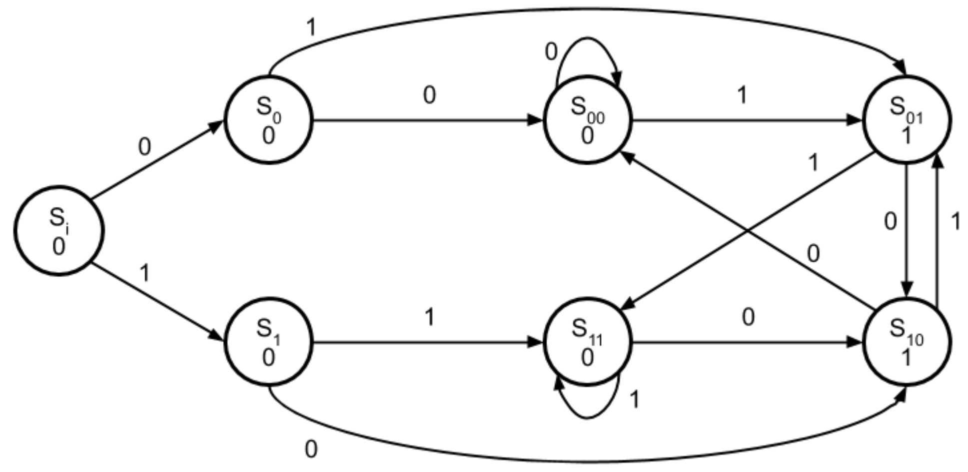
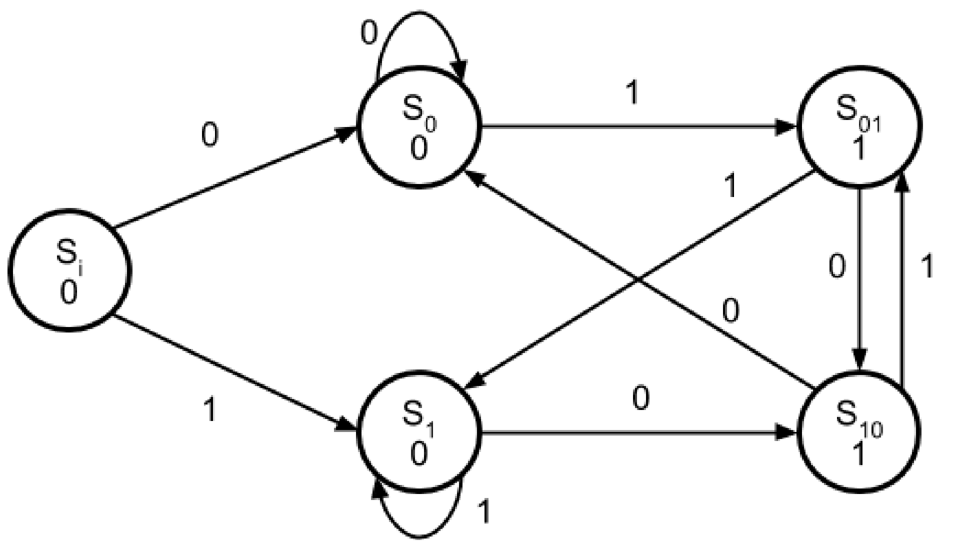
1. Minimization
    a. Did the 7-state Moore model edge detector earlier
    b. Can minimize this down to 5 states (as shown below) and still have functional equivalency



Original state diagram

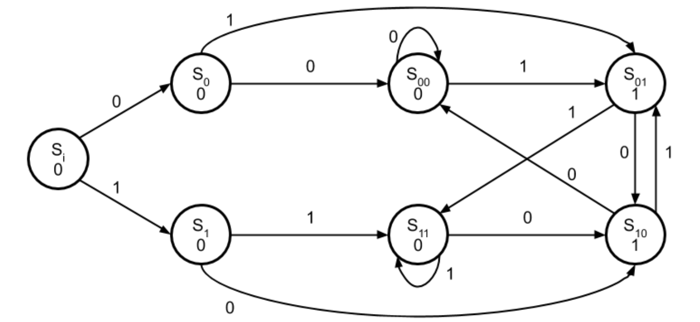


Minimized state diagram

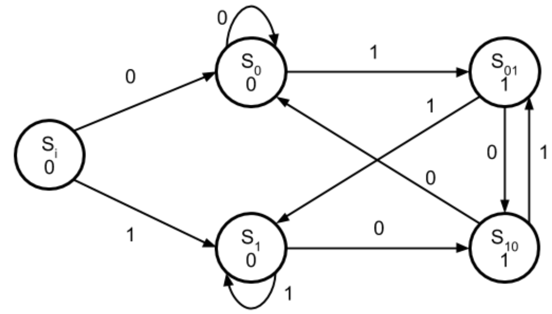
2. Some definitions
    a. Two states S_i and S_j are *equivalent* if and only if for every possible input sequence, the same output sequence will be produced, regardless of whether S_i or S_j is the initial state
    b. A *successor* to state S_i is a state that it transitions to, based on its input
        i. S_0 in the unminimized FSM has S_{00} and S_{01} as its successors
        ii. Differentiate successors based on input
            1. S_{00} is the *0-successor* of S_0
            2. S_{01} is the *1-successor* of S_0
            3. Collectively, all immediate successors of a state form the *k-successors* of the state
    c. A *block* is a subset of states that may be equivalent
    d. A *partition* is a set of blocks where the states in each block are not equivalent to the states in the other blocks

3. Partition Minimization Procedure
    a. Will use the unminimized edge detector FSM for the rest of this example
    b. Start with all states in one partition and in same block
        i. $P_1 = (S_i, S_0, S_1, S_{00}, S_{01}, S_{10}, S_{11})$
    c. Create P_2 by dividing states in P_1 that have same outputs
        i. From definition of *equivalent*, states that have different outputs cannot be equivalent
        ii. $P_2 = (S_i, S_0, S_1, S_{00}, S_{11})\ (S_{01}, S_{10})$
    d. Create P_3 by looking at k-successors of each state
        i. States of a block that have k-successors that in are different blocks from others in the block must be placed in new blocks, grouped by their shared k-successors
        ii. Look at first block $(S_i, S_0, S_1, S_{00}, S_{11})$
            1. 0-successors for the $(S_i, S_0, S_1, S_{00}, S_{11})$ block of P_2 are $S_0, S_{00}, S_{10}, S_{00}, S_{10},$ respectively
                a. Need to divide states into those that stay in the block, and those that move to the (S_{01}, S_{10}) of P_2
                b. Thus, we get (S_i, S_0, S_{00}) and (S_1, S_{11})
            2. 1-successors for (S_i, S_0, S_{00}) are S_1, S_{01}, S_{01}, respectively
                a. So, we divide into (S_i) and (S_0, S_{00})
            3. 1-succesors for (S_1, S_{11}) are S_{11}, S_{11}, so it will not need to be divided
            4. First block of P_2 will be divided into the three blocks (S_i), (S_1, S_{11}) and (S_0, S_{00}) in P_3
        iii. Now look at second block (S_{01}, S_{10})
            1. 0-successors for the (S_{01}, S_{10}) block of P_2 are S_{10}, S_{00}, respectively
                a. These are in different blocks from each other in P_2
            2. Will have to divide into two separate blocks (S_{01}), and (S_{10})
        iv. So $P_3 = (S_i)(S_1, S_{11})(S_0, S_{00})(S_{01})(S_{10})$
    e. Further partitions
        i. Once a block has one state in it, don't need to partition it further
            1. Will still need to use it to determine k-successors, though
        ii. P_4 and further partitions look at each multiple-element block of the previous partition to see if the k-successors of its elements lead to the same blocks of the previous partition
            1. If not, the block must be further divided
            2. If any block splits, then we must continue to another step of partitioning
            3. If no block splits, then we are done
        iii. Look at P_4 now
            1. 0-successors of (S_1, S_{11}) are both block (S_{10}), so that will not cause the block to split
            2. 1-successors are both (S_1, S_{11}), so there is no need to separate the block further during this partitioning step
            3. 0-successors of (S_0, S_{00}) are both block (S_0, S_{00}), so that will not cause the block to split
            4. 1-successors are both (S_{01}), so there is no need to separate the block further during this partitioning step
        iv. Since no block split, the final minimized partition is $P_4 = (S_i)(S_1, S_{11})(S_0, S_{00})(S_{01})(S_{10})$
            1. This matches the five-state minimized state diagram
    f. Special case involving the initial state *i*
        i. Remember that we start our FSMs in the initial state *i*
        ii. If *i* gets combined with other states, we still start in that initial state
        iii. Didn't happen above, but could possibly happen in other minimizations

g. Now use this to create minimized state diagram
    i. K-successors are the same for the multiple-state blocks, use that to combine them together
    ii. Use names of states 0 and 1 as new names for those combined states



Original state diagram



Minimized state diagram

4. Implementing the minimized FSM
   a. From our minimized (equivalent) FSM we get the following state table

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	0
0	0	01	0
1	10	1	0
01	10	1	1
10	0	01	1

Minimized state table

   b. Next, assign binary codes in state transition table
      i. Will need 3 flip flops to represent 5 states, call these A, B, and C
      ii. Again, have done assignment in order here, but not necessarily optimal
         1. Will talk about how to determine optimal state binary code placement later

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

c. Create K-maps for each flip flop based on input and present state
    i. States that weren't assigned form don't cares, like in original FSM

A'

Cx \ AB	00	01	11	10
00	0	1	d	0
01	0	0	d	0
11	0	0	d	d
10	0	1	d	d

$A' = B\bar{x}$

B'

Cx \ AB	00	01	11	10
00	0	0	d	0
01	1	1	d	1
11	1	1	d	d
10	0	0	d	d

$B' = x$

C'

Cx \ AB	00	01	11	10
00	1	0	d	1
01	0	0	d	1
11	1	0	d	d
10	1	0	d	d

$C' = A + \overline{Bx} + \overline{B}C$

d. Use derivations from these K-maps to design initial combinational circuit
e. Create a K-Map based on flip-flops to determine the output combinational circuit
    i. Assign don't cares for the same reason as above

z

C \ AB	00	01	11	10
0	0	0	d	1
1	0	1	d	d

$z = A + BC$