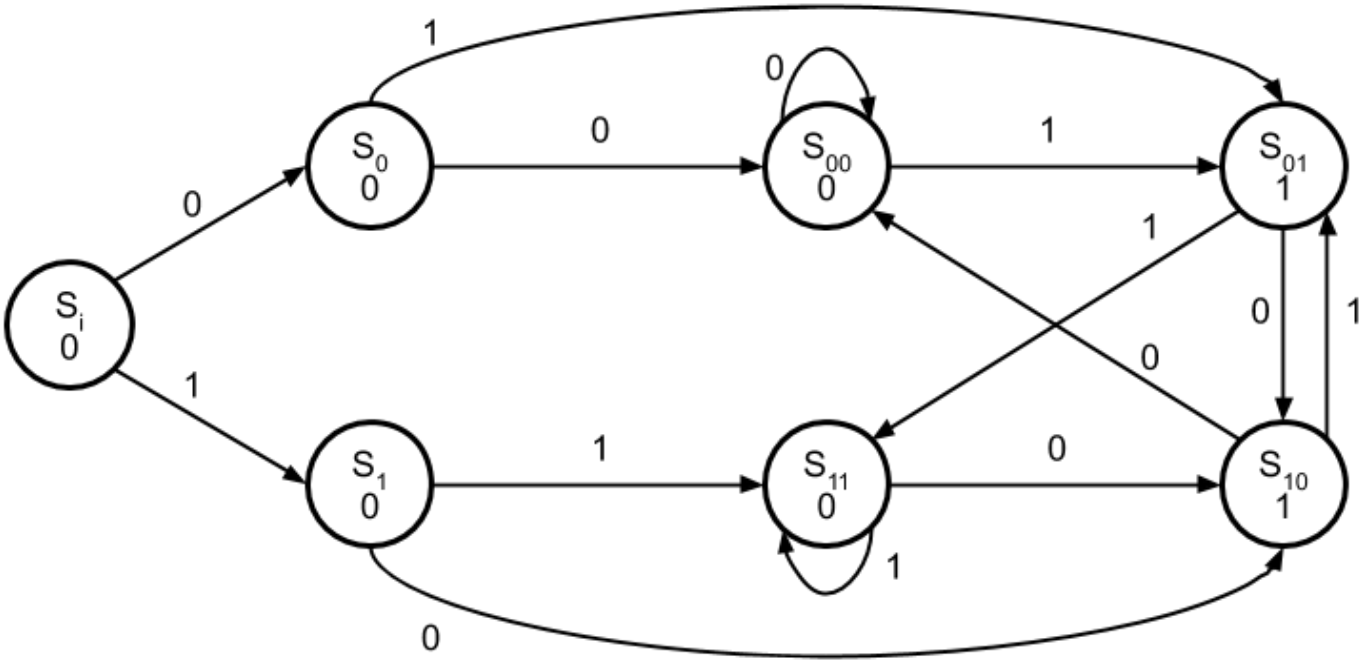
1. Minimization



Original state diagram

Minimized state diagram

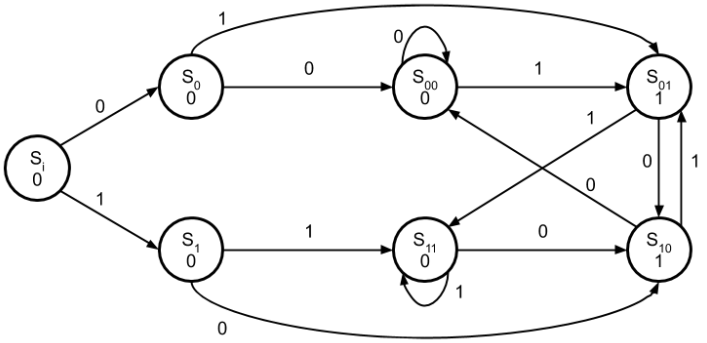
2. Some definitions
    a. *Equivalent*

    b. *Successor*

    c. *Block*

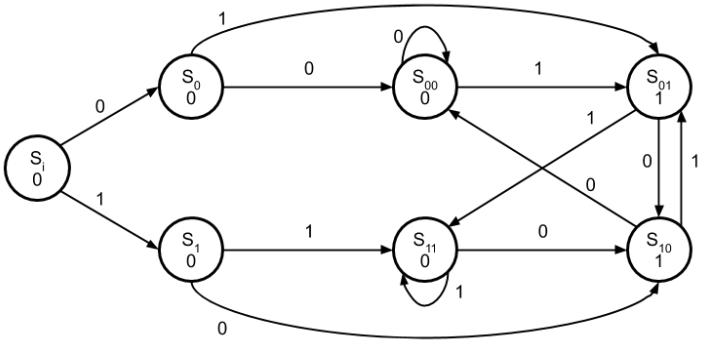
    d. *Partition*

3. Partition minimization procedure
    a. Will use the unminimized edge detector FSM for the rest of this example



    b. P_1

c. P_2

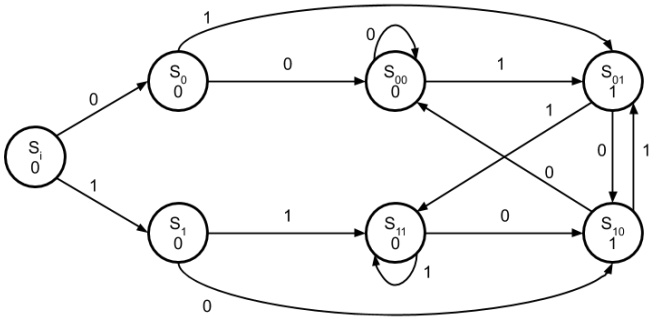


d. P_3

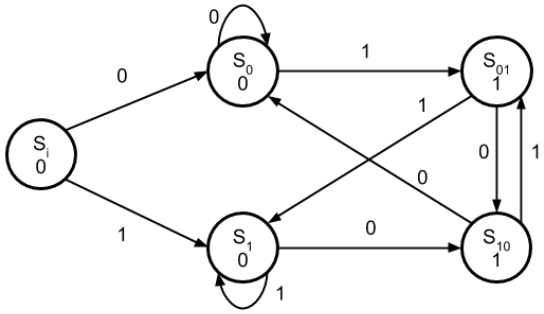
e. Further partitions

f. Look at P_4 now

g. Now use this to create minimized state diagram



Original state diagram



Minimized state diagram

4. Implementing the minimized FSM

Present State	Next State		Output
	$x = 0$	$x = 1$	z
i	0	1	0
0	00	01	0
1	10	11	0
00	00	01	0
01	10	01	1
10	00	01	1
11	10	11	0

Original state table

Present State	Next State		Output
	$x = 0$	$x = 1$	z
i			
0			
1			
01			
10			

Minimized state table

a. Next, assign binary codes in state transition table

Present State	Next State		Output
	$x = 0$	$x = 1$	z
i	0	1	0
0	0	01	0
1	10	1	0
01	10	1	1
10	0	01	1

Present State	Binary Code	Present State			Input	Next State			Output
		A	*B*	*C*	*x*	*A'*	*B'*	*C'*	*z*
i	000								
i	000								
0	001								
0	001								
1	010								
1	010								
01	011								
01	011								
10	100								
10	100								

Present State	Binary Code	Present State			Input	Next State			Output
		A	*B*	*C*	*x*	*A'*	*B'*	*C'*	*z*
i	000	0	0	0	0	0	0	1	0
i	000	0	0	0	1	0	1	0	0
0	001	0	0	1	0	0	0	1	0
0	001	0	0	1	1	0	1	1	0
1	010	0	1	0	0	1	0	0	0
1	010	0	1	0	1	0	1	0	0
01	011	0	1	1	0	1	0	0	1
01	011	0	1	1	1	0	1	0	1
10	100	1	0	0	0	0	0	1	1
10	100	1	0	0	1	0	1	1	1

b. Create K-maps for each flip flop based on input and present state
c. Create a K-Map based on flip-flops to determine the output combinational circuit

A'

Cx \ AB	00	01	11	10
00				
01				
11				
10				

B'

Cx \ AB	00	01	11	10
00				
01				
11				
10				

C'

Cx \ AB	00	01	11	10
00				
01				
11				
10				

z

C \ AB	00	01	11	10
0				
1				