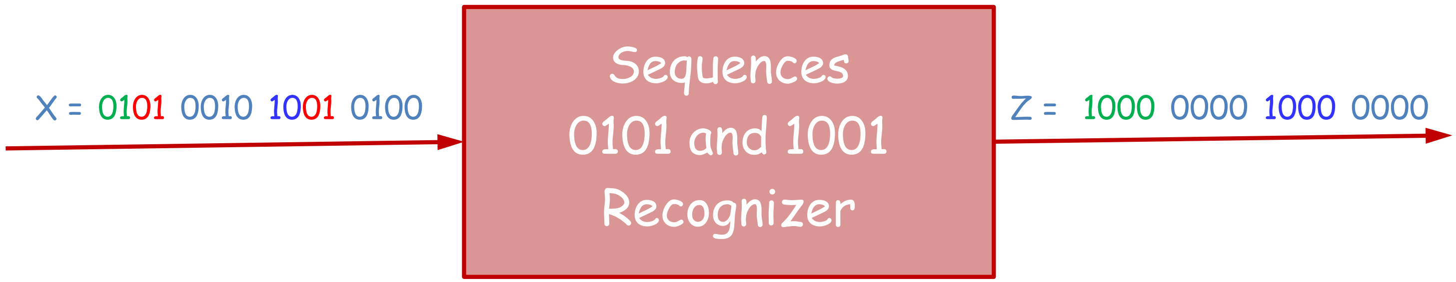
Objective: Eliminate redundant states

- ✓ Reduce the number of states in the state table to the minimum.
  - Remove redundant states
  - Use don't cares effectively
- ✓ Reduction to the minimum number of states reduces
  - The number of F/Fs needed
  - Reduces the number of next states that has to be generated ⇨ Reduced logic.

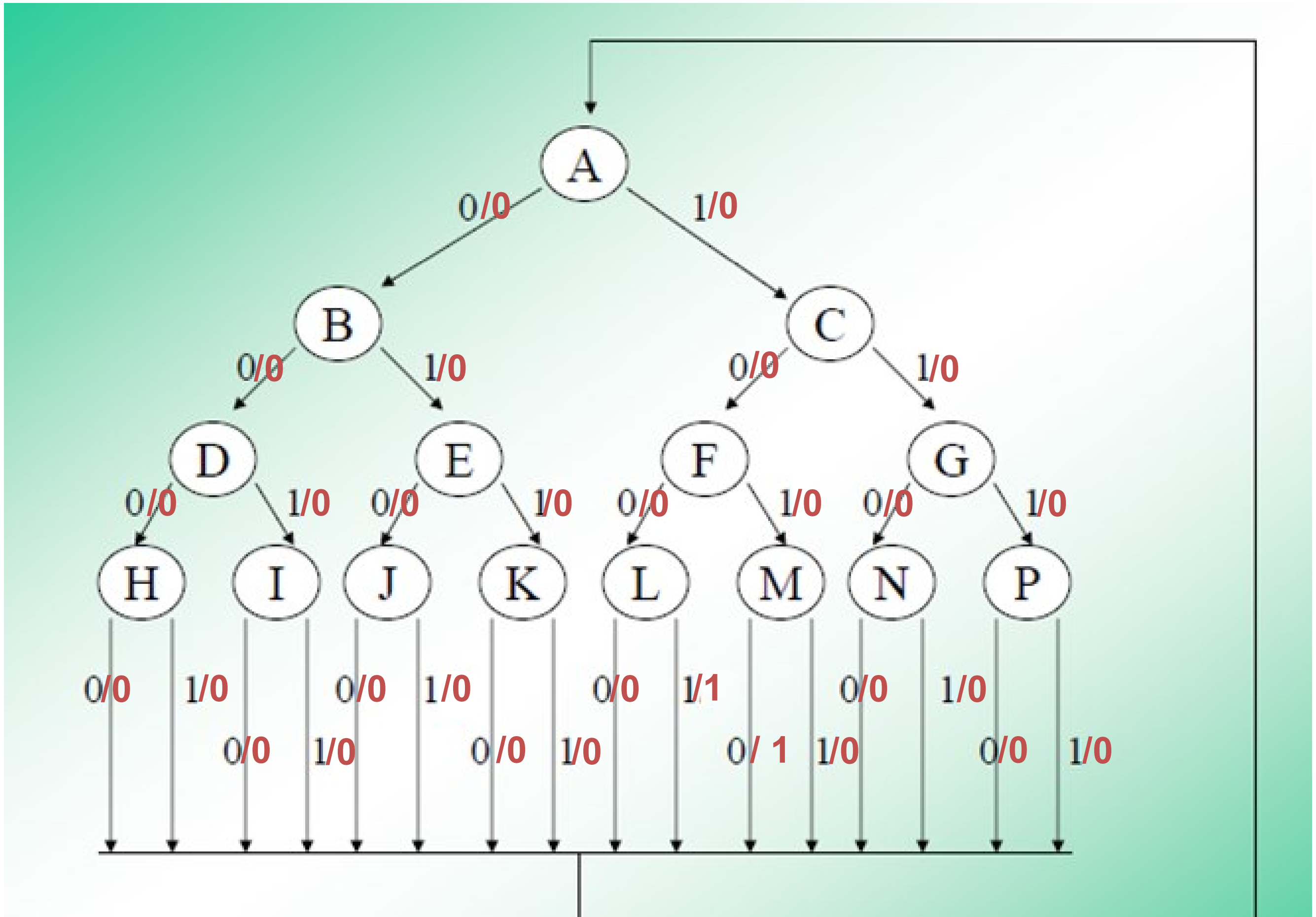
An example circuit

- A sequential circuit has one input X and one output Z.
- The circuit looks at the groups of four consecutive inputs and sets Z=1 if the input sequence 0101 or 1001 occurs.
- The circuit returns to the reset state after four inputs.
- Design the Mealy machine.



Elimination of Redundant States

✓ When first setting up the state table, we will not be overly concerned with inclusion of extra states, and when the table is complete, we will eliminate any redundant states.



State diagram

State table

- ✓ Set up a table for all the possible input combinations
- ✓ For the two sequences when the last bit is a 1 return to reset with Z=1.

Input Sequence	Present State	Next State X = 0	Next State X = 1	Present Output X = 0	Present Output X = 1
reset	A	B	C	0	0
0	B	D	E	0	0
1	C	F	G	0	0
00	D	H	I	0	0
01	E	J	K	0	0
10	F	L	M	0	0
11	G	N	P	0	0
000	H	A	A	0	0
001	I	A	A	0	0
010	J	A	A	0	0
011	K	A	A	0	0
100	L	A	A	0	1
101	M	A	A	1	0
110	N	A	A	0	0
111	P	A	A	0	0

Note on state table generation

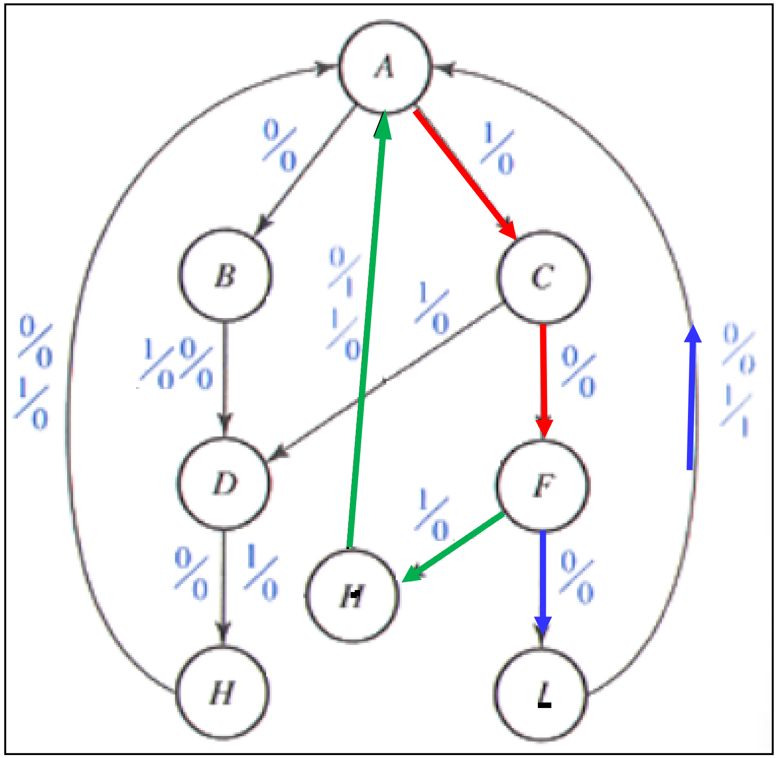
- ✓ When generated by looking at all combinations of inputs the state table is far from minimal.
- ✓ First step is to remove redundant states.
  - There are states that you cannot tell apart
    - Such as H and I – both have A with Z=0 as output.
    - State H is equivalent to state I and state I can be removed from the table.
    - Examining table shows states J, K, N and P are also the same – they can be deleted.

Input Sequence	Present State	Next State $X=0$	Next State $X=1$	Present Output $X=0$	Present Output $X=1$
reset	A	B	C	0	0
0	B	D	E	0	0
1	C	F	G	0	0
00	D	H	I	0	0
01	E	J	K	0	0
10	F	L	M	0	0
11	G	N	P	0	0
000	H	A	A	0	0
~~001~~	~~I~~	~~A~~	~~A~~	~~0~~	~~0~~
~~010~~	~~J~~	~~A~~	~~A~~	~~0~~	~~0~~
~~011~~	~~K~~	~~A~~	~~A~~	~~0~~	~~0~~
100	L	A	A	0	1
101	M	A	A	1	0
~~110~~	~~N~~	~~A~~	~~A~~	~~0~~	~~0~~
~~111~~	~~P~~	~~A~~	~~A~~	~~0~~	~~0~~

The result

- ✓ Reduced state table and graph

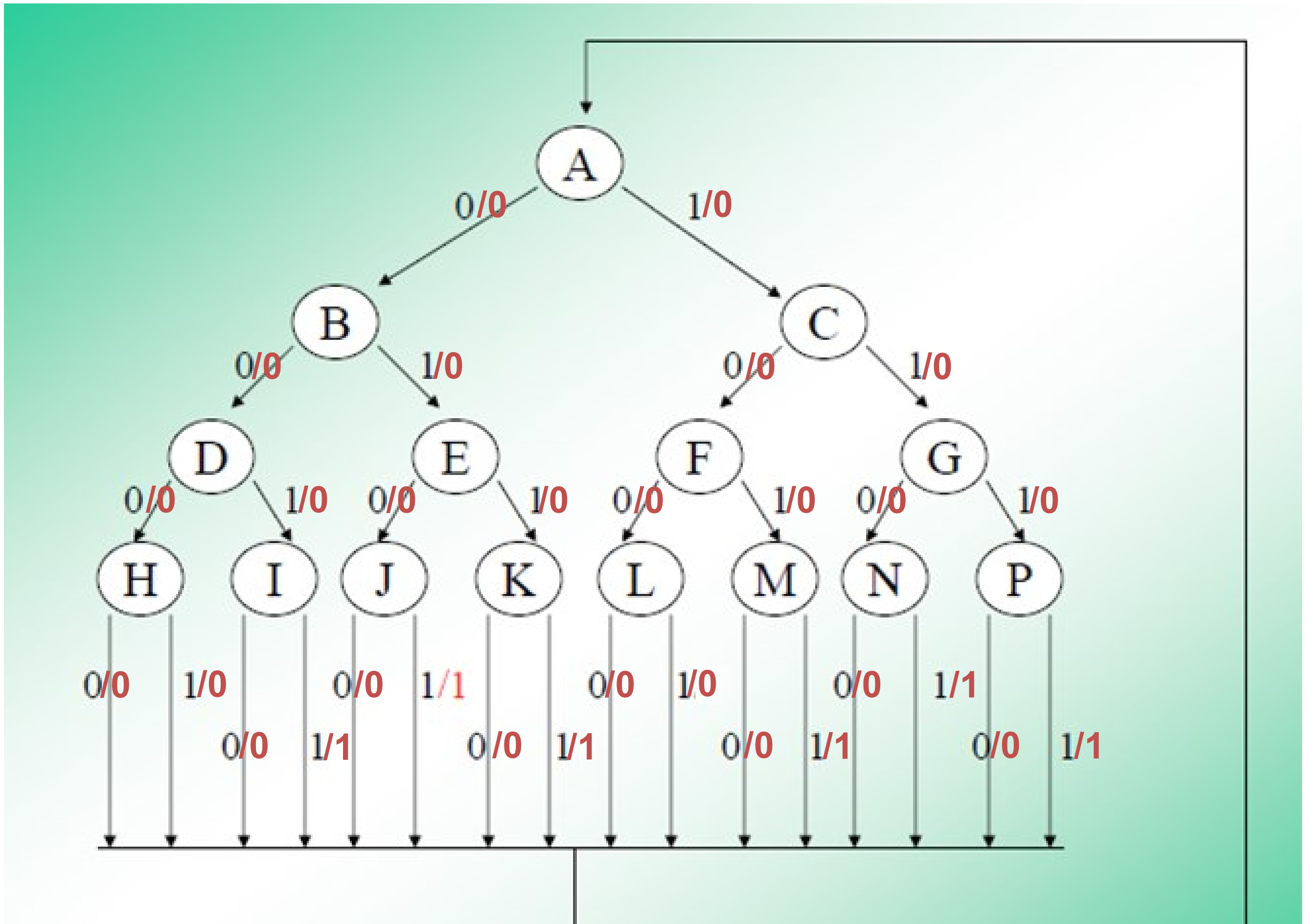
Input Sequence	Present State	Next State $X = 0$	Next State $X = 1$	Present Output $X = 0$	Present Output $X = 1$
reset	A	B	C	0	0
0	B	D	D	0	0
1	C	F	D	0	0
00	D	H	H	0	0
10	F	L	M	0	0
000	H	A	A	0	0
100	L	A	A	0	1
101	M	A	A	1	0



- ✓ Original – 15 states – reduced 8 states

Elimination of Redundant States

- ✓ Design a binary checker that has in input a sequence of BCD numbers and for every four bits (LSB order) has output 0 if the number is $0 \le N \le 9$ and 1 if $10 \le N \le 15$



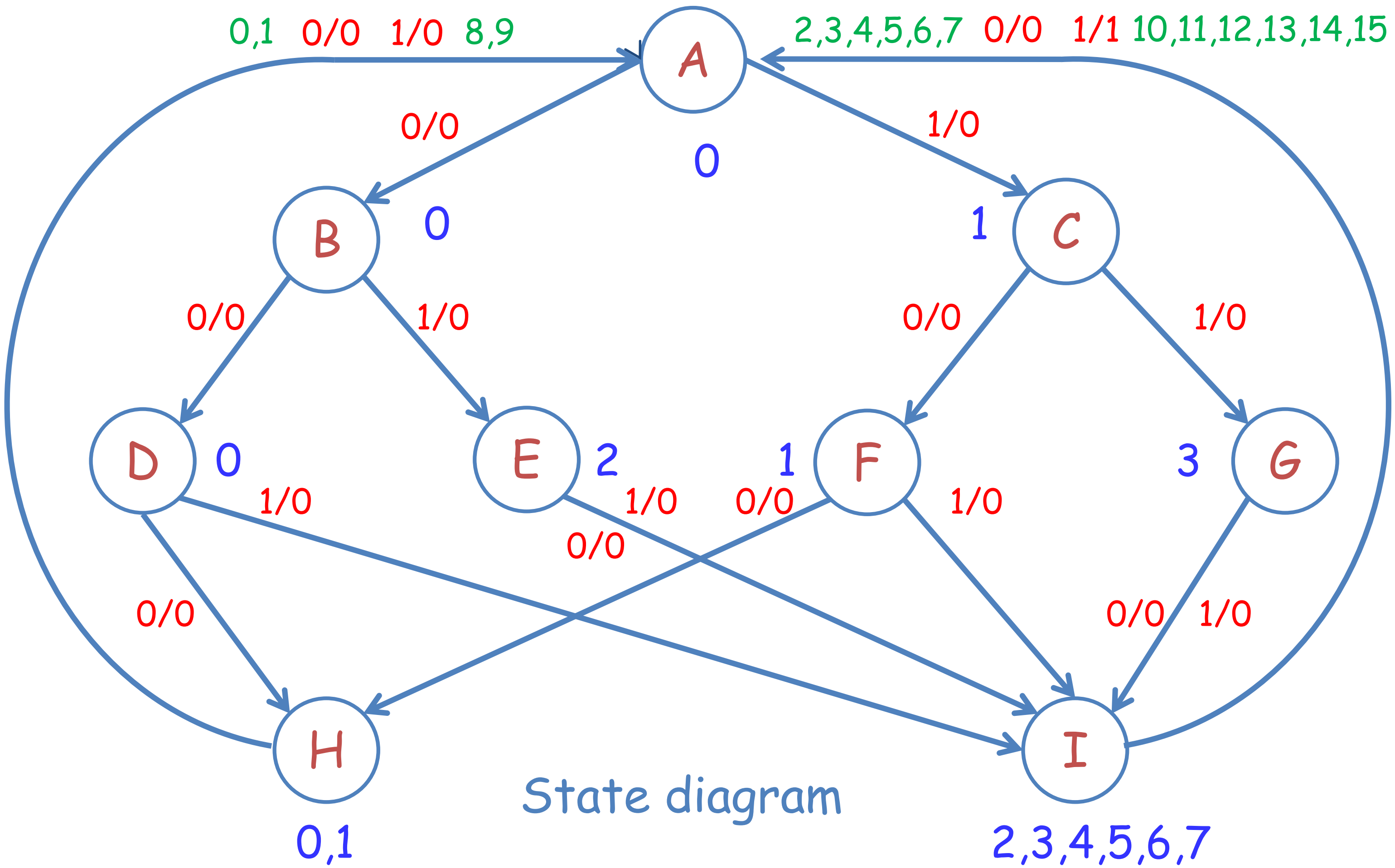
State diagram

Elimination of Redundant States

- ✓ Design a binary checker that has in input a sequence of BCD numbers and for every four bits (LSB order) has output 0 if the number is $0 \le N \le 9$ and 1 if $10 \le N \le 15$

Input Sequence	Present State	Next State $X = 0$	Next State $X = 1$	Present Output $X = 0$	Present Output $X = 1$
reset	A	B	C	0	0
0	B	D	E	0	0
1	C	F	G	0	0
00	D	H	I	0	0
01	E	J	K	0	0
10	F	L	M	0	0
11	G	N	P	0	0
~~000~~	H	A	A	0	0
~~001~~	I	A	A	0	1
~~010~~	~~J~~	~~A~~	~~A~~	~~0~~	~~1~~
~~011~~	~~K~~	~~A~~	~~A~~	~~0~~	~~1~~
~~100~~	~~L~~	~~A~~	~~A~~	~~0~~	~~0~~
~~101~~	~~M~~	~~A~~	~~A~~	~~0~~	~~1~~
~~110~~	~~N~~	~~A~~	~~A~~	~~0~~	~~1~~
~~111~~	~~P~~	~~A~~	~~A~~	~~0~~	~~1~~

Elimination of Redundant States



State diagram

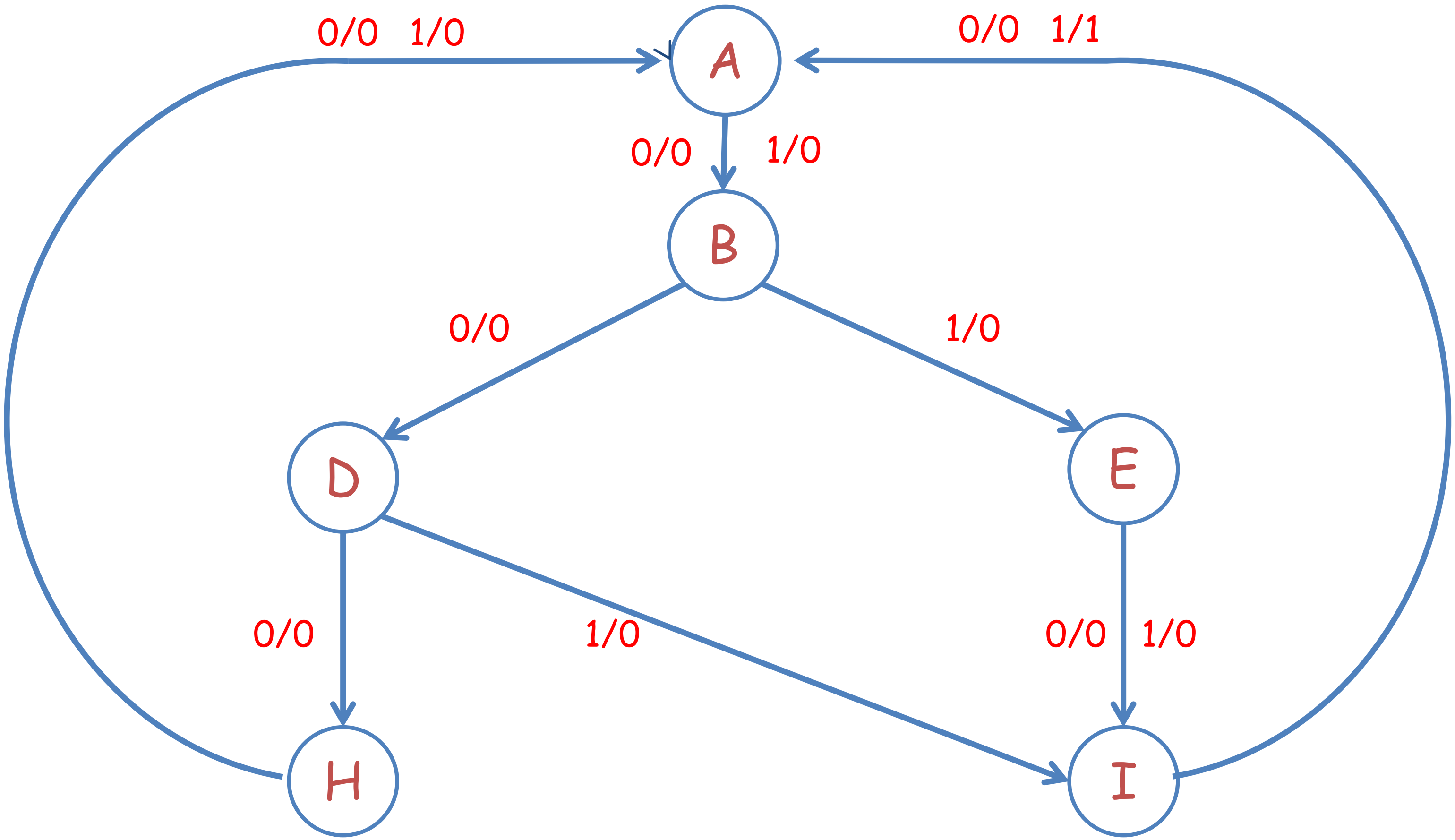
Elimination of Redundant States

	Input Sequence	Present State	Next State $X = 0$	Next State $X = 1$	Present Output $X = 0$	Present Output $X = 1$
0	reset	A	B	C	0	0
0	~~0~~	B	D	E	0	0
1	~~1~~	~~C~~	~~F~~	~~G~~	~~0~~	~~0~~
0	~~00~~	D	H	I	0	0
2	~~01~~	E	I	I	0	0
1	~~10~~	~~F~~	~~H~~	~~I~~	~~0~~	~~0~~
3	~~11~~	~~G~~	~~I~~	~~I~~	~~0~~	~~0~~
0,1	000 100	H	A	A	0	0
2,3,4,5,6,7	001 101	I	A	A	0	1
	010 110					
	011 111					

{D,F}, {E,G} , {B,C}

State table

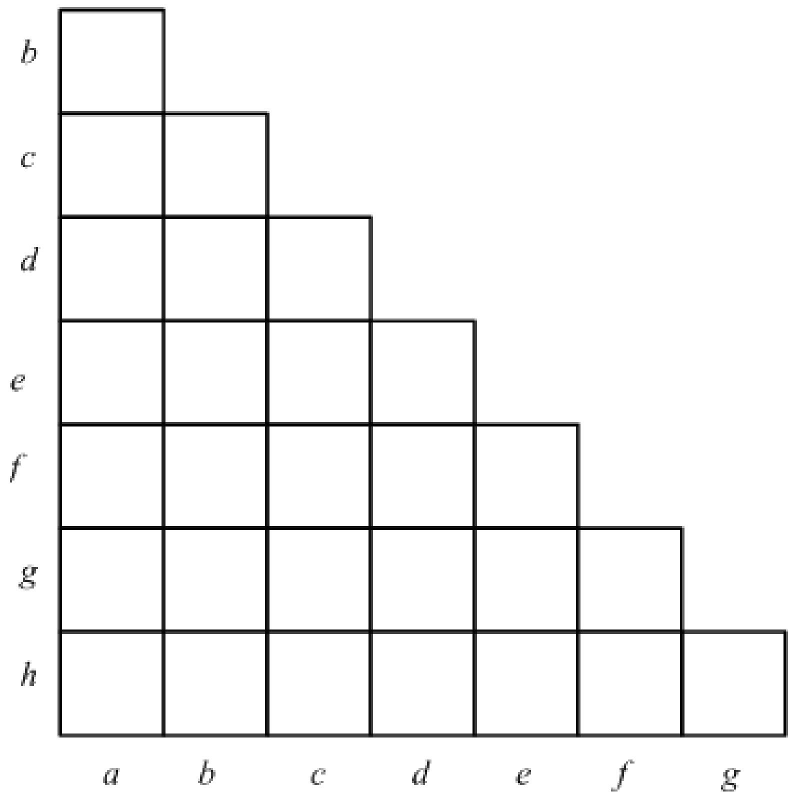
Elimination of Redundant States



State diagram

Implication Tables

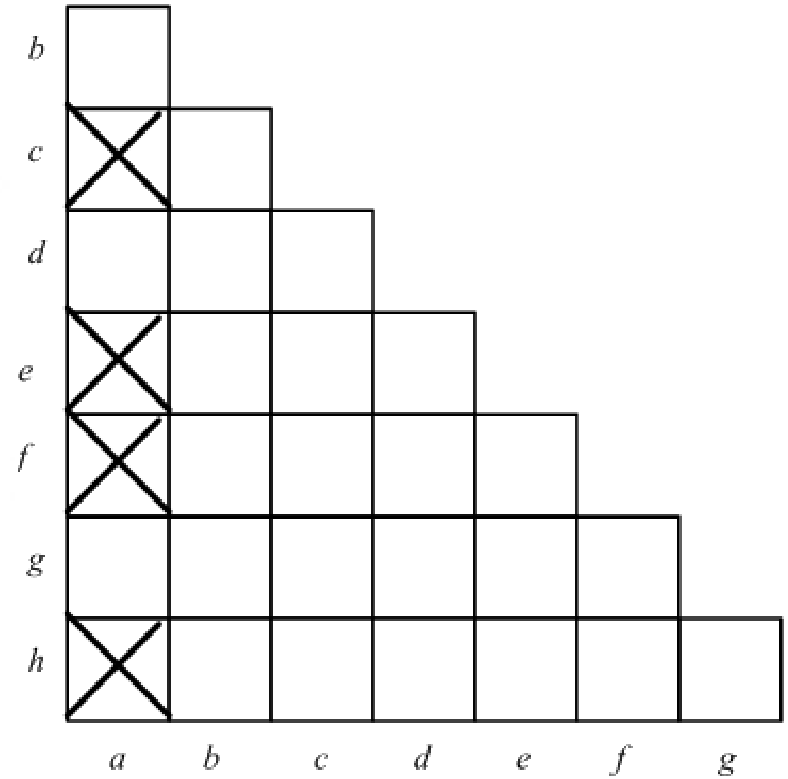
- A procedure for finding all the equivalent states in a state table.
- Use an implication table – a chart that has a square for each pair of states.



Step 1

- ✓ Use a X in the square to eliminate output incompatible states.
- ✓ 1st output of a differes from c, e, f, and h

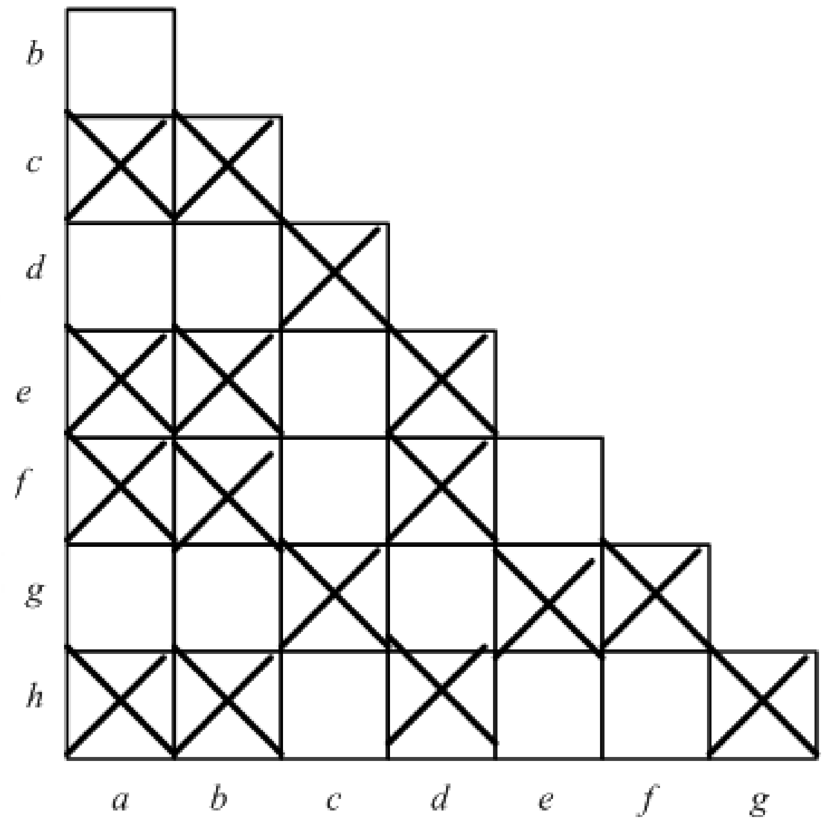
Present State	Next State $X = 0$	Next State 1	Present Output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1



Step 1 continued

✓ Continue to remove output incompatible states

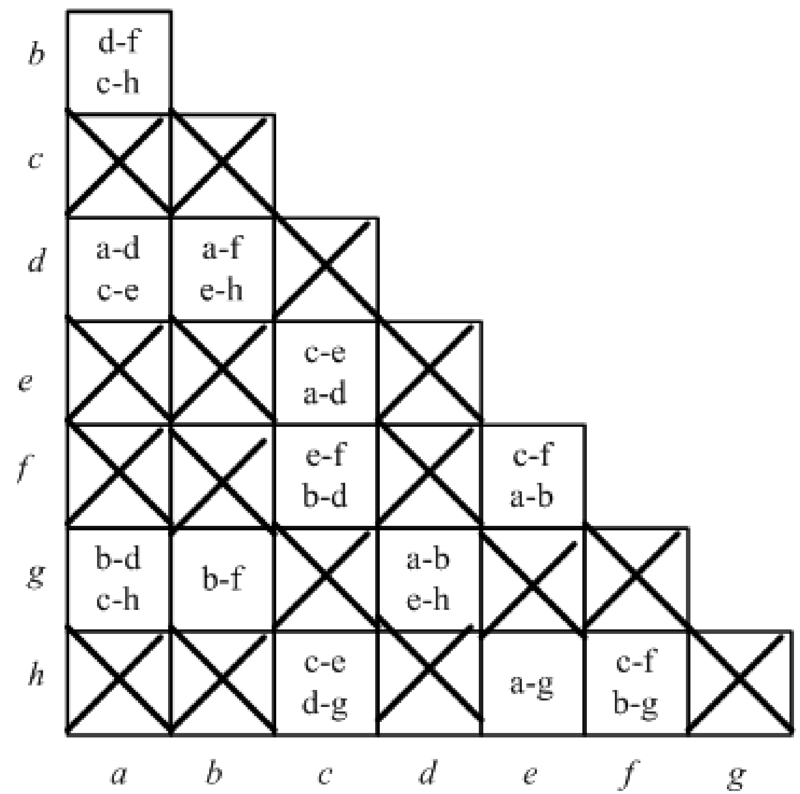
Present State	Next State $X = 0$	1	Present Output
a	*d*	*c*	0
b	*f*	*h*	0
c	*e*	*d*	1
d	*a*	*e*	0
e	*c*	*a*	1
f	*f*	*b*	1
g	*b*	*h*	0
h	*c*	*g*	1



Now what?

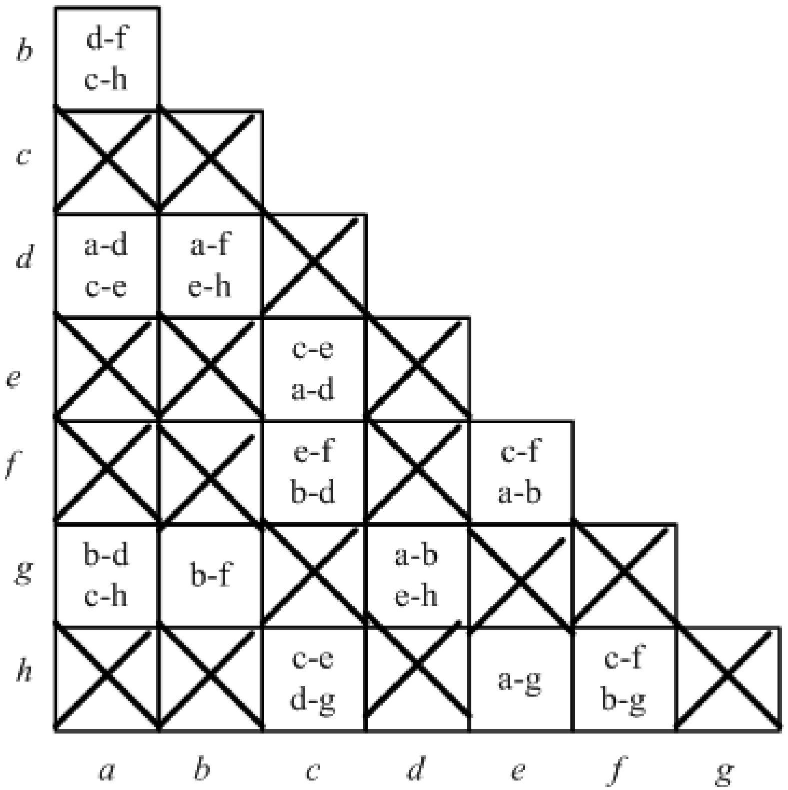
- ✓ Implied pair are now entered into each non X square.
- ✓ Here a≡b iff d≡f and c≡h

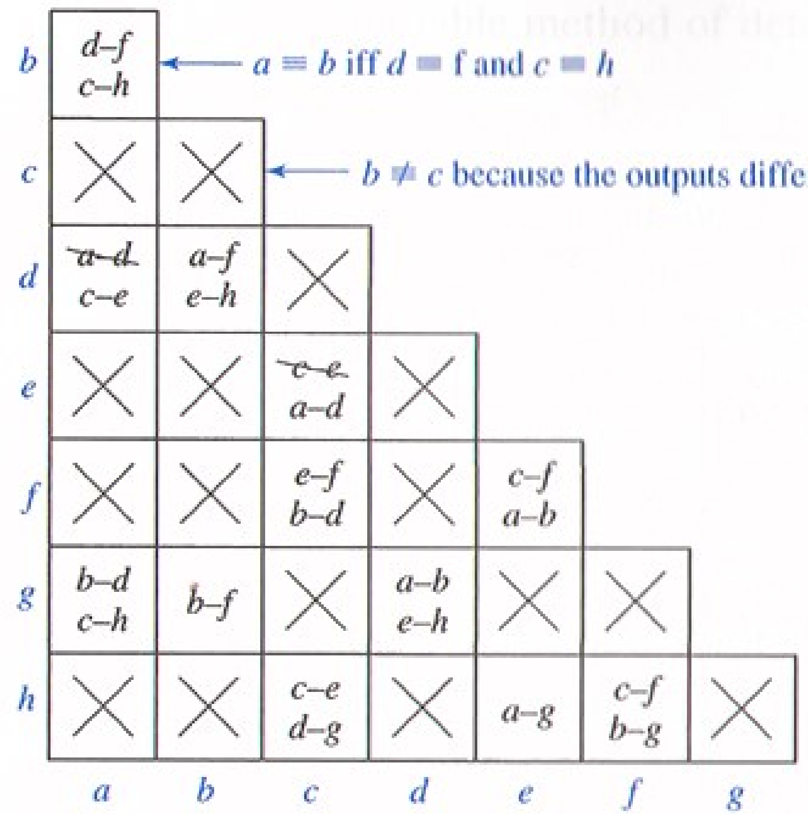
Present State	Next State X = 0	1	Present Output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1



Self redundant pairs

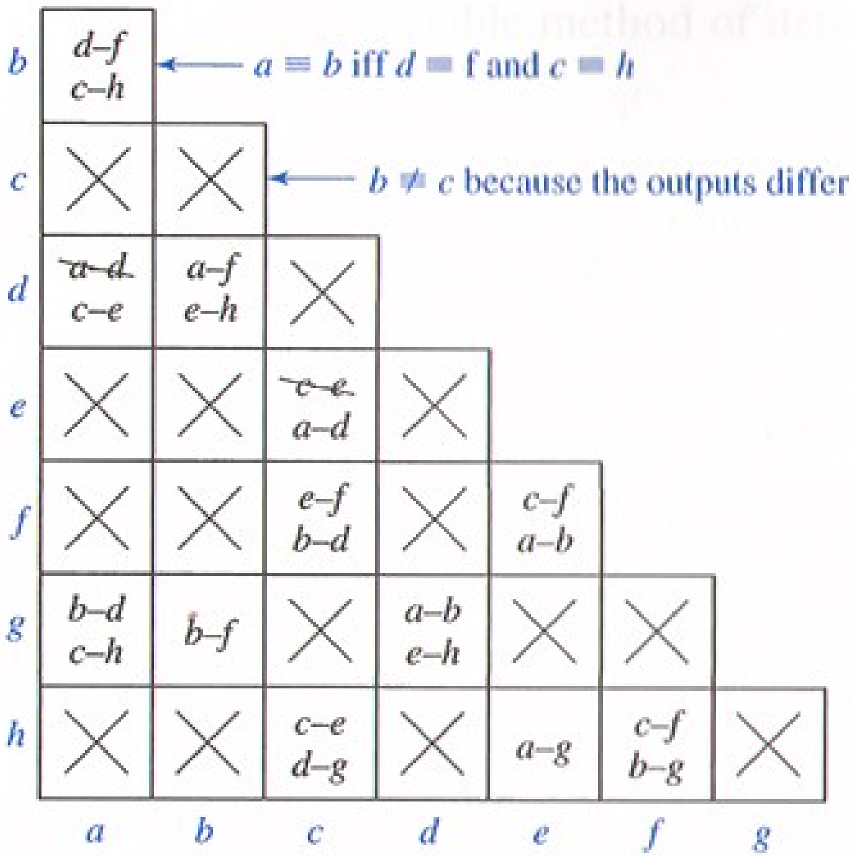
- ✓ Self redundant pairs are removed, i.e., in square a-d it contains a-d.

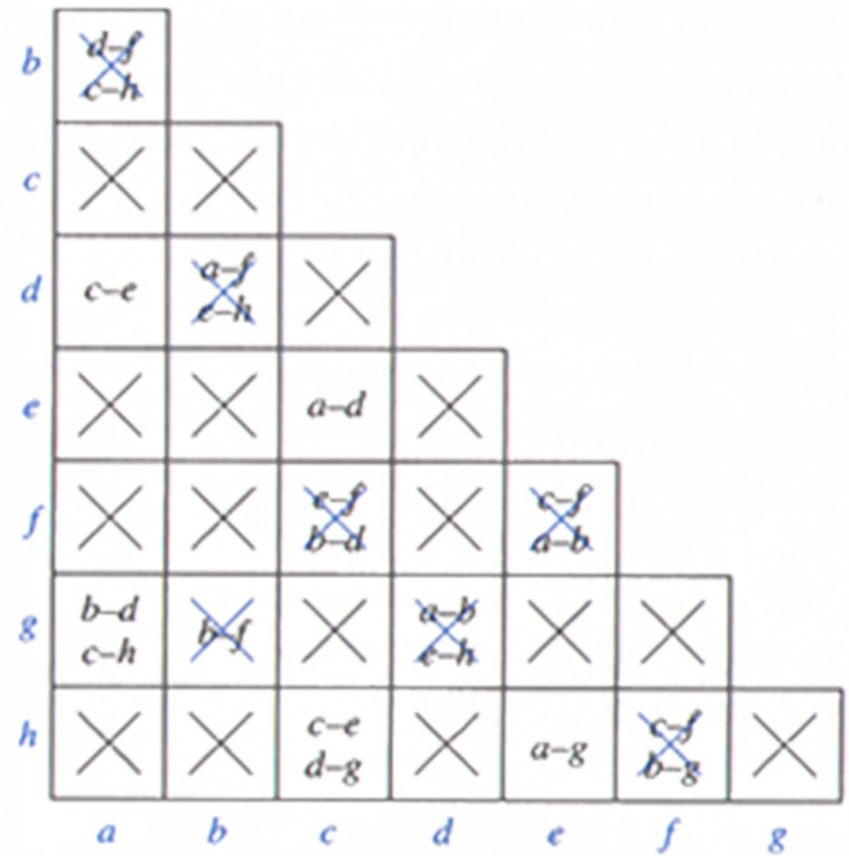




Next pass

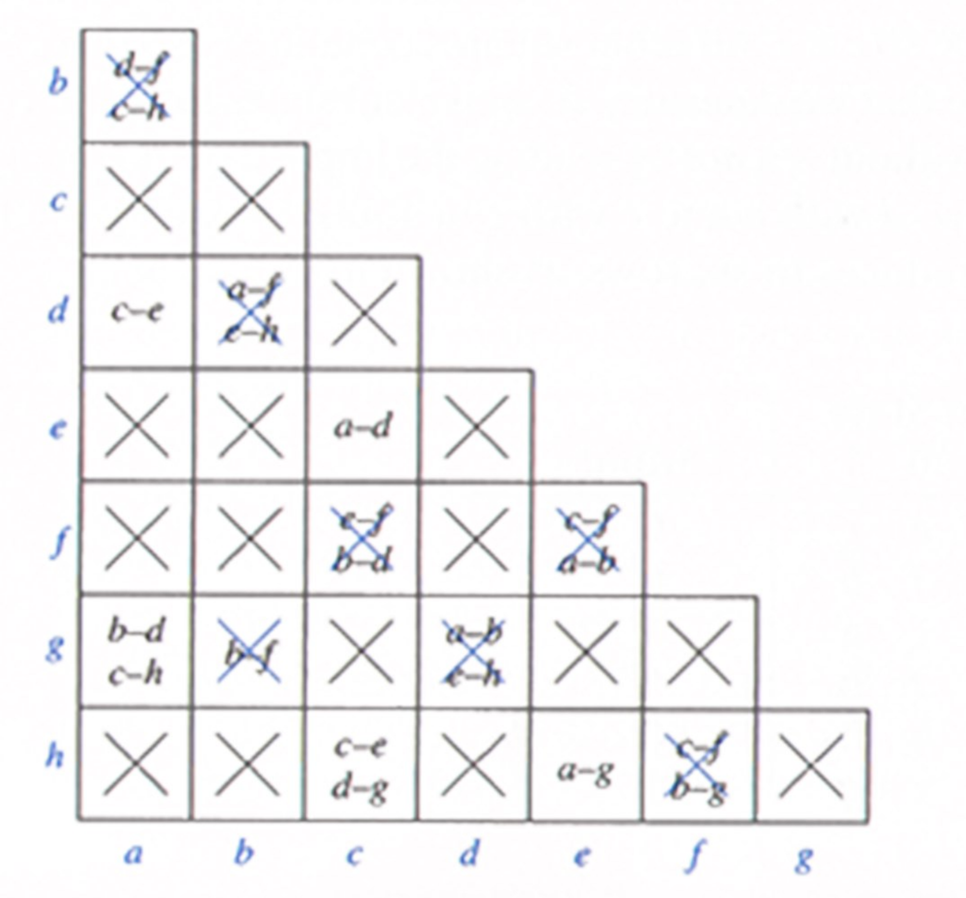
- ✓ X all squares with implied pairs that are not compatible.
- ✓ Such as in a-b have d-f which has an X in it.
- ✓ Run through the chart until no further X's are found.

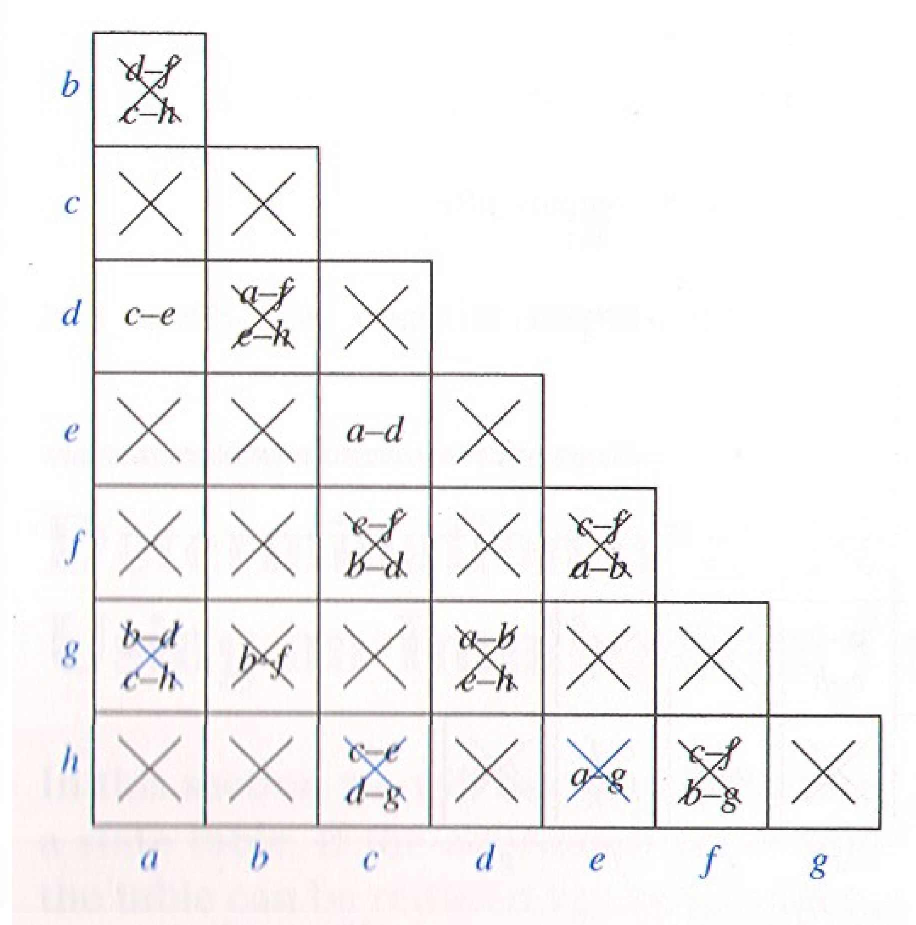




Final step

- ✓ Note that a-d is not X and is equivalent if c≡e, and the same for is c-e: is not X and is equivalent if a≡d. We can conclude that a≡d., i.e. and c≡e.





Reduced table

✓ Removing equivalent states.

Present State	Next State $X = 0$	1	Present Output
a	d	c	0
b	f	h	0
c	e	d	1
d	a	e	0
e	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1

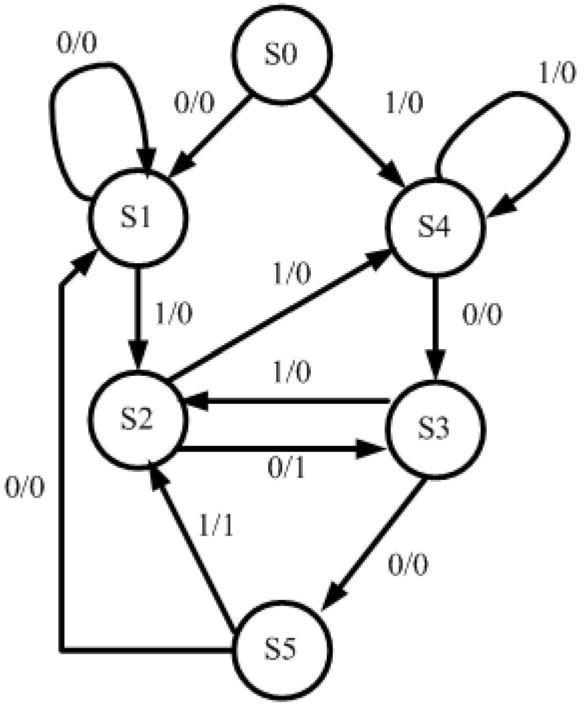
Present State	Next State $X = 0$	1	Present Output
a	a	c	0
b	f	h	0
c	c	a	1
f	f	b	1
g	b	h	0
h	c	g	1

Summary of method

- ✓ Construct a chart with a square for each pair of states.
- ✓ Compare each pair of rows in the state table. X a square if the outputs are different. If the output is the same enter the implied pairs. Remove redundant pairs. If the implied pair is the same place a check mark as $i \equiv j$.
- ✓ Go through the implied pairs and X the square when an implied pair is incompatible.
- ✓ Repeat until no more Xs are added.
- ✓ For any remaining squares not Xed, $i \equiv j$.

Another example

✓ Consider the state diagram:



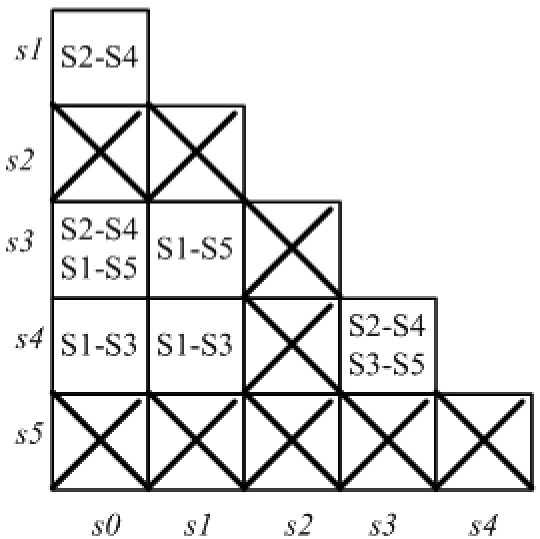
	NEXT STATE		OUTPUT	
Present State	**X=0**	**X=1**	**X=0**	**X=1**
S0	S1	S4	0	0
S1	S1	S2	0	0
S2	S3	S4	1	0
S3	S5	S2	0	0
S4	S3	S4	0	0
S5	S1	S2	0	1

Set up Implication Chart

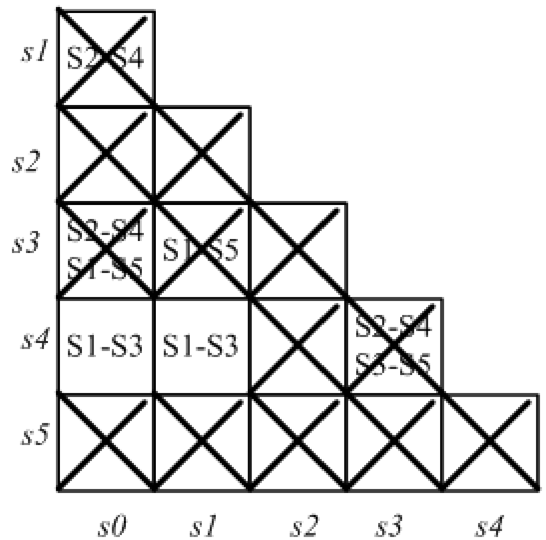
	NEXT STATE		OUTPUT	
Present State	**X=0**	**X=1**	**X=0**	**X=1**
S0	S1	S4	0	0
S1	S1	S2	0	0
S2	S3	S4	1	0
S3	S5	S2	0	0
S4	S3	S4	0	0
S5	S1	S2	0	1

- ✓ Remove output incompatible states
- ✓ and indicate implied pairs

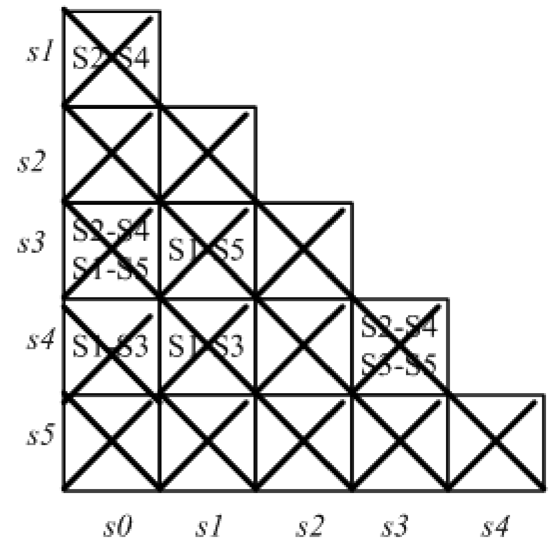
Check implied pairs and X



1st pass



2nd pass



- ✓ In this case the state table is minimal as no state reduction can be done.

Implication Table (another example)

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
a	*d*	*b*	0	0
b	*e*	*a*	0	0
c	*g*	*f*	0	1
d	*a*	*d*	1	0
e	*a*	*d*	1	0
f	*c*	*b*	0	0
g	*a*	*e*	1	0

b	*d* , *e* ✓					
c	×	×				
d	×	×	×			
e	×	×	×	✓		
f	*c* , *d* ×	*c* , *e* × *a* , *b*	×	×	×	
g	×	×	×	*d* , *e* ✓	*d* , *e* ✓	×
	a	*b*	*c*	*d*	*e*	*f*

- ✓ Its clear that (e,d) are equivalent. And this leads (a,b) and (e,g) to be equivalent too.
- ✓ Finally we have [(a,b) , c , (e,d,g) , f] so four states.

Implication Table

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
a	*d*	*b*	0	0
b	*e*	*a*	0	0
c	*g*	*f*	0	1
d	*a*	*d*	1	0
e	*a*	*d*	1	0
f	*c*	*b*	0	0
g	*a*	*e*	1	0

b	*d*, *e* ✓					
c	×	×				
d	×	×	×			
e	×	×	×	✓		
f	*c*, *d* ×	*c*, *e* *a*, *b* ×	×	×	×	
g	×	×	×	*d*, *e* ✓	*d*, *e* ✓	×
	a	*b*	*c*	*d*	*e*	*f*

Present State	Next State		Output	
	X=0	X=1	X=0	X=1
a	*d*	*a*	0	0
c	*d*	*f*	0	1
d	*a*	*d*	1	0
f	*c*	*a*	0	0