Lecture 22

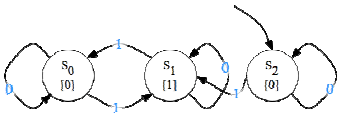
- Logistics
  - HW 8 posted today, due 3/11
  - Lab 9 this week
- Last lecture
  - Robot ant in maze
  - State matching for FSM simplification
- Today
  - General FSM minimization

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)

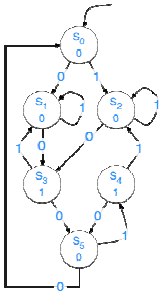
Simple row matching does not guarantee most reduced state machine

	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

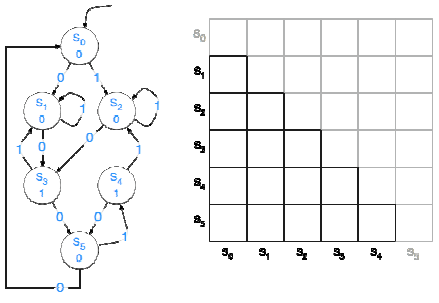
- Here's a slightly funkier FSM as an example



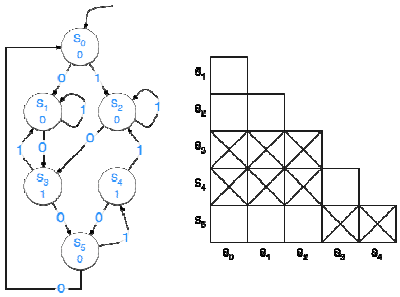
Implication Chart Method

- Basic Idea: assume that all states are grouped together and only split state pairs that must be split
- Algorithm
  1. Draw a table with a box for every pair of states
  2. Use output values to cross out boxes for pairs of states that cannot be combined
  3. Fill rest of table with transition pairs
     - For each box and possible input, list pairs of destination states, one per source state
  4. Repeatedly do:
     - If box (S_i,S_j) contains some transition pair S_k-S_l corresponding to an already crossed-out box then cross out box (S_i,S_j)
     - Until no more boxes can be crossed out
  5. Combine all pairs of states that are not crossed out

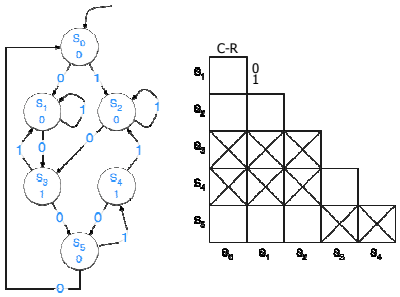
Step 1: Draw the table of pairs of states



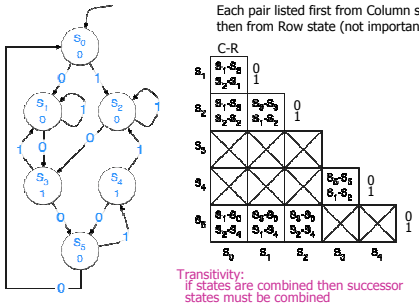
Step 2: Consider the outputs



Step 3: Add transition pairs



Step 3: Add transition pairs



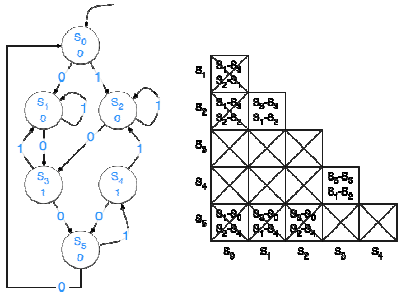
Each pair listed first from Column state then from Row state (not important)

C-R

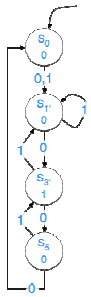
Transitivity:
if states are combined then successor states must be combined

⇒ if successor states cannot be combined then states cannot be combined

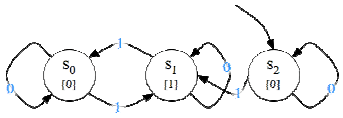
Step 4 (repeated): Consider transitions



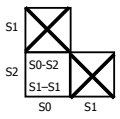
Final reduced FSM



Odd parity checker revisited

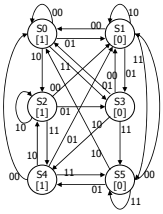


	Next State		
Present State	X=0	X=1	Output
S_0	S_0	S_1	0
S_1	S_1	S_2	1
S_2	S_2	S_1	0



More complex state minimization

- Multiple input example



inputs here

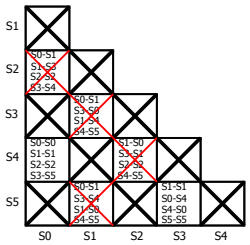
present state	next state 00	01	10	11	output
S0	S0	S1	S2	S3	1
S1	S0	S3	S1	S4	0
S2	S1	S3	S2	S4	1
S3	S1	S0	S4	S5	0
S4	S0	S1	S2	S5	1
S5	S1	S4	S0	S5	0

symbolic state transition table

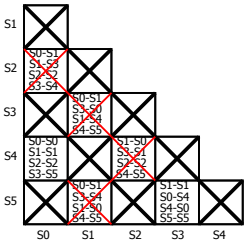
Minimized FSM

- 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)

Optimality and Moore vs Mealy

- Any two (Moore) FSMs with the same functionality lead to the same minimized FSM (up to state names) and this is best possible
  - Proof in CSE 322
- For Mealy FSMs need to do different marking for outputs in step 2
  - List pairs of outputs for each input value and cross out boxes with any pair of different outputs
  - Then erase these output pairs from the non-crossed-out boxes and continue as in Step 3 for the Moore FSM

Minimizing incompletely specified FSMs

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

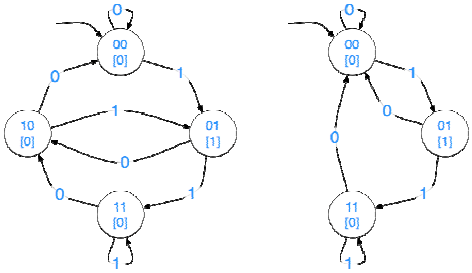
e.g., state output

S0	X 0	S1 is compatible with both S0 and S2
S1	1 X	but S0 and S2 are incompatible
S2	X 1	

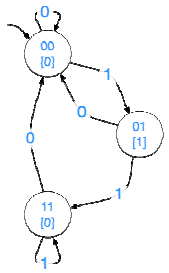
- Hard to determine best grouping of states to yield the smallest number of final states

Minimizing FSMs isn't always good

- Two FSMs for 0 → 1 edge detection



Minimal state diagram: not necessarily best circuit



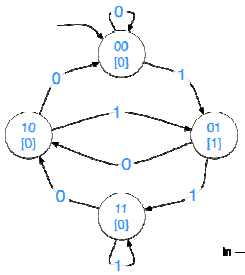
In	Q_1	Q_0	Q_1^+	Q_0^+
0	0	0	0	0
0	0	1	0	0
0	1	1	0	0
1	0	0	0	1
1	0	1	1	1
1	1	1	1	1
–	1	0	–	–

$Q_1^+ = In\ (Q_1 + Q_0)$

$Q_0^+ = In$

$Out = Q_1' Q_0$

Minimal state diagram: not necessarily best circuit

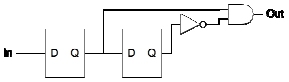


In	Q_1	Q_0	Q_1^+	Q_0^+
0	0	0	0	0
0	0	1	1	0
0	1	0	0	0
0	1	1	1	0
1	0	0	0	1
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1

$Q_1^+ = Q_0$

$Q_0^+ = In$

$Out = Q_1' Q_0$



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