Equivalence

EECS 20

Lecture 12 (February 12, 2001)

Tom Henzinger

Revised by Matteo Zavatteri and Tiziano Villa, Fall 2020

Quiz

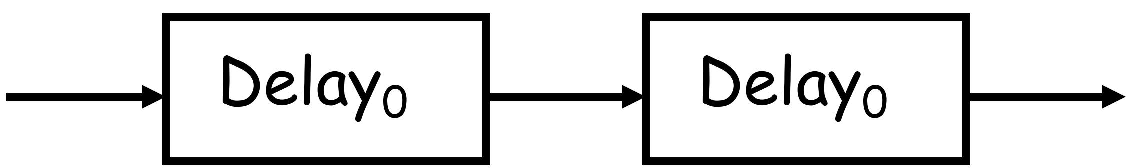
1. Draw the transition diagram of the system

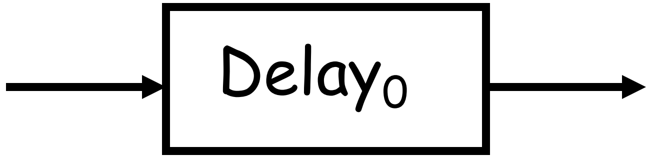
$$\mathrm{Delay}_0 : [\, \mathrm{Nats}_0 \to \mathrm{Bins} \,] \to [\, \mathrm{Nats}_0 \to \mathrm{Bins} \,]$$

$$\forall\, x \in [\, \mathrm{Nats}_0 \to \mathrm{Bins} \,],\ \forall\, y \in \mathrm{Nats}_0,$$

$$(\,\mathrm{Delay}_0\,(x))\,(y) = \begin{cases} 0 & \text{if } y = 0 \\ x\,(y-1) & \text{if } y > 0 \end{cases}$$

2. Draw the transition diagram of the system

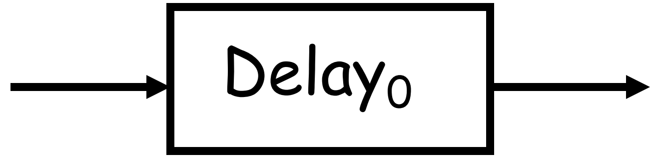




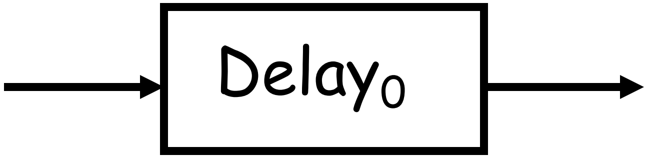
State before time t :

0 if input at time t-1 was 0, or if t = 0

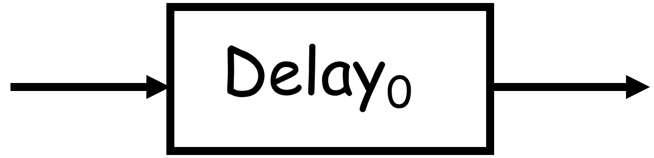
1 if input at time t-1 was 1

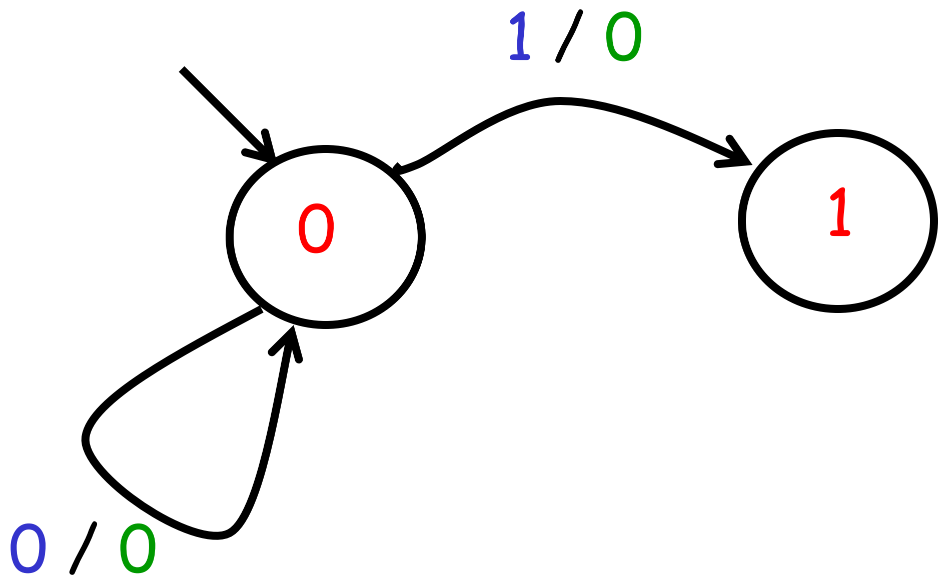

Delay_0

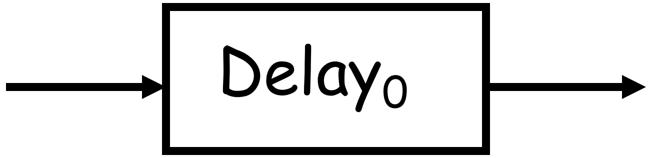

0
1

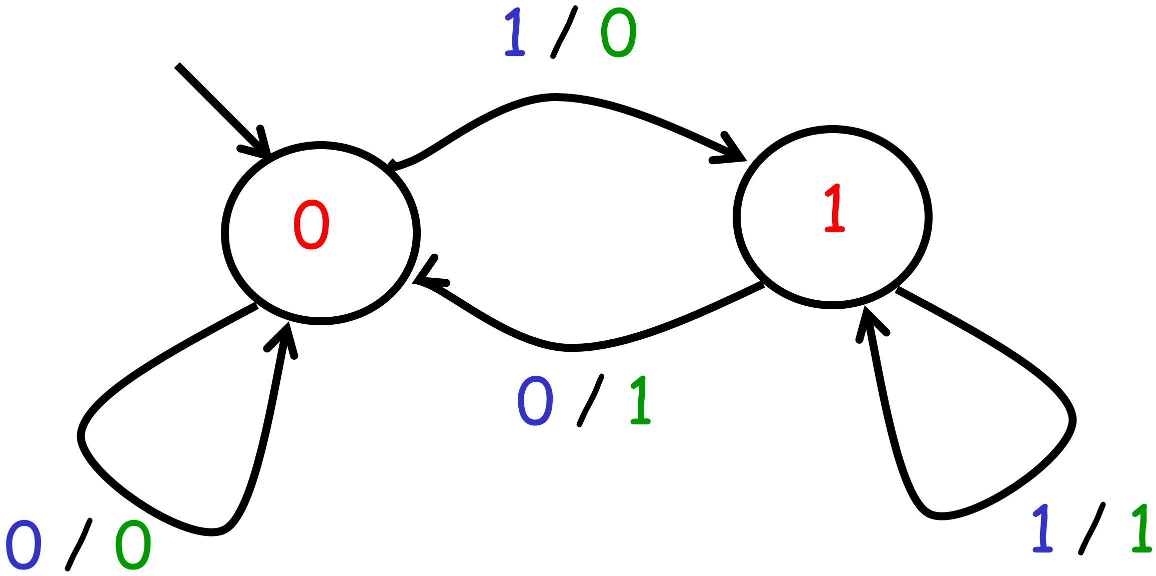

Delay$_0$

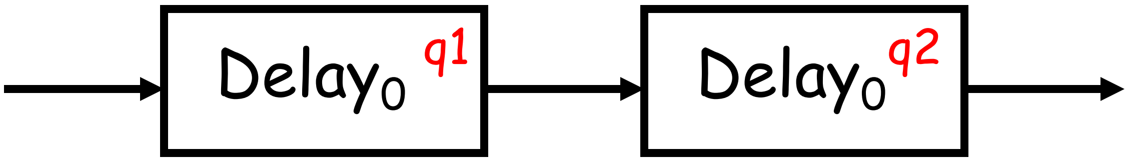

0
1


Delay0

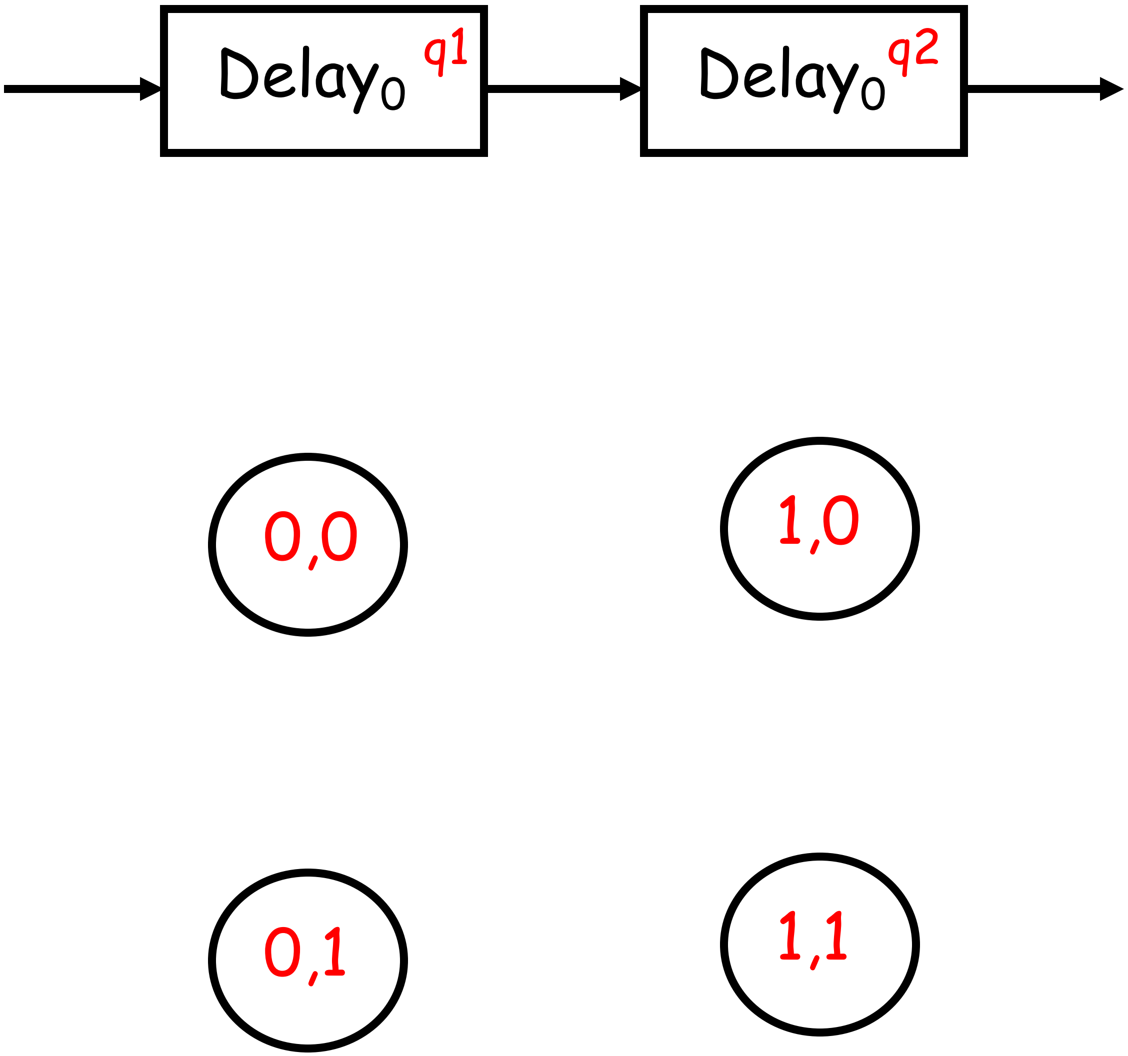

1 / 0
0
1
0 / 0

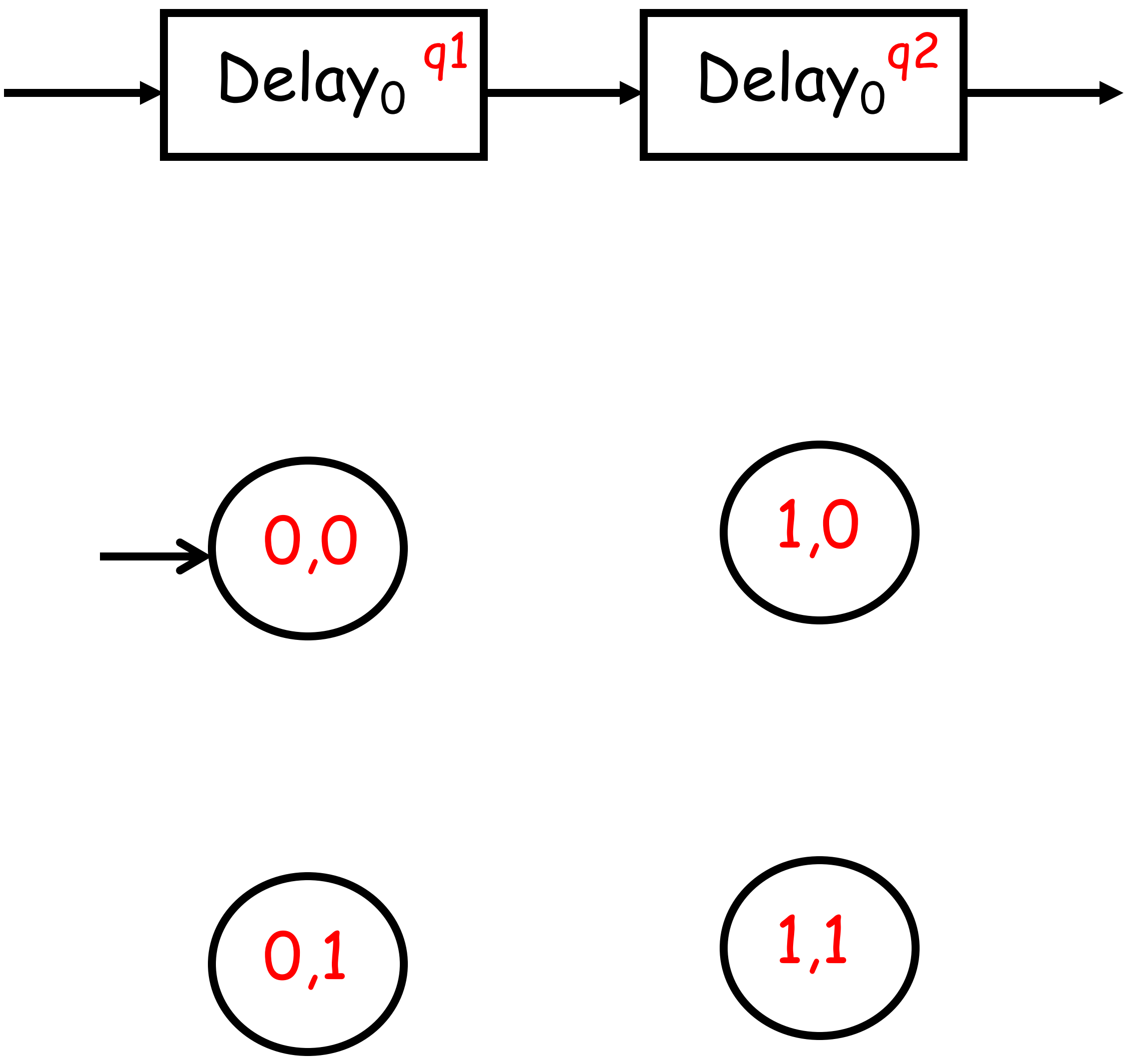

Delay0


1 / 0
0
1
0 / 1
0 / 0
1 / 1

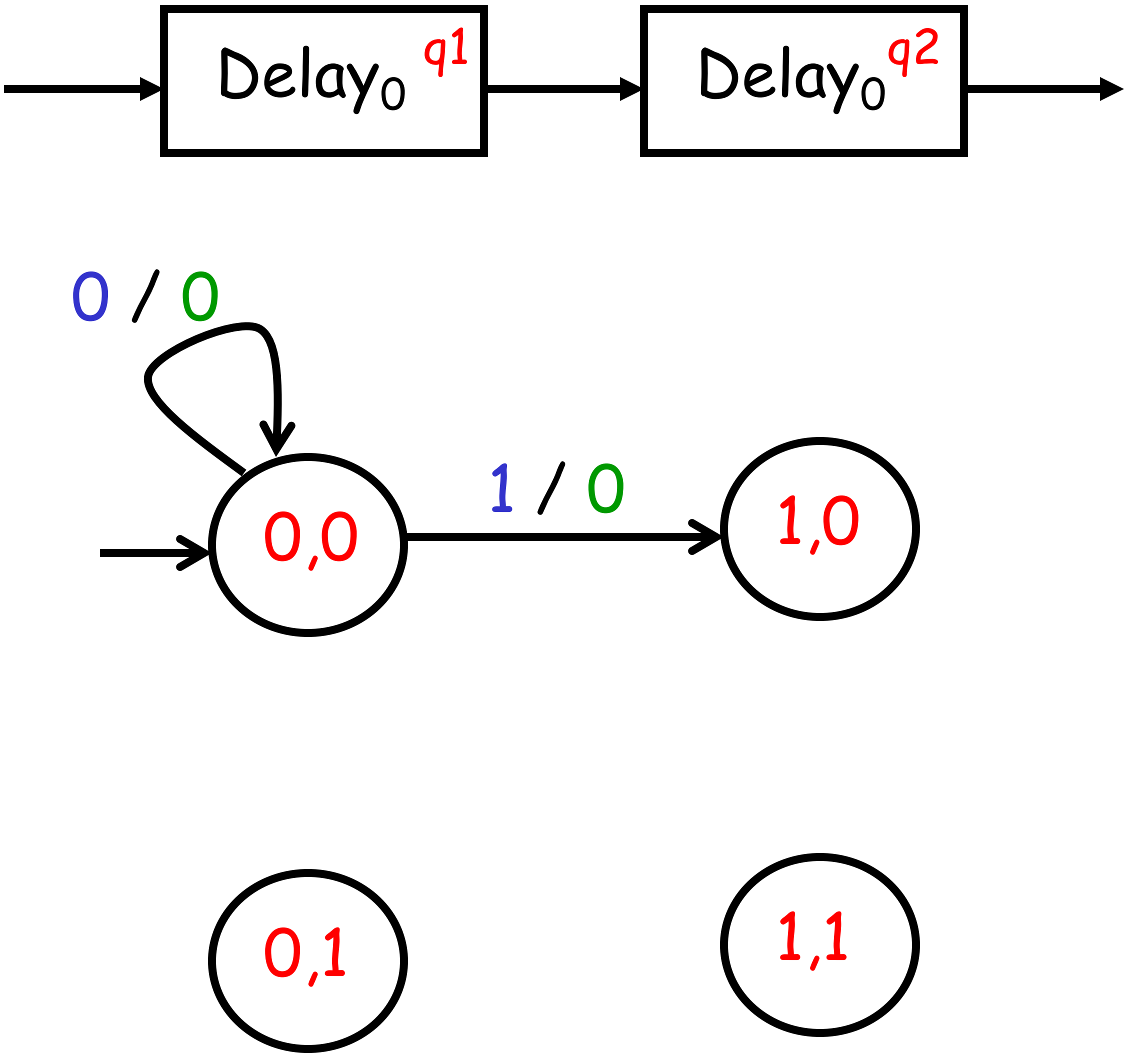


State: (q1, q2)

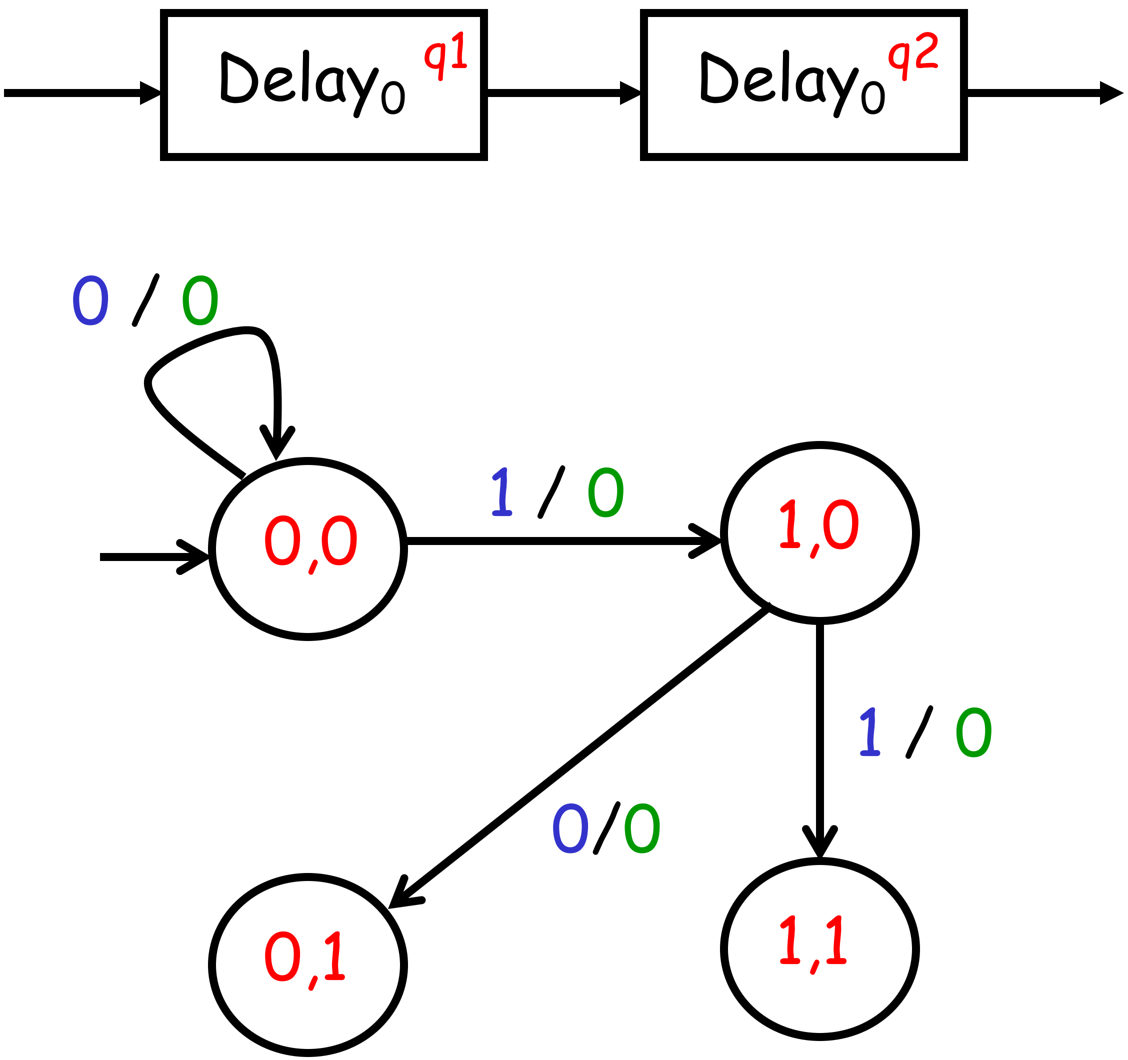

Delay0 q1
Delay0 q2
0,0
1,0
0,1
1,1



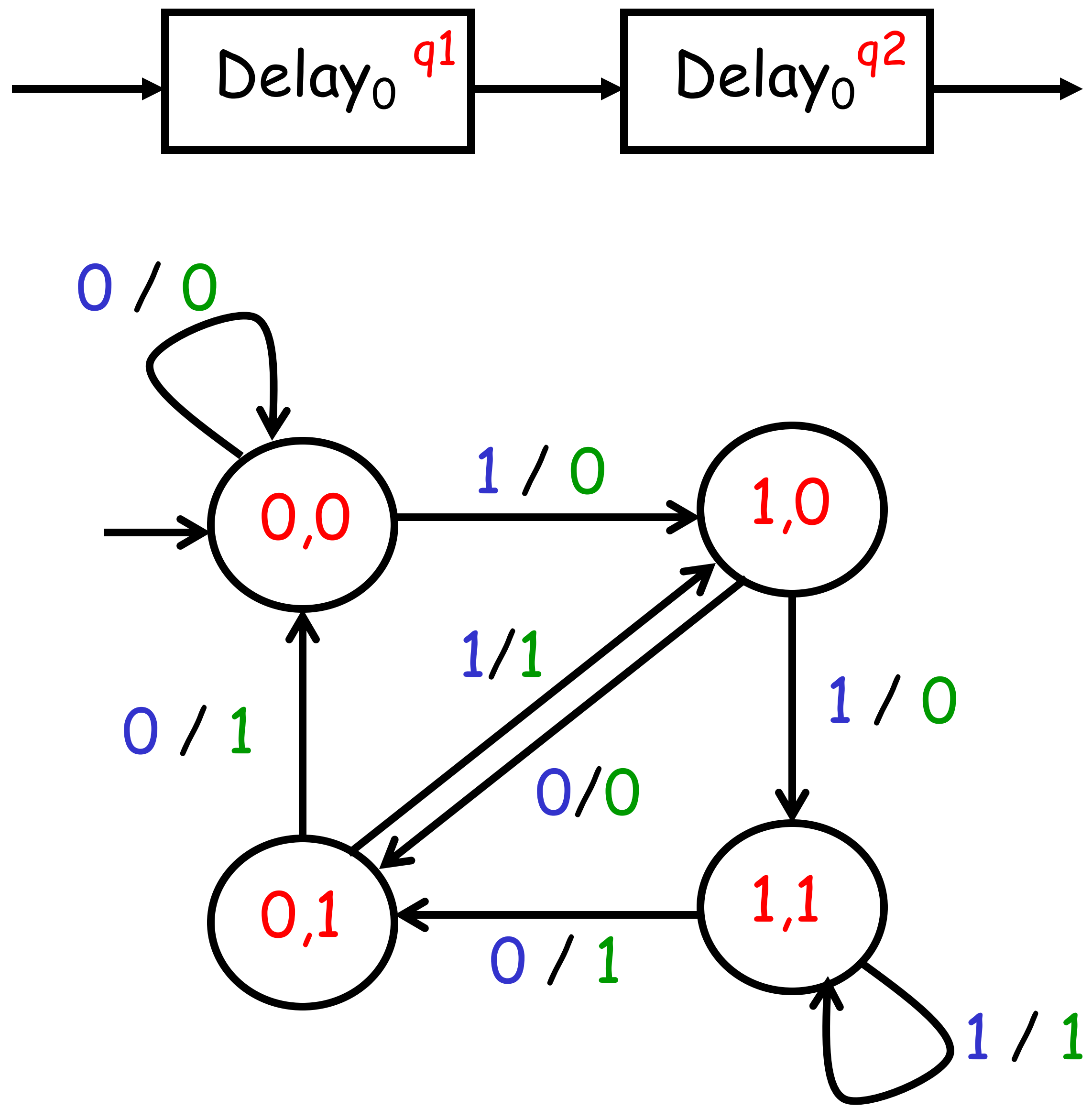
$Delay_0{}^{q1}$
$Delay_0{}^{q2}$
0,0
1,0
0,1
1,1



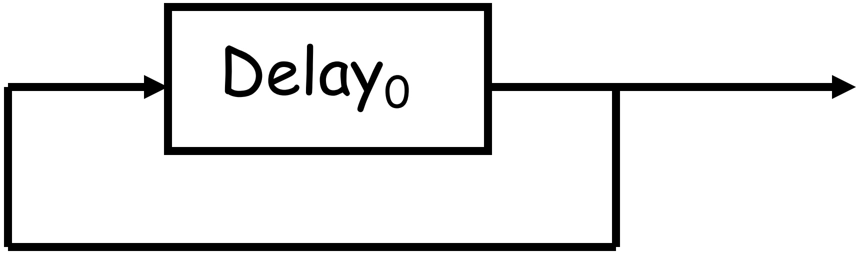
Delay$_0$ q1
Delay$_0$ q2
0 / 0
1 / 0
0,0
1,0
0,1
1,1

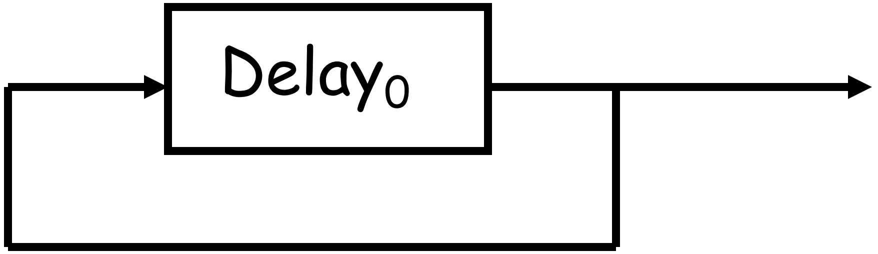


Delay0 q1
Delay0 q2
0 / 0
1 / 0
0,0
1,0
0/0
1 / 0
0,1
1,1

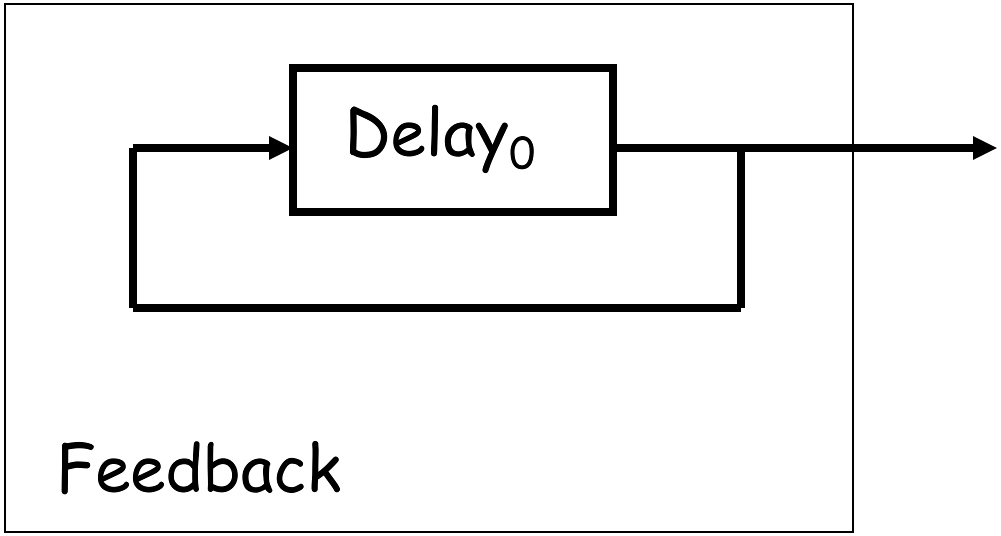


Delay0 q1
Delay0 q2
0 / 0
1 / 0
0,0
1,0
1/1
0 / 1
1 / 0
0/0
0,1
1,1
0 / 1
1 / 1


Delay$_0$

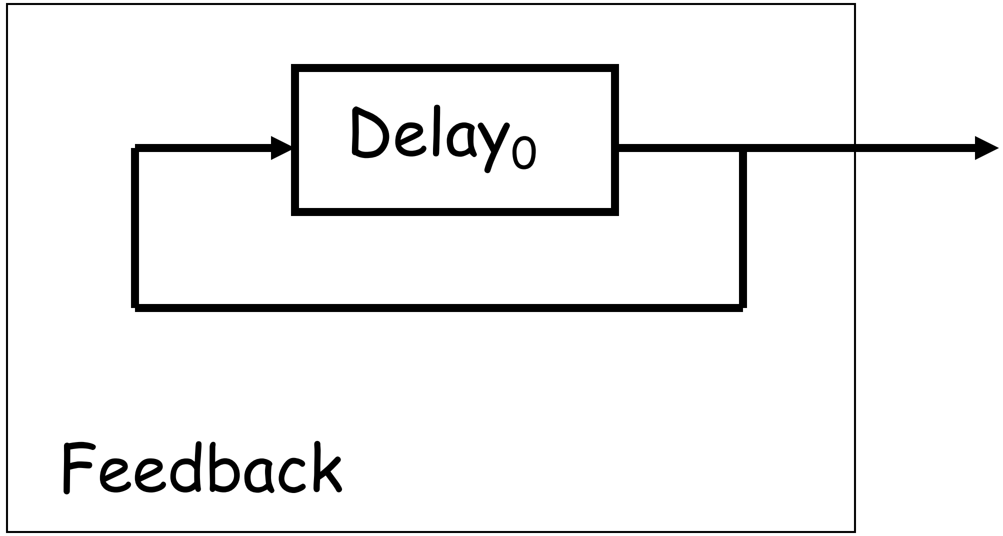


Well-formed !



Feedback : $[\ \text{Nats}_0 \to \text{Unit}\ ] \to [\ \text{Nats}_0 \to \text{Bins}\ ]$

where Unit = $\{ \cdot \}$.



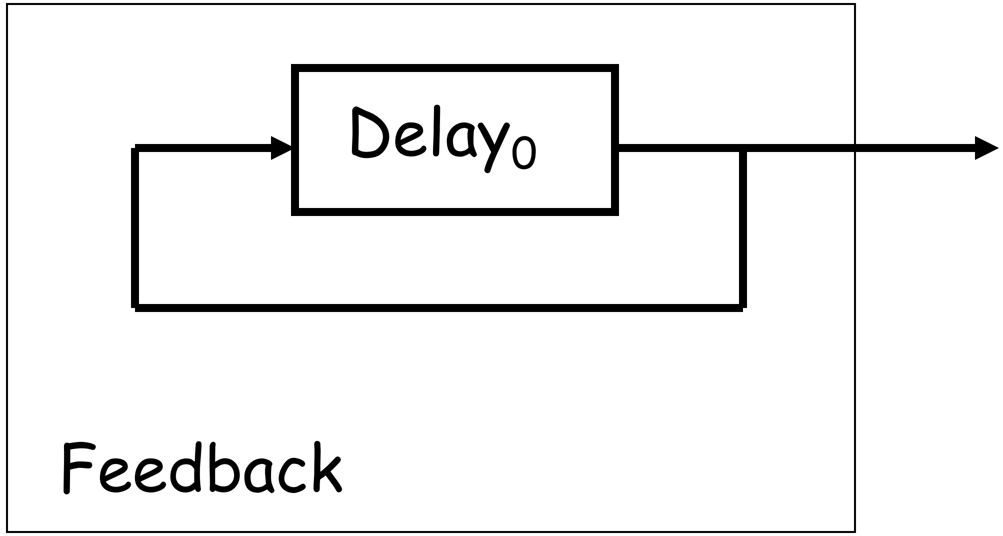
Inputs : Unit

Outputs: Bins

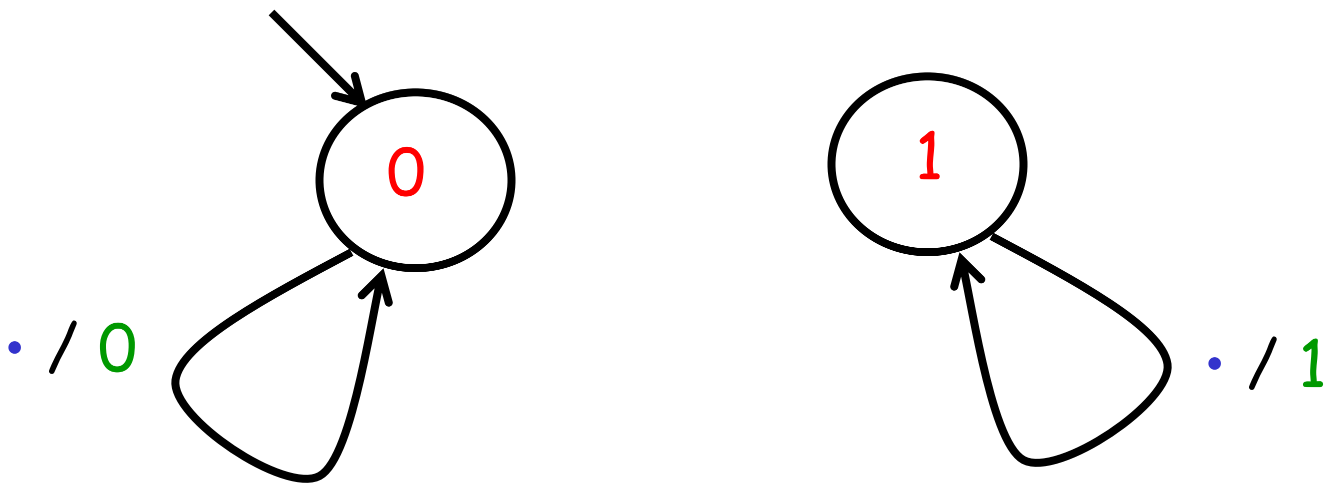
States: { 0, 1 }

initialState = 0

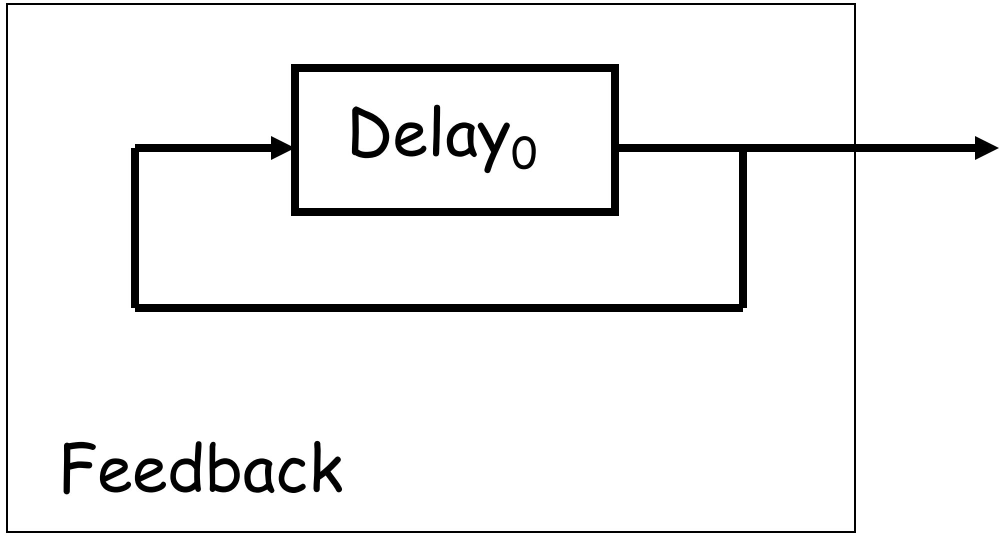
q	x	output (q,x)	nextState (q,x)
0	•	0	0
1	•	1	1



Delay0
Feedback

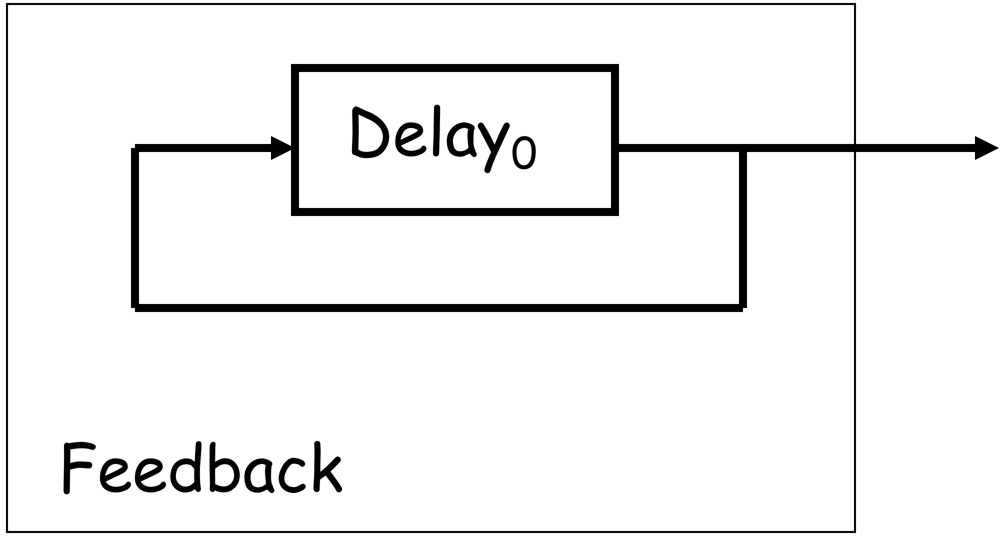


0
1
• / 0
• / 1

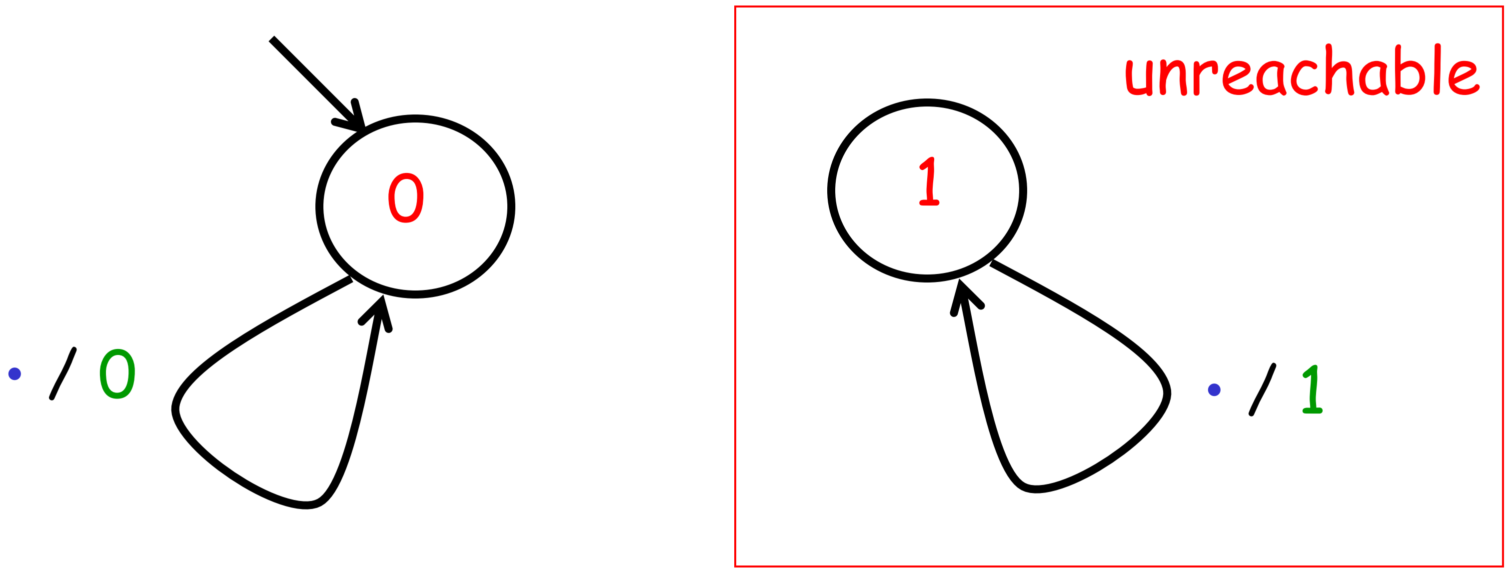


Only one run !

Time	0	1	2	3	4	5	6	7	
Input	•	•	•	•	•	•	•	•	
Output	0	0	0	0	0	0	0	0	
State	0	0	0	0	0	0	0	0	0



Delay0
Feedback



0
• / 0
unreachable
1
• / 1

A state q of a state machine M is unreachable

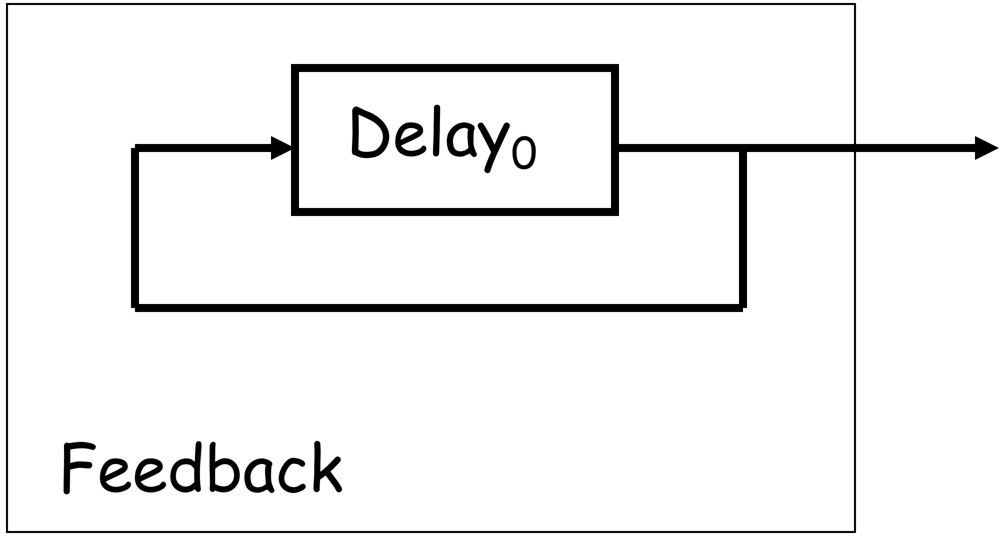
iff

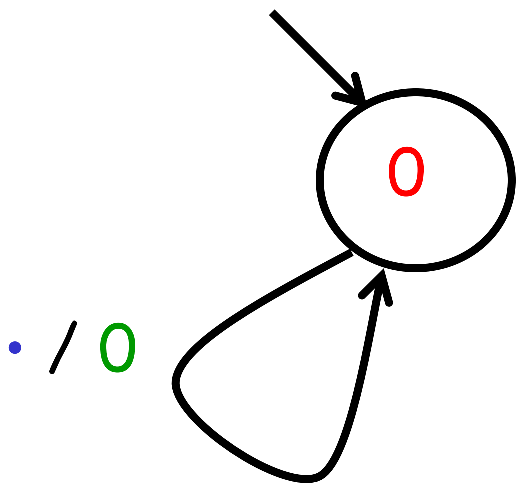
q occurs on no run of M ;

that is, iff

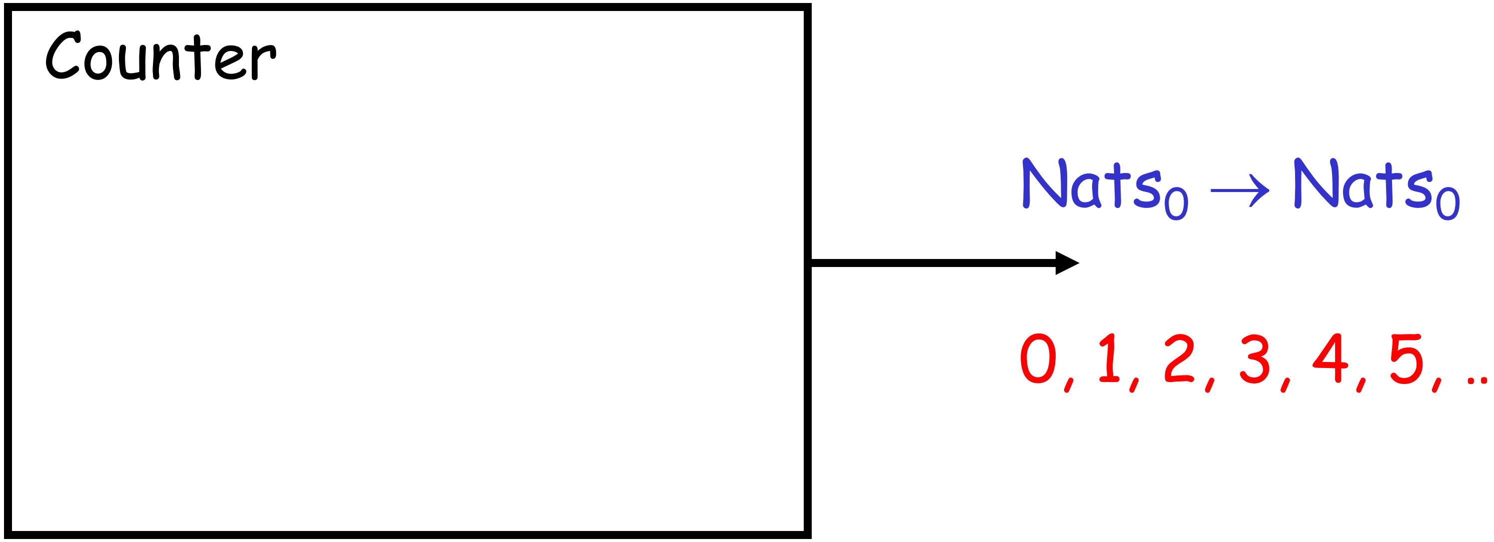
there is no path from the initial state of M to q .

Unreachable states can be removed without changing the system (input/output function) implemented by M .

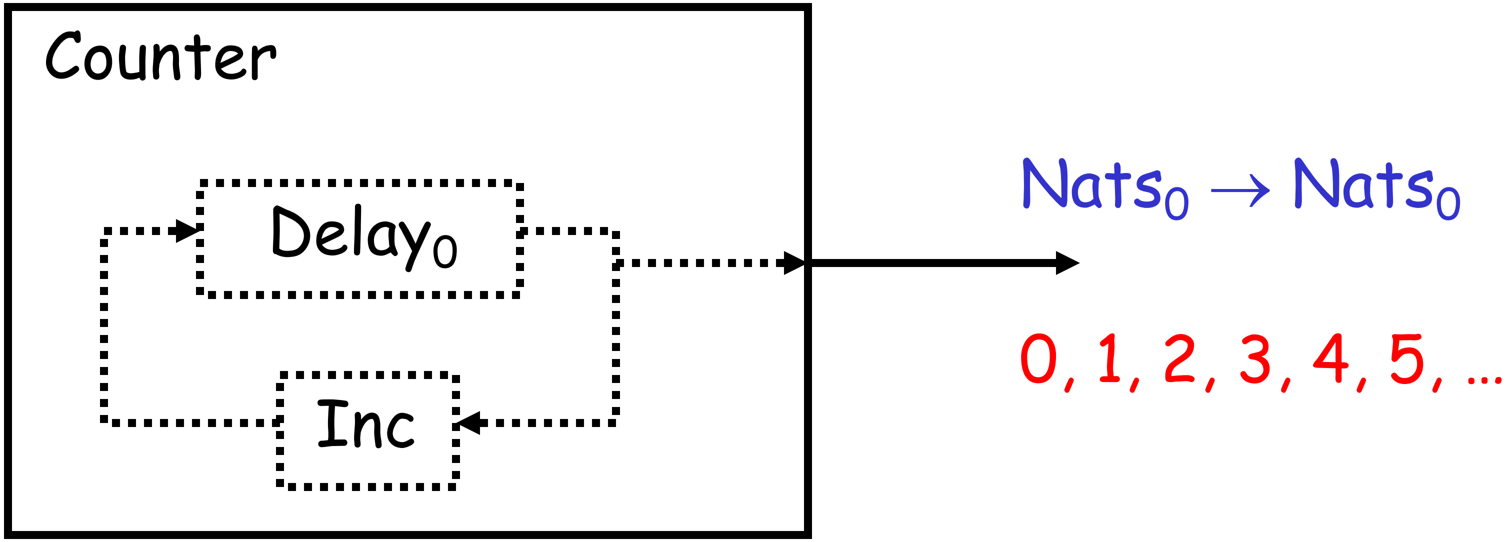

Delay0
Feedback


0
• / 0

A more interesting system without inputs



State-machine implementation of Counter

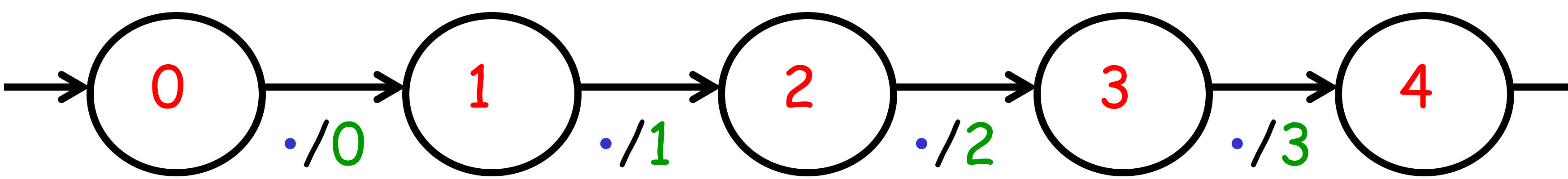


where $\forall\ n \in Nats_0,\ \ inc\,(n) = n + 1\,.$

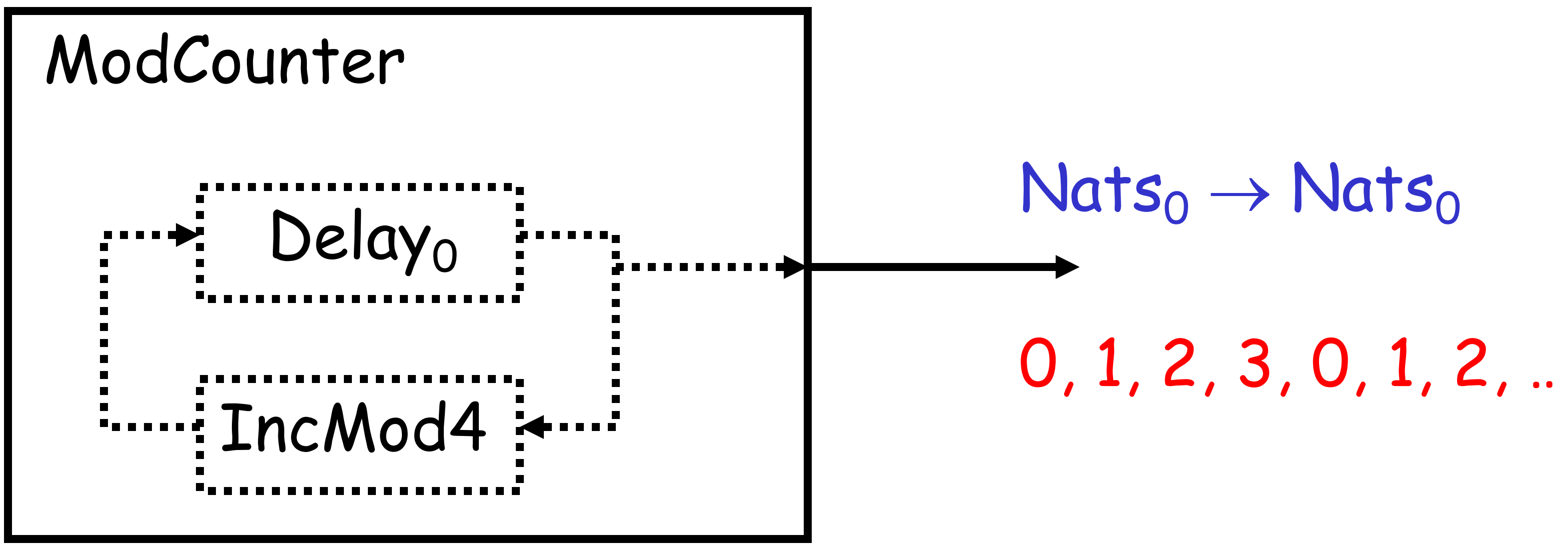
One run

Time	0	1	2	3	4	5	6	7	
Input	•	•	•	•	•	•	•	•	
Output	0	1	2	3	4	5	6	7	
State	0	1	2	3	4	5	6	7	8

Infinitely many states

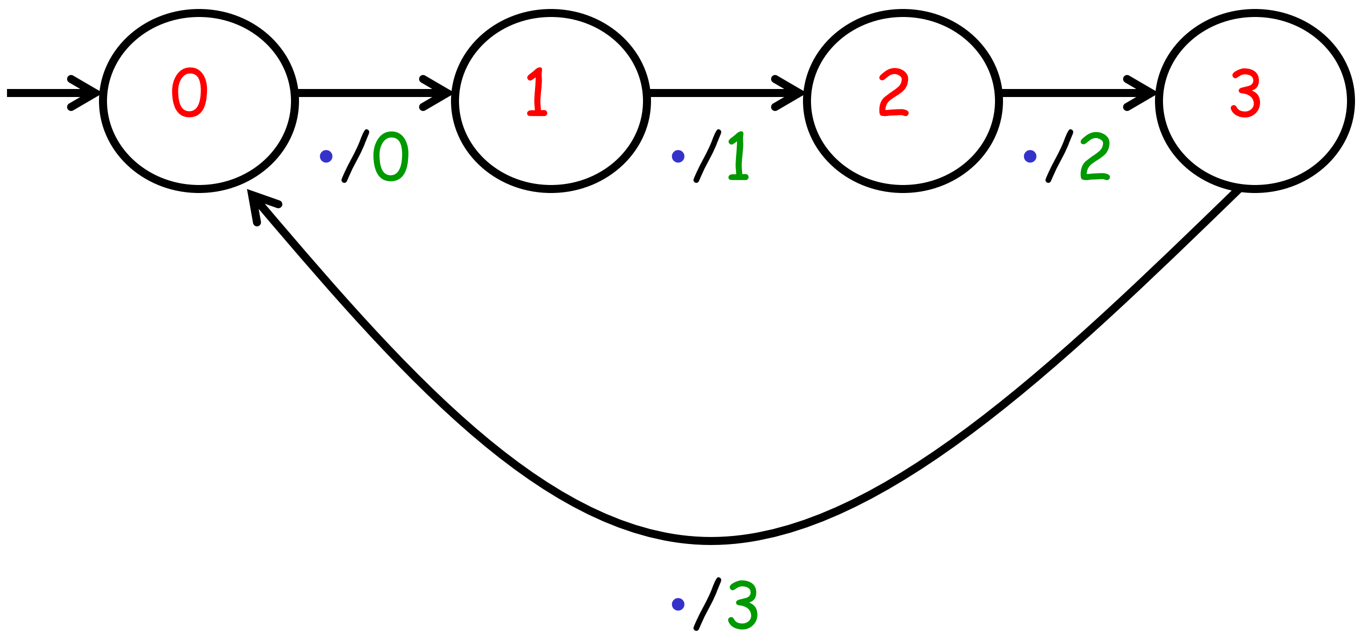


Yet another system without inputs



where $\forall\, n \in Nats_0$, incMod4 (n) = (n + 1) mod 4.

Transition diagram of ModCounter



A discrete-time reactive system can have many different state-machine implementations.

Two state machines M1 and M2 are equivalent

iff

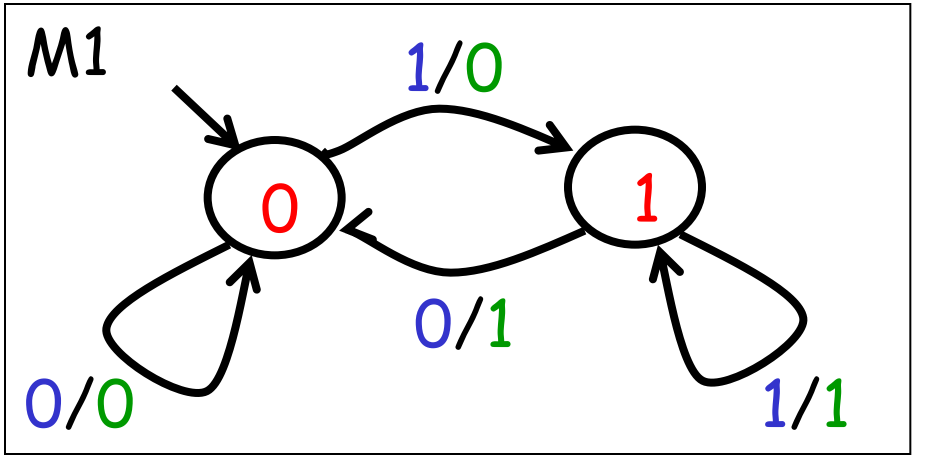
they implement the same system (input/output function);

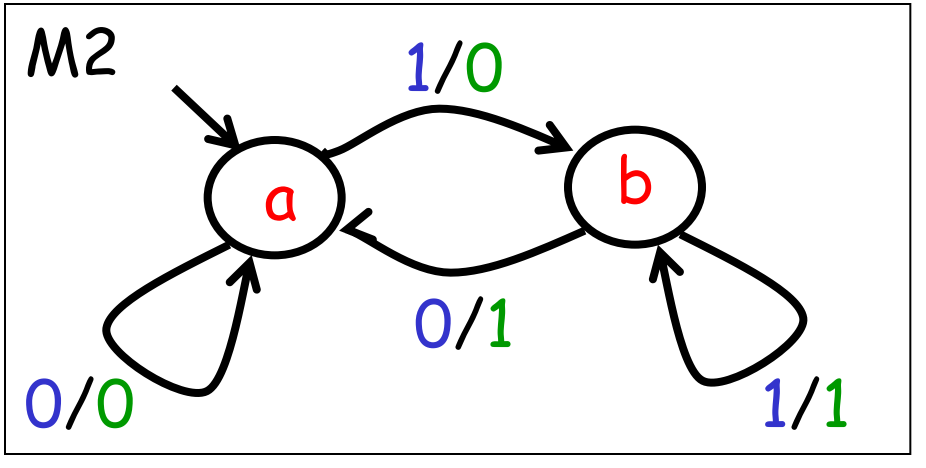
that is, iff

1. Inputs [M1] = Inputs [M2] ,
2. Outputs [M1] = Outputs [M2] , and
3. $\forall\ x \in [\ Nats_0 \to Inputs\ ],\ M1(x) = M2(x)$.

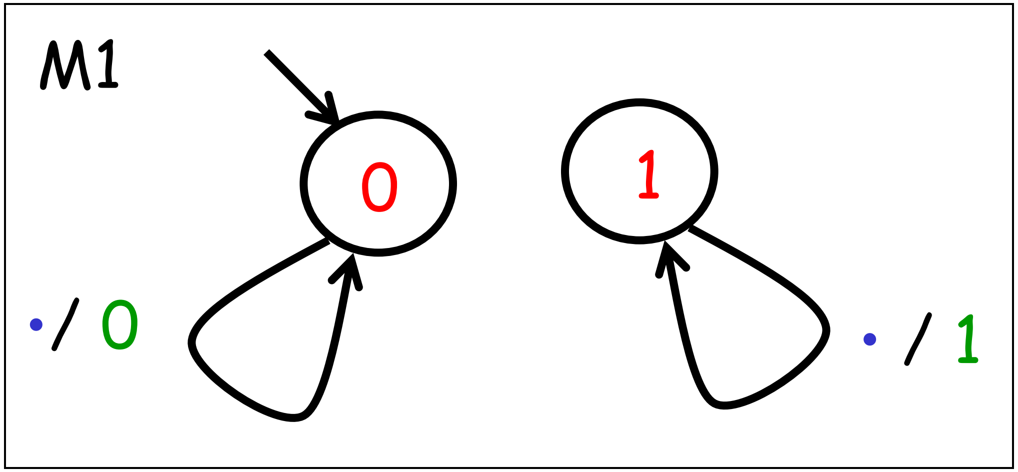
$\in [\ Nats_0 \to Outputs\ ]$

Equivalent State Machines

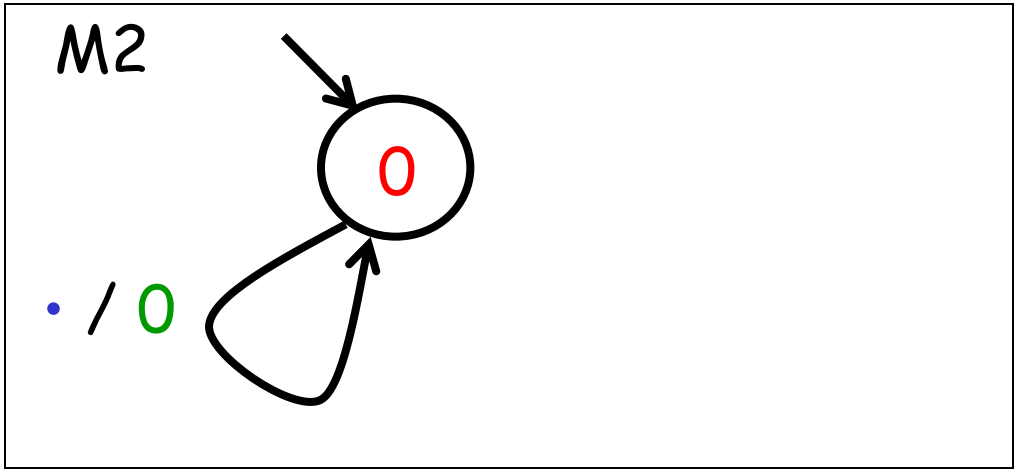




Equivalent State Machines

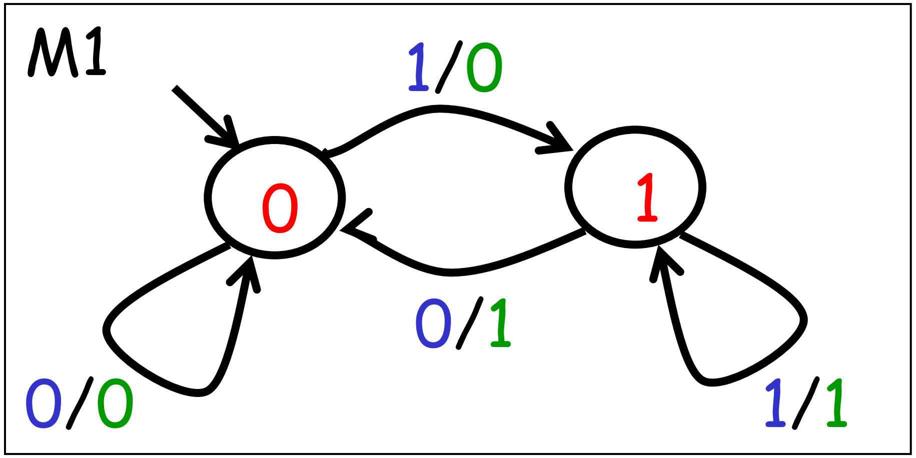


2 states

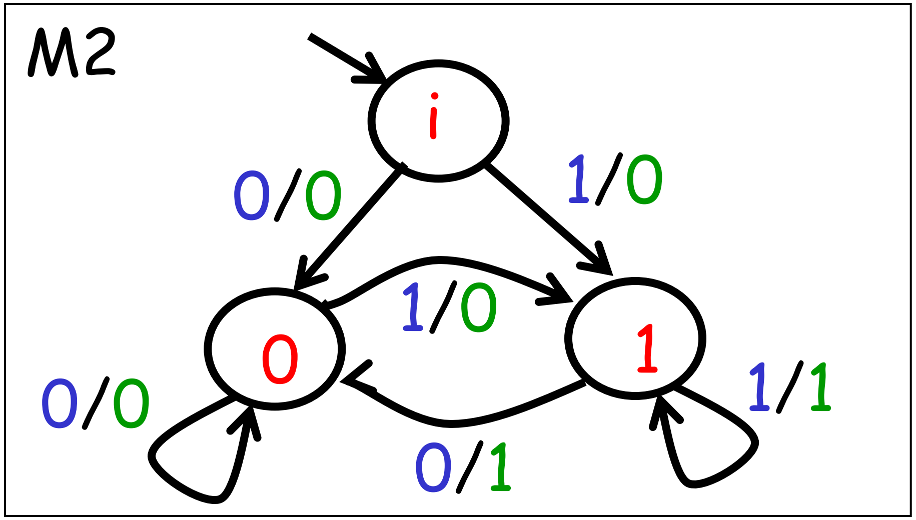


1 state

Equivalent State Machines



2 states



3 states

Theorem :

Two state machines M1 and M2 are equivalent

iff

there exists a bisimulation between M1 and M2 .

A binary relation B between States [M1] and States [M2] ;

that is, $B \subseteq$ States [M1] $\times$ States [M2] .

A binary relation $B \subseteq States[M1] \times States[M2]$ is a bisimulation

iff

1. $(initialState[M1], initialState[M2]) \in B$ and
2. $\forall p \in States[M1], \; \forall q \in States[M2],$

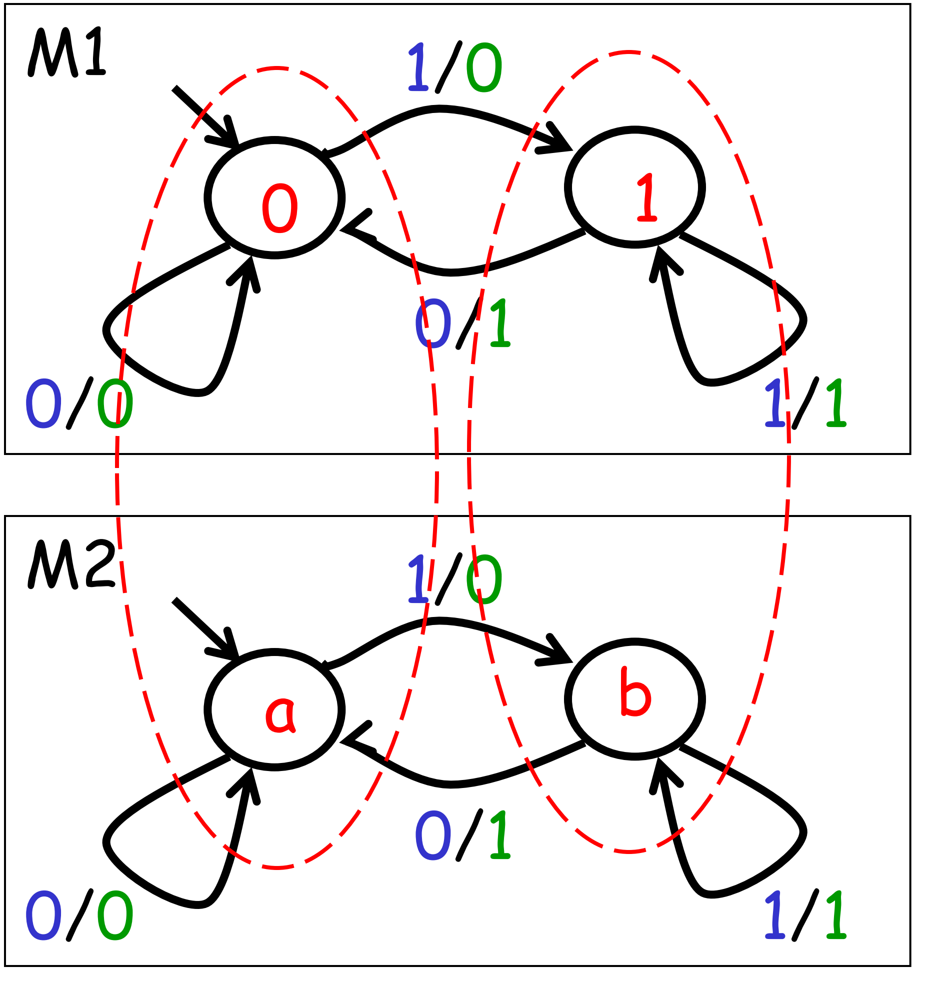
   if $(p, q) \in B$,

   then $\forall x \in Inputs[M1]$,

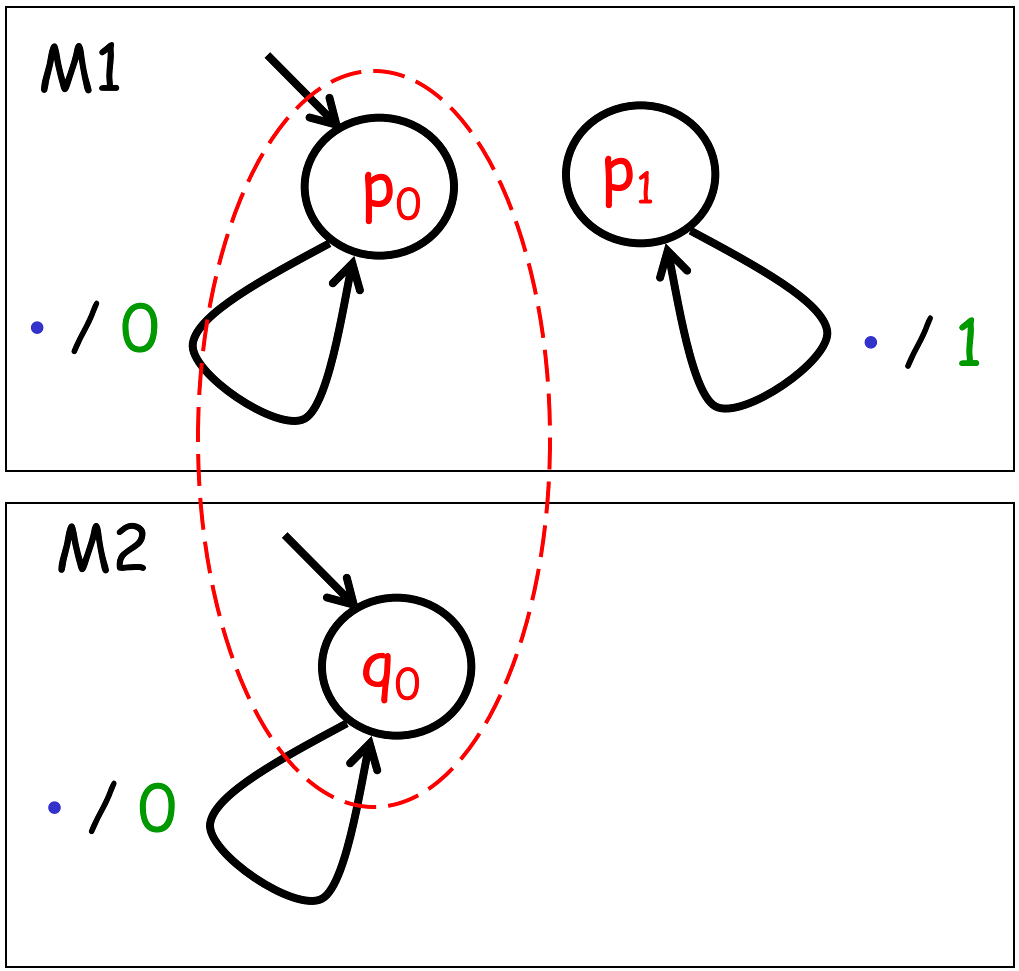
   $output[M1](p, x) = output[M2](q, x)$ and

   $(nextState[M1](p, x), nextState[M2](q, x)) \in B$.

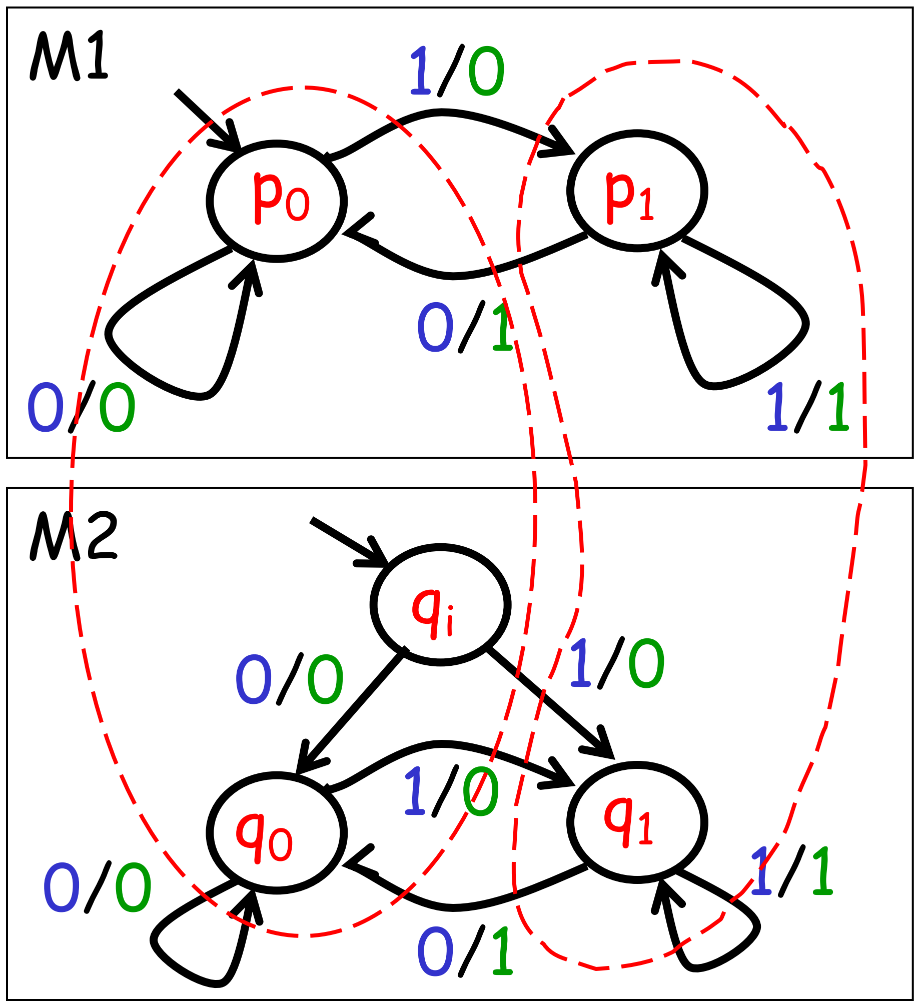
Bisimulation B = { (0, a), (1, b) }



Bisimulation $B = \{ (p_0, q_0) \}$



Bisimulation $B = \{ (p_0, q_i), (p_0, q_0), (p_1, q_1) \}$



Equivalence between state machines:

refers only to input and output signals.

Bisimulation between state machines:

refers to states.

Why is bisimulation useful ?

"Equivalence" says something about infinitely many possible input signals.

For finite state machines, "bisimulation" says something about finitely many possible relationships between states.

Bisimulation, therefore, is easier to check than equivalence.