State reduction and more datapath

In these notes, there are two topics:

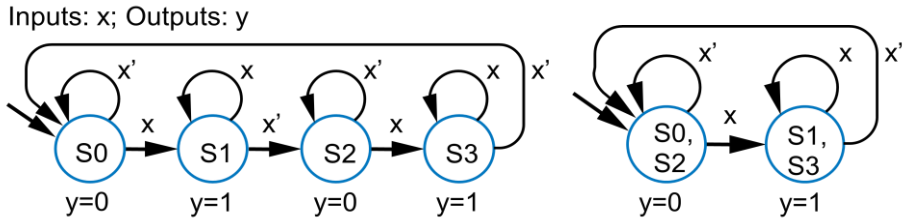
- Finding the minimal number of states for a state machine.
- A nice example of using a bus-based data path. We probably won't get to that (at all) in lecture, but it's a nice enough example you may want to work it and if we have 20 minutes extra at some point (which is quite possible) we'll come back and do it then.

State minimization

One thing to be aware of is that the 2nd edition of our text somehow got *worse* at covering this material. I've posted the relevant 6 pages from the 1st edition on the website, you should take a look at it.

Motivational Example

Consider the following two machines. Are they equivalent?



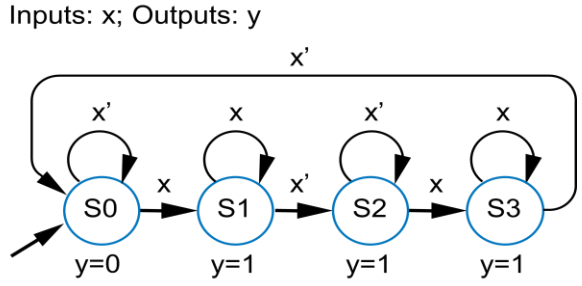
Say x were 0, 1, 1, 0, 0. What would be the output of the larger one? The smaller one? Are they *always* the same? How can you be sure?

What makes states equivalent?

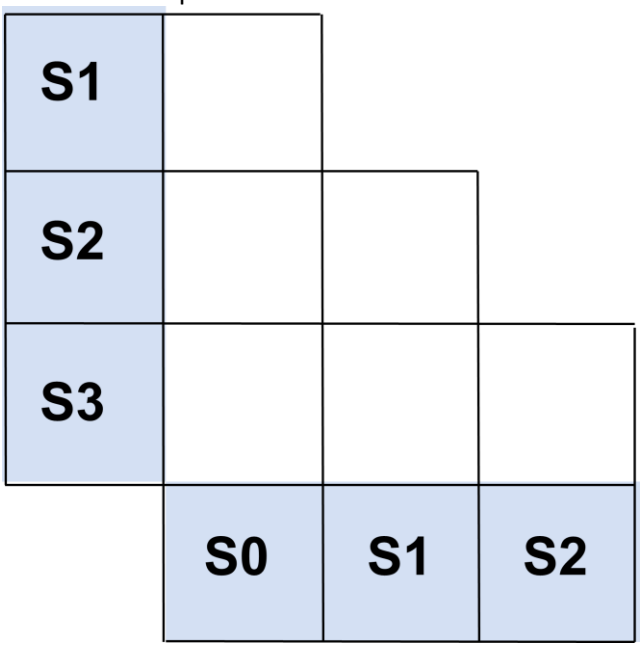
Two states are equivalent if

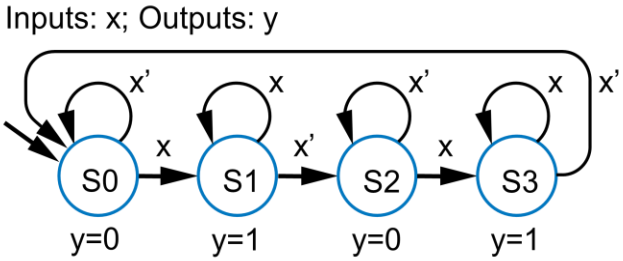
1. They assign the same values to outputs,
   e.g. S0 and S2 both assign y to 0,S1 and S3 both assign y to 1
2. AND, for all possible sequences of inputs, the FSM outputs will be the same starting from either state

Let's look at this example. Are any states equivalent?



Let's use an implication table.





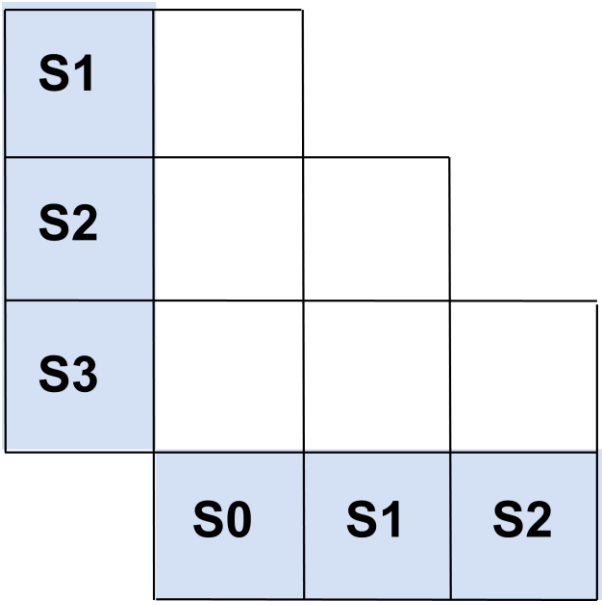
Which states can't be equivalent? Any with differing outputs! Cross those out.

Now we'll follow the following algorithm. Do steps 2&3.

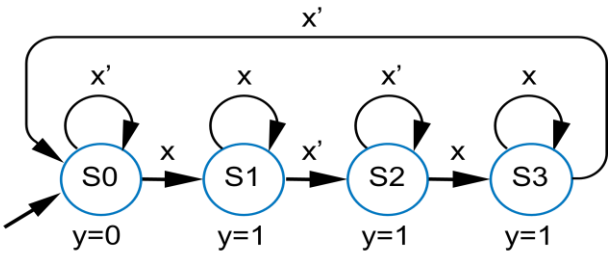
	Step	Description
1	*Mark state pairs having different outputs as nonequivalent*	States having different outputs obviously cannot be equivalent.
2	*For each unmarked state pair, write the next state pairs for the same input values*	
3	*For each unmarked state pair, mark state pairs having nonequivalent next-state pairs as nonequivalent. Repeat this step until no change occurs, or until all states are marked.*	States with nonequivalent next states for the same input values can't be equivalent. Each time through this step is called a ***pass***.
4	*Merge remaining state pairs*	Remaining state pairs must be equivalent.

What does this look like after step 4?

Another Example

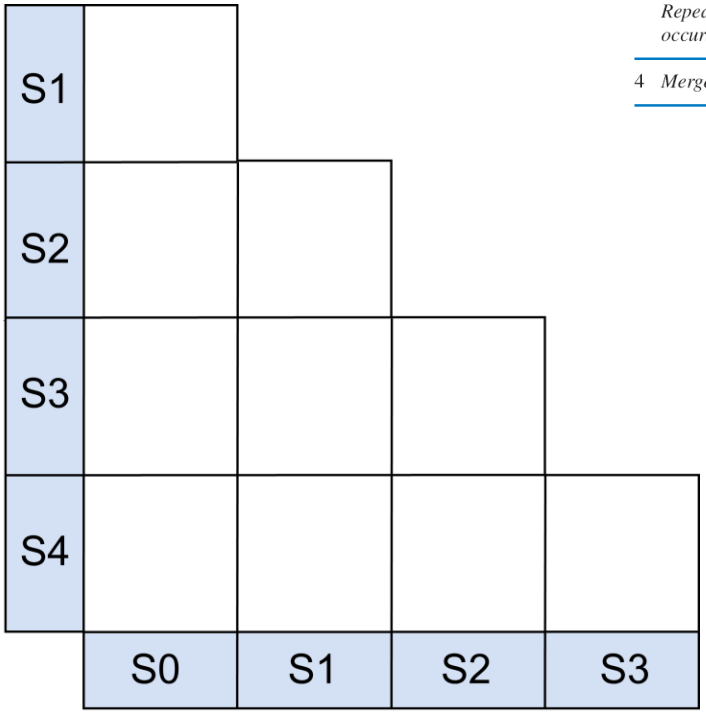


Inputs: x; Outputs: y

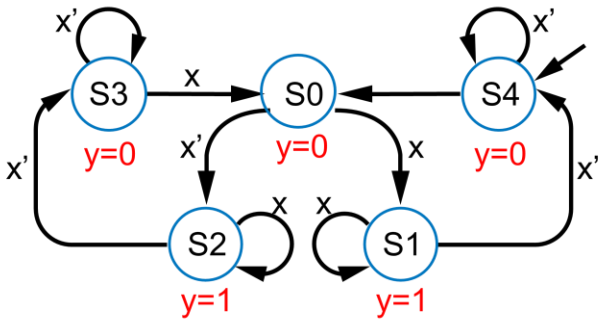


	Step	Description
1	*Mark state pairs having different outputs as nonequivalent*	States having different outputs obviously cannot be equivalent.
2	*For each unmarked state pair, write the next state pairs for the same input values*	
3	*For each unmarked state pair, mark state pairs having nonequivalent next-state pairs as nonequivalent. Repeat this step until no change occurs, or until all states are marked.*	States with nonequivalent next states for the same input values can't be equivalent. Each time through this step is called a ***pass***.
4	*Merge remaining state pairs*	Remaining state pairs must be equivalent.

Now let's do a final, less trivial, example:



Inputs: x; Outputs: y



Bus-based datapath

We probably won't get to this in class. It is an example from Engineering 100, section 250 and it's a fine practice example involving memory.

Design a machine which finds the maximum value of the first 16 memory elements and puts the result in the 17th one.