Simulation Relations between Nondeterministic State Machines

Tom Henzinger

Deterministic System:

for every input signal, there is exactly one output signal.

Function:

$$\text{DetSys} : [\ \text{Time} \to \text{Inputs}\ ] \to [\ \text{Time} \to \text{Outputs}\ ]$$

Nondeterministic System:

for every input signal, there is one or more output signals.

Binary relation:

$$\text{NondetSys} \subseteq [\,\text{Time} \to \text{Inputs}\,] \times [\,\text{Time} \to \text{Outputs}\,]$$

such that $\forall\, x \in [\,\text{Time} \to \text{Inputs}\,],$
$\exists\, y \in [\,\text{Time} \to \text{Outputs}\,],\ (x,y) \in \text{NondetSys}$

Every pair $(x,y) \in \text{NondetSys}$ is called a behavior.

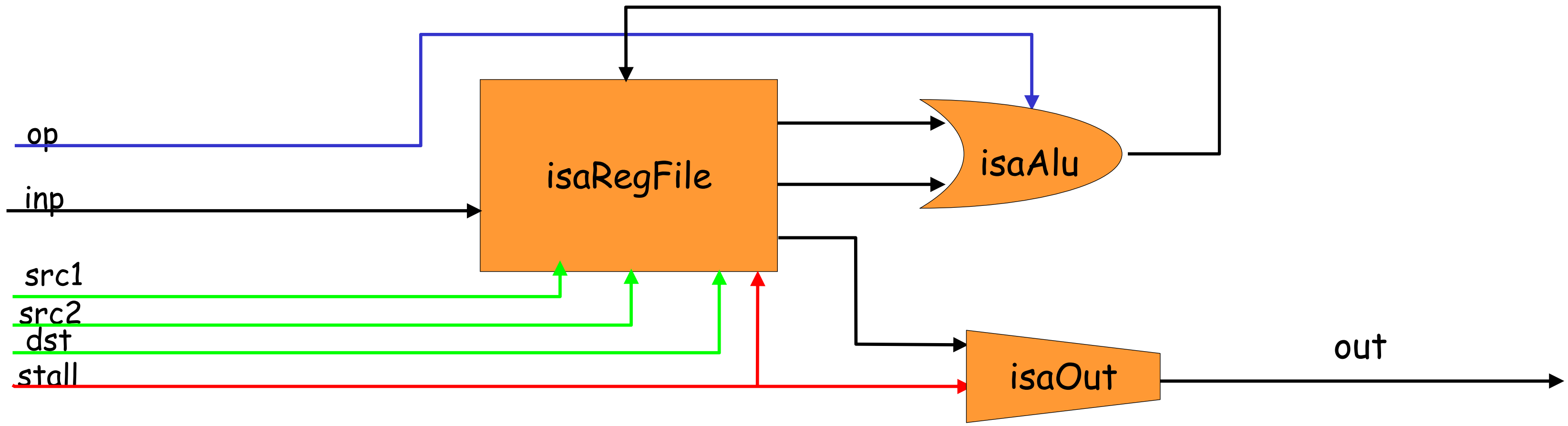
S1 is a more detailed description of S2;

S2 is an abstraction or property of S1.

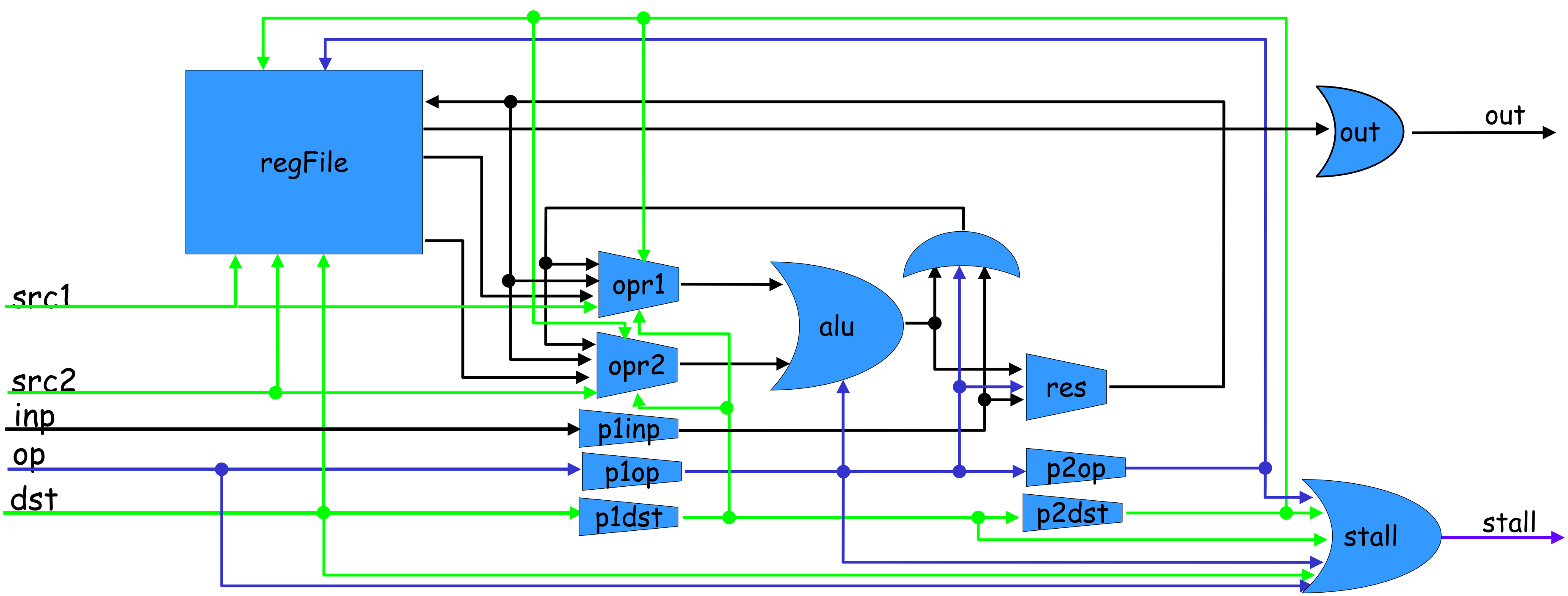
System S1 refines system S2

iff

1. Time [S1] = Time [S2] ,
2. Inputs [S1] = Inputs [S2] ,
3. Outputs [S1] = Outputs [S2] ,
4. Behaviors [S1] $\subseteq$ Behaviors [S2] .



Goal: establish that Pipeline refines ISA.



Abstractions and Properties: Nondeterministic State Machines

Inputs

Outputs

States

initialState $\in$ States

possibleUpdates :

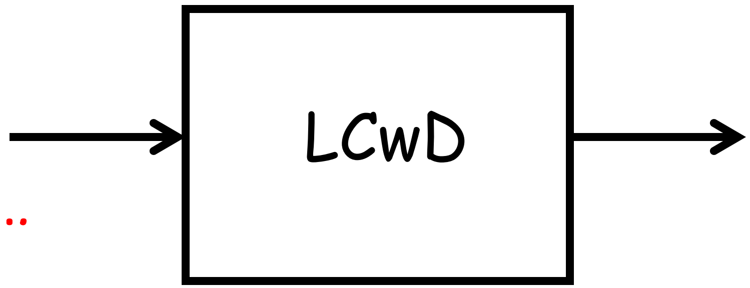
$$\text{States} \times \text{Inputs} \rightarrow P(\text{States} \times \text{Outputs}) \setminus \emptyset$$

receptiveness (i.e., machine must be prepared to accept every input)

Lossy Channel without Delay

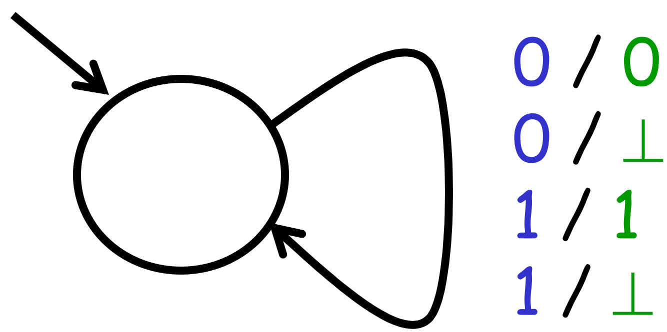
$Nats_0 \rightarrow Bins$

$0\ 1\ 1\ 0\ 0\ 0\ 1\ 1\ \ldots$

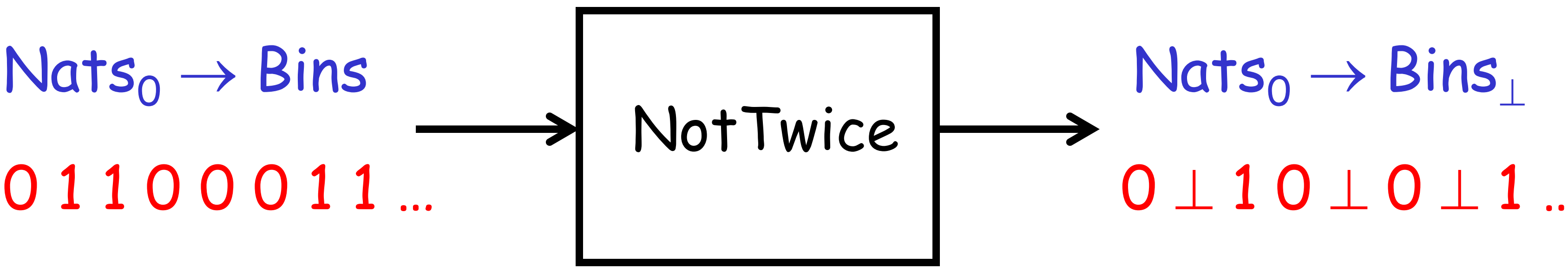


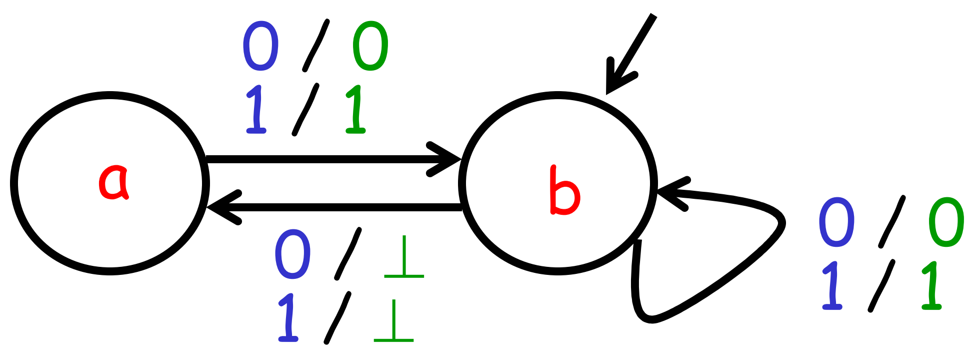
$Nats_0 \rightarrow Bins_\perp$

$0\ \perp\ 1\ 0\ \perp\ \perp\ 1\ 1\ \ldots$



Channel that never drops two in a row





condition on behaviors
(infinitely many)

Theorem :

The nondeterministic state machines M1 refines
the nondeterministic state machine M2

if

there exists a simulation of M1 by M2 .

relation between states
(finitely many)

In the following, assume

Inputs [M1] = Inputs [M2]

Outputs [M1] = Outputs [M2]

A binary relation $S \subseteq States[M1] \times States[M2]$ is a simulation of M1 by M2

iff

1. $(initialState[M1], initialState[M2]) \in S$ and

2. $\forall\, p \in States[M1],\ \forall\, q \in States[M2],$

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

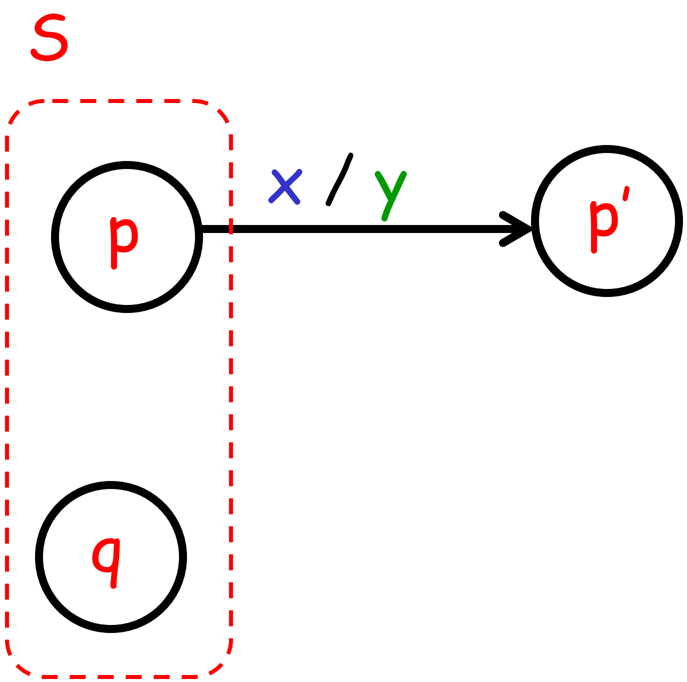
   then $\forall\, x \in Inputs,\ \forall\, y \in Outputs,\ \forall\, p' \in States[M1],$

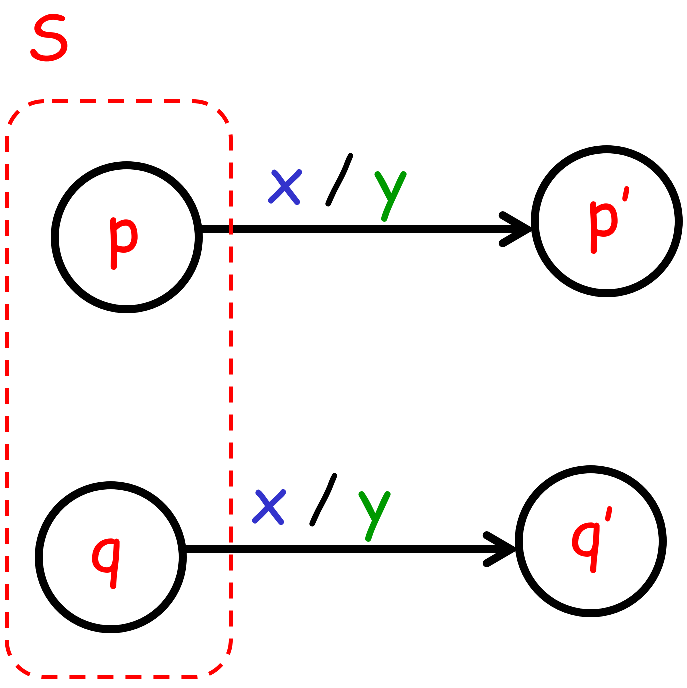
   if $(p', y) \in possibleUpdates[M1](p, x)$

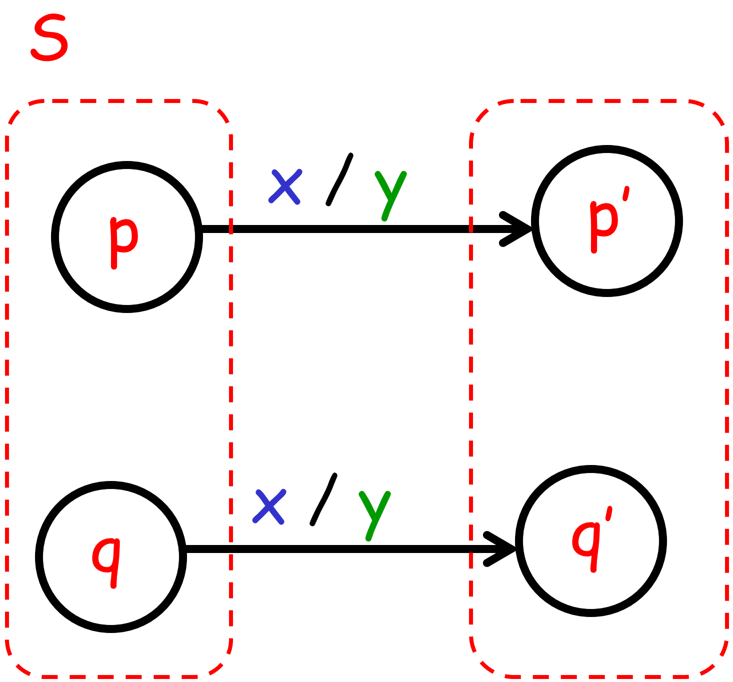
   then $\exists\, q' \in States[M2],$

   $(q', y) \in possibleUpdates[M2](q, x)$ and

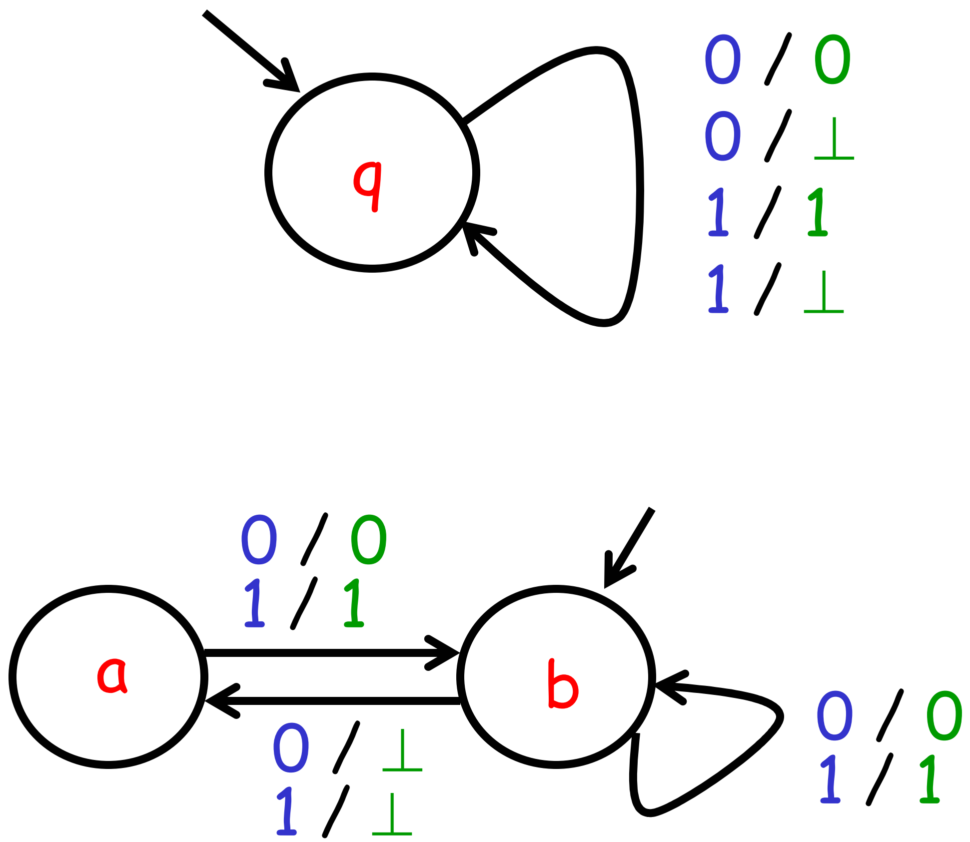
   $(p', q') \in S$.


S
p
x / y
p'
q

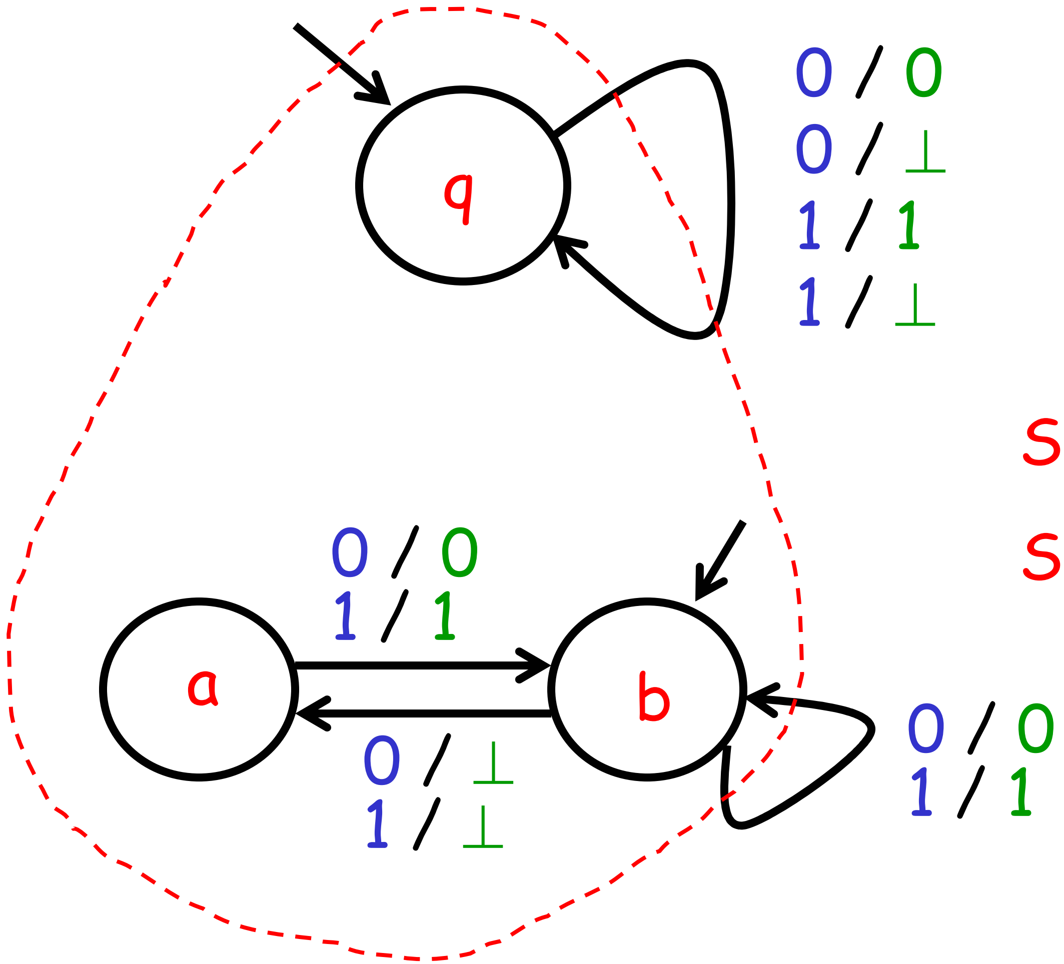

S
p
x / y
p'
q
x / y
q'


S
x / y
x / y
p
q
p'
q'

NotTwice refines Lossy Channel without Delay



NotTwice is simulated by Lossy Channel without Delay



Simulation

S = { (a, q), (b, q) }

Channel that drops every third zero

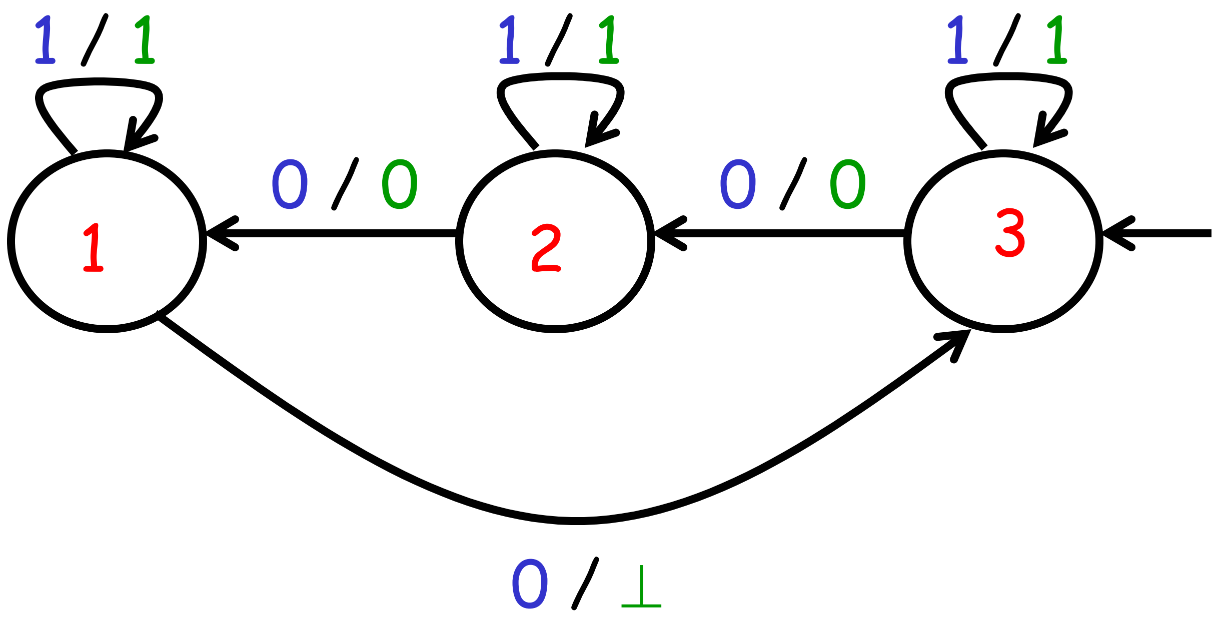
$\mathrm{Nats}_0 \to \mathrm{Bins}$ $\longrightarrow$ **ThirdZero** $\longrightarrow$ $\mathrm{Nats}_0 \to \mathrm{Bins}_\perp$

0 1 1 0 0 0 1 1 … $\longrightarrow$ 0 1 1 0 $\perp$ 0 1 1 …

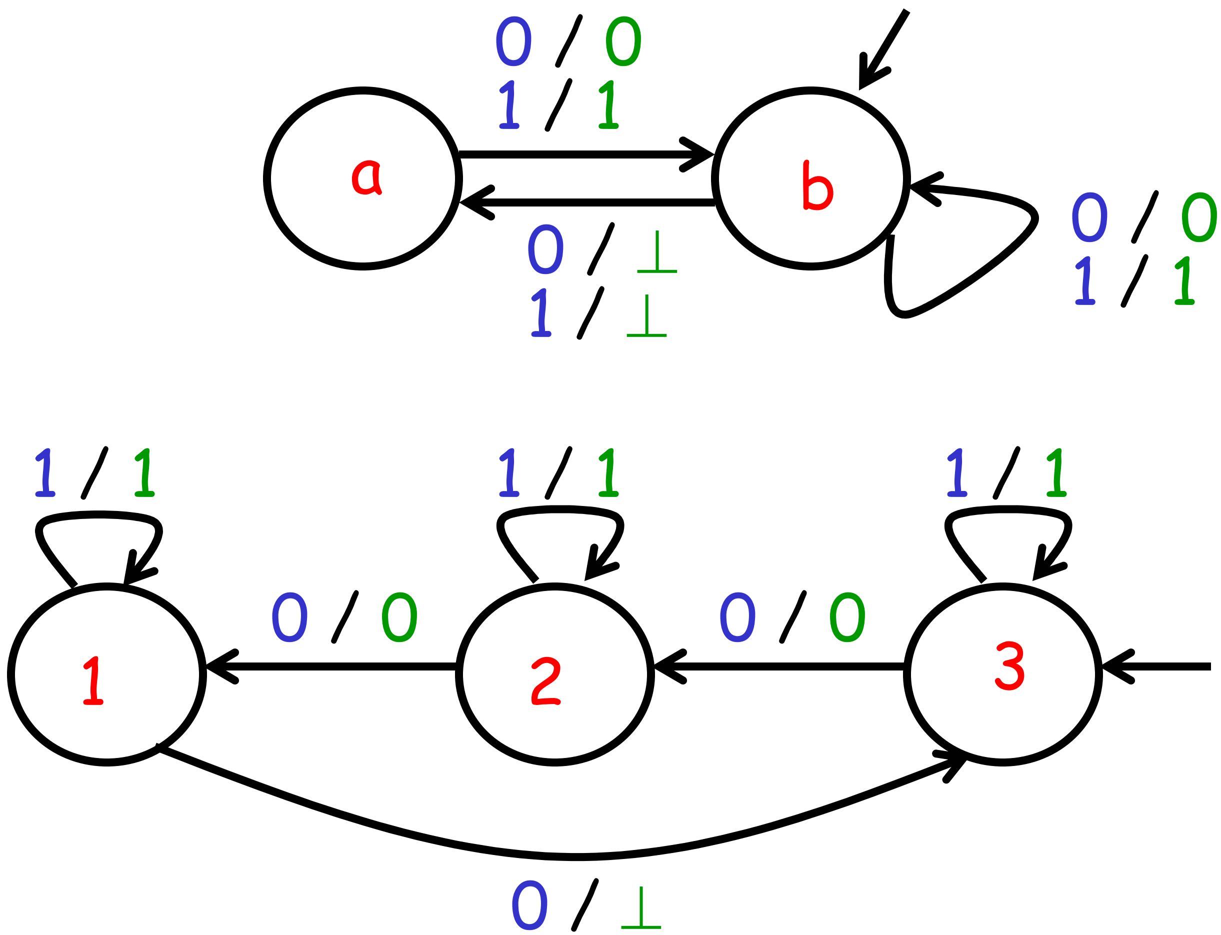
State between time t-1 and time t:

3	third zero from now will be dropped
2	second zero from now will be dropped
1	next zero will be dropped

Channel that drops every third zero



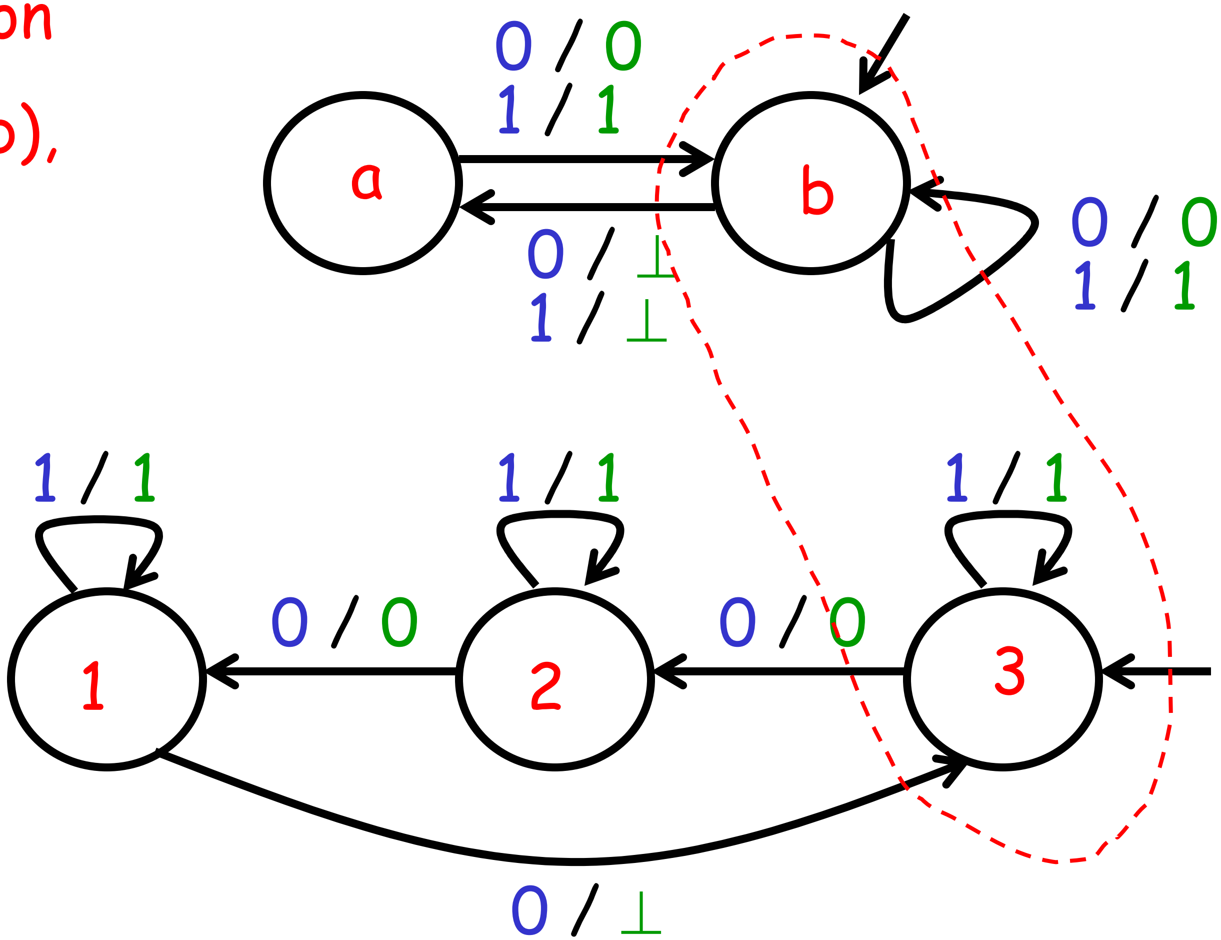
ThirdZero refines NotTwice



ThirdZero is simulated by NotTwice

Simulation

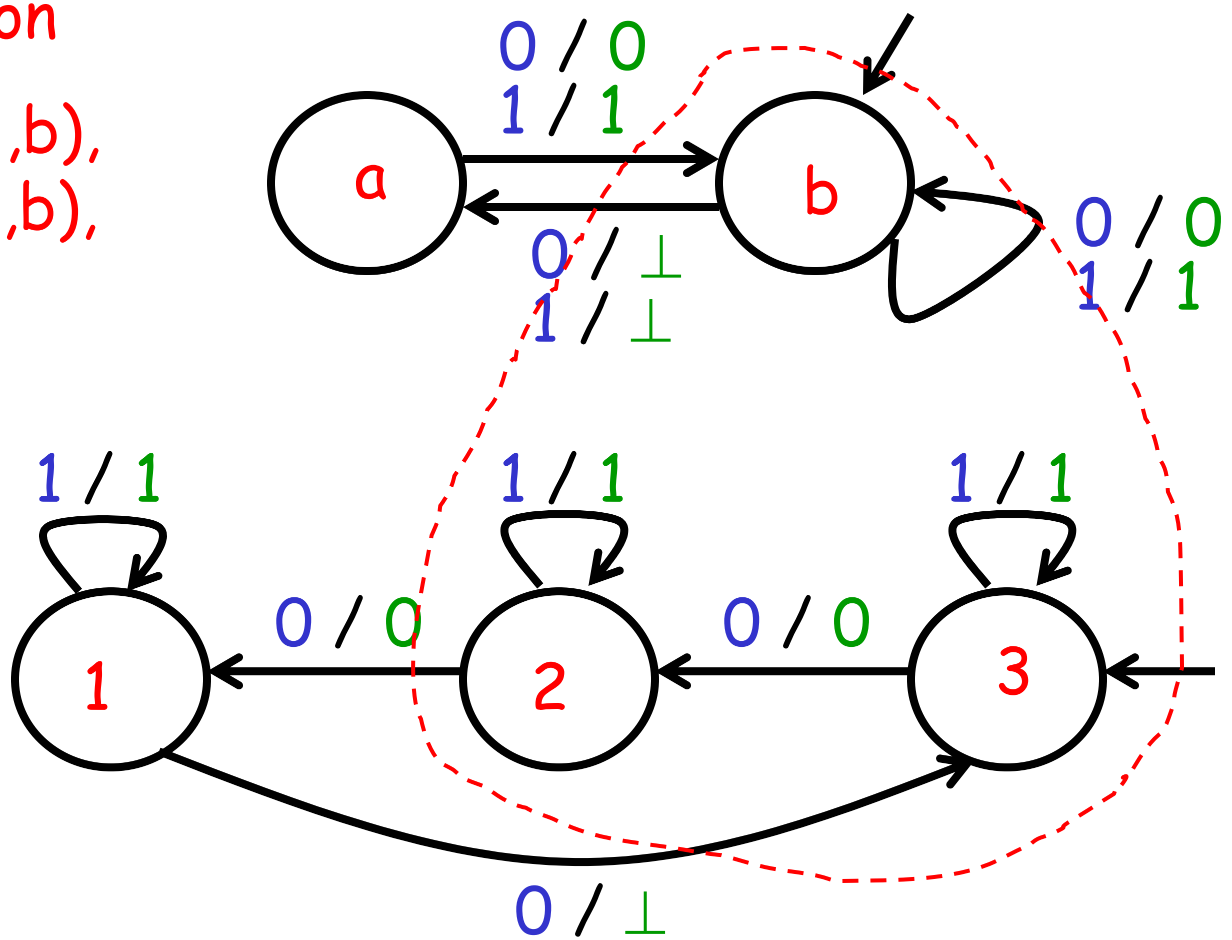
S = { (3,b),



ThirdZero is simulated by NotTwice

Simulation

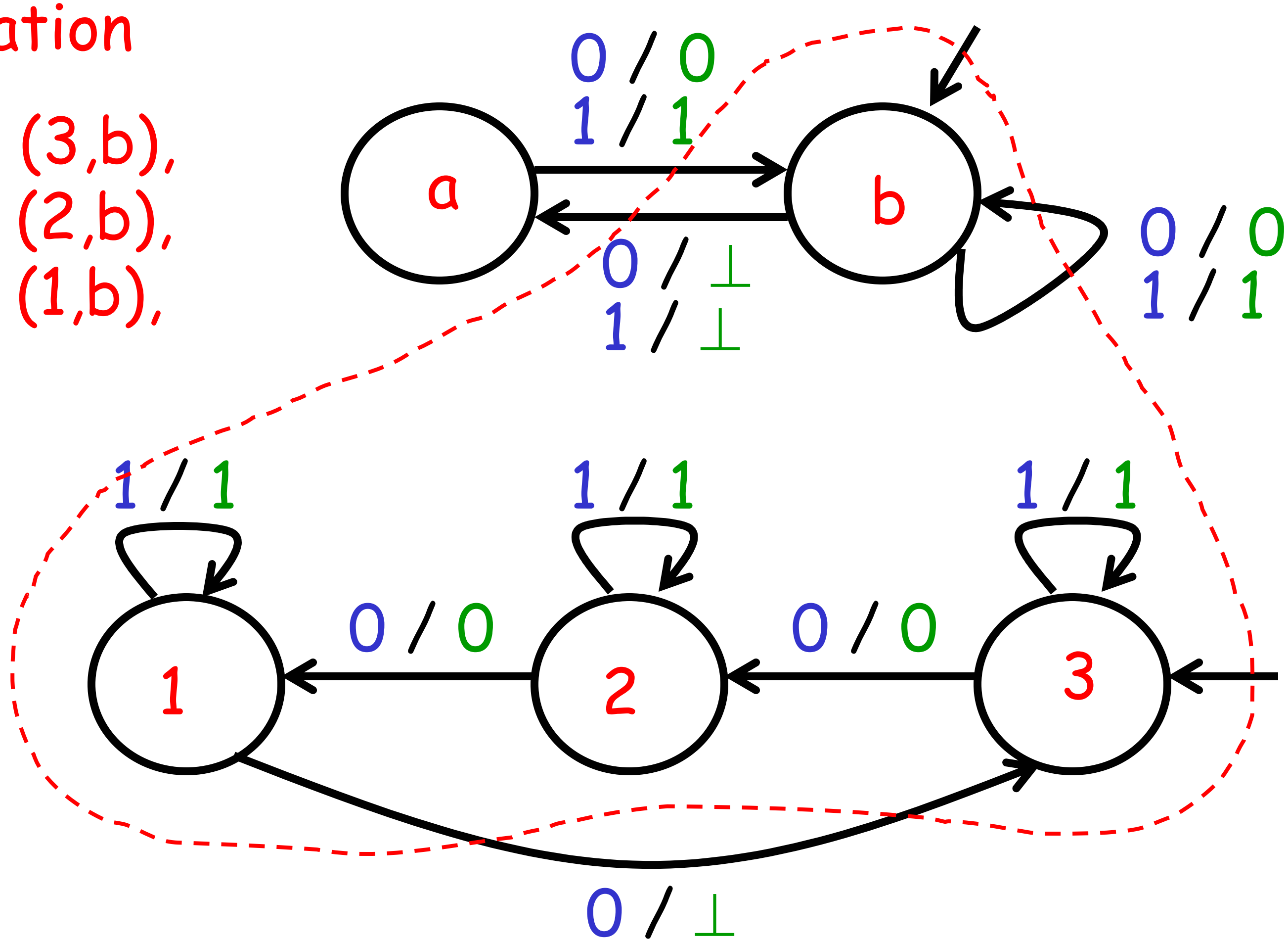
S = { (3,b),
(2,b),



ThirdZero is simulated by NotTwice

Simulation

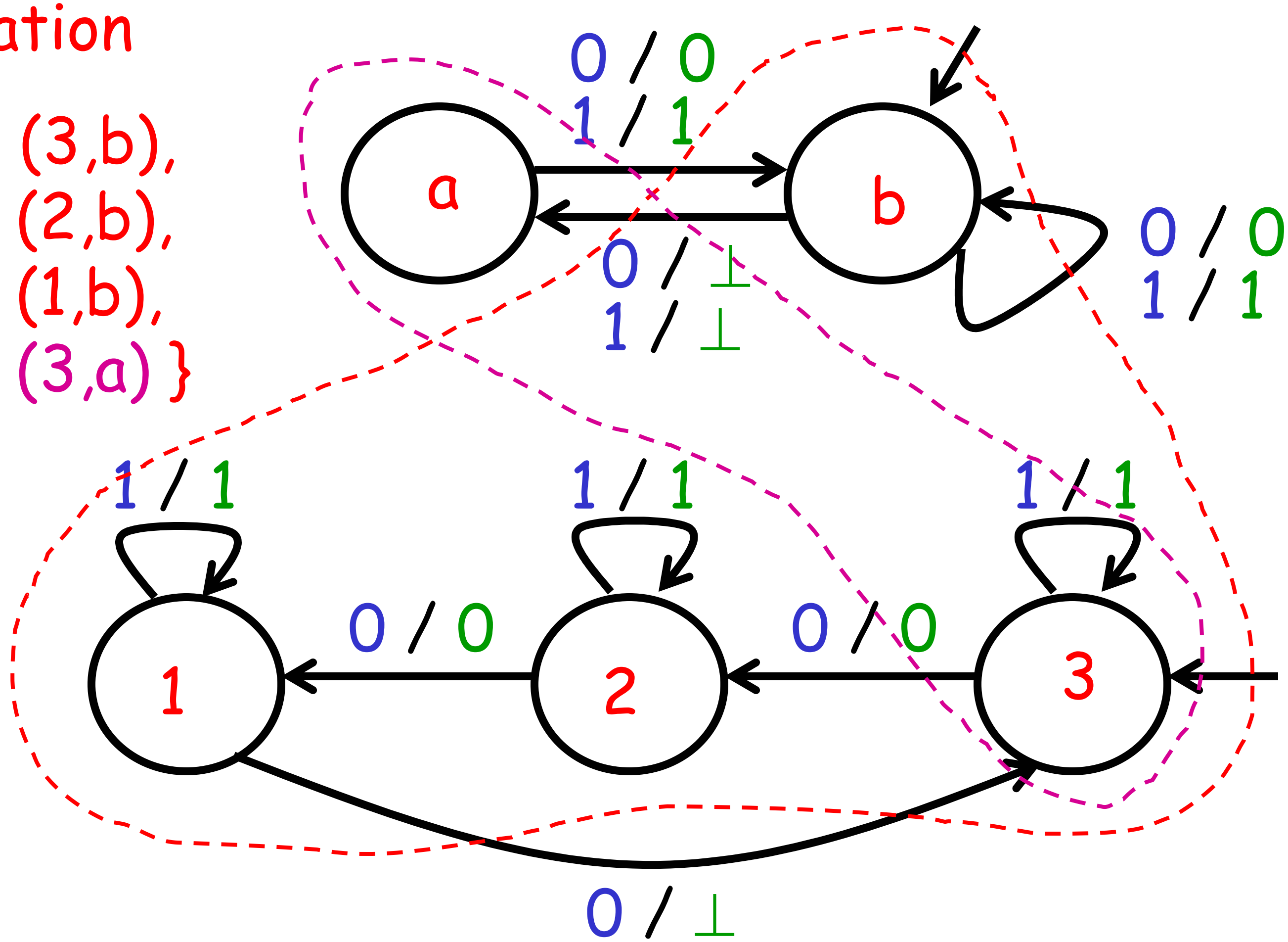
S = { (3,b),
(2,b),
(1,b),



ThirdZero is simulated by NotTwice

Simulation

S = { (3,b),
(2,b),
(1,b),
(3,a) }



Theorem :

The nondeterministic state machines M1 refines the nondeterministic state machine M2

if

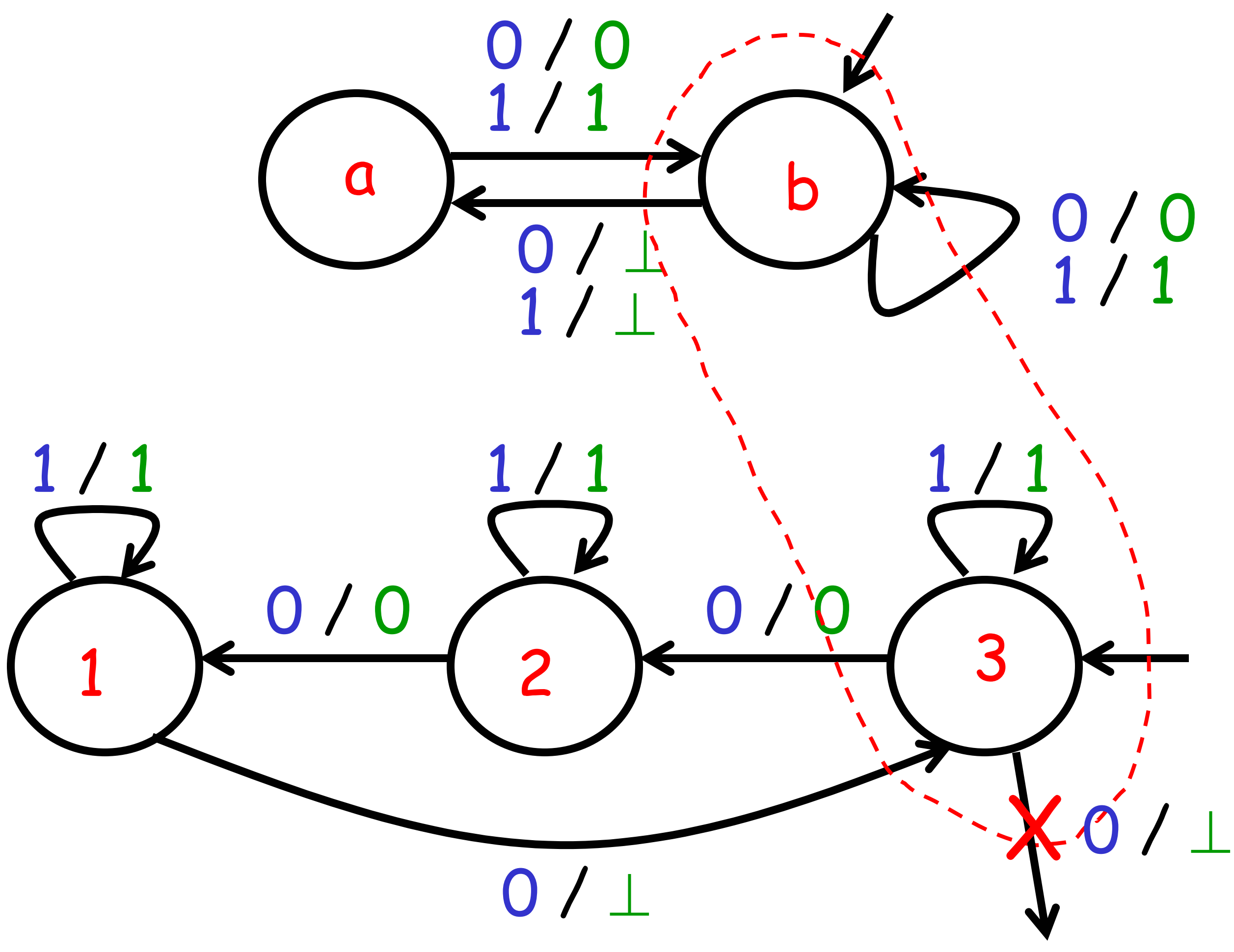
there exists a simulation of M1 by M2 .

Not “if-and-only-if” !

Not symmetric !

(We say that “M2 simulates M1”.)

NotTwice is not simulated by ThirdZero



If M2 is a deterministic state machine, then

M1 is simulated by M2

iff

M1 is equivalent to M2.

If M2 is a deterministic state machine, then

M1 is simulated by M2

iff

M1 is equivalent to M2.

"if and only if"

M1 refines M2, and M2 refines M1

(Behaviors [M1] = Behaviors [M2])

If M2 is a nondeterministic state machine, then

M1 is simulated by M2

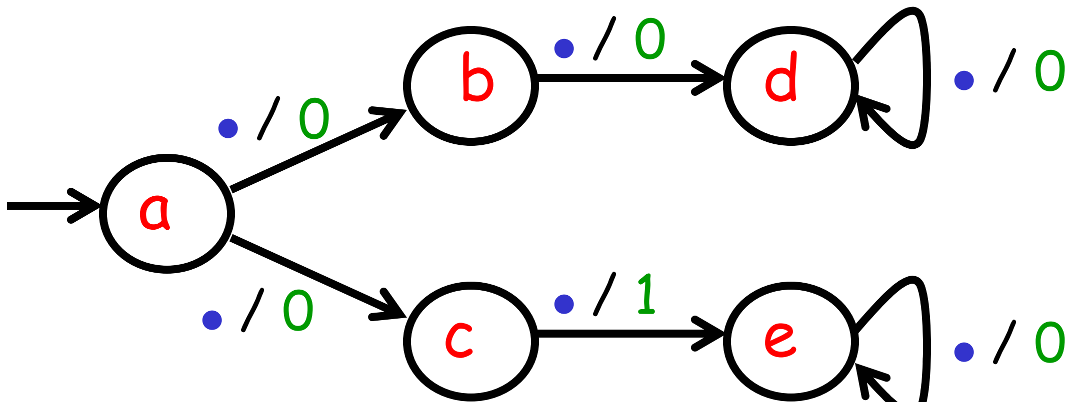
implies

M1 refines M2,

but M1 refine M2 even if M1 is not simulated by M2.

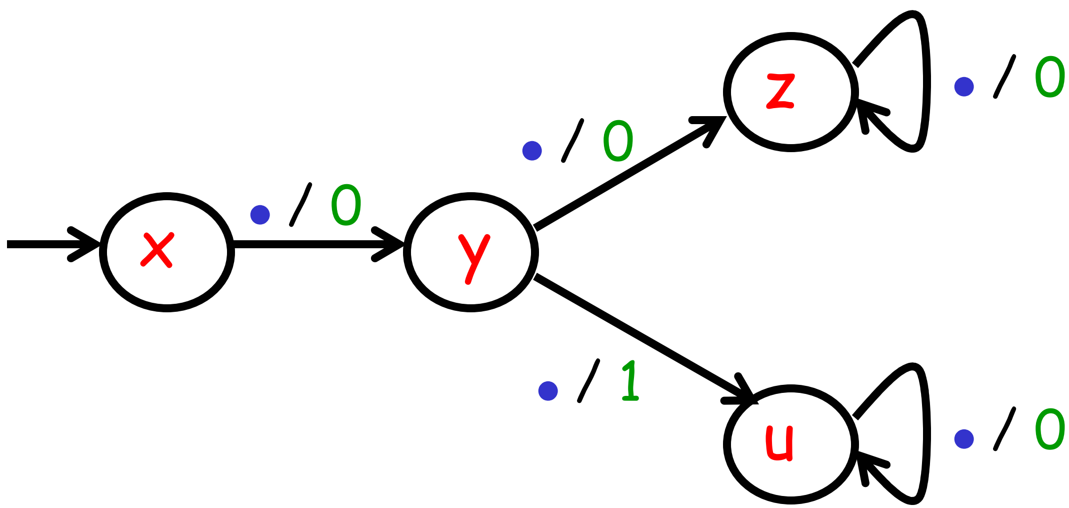
Two behaviors: 00000..., 01000...

M1



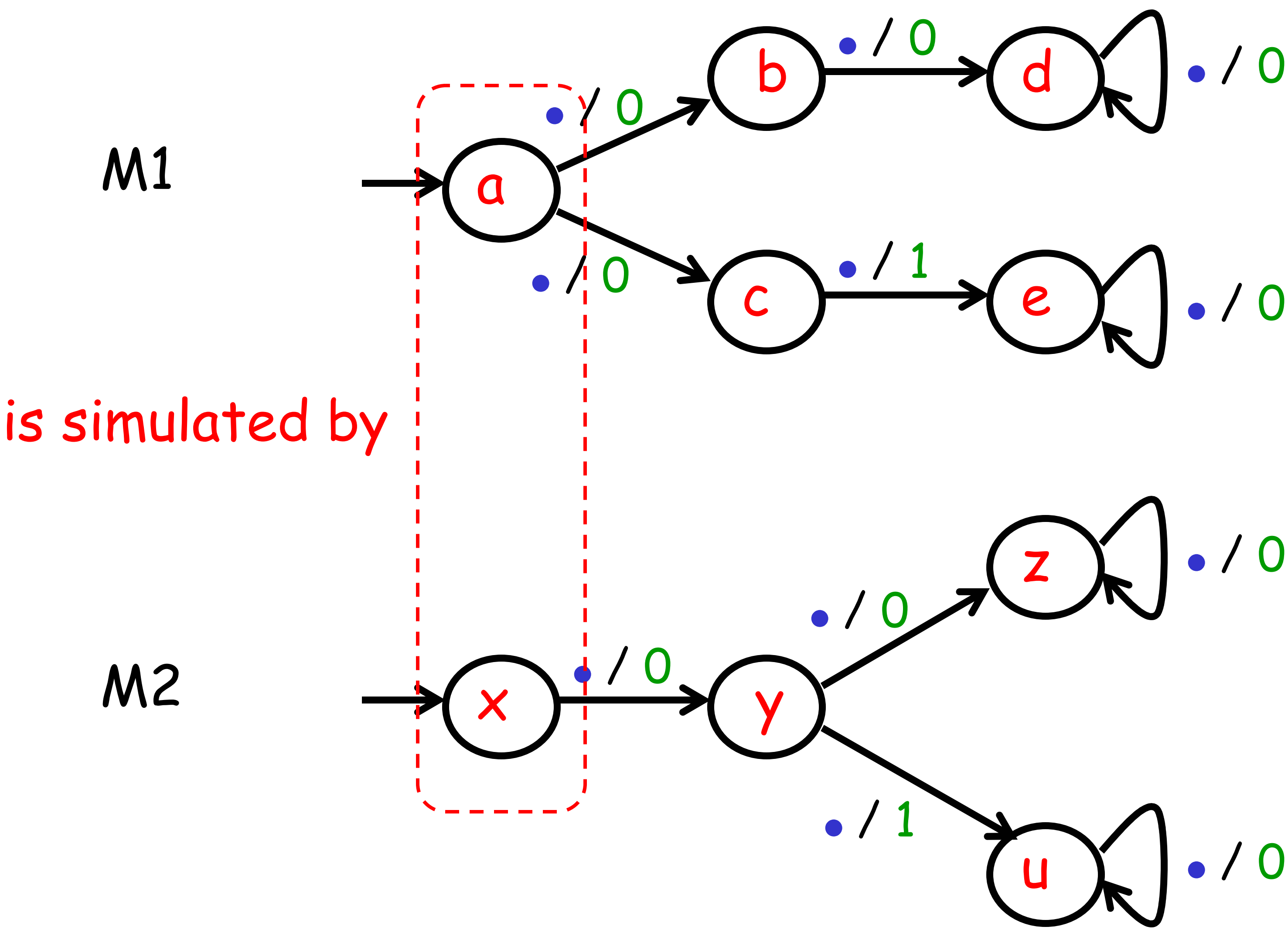
is equivalent to

M2

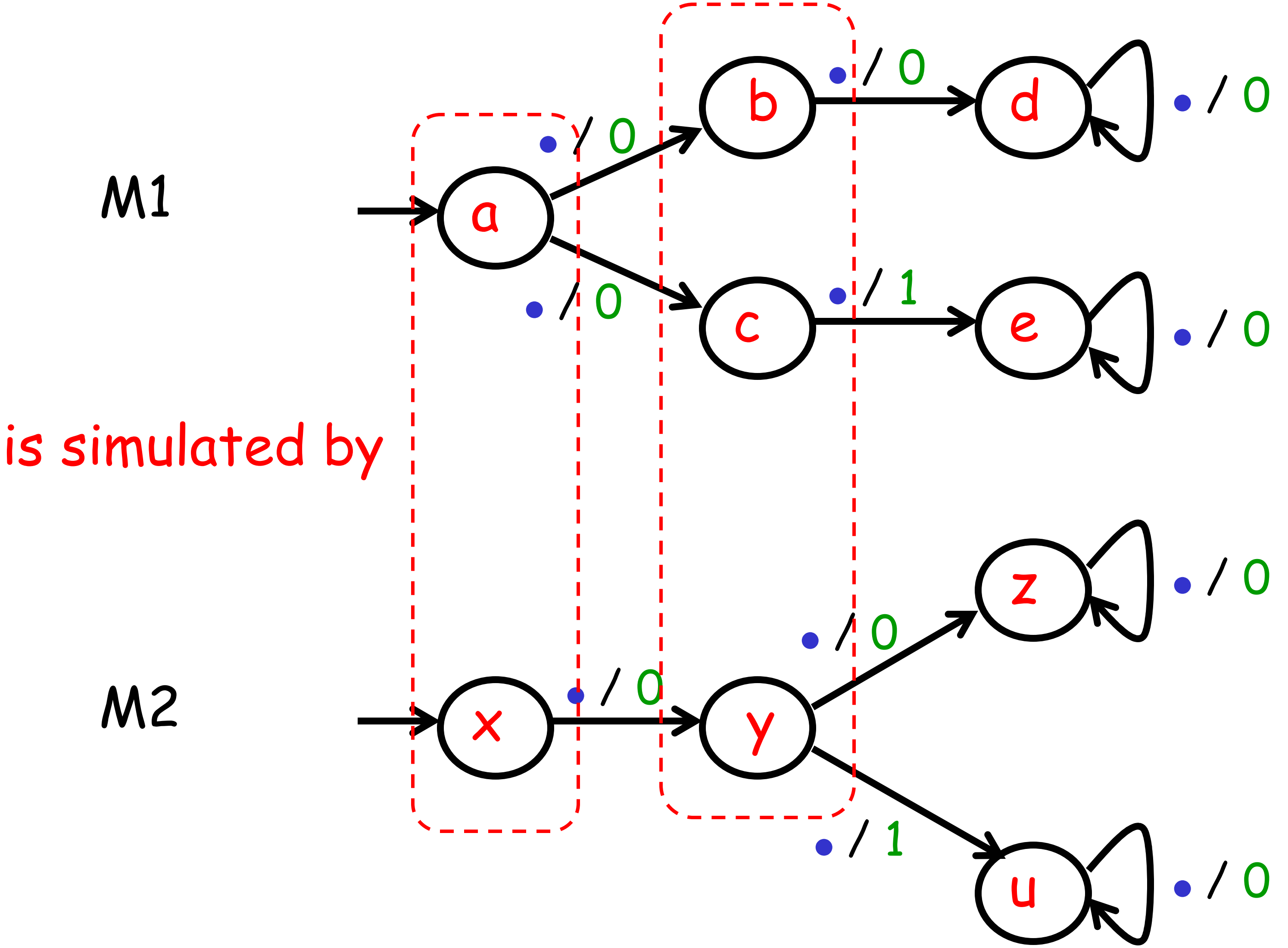


Same two behaviors: 00000..., 01000...

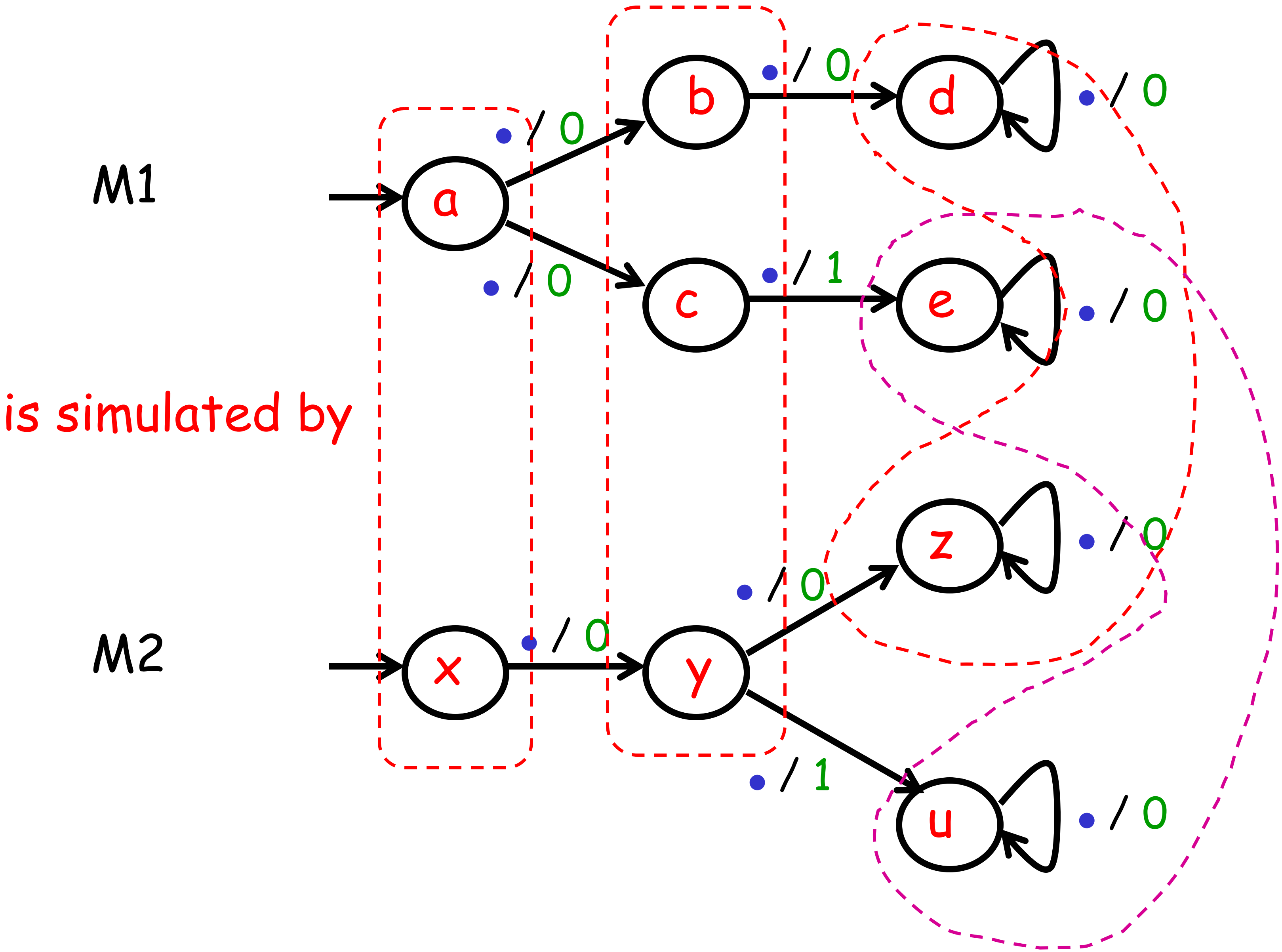
Simulation S = { (a, x),

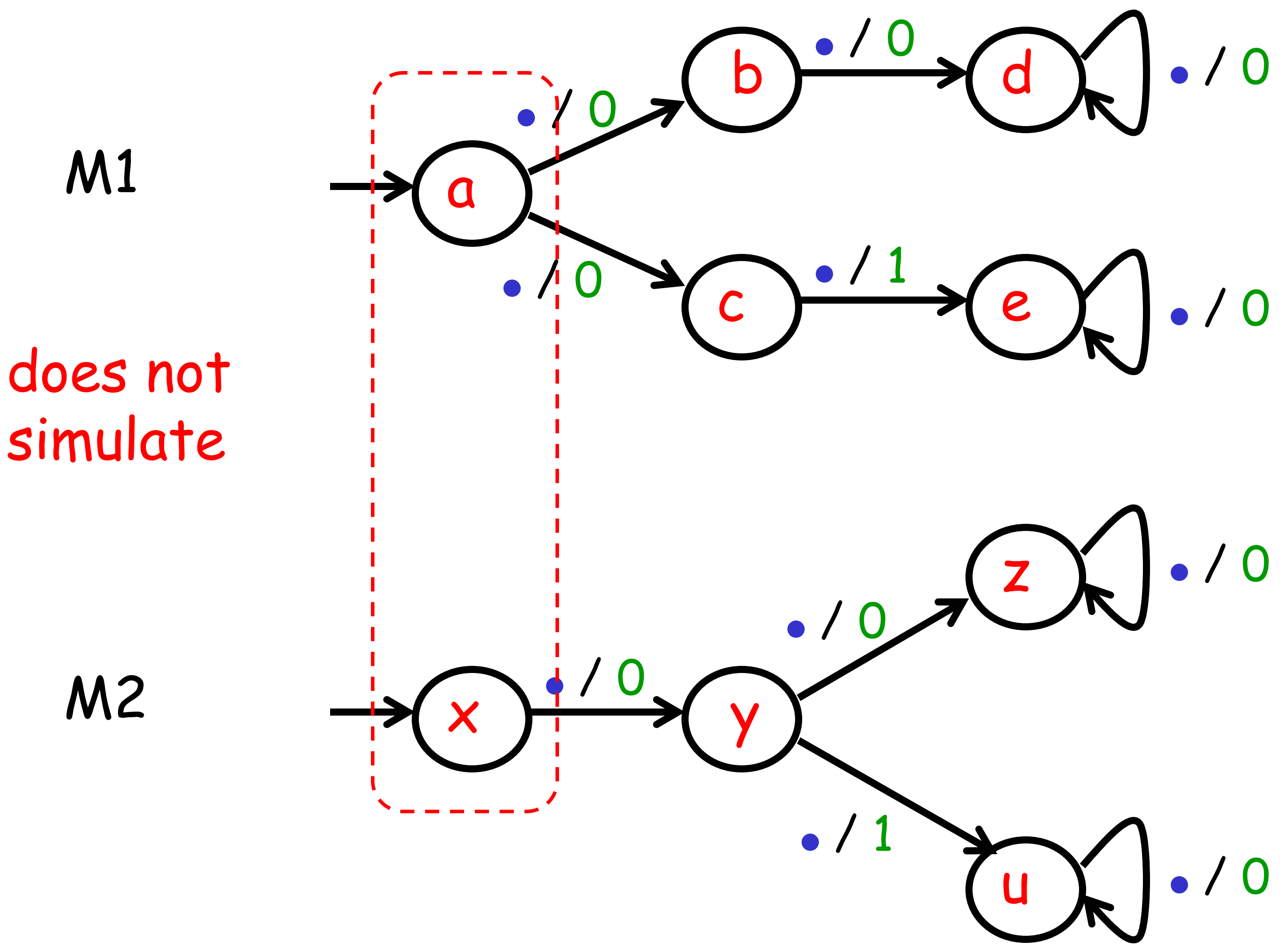


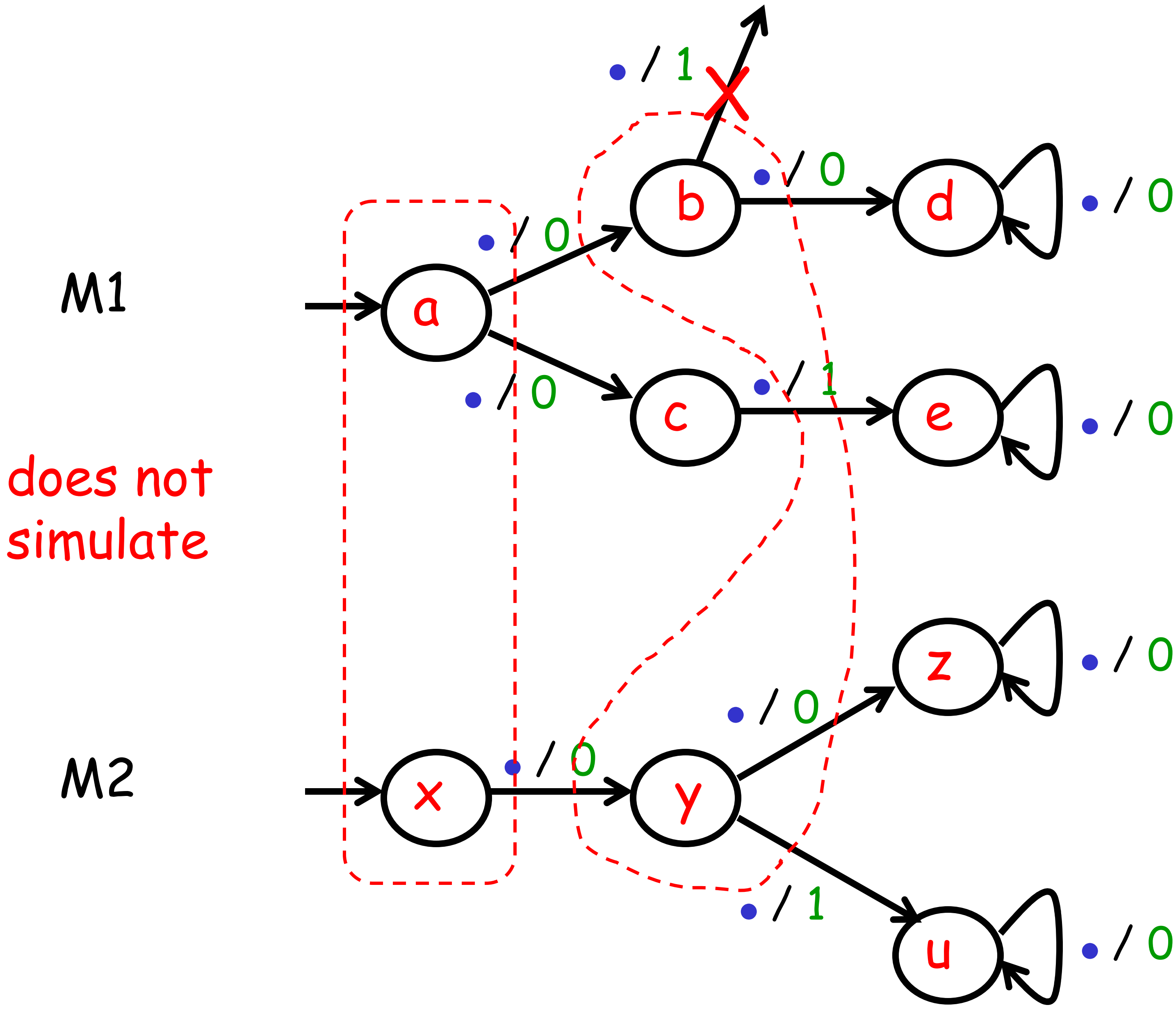
Simulation S = { (a, x), (b, y), (c, y),



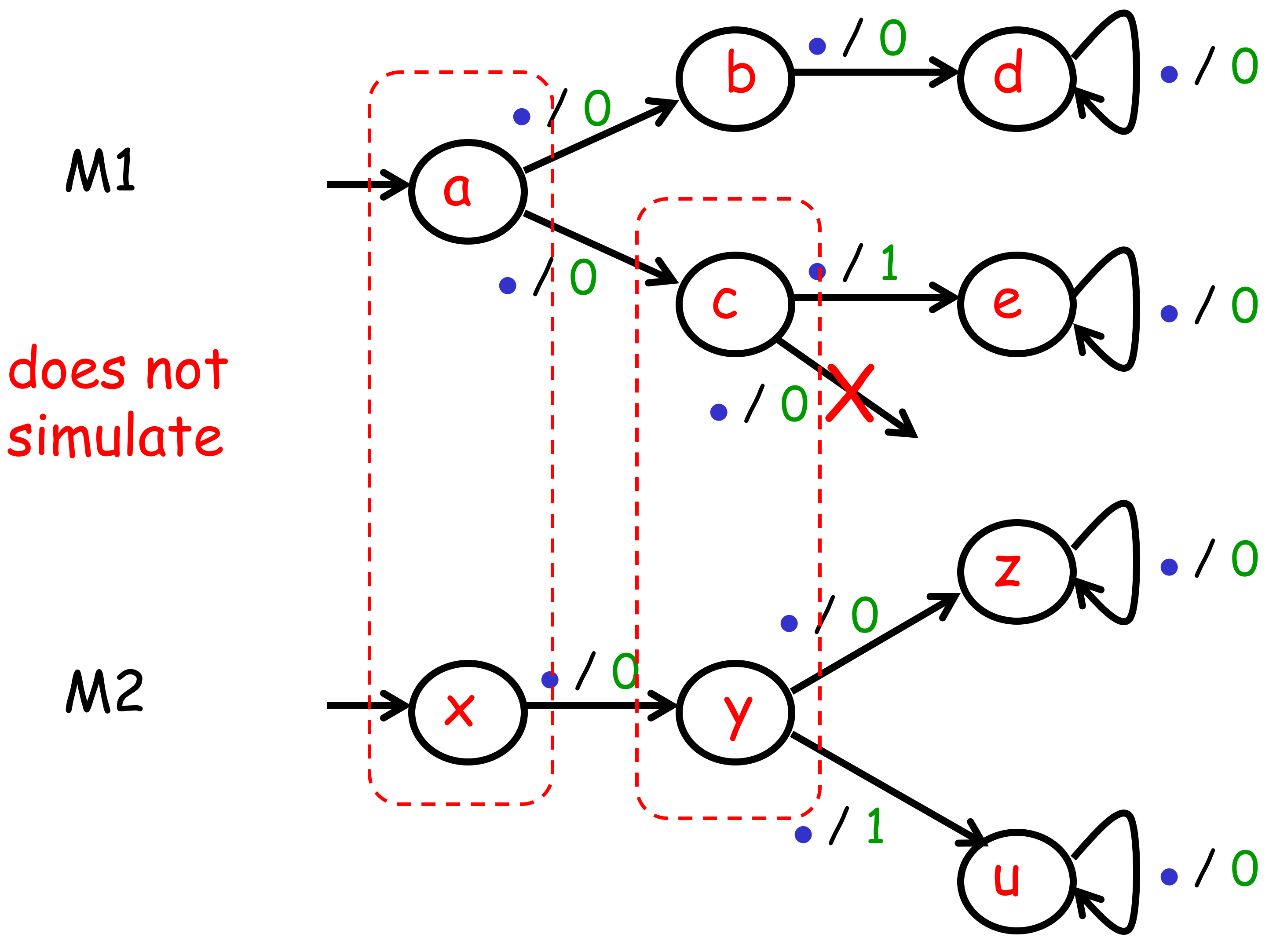
Simulation S = { (a, x), (b, y), (c, y), (d, z), (e, u) }




M1
M2
does not
simulate
a
b
c
d
e
x
y
z
u
• / 0
• / 0
• / 0
• / 1
• / 0
• / 0
• / 0
• / 0
• / 0
• / 1
• / 0
• / 0

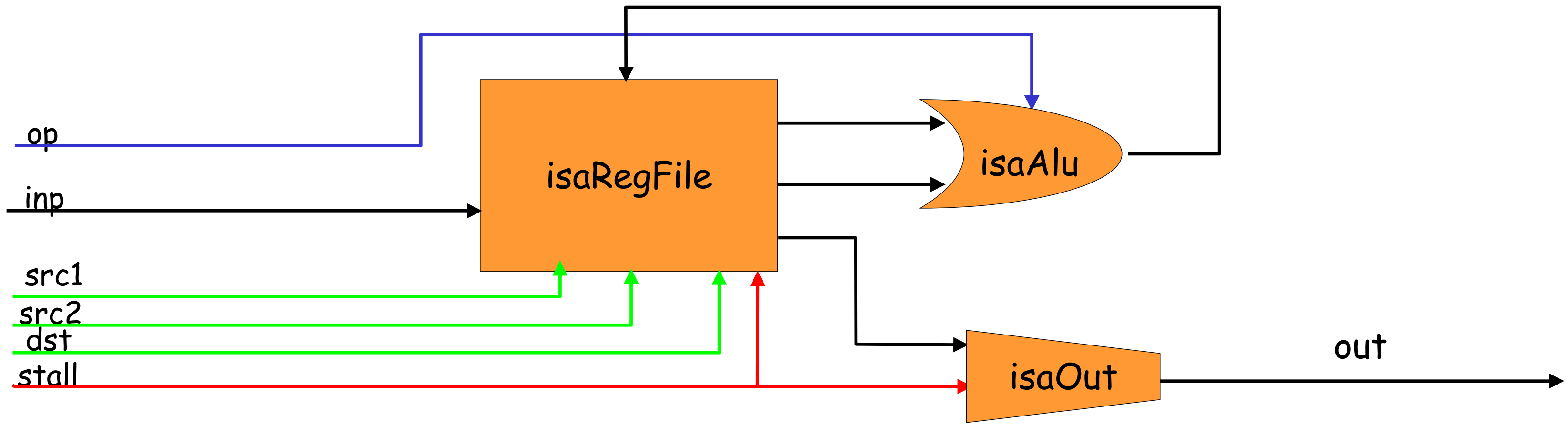


M1
M2
does not simulate
a
b
c
d
e
x
y
z
u
• / 0
• / 0
• / 1
• / 0
• / 1
• / 0
• / 0
• / 0
• / 0
• / 1
• / 0
• / 0

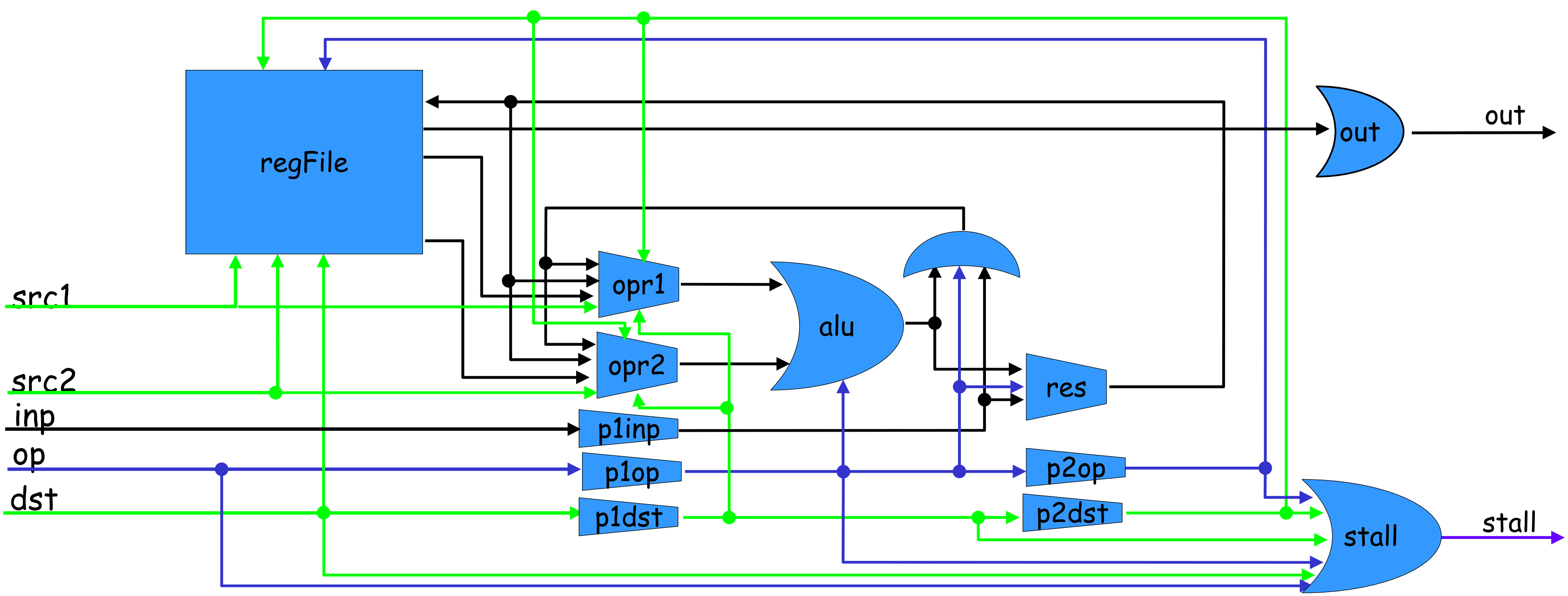


M1
M2
does not simulate
a
b
c
d
e
x
y
z
u
• / 0
• / 0
• / 0
• / 1
• / 0
• / 0
• / 0
• / 0
• / 0
• / 0
• / 1
• / 0
• / 0

In general, it requires a quadratic algorithm to find simulations ...



Goal: establish that ISA simulates Pipeline.



But finding simulations is easy (linear) for special cases of nondeterministic state machines:

1. Deterministic
2. Output-deterministic (“almost deterministic”)

In this cases, the “informal” algorithm we used to find matching pairs of states always works.

A state machine is output-deterministic

iff

for every state and every input-output pair,
there is only one successor state.

Deterministic implies output-deterministic,
but not vice versa.

For example, LCwD and NotTwice are output-deterministic;
ThirdZero is deterministic.

Deterministic: for every input signal x, there is exactly one run of the state machine.

Output-deterministic: for every behavior (x,y), there is exactly one run.

If M2 is an output-deterministic state machine, then a simulation S of M1 by M2 can be found as follows:

1. (initialState [M1], initialState[M2]) $\in$ S.

2. If $(p,q) \in S$ and

   $(p',y) \in$ possibleUpdates [M1] (p,x)

   then

   $\exists q'$ s.t. $(q',y) \in$ possibleUpdates [M2] (q,x),

   and $\forall q'$ — No "guessing" of successor state involved!

   if $(q',y) \in$ possibleUpdates [M2] (q,x),

   then $(p',q') \in S$.