Finite State Machines

Sources
Gang Quan
Univ. of Notre Dame

Review

- Data Flow Graph
  - data dependency
- Control/Data Flow Graph
  - control dependency
- How about a reactive system?

Finite State Machine

- What ?

If the driver turns on the key, and does not fasten the seat belt within 5 seconds

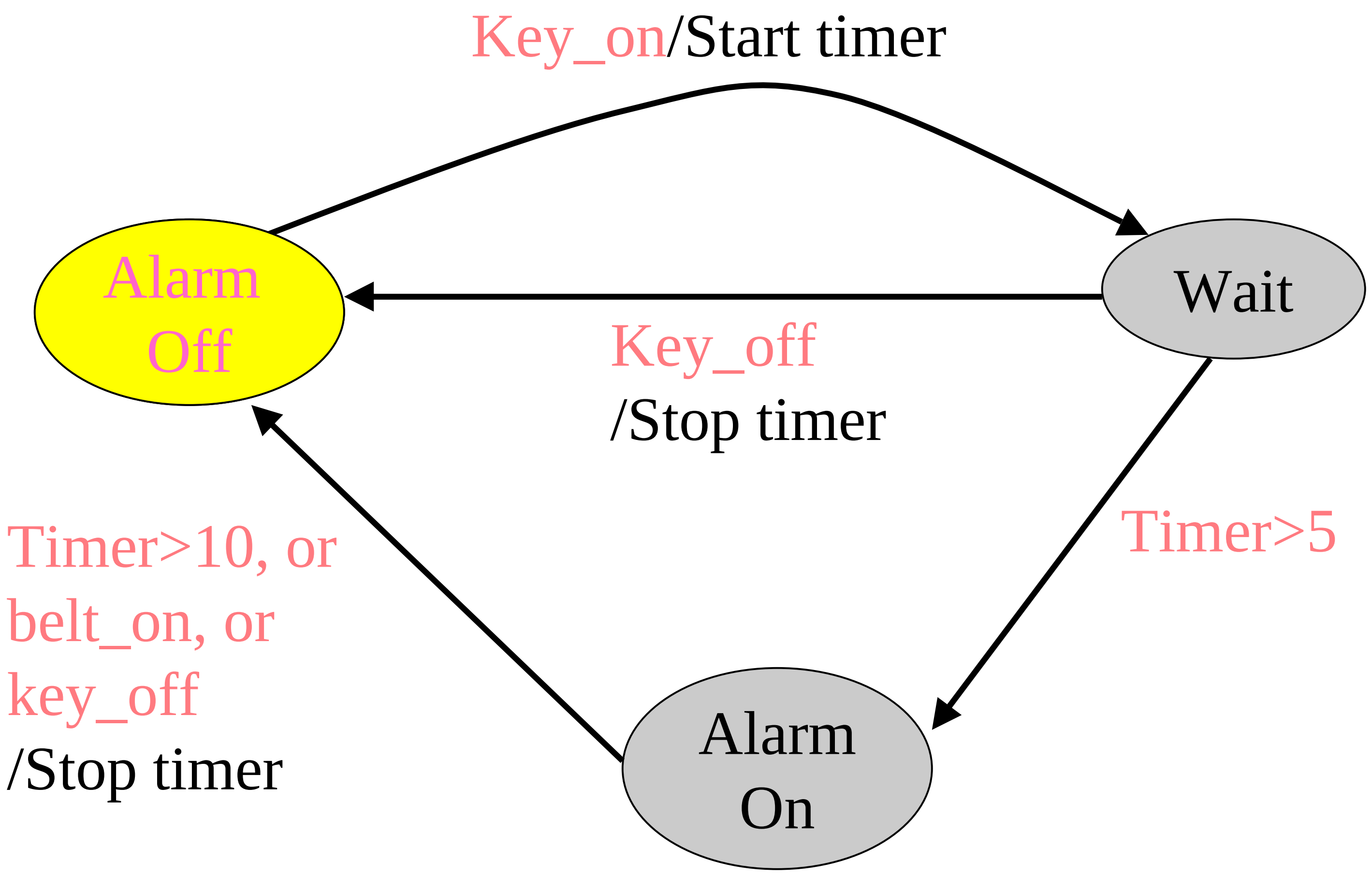
then

an alarm beeps for 5 seconds,

or until the driver fastens the seat belt,

or until the driver turns off the key

An FSM


Key_on/Start timer
Alarm
Off
Wait
Key_off
/Stop timer
Timer>10, or
belt_on, or
key_off
/Stop timer
Timer>5
Alarm
On

An FSM (Cont'd)

- **States**
  - **Alarm off, Alarm on, Wait**
- **Initial State**
  - **Alarm off**
- **Inputs**
  - **Turn on/off the key, fasten the seat belt, timer reads**
- **Outputs**
  - **Start/stop the timer**
- **Start transitions**
  - **Alarm off + Turn on the key → Wait**
- **Output**
  - **Alarm off + Turn on the key → start the timer**

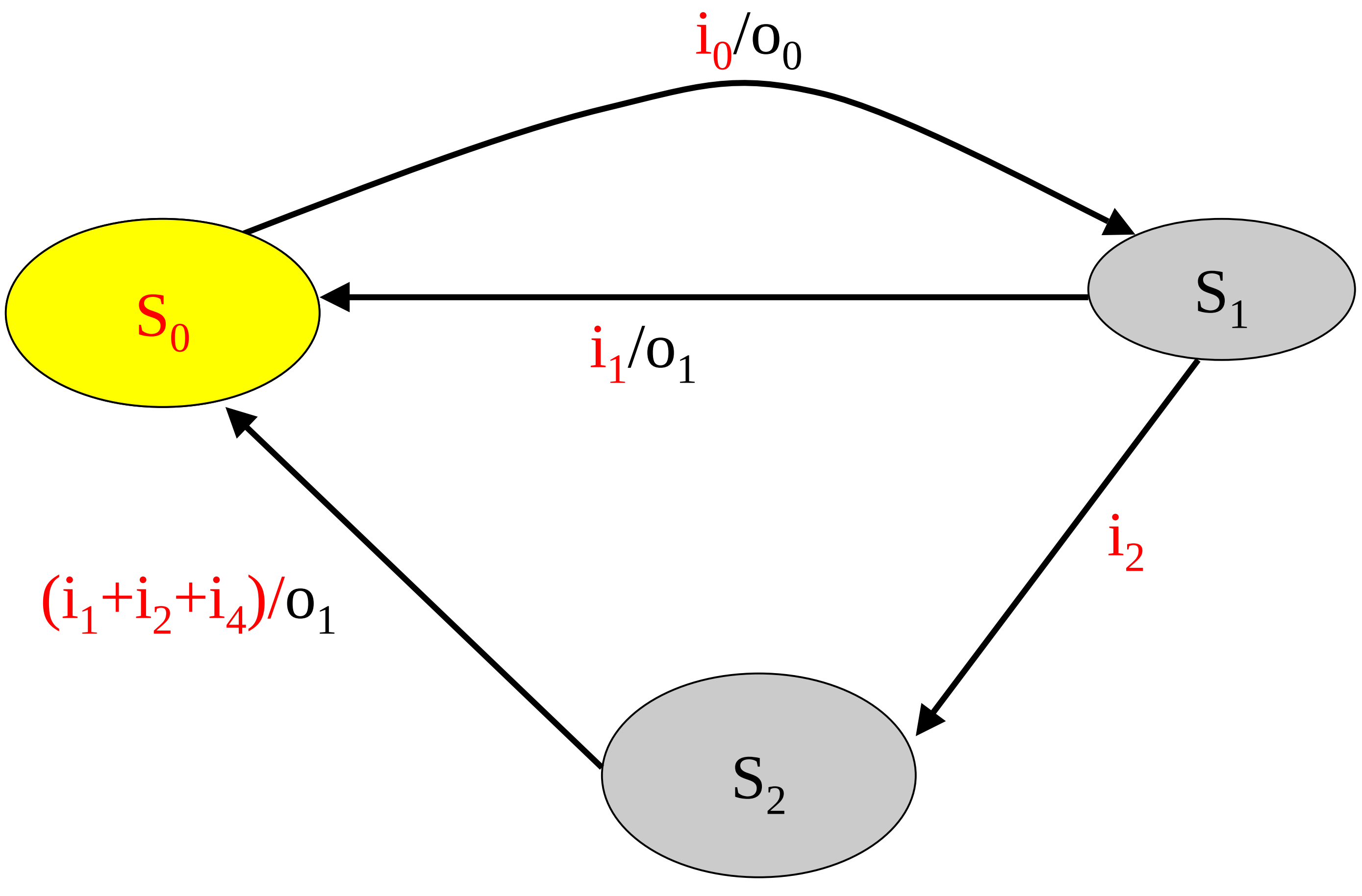
Finite State Machine

- **FSM = (S, I, O, s_0, δ, λ)**
  - **S = {$s_0, s_1, \ldots, s_k$}**
  - **I = {$i_1, i_2, \ldots, i_m$}**
  - **O = {$o_1, o_2, \ldots, o_n$}**
  - **δ: S x I → S (Transition function)**
  - **λ: S x I → O (Output function)**
- **Given an input sequence, an output sequence is produced which is depended on s_0, δ, and λ .**

Representation

- Given
  - **States**
    - **Alarm off (s_0), Alarm on (s_1), Wait (s_2)**
  - **Initial State**
    - **Alarm off (s_0)**
  - **Inputs**
    - **Turn on/off the key (i_0/i_1), fasten the seat belt (i_2), timer > 5 (i_3), time > 10 (i_4)**
  - **Outputs**
    - **Start/stop the timer (o_0/o_1)**

Transition Graph



Transition Function

- **Transition Function**

  s1 = s0 * i0   s0=s1*i1

  s2=s1 * i3   s0=s2*(i1+i2+i4)

- **Output Function**

  $O_0 = s_0 * i_0$ $O_1=s_1*i_1$

  $O_1=s_2*(i_1+i_2+i_4)$

Transition Table

State

	S_0	S_1	S_2
i_0	S_1	X	X
i_1	X	S_0	S_0
i_2	X	S_2	S_0
i_3	X	X	X
i_4	X	X	S_0

Output

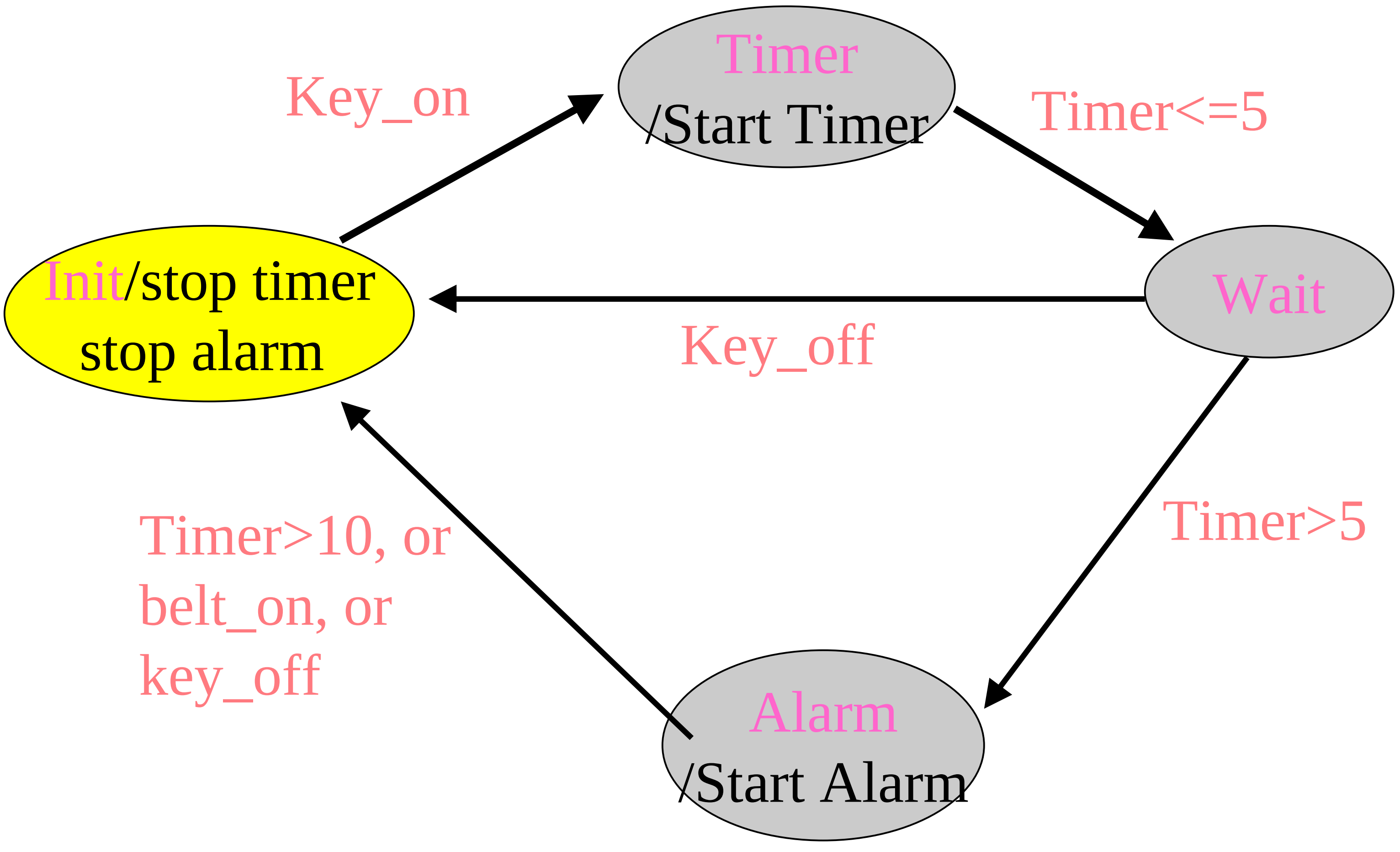
	S_0	S_1	S_2
i_0	o_0	-	-
i_1	-	o_1	o_1
i_2	-	-	o_1
i_3	-	-	-
i_4	-	-	o_1

X: don't care

Mealy Machine and Moore Machine

- Mealy Machine
  - The output is a function of both the current state and the input
- Moore Machine
  - The output is only a function of the current state

Transition Graph For Moore Machine



Mealy/Moore Machine

- An FSM can be realized either by Mealy or Moore machine
- Mealy machine may use less flip-flops and output signals are immediately after the transition
- Moore machine may use more flip-flops and output signals valid except during the transition

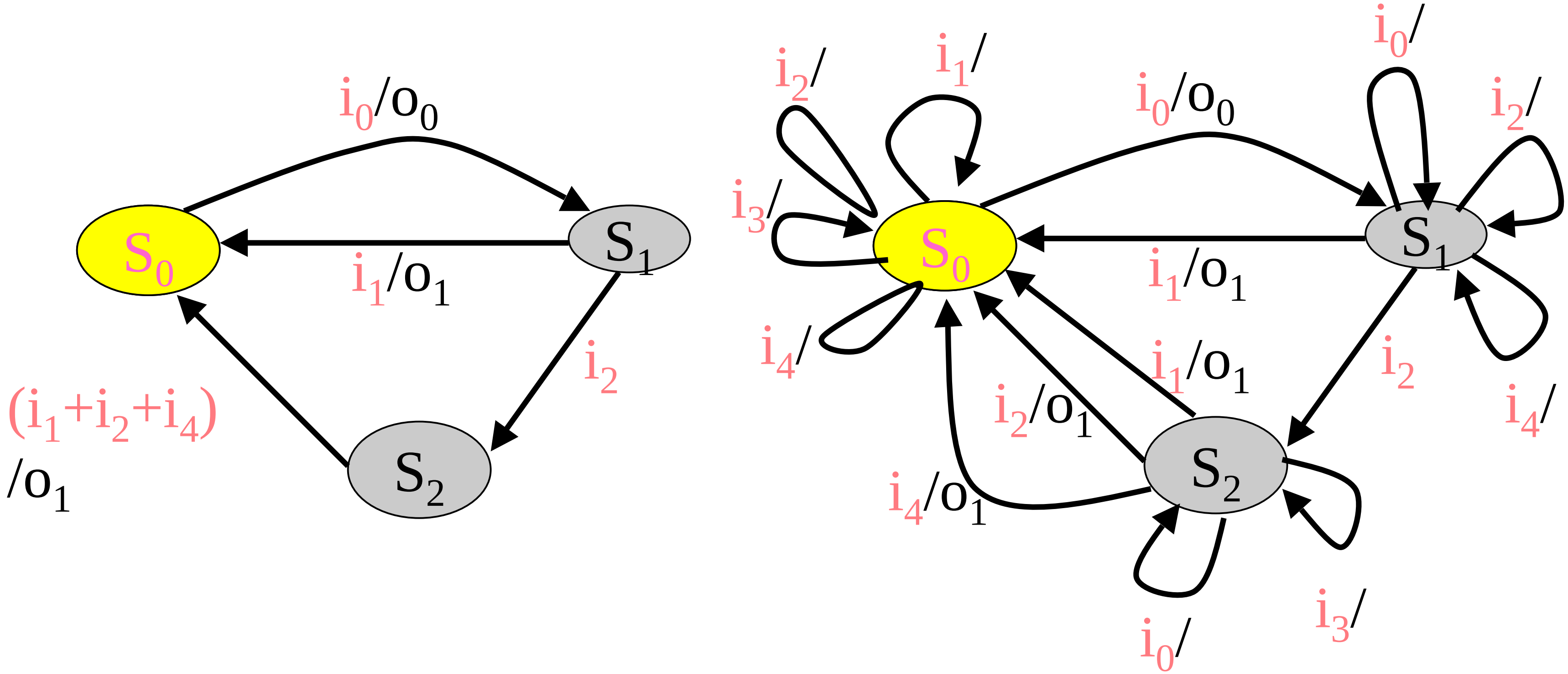
Nondeterministic FSM

- Deterministic FSM
  - Given a state and input, there is exactly one next state
- Nondeterministic FSM (NFSM)
  - Given a state and input, there maybe more than one next state, or a state can transform from one state to anther without any input, or for some given input there no next state at all
- For any NFSM, there is always one equivalent FSM

Nondeterministic FSM

- For unknown/unspecified behavior
- Less states, more compact
- Useful for
  - Optimization
  - Verification
- Can be refined
- For any NFSM, there is always one equivalent DFSM

NFSM and FSM

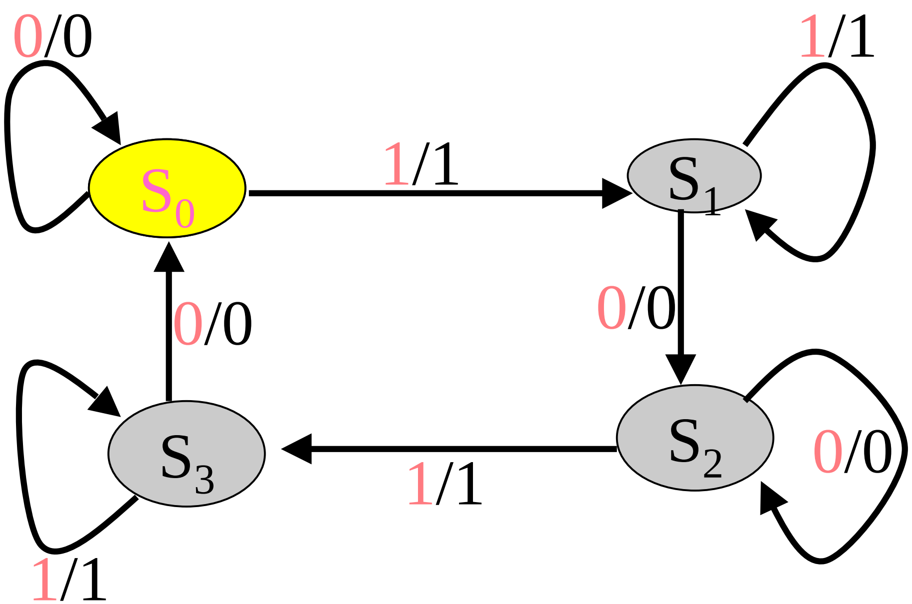


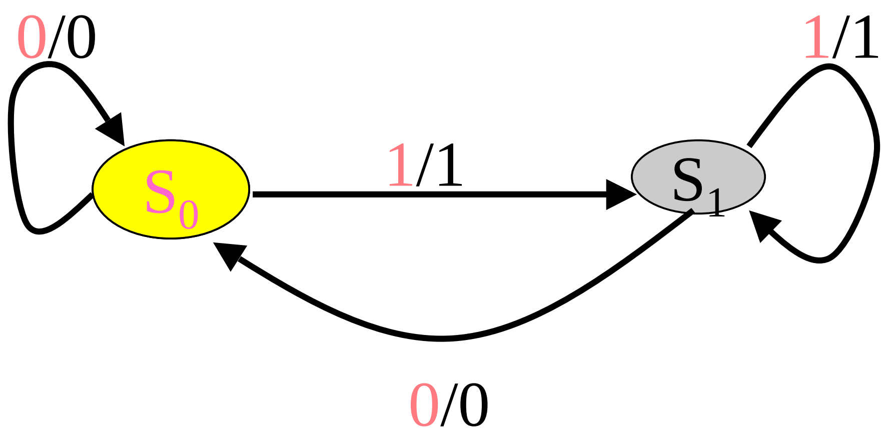
S0
S1
S2
i0/o0
i1/o1
i2
(i1+i2+i4)
/o1
i0/
i1/
i2/
i3/
i4/
i0/o0
i1/o1
i2
i0/
i2/
i4/
i1/o1
i2/o1
i4/o1
i0/
i3/

Equivalence

- Two FSMs are equivalent iff for any given input sequence, identical output sequences are produced

Equivalence





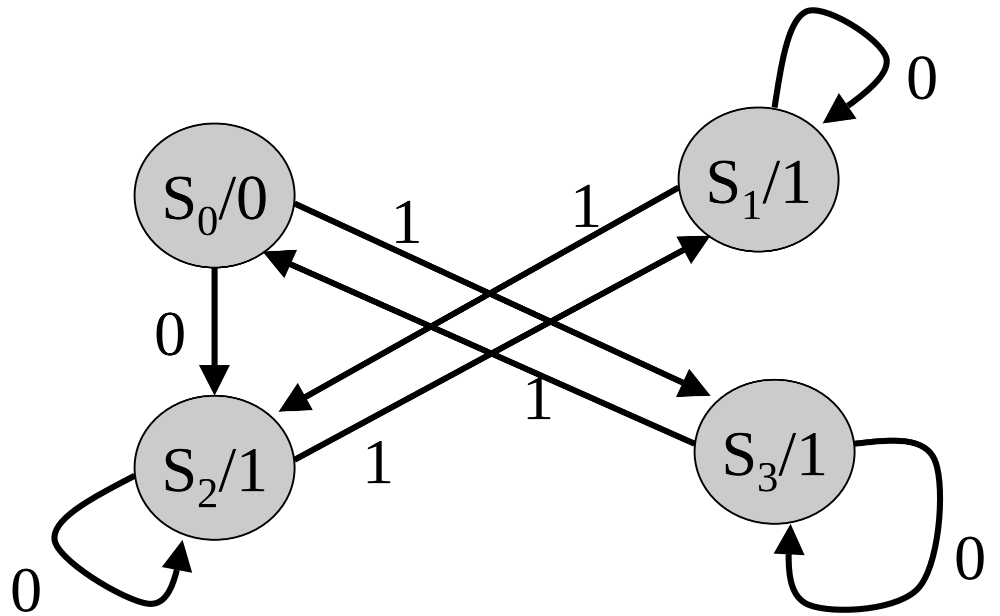
Minimization

- What
  - Given an FSM, find the equivalent FSM with a minimum number of states
    - Two states s1 and s2 in an FSM are equivalent iff each input sequence beginning from s1 yields an output sequence identical to that obtained by starting from s2
- How

Minimization(Moore Machine)

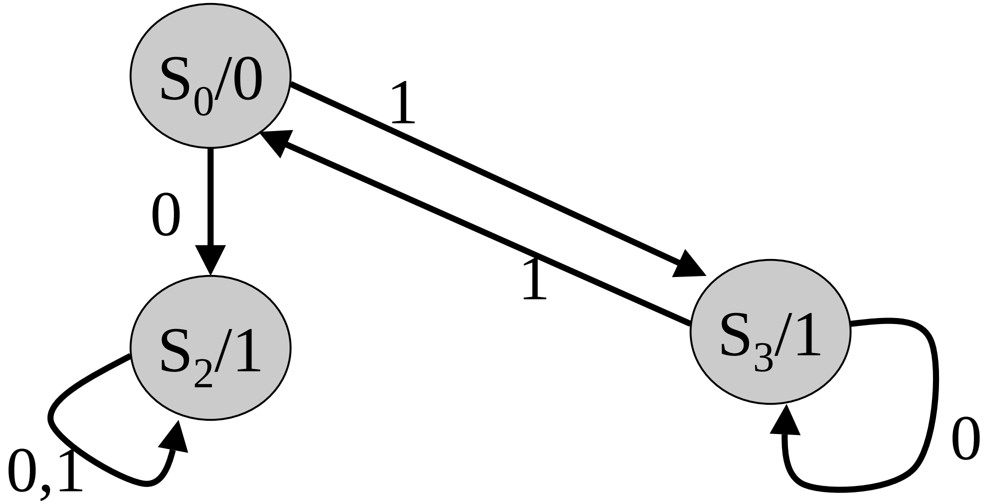
```
For each pair of the states (si,sj)
  If si and sj have different output
       Mark si and si as not equivalent
End for
Do
 for each unmarked pair
    for each input, si and sj are transferred to states which
        are not equivalent
        Mark si and sj as not equivalent
    end for
 end for
until no mark is possible
Unmarked pairs are equivalent
```

Minimization



(s0, s1) (s0,s2) (s0,s3) (s1,s2) (s1,s3) (s2,s3)

Minimization



(s0, s1) (s0,s2) (s0,s3) (s1,s2) (s1,s3) (s2,s3)