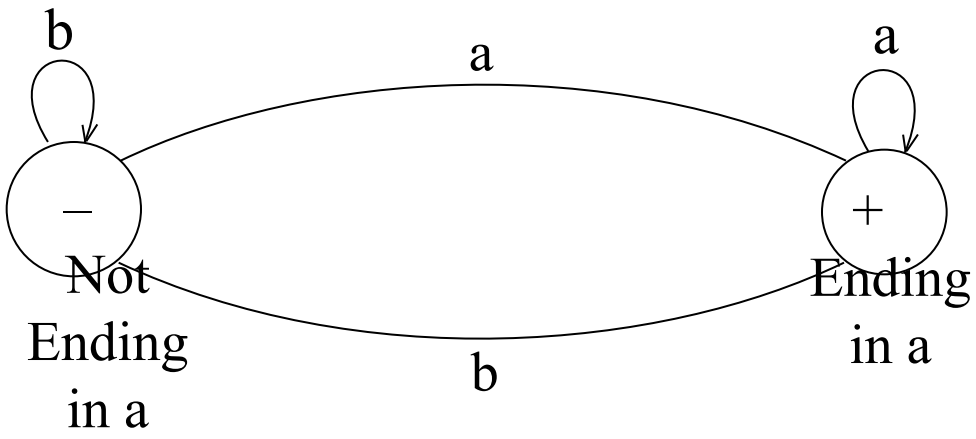
  Consider the language L of strings, defined over Σ={a, b}, **ending in a. T**he language L may be expressed by RE

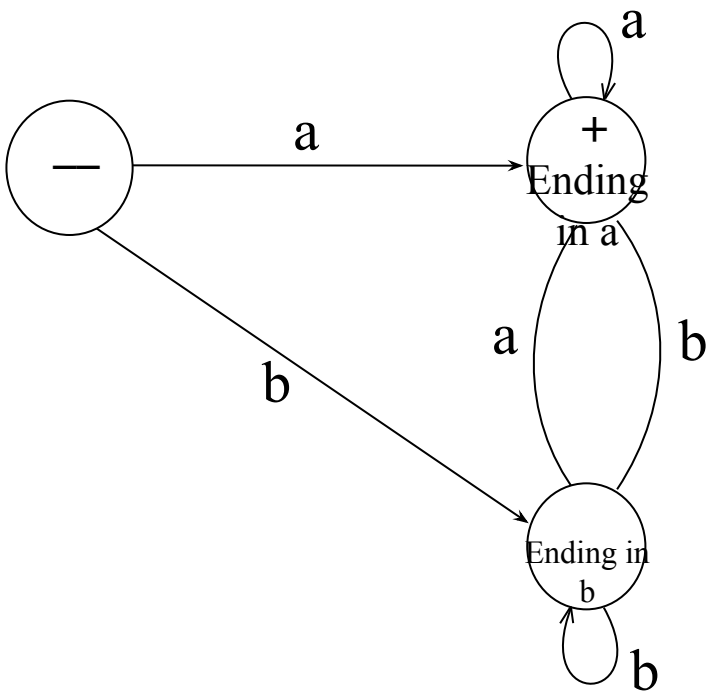
  $(a+b)^*$**a**

This language may be accepted by the following FA



Example continued ...





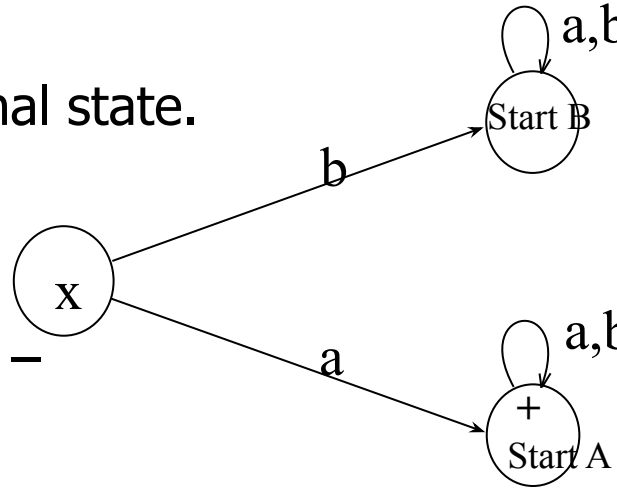
It may be noted that corresponding to a given language there may be more than one FA accepting that language, but for a given FA there is a unique language accepted by that FA.

Example

- The language of strings, defined over Σ={a, b}, **beginning with a**. expressed by R.E **a (a + b)***

Finite Automata

- Σ = {a,b}
- **States:** x, y, z where x is an initial state and z is final state.
- **Transitions:**
  1. At state **x** reading **a** go to state **z**,
  2. At state **x** reading **b** go to state **y**,
  3. At state **y** reading **a**, **b** go to state **y**
  4. At state **z** reading **a**, **b** go to state **z**



Transition Table □

Old States	New States	
	Reading a	Reading b
x -	z	y
y	y	y
z +	z	z

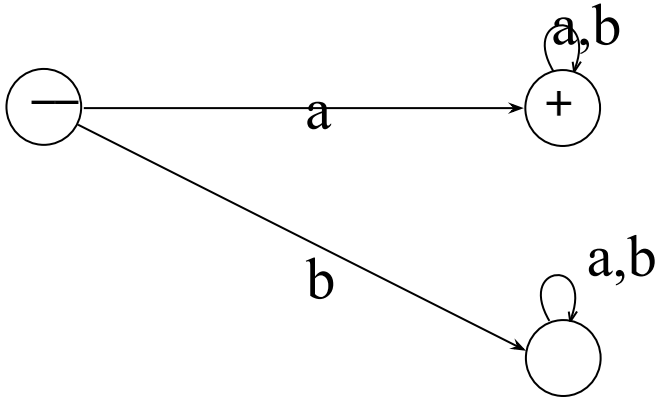
Examples

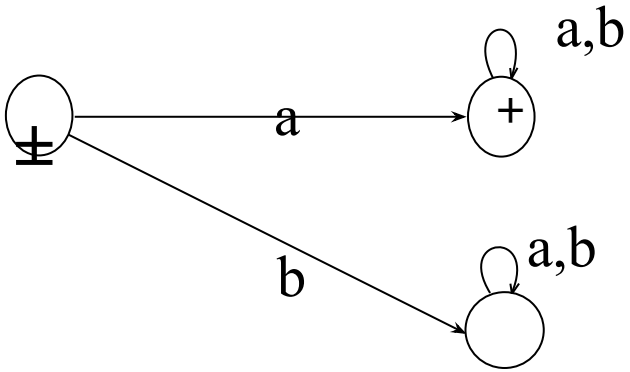
- It is to be noted that given the languages L_1 and L_2 ,where

L_1 = The language of strings, defined over Σ={a, b}, **beginning with a, R.E = a**$(a + b)^*$

L_2 = The language of strings, defined over Σ={a, b}, **not beginning with b, R.E = a** $(a + b)^*$ + **Λ**

The Λ does not belong to L_1 while it does belong to L_2 . This fact may be depicted by the corresponding transition diagrams of L_1 and L_2.



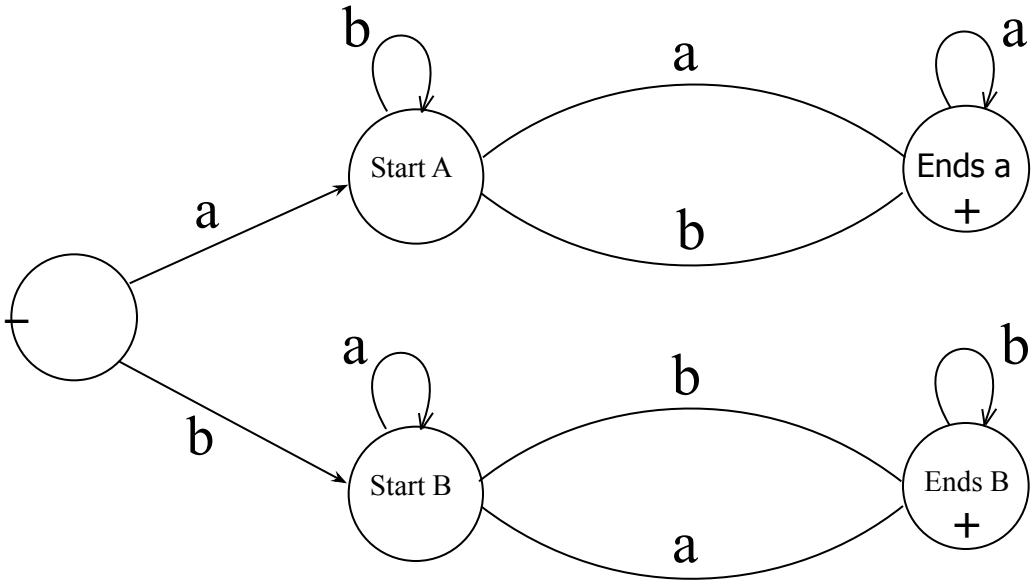


Example

- Consider the Language L of Strings of **length two or more**, defined over Σ = {a, b}, **beginning with and ending in same letters,** regular expression = a (a + b)* a + b (a + b)* b

It is to be noted that if the condition on the length of string is not imposed in the above language then **the strings a and b will then belong to the language.**

This language L may be accepted by the following FA



FA for beginning and ending in same letters

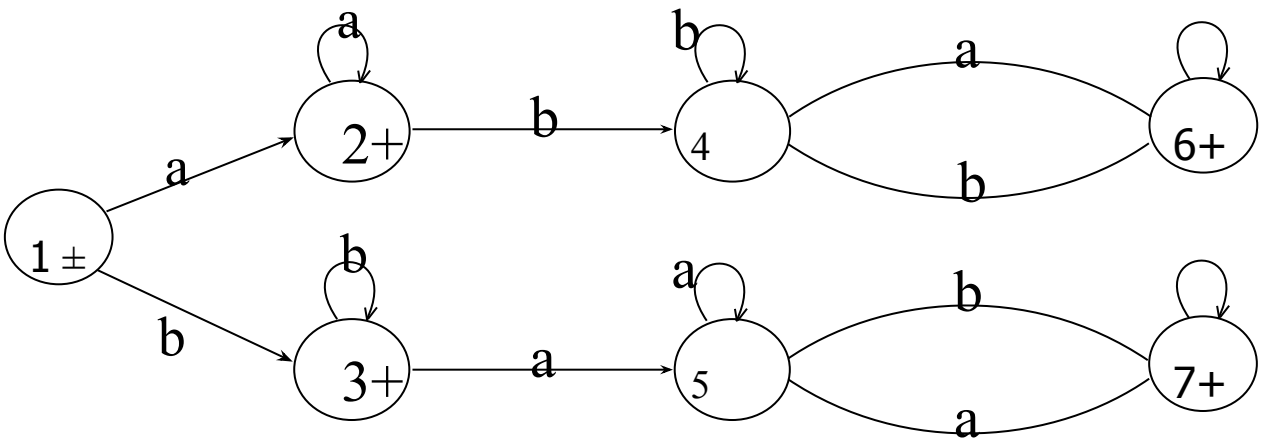
- Build an FA accepting Language L of Strings, defined over Σ = {a, b}, **beginning with and ending in same letters (no restriction on length)**

Solution:

The language L may be expressed by the following regular expression

$$\Lambda + \mathbf{a + b + a(a + b)^{*}a + b(a + b)^{*}b}$$

This language L may be accepted by the following FA



FA for beginning and ending in different letters

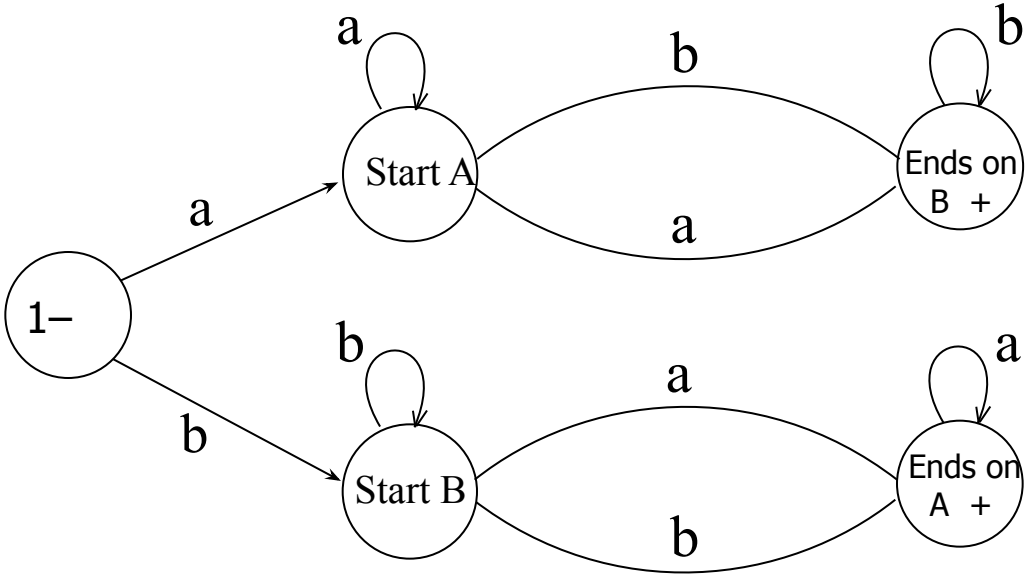
Consider the Language L of Strings , defined over Σ = {a, b}, **beginning with and ending in different letters.**

Solution:

The language L may be expressed by the following regular expression

$$\mathbf{a\,(a + b)^{*}\,b + b\,(a + b)^{*}\,a}$$

This language may be accepted by the following FA



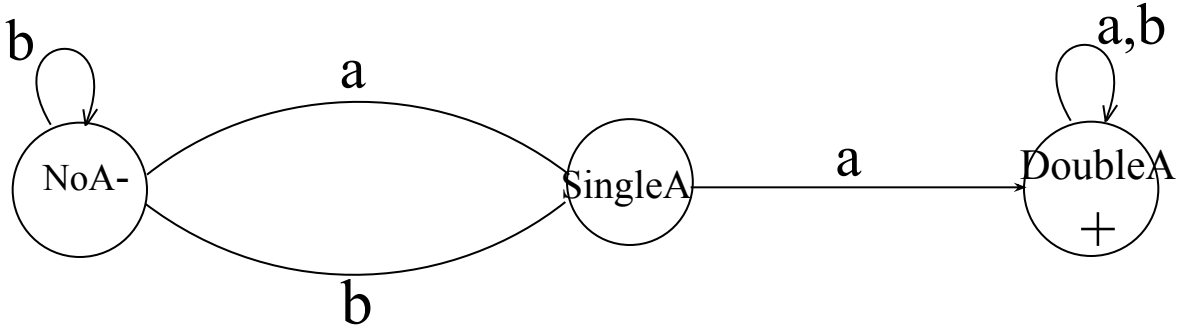
Language Containing Double A

Consider Language L of strings , defined over Σ = {a, b}, **containing double a.**

The language L may be expressed by following R.E

$(a+b)^*$ **(aa)** $(a+b)^*$.

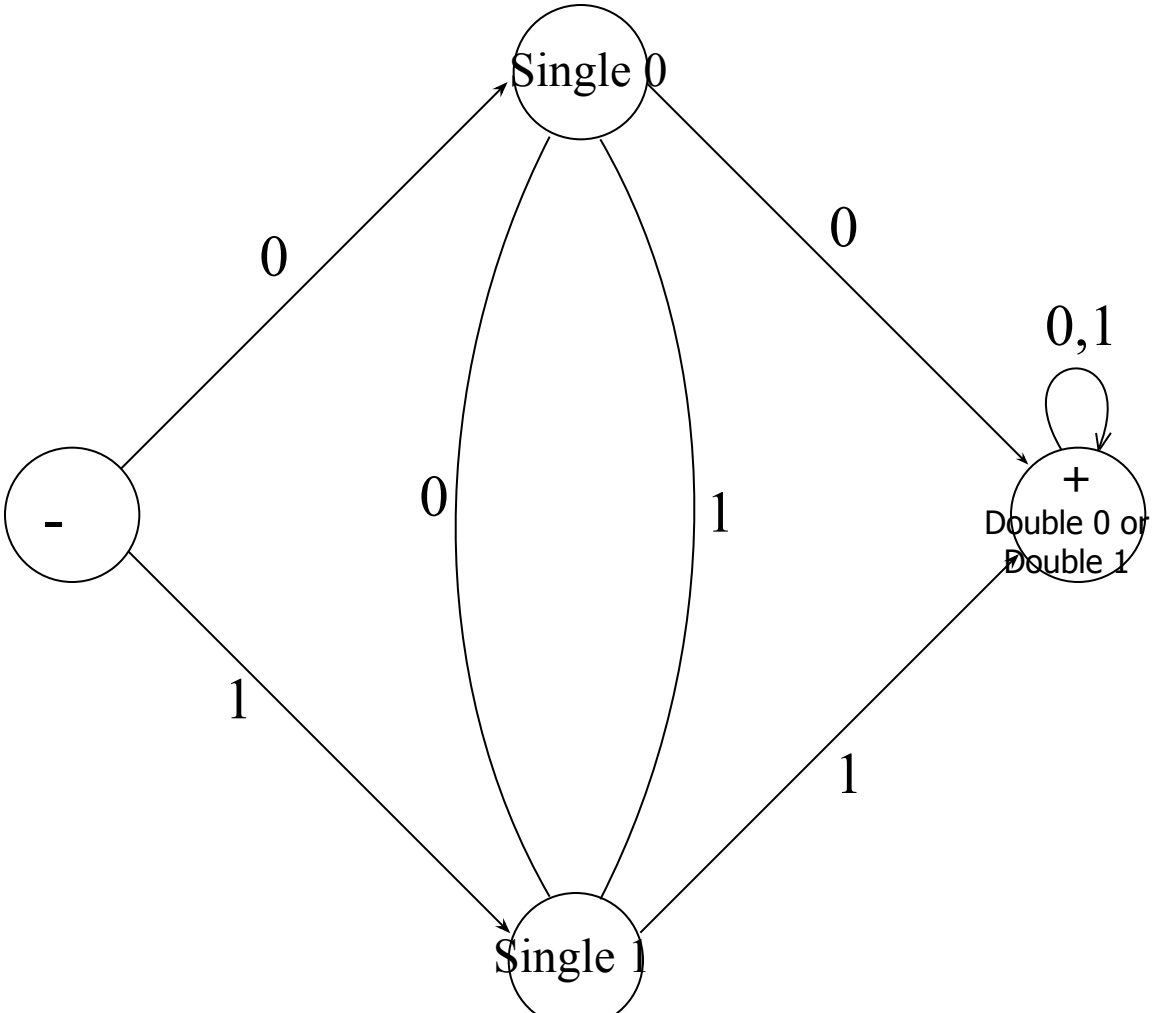
This language may be accepted by the following FA



FA having double 0's or double 1's

Consider language L of strings, defined over Σ={0, 1}, **having double 0's or double 1's,** The language L may be expressed by regular expression $(0+1)^*$ **(00 + 11)** $(0+1)^*$

This language may be accepted by the following FA

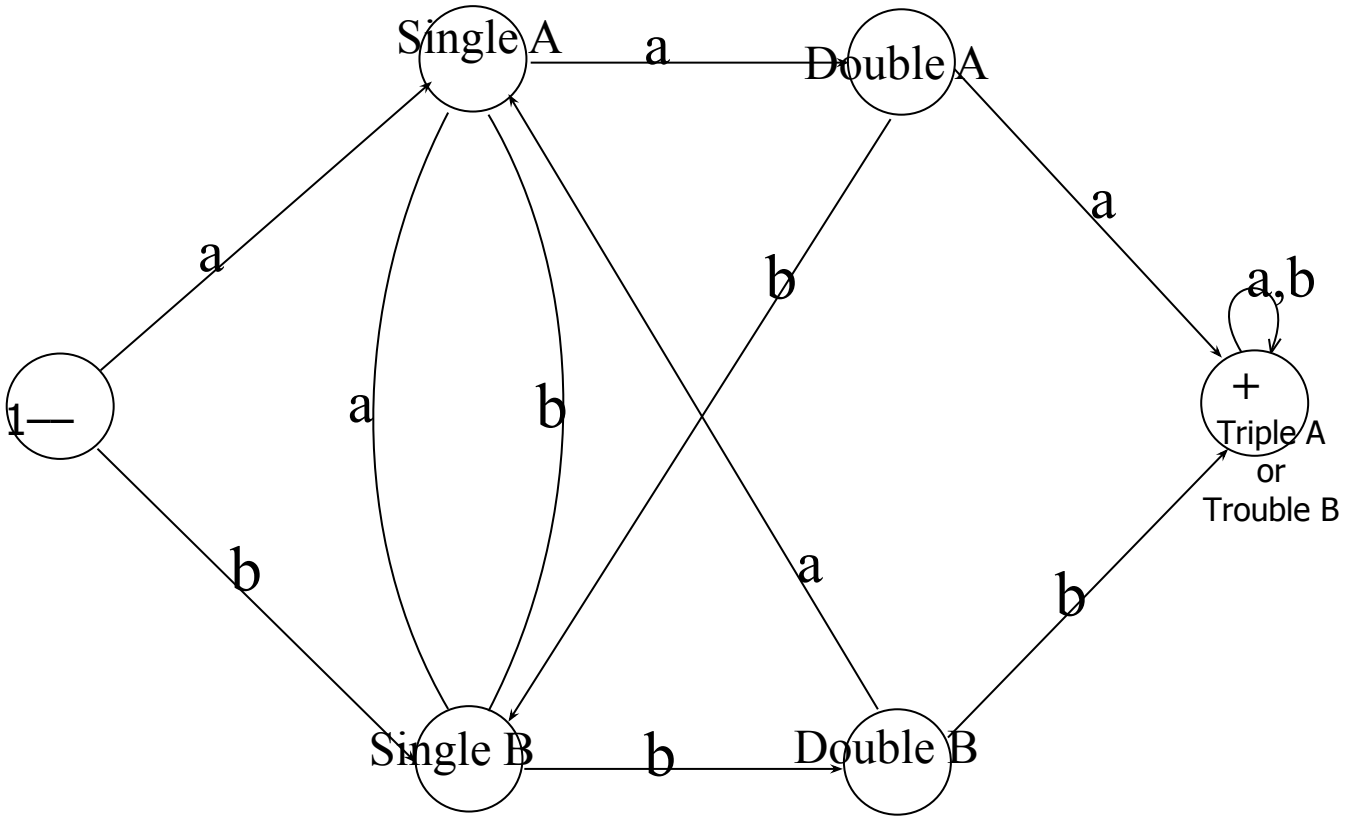


FA for Triple A's Triple B's

Consider the language L of strings, defined over Σ={a, b}, **having triple a's or triple b's.** The language L may be expressed by RE

$(a+b)^*$ **(aaa + bbb)** $(a+b)^*$

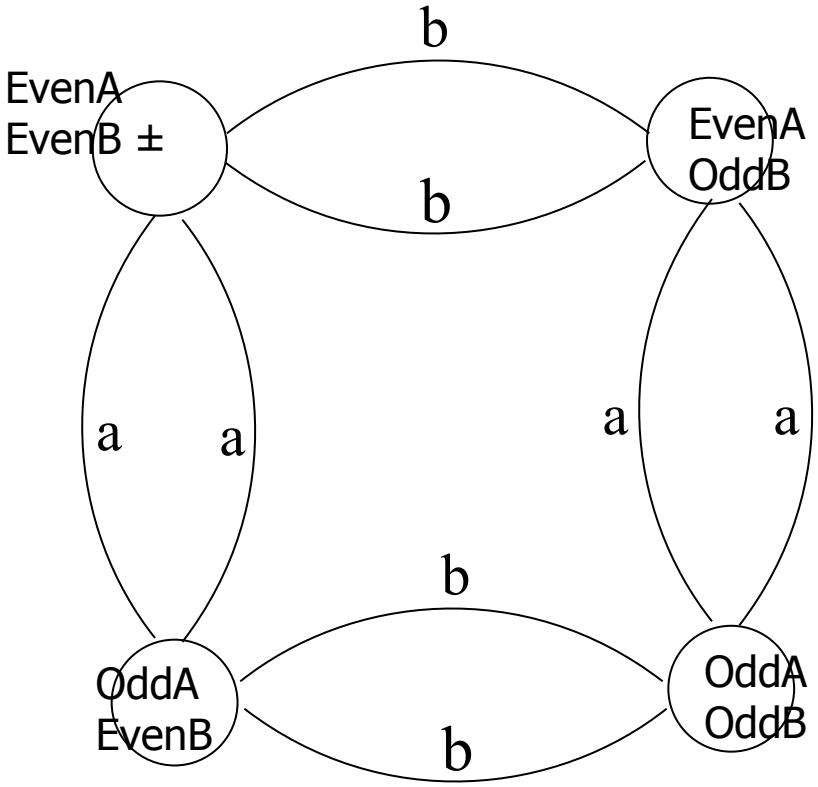
This language may be accepted by the following FA



Even Even FA

- Consider the **EVEN-EVEN** language, defined over Σ={a, b}. As discussed earlier that **EVEN-EVEN** language can be expressed by the regular expression **(aa+bb+(ab+ba)(aa+bb)*(ab+ba))***

  EVEN-EVEN language may be accepted by the following FA



Finite Automaton with Output

- Finite automaton discussed so far, is just associated with the RE or the language.

- There is a question whether does there exist an FA which generates an output string corresponding to each input string ? The answer is yes.

- There are two types of Finite Automata with output.

1) Moore machine
2) Mealy machine

Moore machine

A Moore machine consists of the following

1. A **finite set of states {**q_0, q_1, q_2, …} where **q_0 is initial state**.

2. An **alphabet** of letters Σ = {a,b,c,…} from which the **input strings** are formed.

3. An **alphabet** □={x,y,z,…} **of output characters** from which output strings are generated.

4. A **transition table** that shows for each state and each input letter what state is entered the next.

5. An **output table** that shows what character is printed by each state as it is entered.

Note

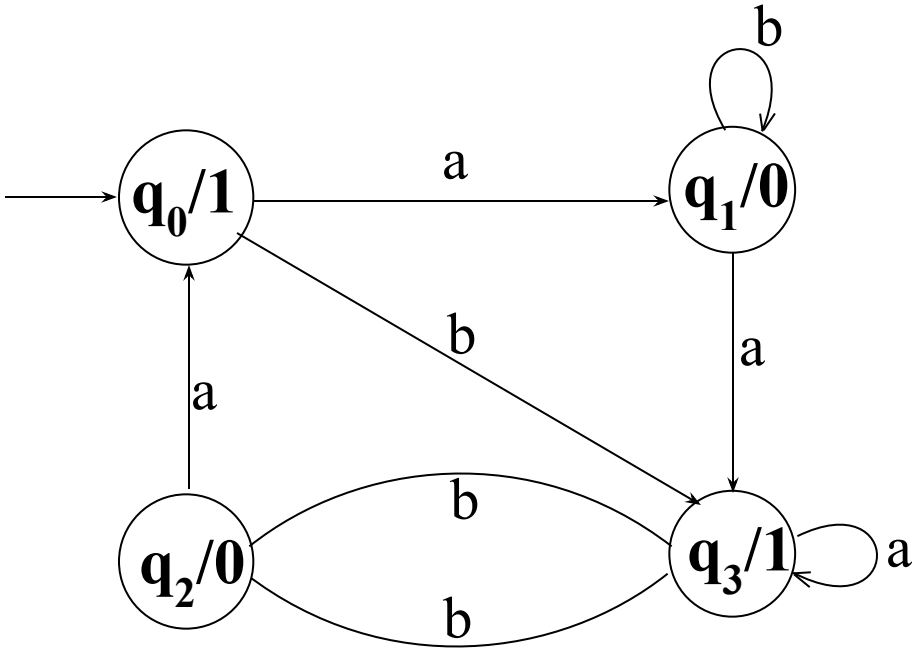
Since in Moore machine no state is designated to be a final state, so there is **no question of accepting** any language by Moore machine.

However in some cases the **relation between an input string and the corresponding output** string may be identified by the Moore machine.

Moreover, the **state to be initial is not important** as if the machine is used several times and is restarted after some time, the machine will be started from the state where it was left off.

Example – Moore Machine

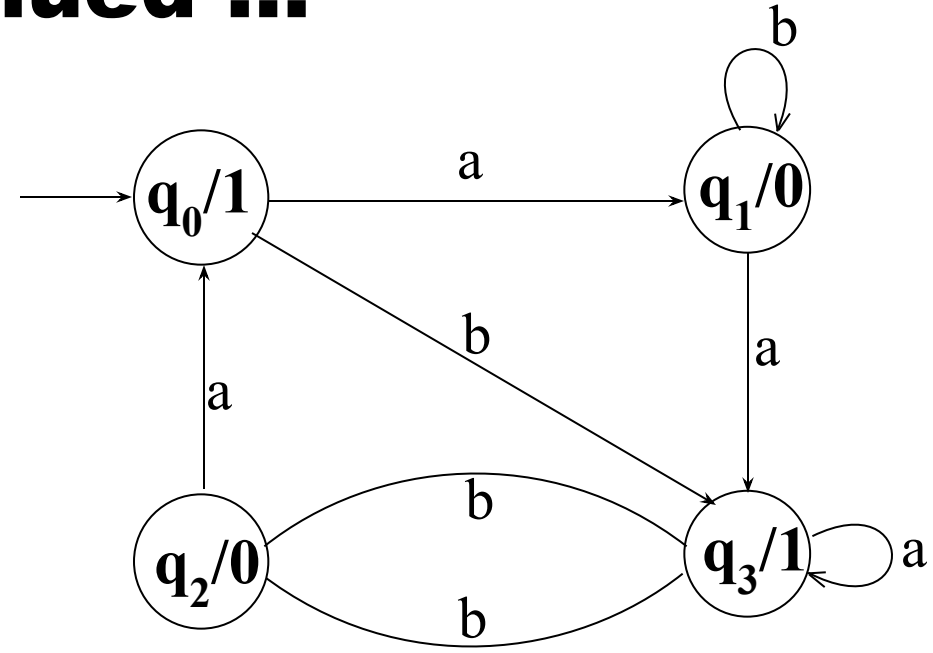
Consider the following Moore machine having the states q_0, q_1, q_2, q_3 where q_0 is the start state and $\Sigma = \{a,b\}$, $\square=\{0,1\}$



Example continued ...

Old States	New States after reading		Characters to be printed
	a	**b**	
q_0-	q_1	q_3	1
q_1	q_3	q_1	0
q_2	q_0	q_3	0
q_3	q_3	q_2	1

Example continued ...



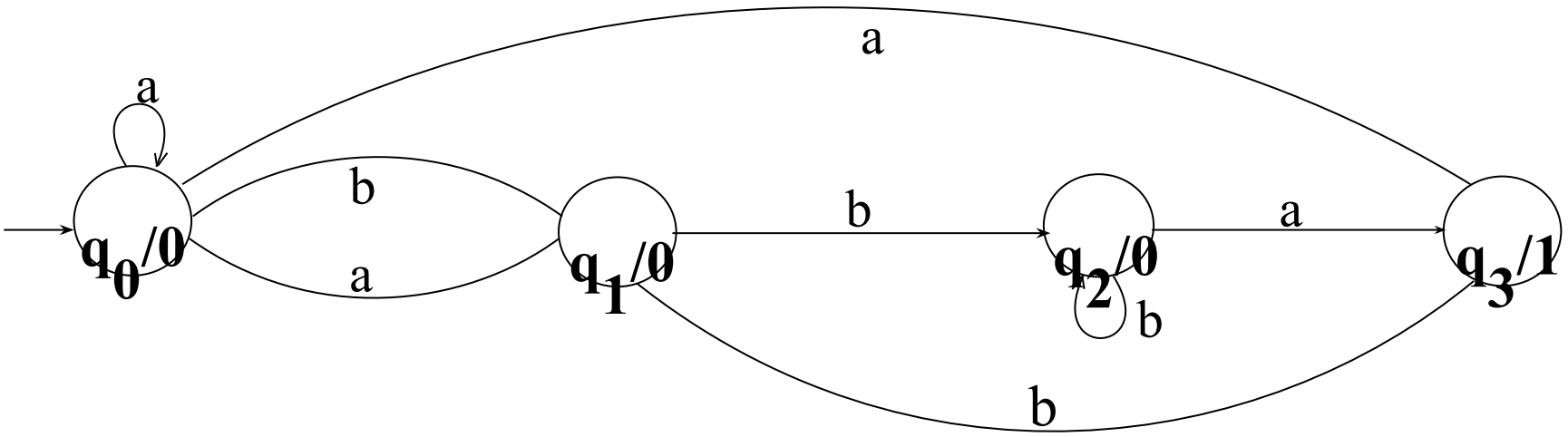
It is to be noted that the states are labeled along with the characters to be printed. Running the string abbabbba over the above machine, the corresponding output string will be 100010101, which can be determined by the following table as well

Example continued ...

Input		a	b	b	a	b	b	b	a
State	q_0	q_1	q_1	q_1	q_3	q_2	q_3	q_2	q_0
output	1	0	0	0	1	0	1	0	1

It may be noted that the length of output string is l more than that of input string as the initial state prints out the extra character 1, before the input string is read.

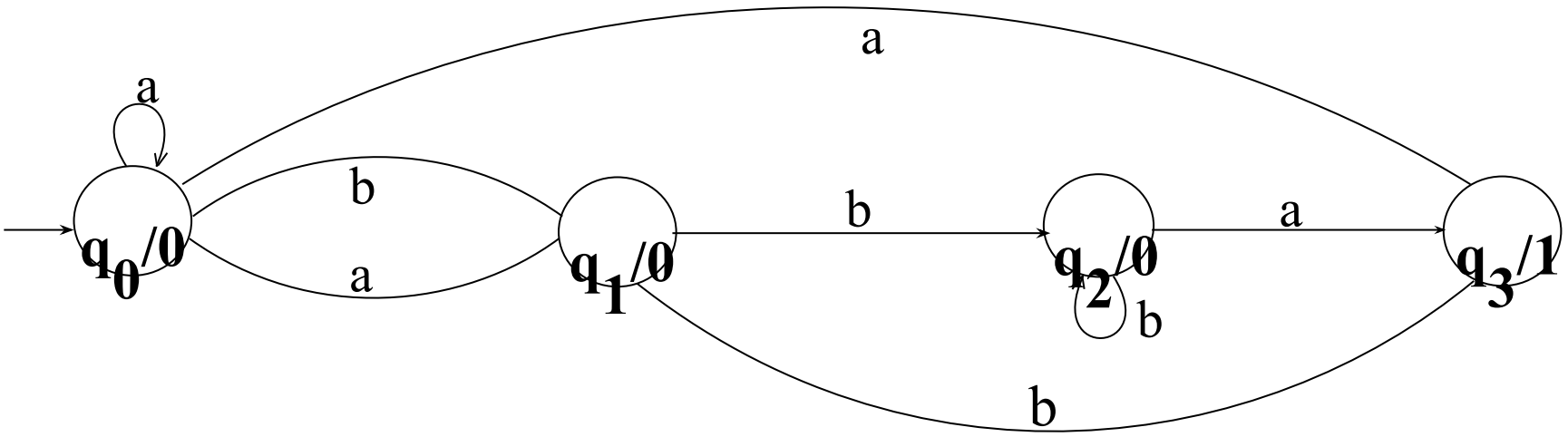
Example 2 Moore Machine



if string **bbbabaabbaa** is run, output string will be **000010000010**, as shown below

Input		b	b	b	a	b	a	a	b	b	a	a
State	q_0	q_1	q_2	q_2	q_3	q_1	q_0	q_0	q_1	q_2	q_3	q_0
output	0	0	0	0	1	0	0	0	0	0	1	0

Example 2 Moore Machine …



q_3 is the only state which prints out the character 1 which shows that the moment the state q_3 is entered, the machine will print out 1.

To enter the state q_3, starting from q_0 the string must contain bba.

to enter the state q_3 once more the string must contain another substirng bba.

In general the input string will visit the state q_3 as many times as the number of substring bba occurs in the input string.

Thus the number of **1's in an output** string will be **same** as the **number of** substring **bba** occurs in the corresponding input string.

Mealy machine

A Mealy machine consists of the following

1. A finite **set of states** $\{q_0, q_1, q_2, \ldots\}$ where $\mathbf{q_0}$ **is initial state**.
2. An **alphabet of input letters** $\Sigma = \{a,b,c,\ldots\}$ from which the input strings are formed.
3. An alphabet $\square=\{x,y,z,\ldots\}$ of **output characters** from which output strings are generated.
4. A **pictorial representation with states and directed edges labeled by an input letter along with an output character**. The directed edges also show how to go from one state to another corresponding to every possible input letter.

It is not possible to give transition table in this case.

Mealy machine continued …

Note:

It is to be noted that since, similar to Moore machine, in Mealy machine **no state is designated to be a final state**, so there is **no question of accepting** any language by Mealy machine.

However in some cases the **relation between an input string and the corresponding output** string may be identified by the Mealy machine.

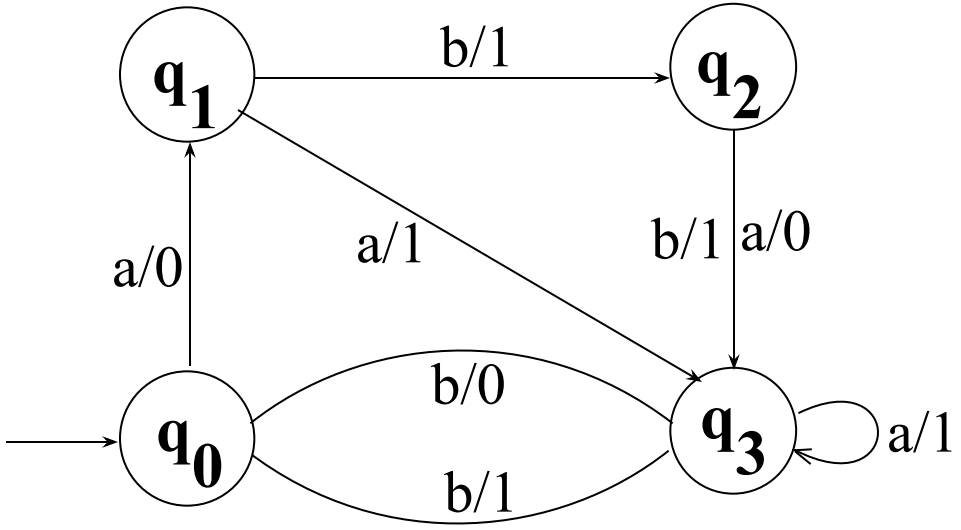
Moreover, the state to be **initial is not important** as if the machine is used several times and is restarted after some time, the machine will be started from the state where it was left off.

Example – Mealy Machine

Consider the following Mealy machine having the states q_0, q_1, q_2, q_3 , where q_0 is the start state and

$\Sigma = \{a,b\}$, $\square=\{0,1\}$

Running the string **abbabbba** over the above machine, the corresponding output string will be **11011010**, which can be determined by the following table as well



Example continued ...

Running the string **abbabbba** over the above machine, the corresponding output string will be **11011010**, which can be determined by the following table as well

It may be noted that in Mealy machine, the length of output string is equal to that of input string.

Input		a	b	b	a	b	b	b	a
States	q_0	q_1	q_2	q_3	q_3	q_0	q_3	q_0	q_1
output		0	1	1	1	1	0	1	0

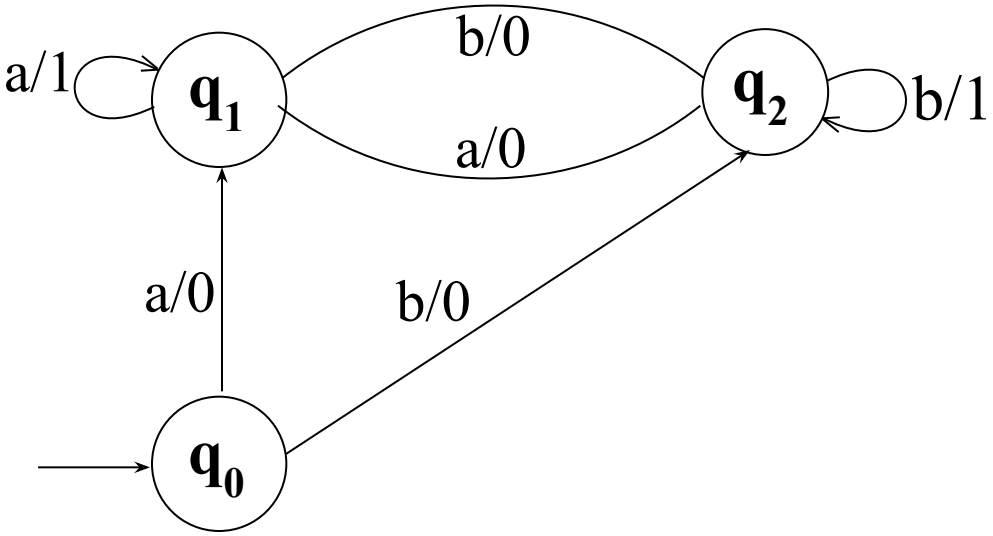
Example 2 – Mealy Machine

Consider the following Mealy machine having the states q_0, q_1, q_2 , where q_0 is the start state and

$\Sigma = \{a,b\}$, $\square=\{0,1\}$

It is observed that in this Mealy machine, if in the output string the **nth character is 1**, it shows that the nth letter in the input string is the **second in the pair of double letter**.

For baba**a**bab**b**a as input string the machine will print 0000**1**000**1**0.

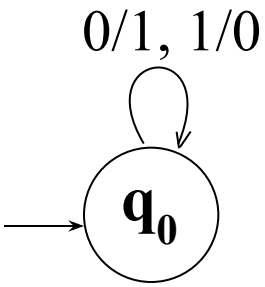


Complementing Mealy Machine

Consider the following Mealy machine having the only state q_0 as the start state and

$\Sigma = \{0,1\}$,

$\square = \{0,1\}$



If **0011010** is run on this machine then the corresponding output string (complement) will be **1100101**.

Constructing the incrementing machine

In the previous example of complementing machine, it has been observed that the input string and the corresponding output string are 1's complement of each other.

There is a question whether the Mealy machine can be constructed, so that the output string is increased, in magnitude, by 1 than the corresponding input string ? The answer is yes.

This machine is called the incrementing machine.

Constructing the incrementing machine continued …

Before the incrementing machine is constructed, consider how 1 is added to a binary number.

Since, if two numbers are added, the addition is performed from right to left, so while increasing the binary number by 1, the string (binary number) must be read by the corresponding Mealy machine from right to left, and hence the output string (binary number) will also be generated from right to left.

Constructing the incrementing machine continued …

Consider the following additions

a) 10010111**0**

+ 1

10010111**1**

It may be observed from the above that

a) If the right most bit of binary number, to be incremented, is 0, the output binary number can be obtained by converting the right most bit to 1 and remaining bits unchanged.

Constructing the incrementing machine continued …

b) 1001100111
$+\ 1$
$100110\mathbf{1000}$

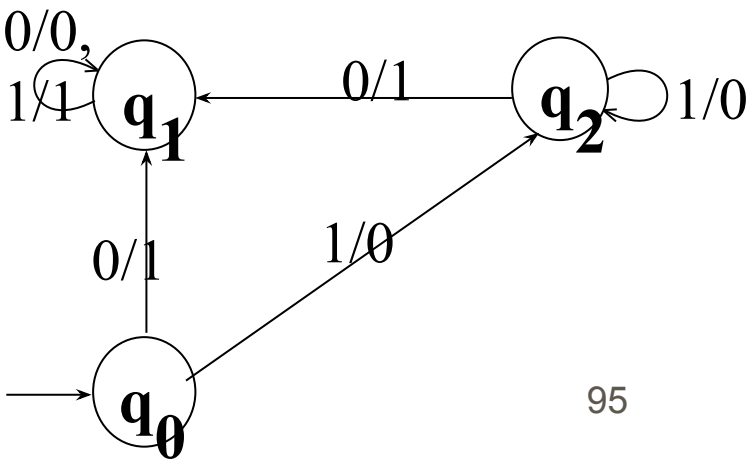
b) If the right most bit of binary number is 1 then the output can be obtained, converting that 1 along with all its concatenated 1's to 0's, then converting the next 0 to 1 and remaining bits unchanged.

The observations (a) and (b) help to construct the following Incrementing (Mealy) machine.

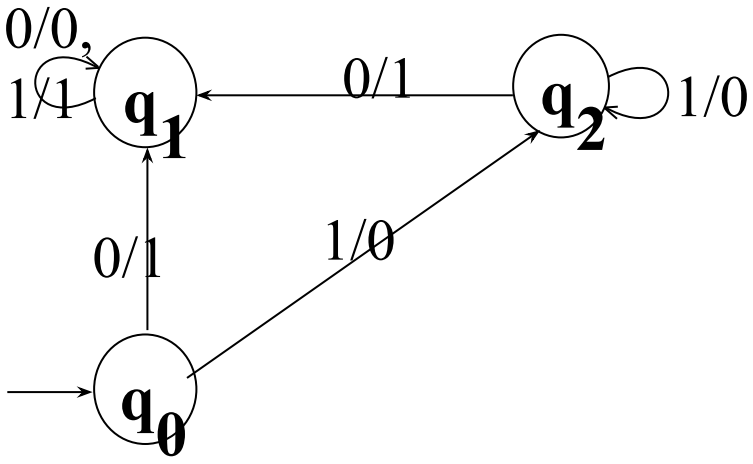
Constructing the incrementing machine continued …

The Mealy machine have the states $\{q_0, q_1, q_2\}$ where q_0 is the start state and $\Sigma = \{0,1\}$, $\square = \{0,1\}$

It may be observed that, in the incrementing machine, if 0 is read at initial state $\mathbf{q_0}$, that 0 is converted to 1 and a no change state $\mathbf{q_1}$ (no carry state) is entered where all 0's and all 1's remain unchanged.
If 1 is read at initial state, that 1 is converted to 0 and the state q_2**(**owe carry state**)** is entered, where all 1's are converted to 0's and at that state if 0 is read that 0 is converted to 1 and the machine goes to no change state.
If the strings 100101110 and 1001100111 are run over this machine, the corresponding output strings will be 100101111 and 1001101000 respectively.



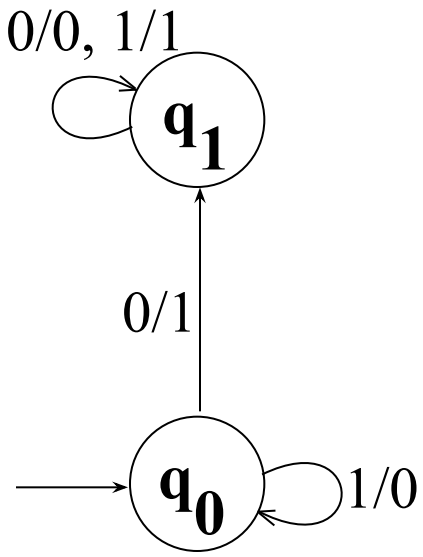
Overflow



It is to be noted that if the string 111111 is run over this incrementing machine, the machine will print out 000000, which is not increased in magnitude by 1.

Such a situation is called an overflow situation, as the length of output string will be same as that of input string.

Incrementing machine with two states



Applications of Incrementing and Complementing machines

1’s complementing and incrementing machines which are basically Mealy machines are very much helpful in computing.

The incrementing machine helps in building a machine that can perform the addition of binary numbers.

Using the complementing machine along with incrementing machine, one can build a machine that can perform the subtraction of binary numbers, as shown in the following method

Subtracting a binary number from another

Method: To subtract a binary b from a binary number a

1. Add 1's complement of b to a (ignoring the overflow, if any)
2. Increase the result, in magnitude, by 1 (use the incrementing machine). Ignoring the overflow if any.

Note: If there is no overflow in (1). Take 1's complement once again in (2), instead. This situation occurs when b is greater than a, in magnitude. Following is an example of subtraction

Example

To subtract the binary number 101 from the binary number 1110, let

a = 1110 and b = 101 = 0101.

(Here the number of digits of b are equated with that of a)

i) Adding 1's complement (1010) of b to a.

1110

+1010

11000 which gives 1000 (ignoring the overflow)

Example continued …

ii) Using the incrementing machine, increase the above result 1000, in magnitude, by 1

1000

+1

1001 which is the same as obtained by ordinary subtraction.

Note

It may be noted that the above method of subtraction of binary numbers may be applied to subtraction of decimal numbers with the change that 9's complement of b will be added to a, instead in step (1). Following is the task in this regard

Task

Subtract 39 from 64

Solution: Taking a=64 and b=39.

i) Adding 9's complement (60) of b to a.

64

+60

124 which gives 24 (ignoring the overflow)

ii) Increasing the above result 24, in magnitude, by 1 24

+1

25 which is the same as obtained by ordinary subtraction.

Equivalent machines

Two machines are said to be **equivalent** if they print the same output string when the same input string is run on them.

Remark: Two Moore machines may be equivalent. Similarly two Mealy machines may also be equivalent, but a Moore machine can't be equivalent to any Mealy machine. However, ignoring the extra character printed by the Moore machine, there exists a Mealy machine which is equivalent to the Moore machine.

Theorem

Statement:

For every Moore machine there is a Mealy machine that is equivalent to it (ignoring the extra character printed by the Moore machine).

Proof: Let M be a Moore machine, then shifting the output characters corresponding to each state to the labels of corresponding incoming transitions, machine thus obtained will be a Mealy machine equivalent to M.

Following is a note

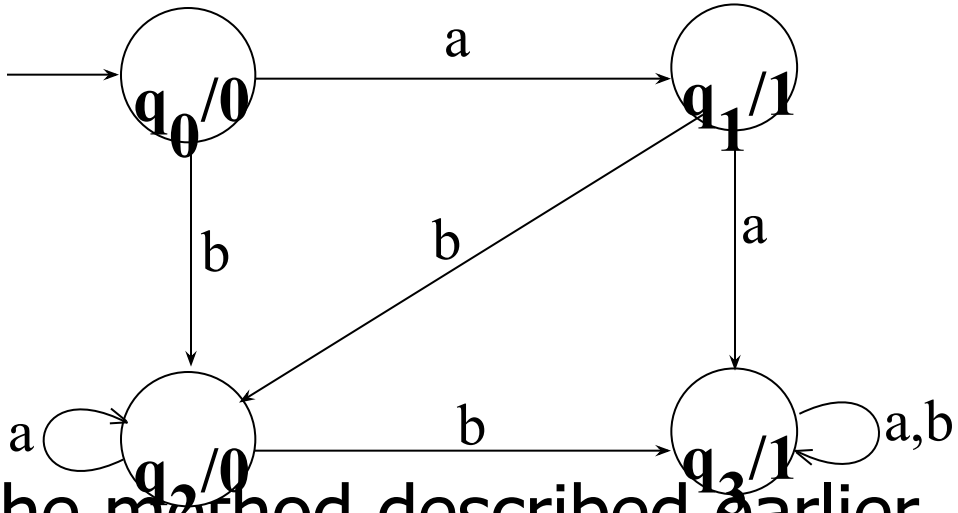
Note

It may be noted that while converting a Moore machine into an equivalent Mealy machine, the output character of a state will be ignored if there is no incoming transition at that state. A loop at a state is also supposed to be an incoming transition.

Following is the example of converting a Moore machine into an equivalent Mealy machine

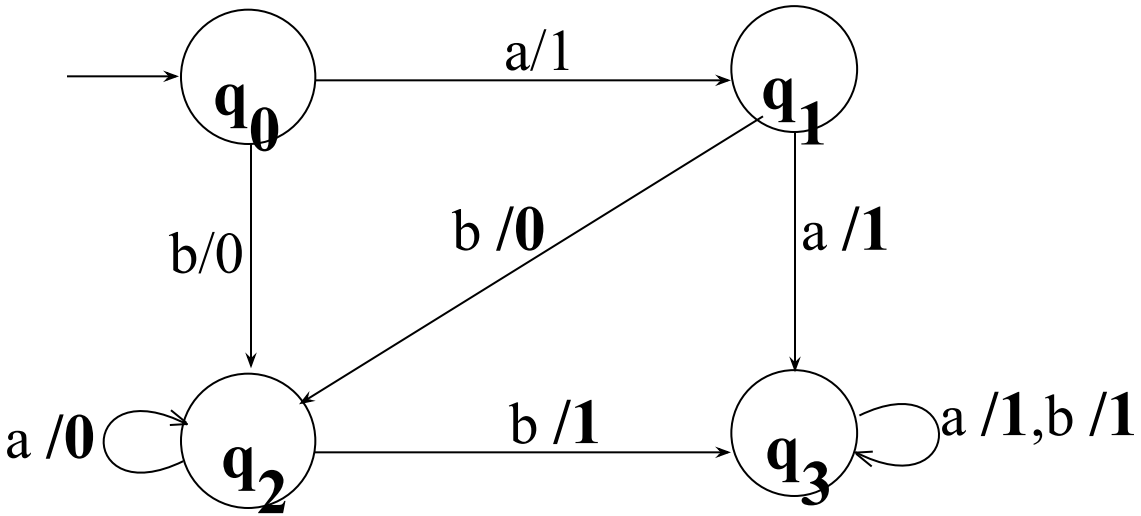
Example

Consider the following Moore machine



Using the method described earlier, the above machine may be equivalent to the following Mealy machine

Example continued …



Running the string abbabbba on both the machines, the output string can be determined by the following table

Example continued …

Input		a	b	b	a	b	b	b	a
States	q_0	q_1	q_2	q_3	q_3	q_3	q_3	q_3	q_3
Moore	0	1	0	1	1	1	1	1	1
Mealy		1	0	1	1	1	1	1	1

Theorem Statement:

For every Mealy machine there is a Moore machine that is equivalent to it (ignoring the extra character printed the Moore machine).

Proof: Let M be a Mealy machine. At each state there are two possibilities for incoming transitions

1. The incoming transitions have the same output character.
2. The incoming transitions have different output characters.

Proof continued …

If all the transitions have same output characters, then shift that character to the corresponding state.

If all the transitions have different output characters, then the state will be converted to as many states as the number of different output characters for these transitions, which shows that if this happens at state q_i then q_i will be converted to q_i^1 and q_i^2 *i.e.* if at q_i there are the transitions with two output characters then q_i^1 for one character and q_i^2 for other character.

Proof continued …

Shift the output characters of the transitions to the corresponding new states q_i^1 and q_i^2. Moreover, these new states q_i^1 and q_i^2 should behave like q_i as well. Continuing the process, the machine thus obtained, will be a Moore machine equivalent to Mealy machine M.

Following is a note

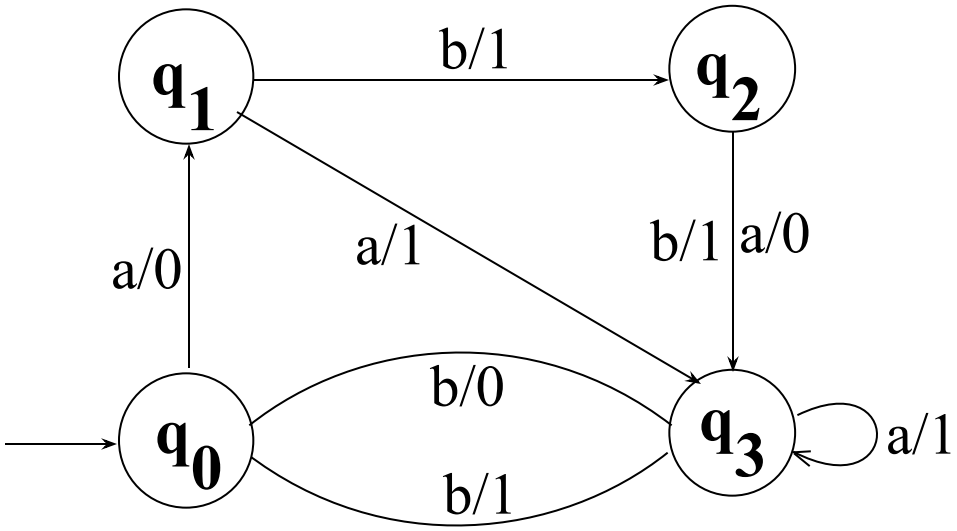
Note

It may be noted that if there is no incoming transition at certain state then any of the output characters may be associated with that state.

It may also be noted that if the initial state is converted into more than one new states then only one of these new states will be considered to be the initial state. Following is an example

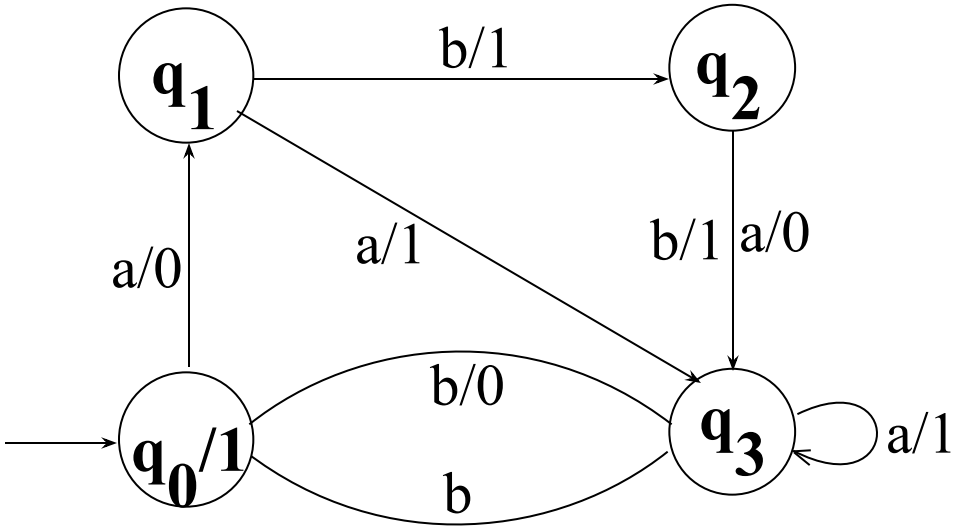
Example

Consider the following Mealy machine



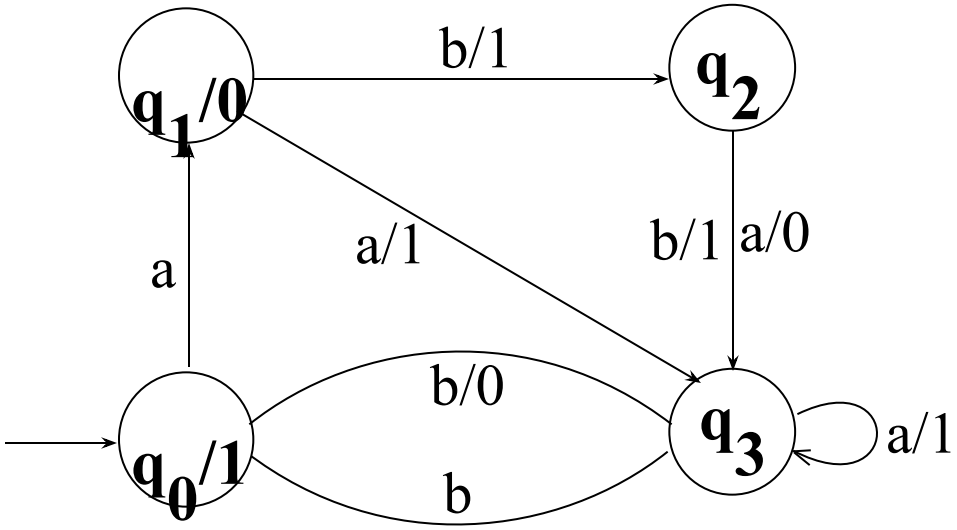
Example continued ...

Shifting the output character 1 of transition b to q_0



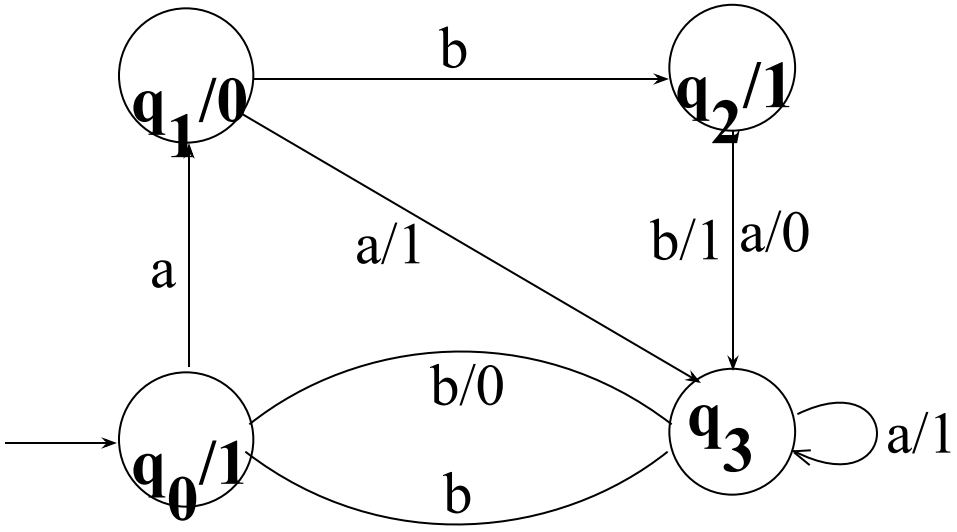
Example continued …

Shifting the output character 0 of transition a to $\mathbf{q_1}$



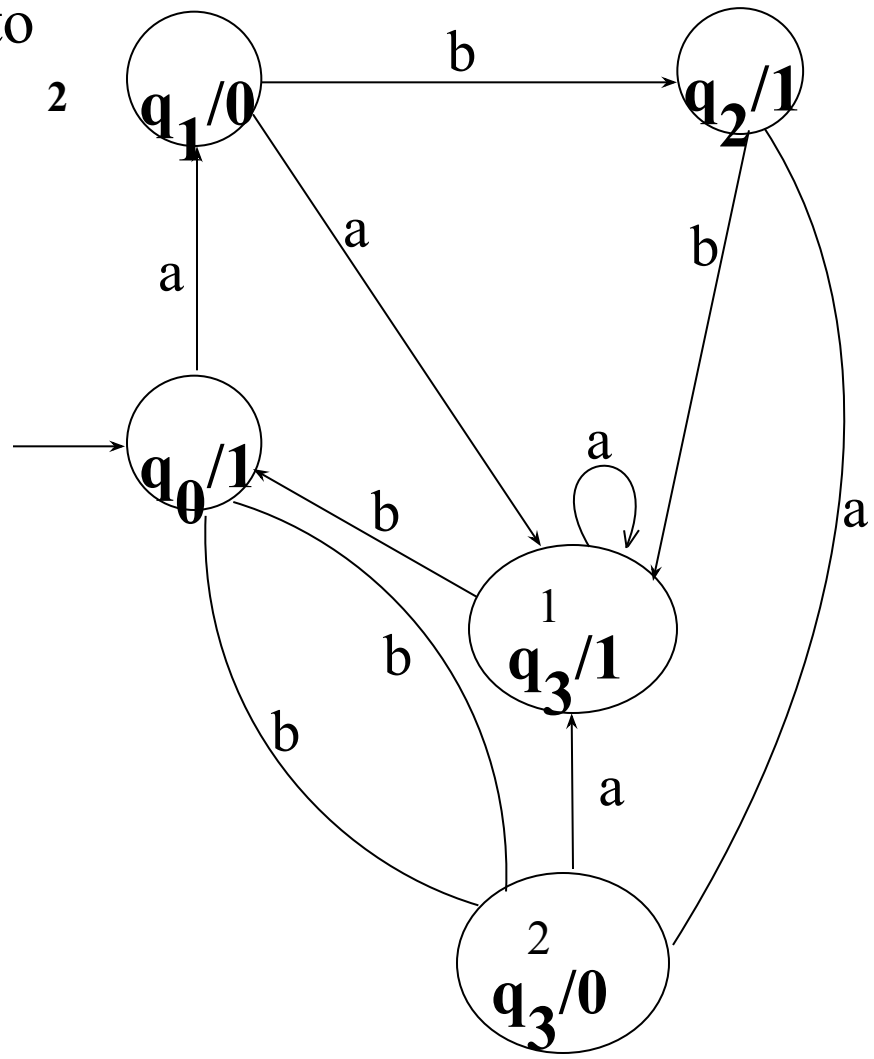
Example continued …

Shifting the output character 1 of transition b to q_2



Example continued ...

Splitting q_3 into q_3 and q_3



Example continued …

Running the string abbabbba on both the machines, the output strings can be determined by the following table

Input		a	b	b	a	b	b	b	a
States	q_0	q_1	q_2	q_3	q_3	q_0	q_3	q_0	q_1
Mealy		0	1	1	1	1	0	1	0
Moore	1	0	1	1	1	1	0	1	0

Solution of the Task

Subtract 39 from 64

Solution: Taking a=64 and b=39.

i) Adding 9's complement (60) of b to a.

64

+60

124 which gives 24 (ignoring the overflow)

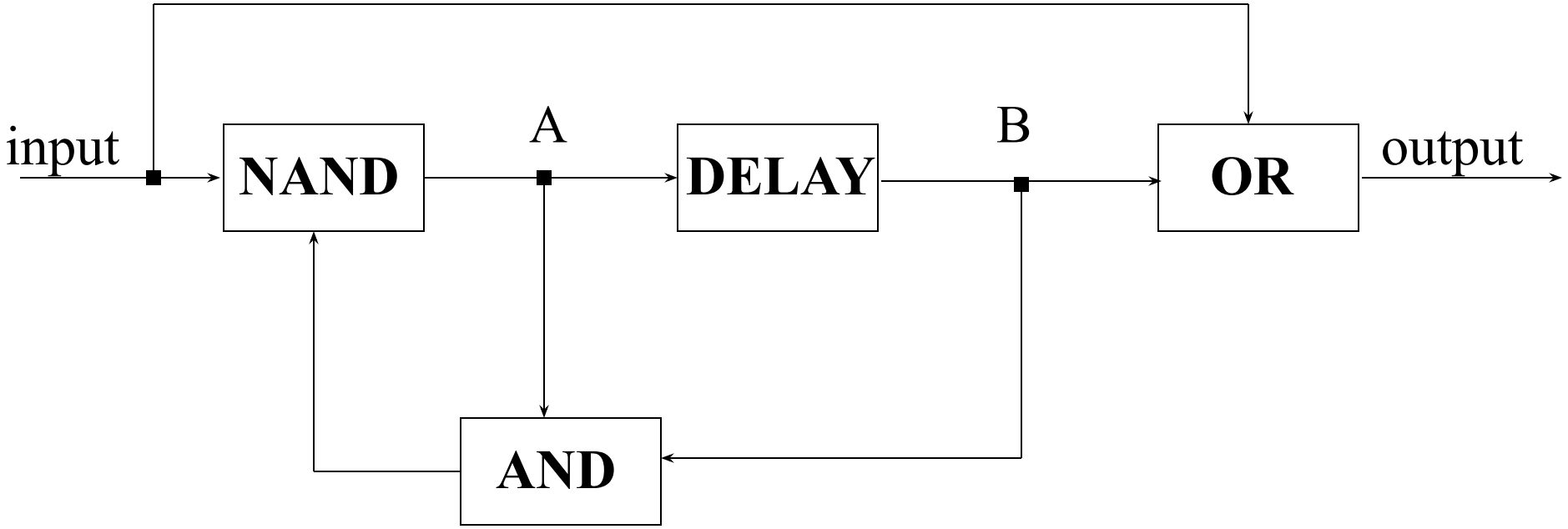
ii) Increasing the above result 24, in magnitude, by 1 24

+1

25 which is the same as obtained by ordinary subtraction.

Example

Consider the following sequential circuit



The following four types of boxes are used in this circuit

Example continued …

1. **NAND box (NOT AND)**: For given inputs, it provides the complement of Boolean AND output.
2. **DELAY box (Flip Flop box)**: It delays the transmission of signal along the wire by one step (clock pulse).
3. **OR box**: For given inputs, it provides the Boolean OR output.
4. **AND box**: For the given inputs, it provides the Boolean AND output.

The current in the wire is indicated by 1 and 0 indicates the absence of the current.

Example continued …

There are two points A and B w.r.t. to which following four states of the machine are identified according to the presence and absence of current at these points *i.e.*

1) **q_0(A=0, B=0)** ☐ (0,0)
2) **q_1 (A=0, B=1)** ☐ (0,1)
3) **q_2 (A=1, B=0)** ☐ (1,0)
4) **q_3 (A=1, B=1)** ☐ (1,1)

Example continued …

The operation of the circuit is such that the machine changes its state after reading 0 or 1. The transitions are determined using the following relations

new B = old A

new A = (input) **NAND** (old A **AND** old B)

output = (input) **OR** (old B)

It is to be noted that old A and old B indicate the presence or absence of current at A and B before inputting any letter. Similarly new A and new B indicate the presence or absence of current after reading certain letter.

Example continued ...

At various discrete pulses of a time clock, input is received by the machine and the corresponding output string is generated.

The transition at the state q_0 after reading the letter 0, can be determined, along with the corresponding output character as under

new B = old A = 0

new A = (input) **NAND** (old A **AND** old B)

= 0 **NAND** (0 **AND** 0) = 0 **NAND** 0

= 1

output = (input) **OR** (old B) = 0 **OR** 0 = 0

Example continued …

Thus after reading 0 at $\mathbf{q_0}$ new B is 0 and new A is 1 *i.e.*machine will be at state (1,0) ☐ $\mathbf{q_2}$ and during this process it's output character will be 0.

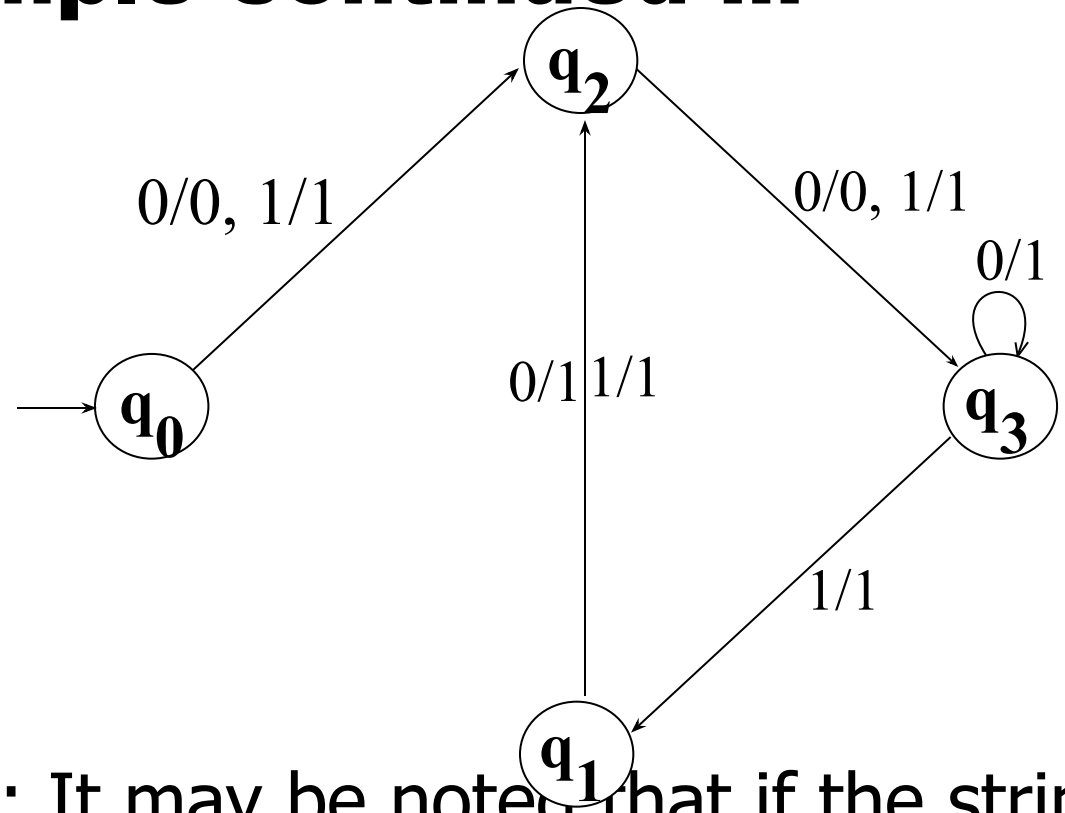
The remaining action of this sequential circuit can be determined as shown by the following suggested transition table of the corresponding Mealy machine

Example continued …

Old state	Inputting 0		Inputting 1	
	State	Output	State	Output
q_0 □(0,0)	(1,0)□q_2	0	(1,0)□q_2	1
q_1 □(0,1)	(1,0)□q_2	1	(1,0)□q_2	1
q_2 □(1,0)	(1,1)□q_3	0	(1,1)□q_3	1
q_3 □(1,1)	(1,1)□q_3	1	(1,1)□q_3	1

The corresponding transition diagram may be as follows

Example continued …



Note: It may be noted that if the string 00 is read at any state, it results ending in state q_3.

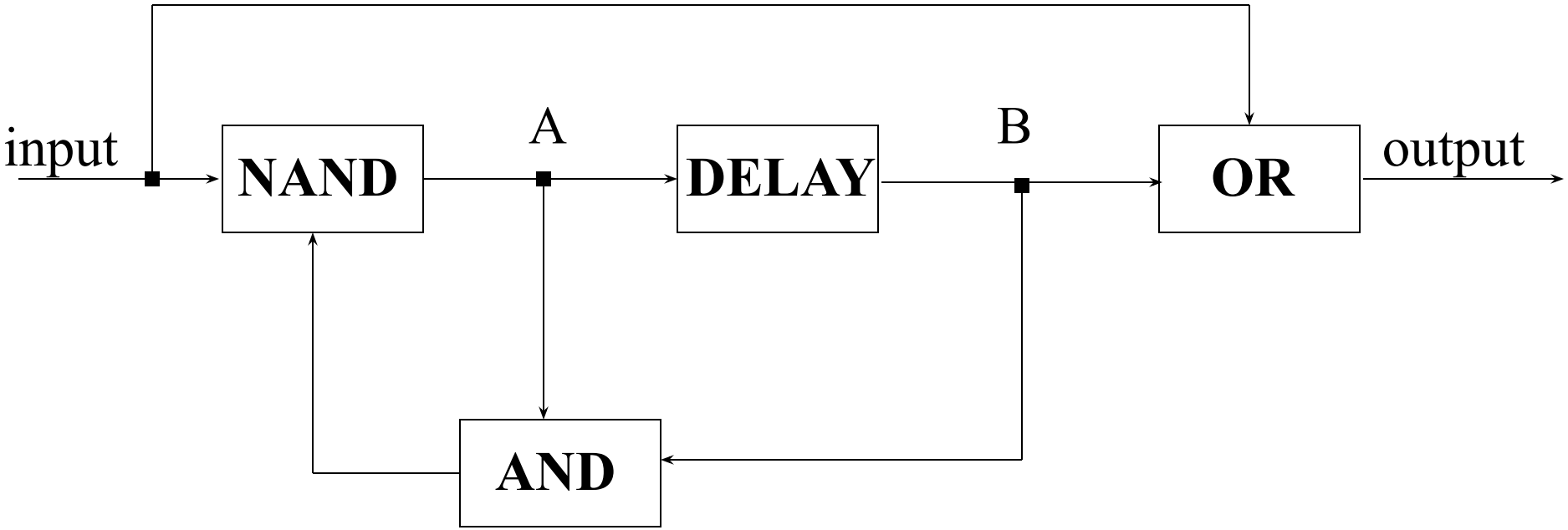
Example continued …

Running the string 01101110 on the previous machine, the output string can be determined by the following table

Input		0	1	1	0	1	1	1	0
States	q_0	q_2	q_3	q_1	q_2	q_3	q_1	q_2	q_3
output		0	1	1	1	1	1	1	0

Following is a note regarding the sequential circuit under consideration

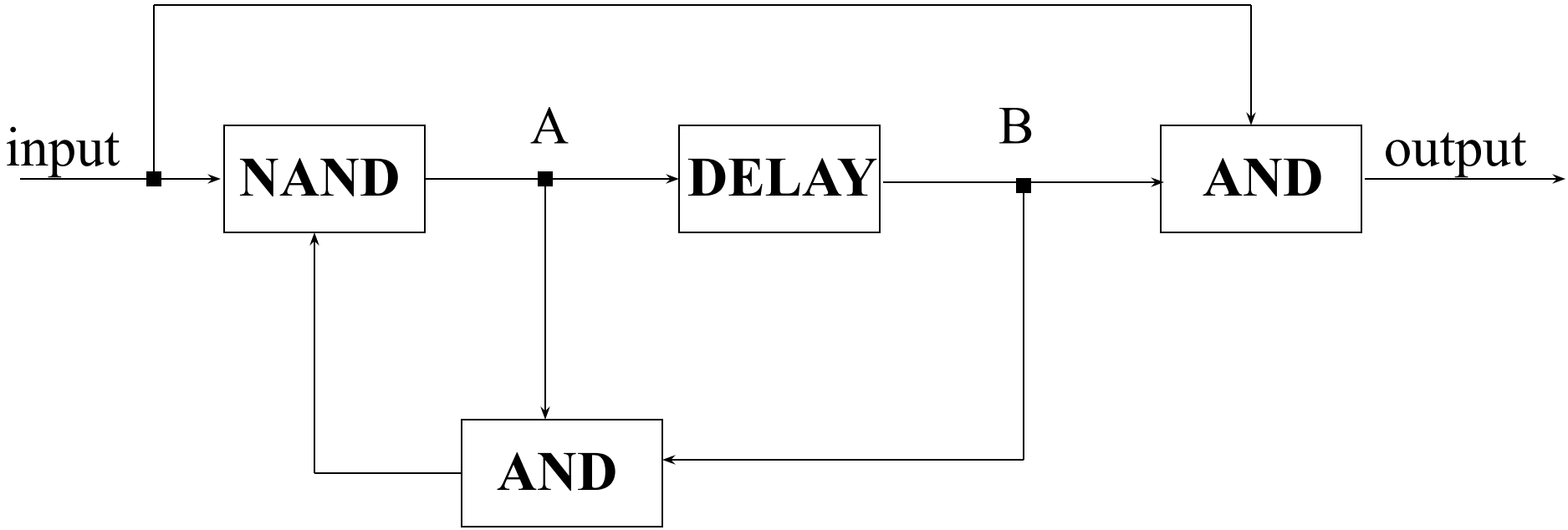
Note



It is to be noted that in this sequential circuit, delay box plays an important role in introducing four states of the machine.

Task

Build a Mealy machine corresponding to the following sequential circuit



Solution of the Task

Subtract 39 from 64

Solution: Taking a=64 and b=39.

i) Adding 9's complement (60) of b to a.

64

+60

124 which gives 24 (ignoring the overflow)

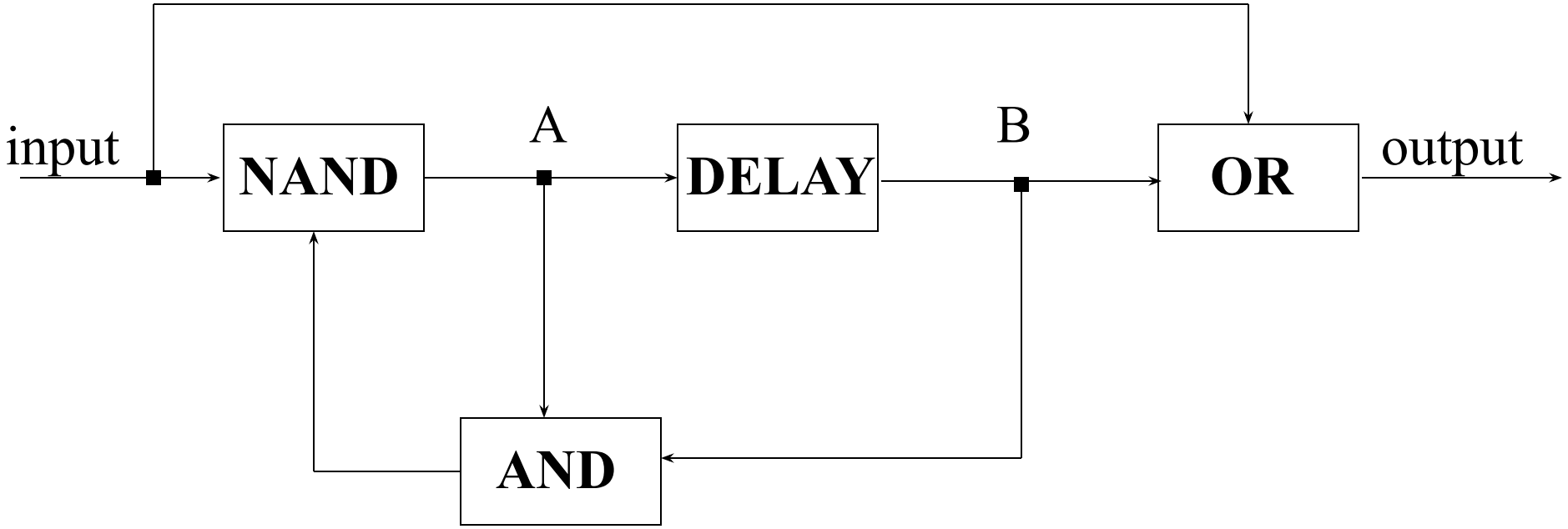
ii) Increasing the above result 24, in magnitude, by 1 24

+1

25 which is the same as obtained by ordinary subtraction.

Example

Consider the following sequential circuit



The following four types of boxes are used in this circuit

Example continued …

1. **NAND box (NOT AND)**: For given inputs, it provides the complement of Boolean AND output.
2. **DELAY box (Flip Flop box)**: It delays the transmission of signal along the wire by one step (clock pulse).
3. **OR box**: For given inputs, it provides the Boolean OR output.
4. **AND box**: For the given inputs, it provides the Boolean AND output.

The current in the wire is indicated by 1 and 0 indicates the absence of the current.

Example continued …

There are two points A and B w.r.t. to which following four states of the machine are identified according to the presence and absence of current at these points *i.e.*

1) **q_0(A=0, B=0)** ☐ (0,0)
2) **q_1 (A=0, B=1)** ☐ (0,1)
3) **q_2 (A=1, B=0)** ☐ (1,0)
4) **q_3 (A=1, B=1)** ☐ (1,1)

Example continued …

The operation of the circuit is such that the machine changes its state after reading 0 or 1. The transitions are determined using the following relations

new B = old A

new A = (input) **NAND** (old A **AND** old B)

output = (input) **OR** (old B)

It is to be noted that old A and old B indicate the presence or absence of current at A and B before inputting any letter. Similarly new A and new B indicate the presence or absence of current after reading certain letter.

Example continued ...

At various discrete pulses of a time clock, input is received by the machine and the corresponding output string is generated.

The transition at the state q_0 after reading the letter 0, can be determined, along with the corresponding output character as under

new B = old A = 0

new A = (input) **NAND** (old A **AND** old B)

= 0 **NAND** (0 **AND** 0) = 0 **NAND** 0

= 1

output = (input) **OR** (old B) = 0 **OR** 0 = 0

Example continued …

Thus after reading 0 at $\mathbf{q_0}$ new B is 0 and new A is 1 *i.e.*machine will be at state (1,0) ☐ $\mathbf{q_2}$ and during this process it's output character will be 0.

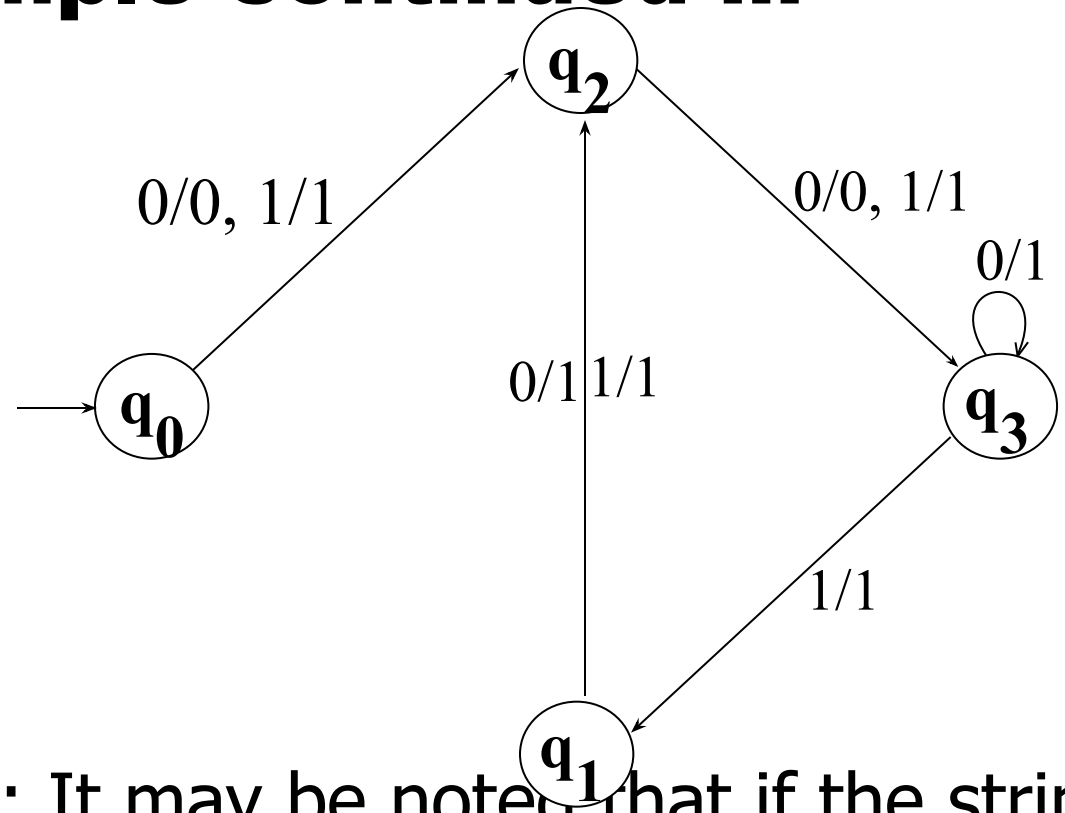
The remaining action of this sequential circuit can be determined as shown by the following suggested transition table of the corresponding Mealy machine

Example continued …

Old state	Inputting 0		Inputting 1	
	State	Output	State	Output
q_0 □(0,0)	(1,0)□q_2	0	(1,0)□q_2	1
q_1 □(0,1)	(1,0)□q_2	1	(1,0)□q_2	1
q_2 □(1,0)	(1,1)□q_3	0	(1,1)□q_3	1
q_3 □(1,1)	(1,1)□q_3	1	(1,1)□q_3	1

The corresponding transition diagram may be as follows

Example continued ...



Note: It may be noted that if the string 00 is read at any state, it results ending in state q_3.

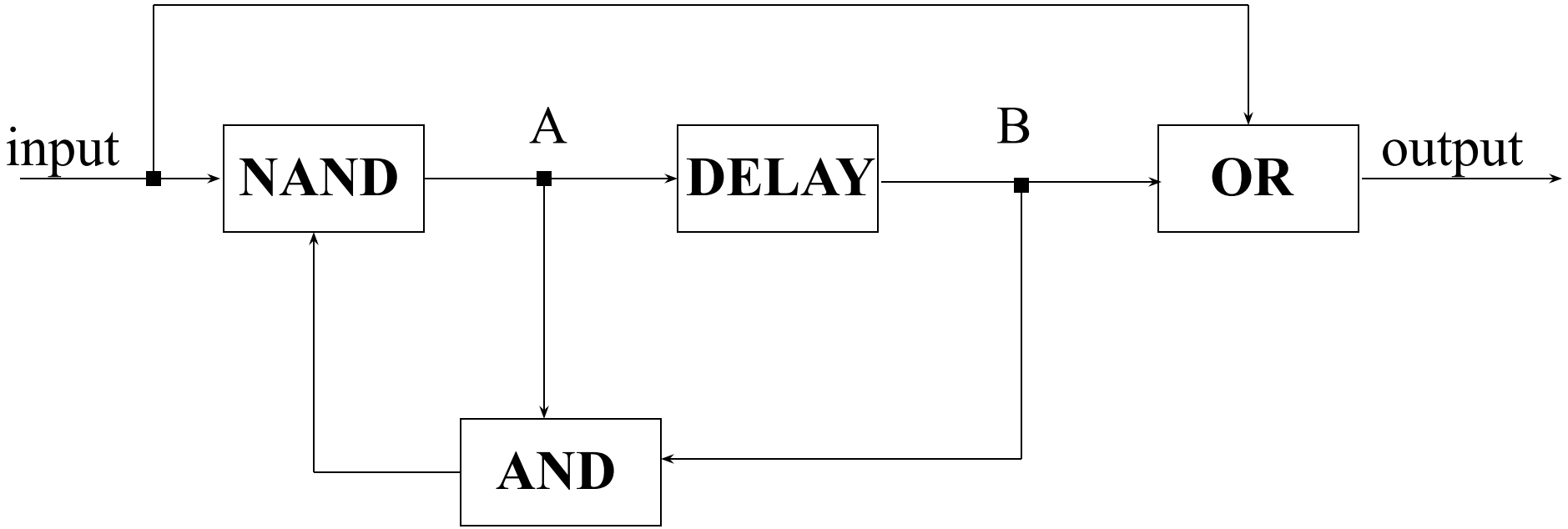
Example continued …

Running the string 01101110 on the previous machine, the output string can be determined by the following table

Input		0	1	1	0	1	1	1	0
States	q_0	q_2	q_3	q_1	q_2	q_3	q_1	q_2	q_3
output		0	1	1	1	1	1	1	0

Following is a note regarding the sequential circuit under consideration

Note



It is to be noted that in this sequential circuit, delay box plays an important role in introducing four states of the machine.

Task

Build a Mealy machine corresponding to the following sequential circuit