Determinization

EECS 20

Lecture 16 (February 26, 2001)

Tom Henzinger

Revised by Matteo Zavatteri and Tiziano Villa, Fall 2020

State machines

Deterministic

$\Downarrow \quad \not\Uparrow$

Output-deterministic

$\Downarrow \quad \not\Uparrow$

Nondeterministic

A state machine is deterministic

iff

1. there is only one initial state, and
2. for every state and every input, there is only one successor state.

Hence, for every input signal there is exactly one run.

A state machine is output-deterministic

iff

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

Hence, for every behavior there is exactly one run.

State machines

Simulations can be found easily

Deterministic

$\Downarrow$

Output-deterministic

$\Downarrow$

Nondeterministic

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

If M2 is an output-deterministic state machine, then

M1 is simulated by M2

iff

M1 refines M2.

If M2 is a nondeterministic state machine, then

M1 is simulated by M2

implies

M1 refines M2.

Fortunately:

For every nondeterministic state machine M, we can find an output-deterministic state machine det(M) that is equivalent to M.

(This is called "subset construction.")

Then, to check if M1 refines M2,
check if M1 is simulated by det(M2):

M1 refines M2

iff

M1 refines det(M2)

iff

M1 is simulated by det(M2).

Then, to check if M1 refines M2,
check if M1 is simulated by det(M2):

M1 refines M2

iff

M1 refines det(M2)

iff

M1 is simulated by det(M2).

output-deterministic

The Subset Construction

Given: nondeterministic state machine M

Find: output-deterministic state machine det(M) that is equivalent to M

The Subset Construction

Given: nondeterministic state machine M

Find: output-deterministic state machine $det(M)$
that is equivalent to M

$$Inputs\,[det(M)] = Inputs\,[M]$$

$$Outputs\,[det(M)] = Outputs\,[M]$$

The Subset Construction

Let initialState [det(M)] = possibleInitialStates [M] ;

Let States [det(M)] = { initialState [det(M)] } ;

Repeat as long as new transitions can be added to det(M) :

Choose $P \in$ States [det(M)] and $(x,y) \in$ Inputs $\times$ Outputs ;

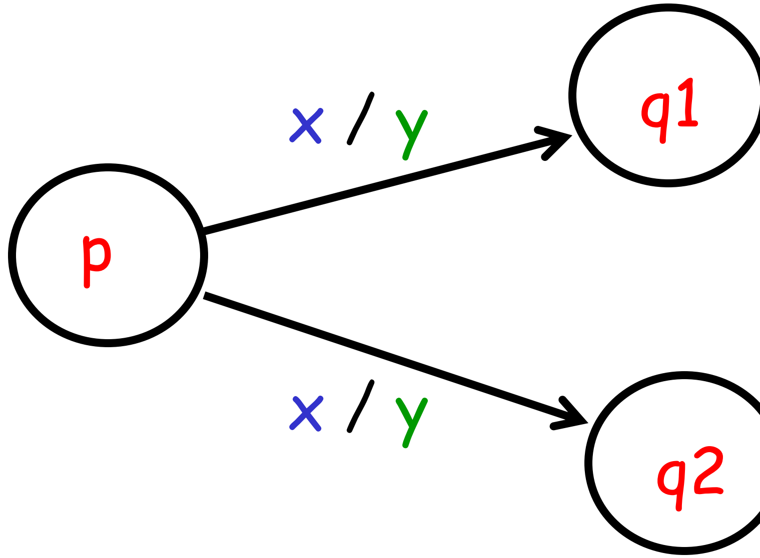
Let $Q = \{ q \in$ States [M] $\mid \exists\, p \in P,\ (q,y) \in$ possibleUpdates [M] $(p,x) \}$;

If $Q \neq \emptyset$ then

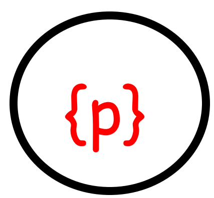
Let States [det(M)] = States [det(M)] $\cup \{Q\}$;

Let update [det(M)] $(P,x) = (Q,y)$.

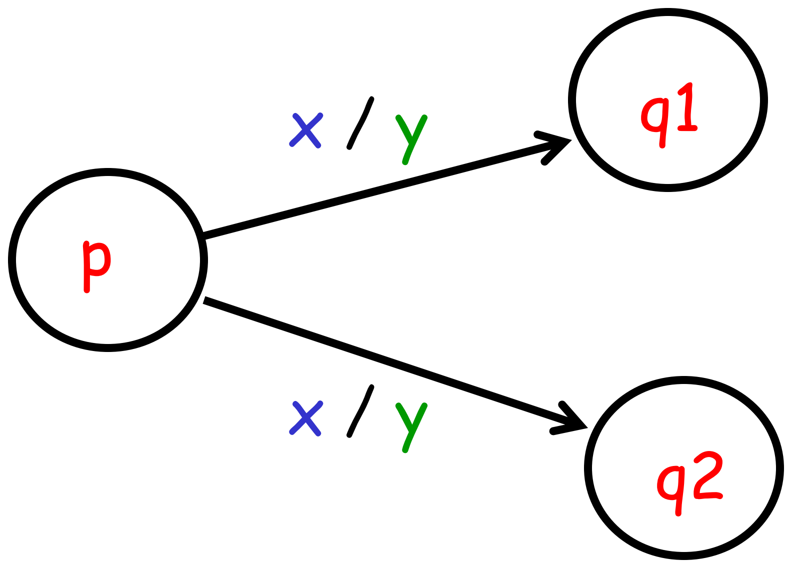
M



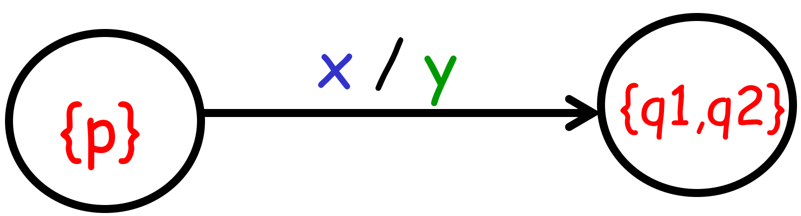
det(M)



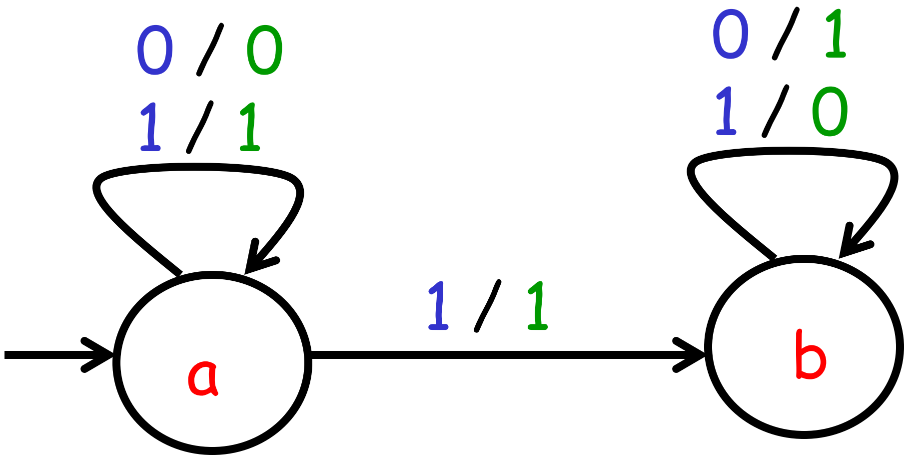
M



det(M)

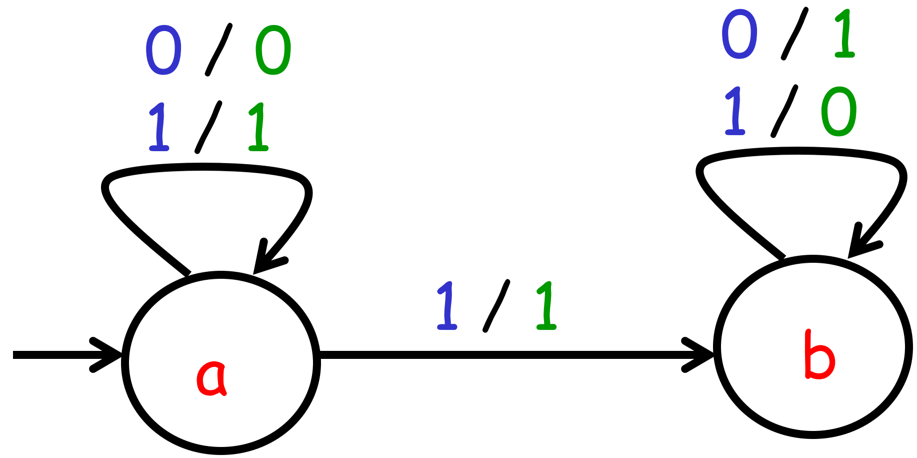


M

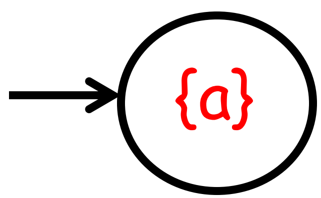


det(M)

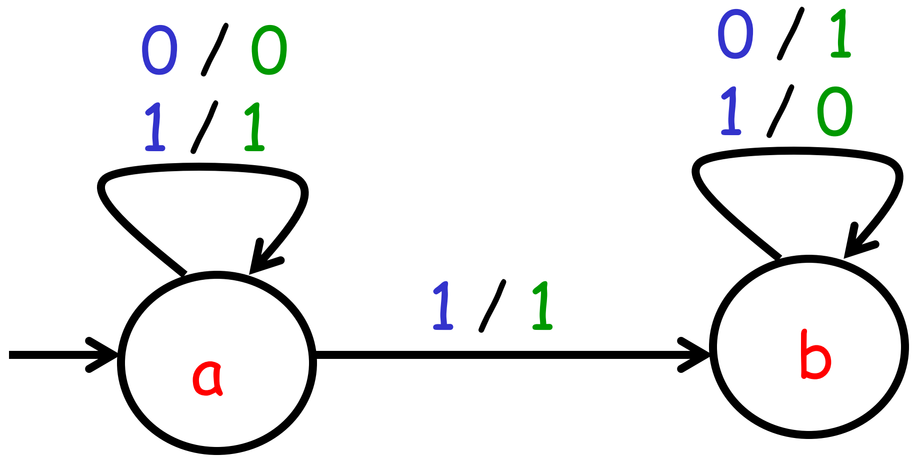
M



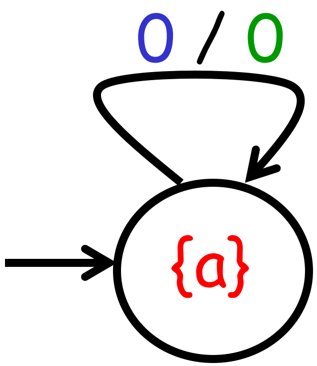
det(M)



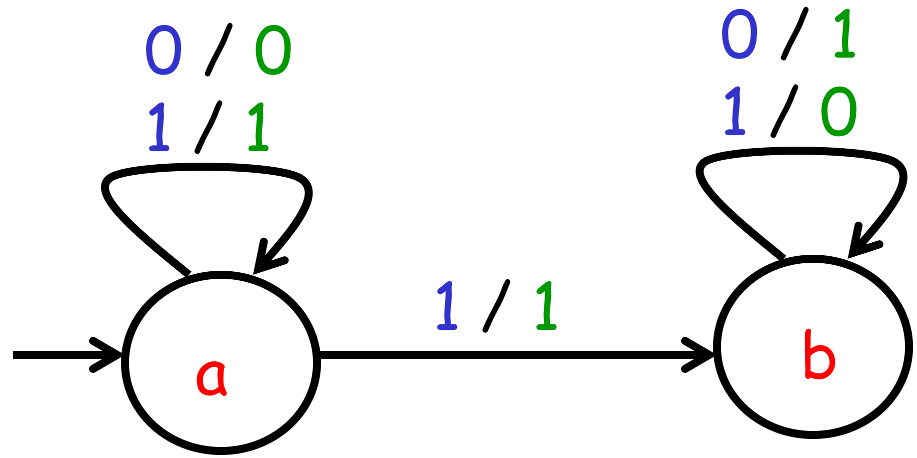
M



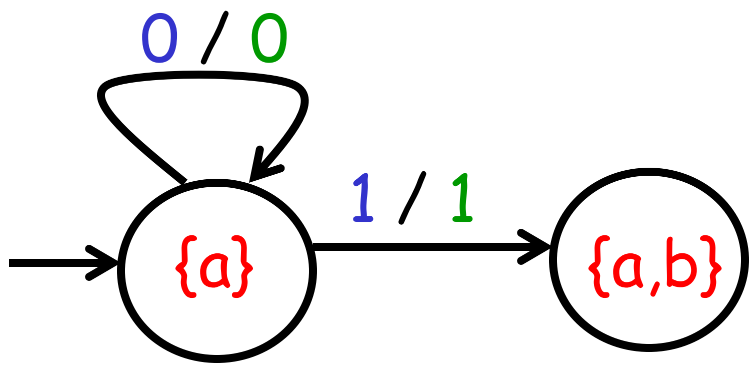
det(M)



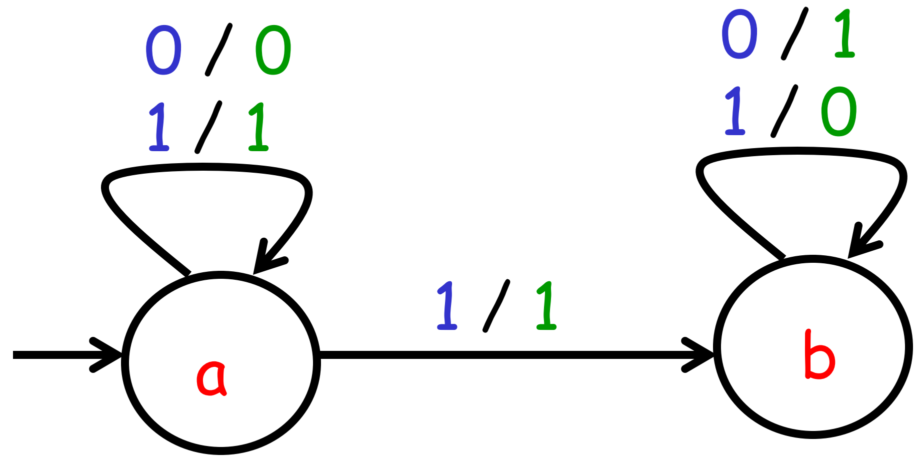
M



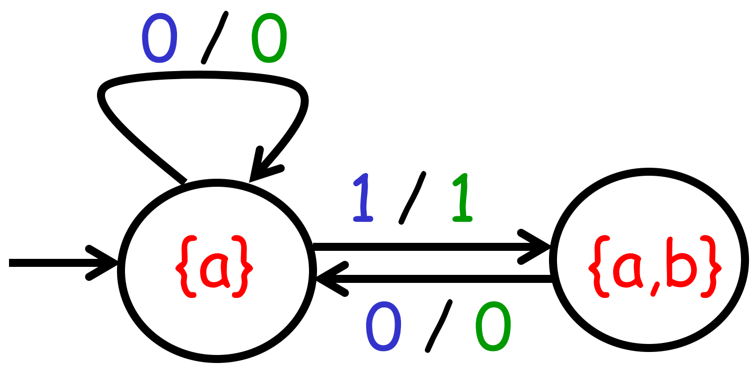
det(M)



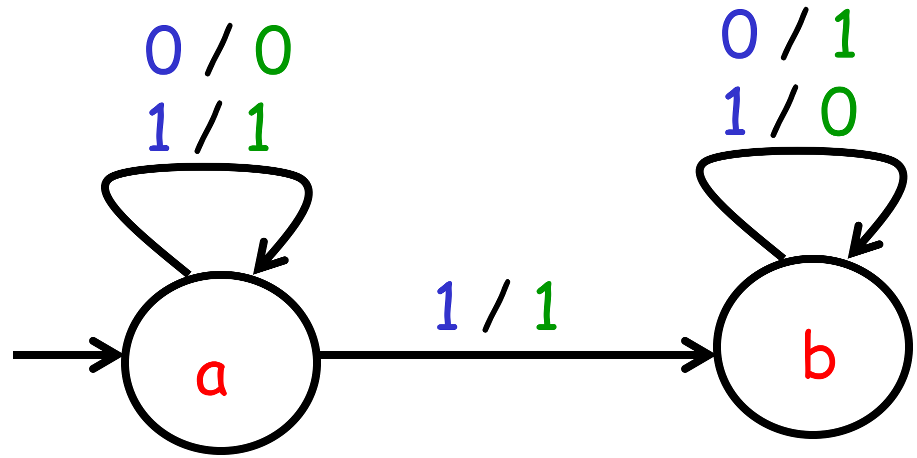
M



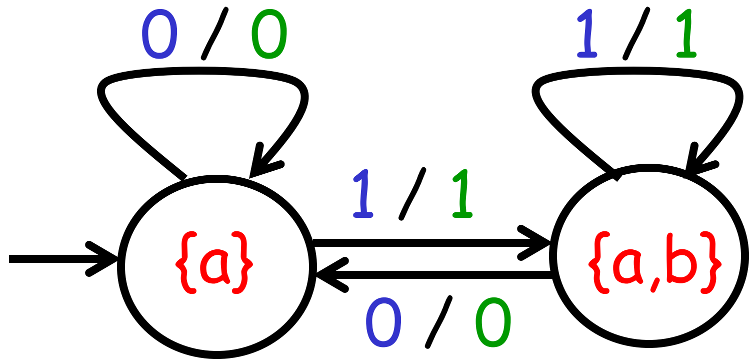
det(M)



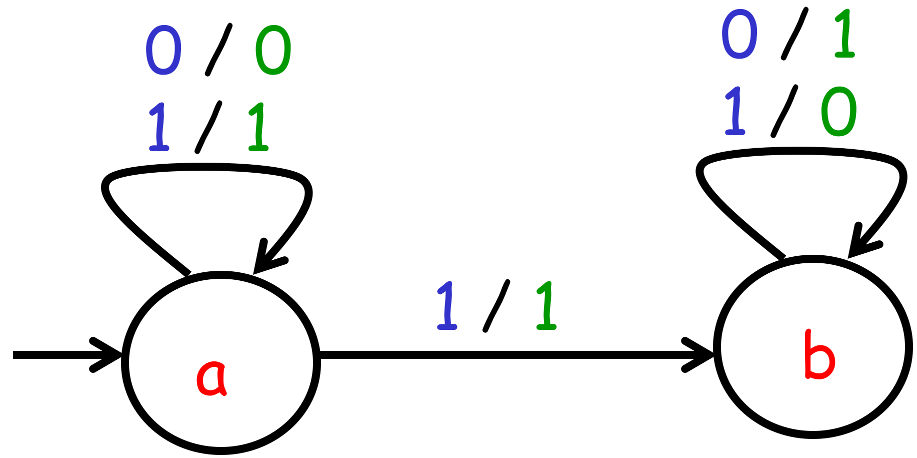
M



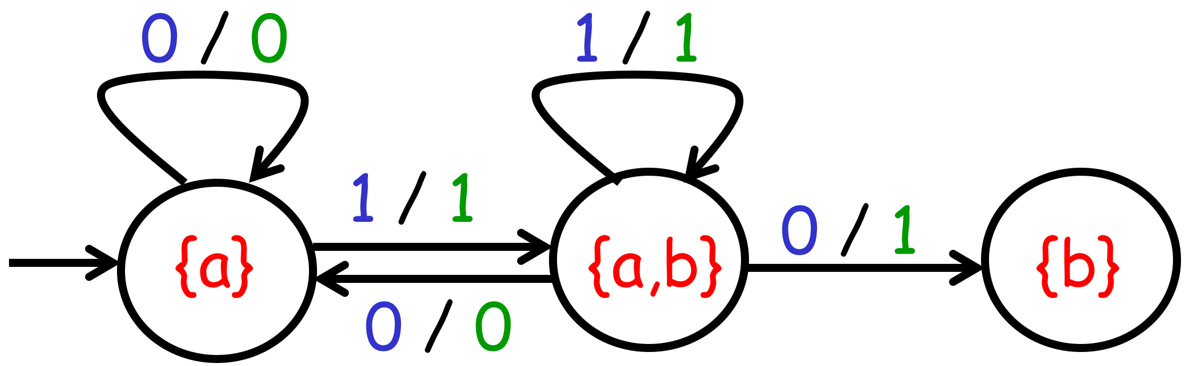
det(M)



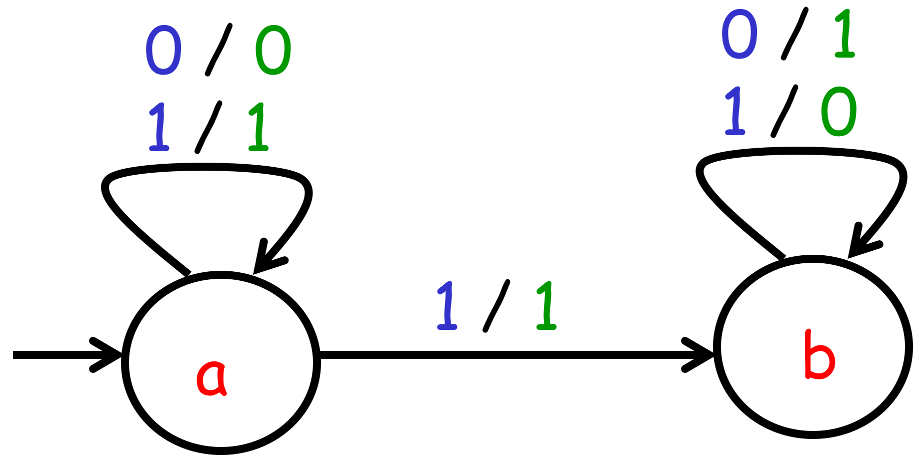
M



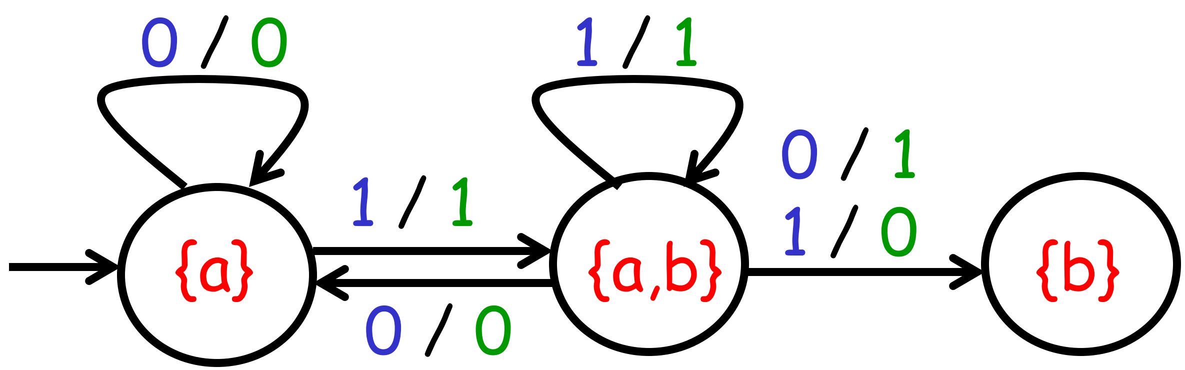
det(M)



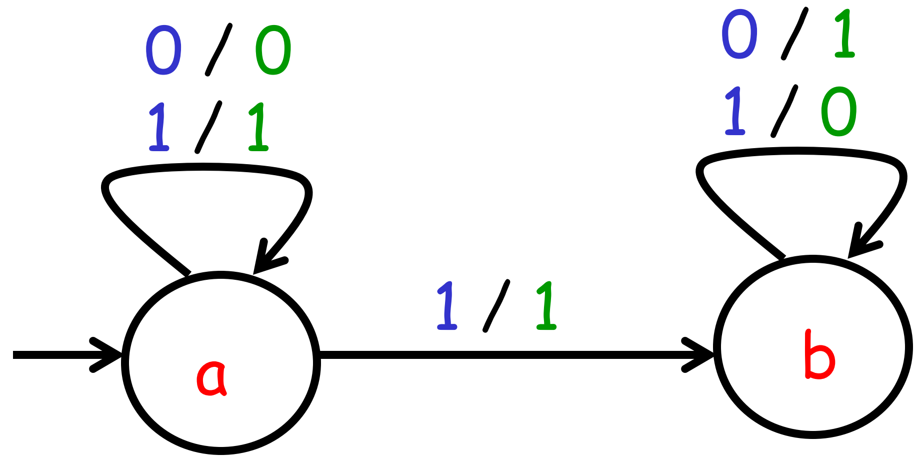
M



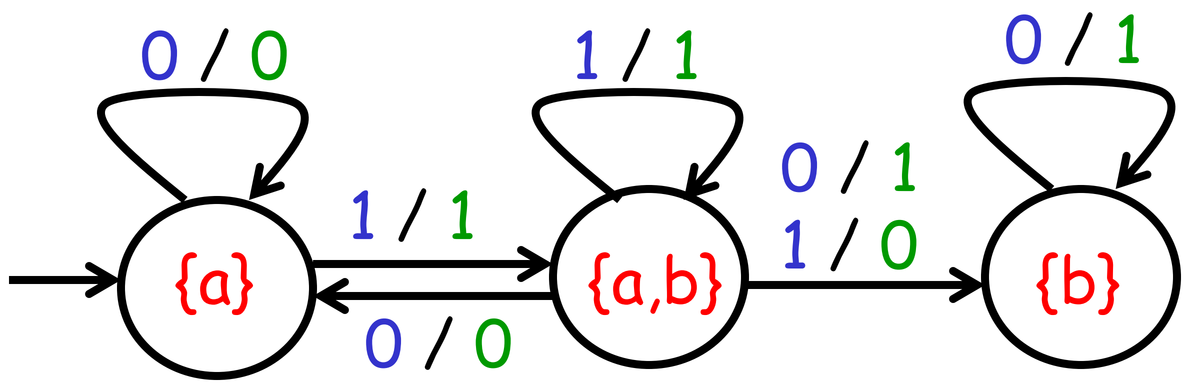
det(M)



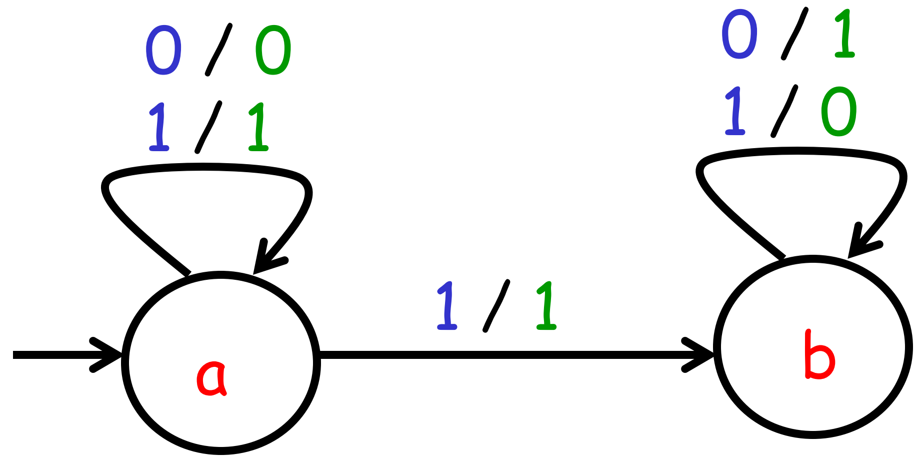
M



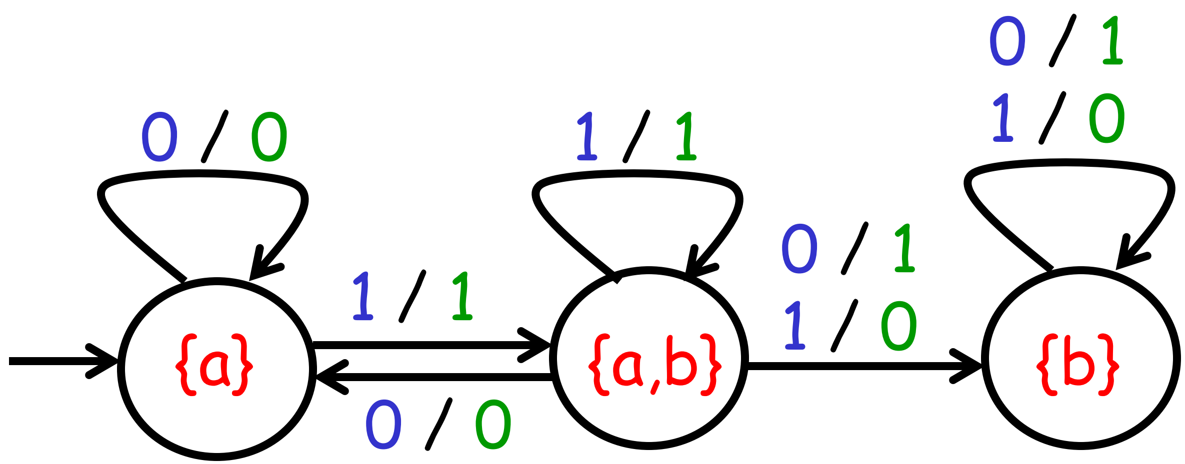
det(M)



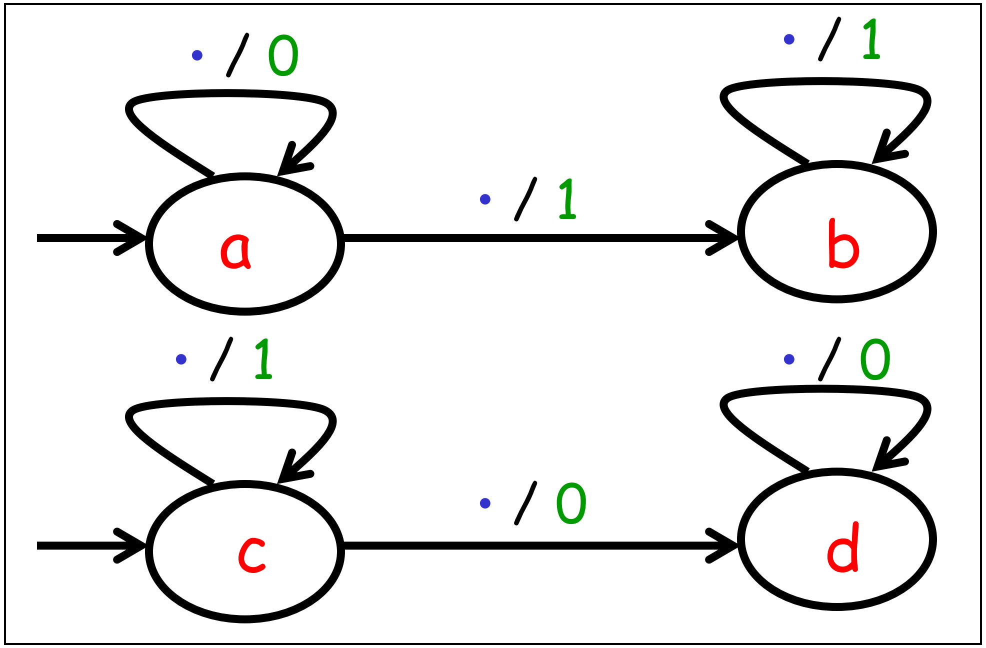
M



det(M)

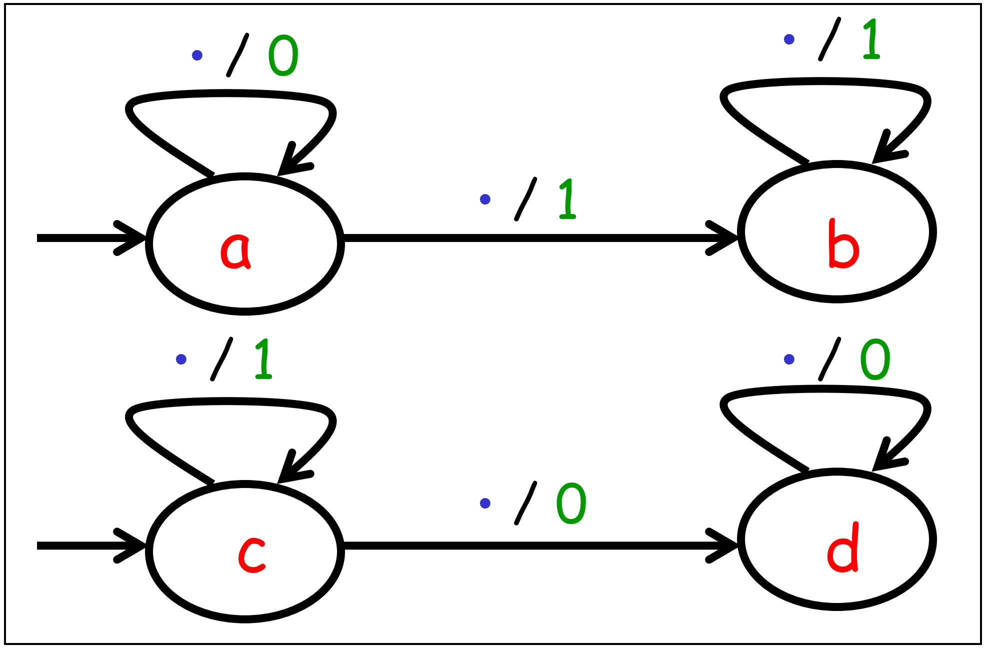


M

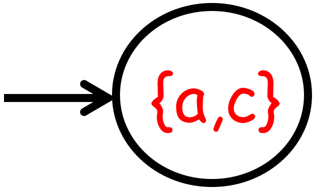


det(M)

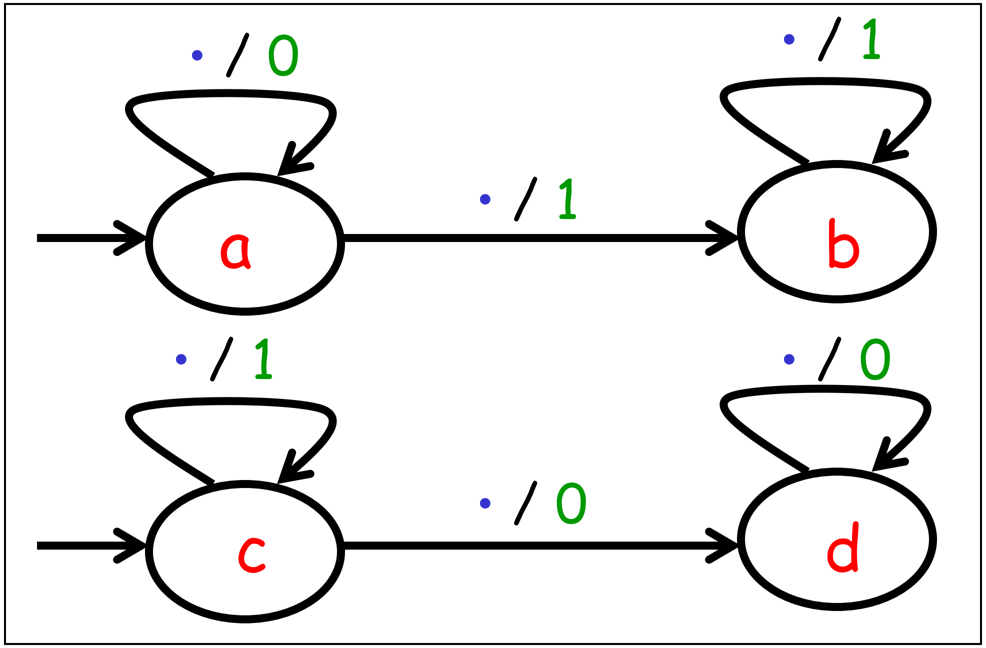
M



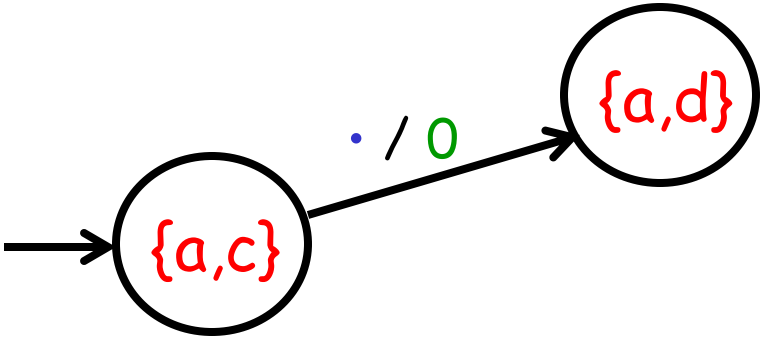
det(M)



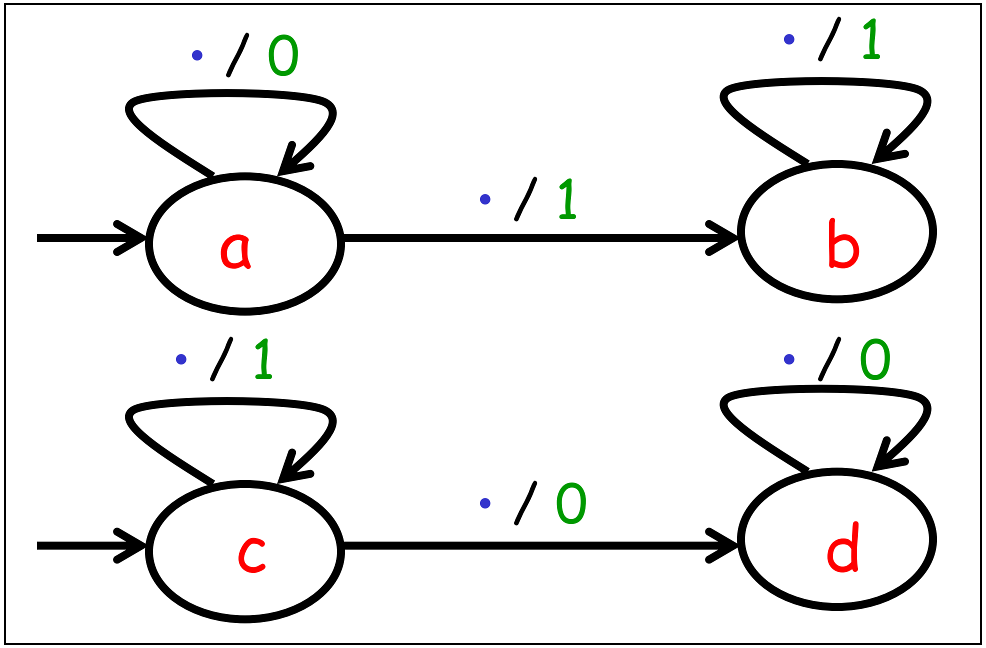
M



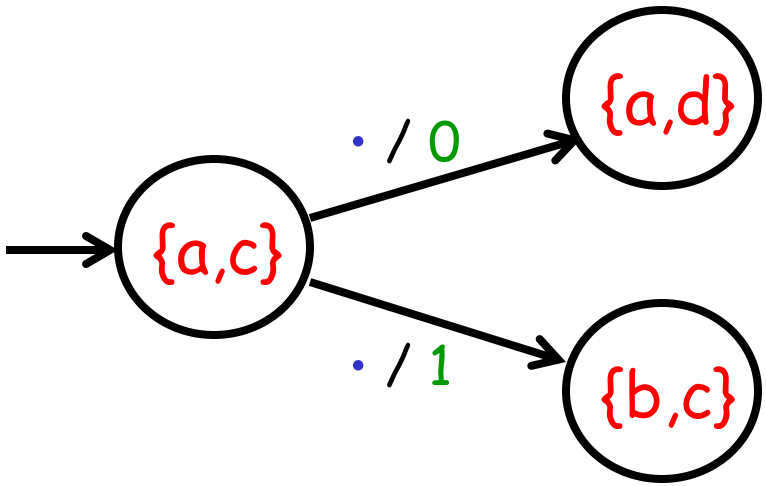
det(M)



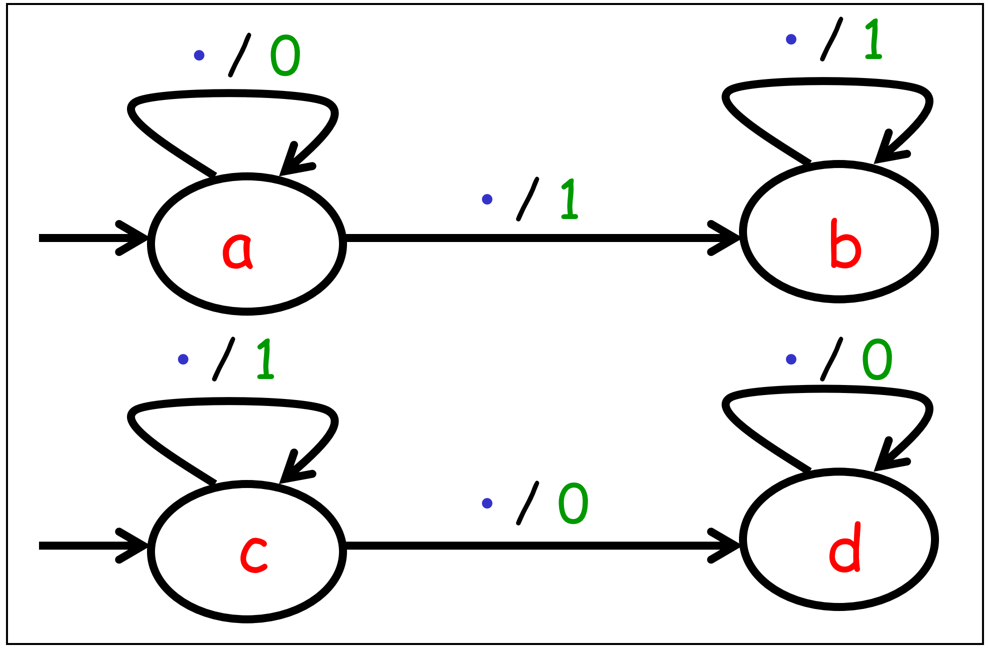
M



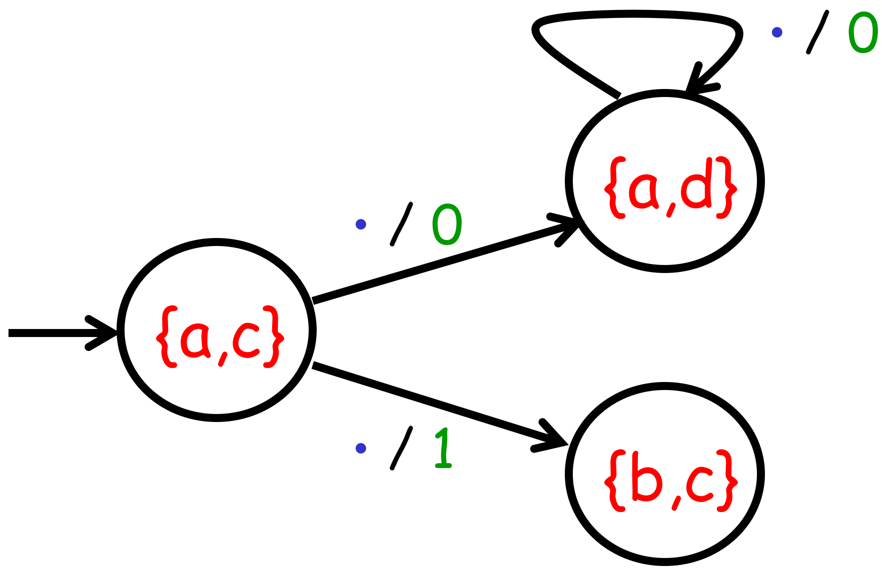
det(M)



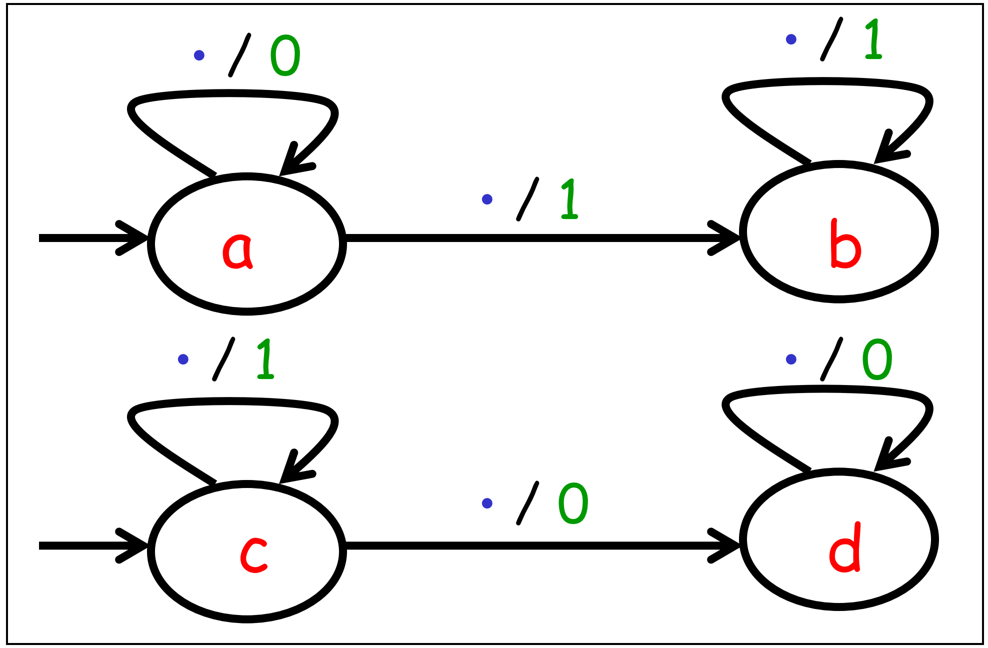
M



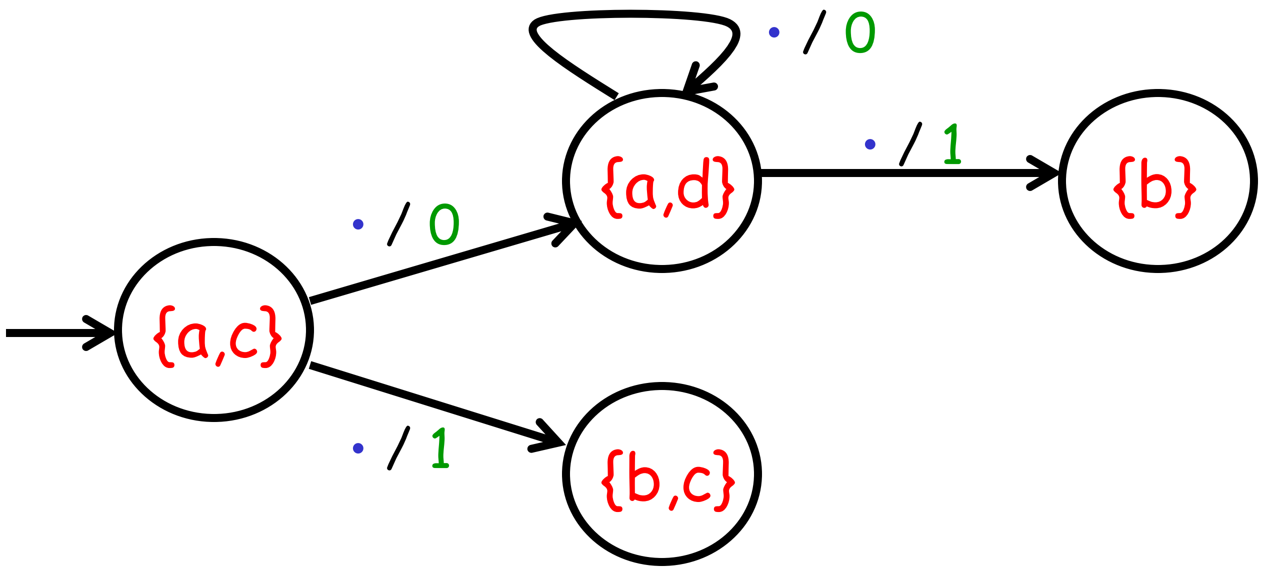
det(M)



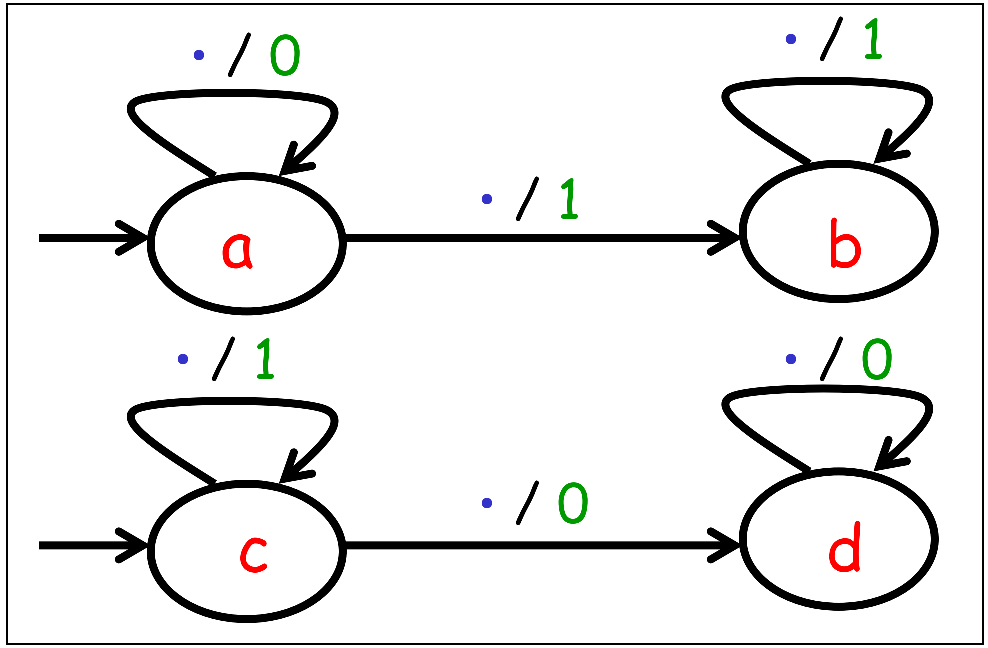
M



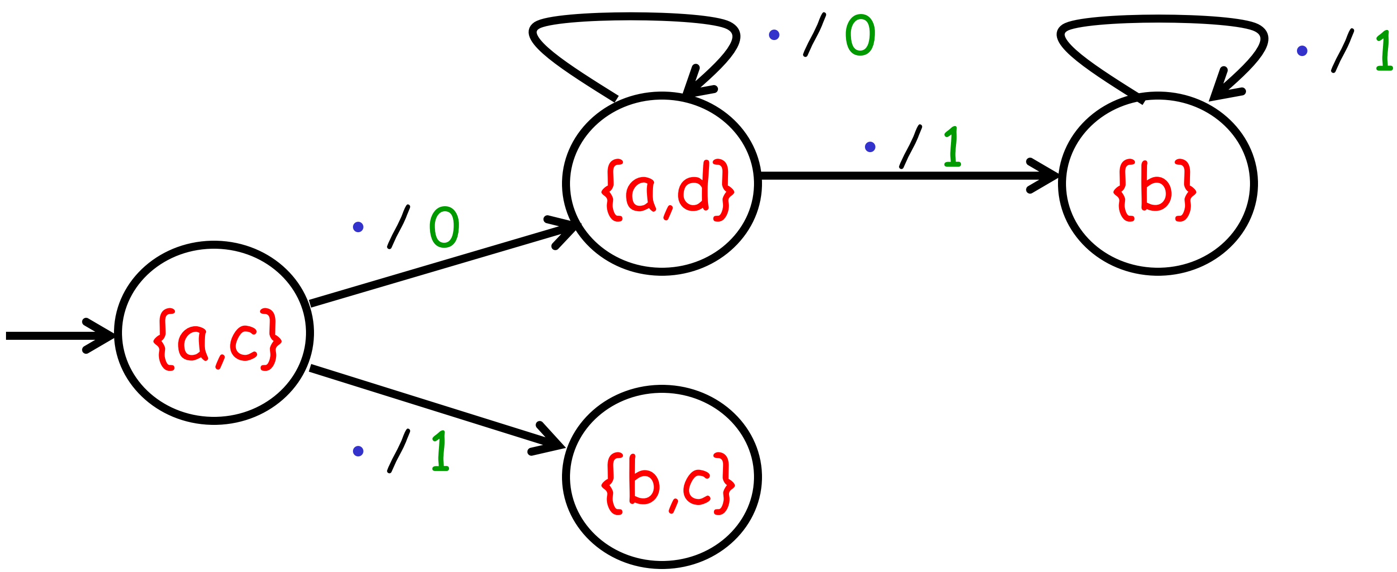
det(M)



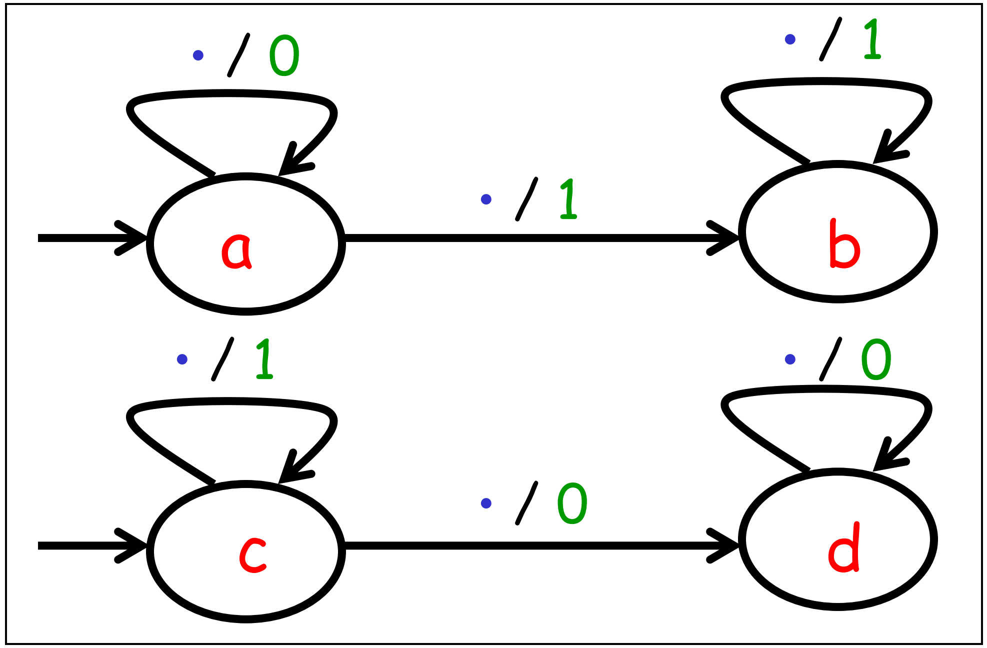
M



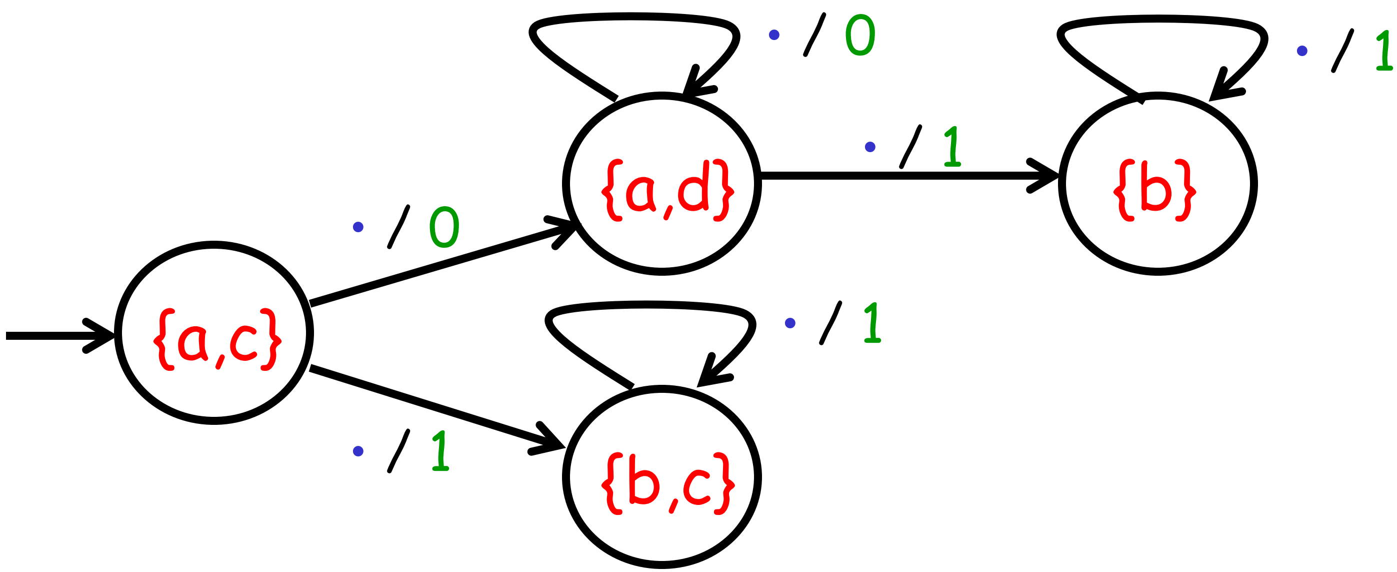
det(M)



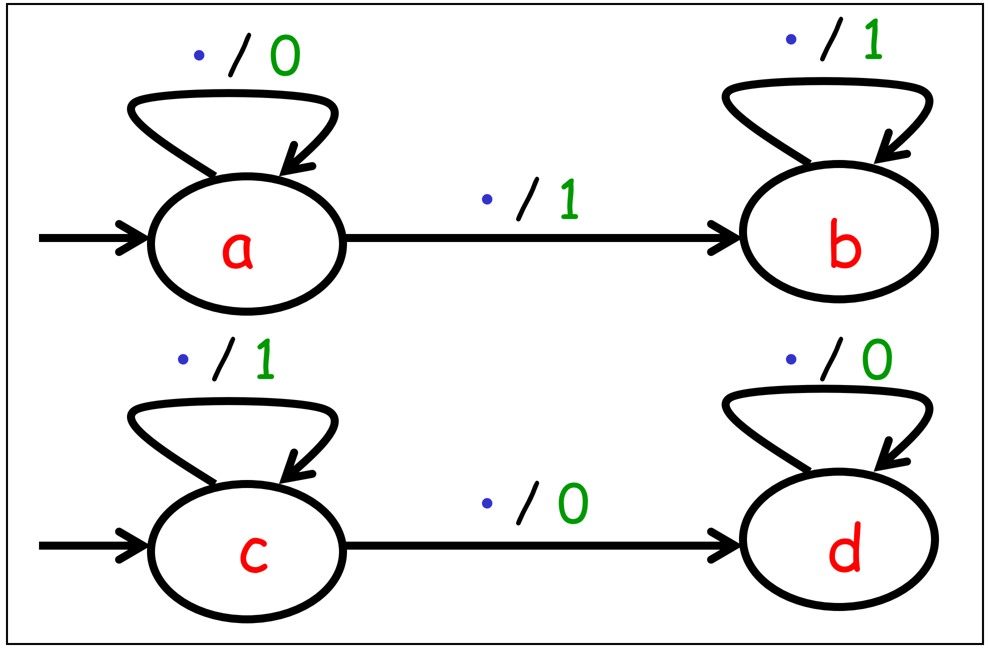
M



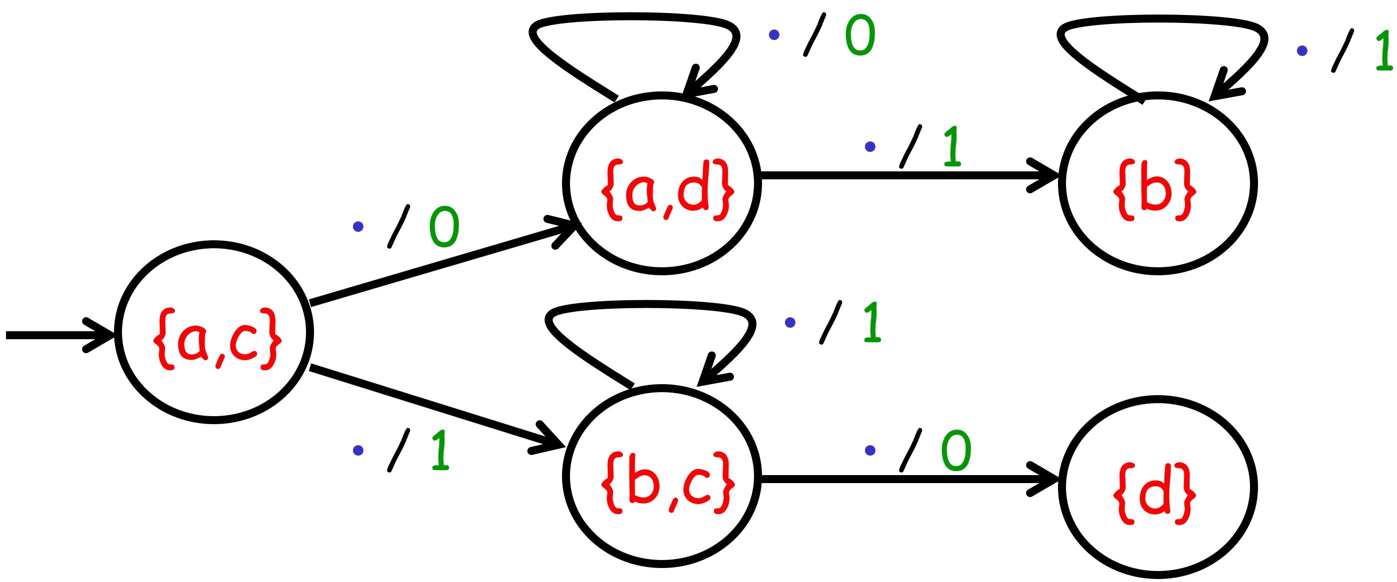
det(M)



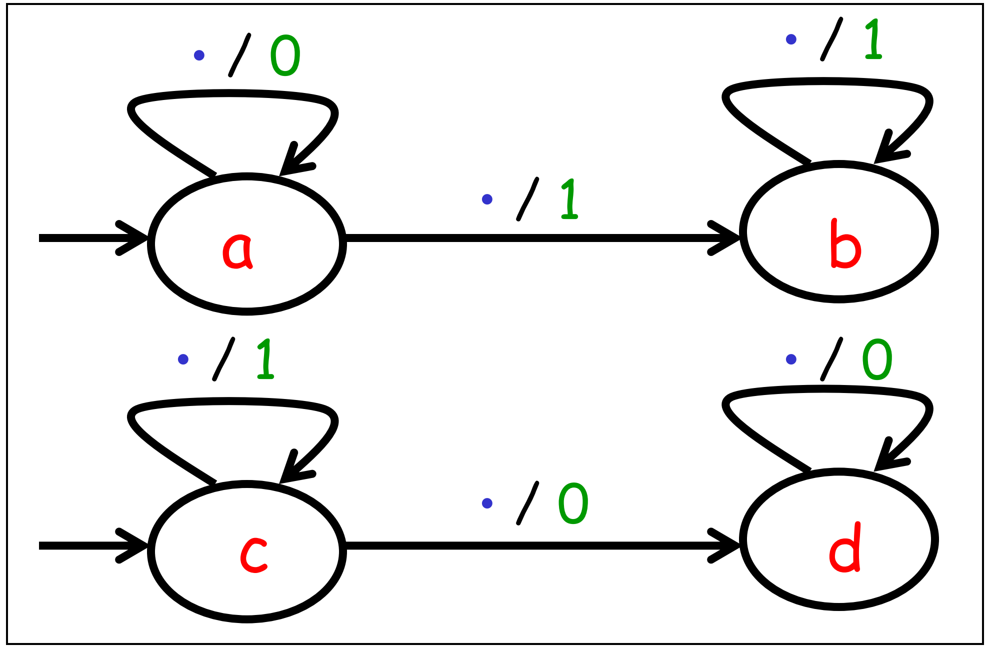
M



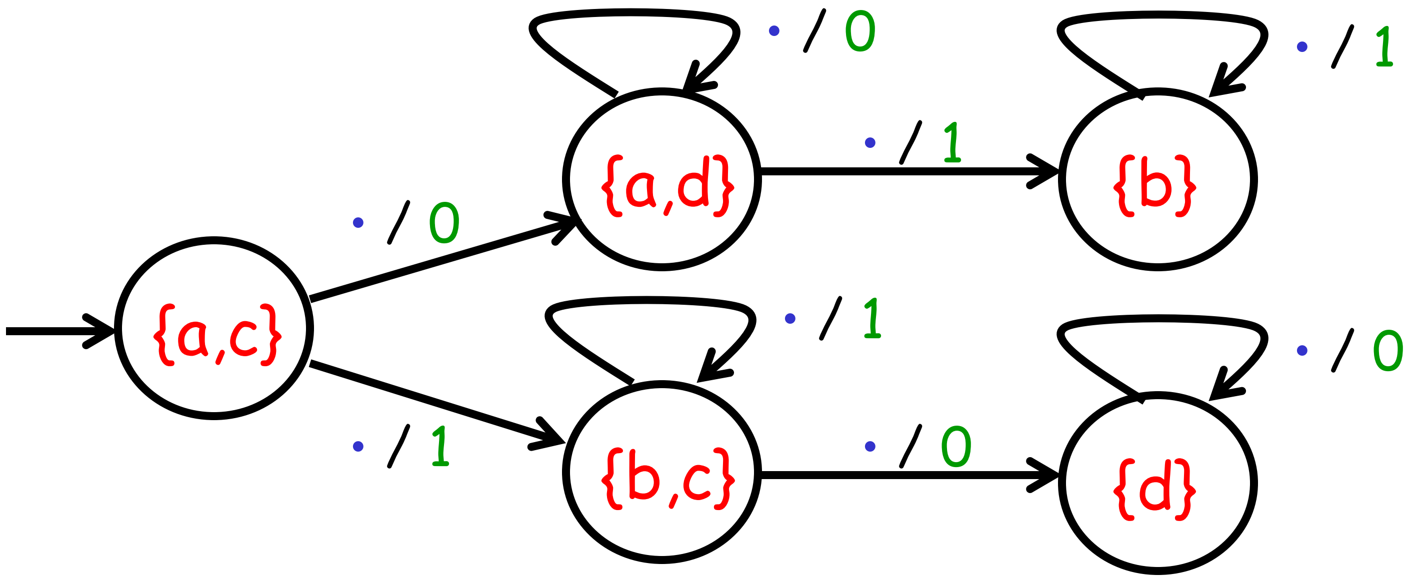
det(M)



M



det(M)



For a given reactive system S:

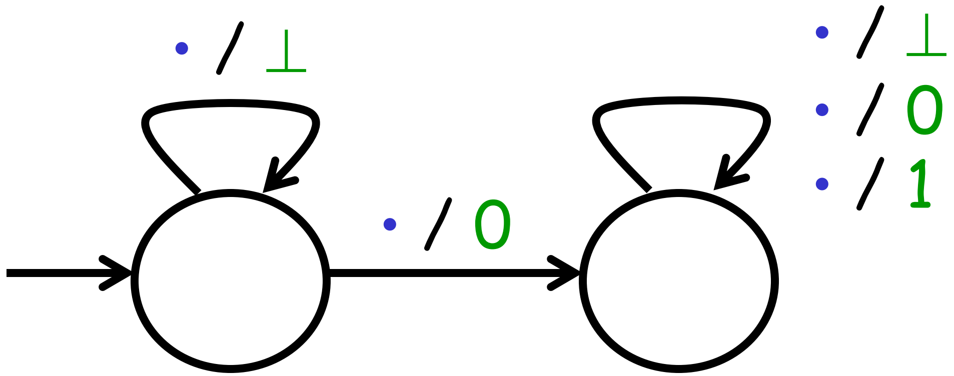
1. If there is a nondeterministic state machine that implements S with n states, then there is an output-deterministic state machine that implements S with 2^n states.

   [Subset construction]

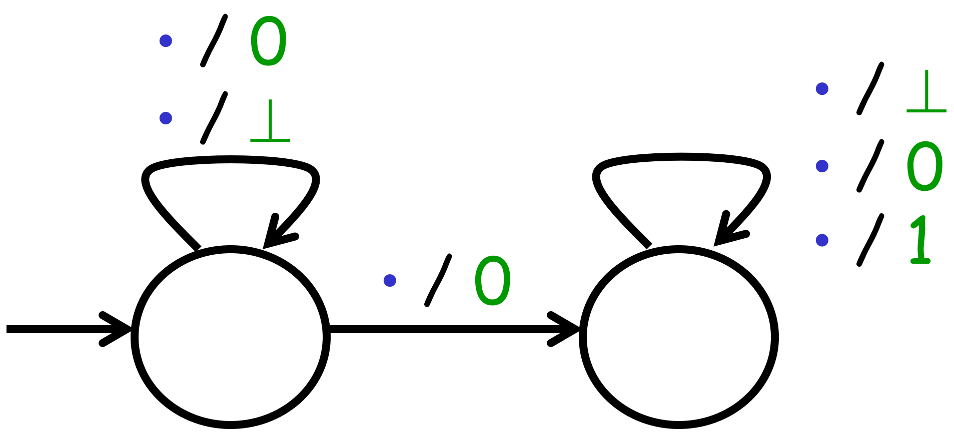
2. There may not be an output-deterministic state machine that implements S with fewer than 2^n states.

   [Homework 5, Problem C]

3. There may not be a unique nondeterministic state machine that implements S with the fewest states.



equivalent but
not isomorphic to



So what does minimization do for nondeterministic state machines ?

The Nondeterministic Minimization Algorithm

Input : nondeterministic state machine M

Output : minimize (M), the state machine with the fewest states that is bisimilar to M

(the result is unique up to renaming of states)

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

iff

A1. $\forall\, p \in possibleInitialStates[M1]$,

$\exists\, q \in possibleInitialStates[M2]$, $(p, q) \in B$, and

A2. $\forall\, p \in States[M1]$, $\forall\, q \in States[M2]$,

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

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

$(p', q') \in B$, and

and

B1. $\forall\, q \in \text{possibleInitialStates}\,[M2]\,,$

$\exists\, p \in \text{possibleInitialStates}\,[M1],\ (\,p, q\,) \in B,$ and

B2. $\forall\, p \in \text{States}\,[M1]\,,\ \forall\, q \in \text{States}\,[M2]\,,$

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

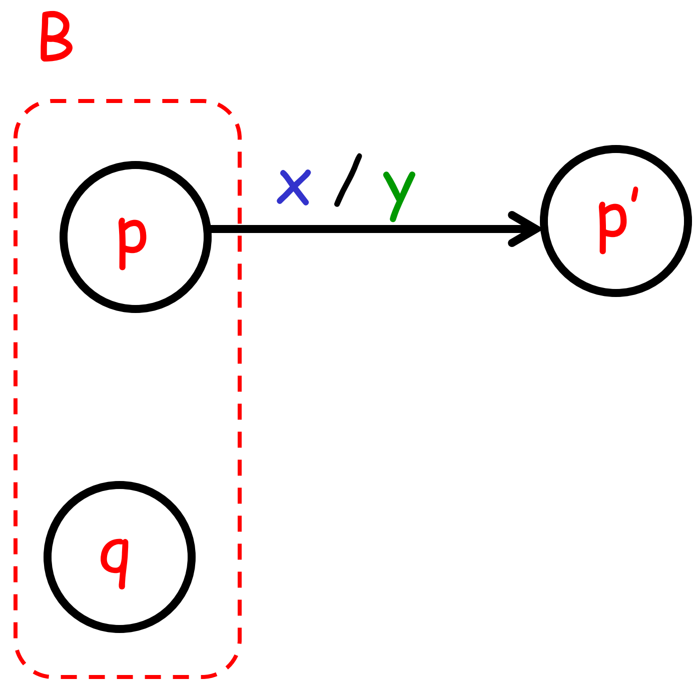
then $\forall\, x \in \text{Inputs}\,,\ \forall\, y \in \text{Outputs}\,,\ \forall\, q' \in \text{States}\,[M2]\,,$

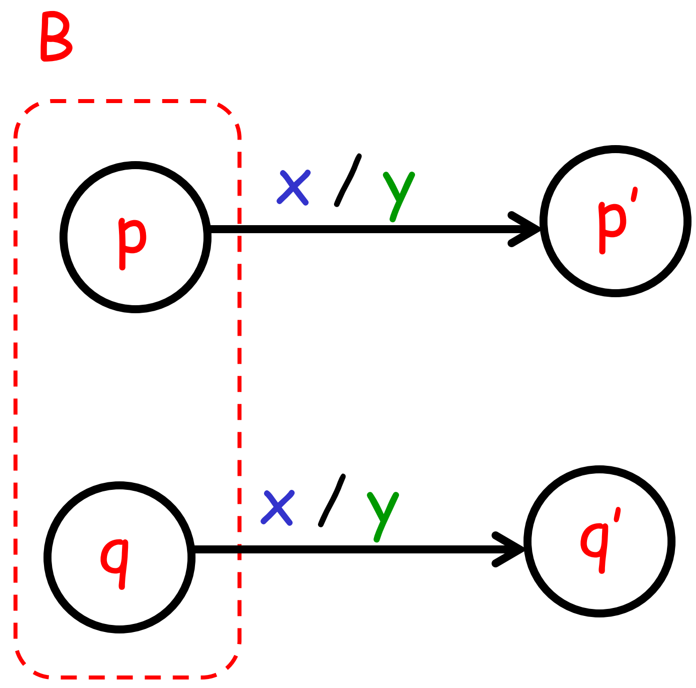
if $(\,q', y\,) \in \text{possibleUpdates}\,[M2]\,(\,q, x\,)$

then $\exists\, p' \in \text{States}\,[M1]\,,$

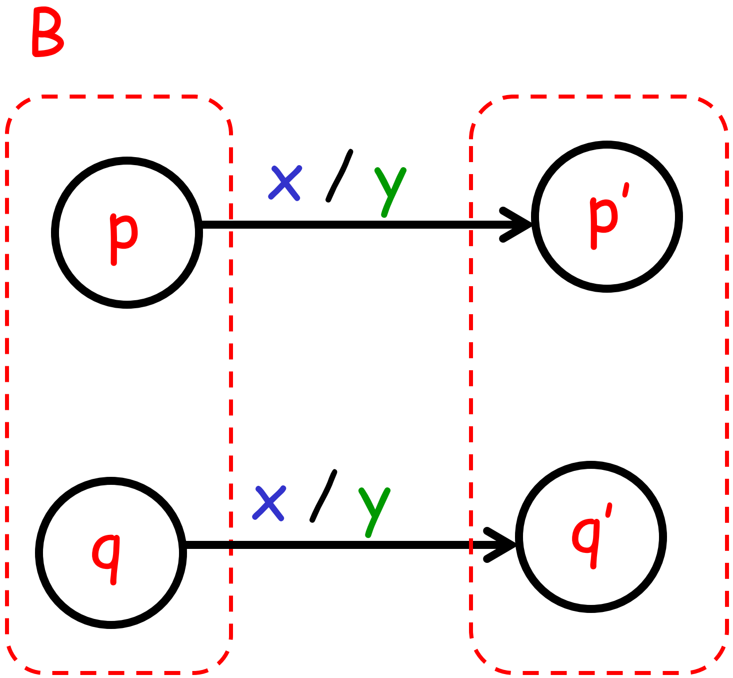
$(\,p', y\,) \in \text{possibleUpdates}\,[M1]\,(\,p, x\,)$ and

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

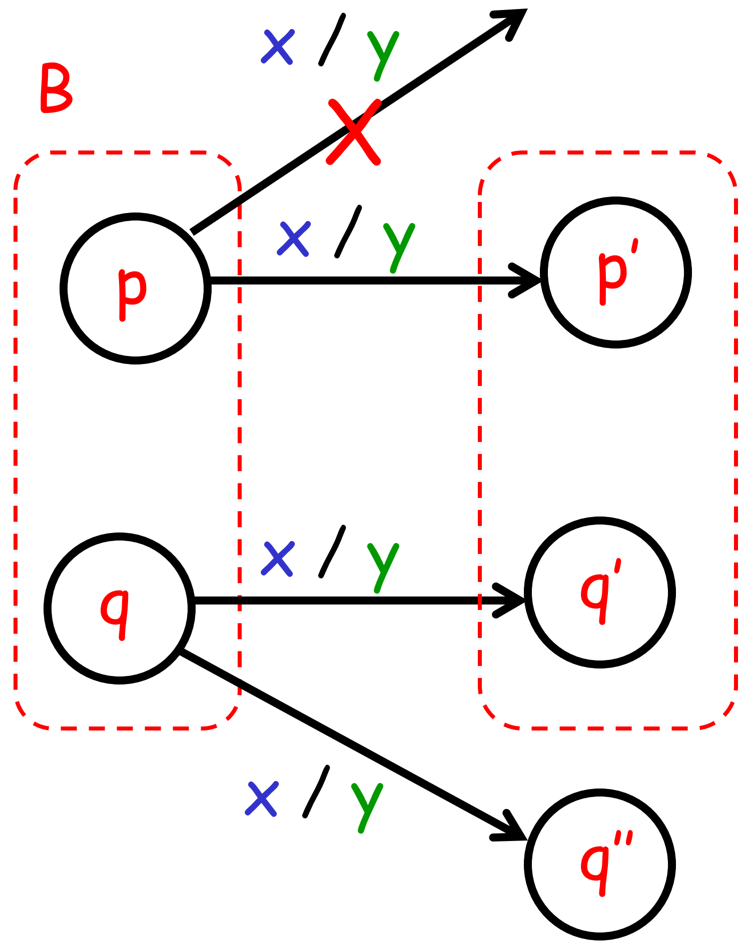

B
p
q
x / y
p'



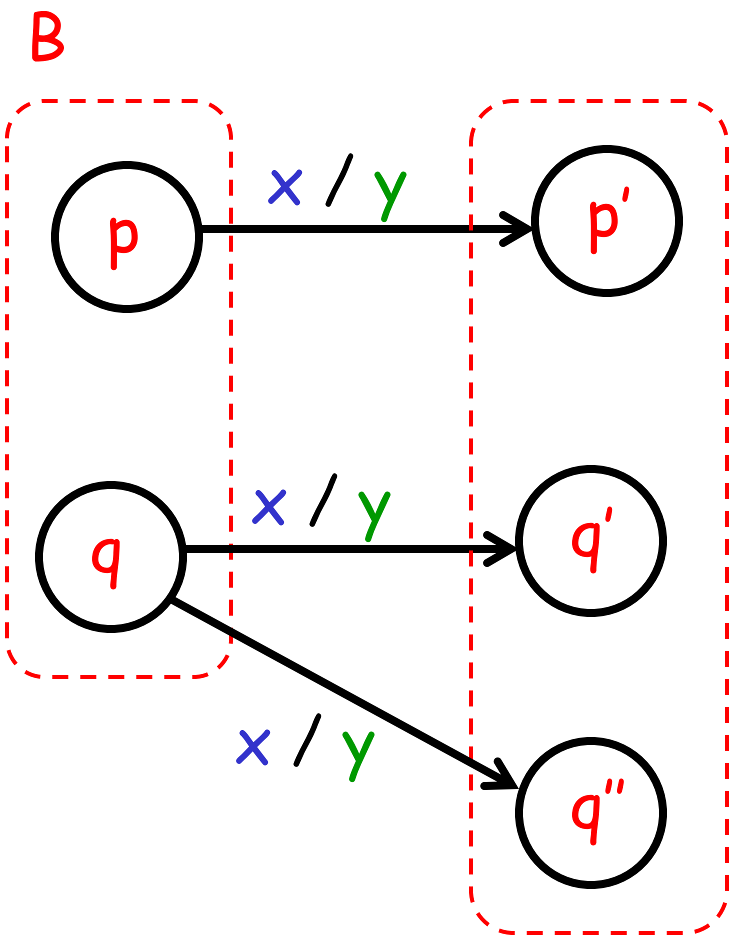
B
p
x / y
p'
q
x / y
q'



B
p
x / y
p'
q
x / y
q'



B
x / y
x / y
x / y
x / y
p
q
p'
q'
q''


B
p
x / y
p'
q
x / y
q'
x / y
q''

For deterministic state machines M1 and M2,

M1 is equivalent to M2

$\Downarrow \quad \Uparrow$

M1 and M2 are bisimilar.

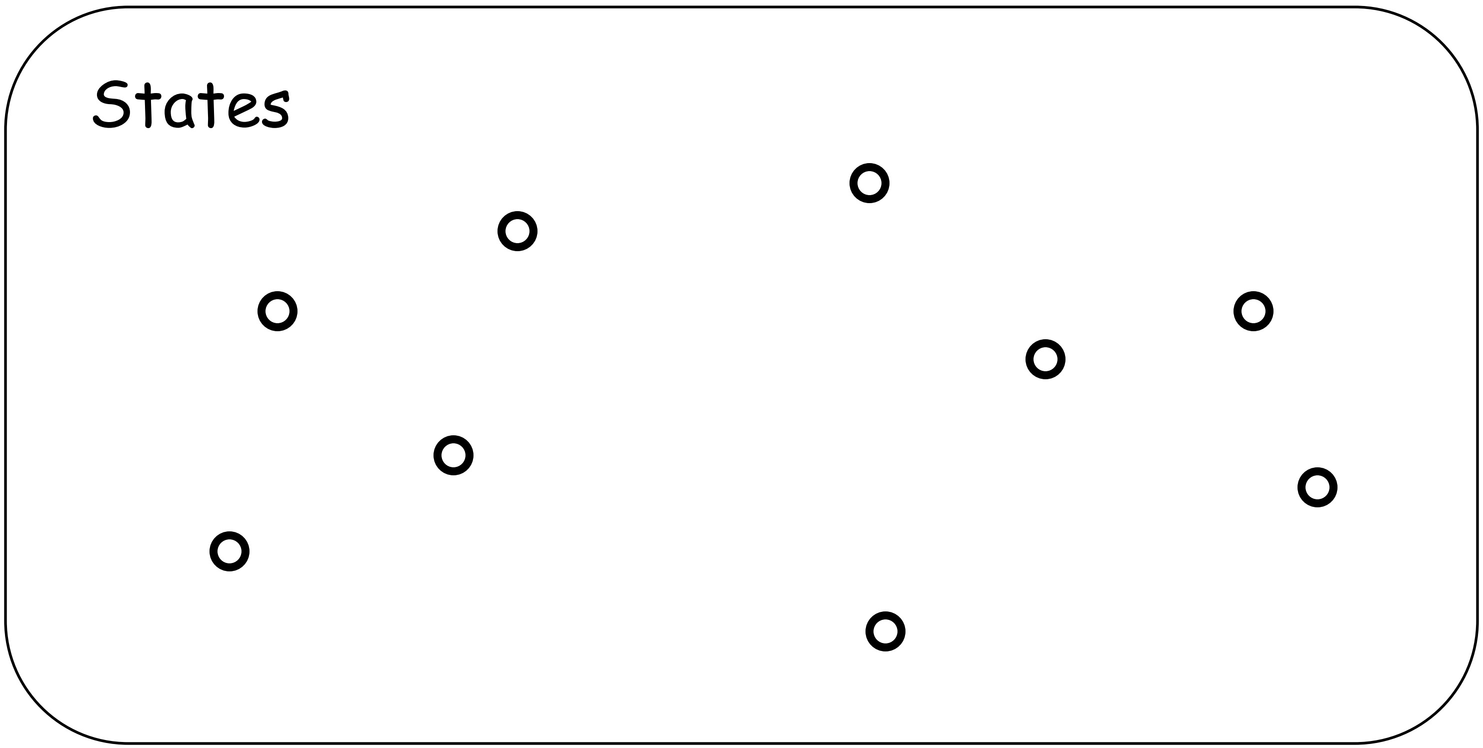
For nondeterministic state machines M1 and M2,

M1 is equivalent to M2

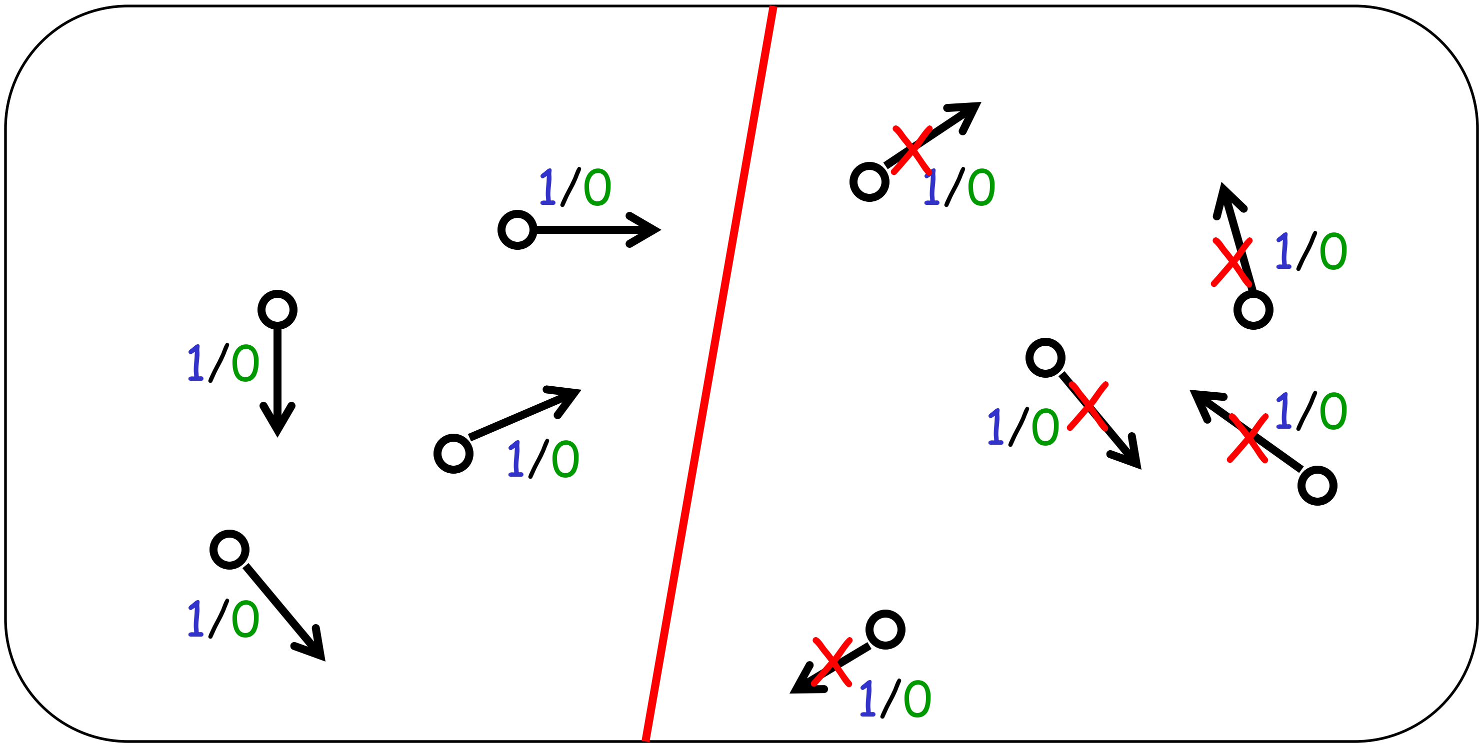
$\not\Downarrow \quad \Uparrow$

M1 and M2 are bisimilar.

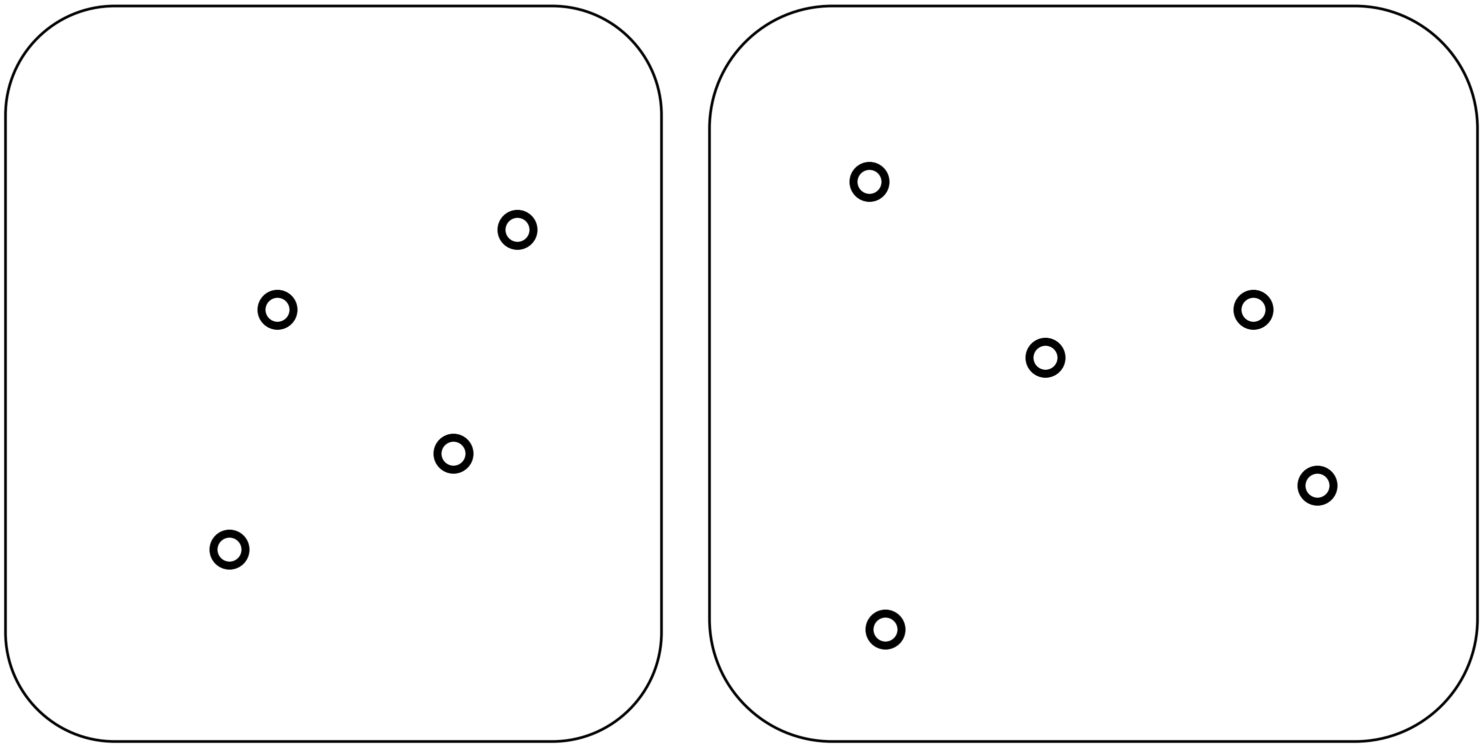
The Nondeterministic Minimization Algorithm



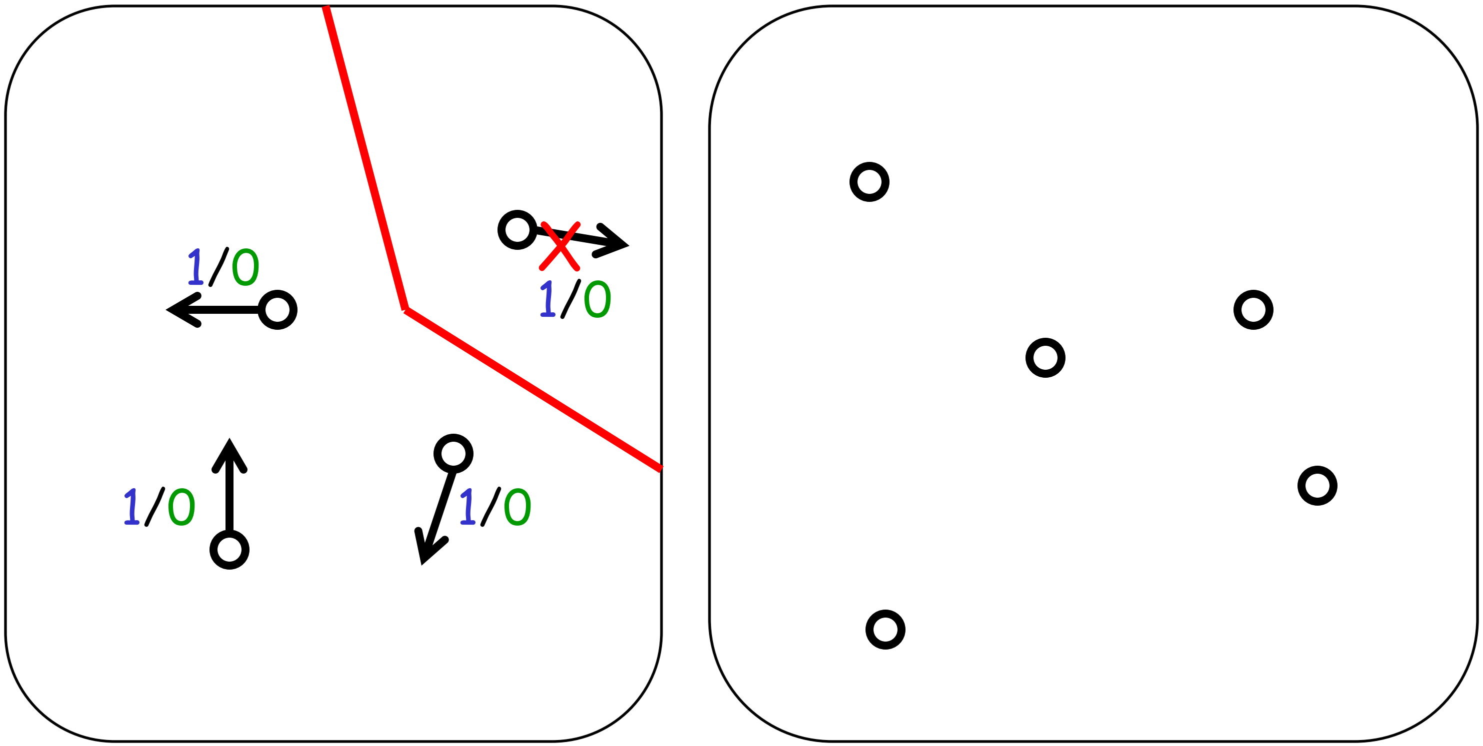
The Nondeterministic Minimization Algorithm



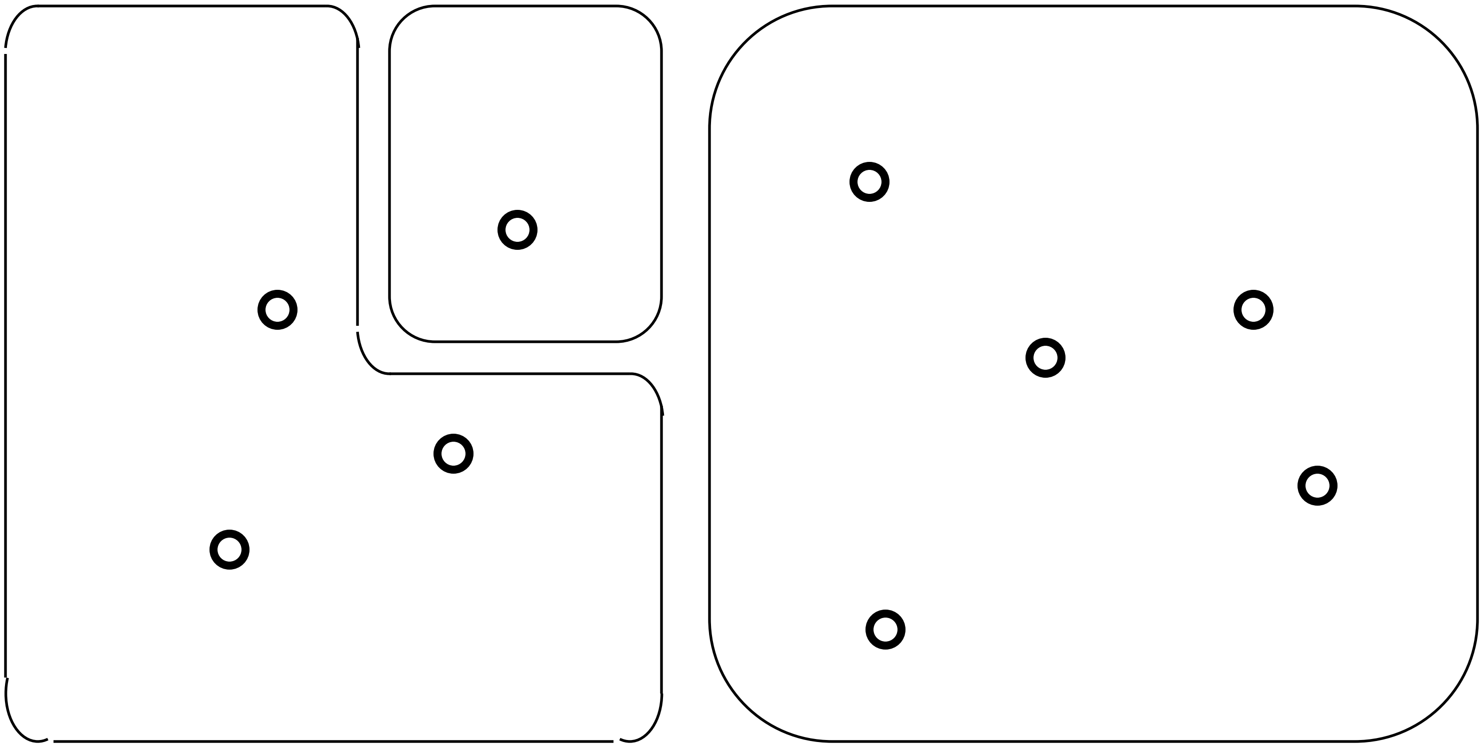
The Nondeterministic Minimization Algorithm



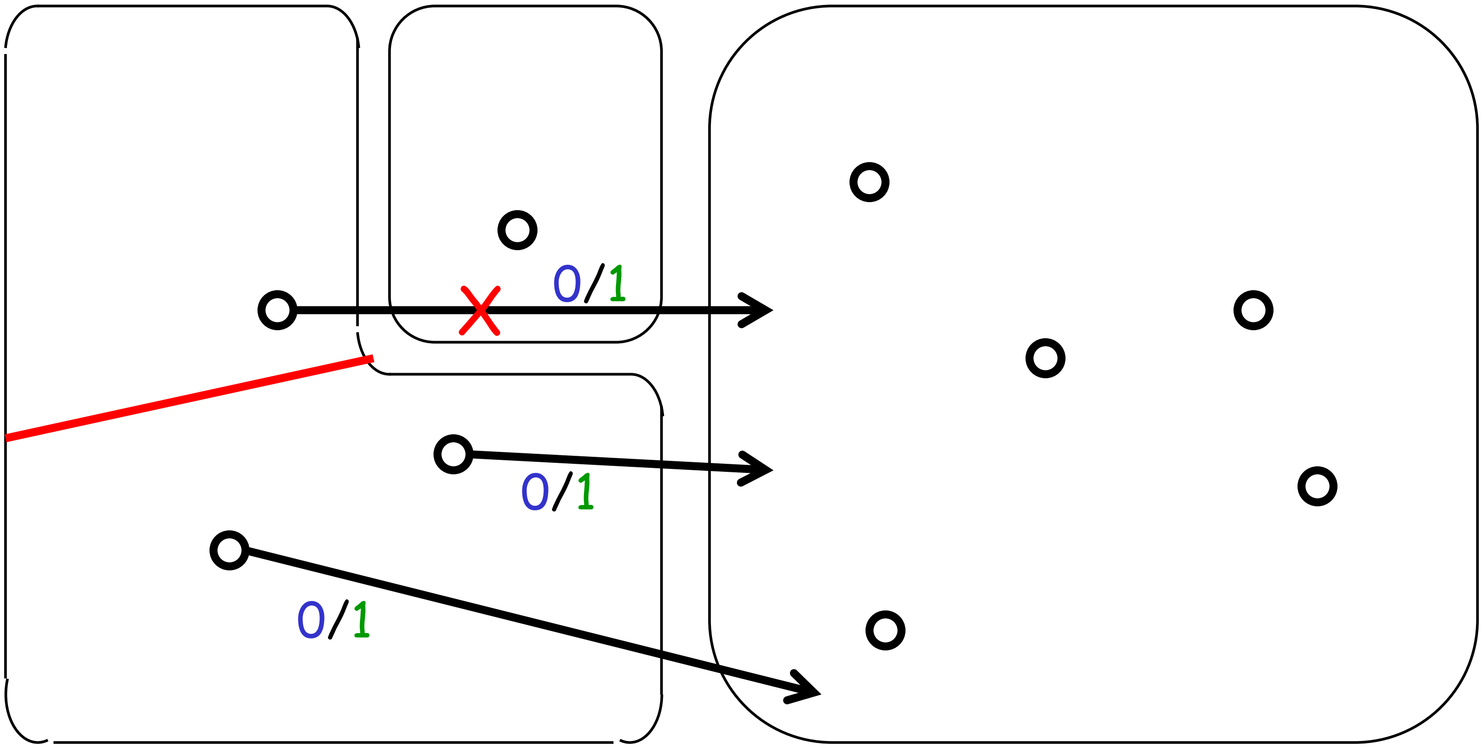
The Nondeterministic Minimization Algorithm



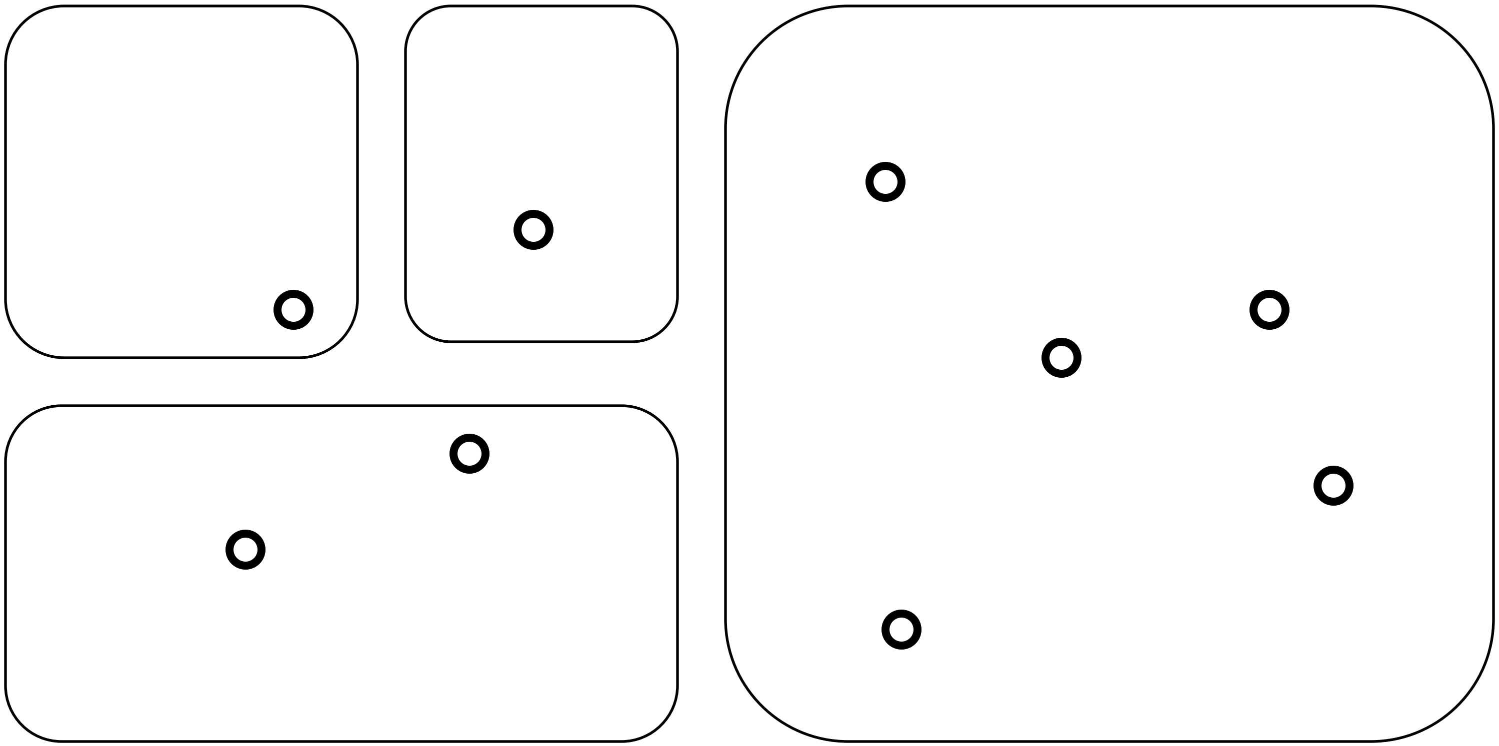
The Nondeterministic Minimization Algorithm



The Nondeterministic Minimization Algorithm



The Nondeterministic Minimization Algorithm



The Nondeterministic Minimization Algorithm

1. Let Q be set of all reachable states of M.
2. Maintain a set P of state sets:

   Initially let P = { Q }.

   Repeat until no longer possible: split P.
3. When done, every state set in P represents a single state of the smallest nondeterministic state machine bisimilar to M.

Split P

If there exist

- two state sets $R \in P$ and $R' \in P$
- two states $r1 \in R$ and $r2 \in R$
- an input $x \in Inputs$
- an output $y \in Outputs$

such that

$\exists\, r' \in R',\ (r', y) \in possibleUpdates(r1, x)$ and

$\forall\, r' \in R',\ (r', y) \notin possibleUpdates(r2, x)$

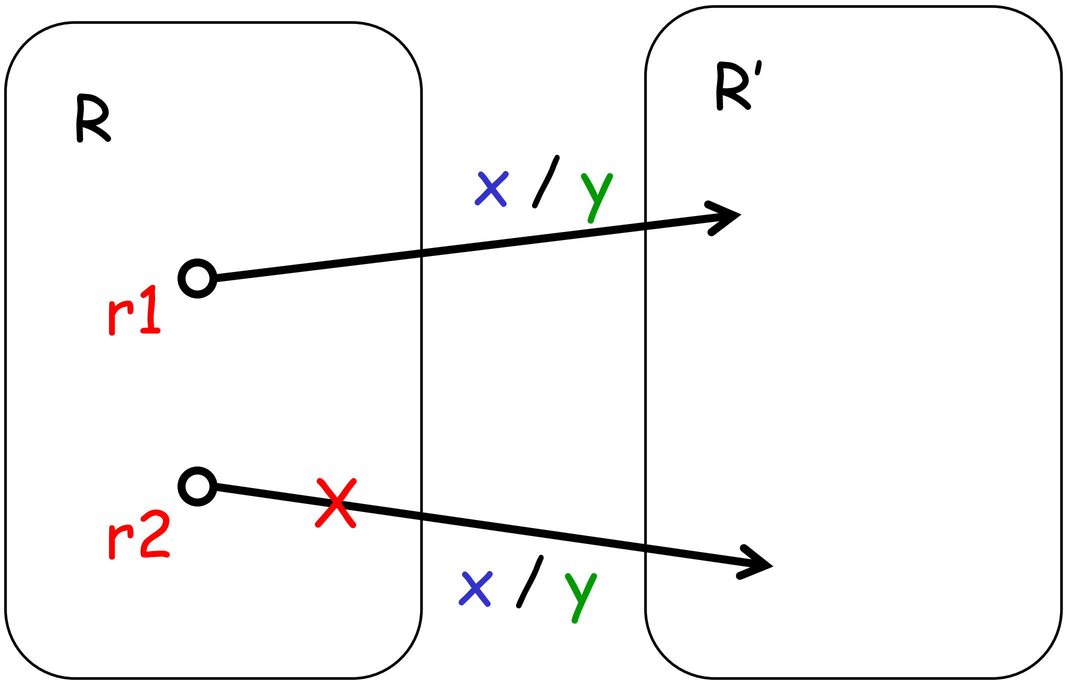
then

let $R1 = \{ r \in R \mid \exists\, r' \in R',\ (r', y) \in possibleUpdates(r, x) \}$;

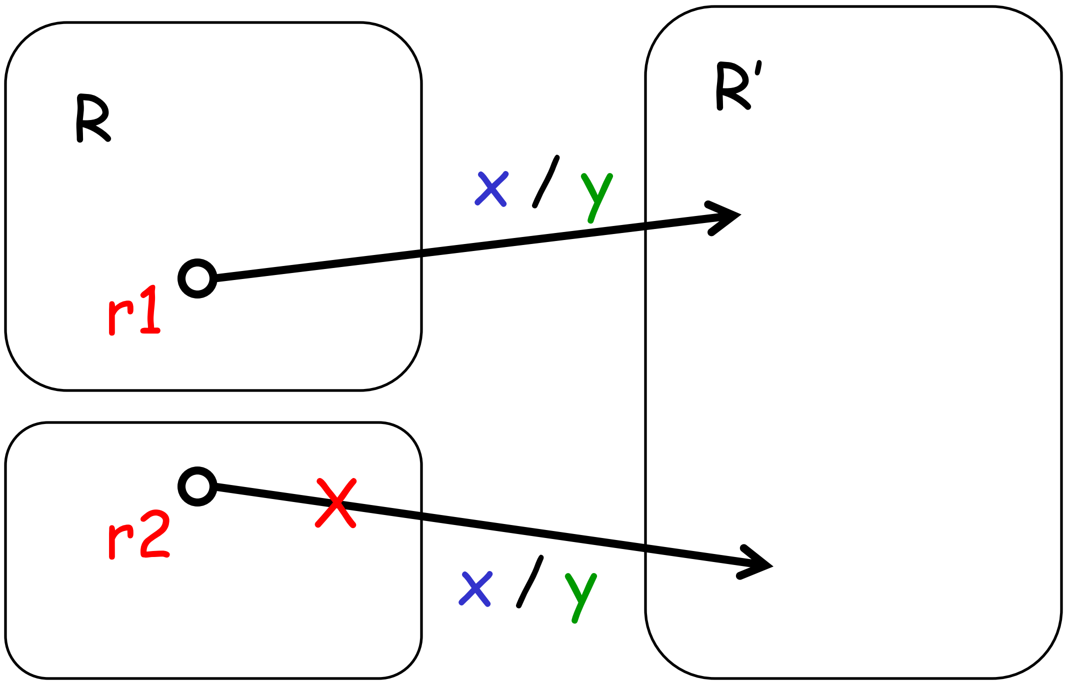
let $R2 = R \setminus R1$;

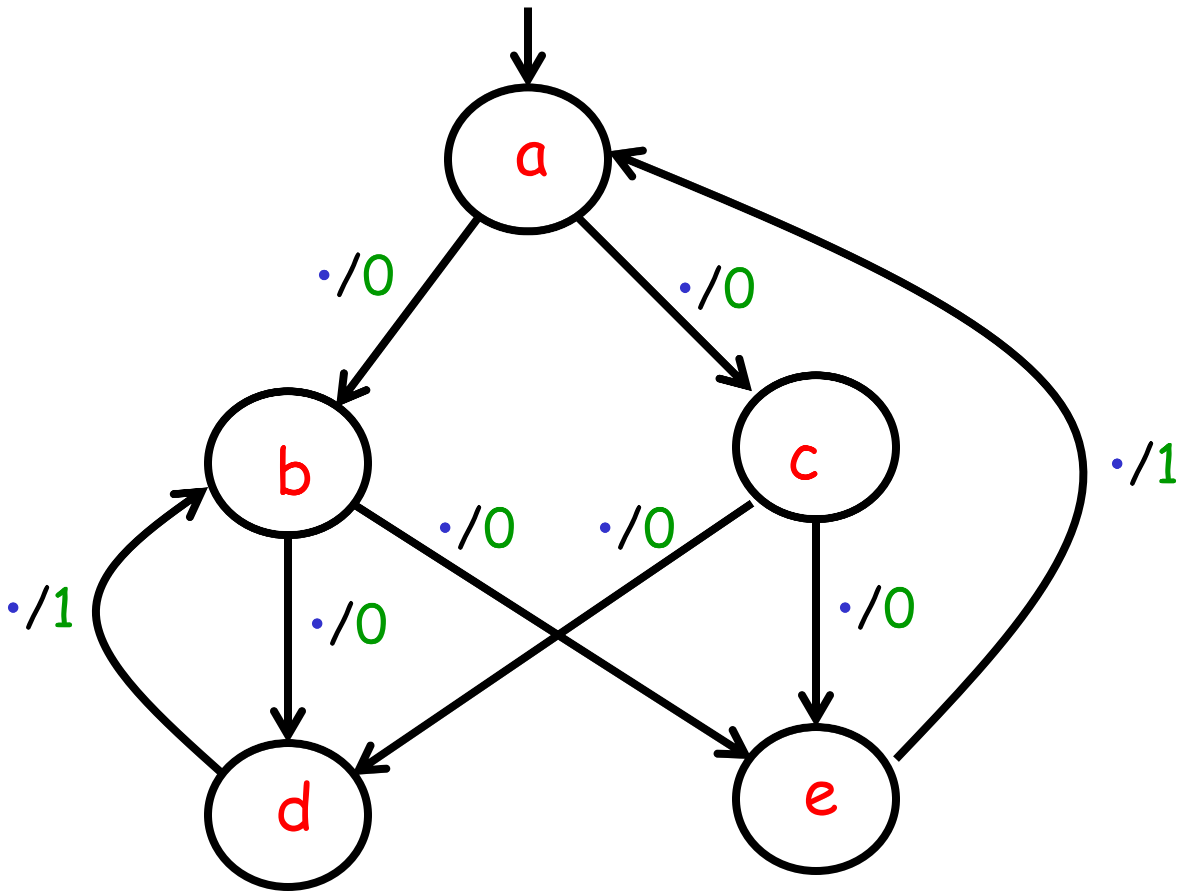
let $P = (P \setminus \{R\}) \cup \{R1, R2\}$.

Split

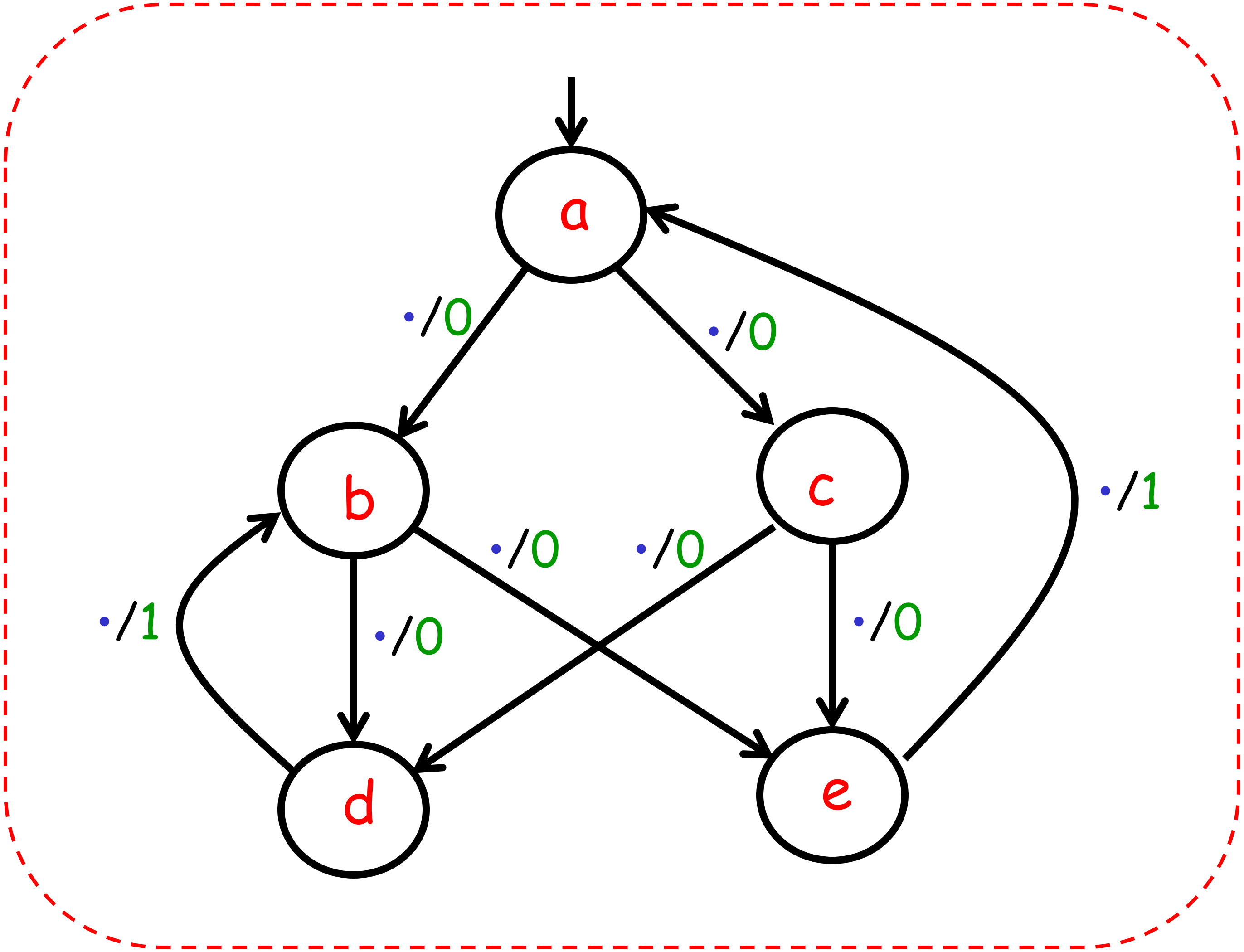


Split

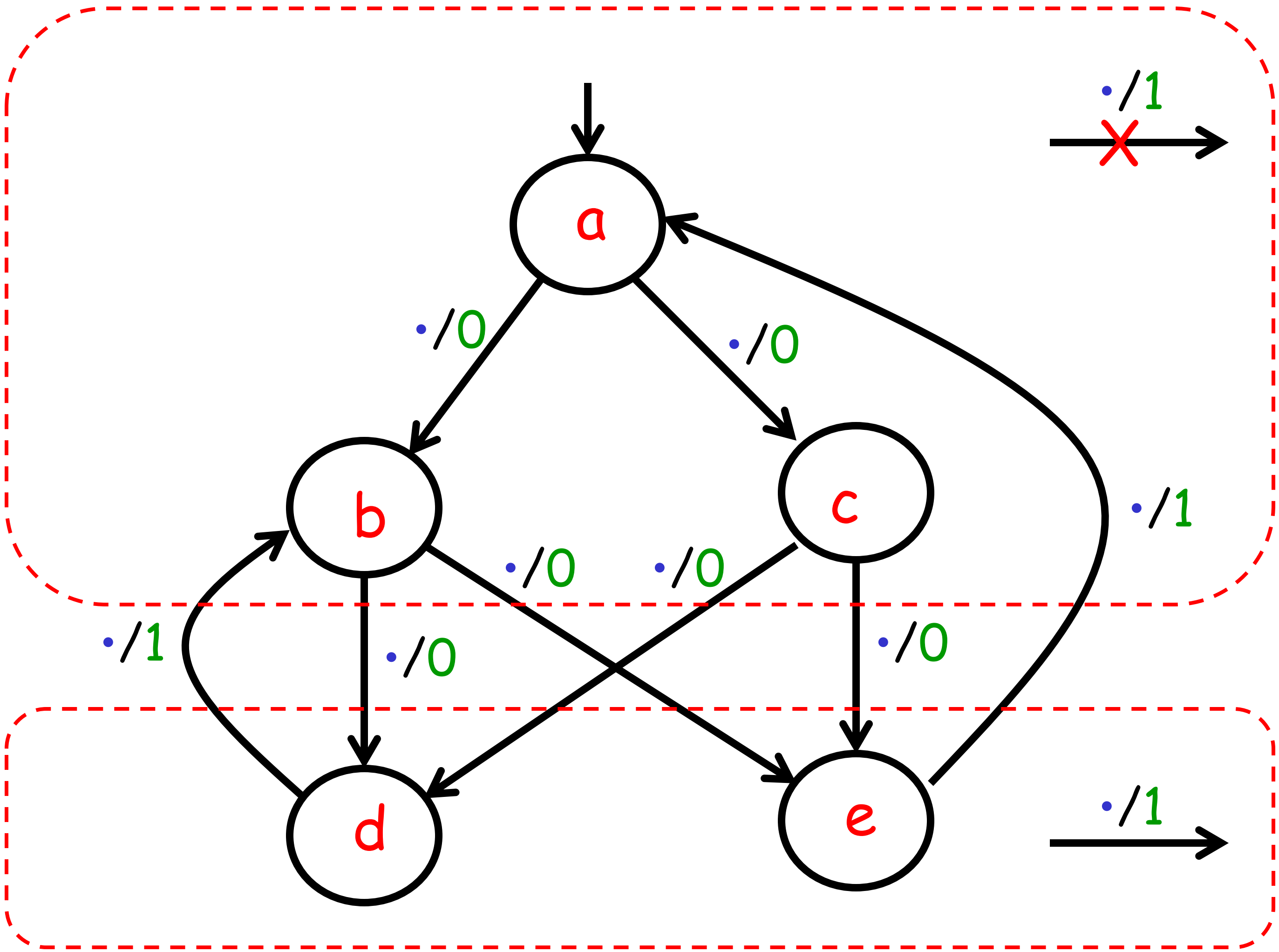




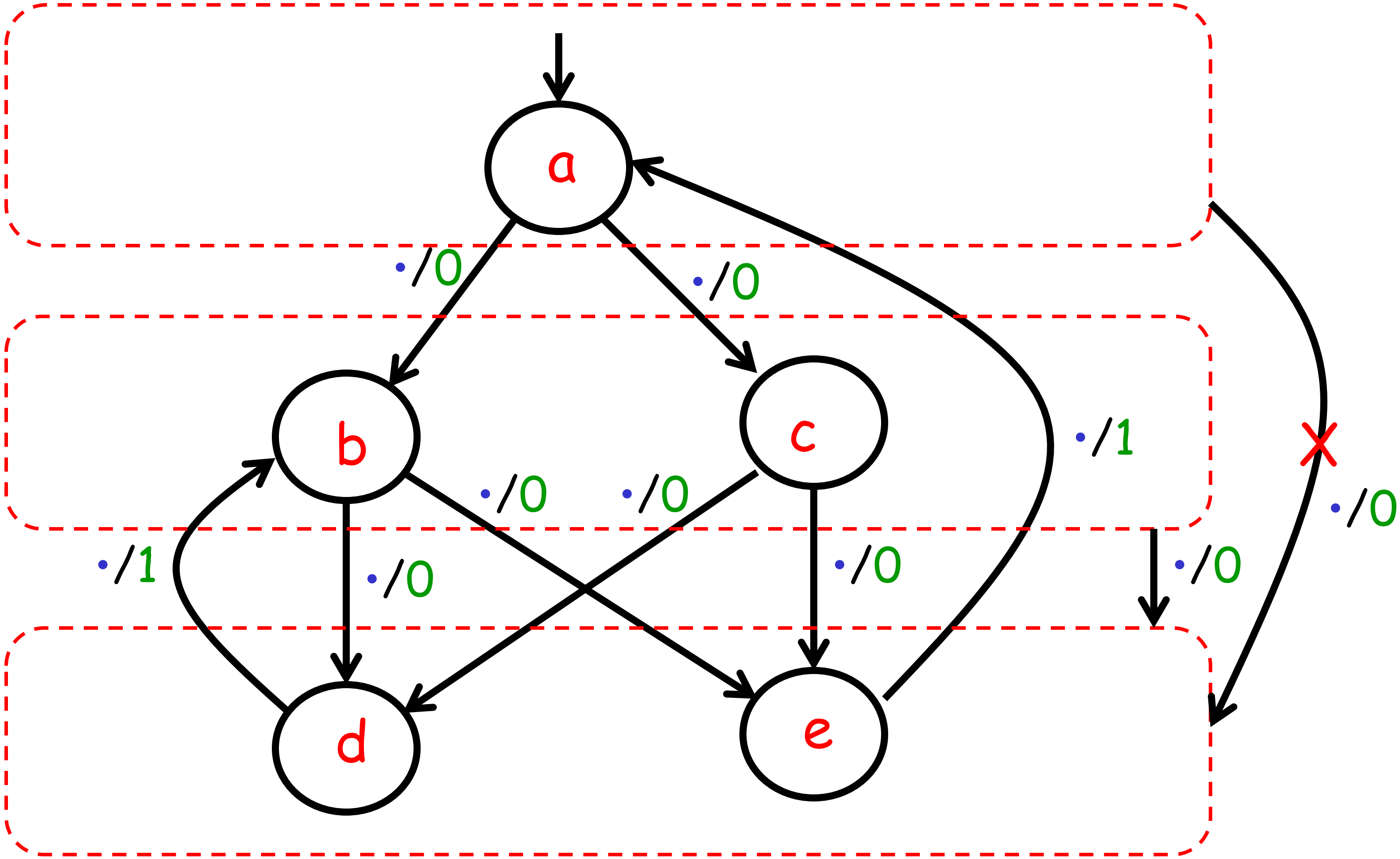
M



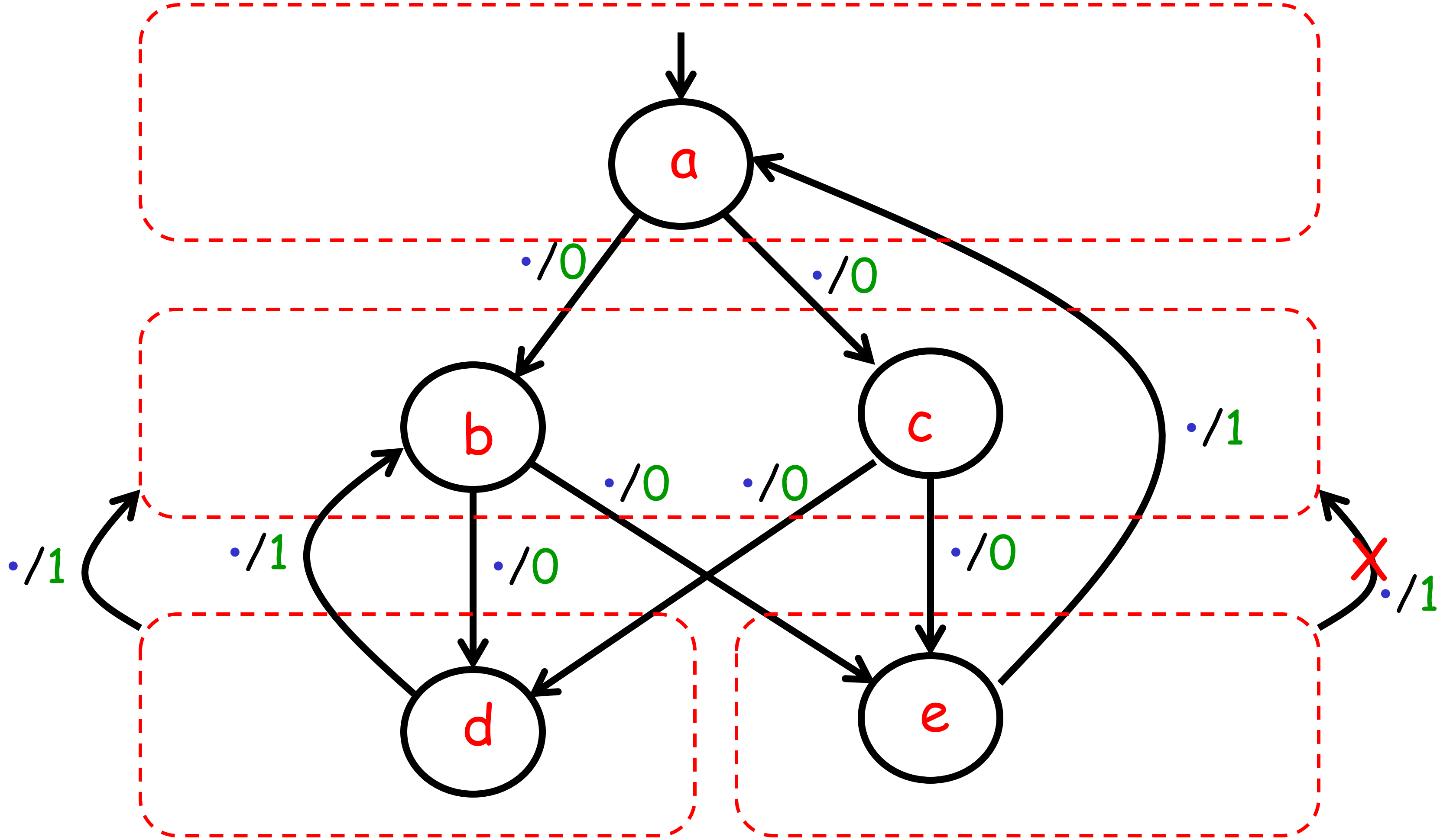
M



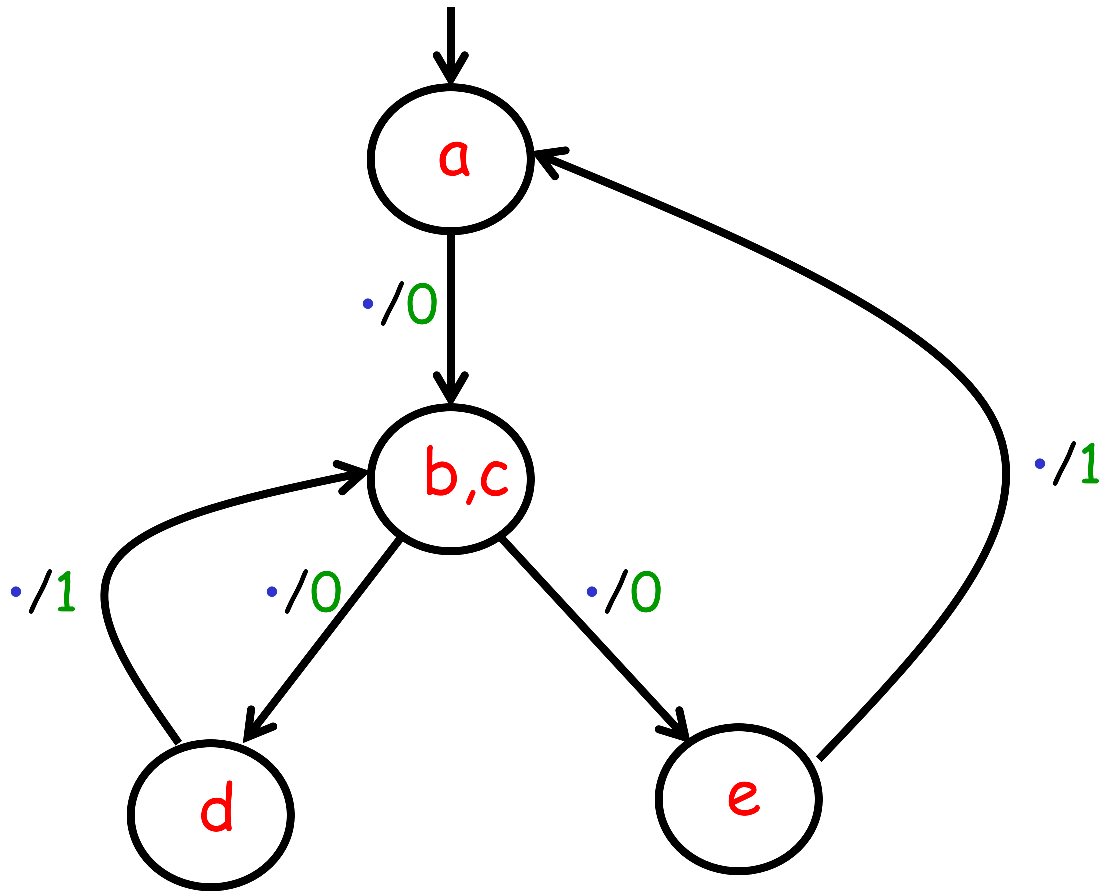
M



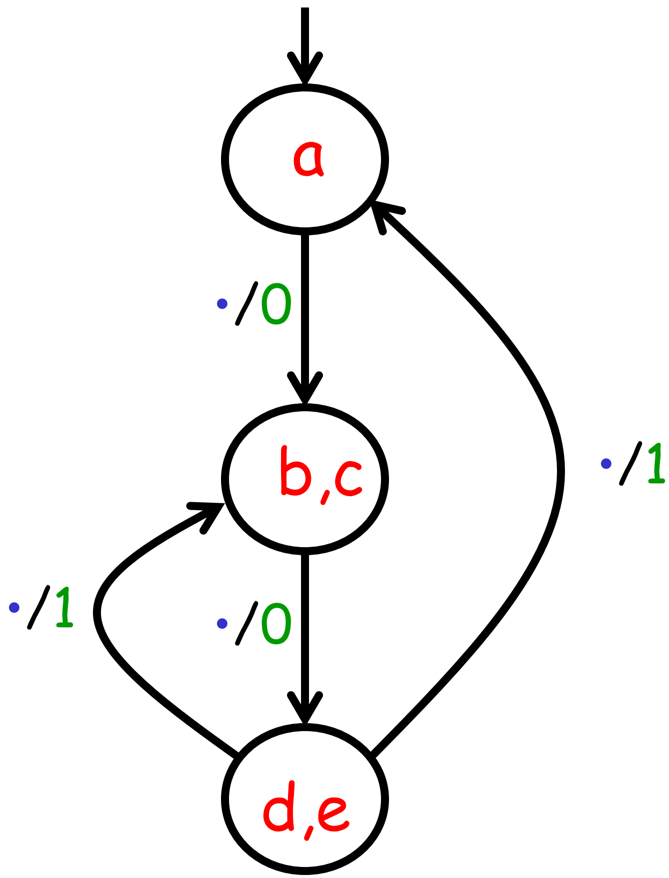
M



M



Minimal state machine bisimilar to M



Minimal state machine equivalent to M

(in general, this is difficult to find)