Lecture 25

- Logistics
  - HW8 posted today, due 12/5
  - Lab9 this week
  - No Class for the rest of the week!
- Last lecture
  - Robot ant in maze
  - Started on FSM simplification a little bit
- Today
  - More on FSM simplification

The "WHY" slide

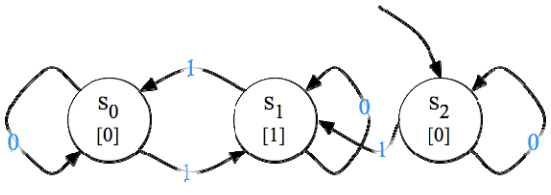
- FSM minimization
  - It is best to minimize FSM before expressing it as a logic circuit. As you saw in the ant robot example, minimization step is about looking for some patterns and merging states. There are systematic ways to do this (rather than the way we'd done it for the ant example) and we will learn them here.

Two Methods for FSM Minimization

- Row matching
  - Easier to do by hand
  - Misses minimization opportunities
- Implication table
  - Guaranteed to find the most reduced FSM
  - More complicated algorithm (but still relatively easy to write a program to do it)

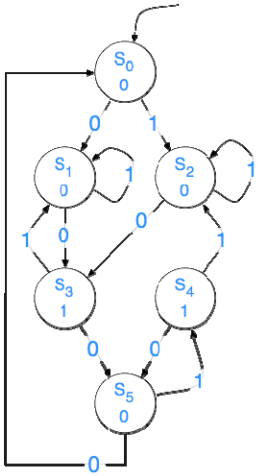
Simple row matching does not guarantee most reduced state machine

	Next State		
Present State	X=0	X=1	Output
S_0	S_0	S_1	0
S_1	S_1	S_2	1
S_2	S_2	S_1	0

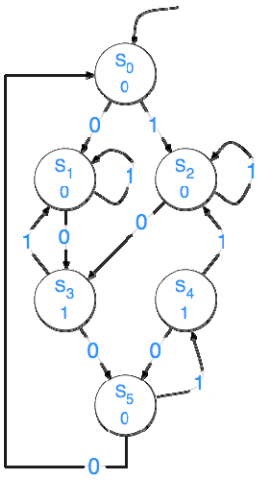


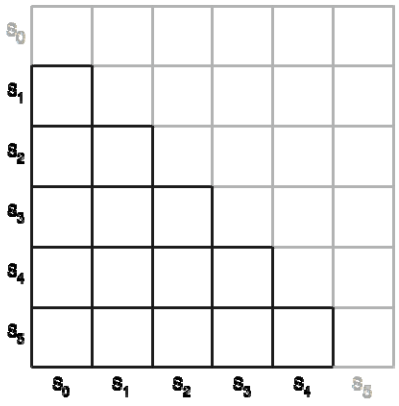
The Implication chart method

- Here's a slightly funkier FSM as an example

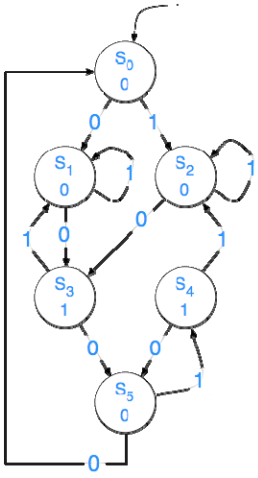


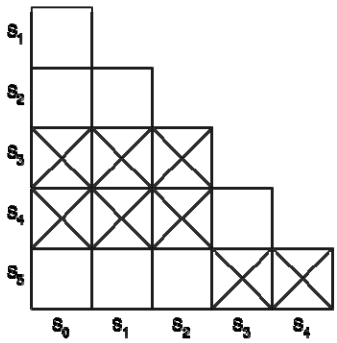
Step 1: Draw the table



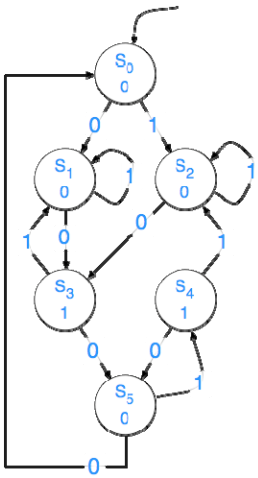


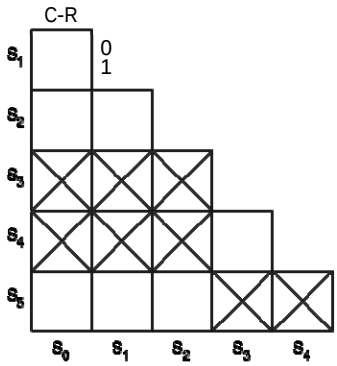
Step 2: Consider the outputs



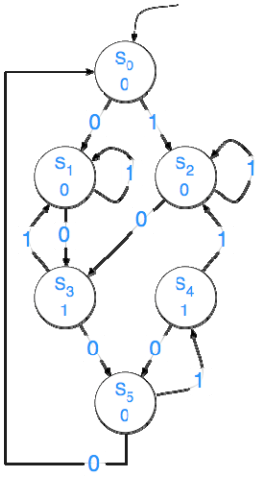


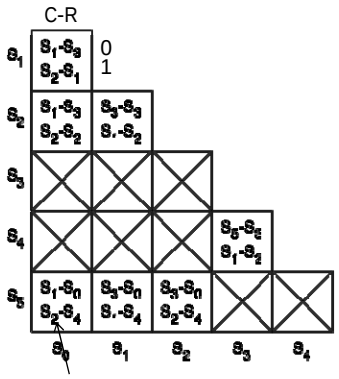
Step 3: Add transition pairs





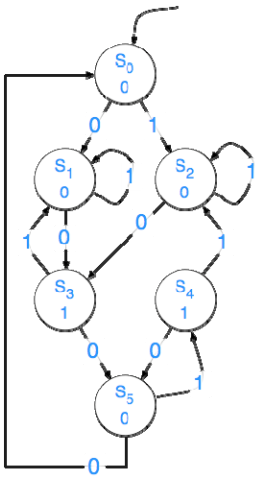
Step 3: Add transition pairs

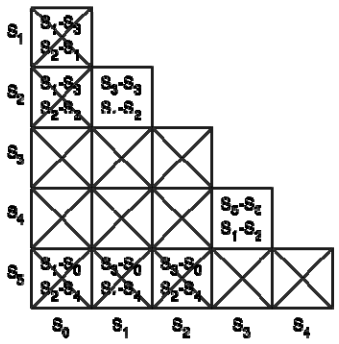




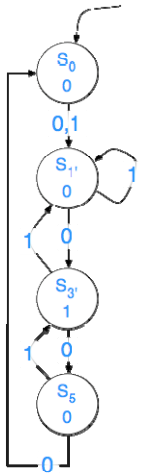
Implied State Pairs

Step 4 (repeated): Consider transitions

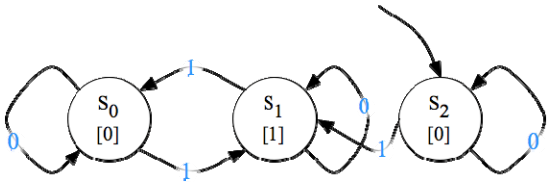




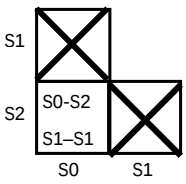
Final reduced FSM



Odd parity checker revisited

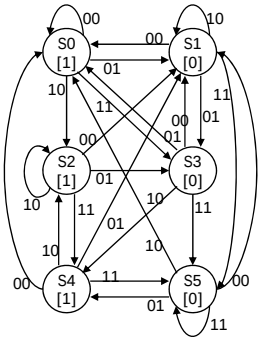


	Next State		
Present State	X=0	X=1	Output
S_0	S_0	S_1	0
S_1	S_1	S_2	1
S_2	S_2	S_1	0



More complex state minimization

- Multiple input example



inputs here

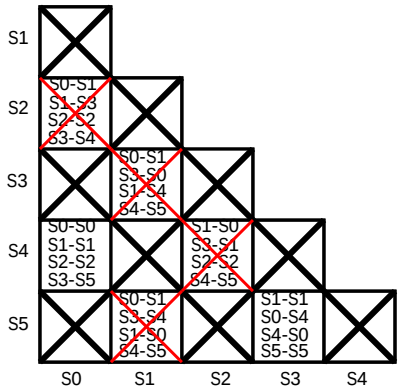
present state	next state 00	01	10	11	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S4	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

symbolic state transition table

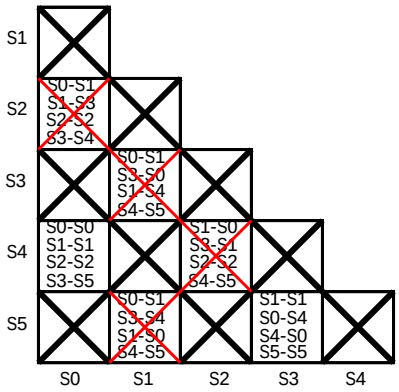
Minimized FSM

- Implication chart method
  - cross out incompatible states based on outputs
  - then cross out more cells if indexed chart entries are already crossed out

present state	next state 00	01	10	11	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S4	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0



Minimized FSM



present state	next state 00	01	10	11	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S4	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

present state	next state 00	01	10	11	output
S0'	S0'	S1	S2	S3'	1
S1	S0'	S3'	S1	S3'	0
S2	S1	S3'	S2	S0'	1
S3'	S1	S0'	S0'	S3'	0

minimized state table
(S0==S4) (S3==S5)

Minimizing incompletely specified FSMs

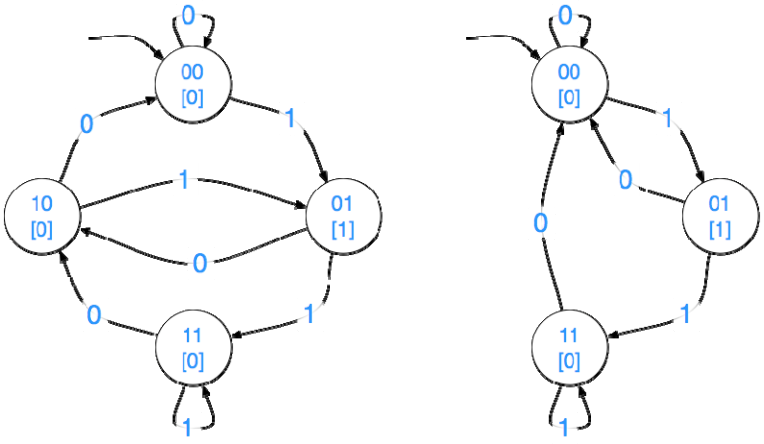
- Equivalence of states is transitive when machine is fully specified
- But its not transitive when don't cares are present

  e.g., state output
  - S0 X 0 S1 is compatible with both S0 and S2
  - S1 1 X but S0 and S2 are incompatible
  - S2 X 1

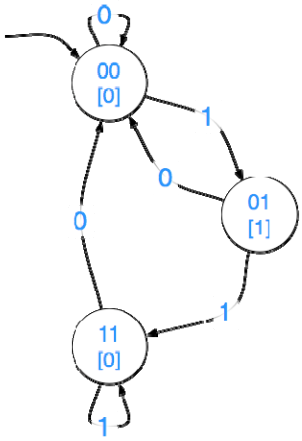
- Hard to determining best grouping of states to yield the smallest number of final states

Minimizing FSMs isn't always good

- Two FSMs for 0->1 edge detection



Minimal state diagram -> not necessarily best circuit



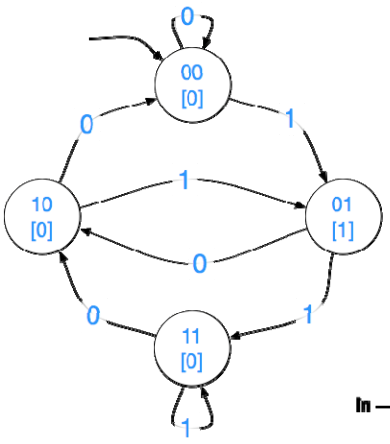
In	Q_1	Q_0	Q_1^+	Q_0^+
0	0	0	0	0
0	0	1	0	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	1
1	1	1	1	1
–	1	0	0	0

$Q_1^+ = In\ (Q_1 \text{ xor } Q_0)$

$Q_0^+ = In\ Q_1'\ Q_0'$

$Out = Q_1'\ Q_0$

Minimal state diagram -> not necessarily best circuit

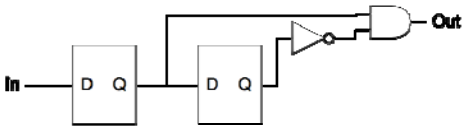


In	Q_1	Q_0	Q_1^+	Q_0^+
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1

$Q_1^+ = Q_0$

$Q_0^+ = In$

$Out = Q_1' Q_0$



A little perspective

- These kinds of optimizations are what CAD(Computer Aided Design)/EDA(Electronic Design Automation) is all about
- The interesting problems are almost always computationally intractable to solve optimally
- People **really** care about the automation of the design of billion-transistor chips