50.002 Computational Structures

Information Systems Technology and Design

Finite State Machines

1 Introduction

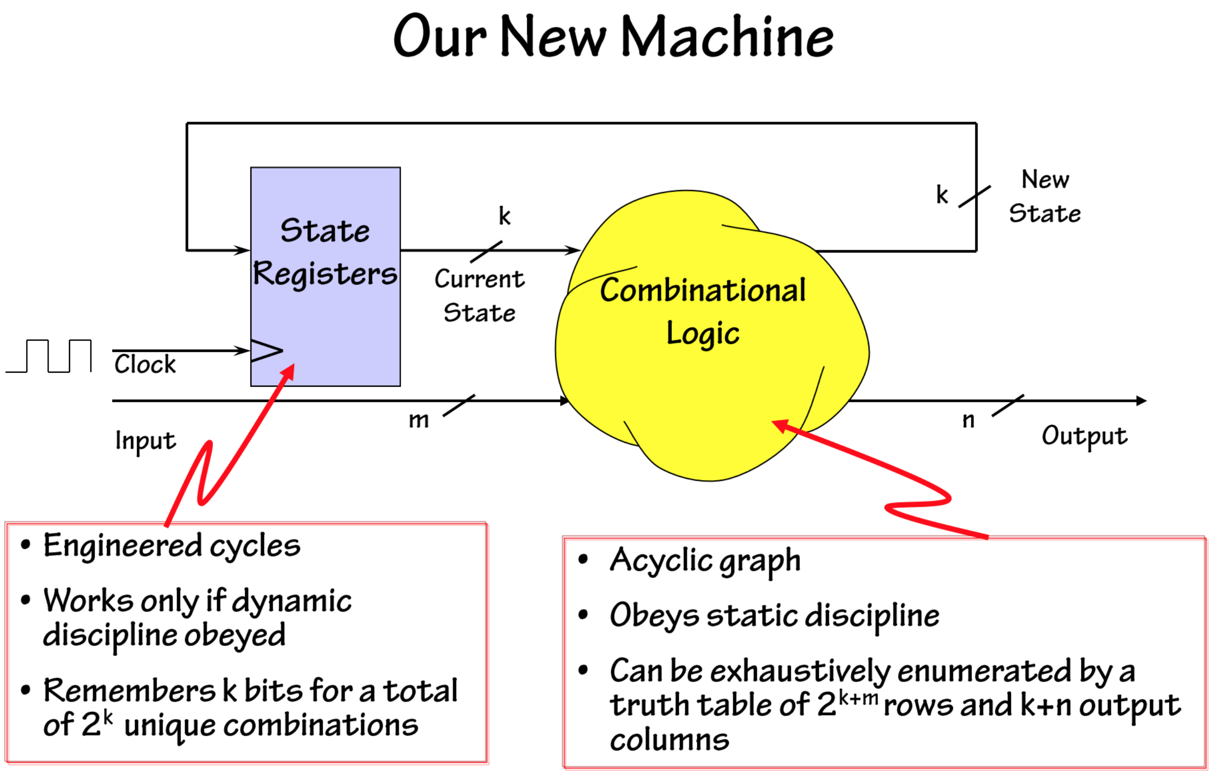

Our New Machine
State Registers
k
Current State
Combinational Logic
k
New State
Clock
Input
m
n
Output
• Engineered cycles
• Works only if dynamic discipline obeyed
• Remembers k bits for a total of 2^k unique combinations
• Acyclic graph
• Obeys static discipline
• Can be exhaustively enumerated by a truth table of 2^k+m rows and k+n output columns

Figure 1

In week 1 and 2 we learned about Combinational Logic. In Week 3, we learned about the state registers (flip flop, memory device). In this note, we are going to learn how to represent states in our machine.

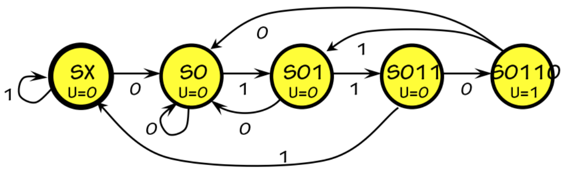
2 Abstraction of Finite State Machine

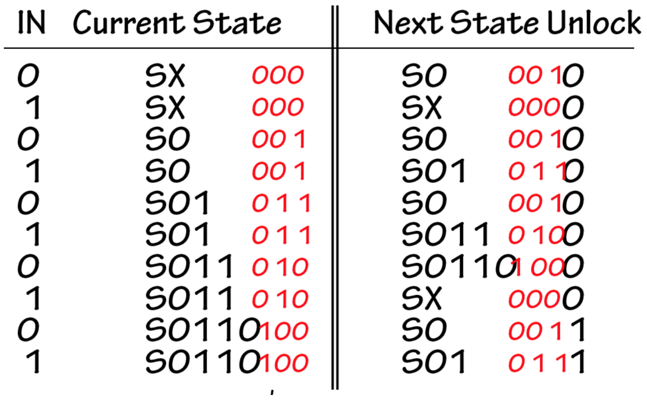
A FSM (Finite State Machine) has:

1. A set of k states: $S_1, S_2, ..., S_k$ (one of them should be the "initial" state)
2. A set of m inputs: $I_1, I_2, ..., I_m$
3. A set of n outputs: $O_1, O_2, ..., O_n$
4. Transition rules $s'(S_i, I_i)$ for each of the k states and m inputs
5. Output rules $o(S)$ for each of the k states

A state machine is required when the same input produces multiple different outputs.

3 State Transition Diagram





IN	Current State		Next State		Unlock
0	SX	000	S0	001	0
1	SX	000	SX	000	0
0	S0	001	S0	001	0
1	S0	001	S01	011	0
0	S01	011	S0	001	0
1	S01	011	S011	010	0
0	S011	010	S0110	100	0
1	S011	010	SX	000	0
0	S0110	100	S0	001	1
1	S0110	100	S01	011	1

Figure 2

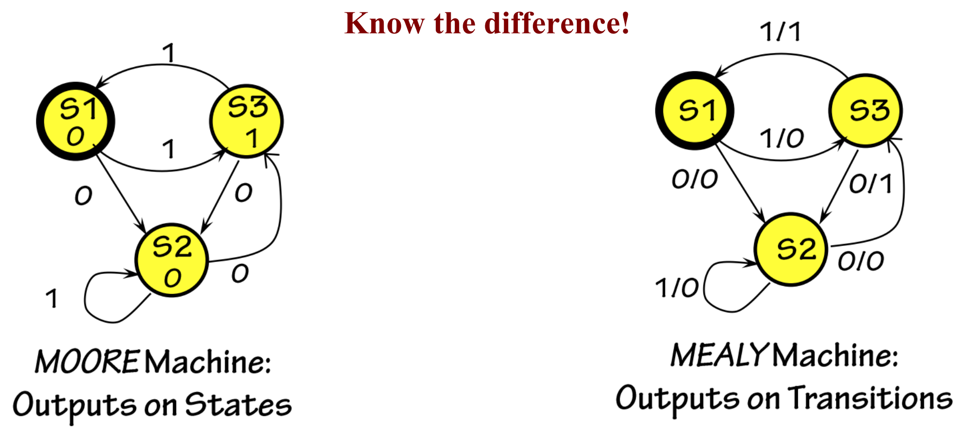
Look at the state transition diagram above. It represents how a "lock" works:

1. This is a digital lock with a password of 0110
2. That SX in bold is the initial state
3. The arrows are the possible transitions between states
4. The little numbers beside the arrows are the kind of input required for state transition to happen

5. The word U inside each state circle is the output of each state. If it is unlocked, $U = 1$, otherwise $U = 0$ for a locked output
6. State transition diagram can be represented in terms of truth table
7. Here we have 5 states, so we need $\log_2(5) = 3$ (rounded up) bits to represent the states
8. Hence, s state bits $= 2^s$ possible states

4 MOORE and MEALY Machine

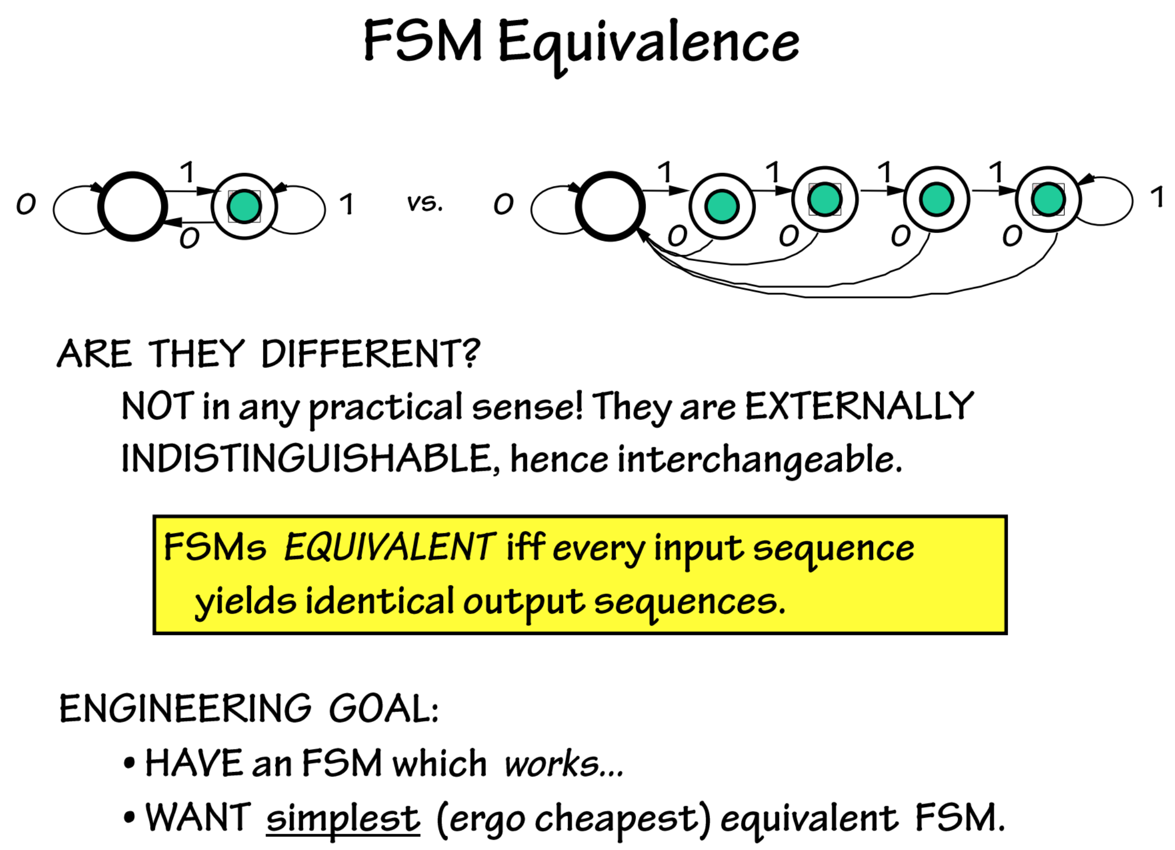
There are two ways to draw the state transition diagram. The one shown in the previous section is called the **Moore** machine, meaning that the output is drawn on the states and **depends on states**. The second type is called the **Mealy** machine, where the output is drawn on the transition arcs and **depends on both inputs and state**:



- Arcs leaving a state must be:
- (1) mutually exclusive
  - *can't have two choices for a given input value*
- (2) collectively exhaustive
  - *every state must specify what happens for each possible input combination. "Nothing happens" means arc back to itself.*

Figure 3

5 FSM Equivalence


0
1
0
1
vs.
0
1
1
1
1
0
0
0
0
1

FSM Equivalence

ARE THEY DIFFERENT?

NOT in any practical sense! They are EXTERNALLY INDISTINGUISHABLE, hence interchangeable.

FSMs *EQUIVALENT* iff every input sequence yields identical output sequences.

ENGINEERING GOAL:

- HAVE an FSM which *works...*
- WANT simplest (ergo cheapest) equivalent FSM.

Figure 4

How to reduce FSM states:

1. Two states S_i and S_j are identical if
   (a) States have identical output
   (b) **Every** input ends (transit to) equivalent states
2. Find pairs of equivalent states, merge them