By this process, the merger table is

B	(BD) (BC)			
C	(BD)	(BC)		
D	(CE) (BC)	√	×	
E	(CE) (BC)	√	(CE) (BC)	√
	A	B	C	D

The boxes which are not crossed are compatible pairs. So, the compatible pairs are (AB), (AC), (AD), (AE), (BC), (BD), (BE), (CD), and (DE).

Example 4.13 Construct a merger table of the following machine and fi nd the compatible pairs.

Solution:

	Next State,z			
Present State	I_1	I_2	I_3	I_4
A	$C,-$	$-,-$	$-,-$	$-,-$
B	$-,-$	$C,-$	$D,-$	$E,-$
C	$-,-$	$F,0$	$B,-$	$-,-$
D	$E,-$	$-,1$	$-,-$	$A,-$
E	$-,-$	$B,-$	$-,-$	$C,-$
F	$B,-$	$-,-$	$E,-$	$-,-$

For the states AB, for all the inputs, next states and output do not conflict. So a $\surd$ (tick) is placed in the box labelled AB.

For states AC, next states and outputs do not conflict for all the inputs. So a $\surd$ (tick) is placed in the box labelled AC.

For states AD, the outputs do not conflict, but the next states for input I_1 conflict. So, the conflicting next state pair (CE) is placed in the box labelled AD.

In the box labelled (AE), a $\surd$ (tick) is placed.

In the box (AF), the conflicting next state pair (BC) is placed.

In the box (BC), the conflicting next state pairs (CF) and (BD) are placed.

In the box (BD), the conflicting next state pair (AE) is placed.

In the box (BE), the conflicting next state pairs (BC) and (CE) are placed.

In the box (BF), the conflicting next state pair (DE) is placed.

The outputs for I_2 for the state (CD) conflict. So a $\times$ is placed in the box (CD).

In the box (CE), the conflicting next state pair (BF) is placed.

In the box (CF), the conflicting next state pair (BE) is placed.

By this process, the constructed merger table is

B	√				
C	√	(CF) (BD)			
D	(CE)	(AE)	×		
E	√	(BC) (CE)	(BF)	(AC)	
F	(BC)	(DE)	(BE)	(BE)	√
	A	B	C	D	E

The compatible pairs are (AB), (AC), (AD), (AE), (AF), (BC), (BD), (DE), (DF), (CE), (CF), (DE), (DF), and (EF).

4.9 Finite Memory and Definite Memory Machine

If we recall the defi nition of an FSM, it is told that an FSM is a machine whose past histories can affect its future behaviour in a fi nite number of ways. It means that the present behaviour of the machine is dependent on its past histories. To memorize the past histories, an FSM needs memory elements. The amount of past input and corresponding output information is necessary to determine the machine's future behaviour. This is called the memory span of the machine.

Let us assume that a machine is deterministic (for a single state with single input, only one next state is produced) and completely specifi ed. For this type of a machine, if the initial state and the input sequence are known, one can easily fi nd the output sequence and the corresponding fi nal state. One interesting thing is that, this output sequence and the fi nal state are unique. But the reverse is not always true. If the fi nal state and the output sequence are known, it is not always possible to determine uniquely the input sequence. This section describes the minimum amount of past input–output information required to fi nd the future behaviour of the machine and the condition under which the input to the machine can be constructed from the output produced.

4.9.1 Finite Memory Machine

An FSM M is called a finite memory machine of order μ if μ is the least integer so that the present state of the machine M can be obtained uniquely from the knowledge of last μ number of inputs and the corresponding μ number of outputs.

There are two methods to find whether a machine is finite or not

1. Testing table and testing graph for finite memory
2. Vanishing connection matrix.

4.9.1.1 Testing Table and Testing Graph for Finite Memory Method

The testing table for finite memory is divided into two halves. The upper half contains a single state input–output combination. If, in a machine, there are two types of inputs and two types of outputs, say

0 and 1, the input–output combinations are 0/0, 0/1, 1/0, and 1/1. Here, 0/0 means 0 input and 0 outputs, that is, for the cases we are getting output 0 for input 0, and 0/1 means 0 input and 1 output, that is, for the cases we are getting output 1 for input 0.

The lower half of the table contains all the combinations of the present states taking two into combination. For four present states, (say, A, B, C, and D) there are ${}^4C_2$, which is six, combinations: AB, AC, AD, BC, BD, and CD.

The table is constructed according to the machine given.

The pair of the present state combination is called the uncertainty pair. And its successor is called the implied pair.

In the testing graph for fi nite memory,

1. The number of nodes will be the number of present state combination taking two into account.
2. There will be a directed arc with a label of input–output combination, from $S_iS_j[i \neq j]$ to $S_pS_q[p \neq q]$, if S_pS_q is the implied pair of S_iS_j.

If the testing graph is loop-free, the machine is of finite memory. The order of finiteness is the length of the longest path in the testing Graph $(l) + 1$, i.e., $\mu = l + 1$.

4.9.1.2 Vanishing Connection Matrix Method

If the number of states increases, then it becomes diffi cult to fi nd the longest path in the Testing graph for fi nite memory. There is an easy method to determine whether a machine is fi nite or not, and if fi nite, to fi nd its order of fi niteness. The process is called vanishing connection matrix method.

4.9.2 Constructing the Method of Connection Matrix

1. The number of rows will be equal to the number of columns ($p \times p$ matrix) .
2. The rows and columns will be labelled with the pair of the present state combinations. The labels associated with the corresponding rows and columns will be identical.
3. In the matrix, the (i, j) th entry will be 1 if there is an entry in the $(S_a S_b)$ and $(S_p S_q)$ combination in the corresponding testing table. Otherwise, the entry will be 0.

4.9.3 Vanishing of Connection Matrix

1. Delete all the rows having 0's in all positions and delete the corresponding columns also.
2. Repeat this step until one of the following steps is achieved
   (a) No row having 0's in all positions left
   (b) The matrix vanishes, which means there are no rows and columns left.

If the condition 2(a) arrives, the machine is not of fi nite memory.

If the condition 2(b) arrives, the machine is of fi nite memory and the number of steps required to vanish the matrix is the order of finiteness of the machine.

The following examples describe the processes in detail.

Example 3.21 `Test whether the following machine is of fi nite memory or not by using testing table{testing graph and vanishing matrix method.`

Solution:

	Next State,z	
Present State	X=0	X=1
A	$D, 1$	$A, 1$
B	$D, 0$	$A, 1$
C	$B, 1$	$B, 1$
D	$A, 1$	$C, 1$

- Testing Table and Testing Graph for Finite Memory Method: A table which is divided into two halves is constructed. The machine has two inputs and two outputs. There are four input–output combinations namely 0/0, 0/1, 1/0, and 1/1. The upper half of the machine contains single state input–output combination and the lower half contains two state input–output combinations. There are four states, and so six combination pairs are made. The testing table becomes

Present State	0/0	0/1	1/0	1/1
A	D	–	–	A
B	–	D	–	A
C	–	B	–	B
D	–	A	–	C
AB	–	–	–	AA
AC	–	–	–	AB
AD	–	–	–	AC
BC	–	BD	–	AB
BD	–	AD	–	AC
CD	–	AB	–	BC

In the testing table, there are six present states combinations. So, in the testing table there are six nodes. There is a directed arc with a label of input–output combination, from $S_iS_j[i \neq j]$ to $S_pS_q[p \neq q]$, if S_pS_q is the implied pair of S_iS_j. The testing graph for fi nite memory is given in Fig. 4.20.

(There will be no arc from AB to AA as AA, is the repetition of same state 'A'.)

The testing graph is loop-free. The longest path in the testing graph is $5(CD \rightarrow CBC \rightarrow CBD \rightarrow CAD \rightarrow CAC \rightarrow CAB)$, and so the order of defi niteness $\mu = 5 + 1 = 6$.

- **Vanishing Connection Matrix Method:** According to the rule of the construction of the connection matrix, a table is constructed with six rows and six columns labelled with the present state

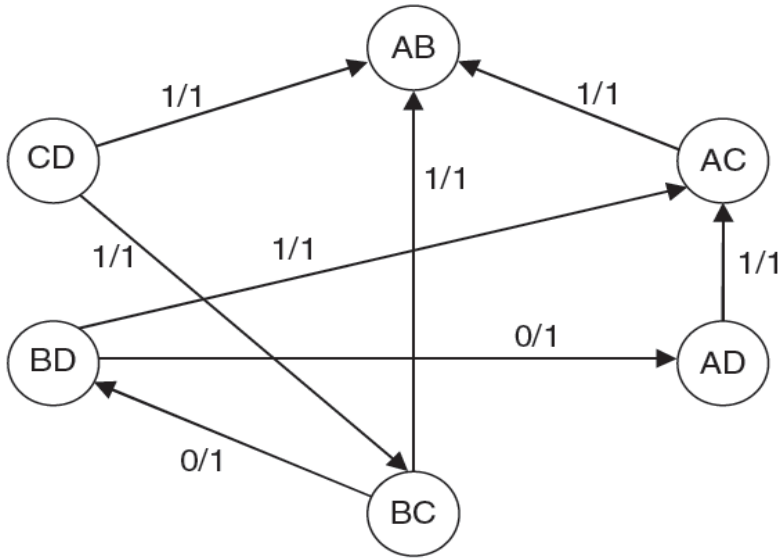

Fig. 4.20 *Testing Graph for Finite Memory*