Week 1- Introduction to model checking

B. Srivathsan

Chennai Mathematical Institute

NPTEL-course

July - November 2015

Course overview

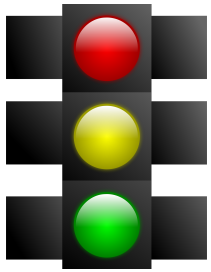
What are we **interested** in?

What are we **interested** in?

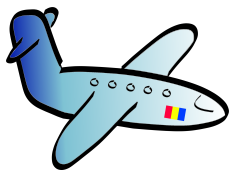
Software Controllers

Code that controls the working of an

Information and Communication (ICT) device



Traffic lights controller

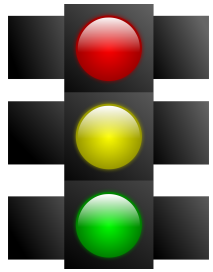


Flight control

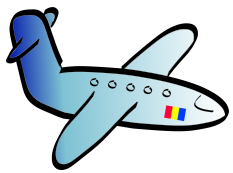
Automatic gear control

ATM

Pacemaker



Traffic lights controller



Flight control

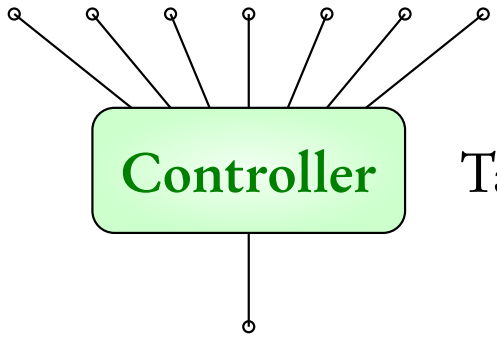
Automatic gear control

ATM

Pacemaker

Lifts, Automatic doors, Hardware circuits, Netbanking ... and many more!

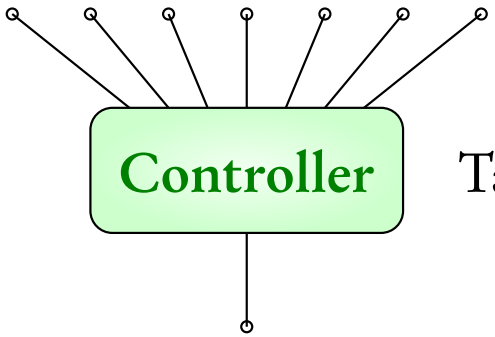
Listens to **various inputs**



Takes certain **decisions**

Gives **output action**

Listens to **various inputs**



Takes certain **decisions**

Gives **output action**

Many **safety-critical** systems controlled by code

How **reliable** is the controlling code?

- **decision making** should be correct
- **all possible scenarios** should be considered

Bugs are costly

- **Intel's Pentium II processor:**

  Error in floating point division code (1994)

  - Loss of 475 million US dollars

- **Ariane 5 rocket:**

  Error in the control software (1996)

  - Crashed 36 seconds after launch

- **Therac-25 radiation therapy machine:**

  Error in control software (1985 - 1987)

  - Death of 6 patients due to radiation overdose

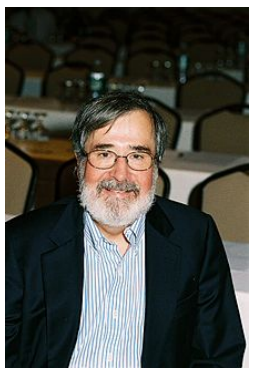
Goal: Make **low-defect** software controllers

Traditional testing **insufficient** for safety-critical systems

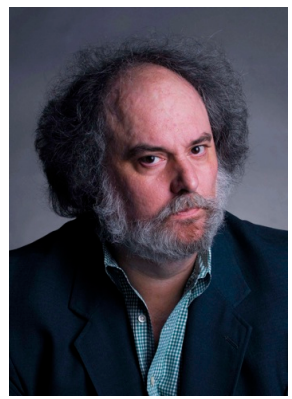
Goal: Make **low-defect** software controllers

Traditional testing **insufficient** for safety-critical systems

→ A new **verification technology** called **Model-checking**



Edmund Clarke

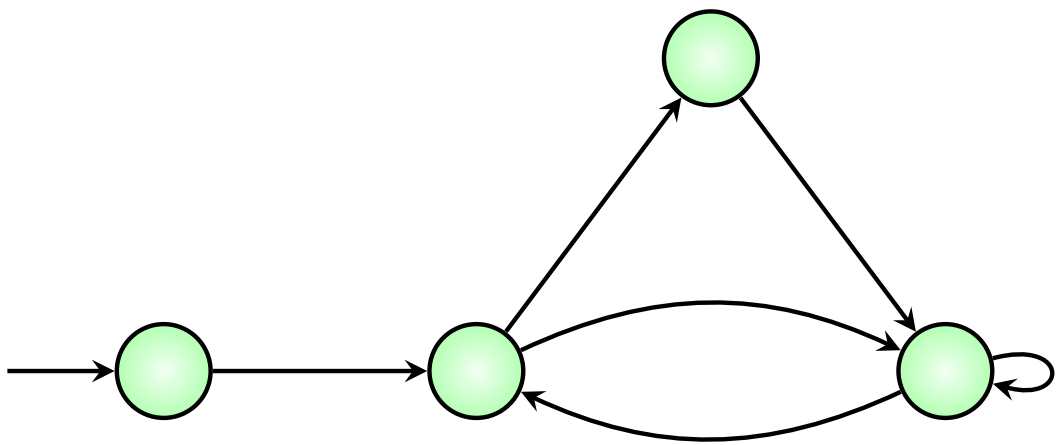


Allen Emerson

Joseph Sifakis

Model Checking

Uses **finite state machines** to model and **verify** controllers



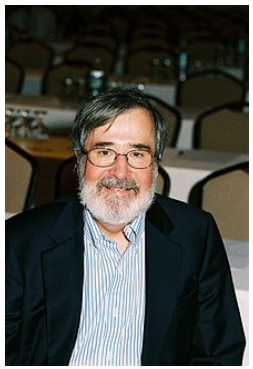
Some places where **Model Checking** technology is **used**

- *Hardware:* Intel, IBM, Synopsys
- *Avionics:* Rockwell Collins, Honeywell
- *Automobiles:* Toyota
- *Space:* NASA, European Space Agency
- *Others:* Microsoft Research, Tata, Mathworks

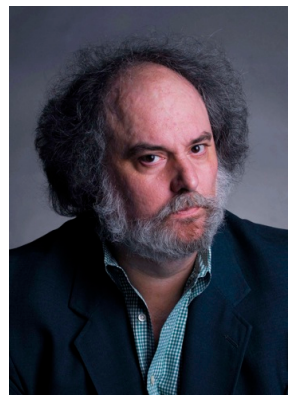
Some places where **Model Checking** technology is **used**

- *Hardware:* Intel, IBM, Synopsys
- *Avionics:* Rockwell Collins, Honeywell
- *Automobiles:* Toyota
- *Space:* NASA, European Space Agency
- *Others:* Microsoft Research, Tata, Mathworks

Backed by many **university groups** from all over the world!



Edmund Clarke



Allen Emerson

Joseph Sifakis

Turing Award'07 for their work on Model-checking

Why do this course?

- Various **industries adopting** model-checking into their design cycle
- Need engineers **qualified** in model-checking technology
- Scope for **higher studies**

In this course

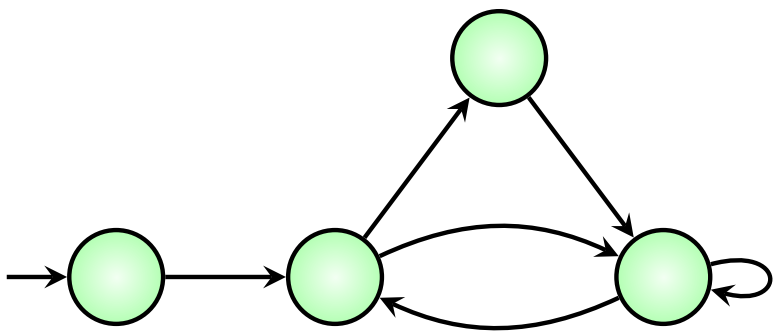
Introduction to **techniques and tools** used in Model-Checking

Book: Principles of Model Checking,
Christel Baier and Joost-Pieter Katoen, MIT Press (2008)

In this course

Introduction to **techniques and tools** used in Model-Checking

Book: Principles of Model Checking,
Christel Baier and Joost-Pieter Katoen, MIT Press (2008)



$$(\{q_1, q_2, q_3, q_4\}, \delta)$$

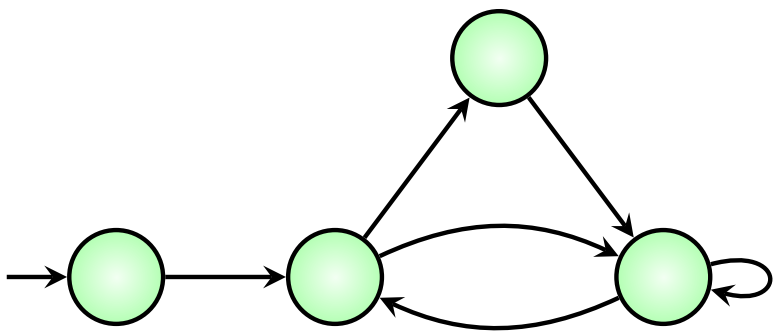
$$\delta(q_1) = q_2, \delta(q_2) = \{q_3, q_4\}$$

$$\delta(q_3) = q_4, \delta(q_4) = \{q_2, q_4\}$$

In this course

Introduction to **techniques and tools** used in Model-Checking

Book: Principles of Model Checking,
Christel Baier and Joost-Pieter Katoen, MIT Press (2008)



$$(\{q_1, q_2, q_3, q_4\}, \delta)$$

$$\delta(q_1) = q_2, \delta(q_2) = \{q_3, q_4\}$$

$$\delta(q_3) = q_4, \delta(q_4) = \{q_2, q_4\}$$

Bachelors/Masters in CS/IT/EEE/ECE **welcome!**

Hope you'll **enjoy** the course!

Week-1: Introduction to model checking

B. Srivathsan

Chennai Mathematical Institute

NPTEL-course

July - November 2015

Module 1:

Modeling code behaviour

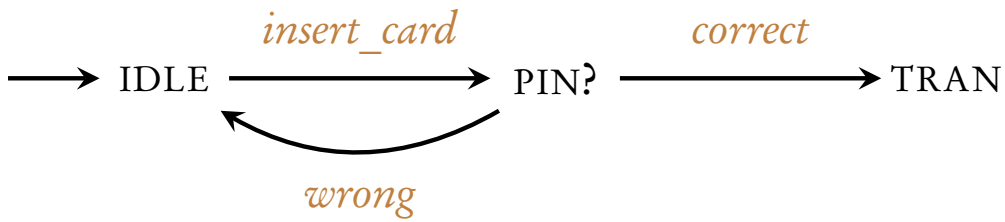

ATM

→ IDLE

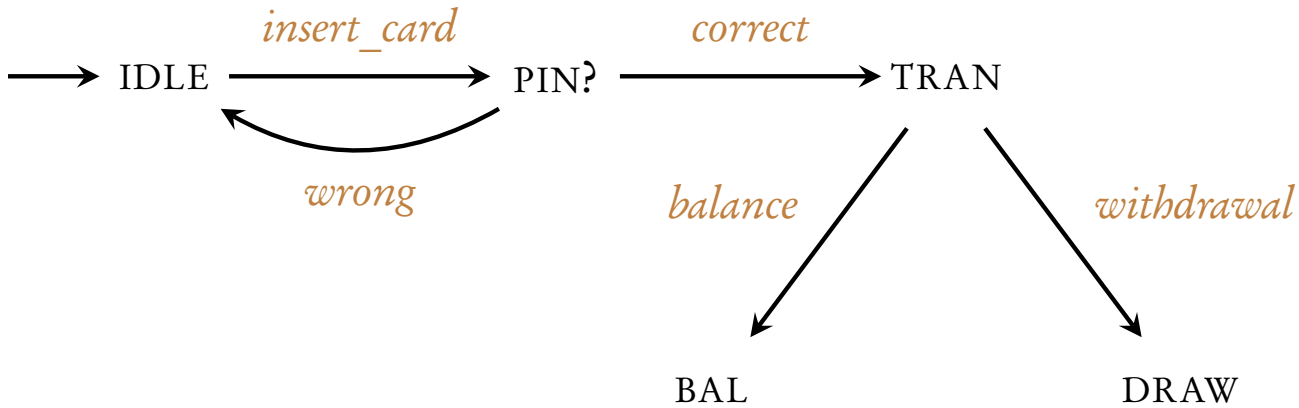

ATM


insert_card
IDLE
PIN?

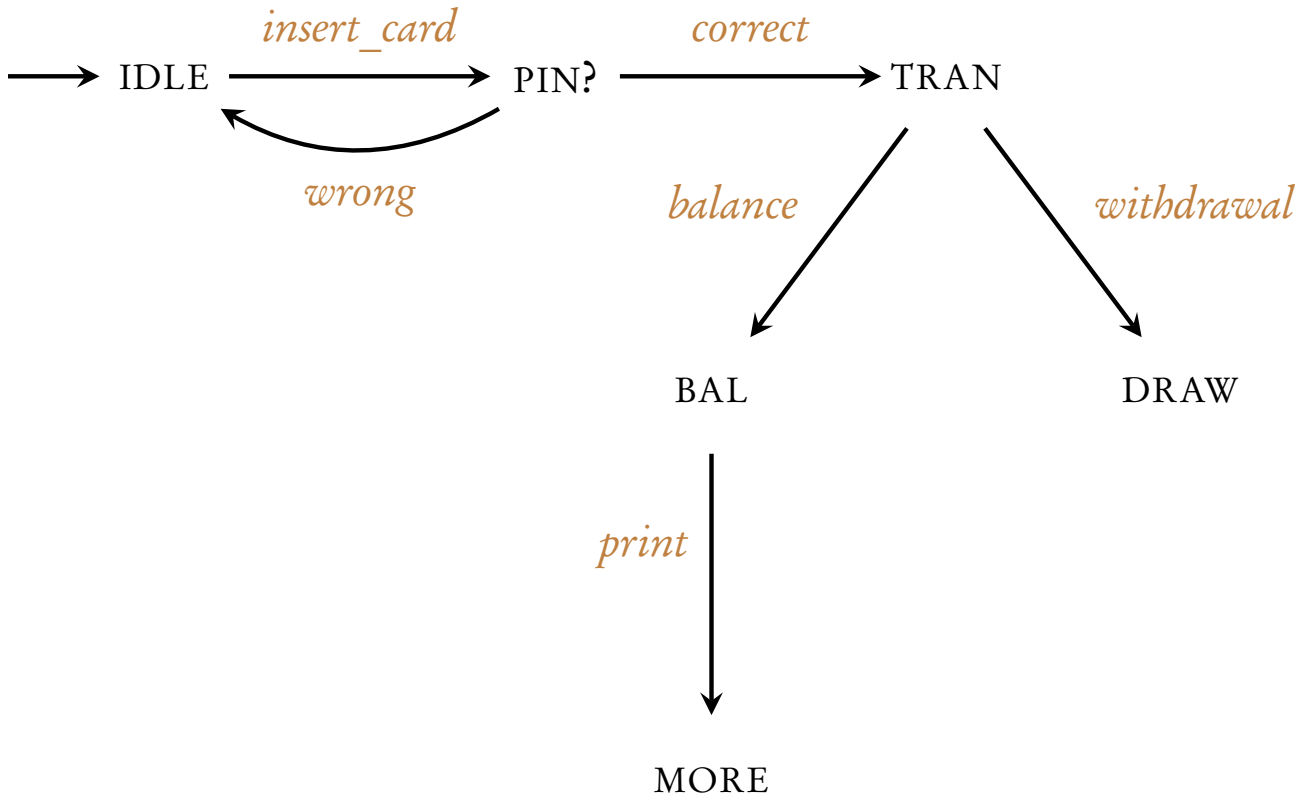

ATM


insert_card
correct
IDLE
PIN?
TRAN
wrong

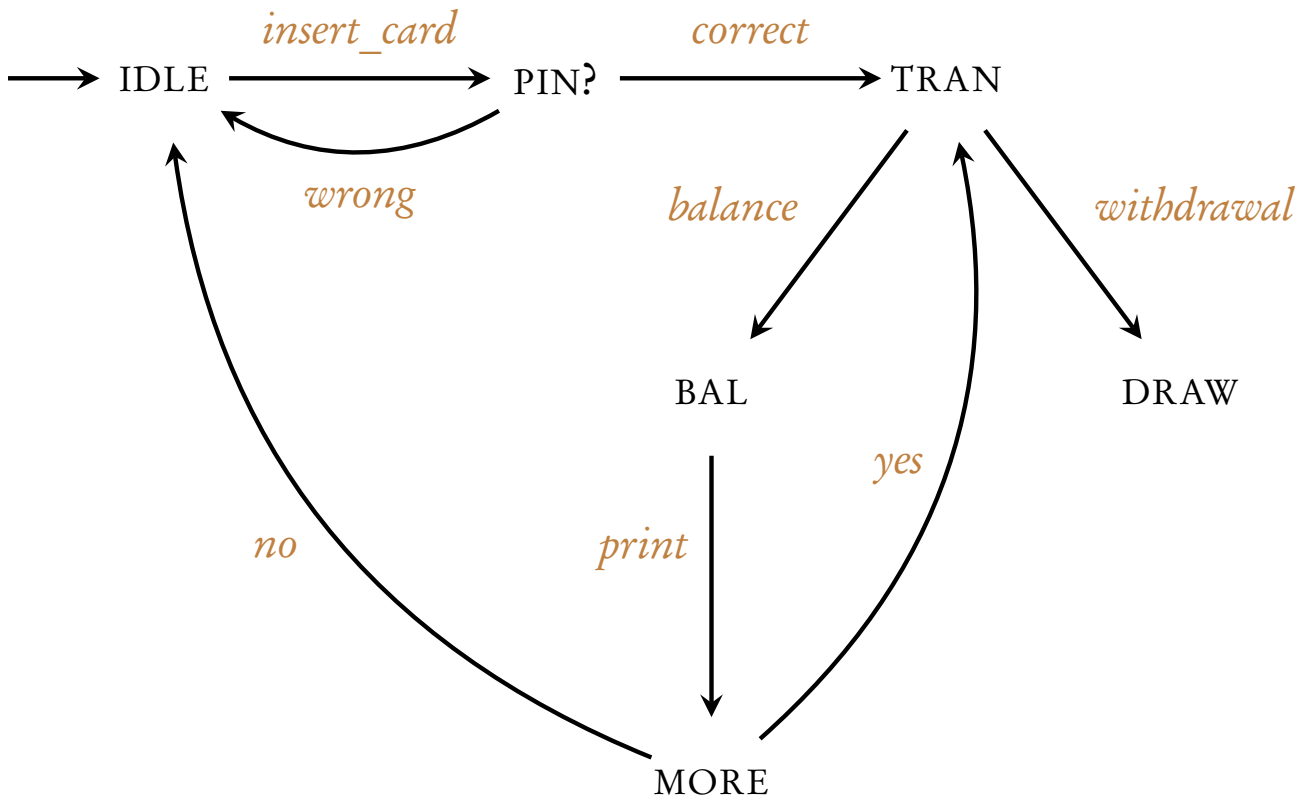

ATM


IDLE
insert_card
PIN?
correct
TRAN
wrong
balance
withdrawal
BAL
DRAW

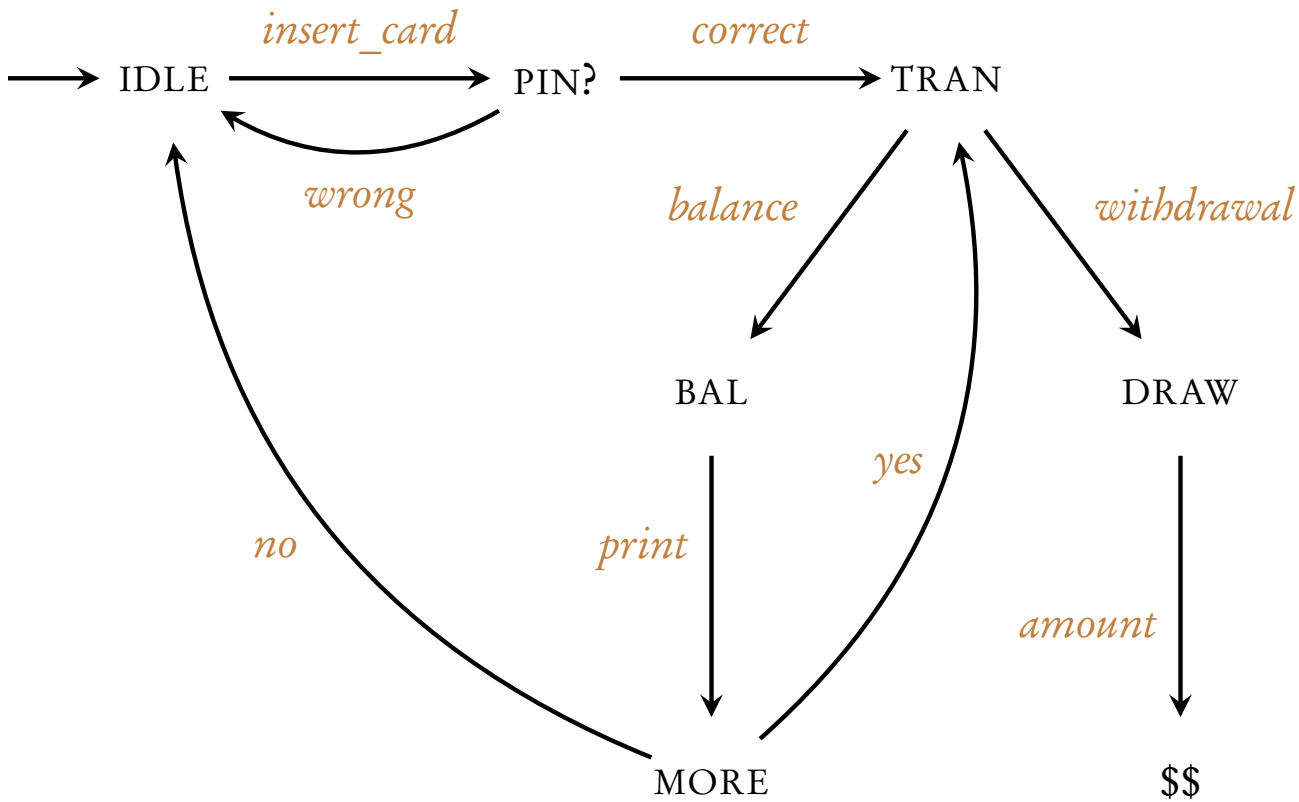

ATM


IDLE
insert_card
PIN?
correct
TRAN
wrong
balance
withdrawal
BAL
DRAW
print
MORE

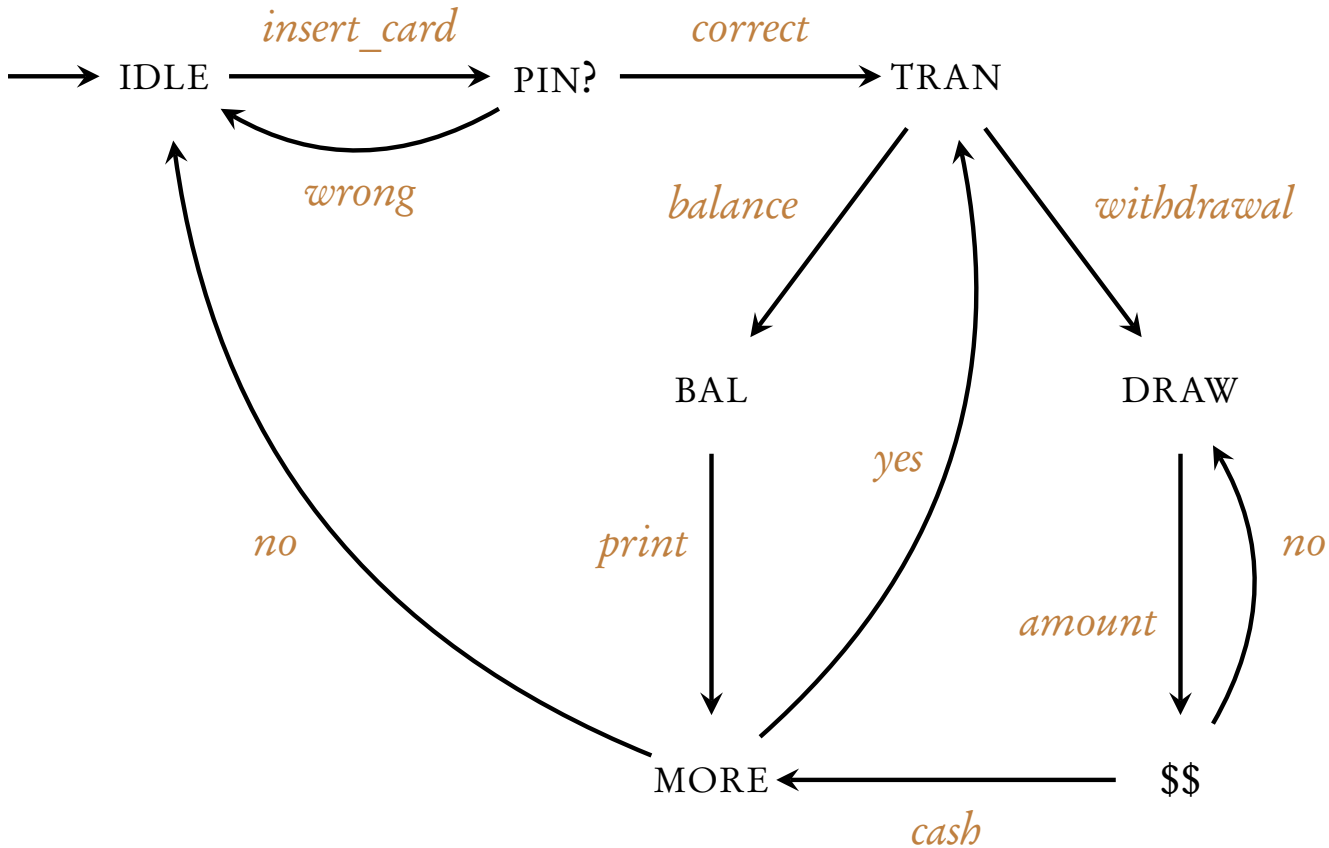

ATM


IDLE
PIN?
TRAN
BAL
DRAW
MORE
insert_card
correct
wrong
balance
withdrawal
yes
no
print

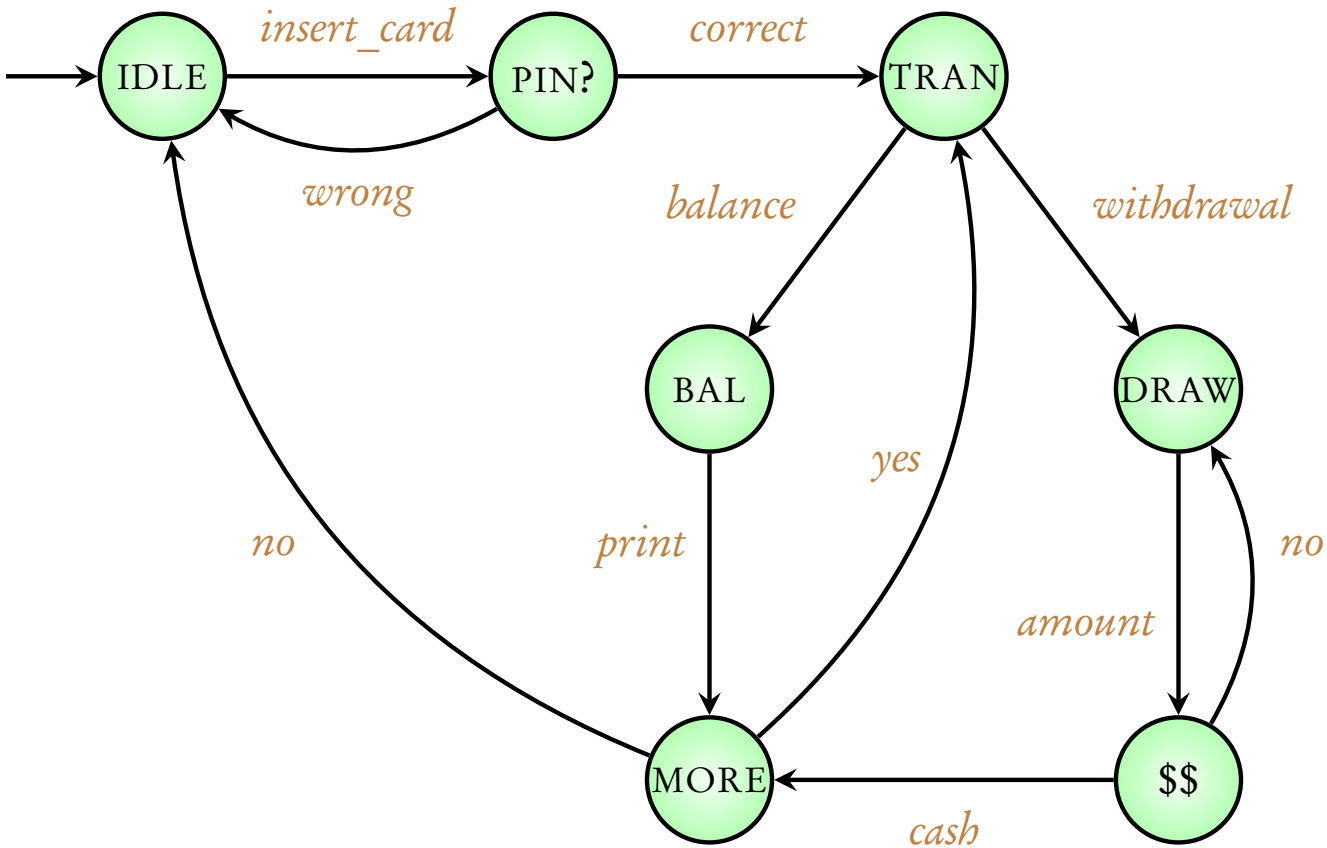

ATM


IDLE
insert_card
PIN?
correct
TRAN
wrong
balance
withdrawal
BAL
DRAW
yes
no
print
amount
MORE
$$

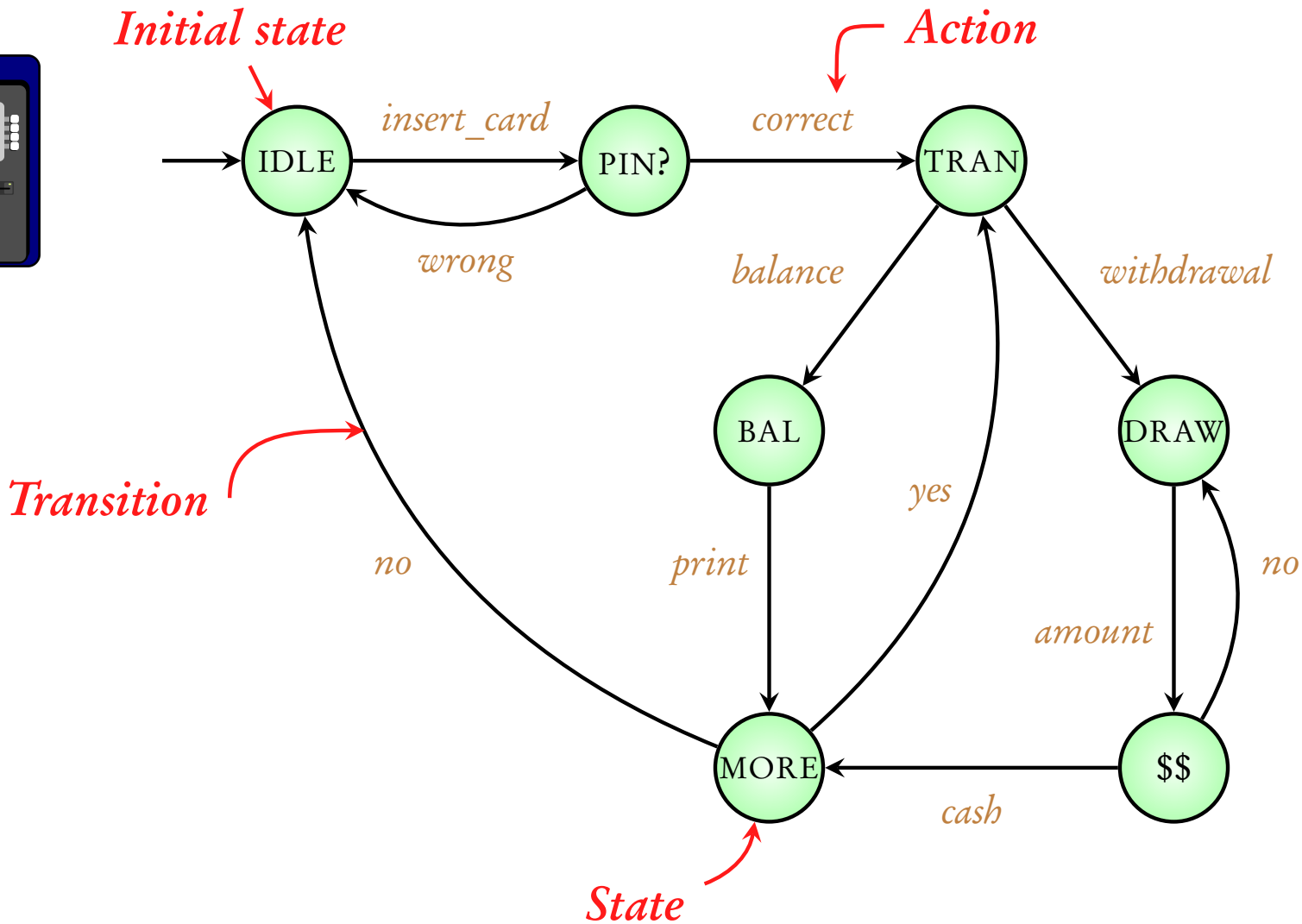

ATM


IDLE
insert_card
PIN?
correct
TRAN
wrong
balance
withdrawal
BAL
DRAW
yes
no
print
amount
no
MORE
$$
cash

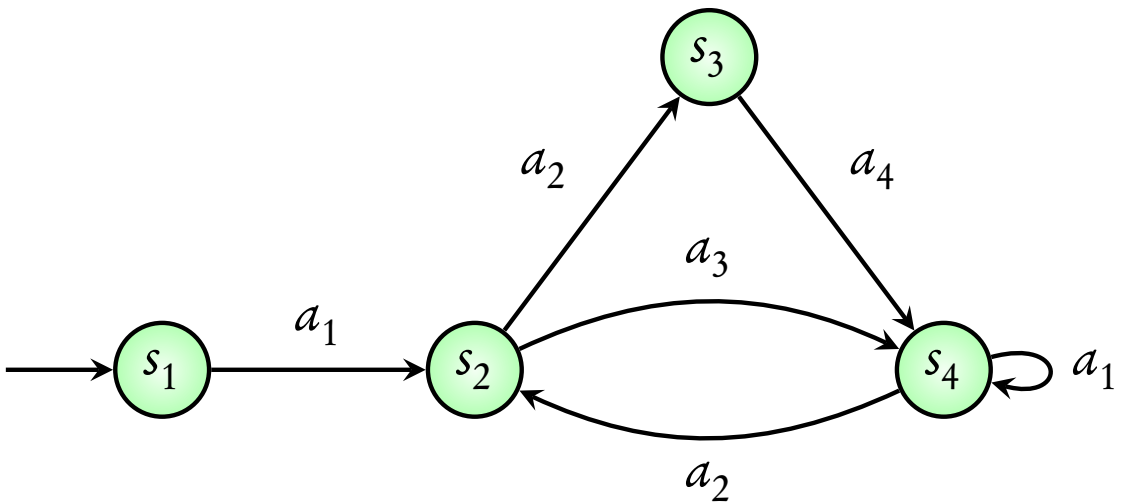

ATM


IDLE
PIN?
TRAN
BAL
DRAW
MORE
$$
insert_card
correct
wrong
balance
withdrawal
yes
no
print
no
amount
cash


ATM


Initial state
Action
IDLE
insert_card
PIN?
correct
TRAN
wrong
balance
withdrawal
BAL
DRAW
Transition
yes
no
print
no
amount
MORE
$$
cash
State

Transition system



States

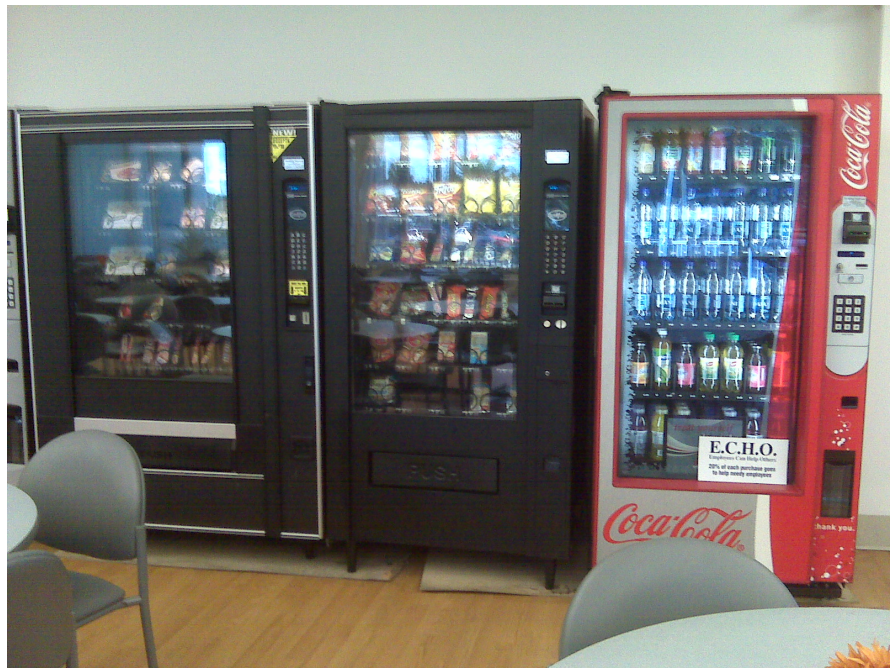
Actions

Transitions

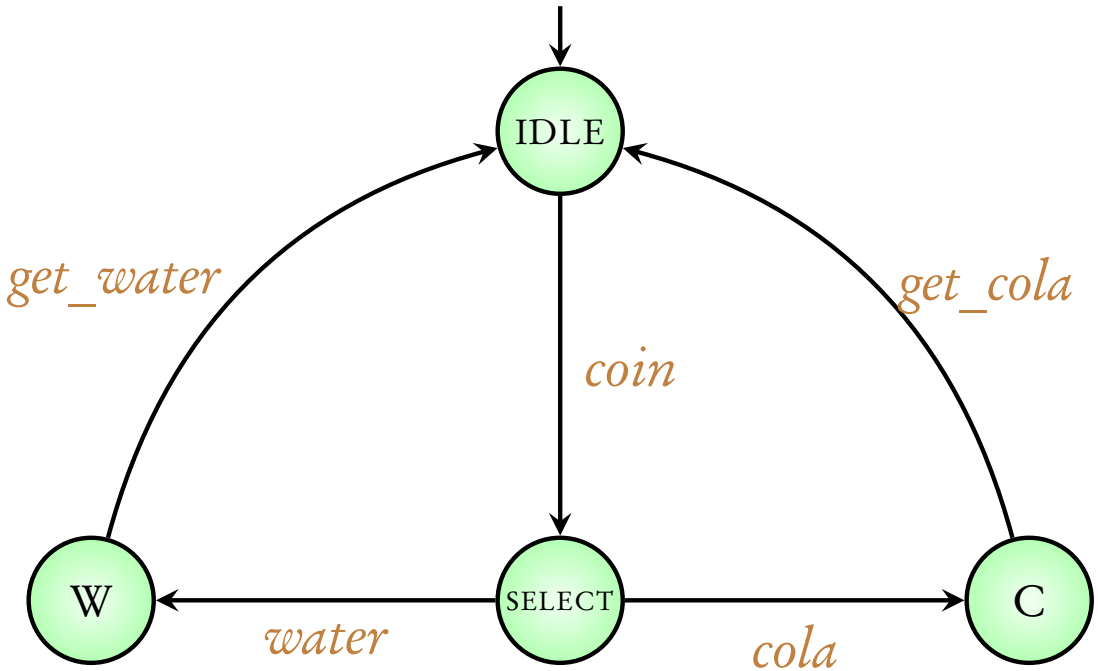
Initial state

Other names: Finite-state machines, State-transition graphs

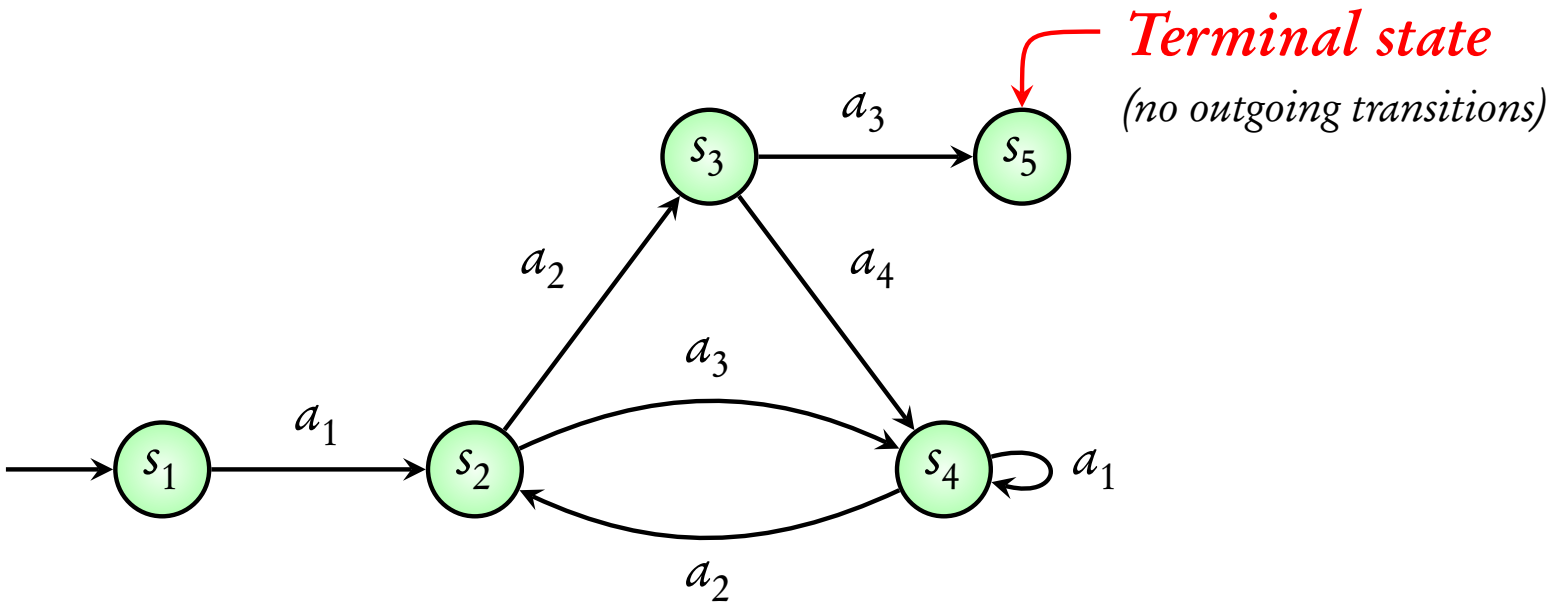
Coming next: Modeling a vending machine

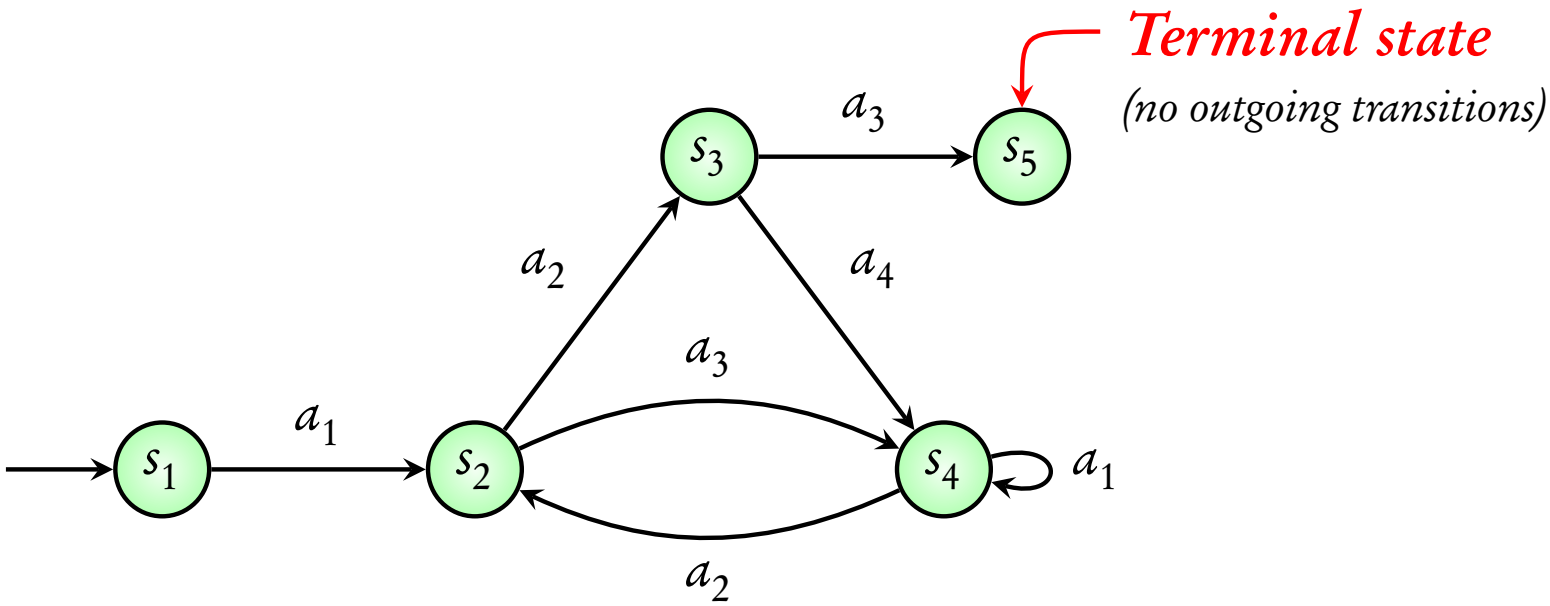


"Vending machines at hospital" by PCHS-NJROTC - Own work
Licensed under CC BY - SA 3.0 via Wikimedia Commons


IDLE
get_water
coin
get_cola
W
SELECT
C
water
cola

Coming next: some **terminology**

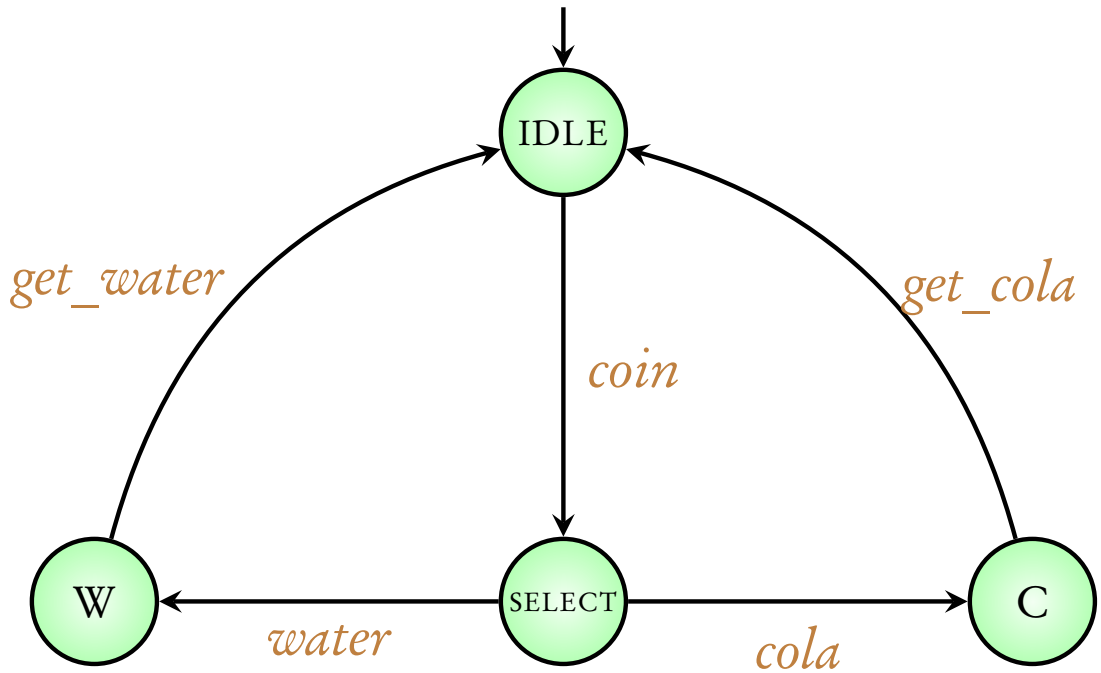

Terminal state
(no outgoing transitions)
s_1
s_2
s_3
s_4
s_5
a_1
a_2
a_3
a_4
a_3
a_1
a_2



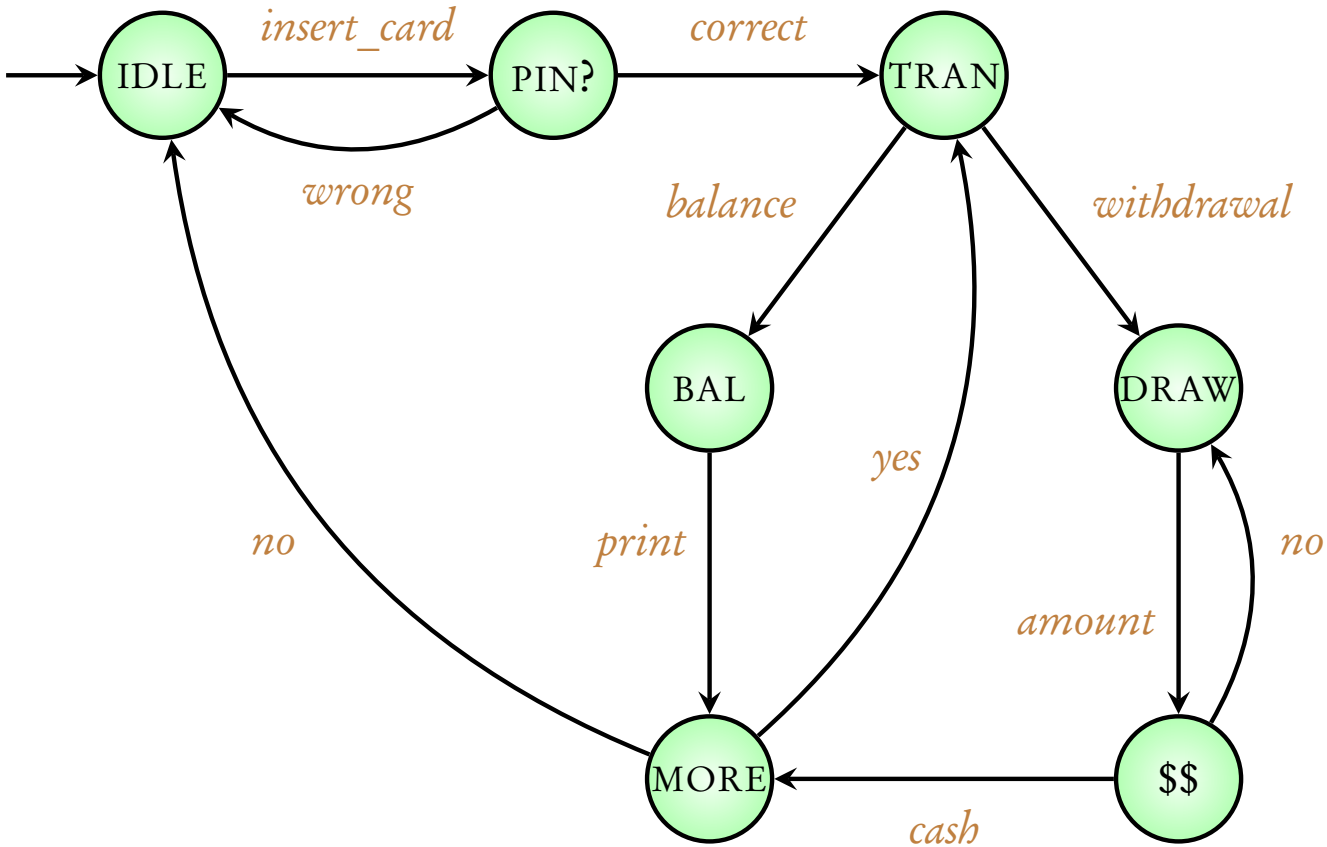
Execution: $s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_3} s_4 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_4 \xrightarrow{a_2} s_2 \xrightarrow{a_3} s_4 \ldots$

$s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_4} s_4 \xrightarrow{a_1} s_4 \xrightarrow{a_1} s_4 \xrightarrow{a_1} s_4 \ldots$

$s_1 \xrightarrow{a_1} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_4} s_4 \xrightarrow{a_2} s_2 \xrightarrow{a_2} s_3 \xrightarrow{a_3} s_5$



Execution: IDLE $\xrightarrow{coin}$ SELECT $\xrightarrow{water}$ W $\xrightarrow{get\_water}$ IDLE $\xrightarrow{coin}$ SELECT $\xrightarrow{cola}$ $\cdots$



Execution: IDLE $\xrightarrow{insert\_card}$ PIN $\xrightarrow{wrong}$ IDLE $\xrightarrow{insert\_card}$ PIN $\cdots$

Summary

Transition Systems

States, actions, transitions

Executions

Reference: Principles of Model Checking, *Baier and Katoen*, MIT Press (2008)

Pages 19 - 26

Week-1: Introduction to model checking

B. Srivathsan

Chennai Mathematical Institute

NPTEL-course

July - November 2015

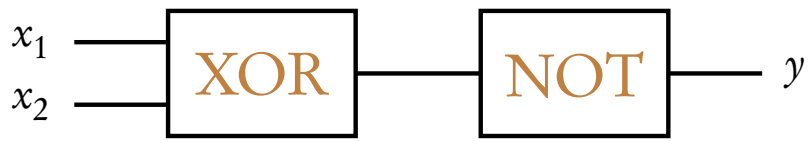
Module 2:
Modeling hardware circuits

x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1

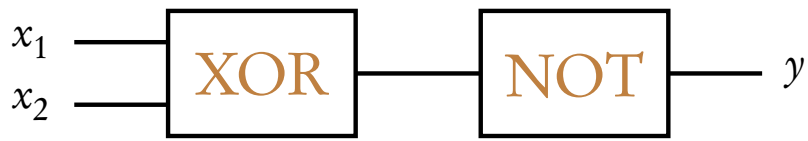
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

x	y
0	1
1	0

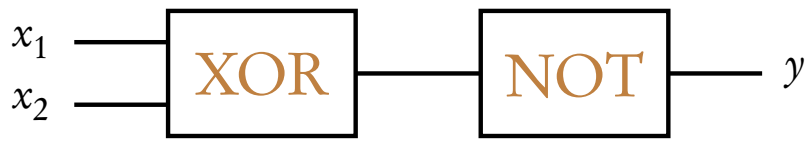
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	0


x_1
x_2
XOR
NOT
y

$$y = \text{NOT}\,(\,\text{XOR}\,(x_1, x_2)\,)$$

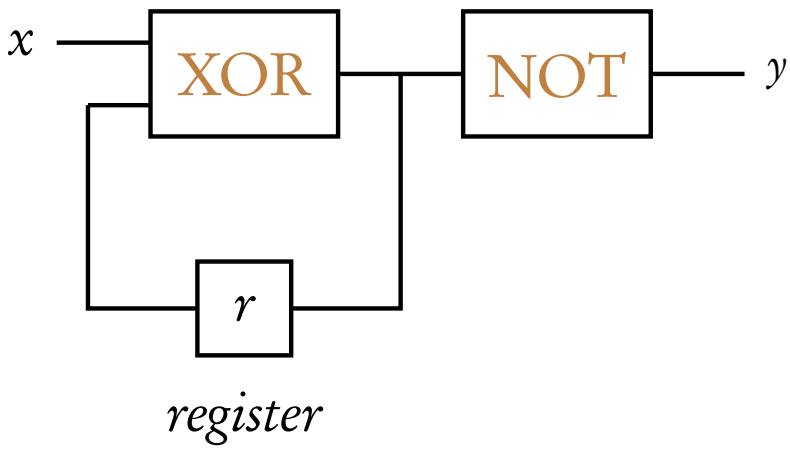


$$y = \text{NOT}\,(\,\text{XOR}\,(x_1, x_2)\,)$$



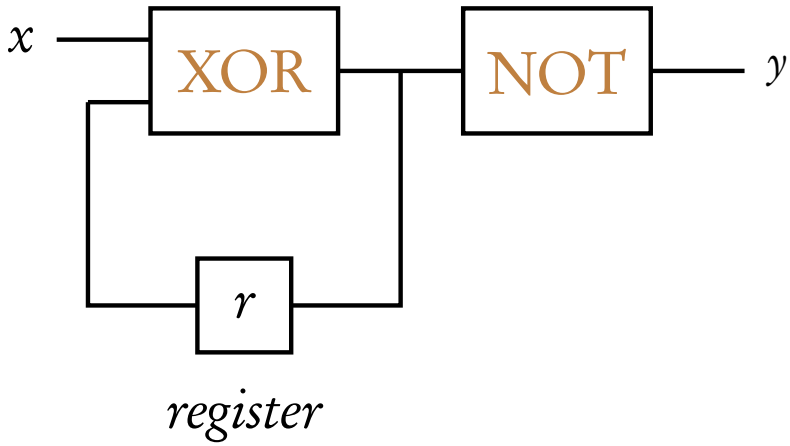
x_1	x_2	y
0	0	1
0	1	0
1	0	0
1	1	1

$$y = \text{NOT}\,(\,\text{XOR}\,(x, r)\,)$$



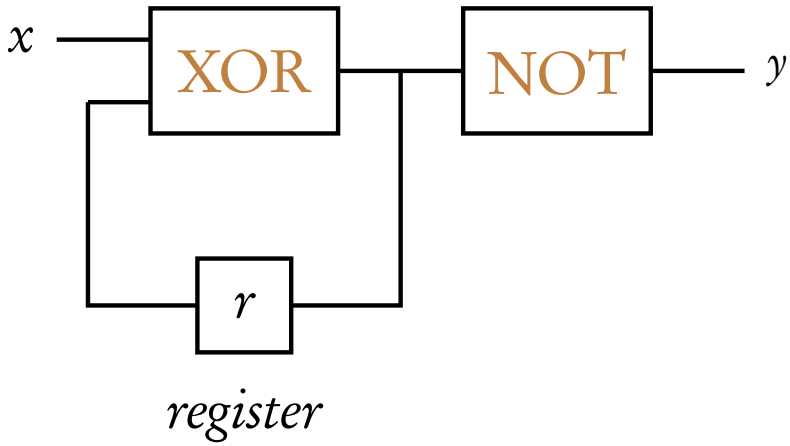
$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

$$r_{next} = \text{XOR}\ (x, r)$$



$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

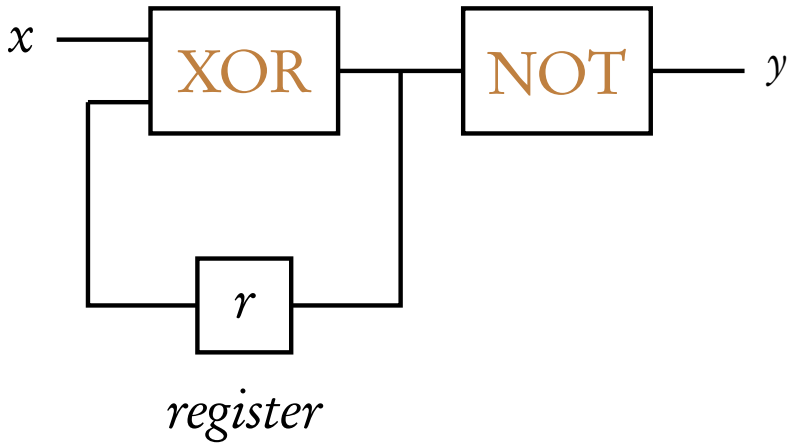
$$r_{next} = \text{XOR}\ (x, r)$$

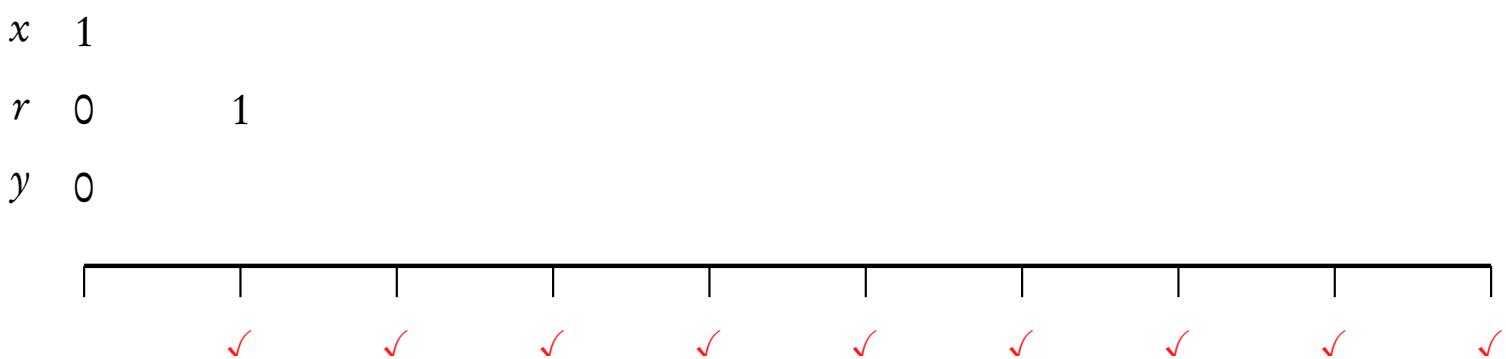


x	1
r	0
y	0

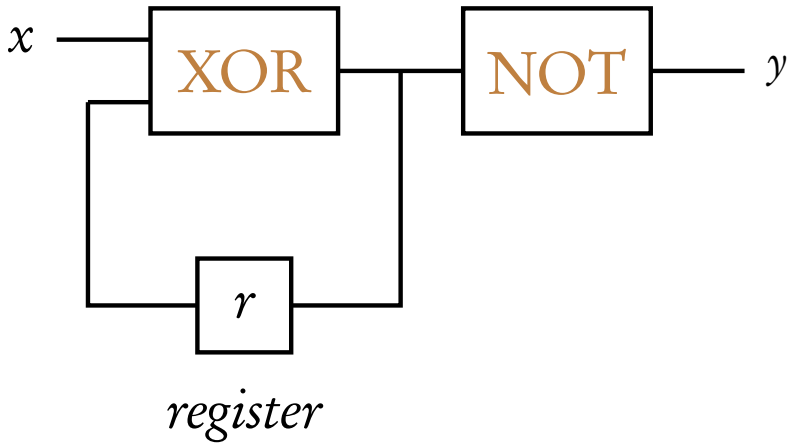
$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

$$r_{next} = \text{XOR}\ (x, r)$$

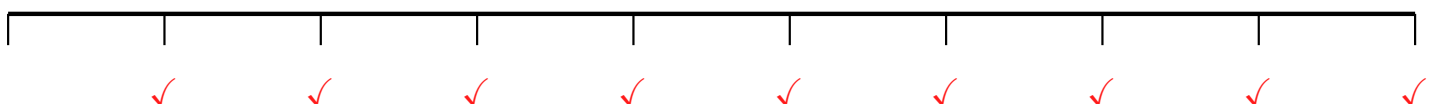




$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$
$$r_{next} = \text{XOR}\ (x, r)$$

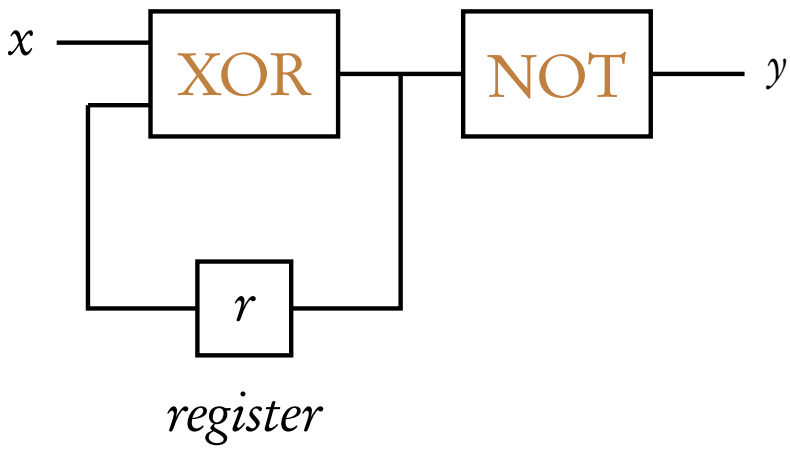


x	1	1
r	0	1
y	0	1

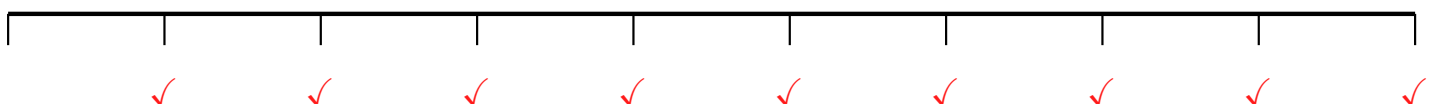


$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

$$r_{next} = \text{XOR}\ (x, r)$$

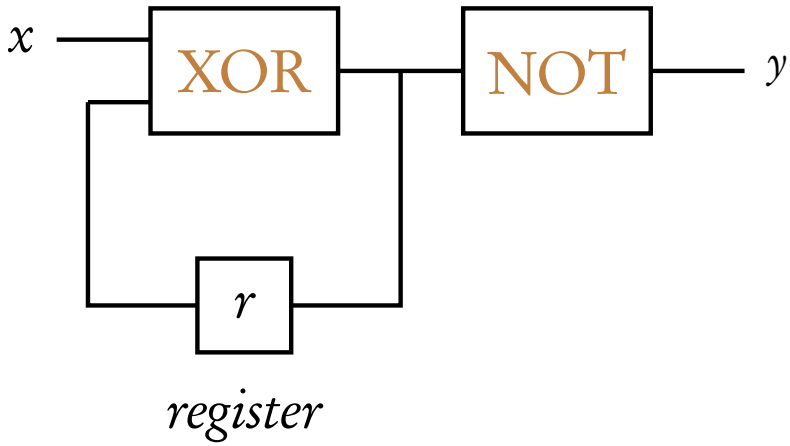


x	1	1	
r	0	1	0
y	0	1	

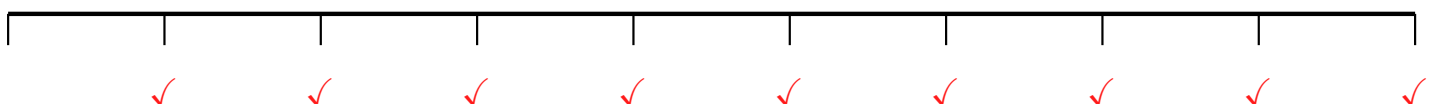


$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

$$r_{next} = \text{XOR}\ (x, r)$$

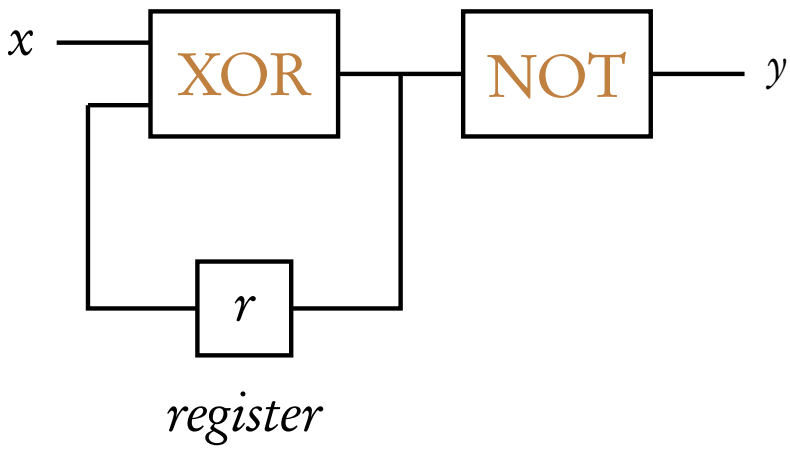


x	1	1	0
r	0	1	0
y	0	1	1

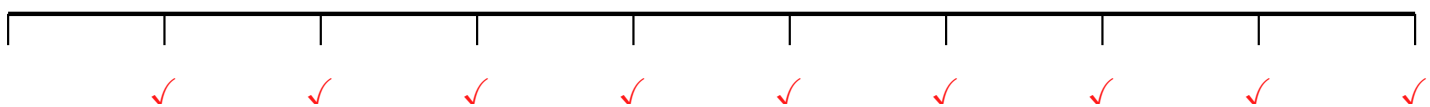


$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

$$r_{next} = \text{XOR}\ (x, r)$$

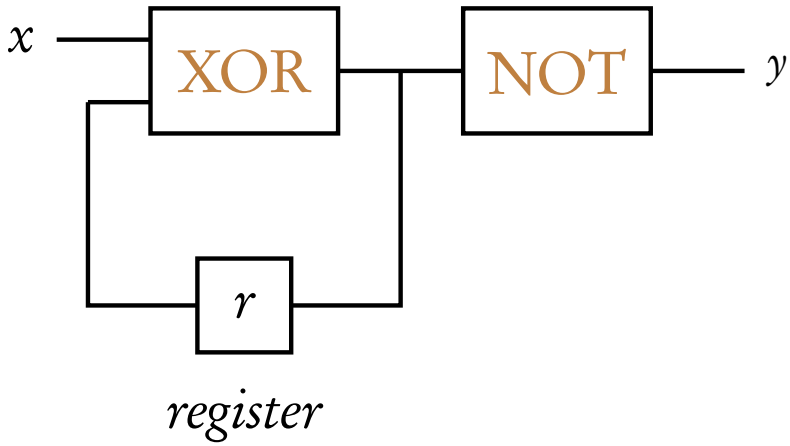


x	1	1	0	
r	0	1	0	0
y	0	1	1	

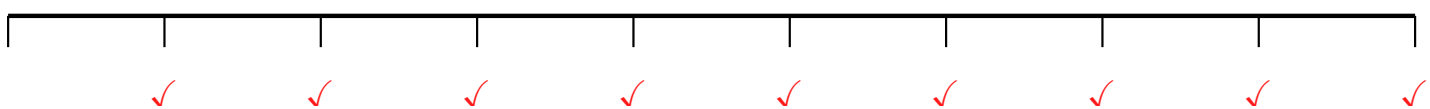


$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

$$r_{next} = \text{XOR}\ (x, r)$$

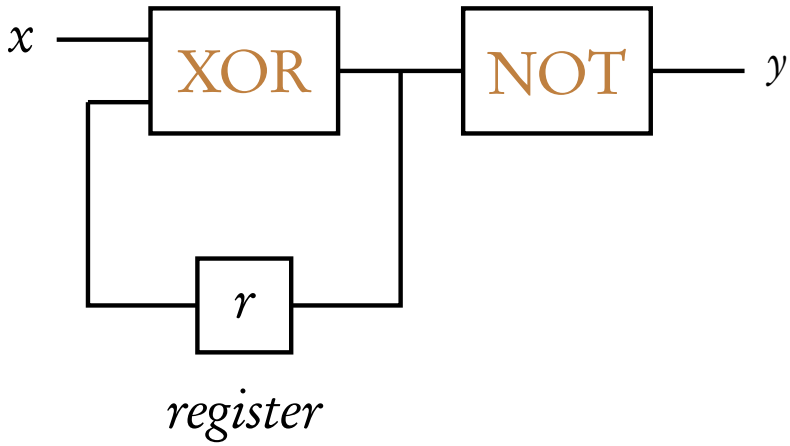


x	1	1	0	1
r	0	1	0	0
y	0	1	1	0

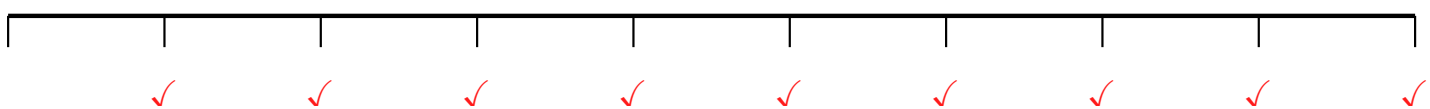


$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

$$r_{next} = \text{XOR}\ (x, r)$$

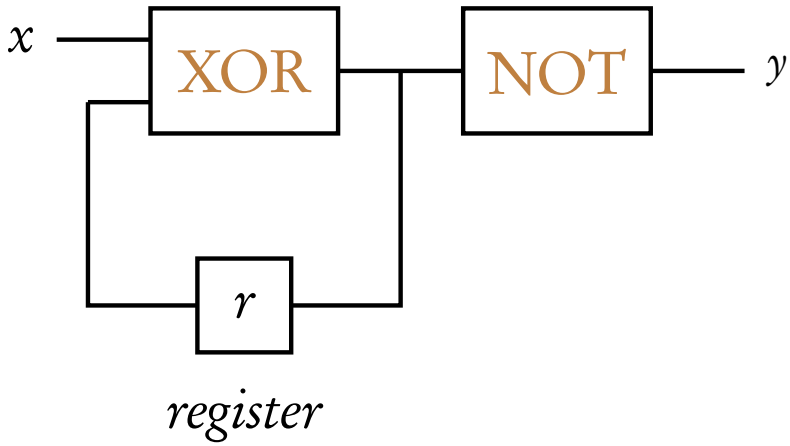


x	1	1	0	1	
r	0	1	0	0	1
y	0	1	1	0	

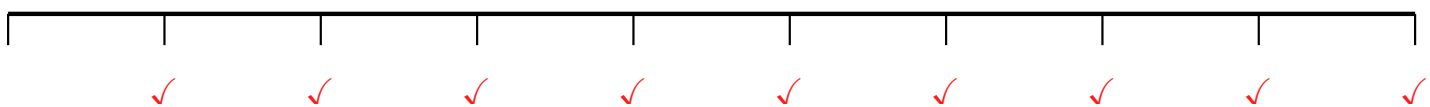


$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

$$r_{next} = \text{XOR}\ (x, r)$$

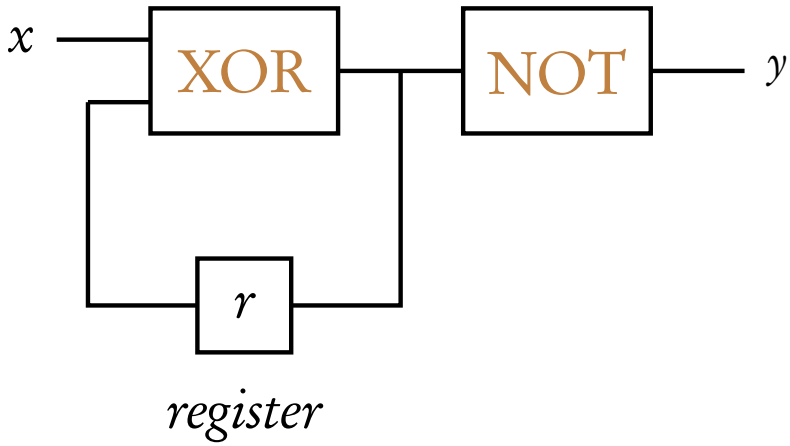


x	1	1	0	1	1
r	0	1	0	0	1
y	0	1	1	0	1

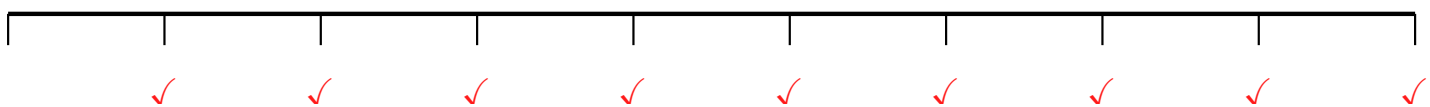


$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

$$r_{next} = \text{XOR}\ (x, r)$$

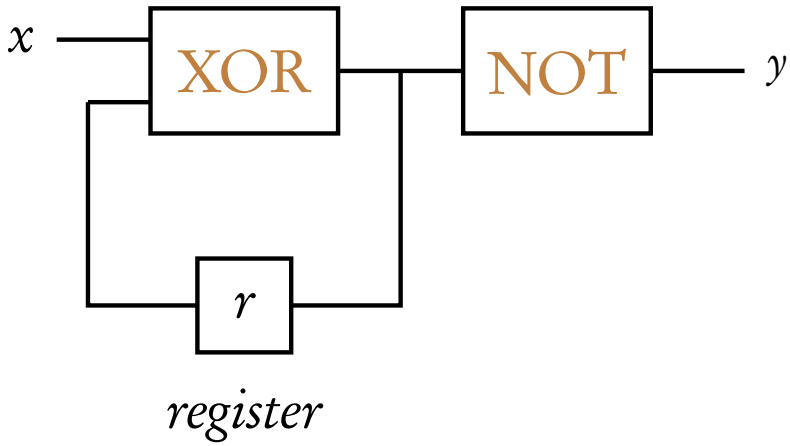


x	1	1	0	1	1	1	0	0	1	0
r	0	1	0	0	1	0	1	1	1	0 •••
y	0	1	1	0	1	0	0	0	1	1



$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

$$r_{next} = \text{XOR}\ (x, r)$$



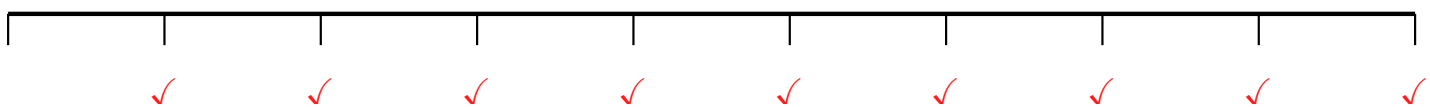
$x = 0, r = 0, y = 1$

$x = 1, r = 0, y = 0$

$x = 0, r = 1, y = 0$

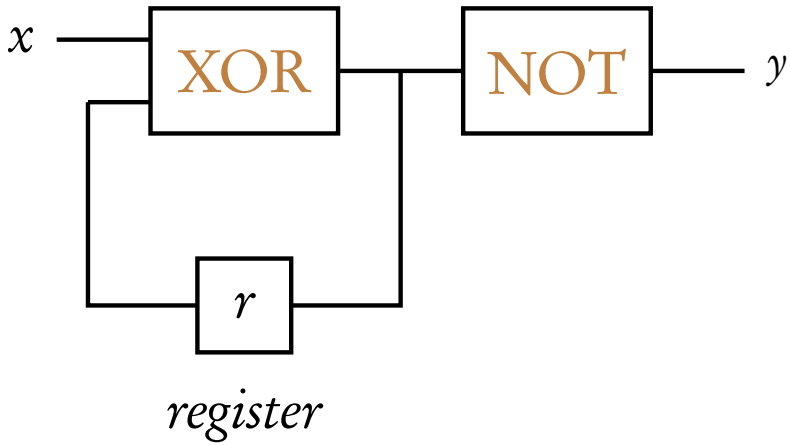
$x = 1, r = 1, y = 1$

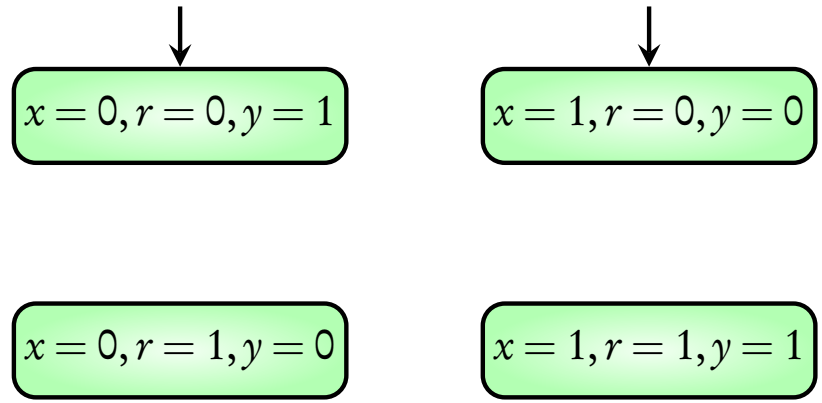
x	1	1	0	1	1	1	0	0	1	0
r	0	1	0	0	1	0	1	1	1	0
y	0	1	1	0	1	0	0	0	1	1



$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

$$r_{next} = \text{XOR}\ (x, r)$$

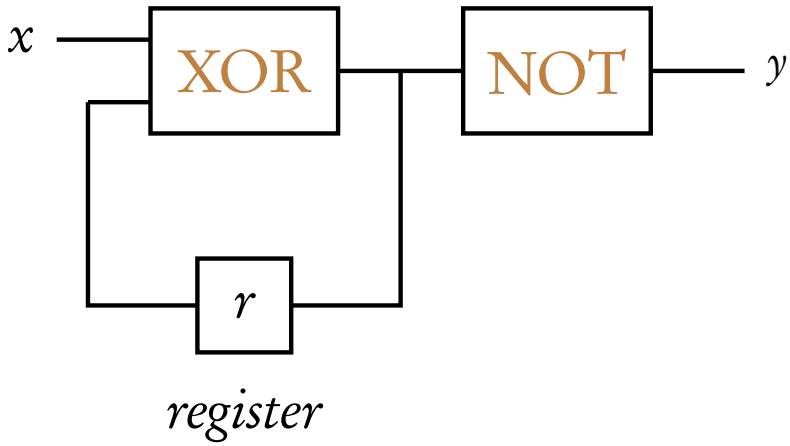


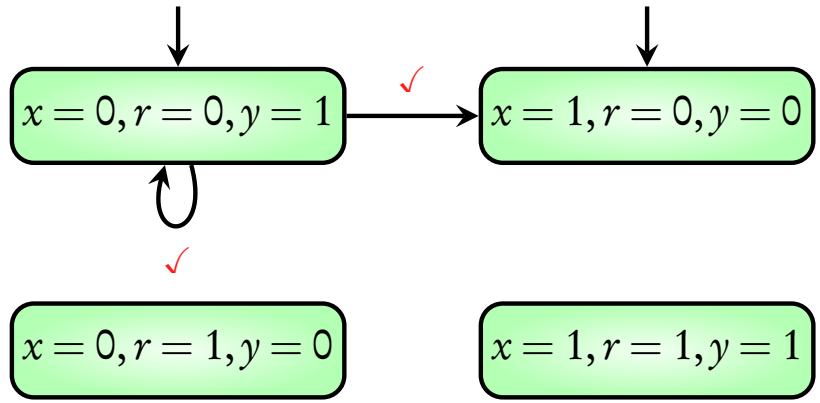


x	1	1	0	1	1	1	0	0	1	0
r	0	1	0	0	1	0	1	1	1	0
y	0	1	1	0	1	0	0	0	1	1

$$y = \text{NOT}\,(\,\text{XOR}\,(x, r)\,)$$

$$r_{next} = \text{XOR}\,(x, r)$$

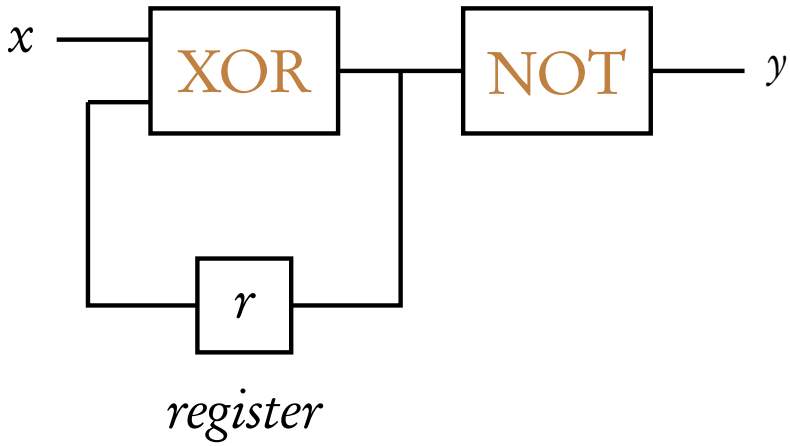


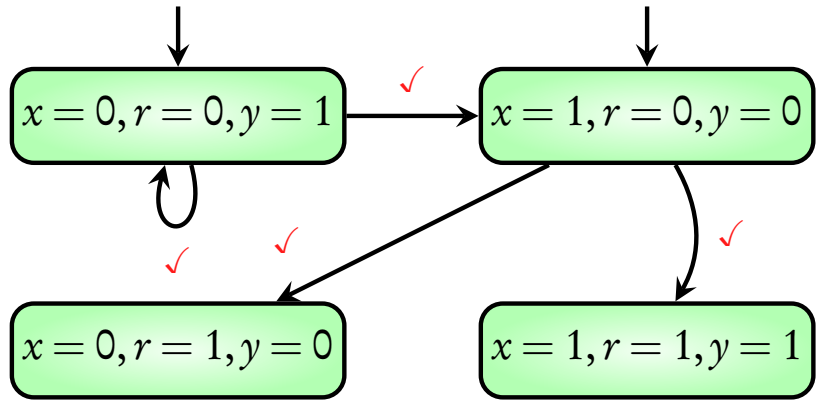


x	1	1	0	1	1	1	0	0	1	0	
r	0	1	0	0	1	0	1	1	1	0	• • •
y	0	1	1	0	1	0	0	0	1	1	

$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

$$r_{next} = \text{XOR}\ (x, r)$$

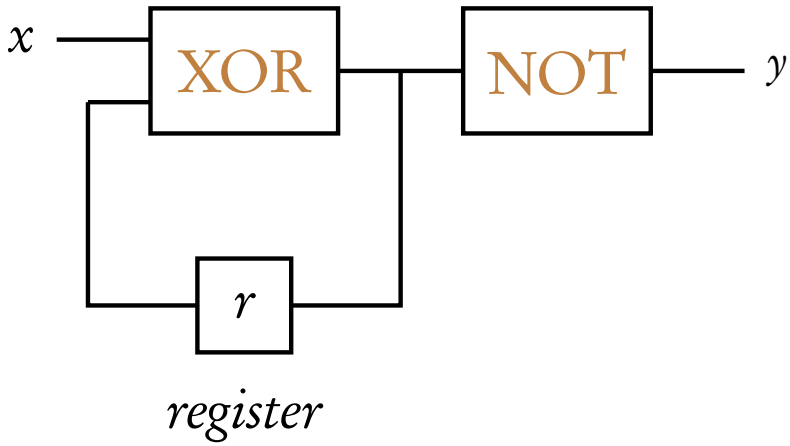


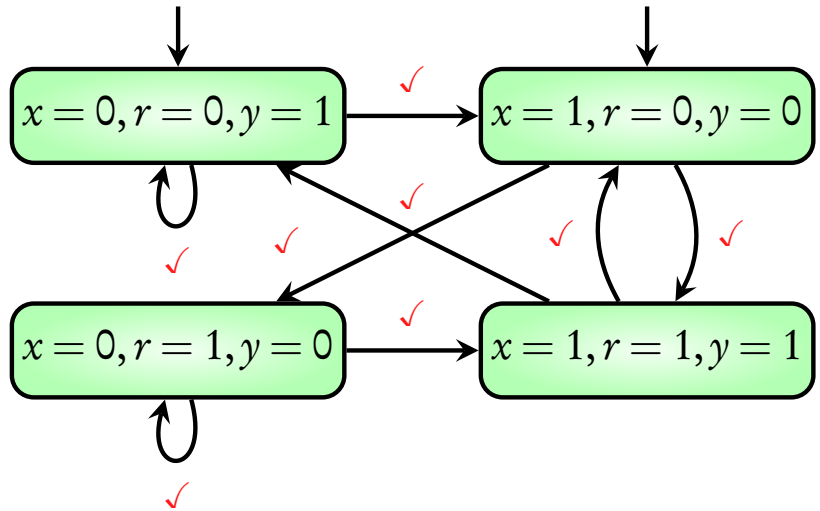


x	1	1	0	1	1	1	0	0	1	0
r	0	1	0	0	1	0	1	1	1	0
y	0	1	1	0	1	0	0	0	1	1

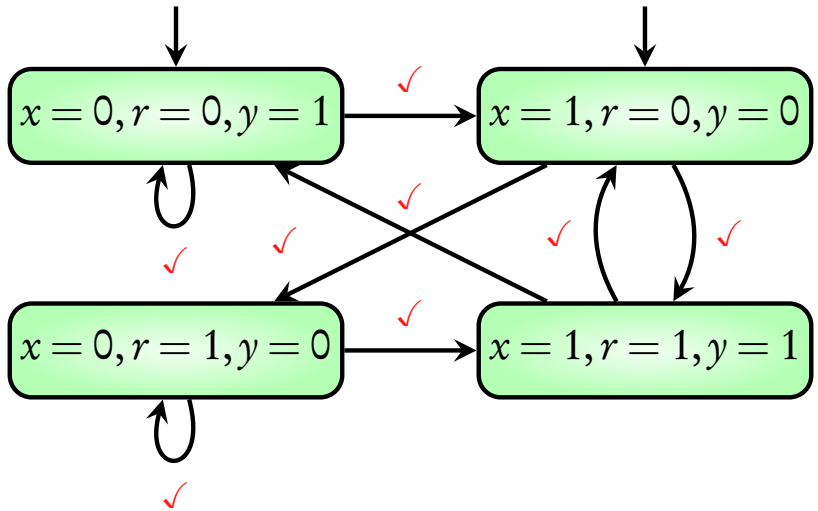
$$y = \text{NOT}\ (\ \text{XOR}\ (x, r)\ )$$

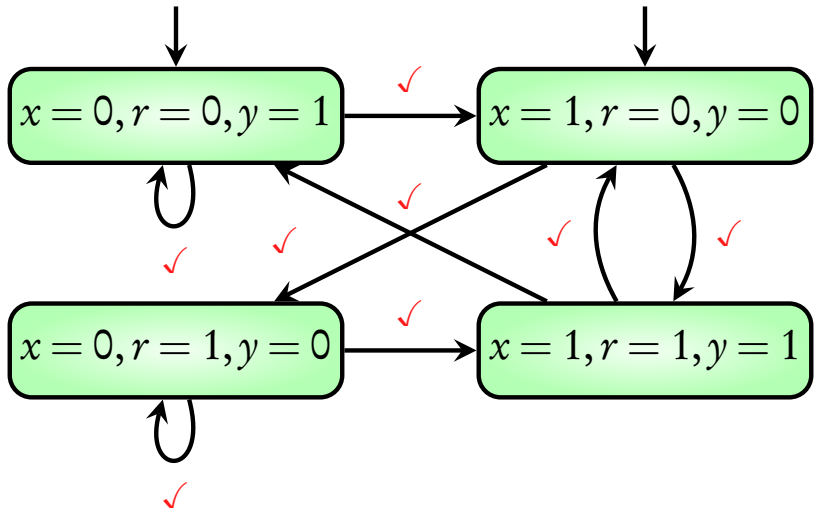
$$r_{next} = \text{XOR}\ (x, r)$$





x	1	1	0	1	1	1	0	0	1	0
r	0	1	0	0	1	0	1	1	1	0
y	0	1	1	0	1	0	0	0	1	1


$x = 0, r = 0, y = 1$
$x = 1, r = 0, y = 0$
$x = 0, r = 1, y = 0$
$x = 1, r = 1, y = 1$

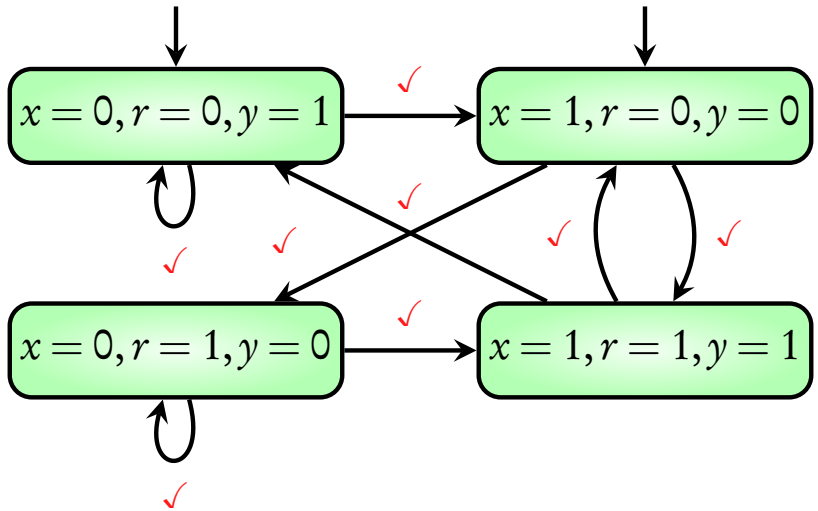


More than one initial state

States with **more than one transition** on an action

Non-deterministic

transition system



More than one initial state

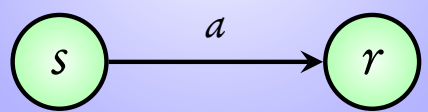
States with **more than one transition** on an action

Transition Systems

Deterministic

Single initial state

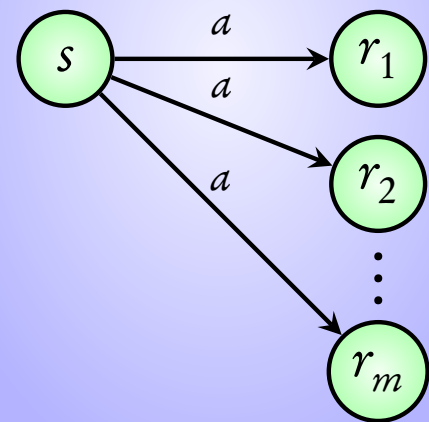
and



Non-deterministic

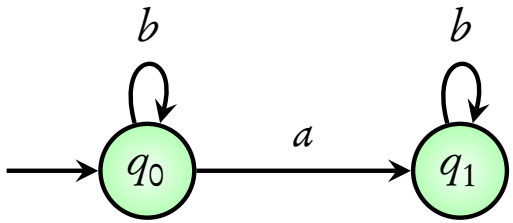
Multiple initial states

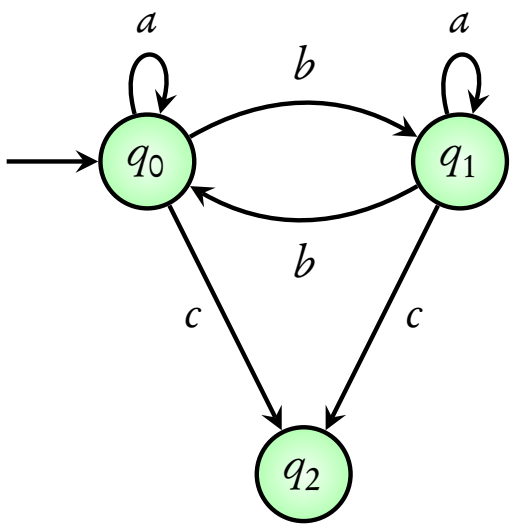
or

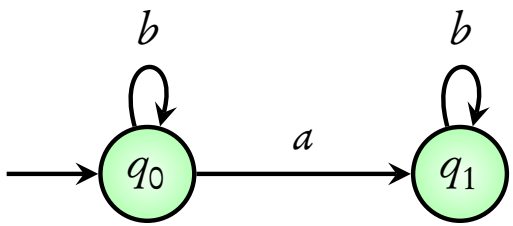


Coming next: **examples** of deterministic and non-deterministic transition systems

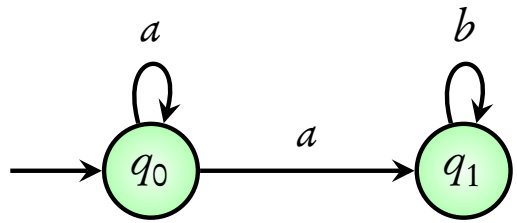
Deterministic

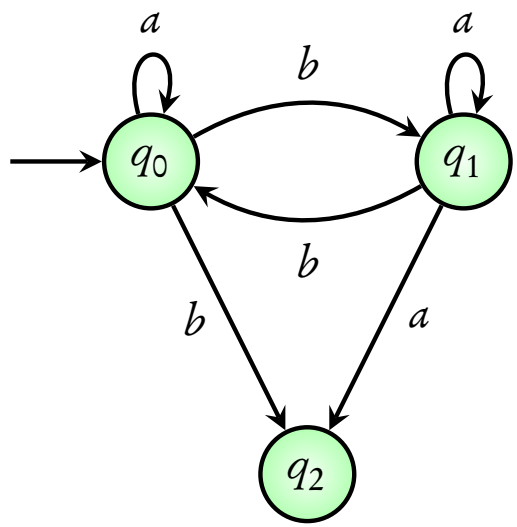


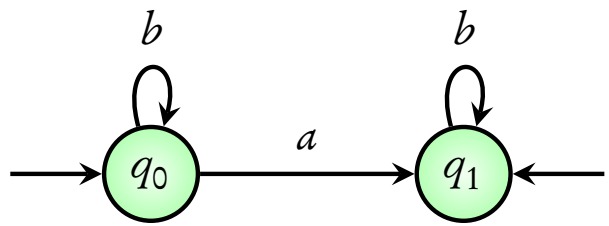




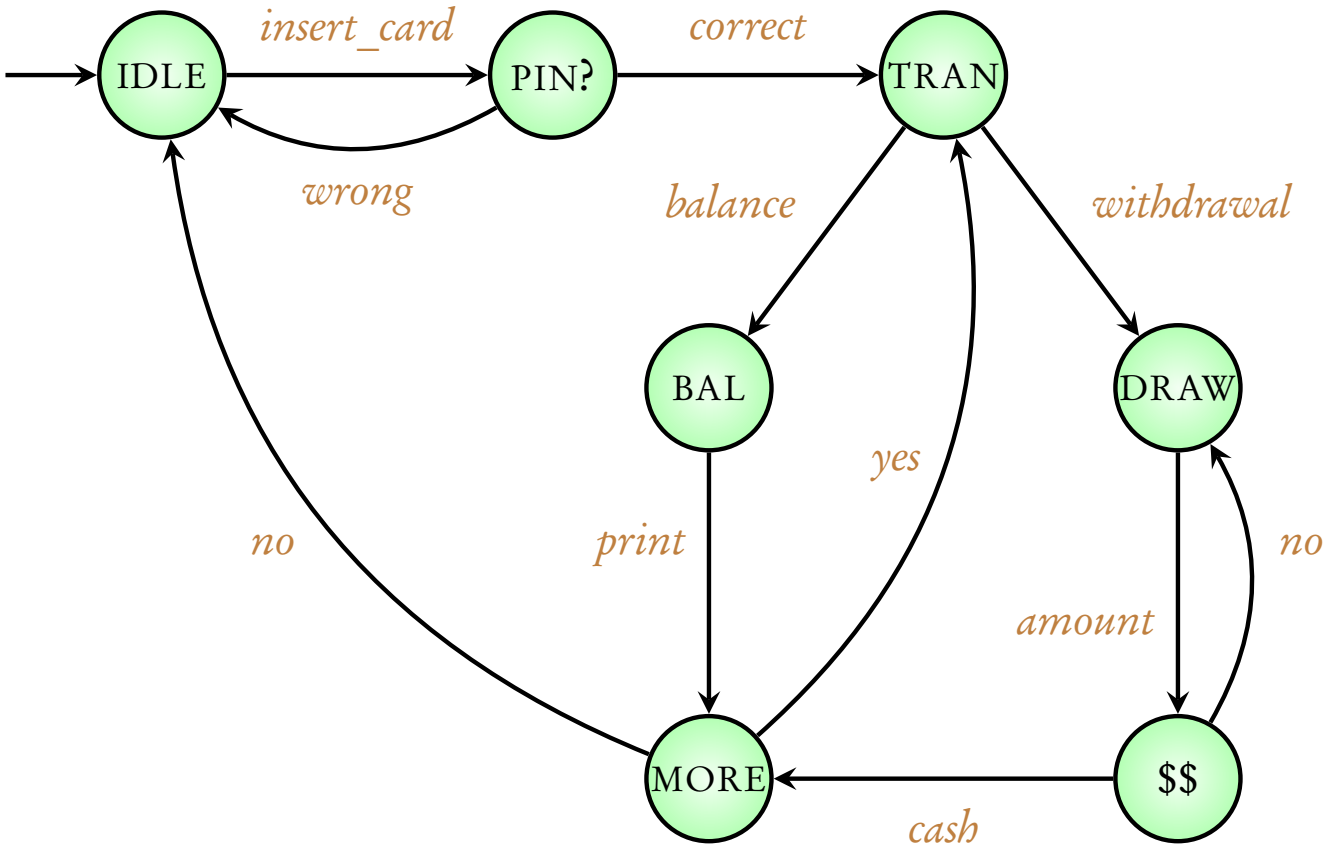
Non-deterministic



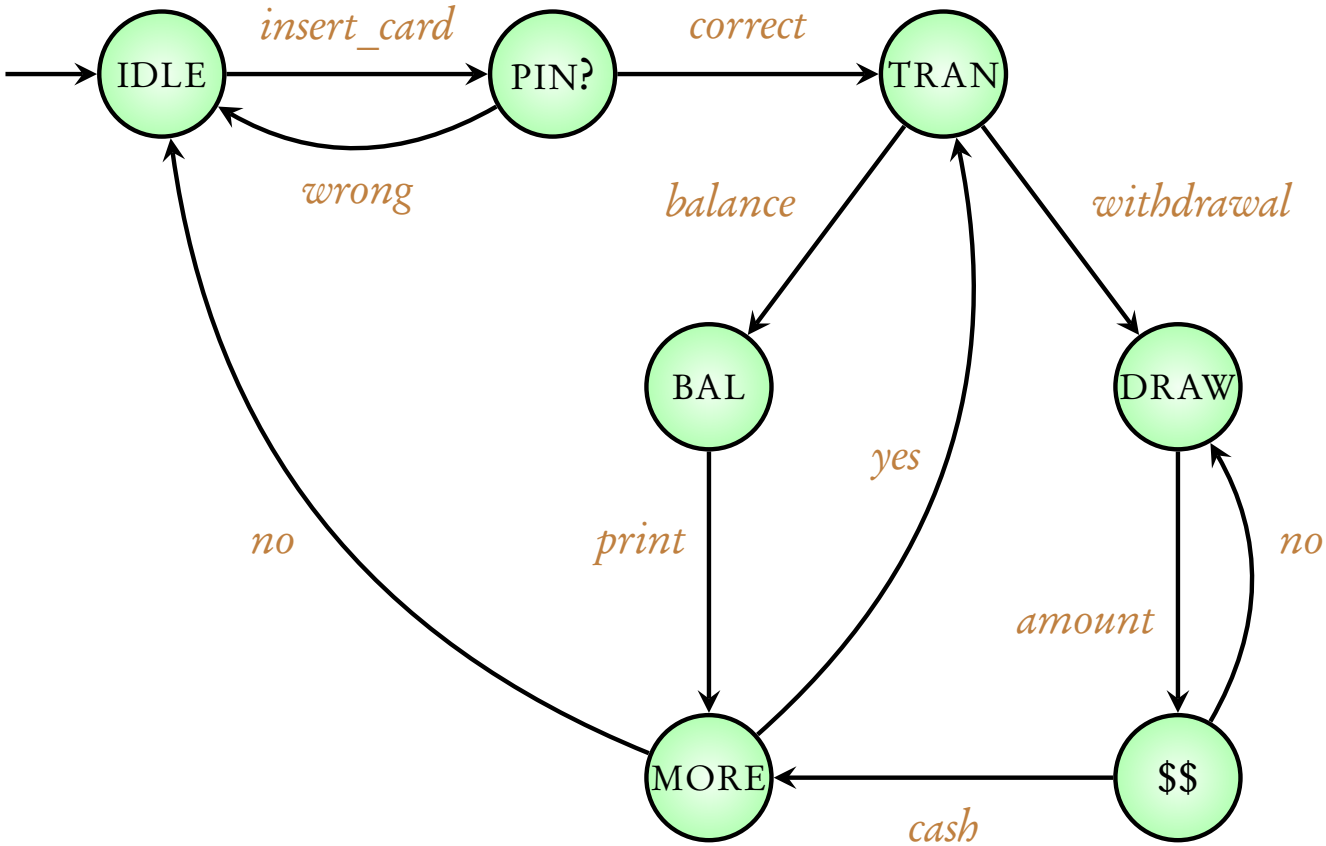




Model of ATM

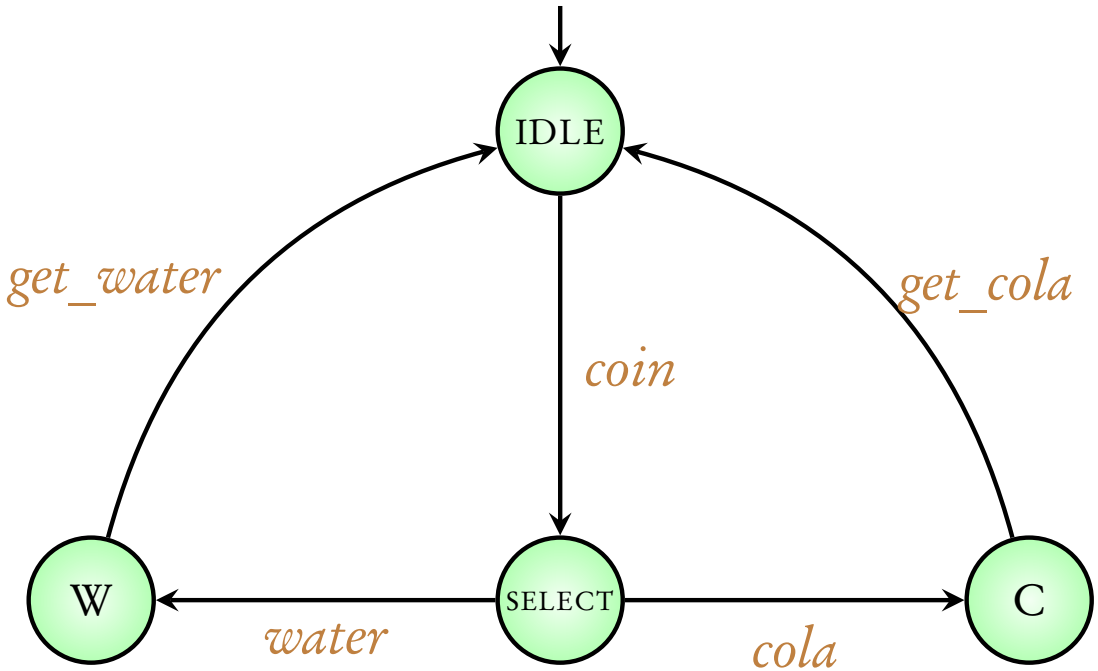

IDLE
PIN?
TRAN
BAL
DRAW
MORE
$$
insert_card
correct
wrong
balance
withdrawal
yes
no
print
no
amount
cash

Model of ATM

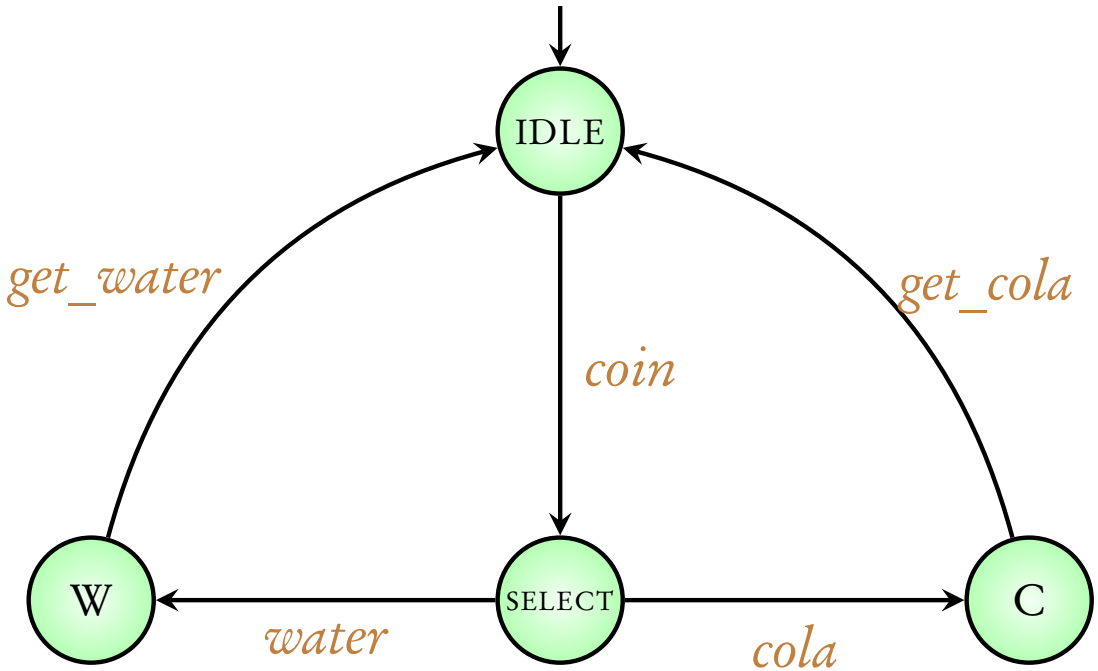


Deterministic transition system

Model of vending machine

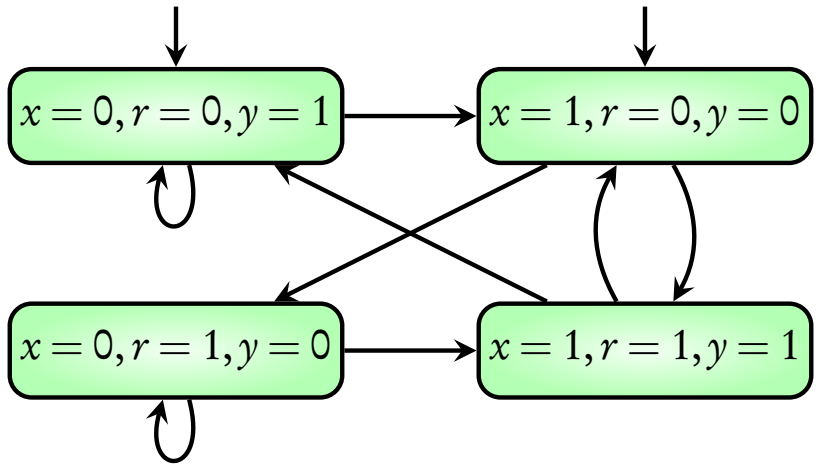


Model of vending machine

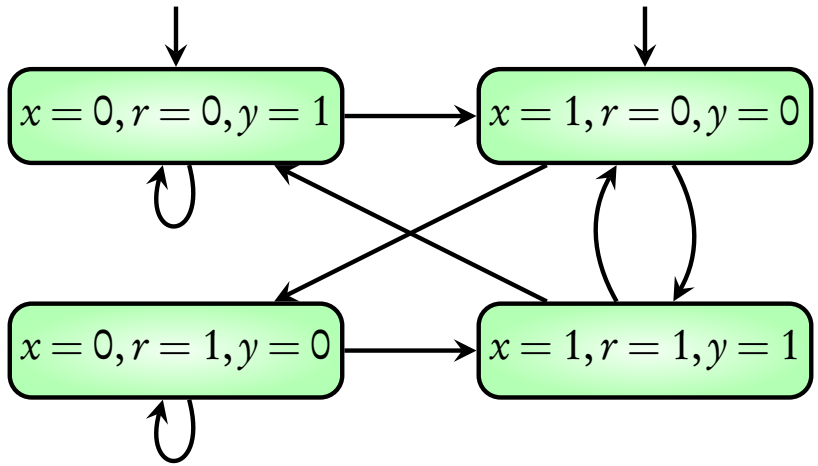


Deterministic transition system

Model of hardware circuit

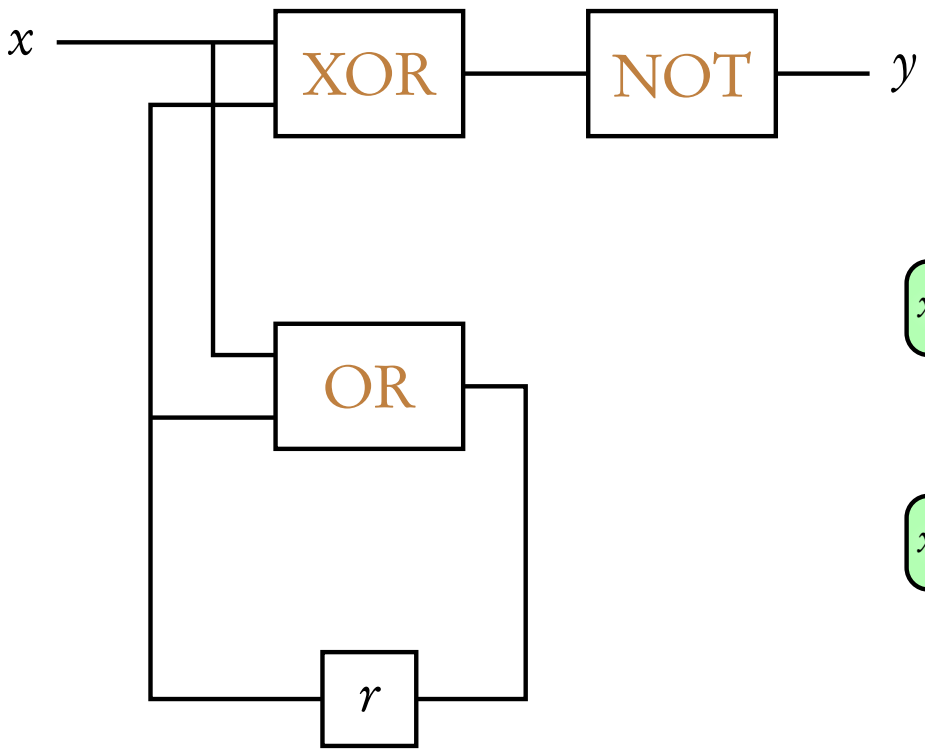


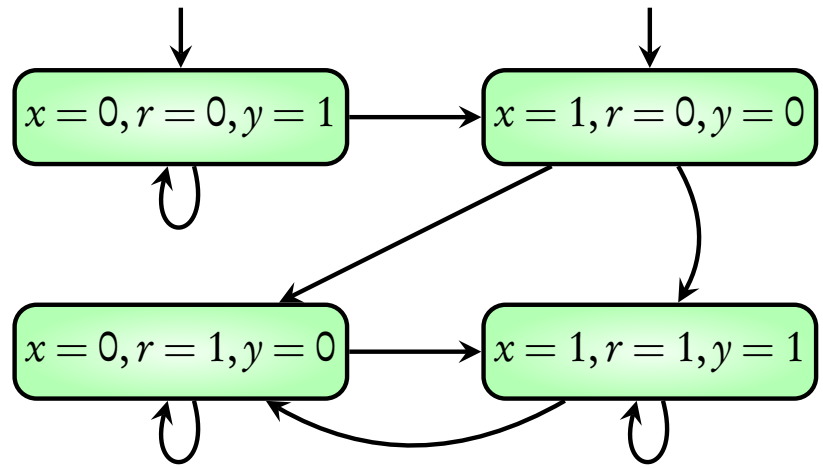
Model of hardware circuit



Non-deterministic transition system: to model **incomplete information**

Coming next: Another example of hardware circuit


x
XOR
NOT
y
OR
r


x = 0, r = 0, y = 1
x = 1, r = 0, y = 0
x = 0, r = 1, y = 0
x = 1, r = 1, y = 1

Summary

Hardware Circuits

Modeling using transition systems

Non-determinism

Reference: Principles of Model Checking, *Baier and Katoen*, MIT Press (2008)

Pages 26 - 29

Week-1: Introduction to model checking

B. Srivathsan

Chennai Mathematical Institute

NPTEL-course

July - November 2015

Module 3:

Modeling data-dependent programs

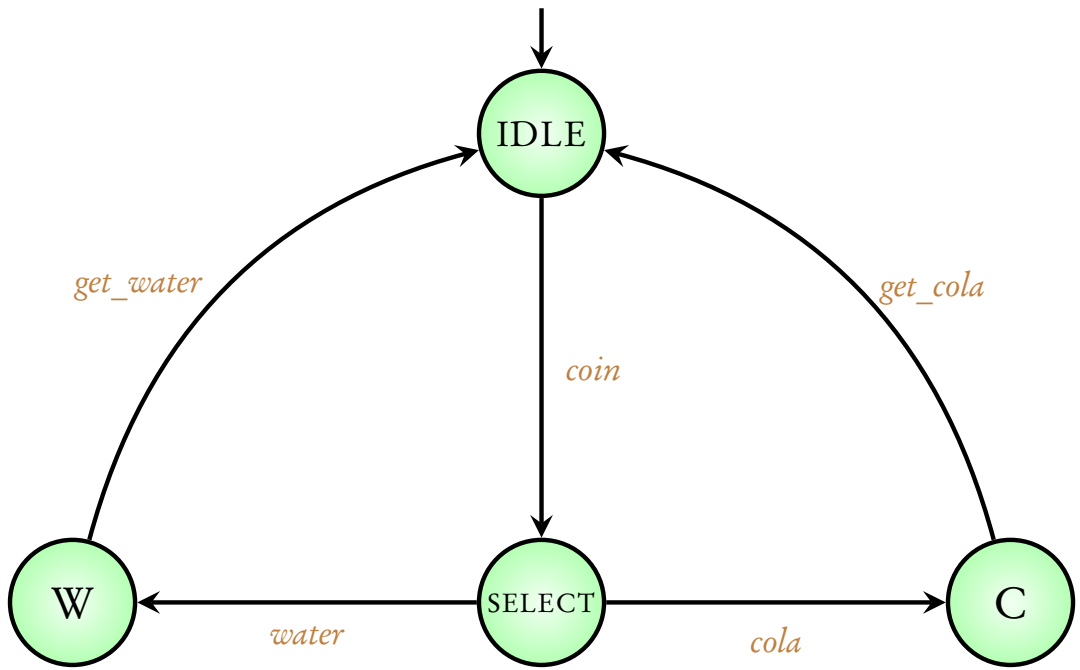
Data-dependent programs:

Variables + Conditional branching + Assignments

Data-dependent programs:

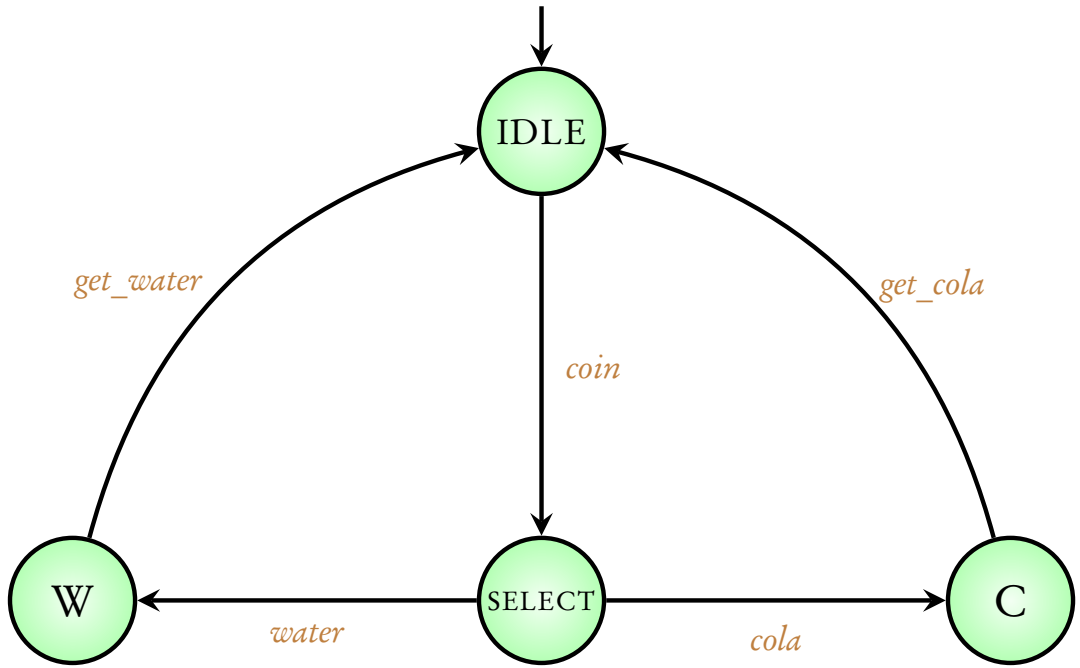
Variables + Conditional branching + Assignments

Coming next: vending machine revisited


IDLE
get_water
coin
get_cola
W
SELECT
C
water
cola

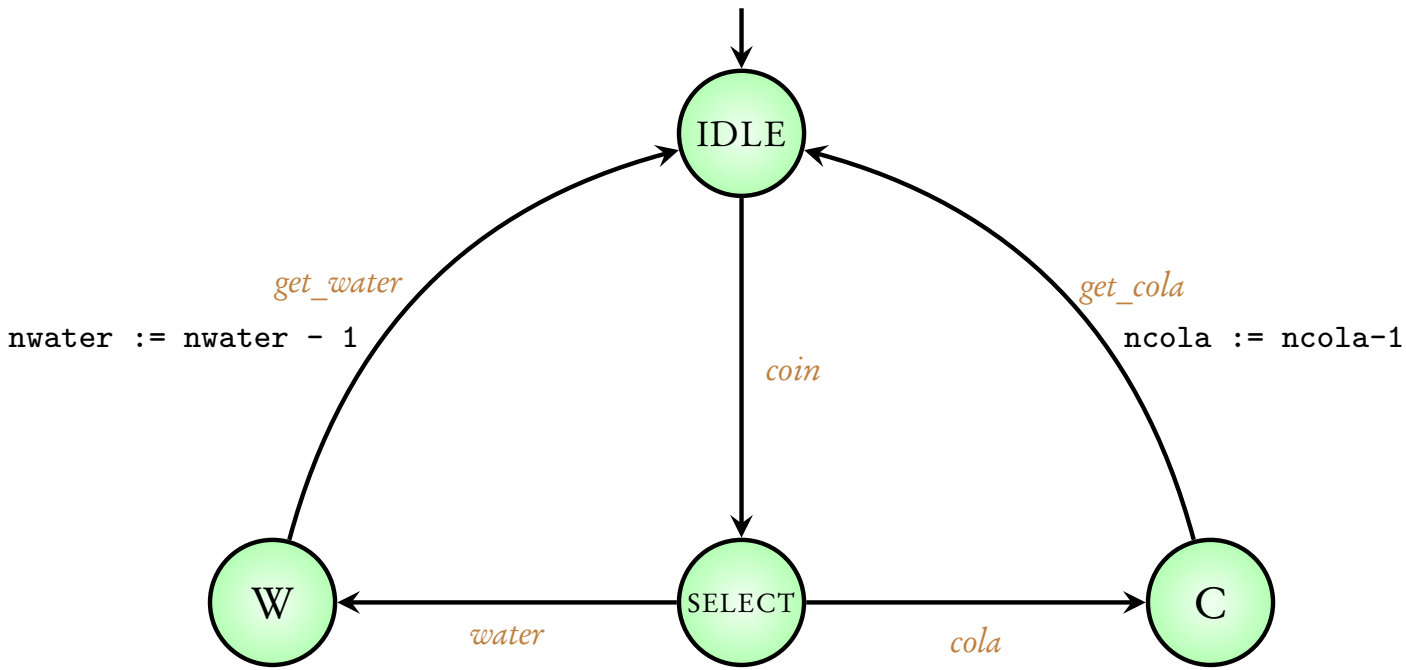
Variables:

`nwater, ncola, max`



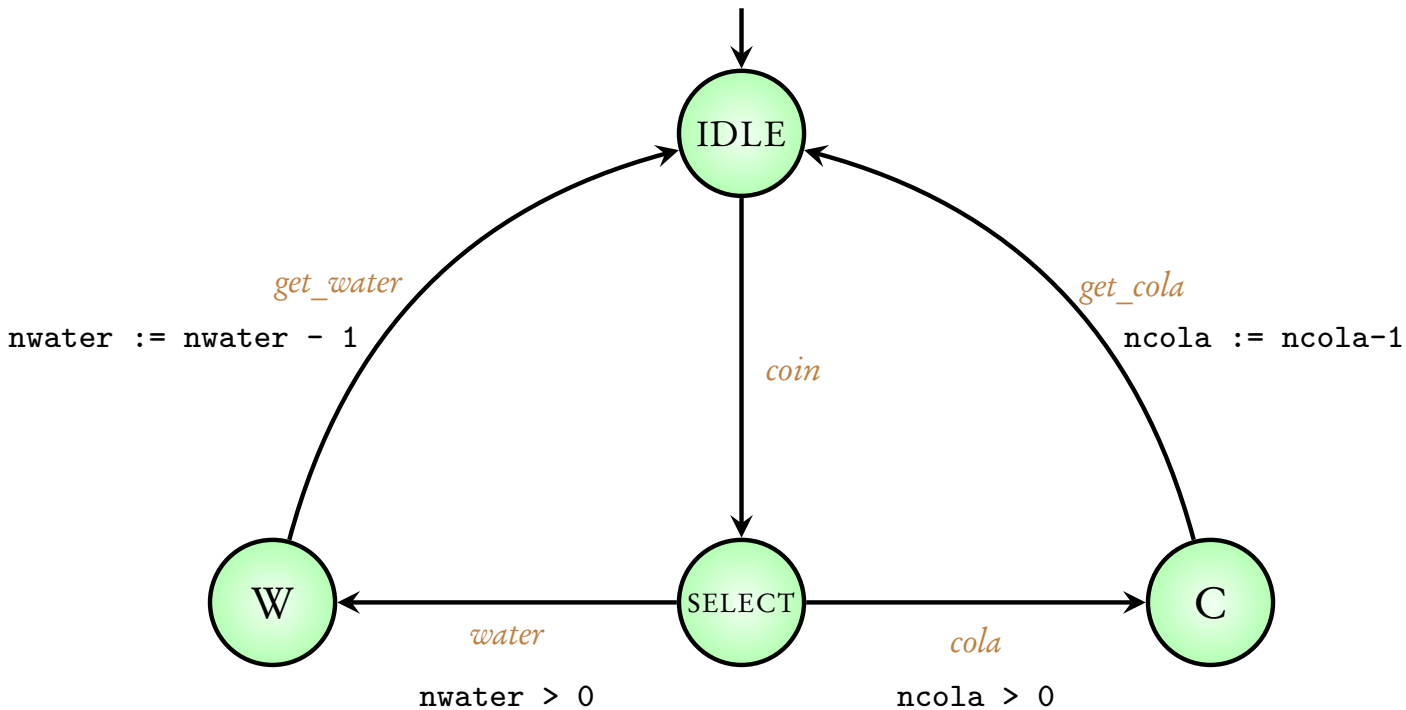
Variables:

`nwater, ncola, max`



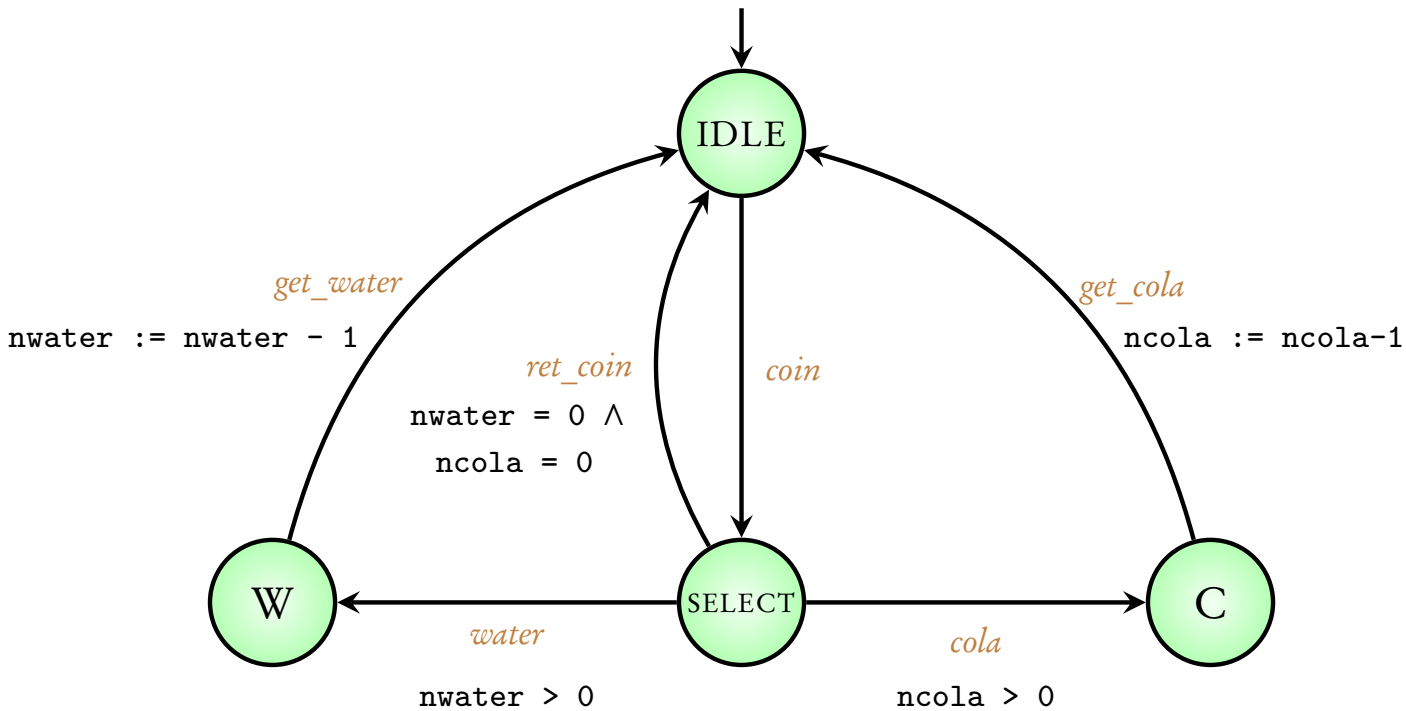
Variables:

`nwater, ncola, max`



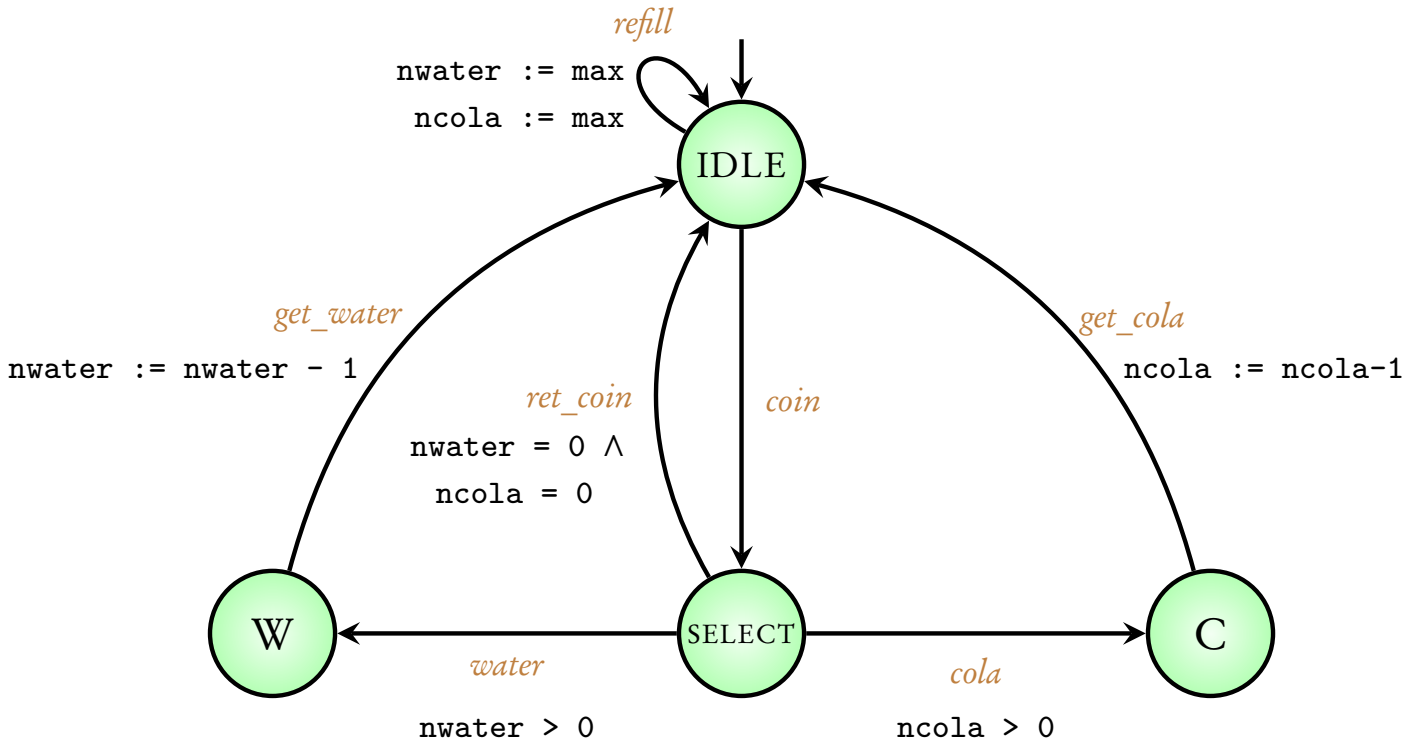
Variables:

`nwater, ncola, max`



Variables:

`nwater, ncola, max`

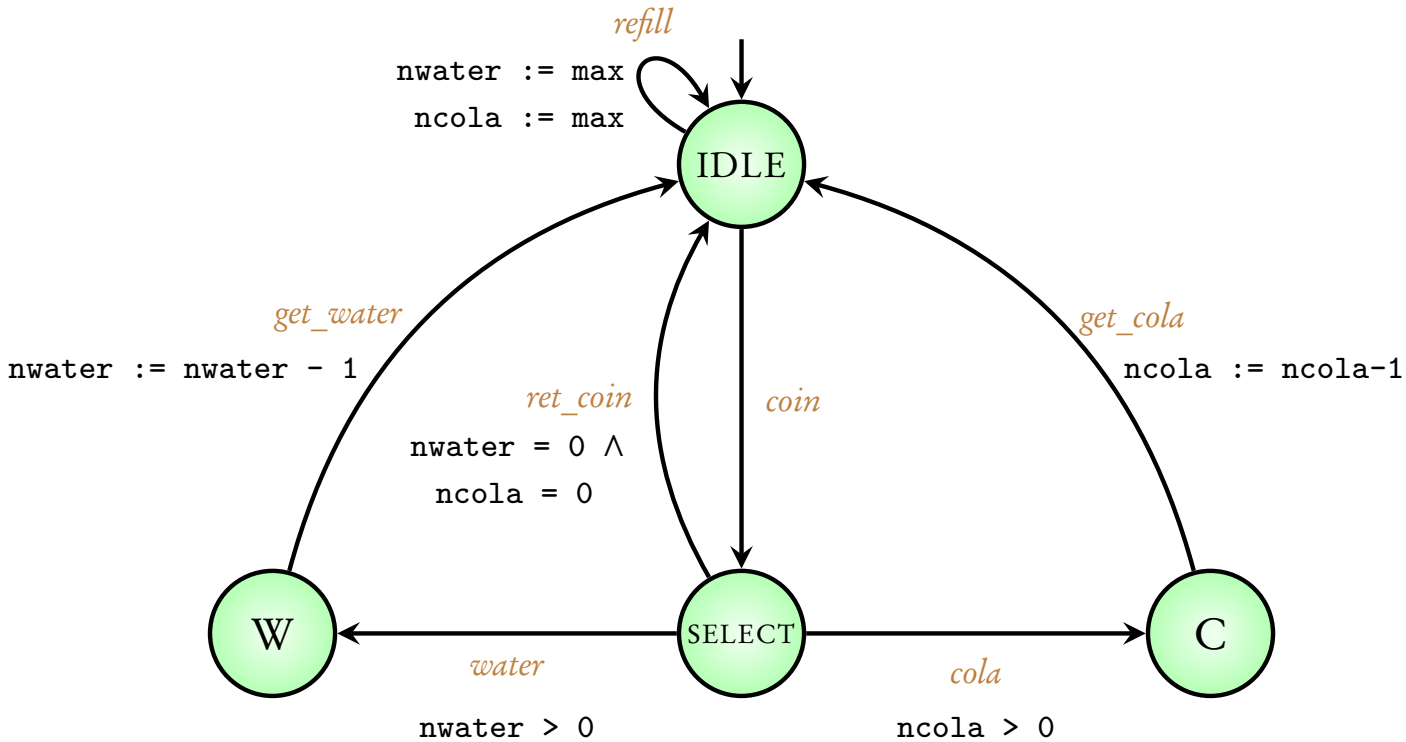


Initial condition:

`nwater = max, ncola = max`

Variables:

`nwater, ncola, max`

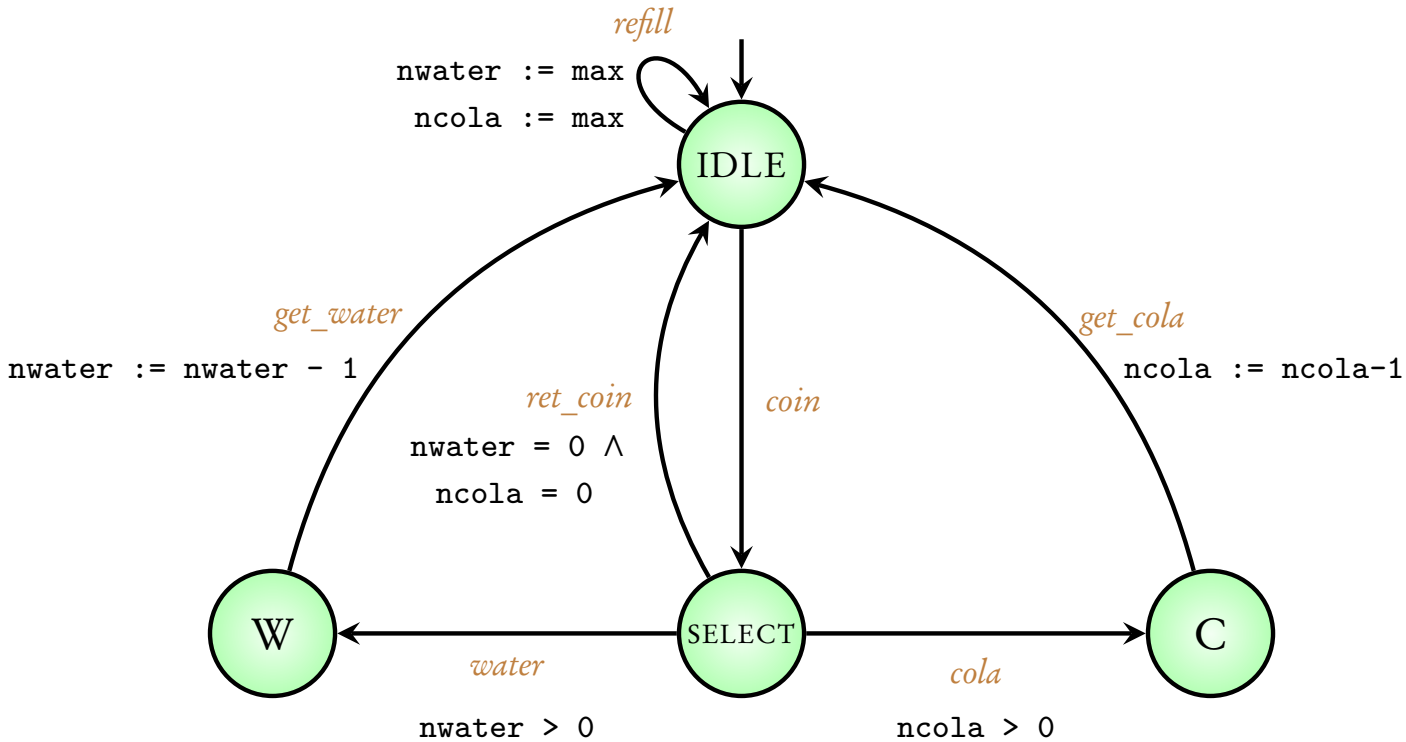


Initial condition:

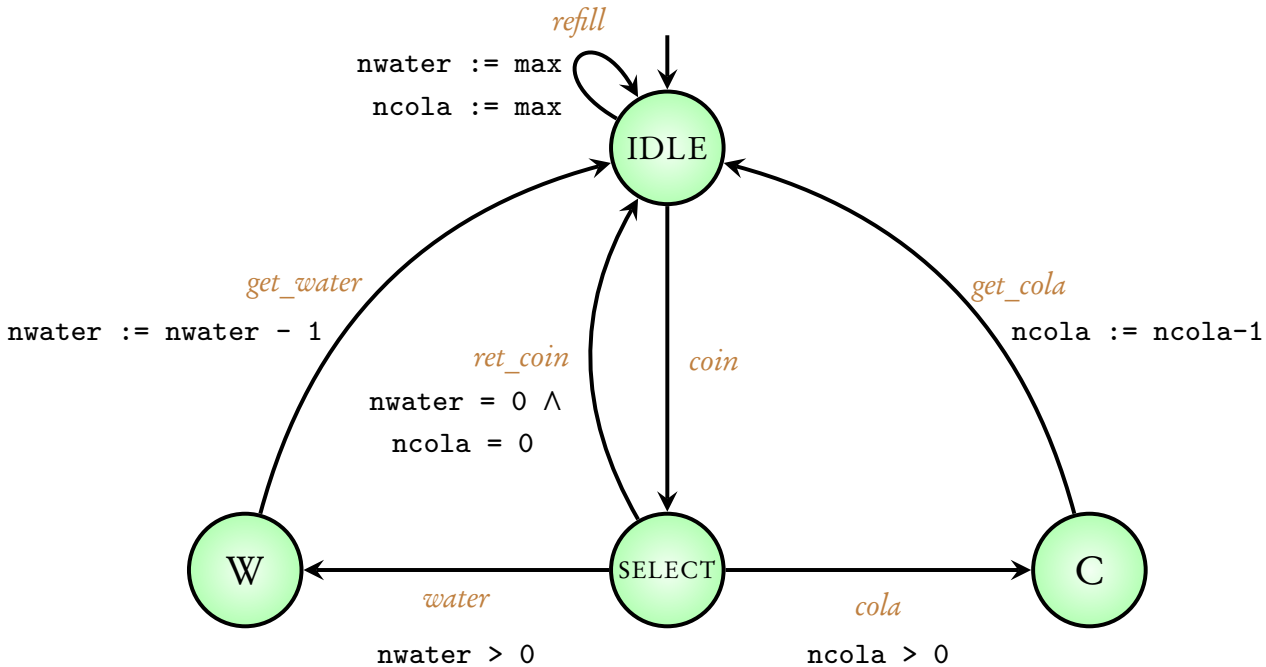
`nwater = max, ncola = max`

Variables:

`nwater, ncola, max`

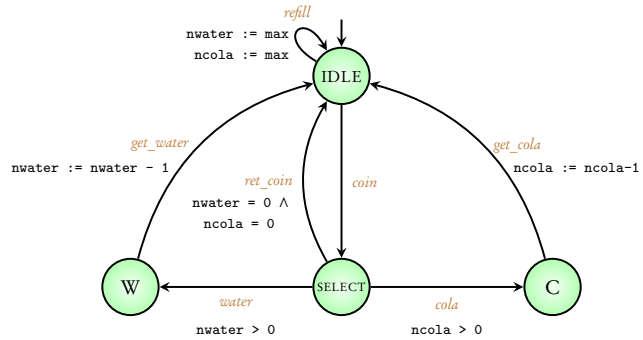


Program graph

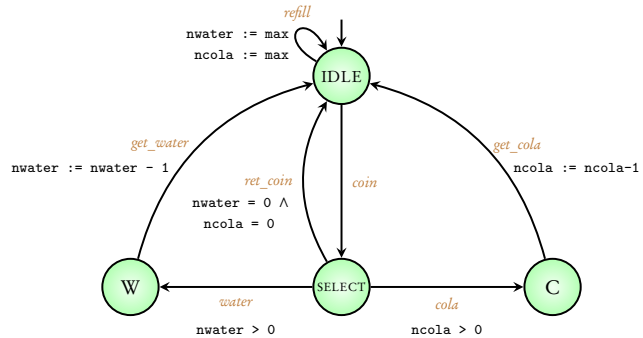


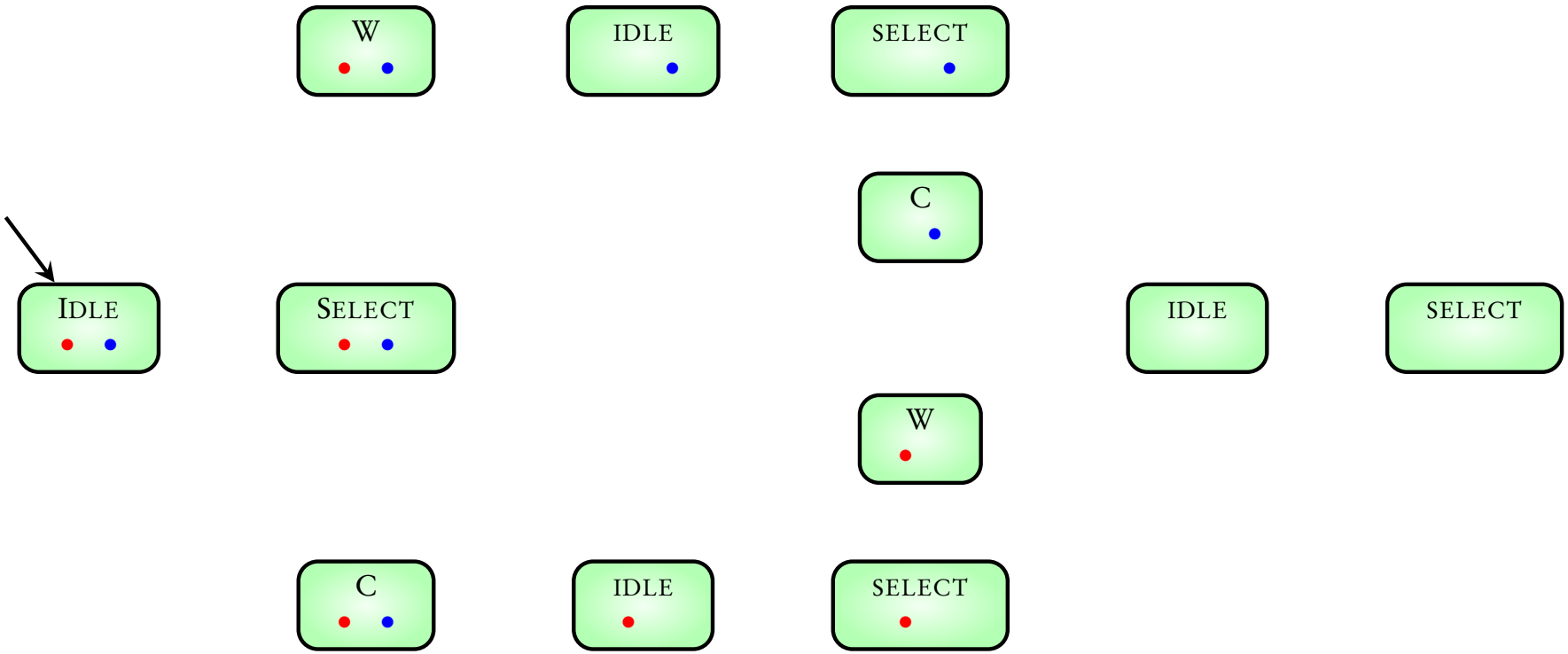
Coming next: Transition system corresponding to `max` = 1

$\texttt{max} = 1$

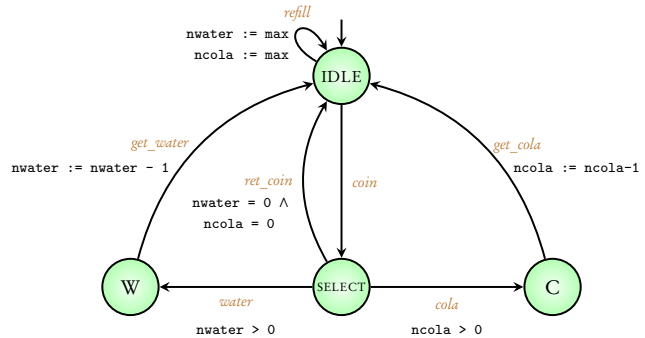

refill
nwater := max
ncola := max
IDLE
get_water
nwater := nwater - 1
ret_coin
nwater = 0 ∧
ncola = 0
coin
get_cola
ncola := ncola-1
W
SELECT
C
water
nwater > 0
cola
ncola > 0

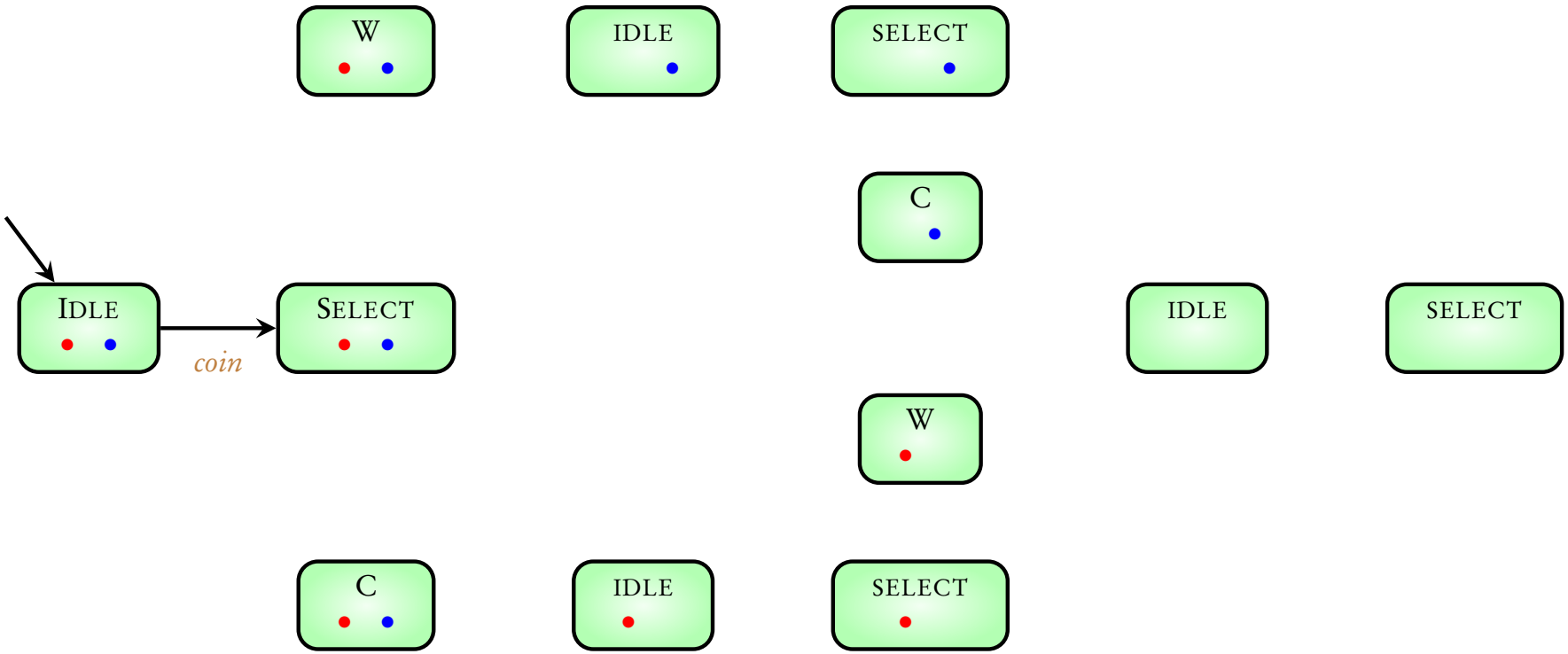
$\texttt{max} = 1$


refill
nwater := max
ncola := max
IDLE
get_water
nwater := nwater - 1
ret_coin
nwater = 0 ∧
ncola = 0
coin
get_cola
ncola := ncola-1
W
SELECT
C
water
nwater > 0
cola
ncola > 0

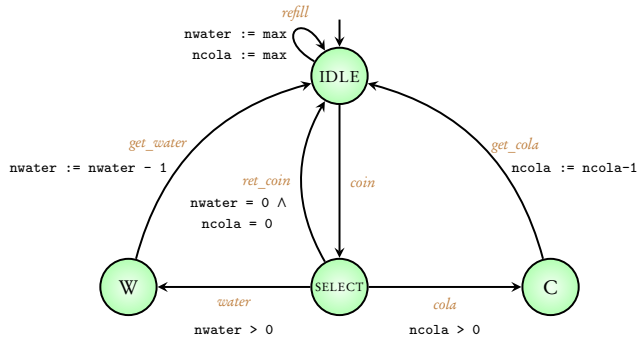

W
IDLE
SELECT
C
IDLE
SELECT
IDLE
SELECT
W
C
IDLE
SELECT

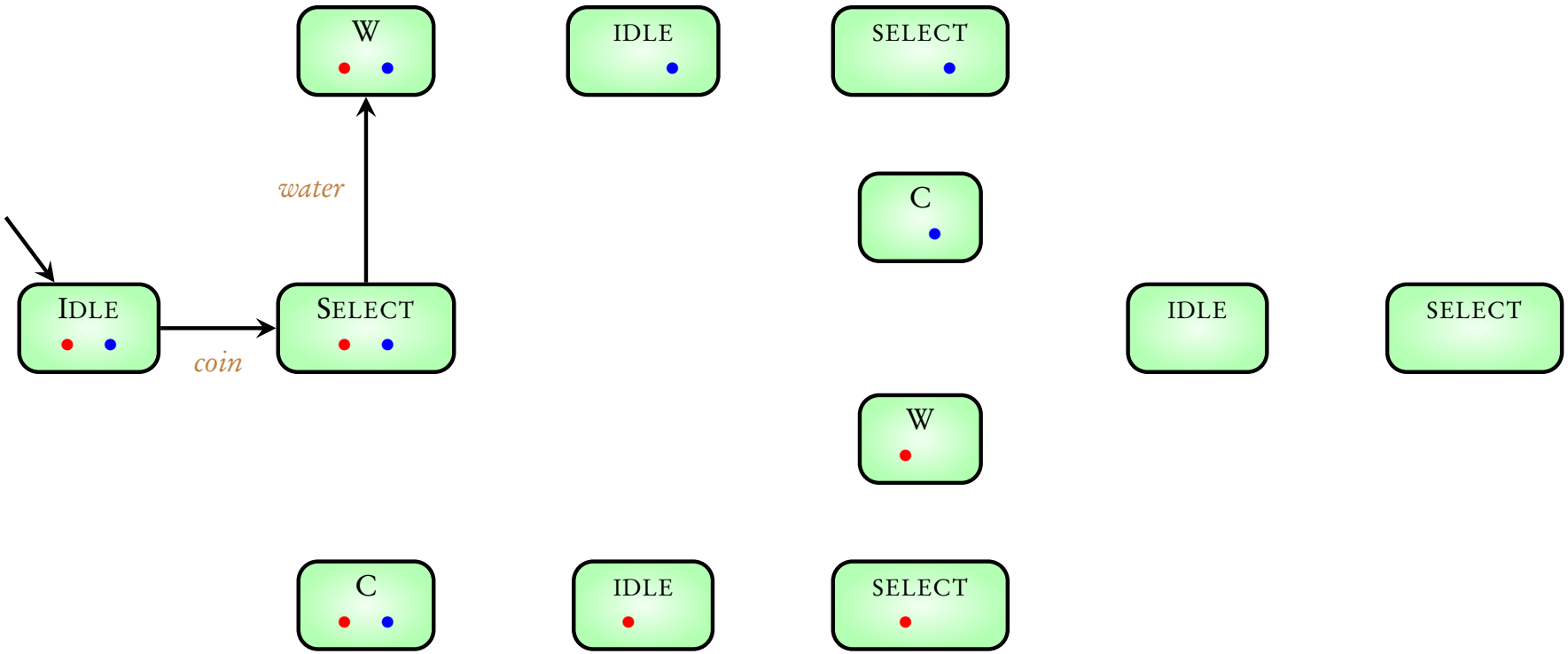
$\texttt{max} = 1$


refill
nwater := max
ncola := max
IDLE
get_water
nwater := nwater - 1
ret_coin
nwater = 0 ∧
ncola = 0
coin
get_cola
ncola := ncola-1
W
SELECT
C
water
nwater > 0
cola
ncola > 0

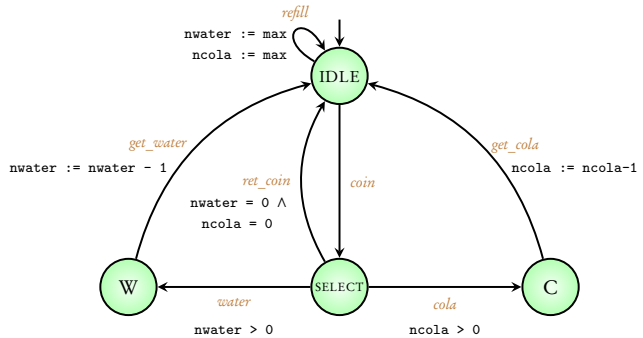

W
IDLE
SELECT
C
IDLE
SELECT
coin
IDLE
SELECT
W
C
IDLE
SELECT

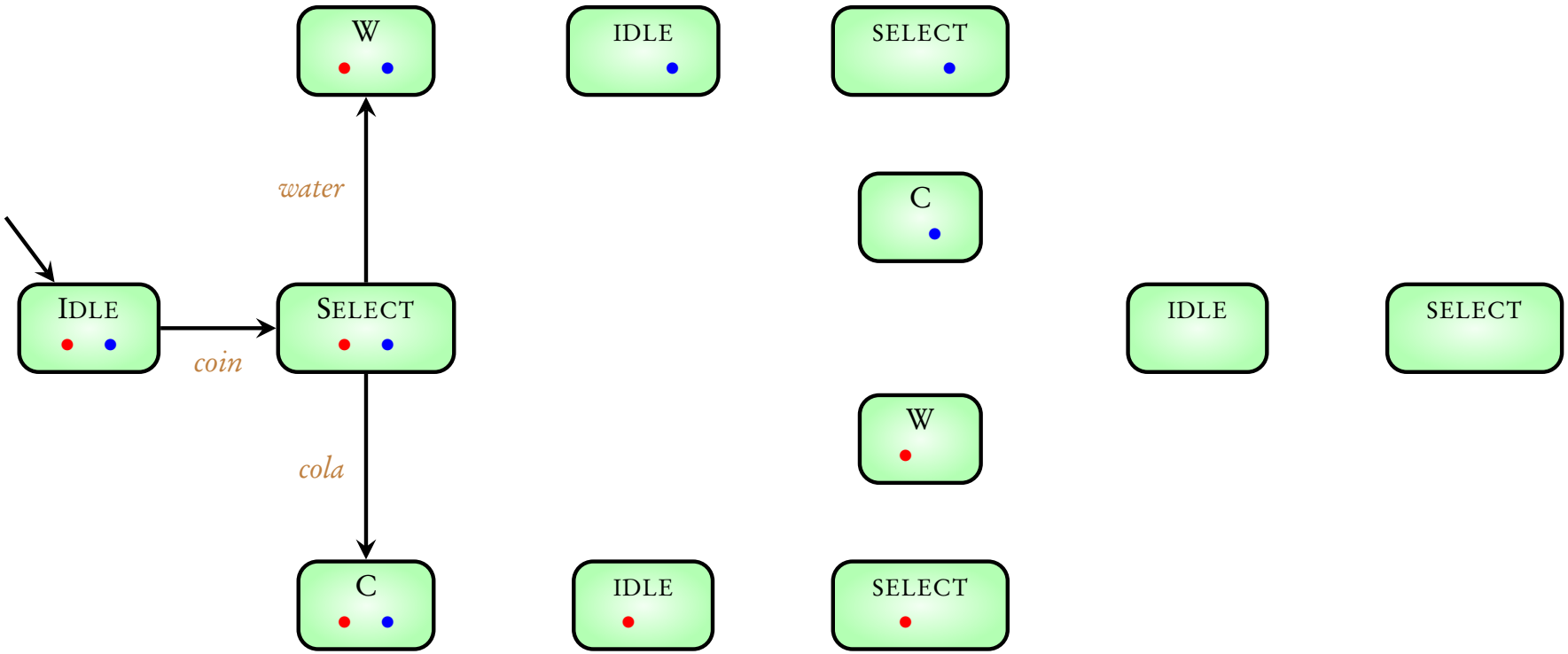
$\texttt{max} = 1$


refill
nwater := max
ncola := max
IDLE
get_water
nwater := nwater - 1
ret_coin
nwater = 0 ∧
ncola = 0
coin
get_cola
ncola := ncola-1
W
SELECT
C
water
nwater > 0
cola
ncola > 0

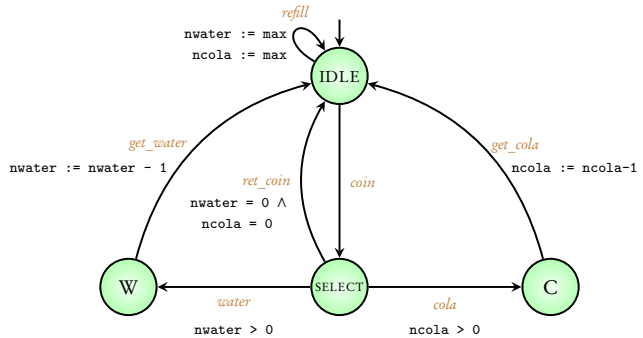

W
IDLE
SELECT
water
C
IDLE
SELECT
IDLE
coin
SELECT
W
C
IDLE
SELECT

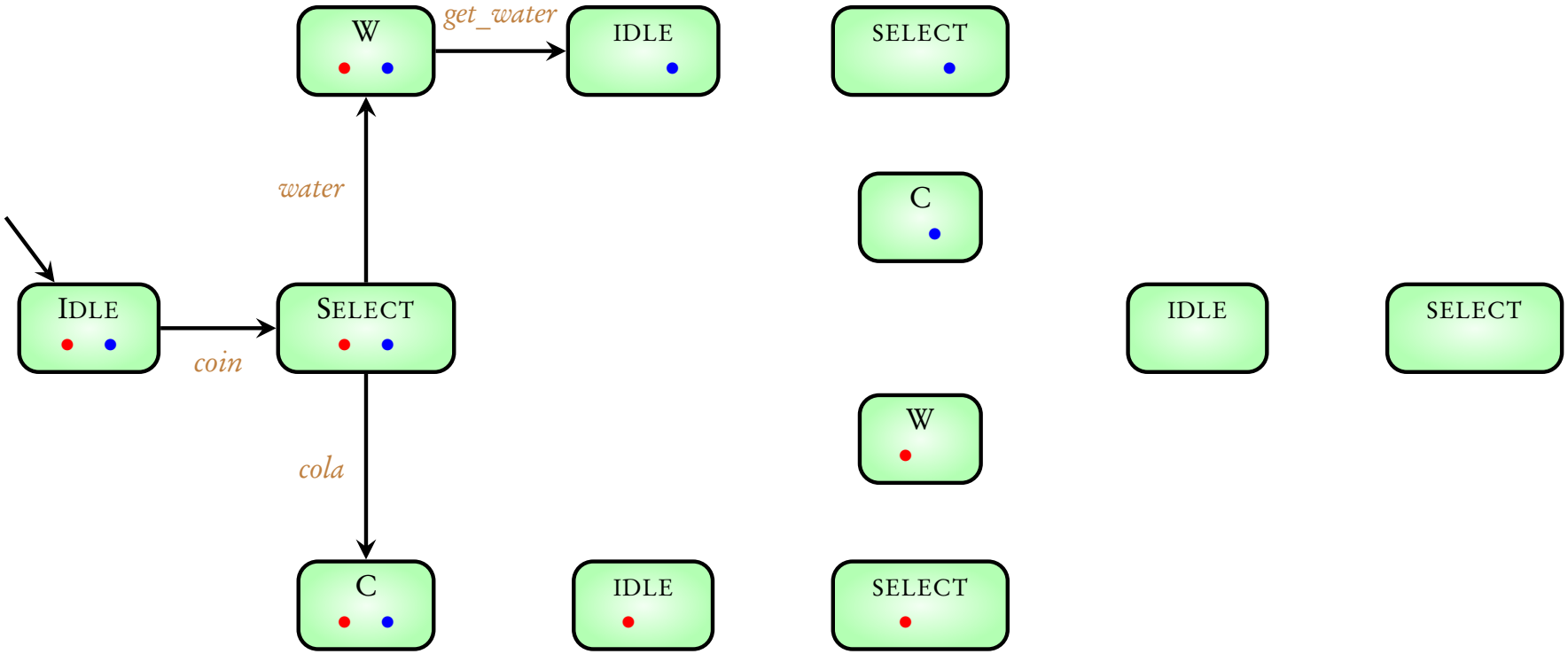
$\texttt{max} = 1$


refill
nwater := max
ncola := max
IDLE
get_water
nwater := nwater - 1
ret_coin
nwater = 0 ∧
ncola = 0
coin
get_cola
ncola := ncola-1
W
SELECT
C
water
nwater > 0
cola
ncola > 0

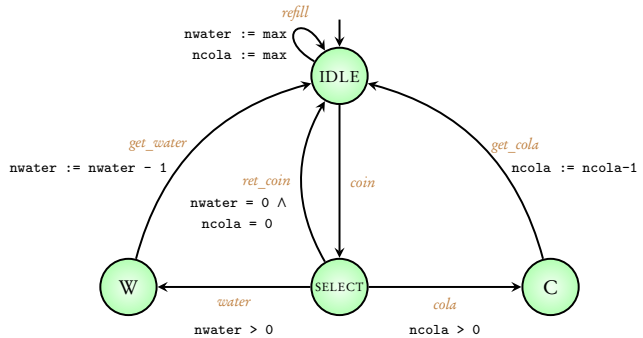

W
IDLE
SELECT
water
C
IDLE
SELECT
coin
cola
W
C
IDLE
SELECT

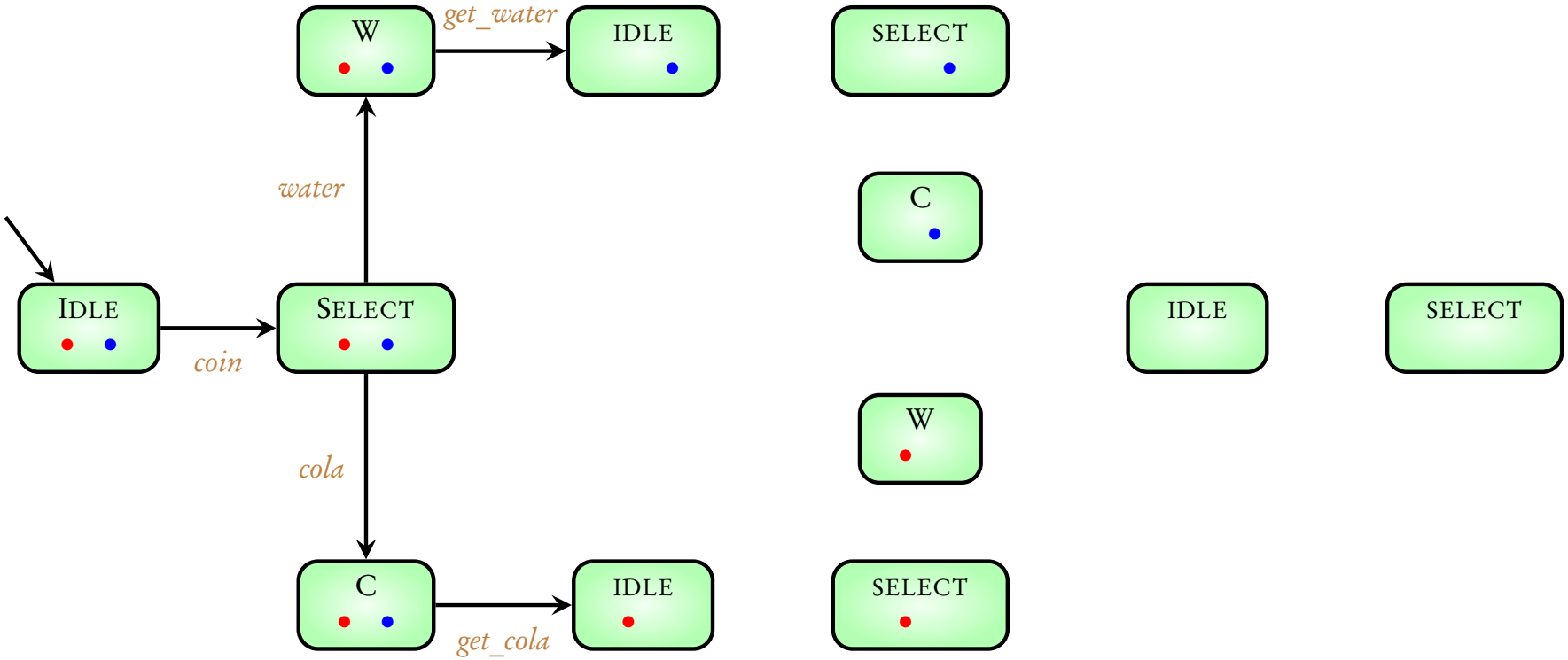
$\texttt{max} = 1$


refill
nwater := max
ncola := max
IDLE
get_water
nwater := nwater - 1
ret_coin
nwater = 0 ∧
ncola = 0
coin
get_cola
ncola := ncola-1
W
SELECT
C
water
nwater > 0
cola
ncola > 0

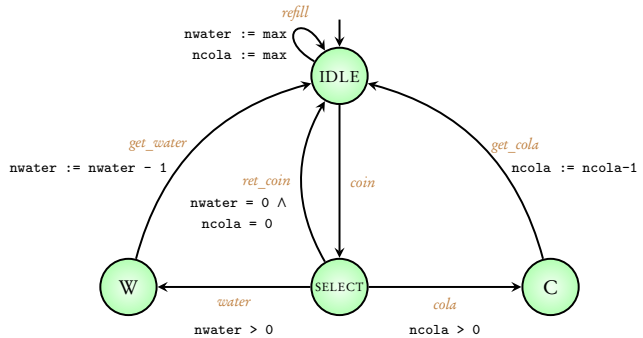

W
get_water
IDLE
SELECT
water
C
IDLE
coin
SELECT
IDLE
SELECT
cola
W
C
IDLE
SELECT

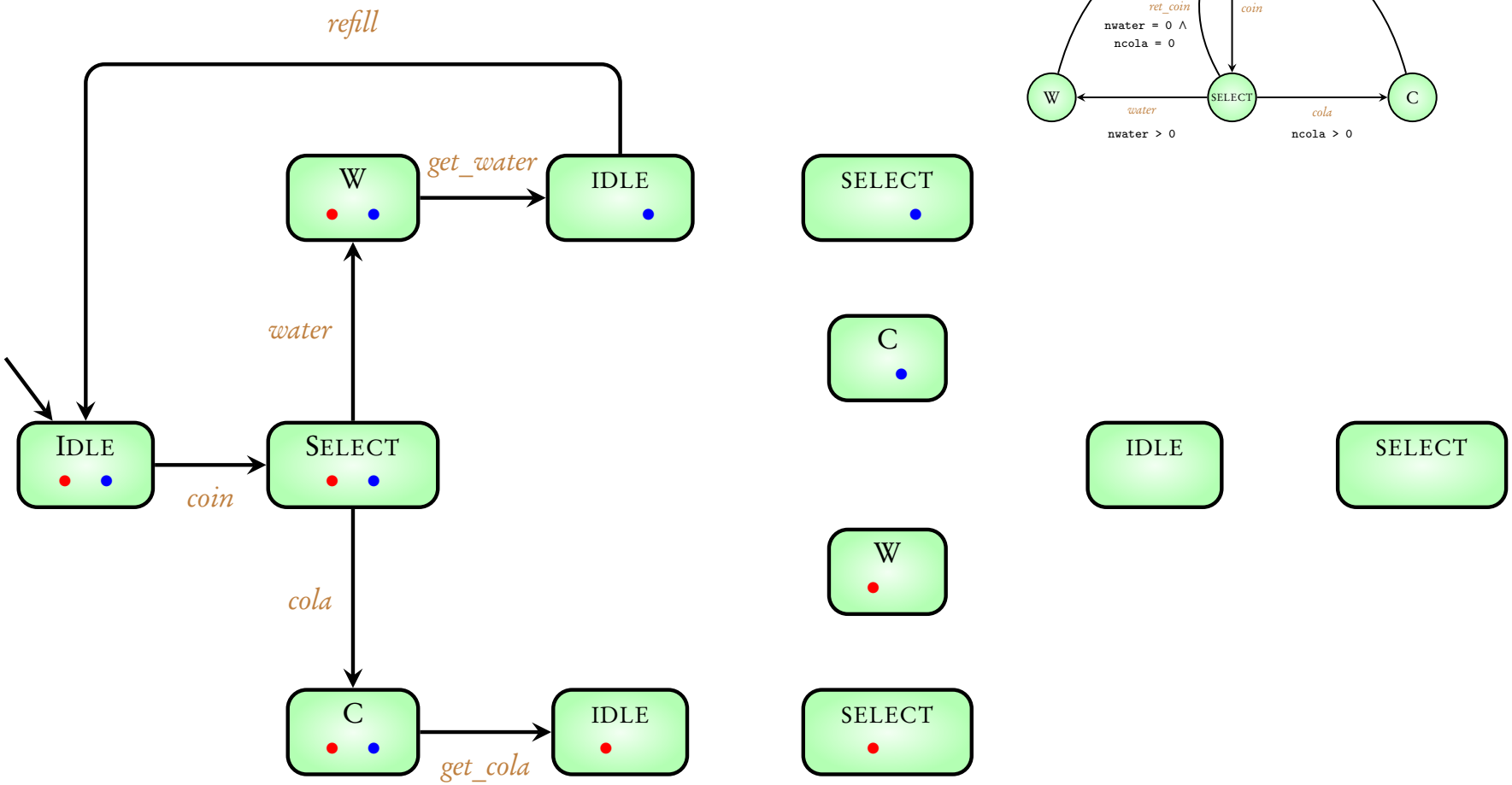
$\texttt{max} = 1$


refill
nwater := max
ncola := max
IDLE
get_water
nwater := nwater - 1
ret_coin
nwater = 0 ∧
ncola = 0
coin
get_cola
ncola := ncola-1
W
SELECT
C
water
nwater > 0
cola
ncola > 0

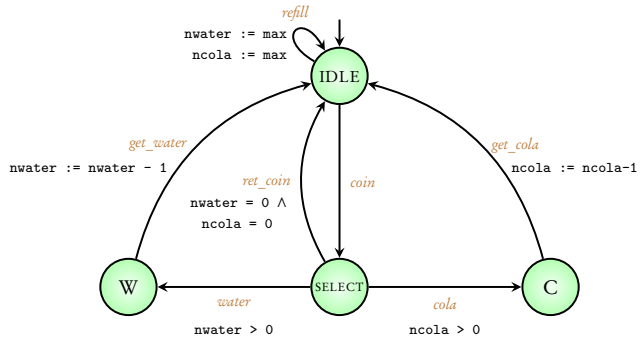

W
get_water
IDLE
SELECT
water
C
IDLE
coin
SELECT
IDLE
SELECT
cola
W
C
IDLE
SELECT
get_cola

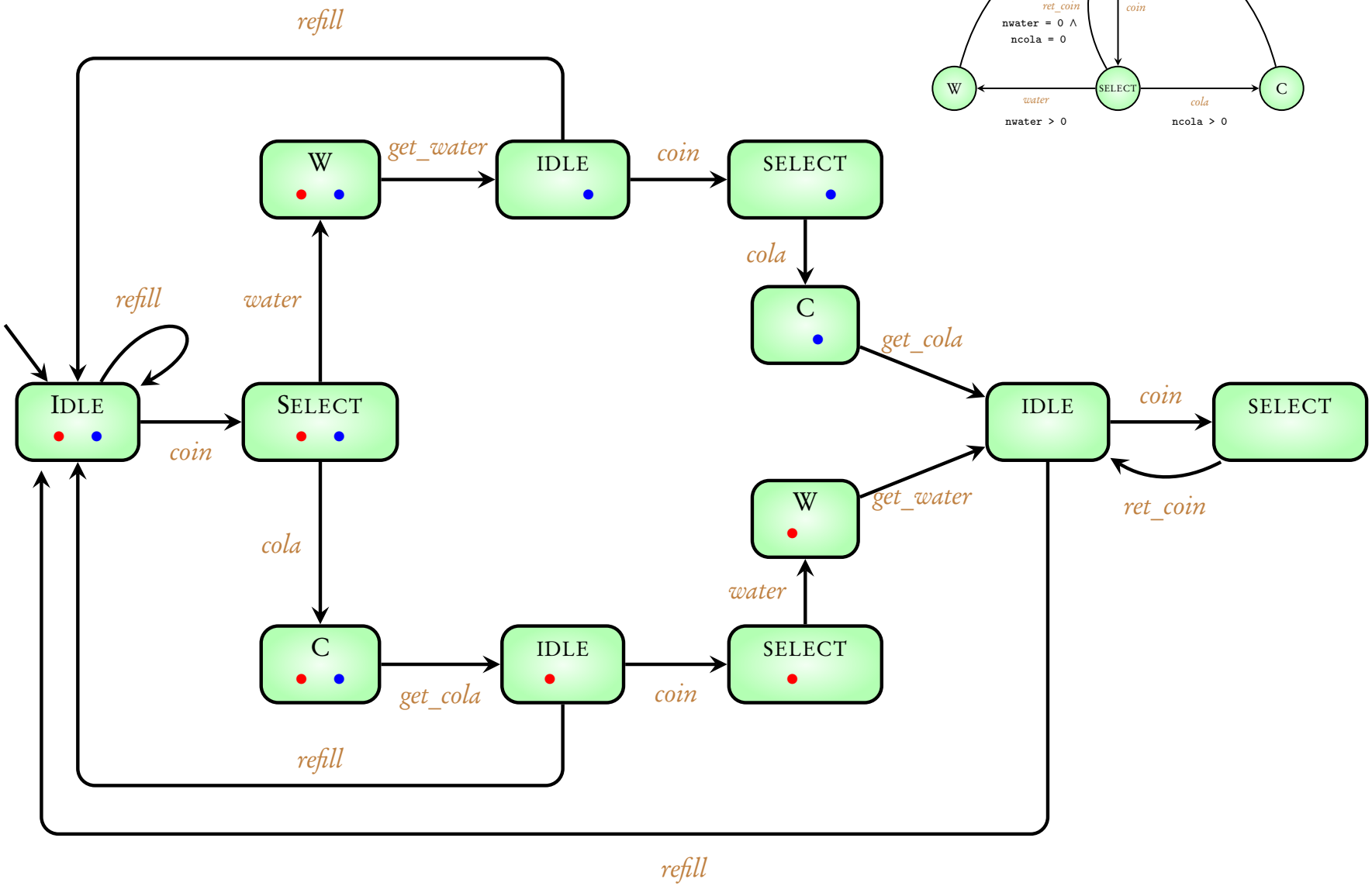
$\texttt{max} = 1$


refill
nwater := max
ncola := max
IDLE
get_water
nwater := nwater - 1
ret_coin
nwater = 0 ∧
ncola = 0
coin
get_cola
ncola := ncola-1
W
SELECT
C
water
nwater > 0
cola
ncola > 0


refill
W
get_water
IDLE
SELECT
water
C
IDLE
SELECT
IDLE
coin
SELECT
cola
W
C
IDLE
SELECT
get_cola

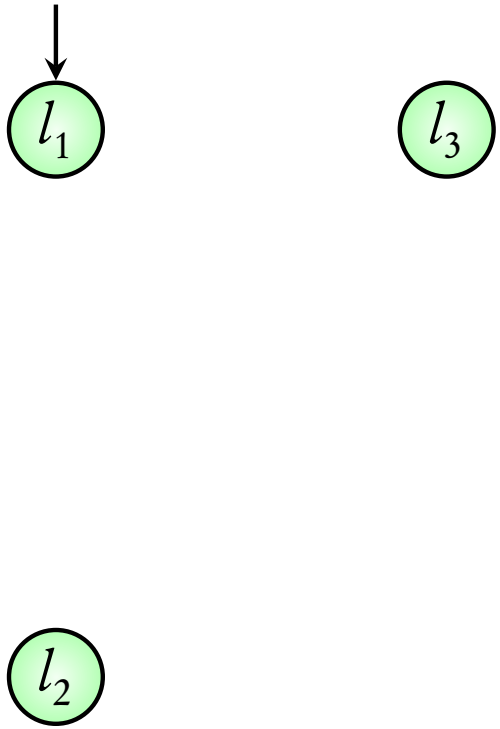
$\texttt{max} = 1$


refill
nwater := max
ncola := max
IDLE
get_water
nwater := nwater - 1
ret_coin
nwater = 0 ∧
ncola = 0
coin
get_cola
ncola := ncola-1
W
SELECT
C
water
nwater > 0
cola
ncola > 0

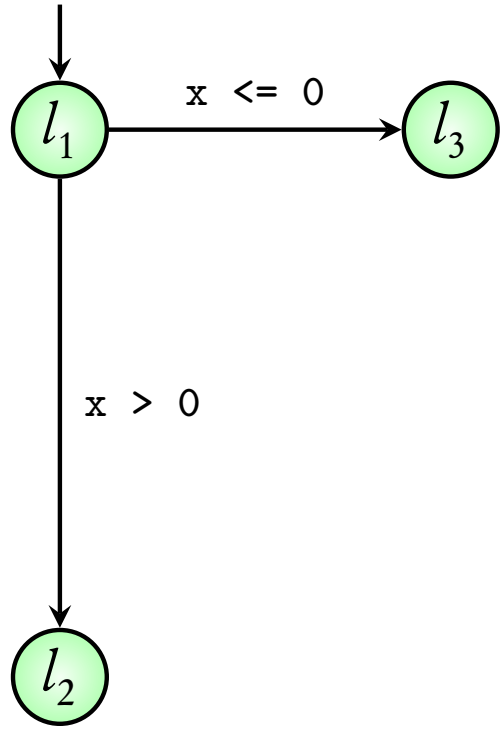

refill
W
get_water
IDLE
coin
SELECT
cola
C
get_cola
refill
water
IDLE
coin
SELECT
coin
SELECT
cola
W
get_water
ret_coin
water
C
IDLE
SELECT
get_cola
coin
refill
refill

```
    ...
while (x > 0)
    if (x mod 2 = 0)
        x := x - 2
    else  x:= x - 1
    ...
```

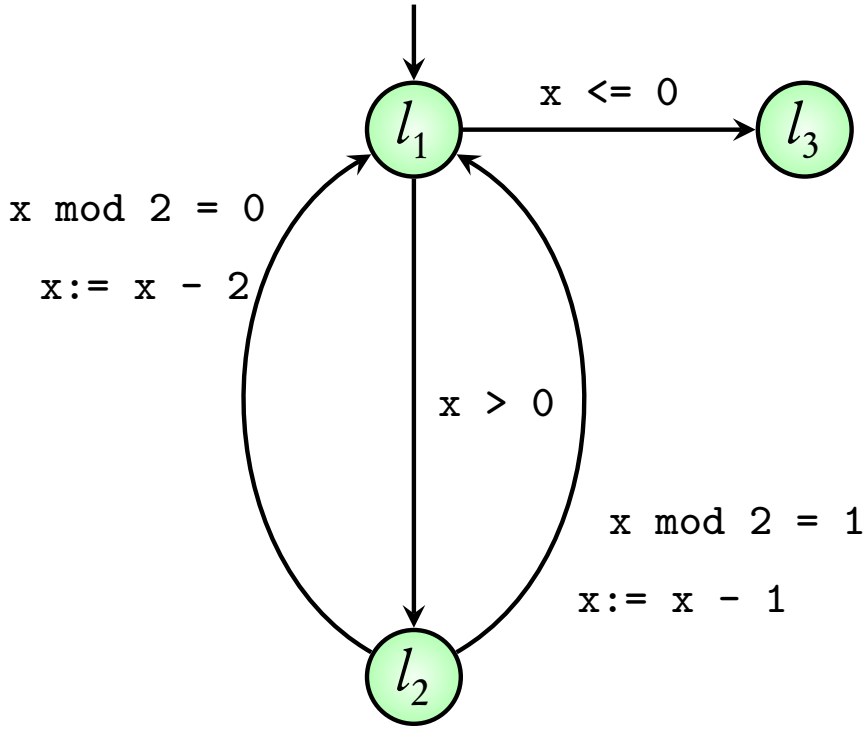
```
    ...
l1: while ( x > 0 )
        if ( x mod 2 = 0)
l2:         x := x - 2
        else  x:= x - 1
        ...
l3:
```



```
...
l1: while ( x > 0 )
      if ( x mod 2 = 0)
l2:       x := x - 2
      else  x:= x - 1
...
l3:
```



```
...
l1: while ( x > 0 )
      if ( x mod 2 = 0)
l2:       x := x - 2
      else  x:= x - 1
...
l3:
```



...

l_1: **while** (x > 0)

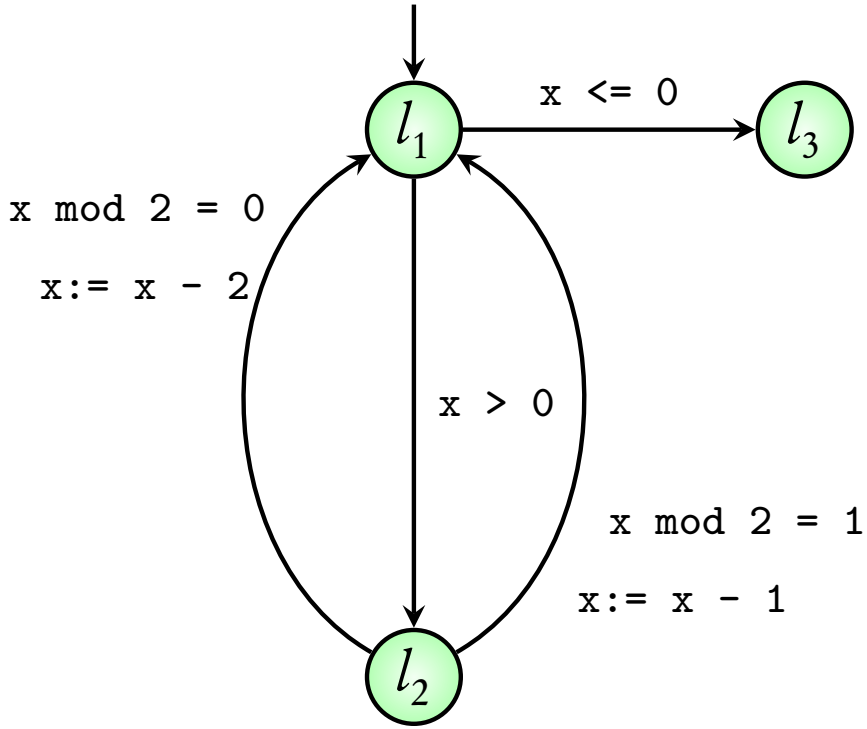
if (x mod 2 = 0)

l_2: x := x - 2

else x:= x - 1

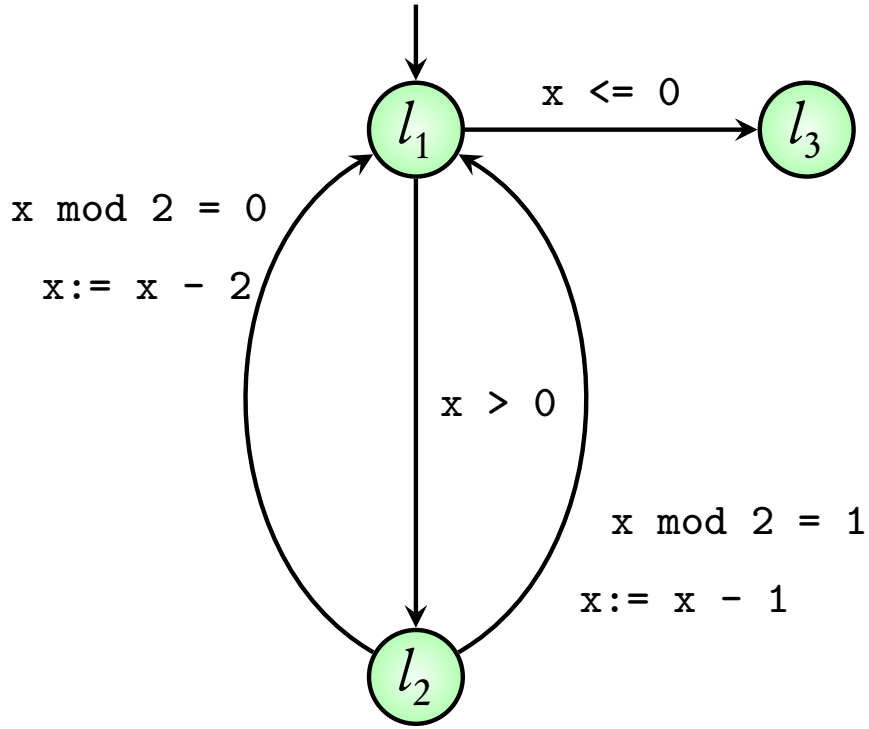
...

l_3:



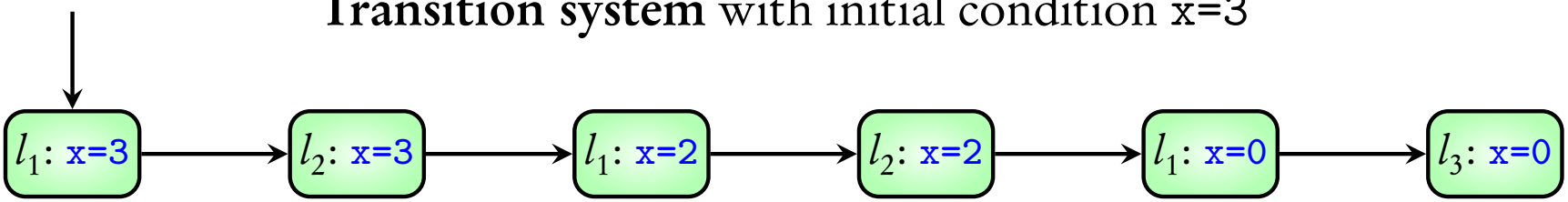
Program graph

```
. . .
l1: while ( x > 0 )
      if ( x mod 2 = 0)
l2:       x := x - 2
      else  x:= x - 1
. . .
l3:
```



Program graph

Transition system with initial condition x=3



Summary

Data-dependent programs

Program graphs

Transition systems of program graphs

Reference: Principles of Model Checking, *Baier and Katoen*, MIT Press (2008)

Pages 29 - 34

Week-1: Introduction to model checking

B. Srivathsan

Chennai Mathematical Institute

NPTEL-course

July - November 2015

Module 4:

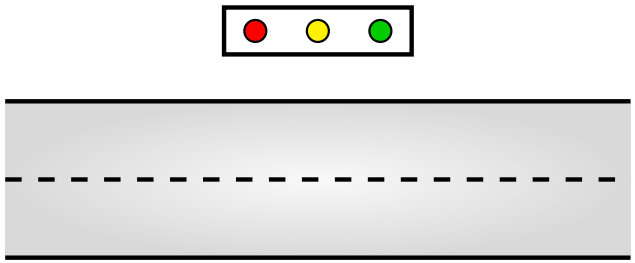
Modeling concurrent systems

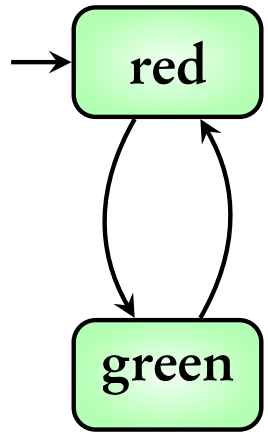
Concurrent systems

Independent

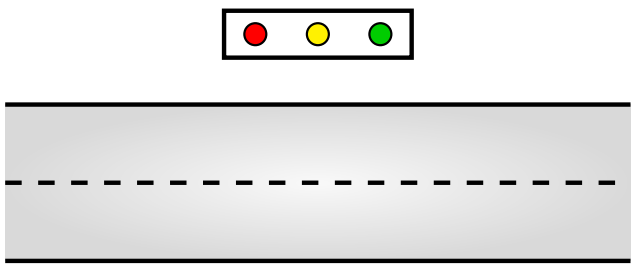
Shared variables

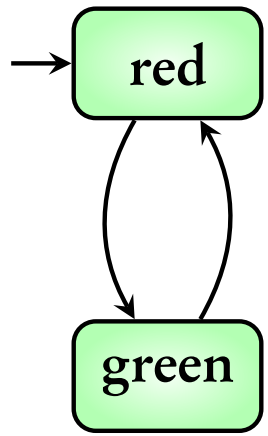
Shared actions



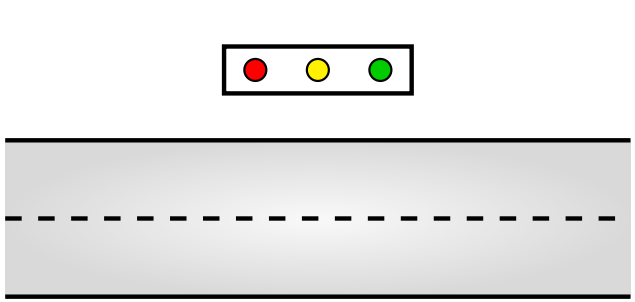


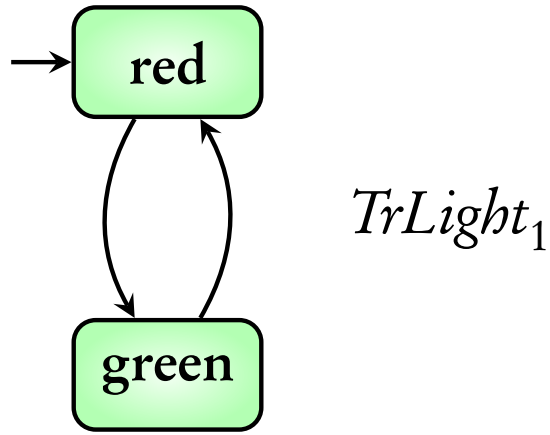
$TrLight_1$



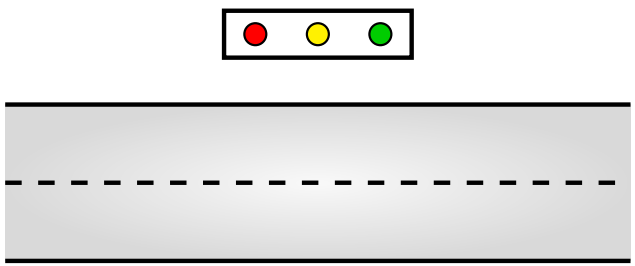


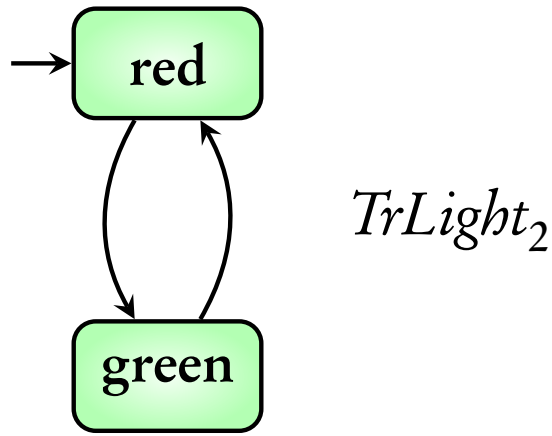
$TrLight_2$





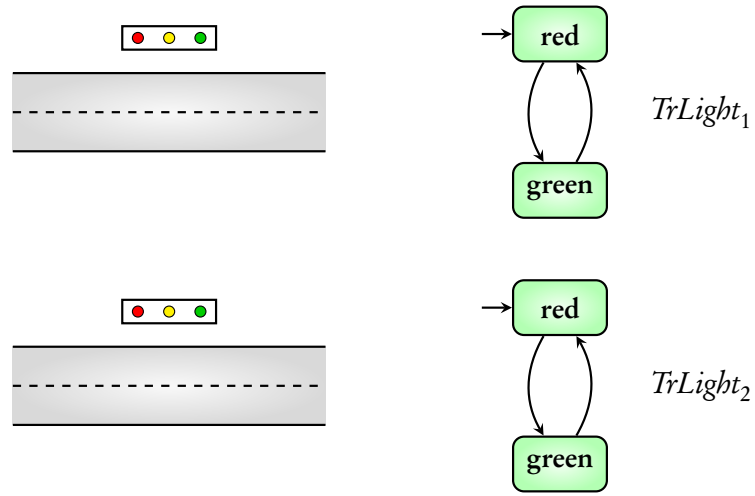
$TrLight_1$

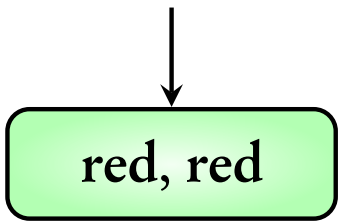


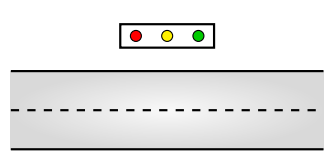


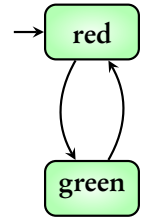
$TrLight_2$

What is the transition system for the **joint behaviour**?

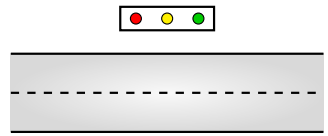

red
green
$TrLight_1$
red
green
$TrLight_2$

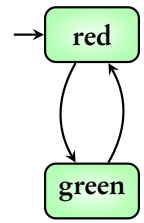




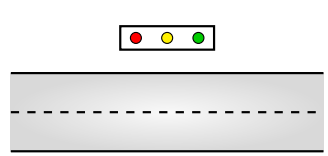


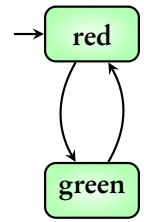
$TrLight_1$



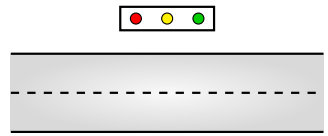


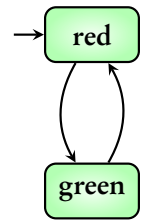
$TrLight_2$



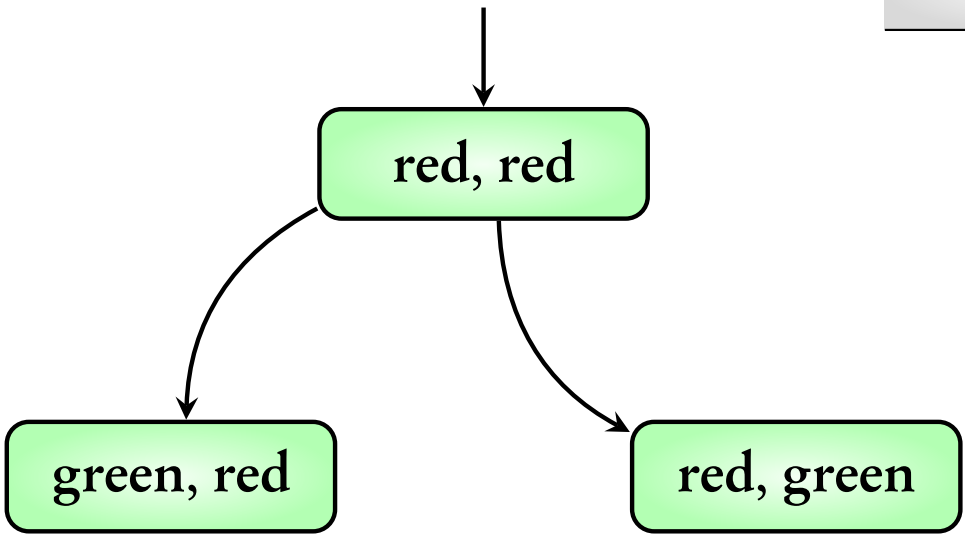


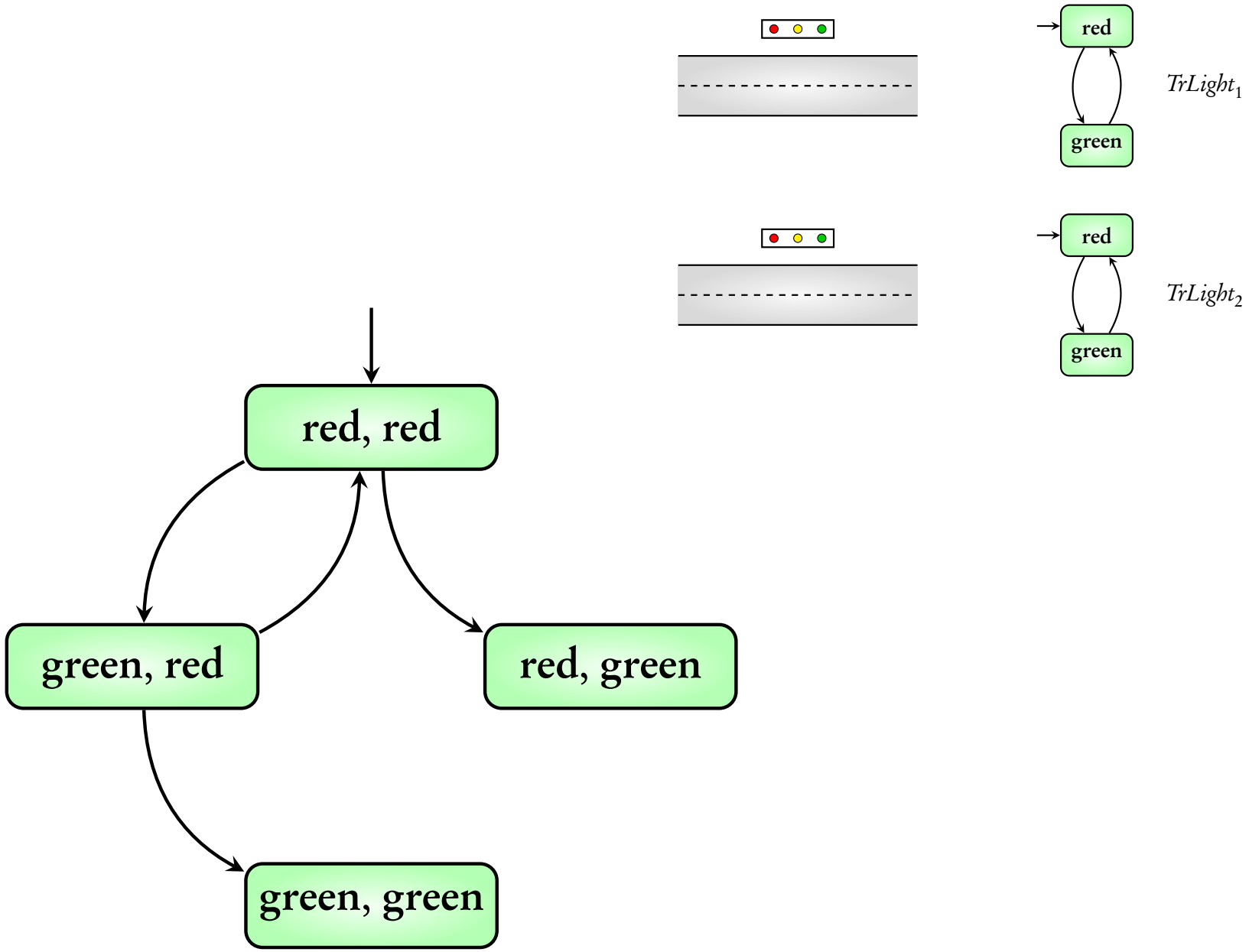
$TrLight_1$

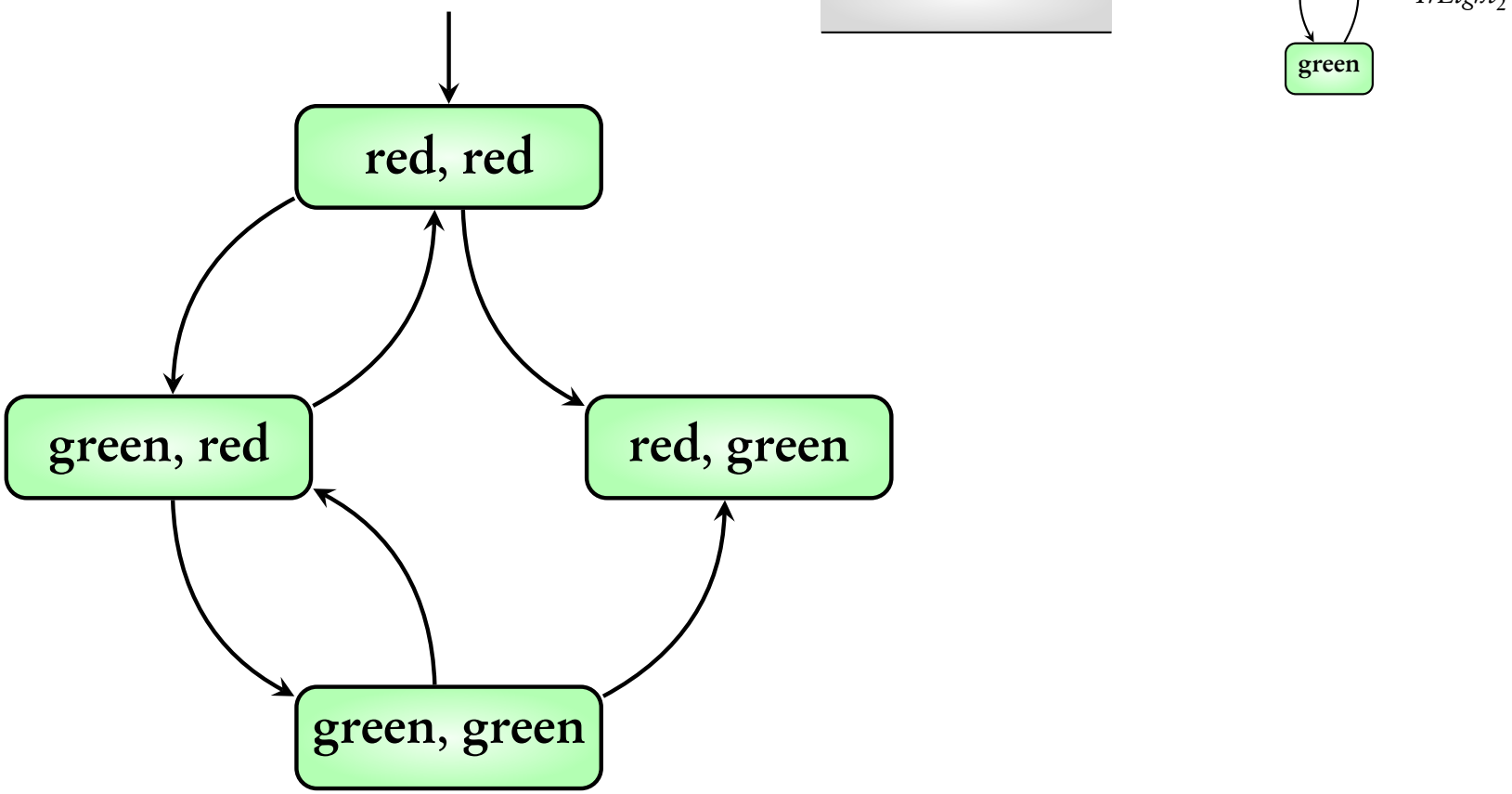


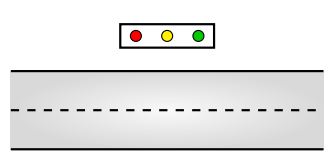


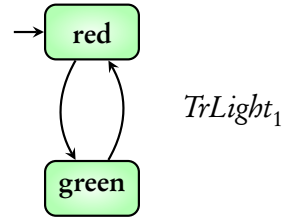
$TrLight_2$

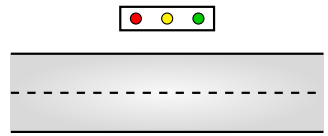


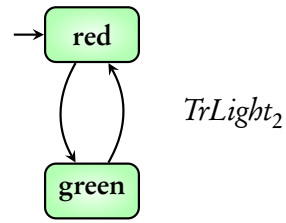

red
green
$TrLight_1$
red
green
$TrLight_2$
red, red
green, red
red, green
green, green

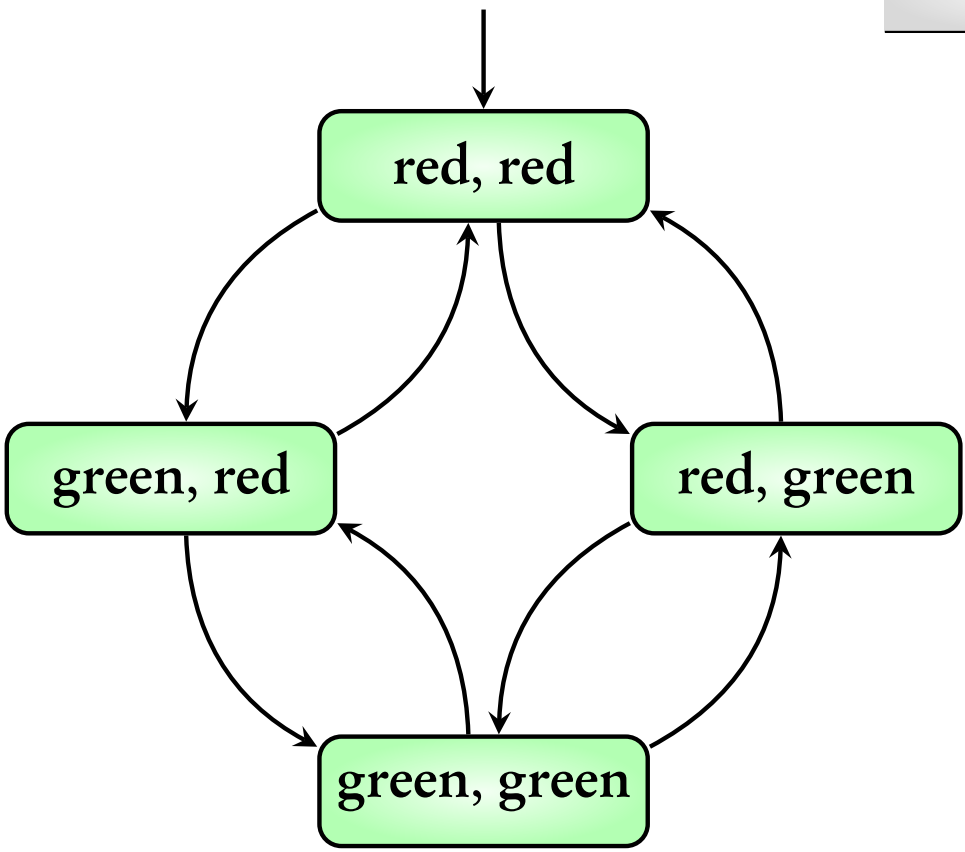

red, red
green, red
red, green
green, green

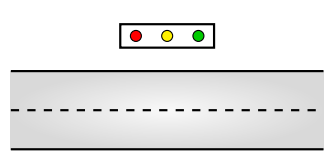


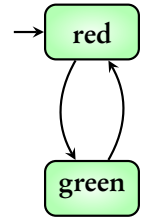

red
green
$TrLight_1$



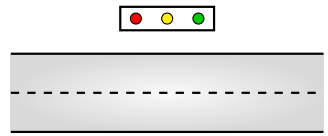

red
green
$TrLight_2$

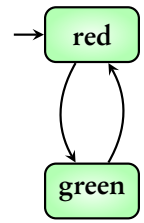




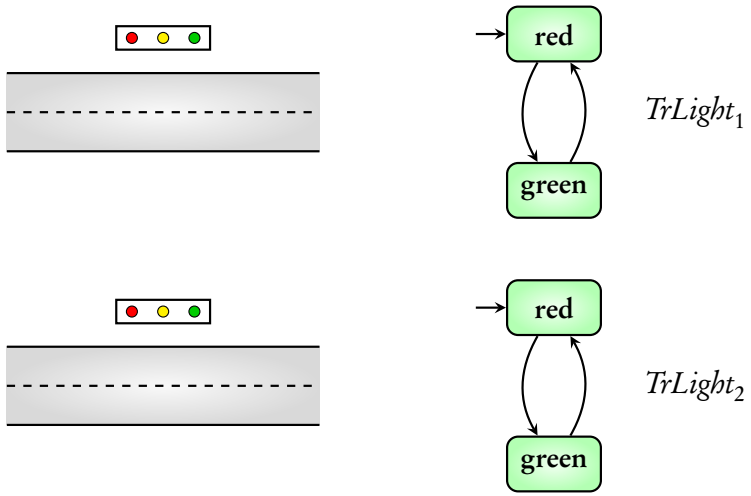


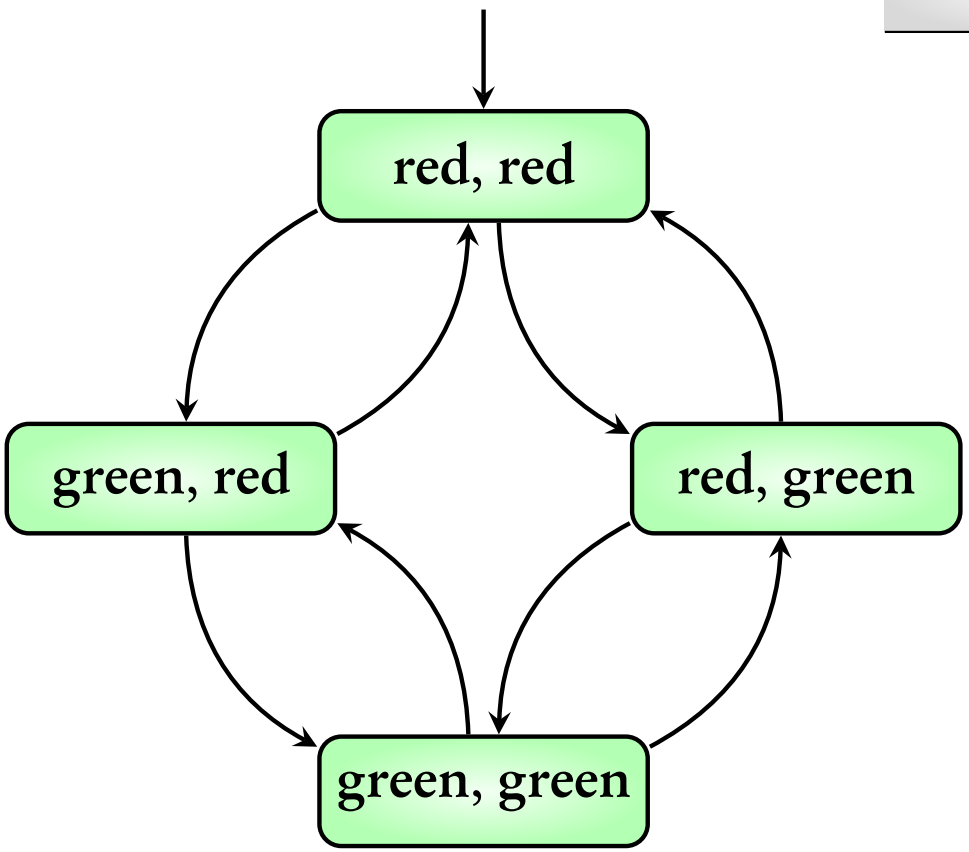
$TrLight_1$



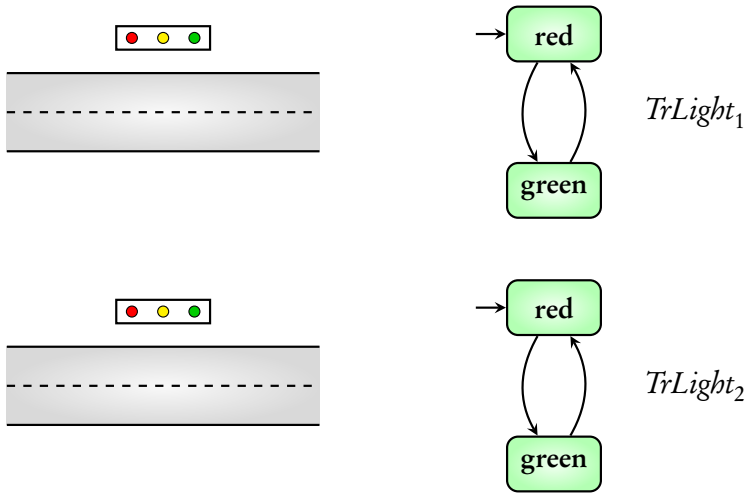


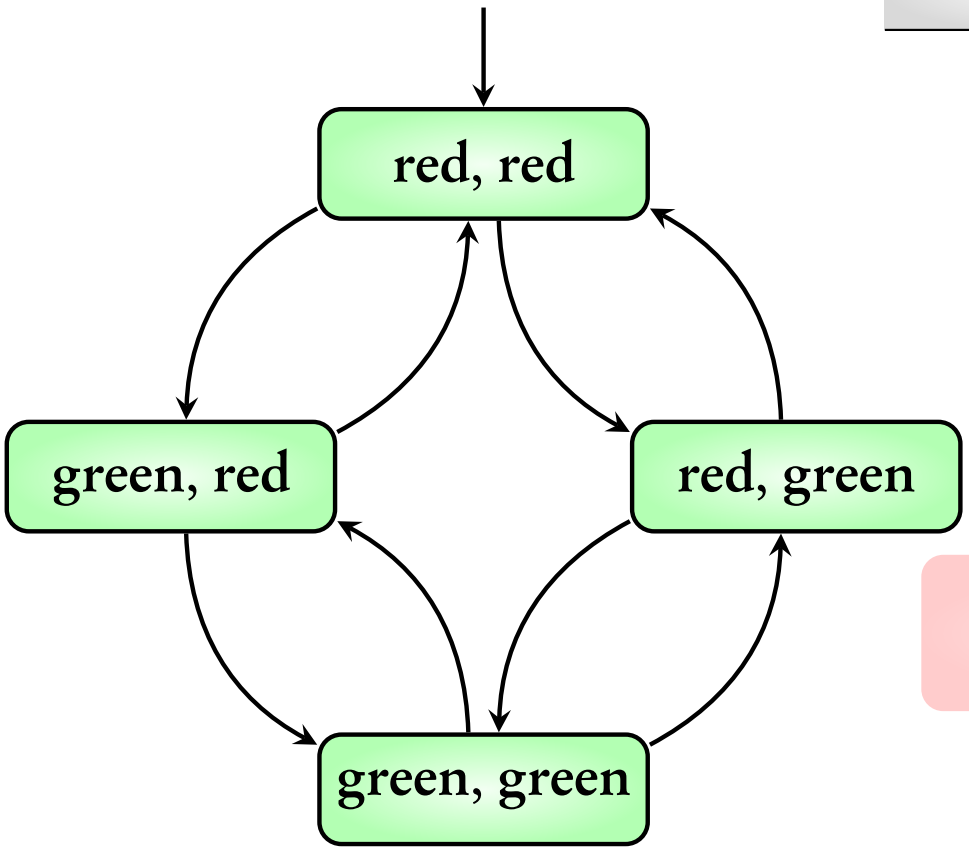
$TrLight_2$





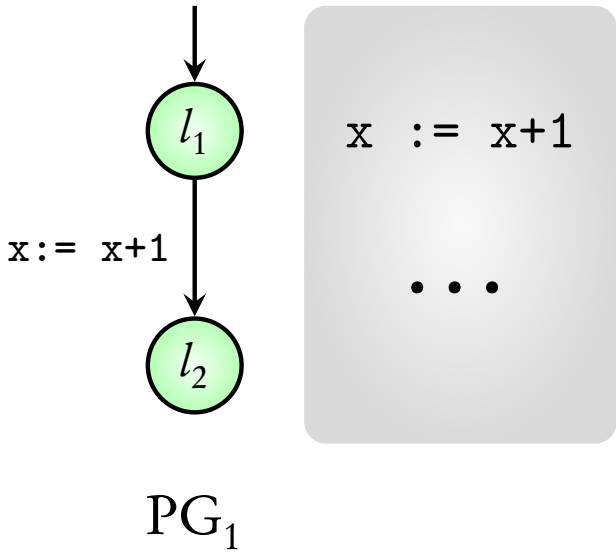
$$TrLight_1 \;|||\; TrLight_2$$



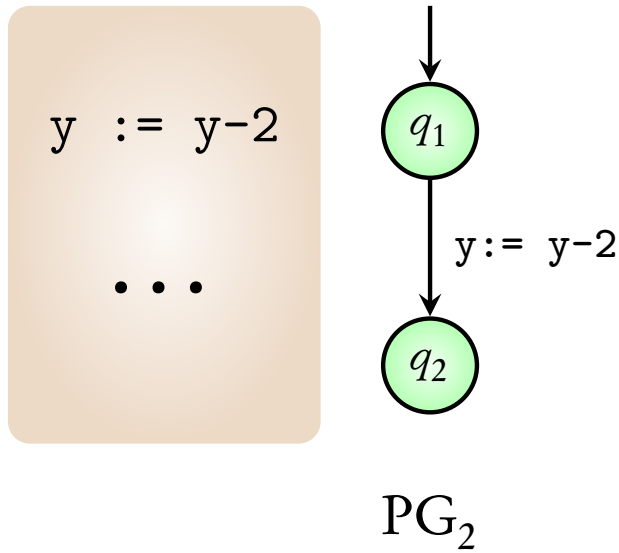


$$TrLight_1 \;|||\; TrLight_2$$

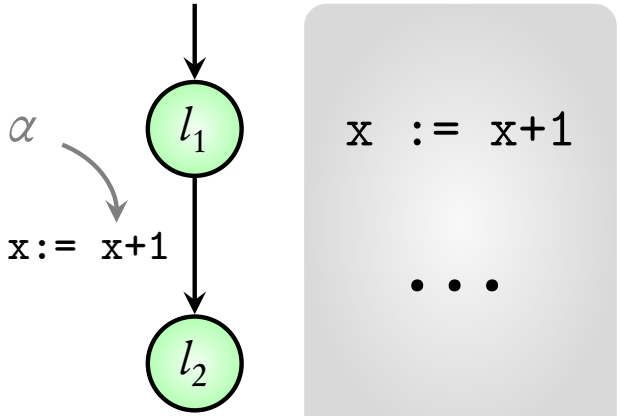
||| : Interleaving operator



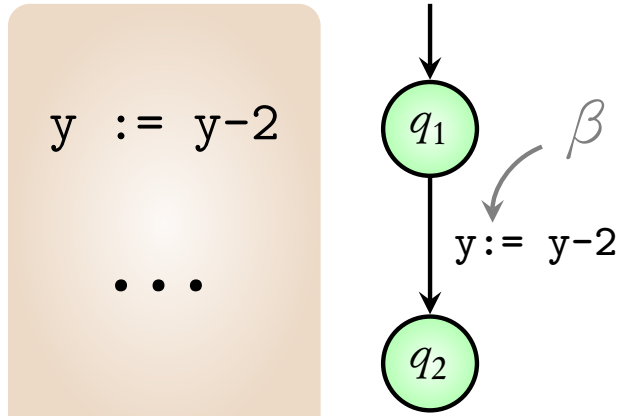
PG_1



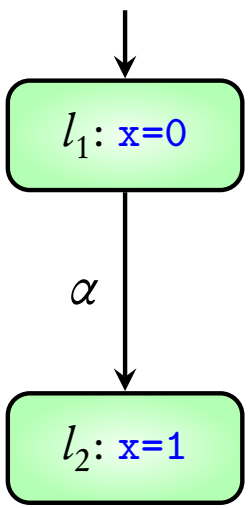
PG_2



PG_1

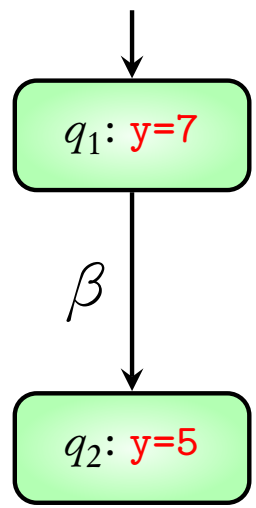


PG_2



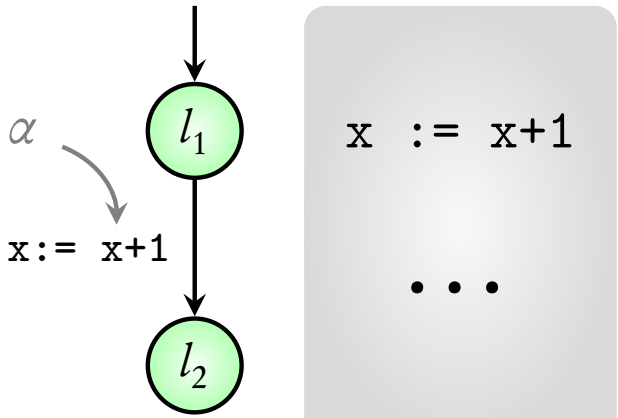
TS_1

(*initially* x=0)

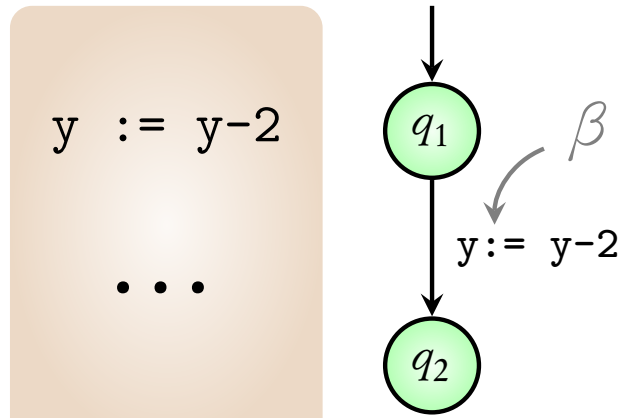


TS_2

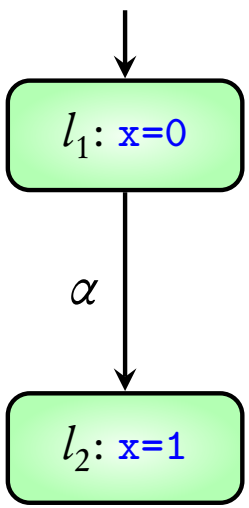
(*initially* y=7)



PG_1

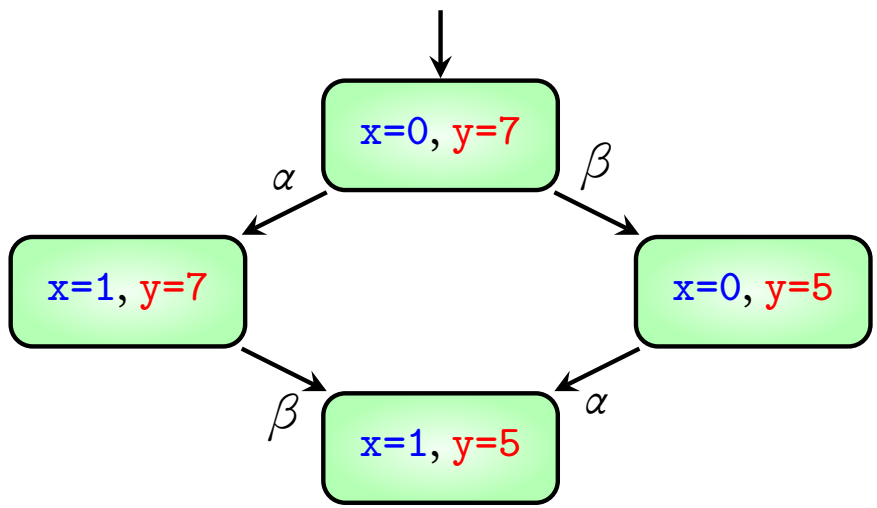


PG_2

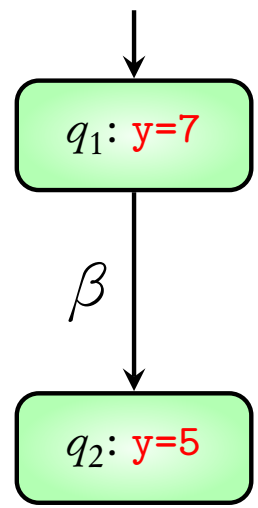


TS_1

(*initially* x=0)

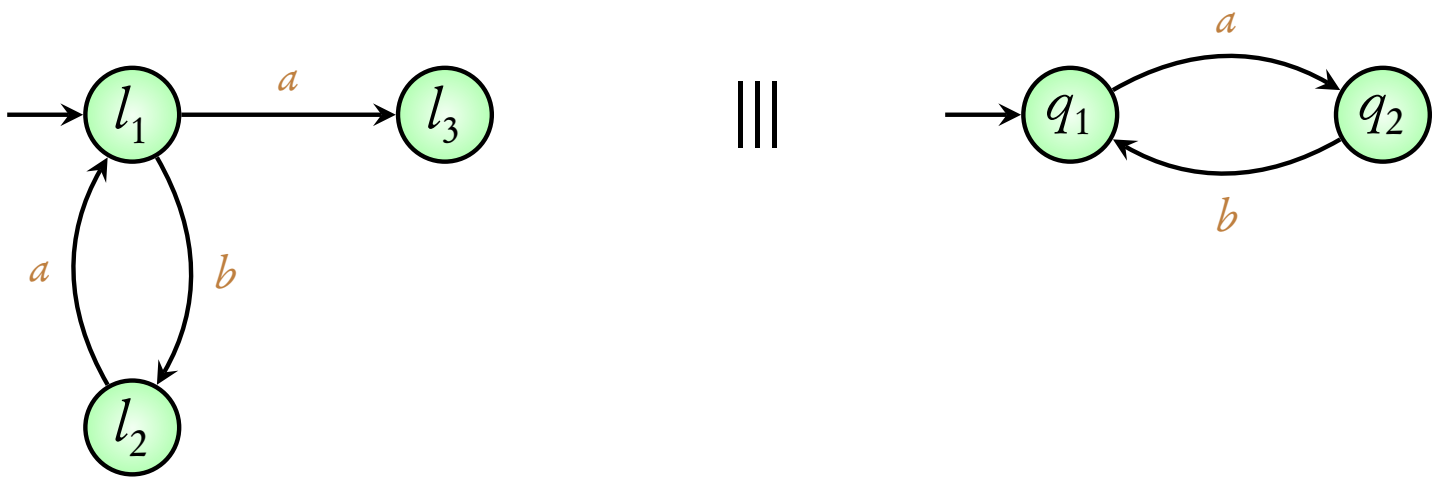


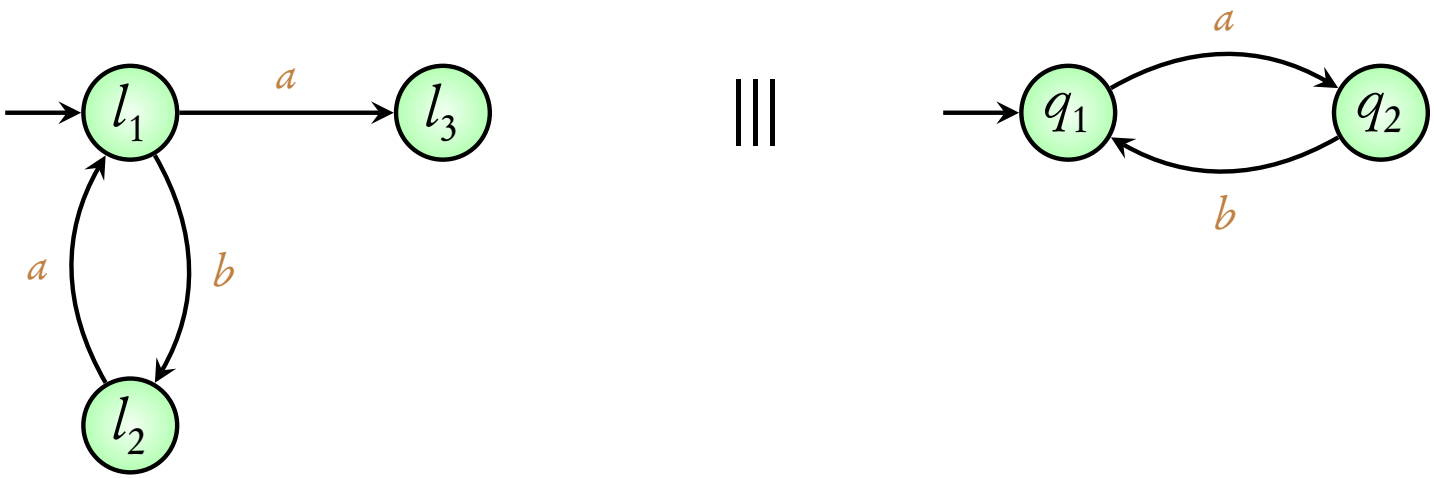
$TS_1 \;|||\; TS_2$

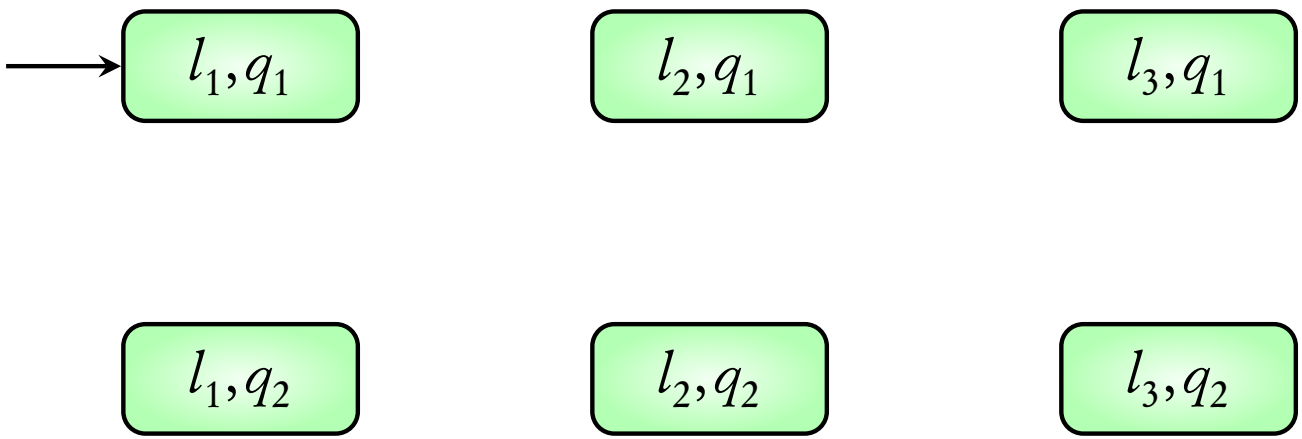


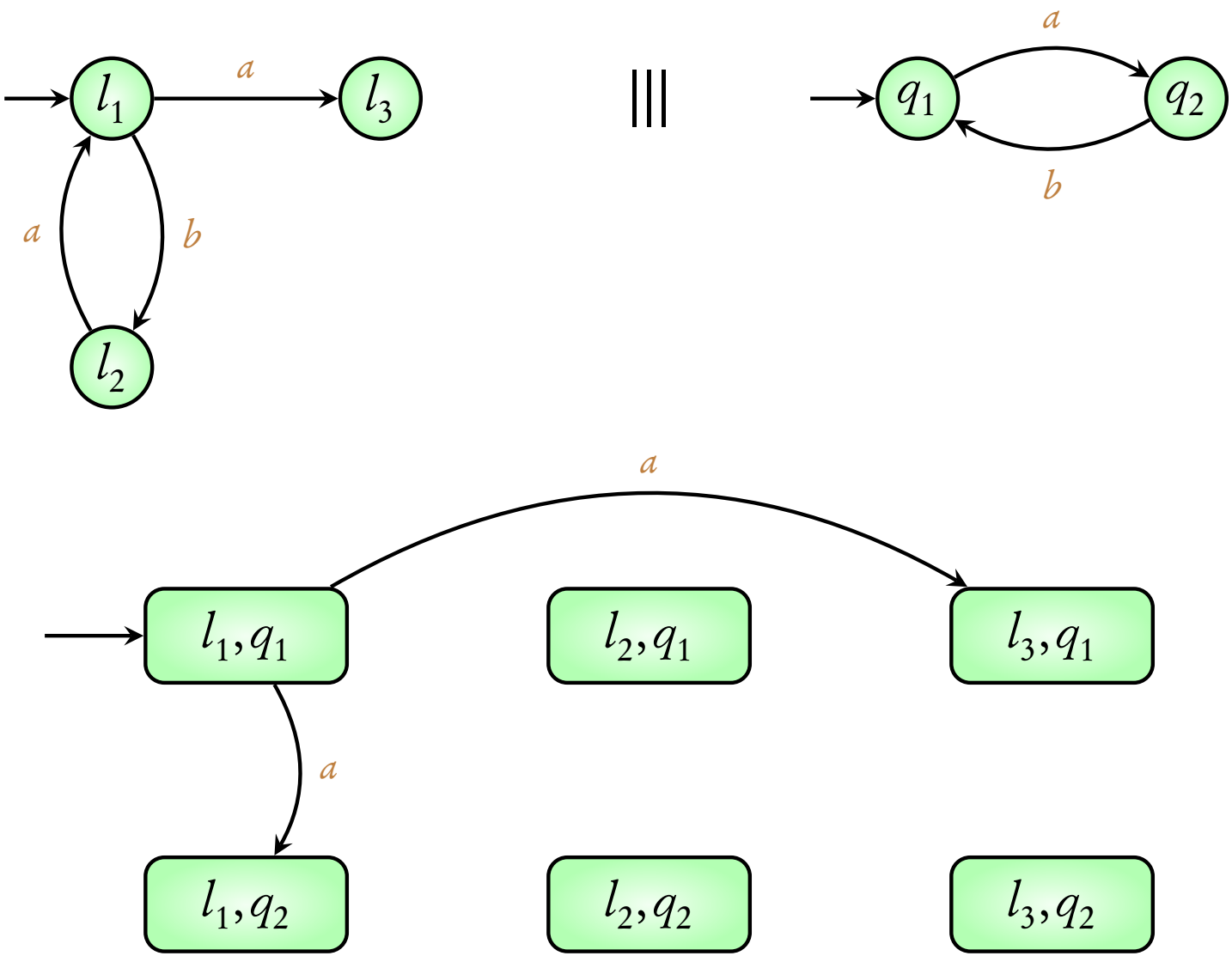
TS_2

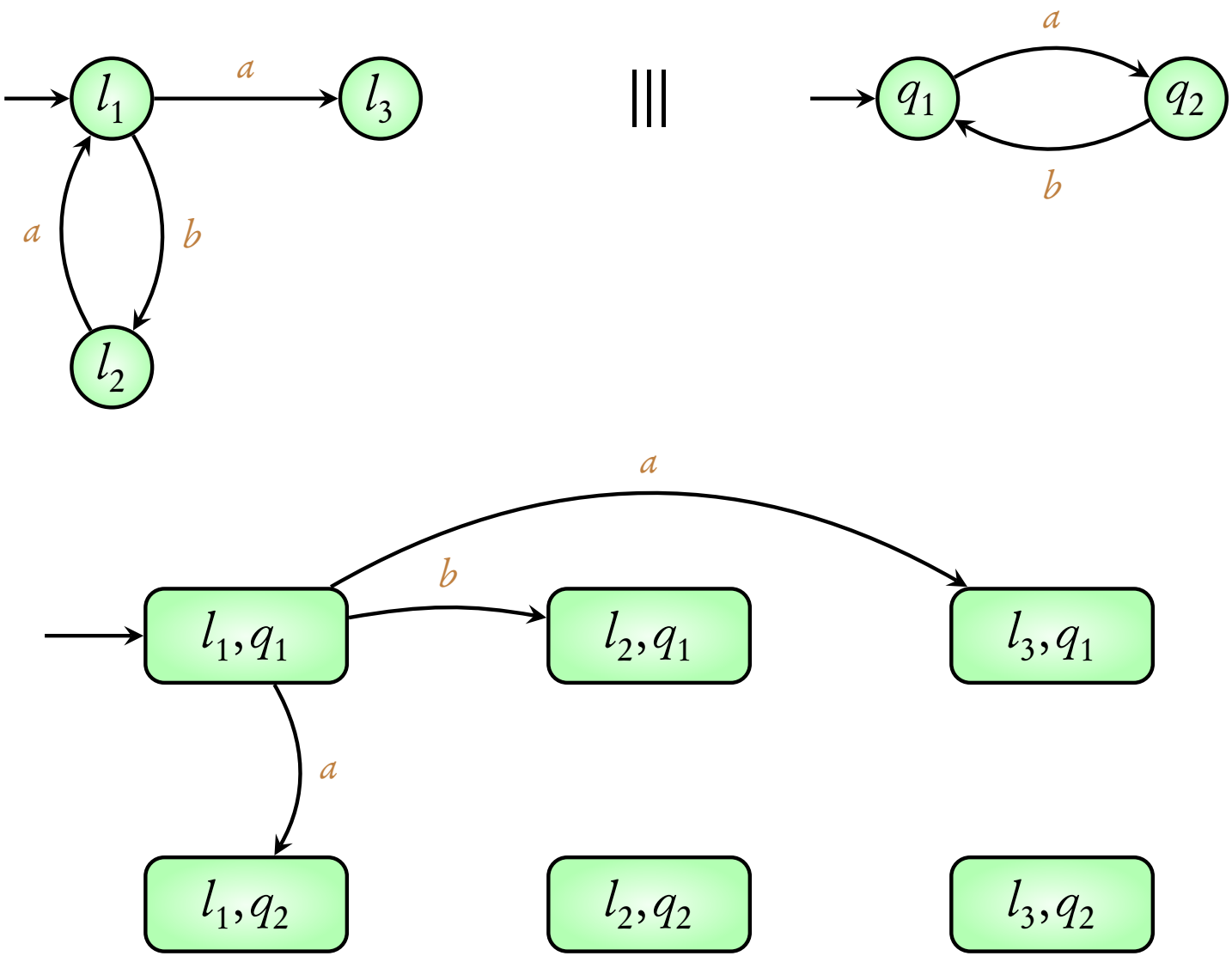
(*initially* y=7)

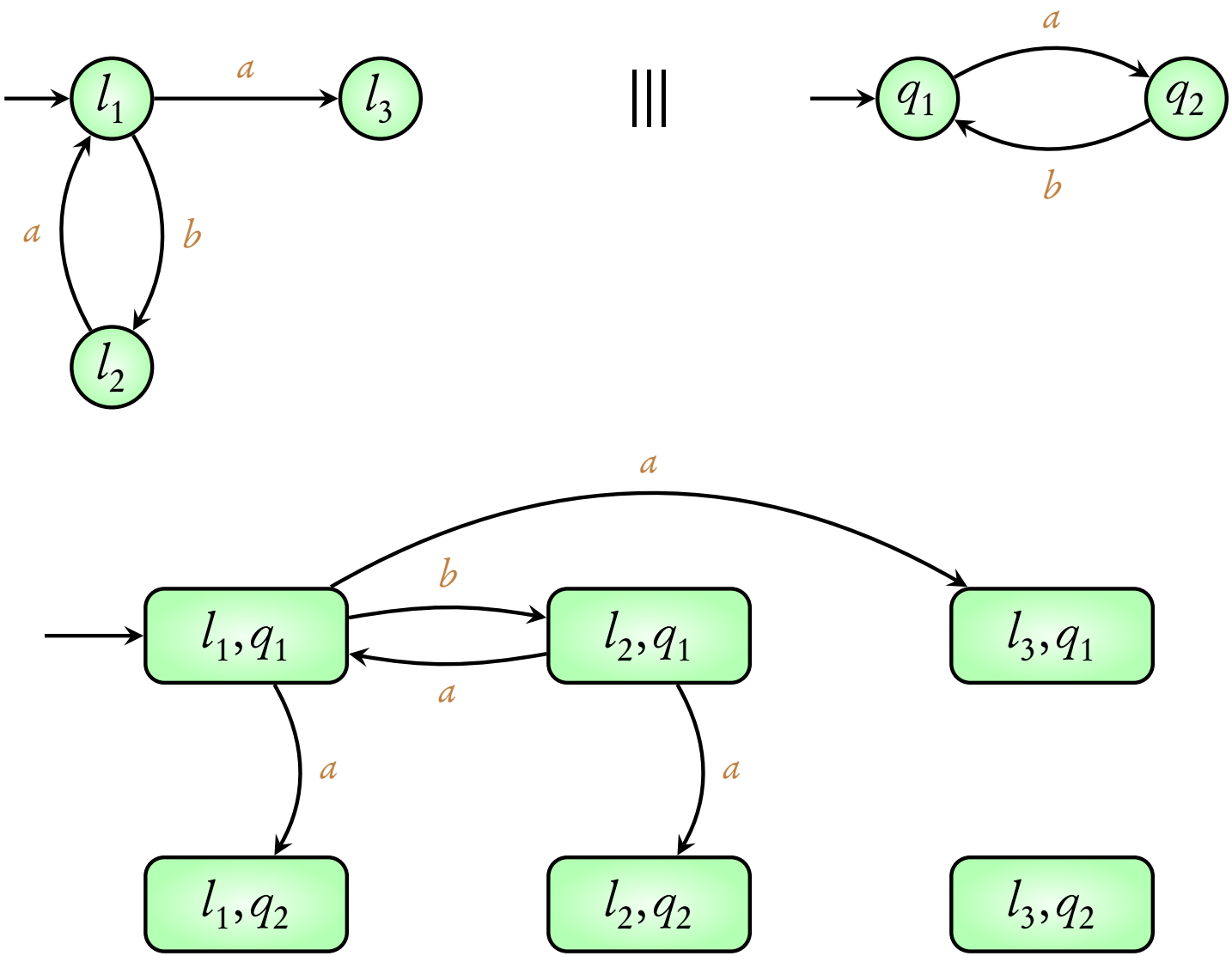

l_1
l_3
l_2
a
a
b
|||
q_1
q_2
a
b

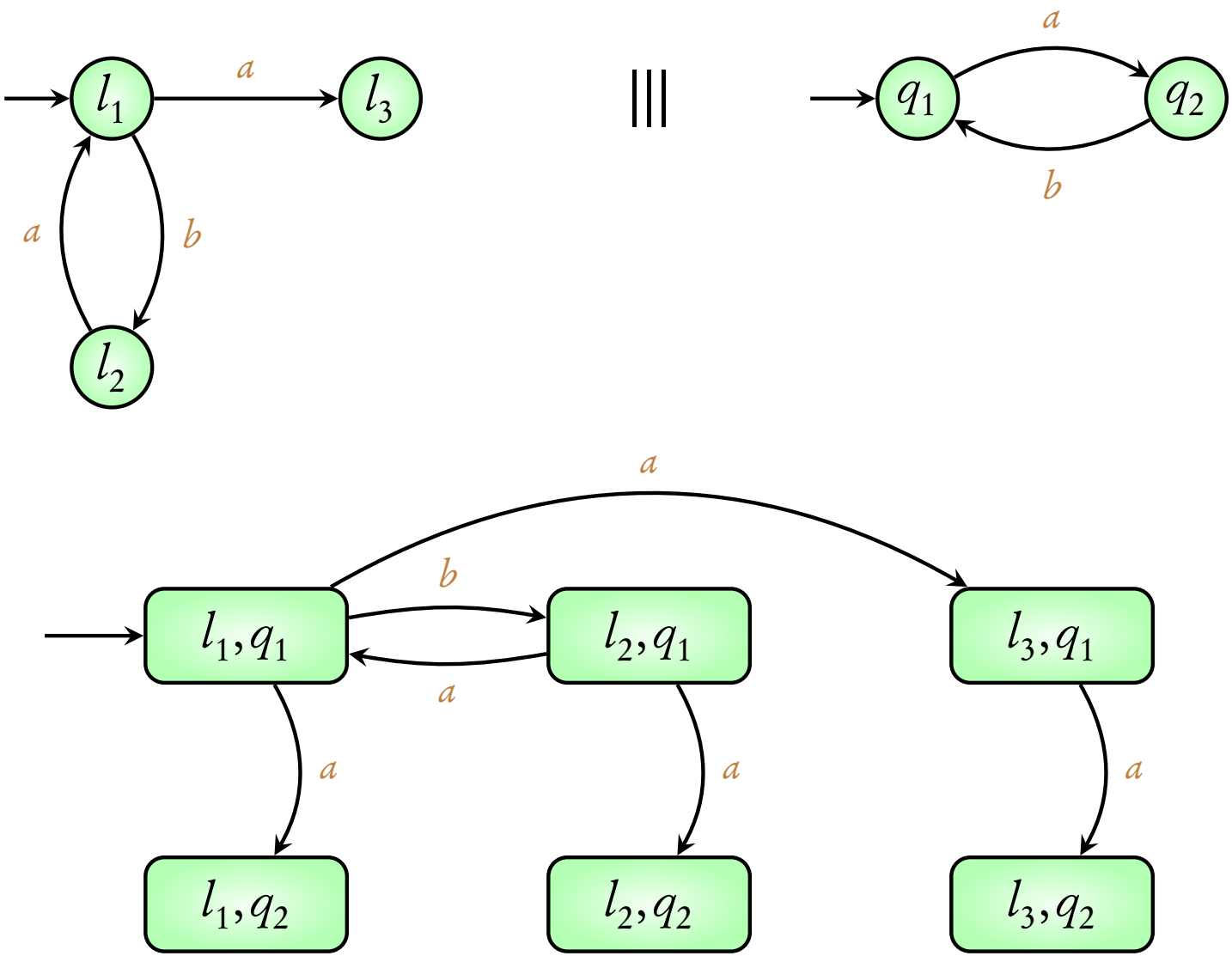

l_1
l_3
l_2
a
a
b
|||
q_1
q_2
a
b

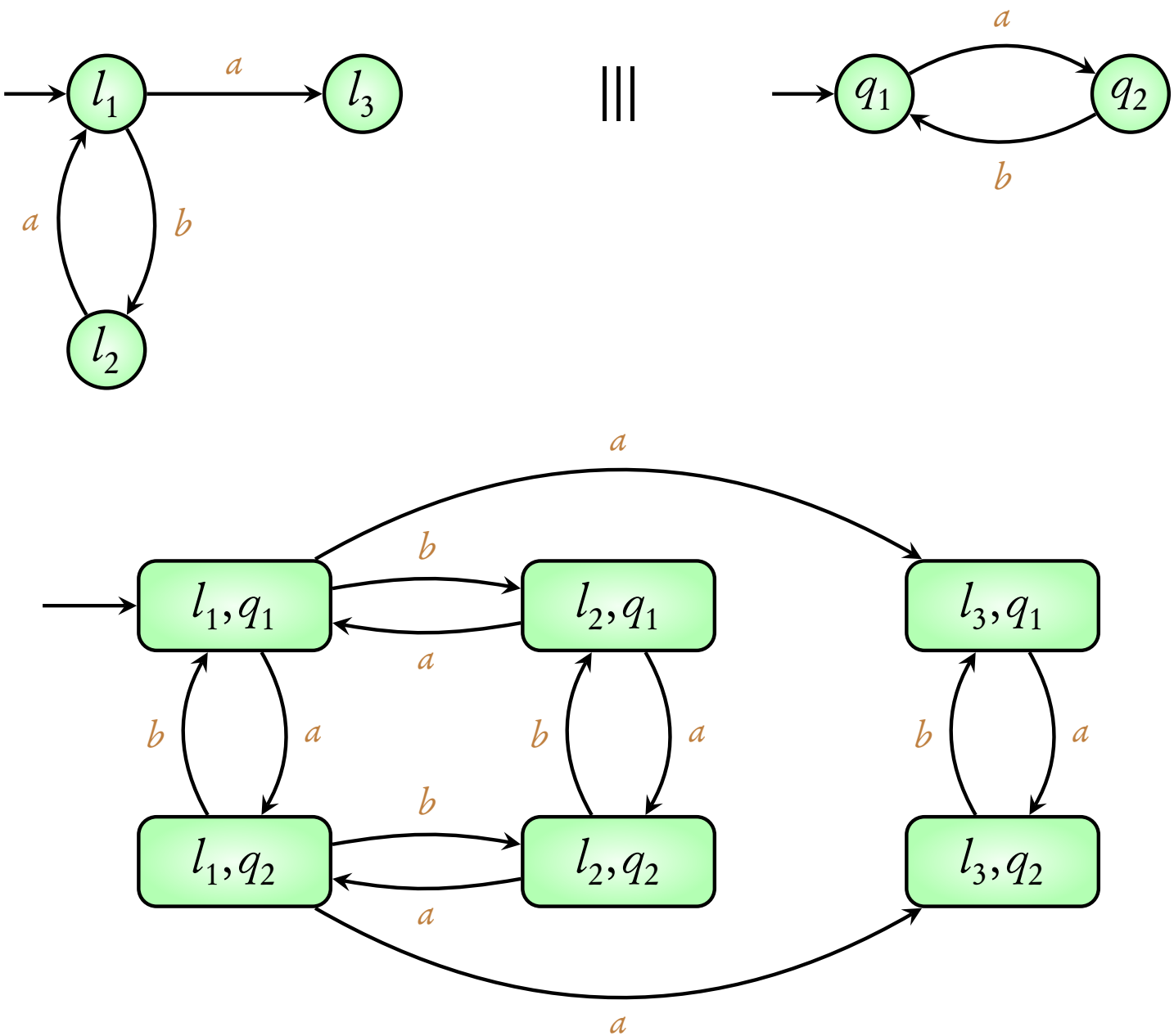

l_1, q_1
l_2, q_1
l_3, q_1
l_1, q_2
l_2, q_2
l_3, q_2


l_1
l_3
l_2
a
a
b
|||
q_1
q_2
a
b
a
l_1, q_1
l_2, q_1
l_3, q_1
a
l_1, q_2
l_2, q_2
l_3, q_2


l_1
l_3
l_2
a
a
b
|||
q_1
q_2
a
b
a
b
l_1, q_1
l_2, q_1
l_3, q_1
a
l_1, q_2
l_2, q_2
l_3, q_2


l_1
l_3
l_2
a
a
b
|||
q_1
q_2
a
b
a
b
a
l_1, q_1
l_2, q_1
l_3, q_1
a
a
l_1, q_2
l_2, q_2
l_3, q_2


l_1
l_3
l_2
a
a
b
|||
q_1
q_2
a
b
a
b
a
l_1, q_1
l_2, q_1
l_3, q_1
l_1, q_2
l_2, q_2
l_3, q_2
a
a
a


l_1
l_3
l_2
a
a
b
|||
q_1
q_2
a
b
l_1, q_1
l_2, q_1
l_3, q_1
l_1, q_2
l_2, q_2
l_3, q_2
a
b
a
b
a
b
a
b
a
b
a

Multiple systems

$$TS_1 \;|||\; TS_2 \;|||\; \ldots \;|||\; TS_n$$

Multiple systems

$$TS_1 \;|||\; TS_2 \;|||\; \ldots \;|||\; TS_n$$

Exercise: Try out an example of interleaving **three** systems

Concurrent systems

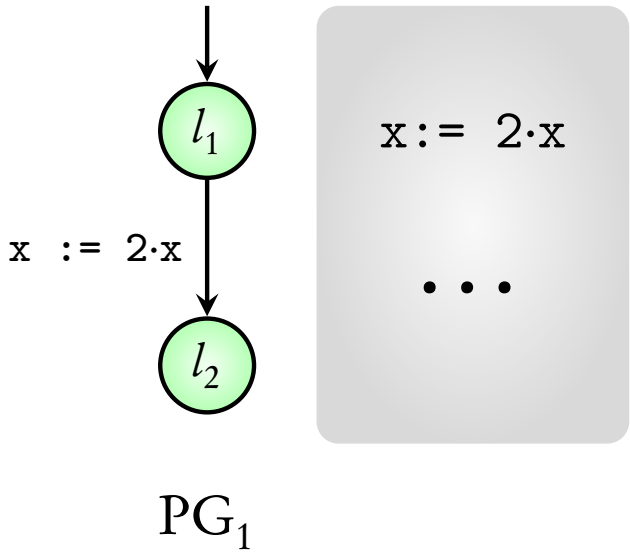
Independent

Interleaving

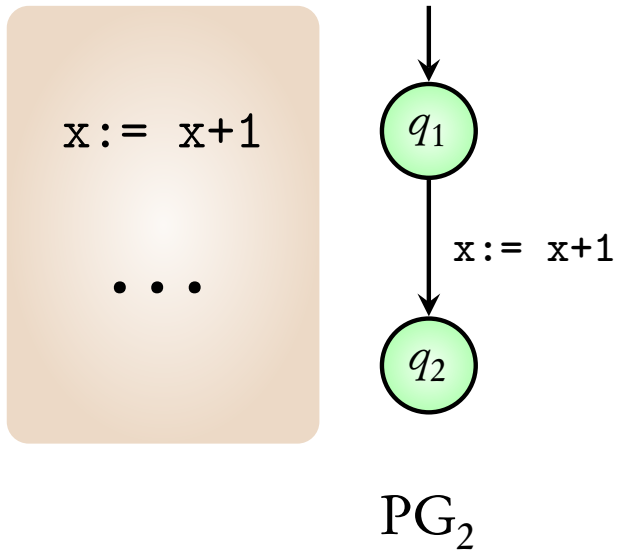
$TS_1 \;|||\; TS_2 \;|||\; \ldots \;|||\; TS_n$

Shared variables

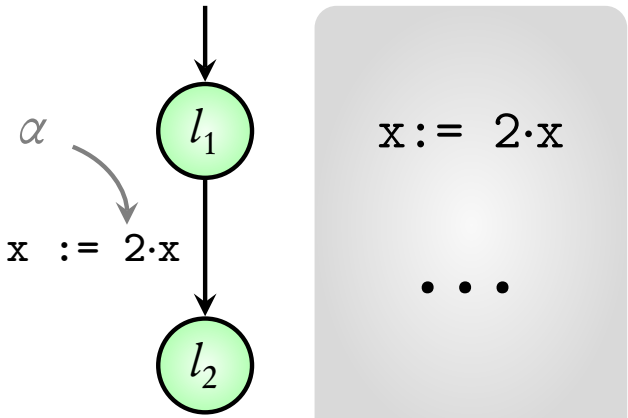
Shared actions



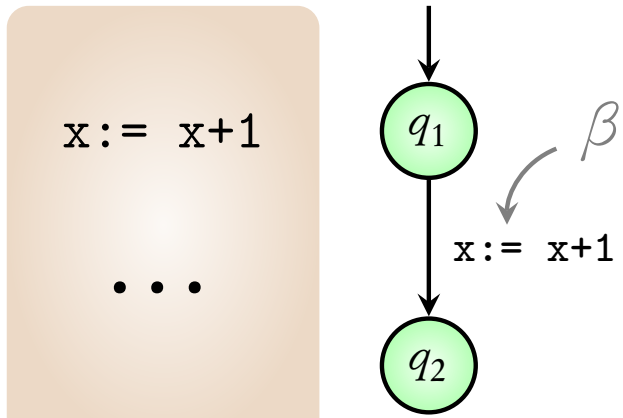
PG_1



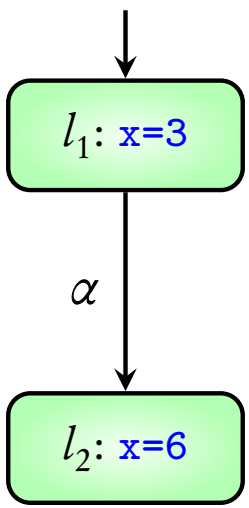
PG_2



PG_1

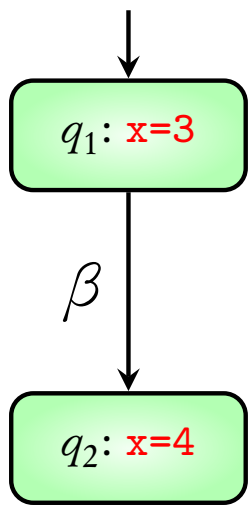


PG_2



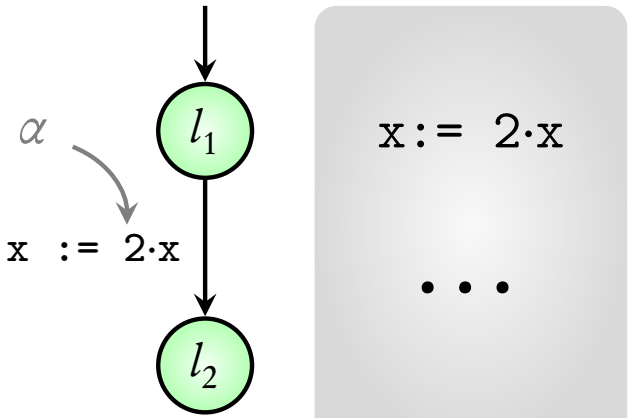
TS_1

(*initially* x=3)

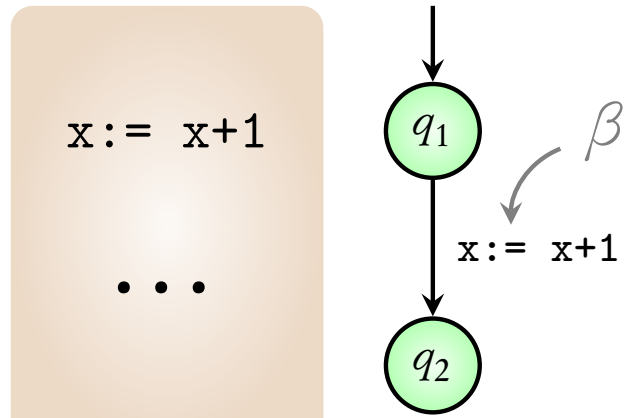


TS_2

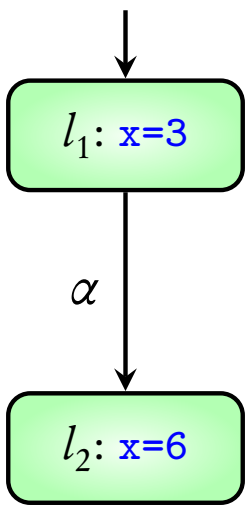
(*initially* x=3)



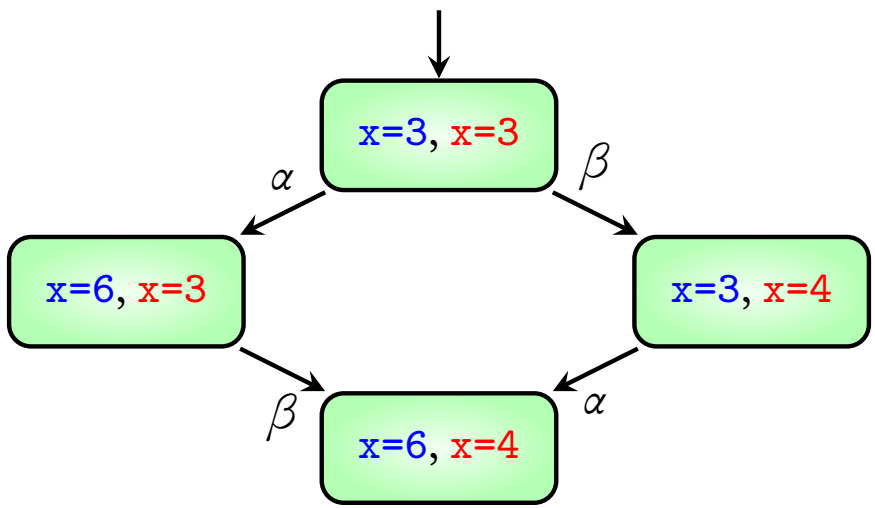
PG_1



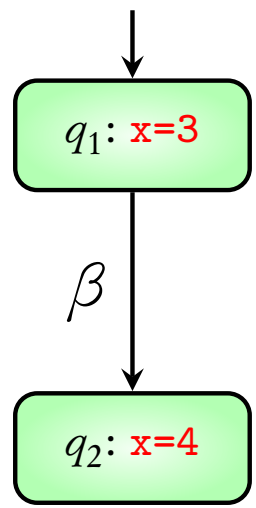
PG_2



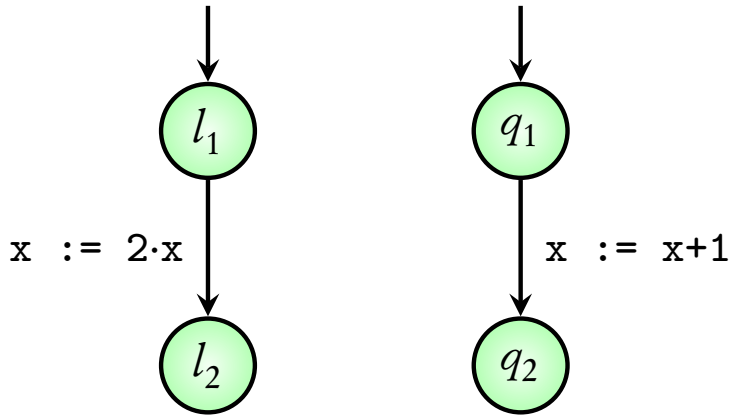
TS_1
(*initially* x=3)



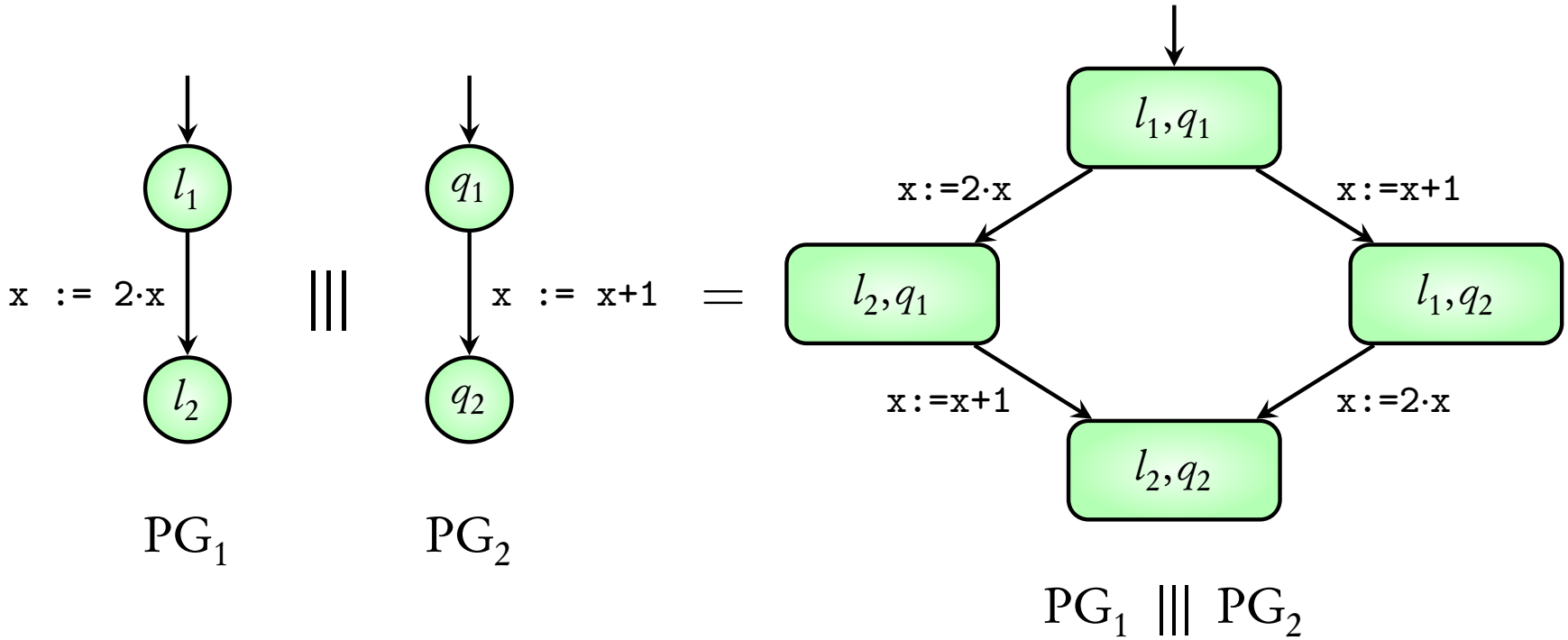
$TS_1 \;|||\; TS_2$



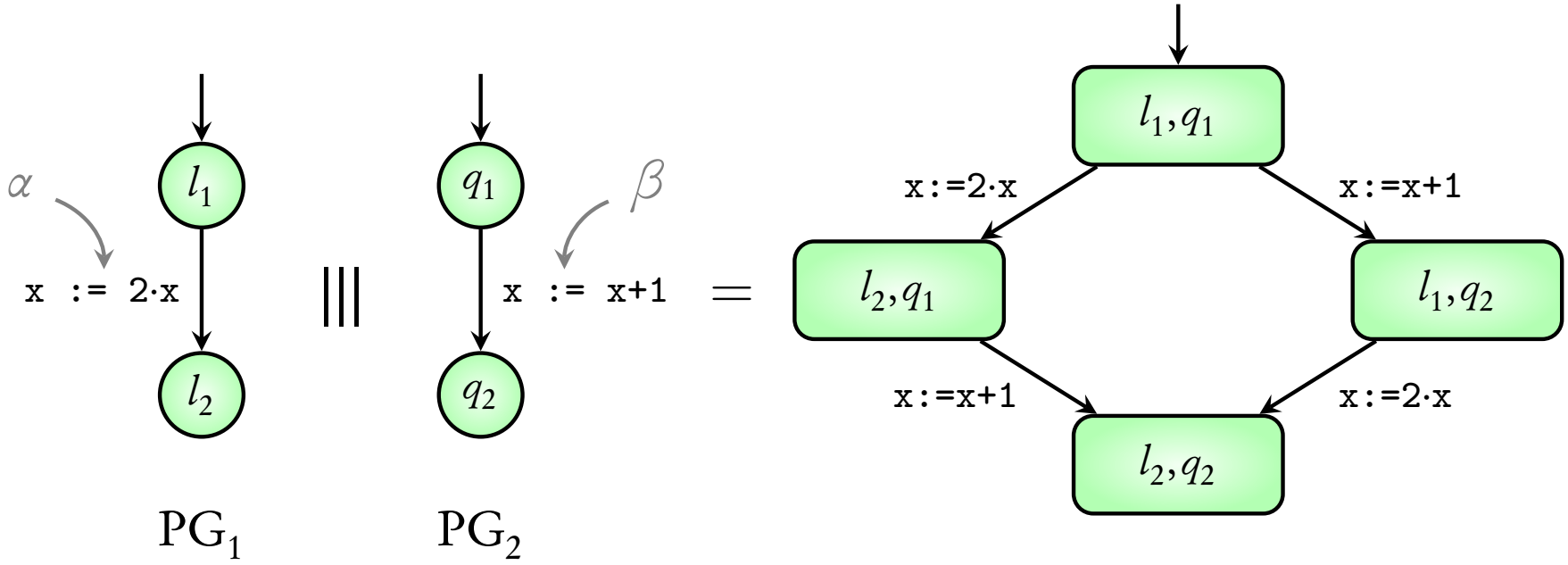
TS_2
(*initially* x=3)



PG_1 PG_2



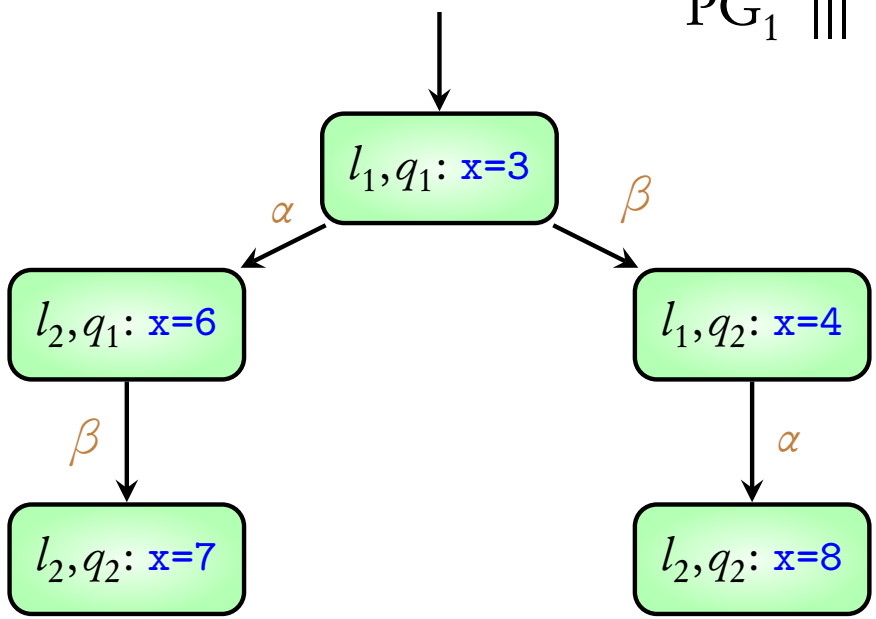
PG_1 PG_2 $\mathrm{PG}_1 \;|||\; \mathrm{PG}_2$



PG_1 PG_2

$PG_1 \;|||\; PG_2$

TS($PG_1 \;|||\; PG_2$):



Concurrent systems

Independent

Interleaving

$TS_1 \;|||\; TS_2 \;|||\; \ldots \;|||\; TS_n$

Shared variables

$TS(\,PG_1 \;|||\; PG_2 \;|||\; \ldots \;|||\; PG_n\,)$

Shared actions

Coming next: Another example

```
while x < 200
    x := x+1
```

```
while x>0
    x := x-1
```

```
while x=200
    x := 0
```

```
while x < 200
    x := x+1
```

```
while x>0
    x := x-1
```

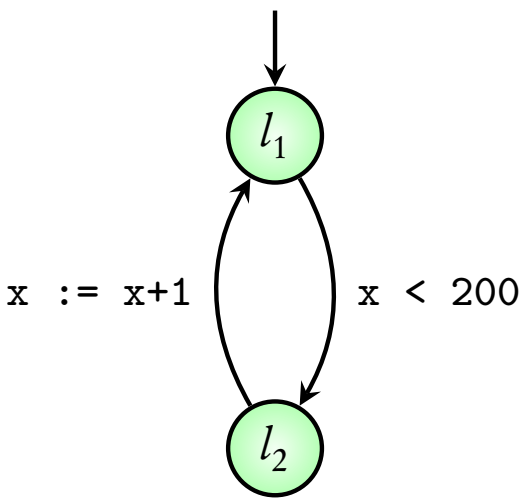
```
while x=200
    x := 0
```

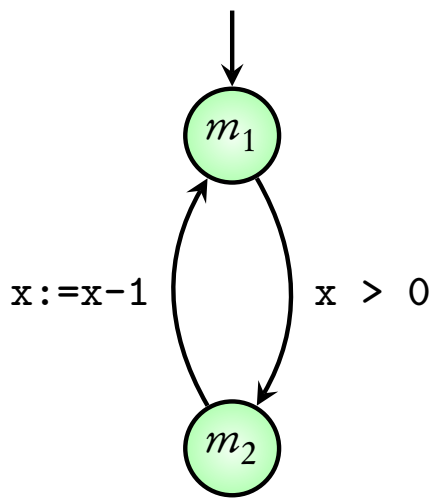
Is the value of x **always between** 0 and 200?

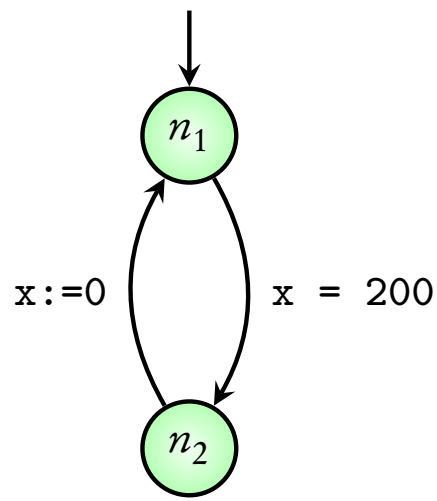
```
while x < 200
    x := x+1
```

```
while x>0
    x := x-1
```

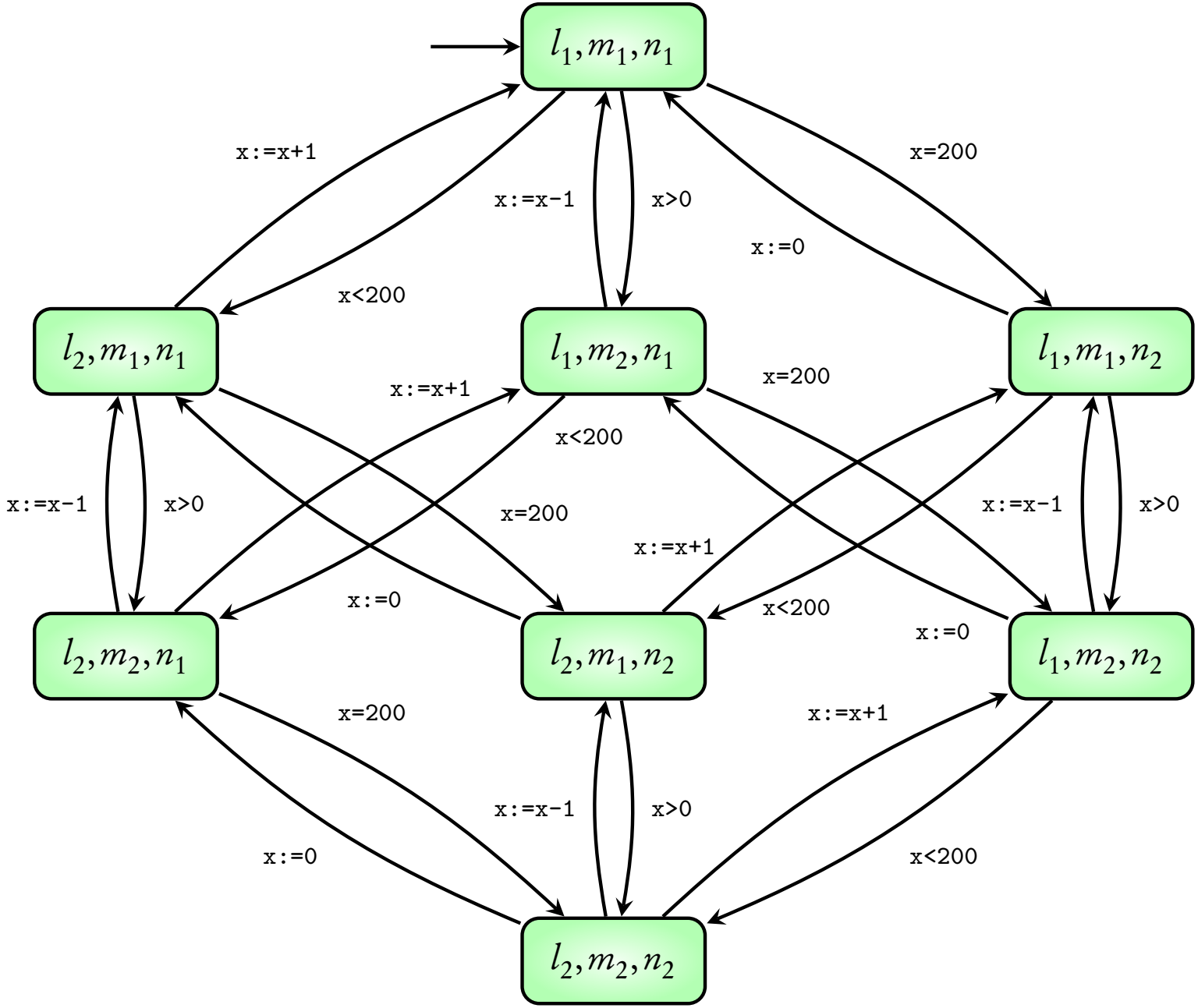
```
while x=200
    x := 0
```

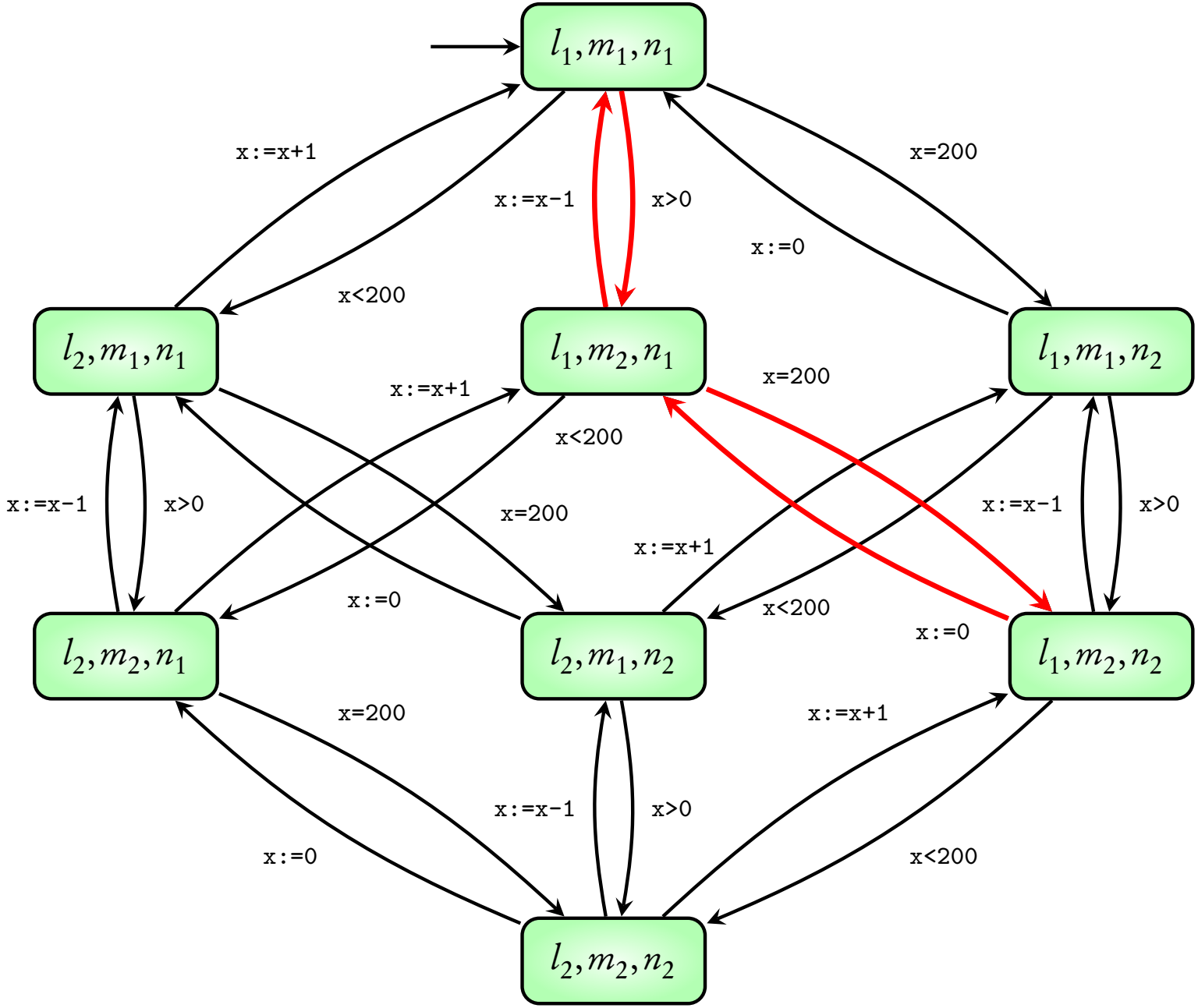


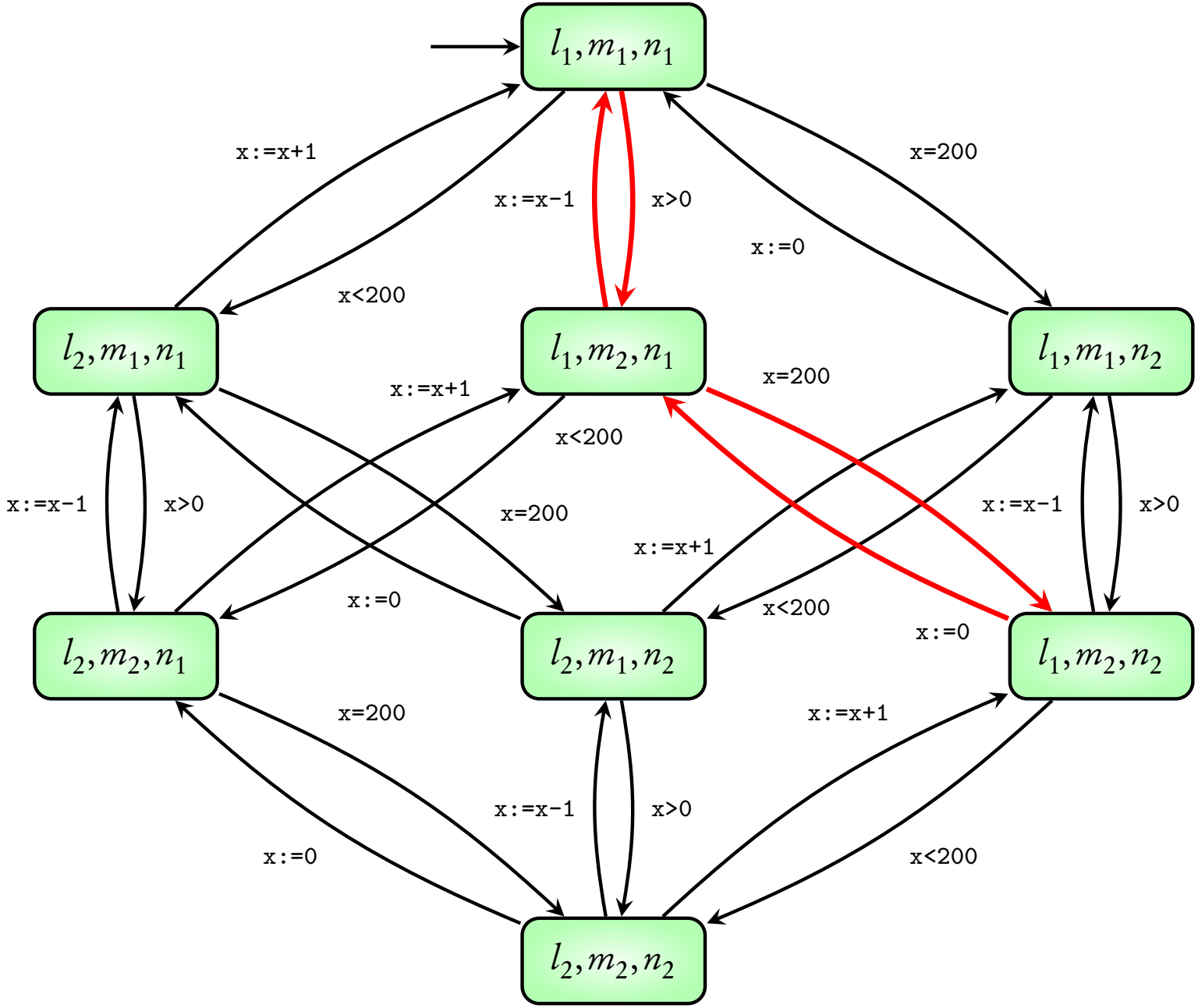




Is the value of x **always between** 0 and 200?

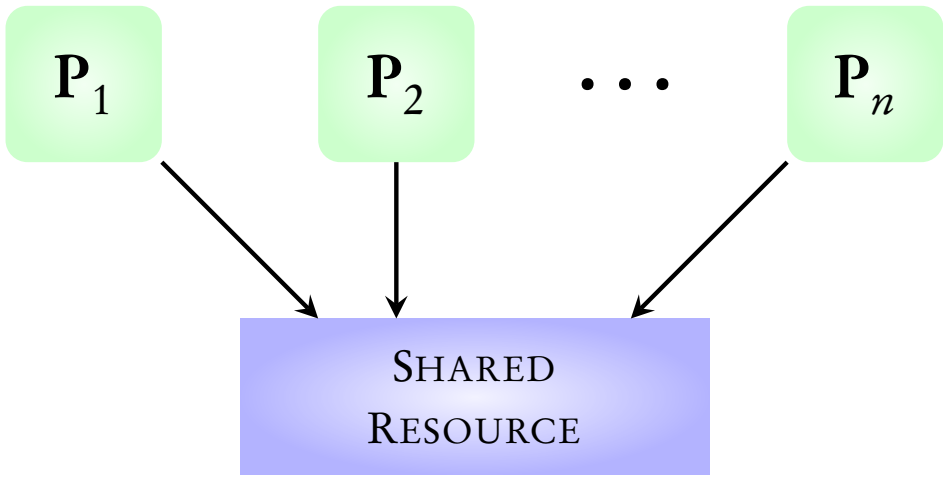

l_1, m_1, n_1
l_2, m_1, n_1
l_1, m_2, n_1
l_1, m_1, n_2
l_2, m_2, n_1
l_2, m_1, n_2
l_1, m_2, n_2
l_2, m_2, n_2
x:=x+1
x:=x-1
x>0
x=200
x:=0
x<200
x:=x+1
x=200
x<200
x:=x-1
x>0
x=200
x:=x+1
x:=0
x<200
x:=0
x=200
x:=x+1
x:=x-1
x>0
x:=0
x<200


l_1, m_1, n_1
l_2, m_1, n_1
l_1, m_2, n_1
l_1, m_1, n_2
l_2, m_2, n_1
l_2, m_1, n_2
l_1, m_2, n_2
l_2, m_2, n_2
x:=x+1
x:=x-1
x>0
x=200
x:=0
x<200
x:=x+1
x=200
x<200
x:=x-1
x>0
x=200
x:=x+1
x:=0
x<200
x:=0
x=200
x:=x+1
x:=x-1
x>0
x:=0
x<200



Is the value of x **always between** 0 and 200? **No**

Coming next: Mutual exclusion



(variable, printer, ...)

Mutual Exclusion: **No two processes** can access the resource **simultaneously**

Goal: Modeling the **protocols** used for mutual exclusion

P_1

loop forever

⋮ *non-critical actions*

request

critical section

release

⋮ *non-critical actions*

end loop

P_2

loop forever

⋮ *non-critical actions*

request

critical section

release

⋮ *non-critical actions*

end loop

P_1

loop forever

⋮ *non-critical actions*

request

critical section

release

⋮ *non-critical actions*

end loop

P_2

loop forever

⋮ *non-critical actions*

request

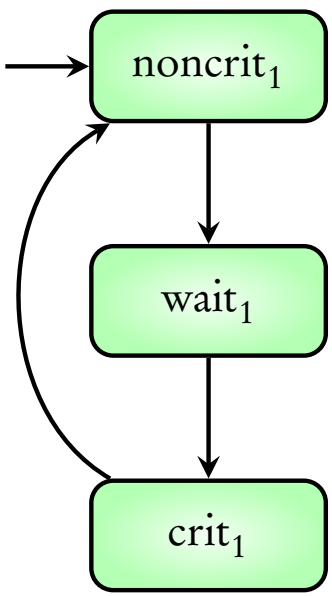
critical section

release

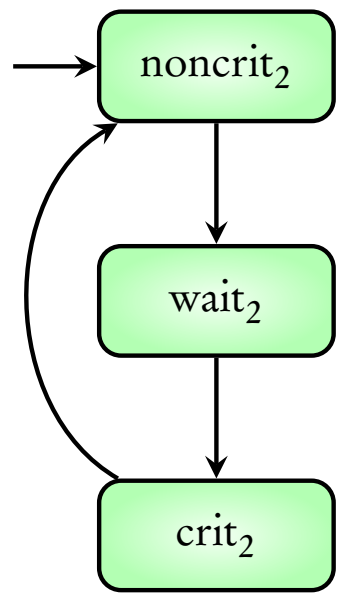
⋮ *non-critical actions*

end loop

PG_1



PG_2



P$_1$

loop forever

⋮ *non-critical actions*

⟨ if y>0: y:=y−1 ⟩ *request*

critical section

y:=y+1 *release*

⋮ *non-critical actions*

end loop

P$_2$

loop forever

⋮ *non-critical actions*

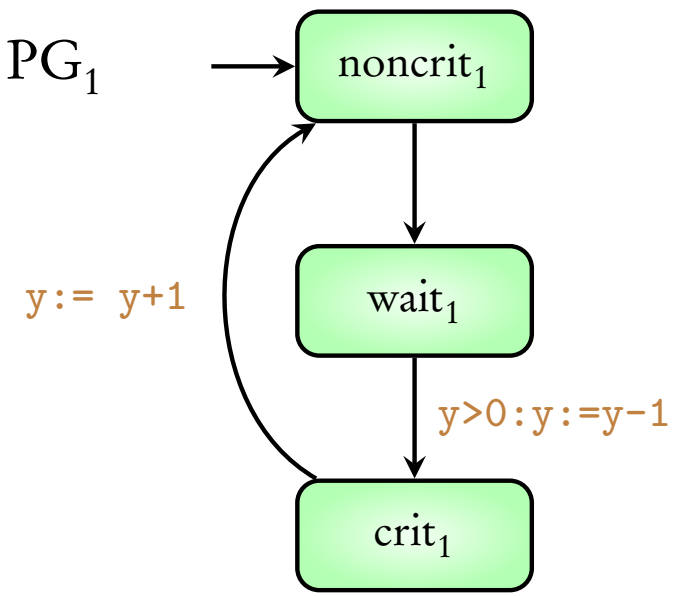
⟨ if y>0: y:=y−1 ⟩ *request*

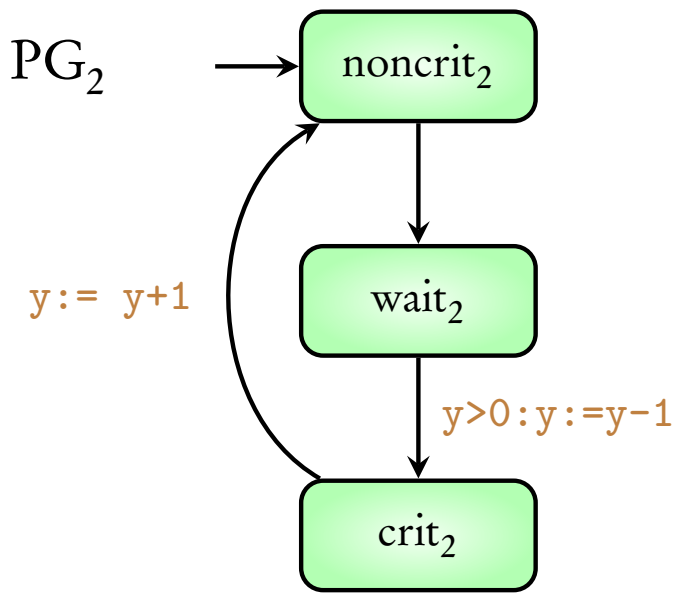
critical section

y:=y+1 *release*

⋮ *non-critical actions*

end loop





P$_1$

loop forever

⋮ *non-critical actions*

⟨ if y>0: y:=y−1 ⟩ *request*

critical section

y:=y+1 *release*

⋮ *non-critical actions*

end loop

P$_2$

loop forever

⋮ *non-critical actions*

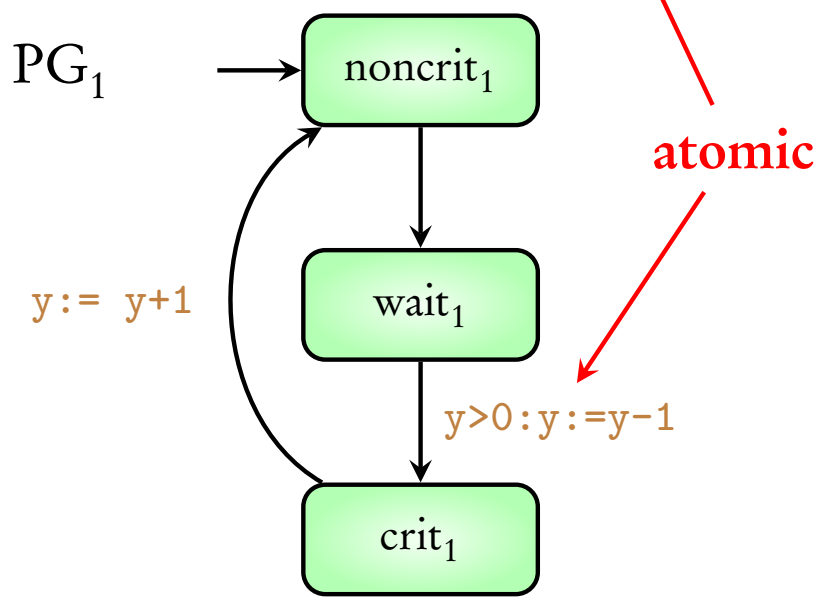
⟨ if y>0: y:=y−1 ⟩ *request*

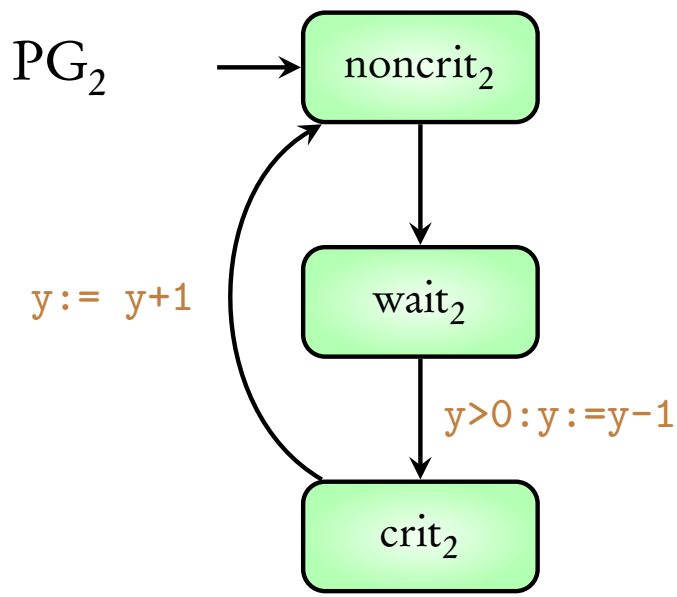
critical section

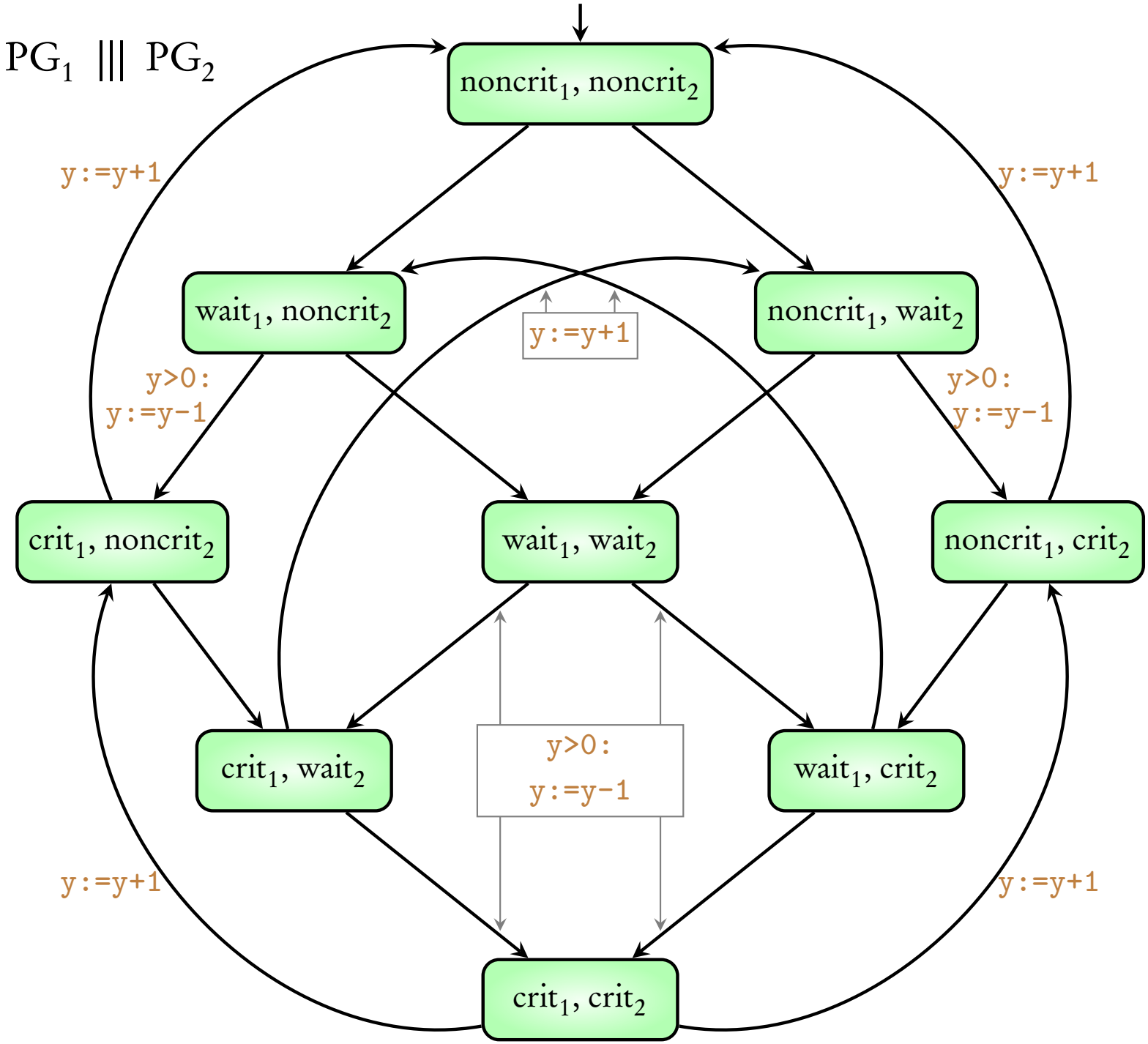
y:=y+1 *release*

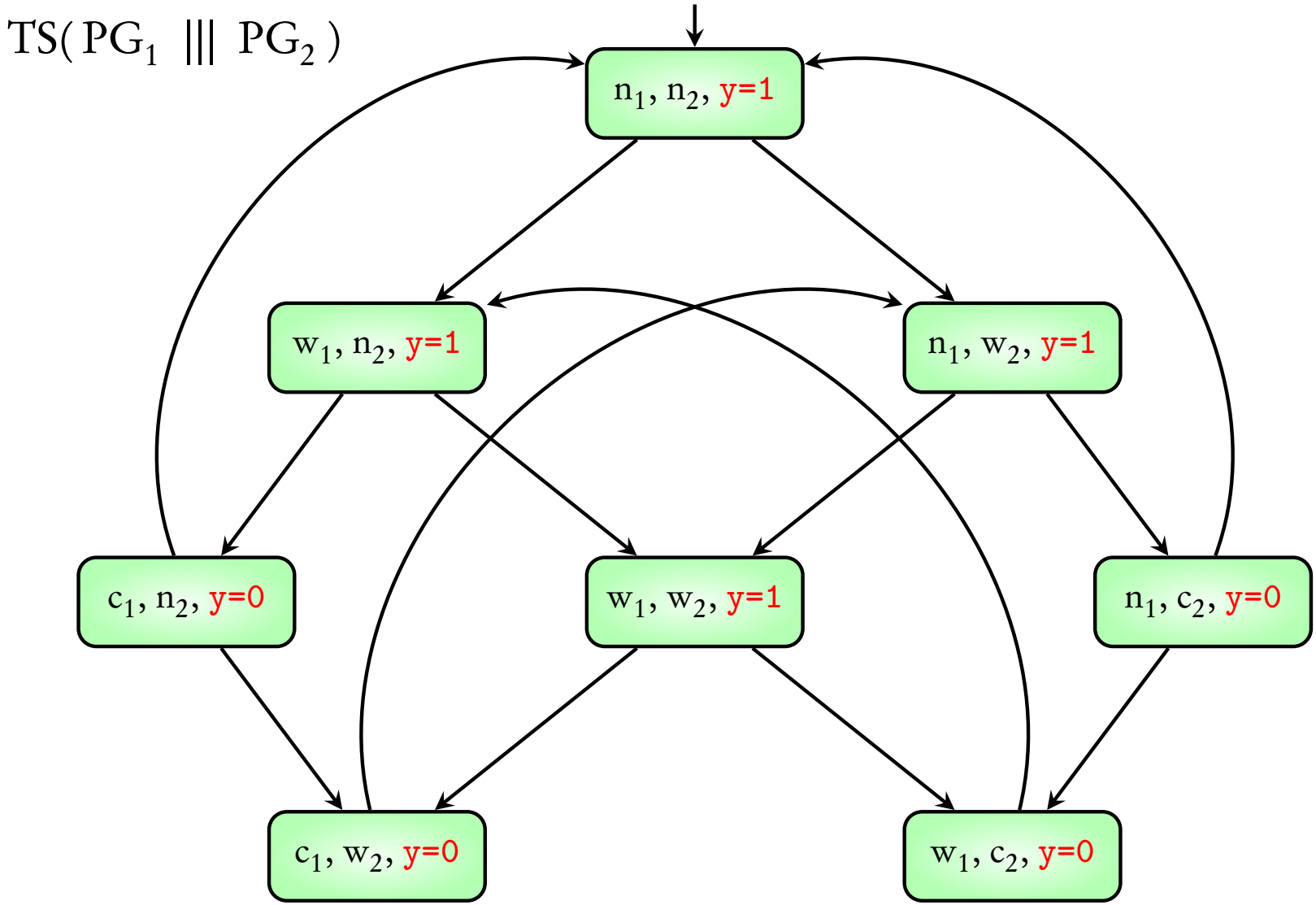
⋮ *non-critical actions*

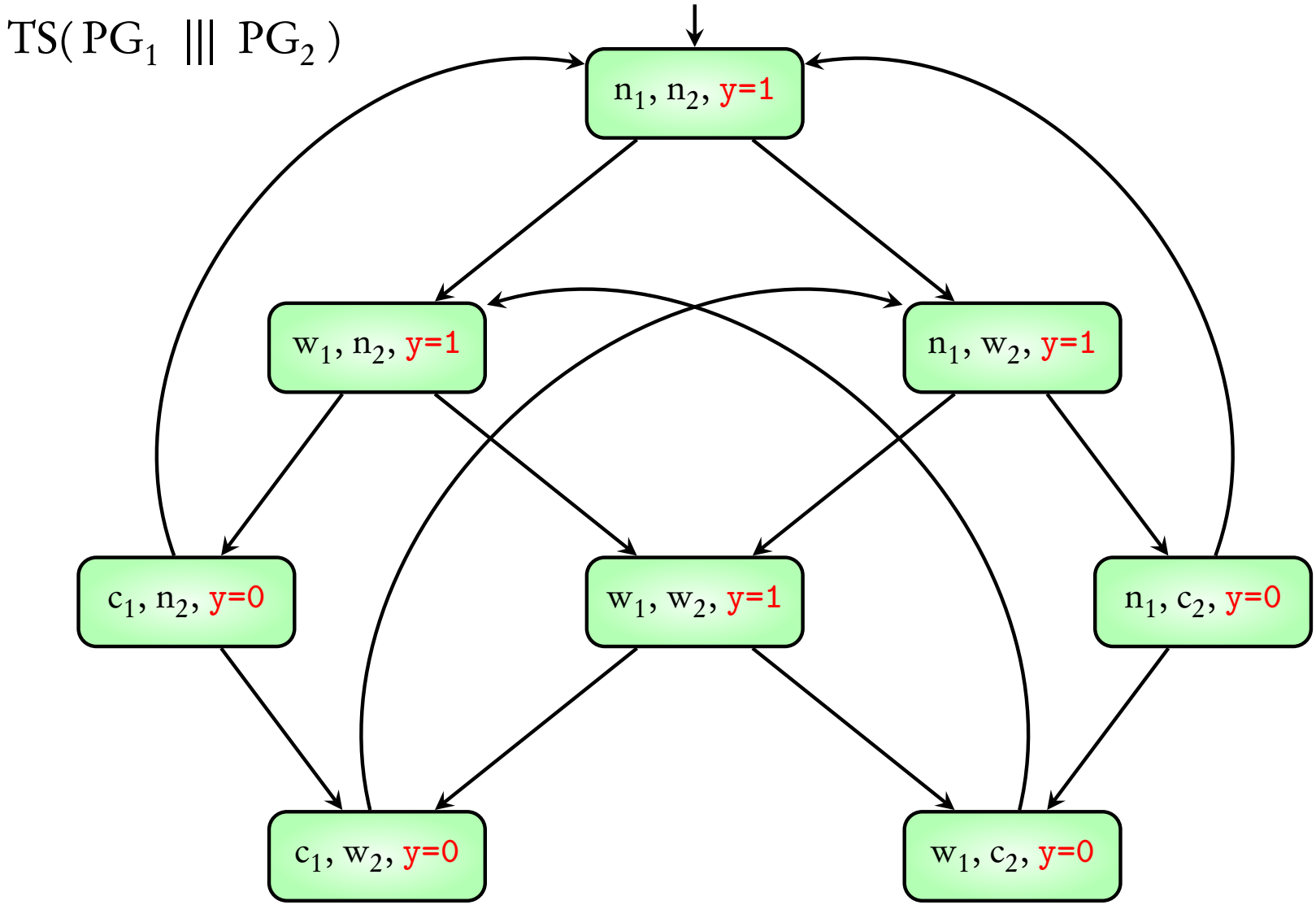
end loop






PG1 ||| PG2
noncrit1, noncrit2
y:=y+1
y:=y+1
wait1, noncrit2
noncrit1, wait2
y:=y+1
y>0:
y:=y-1
y>0:
y:=y-1
crit1, noncrit2
wait1, wait2
noncrit1, crit2
crit1, wait2
y>0:
y:=y-1
wait1, crit2
y:=y+1
y:=y+1
crit1, crit2


TS(PG1 ||| PG2)
n1, n2, y=1
w1, n2, y=1
n1, w2, y=1
c1, n2, y=0
w1, w2, y=1
n1, c2, y=0
c1, w2, y=0
w1, c2, y=0



Both processes **cannot be** in critical section **simultaneously**

Concurrent systems

Independent

Interleaving

$TS_1 \;|||\; TS_2 \;|||\; \ldots \;|||\; TS_n$

Shared variables

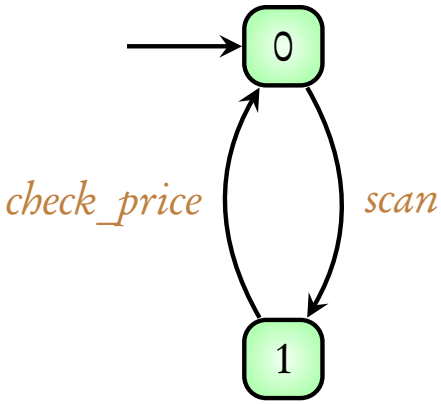
$TS(\,PG_1 \;|||\; PG_2 \;|||\; \ldots \;|||\; PG_n\,)$

Mutual Exclusion

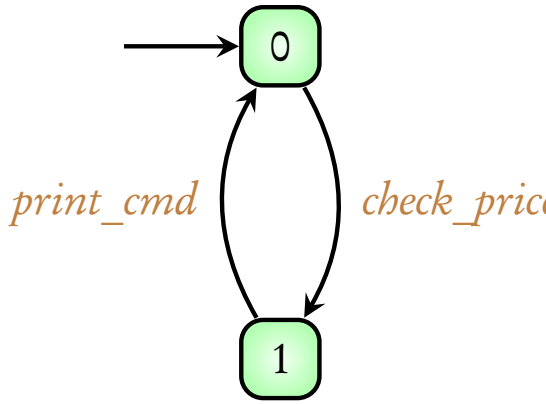
Shared actions

Coming next: **Book-keeping** system in a supermarket

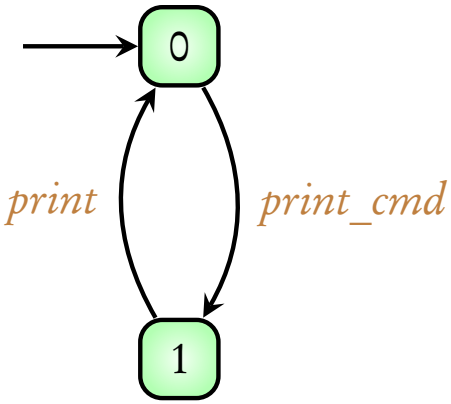
Bar-Code Reader (BCR)



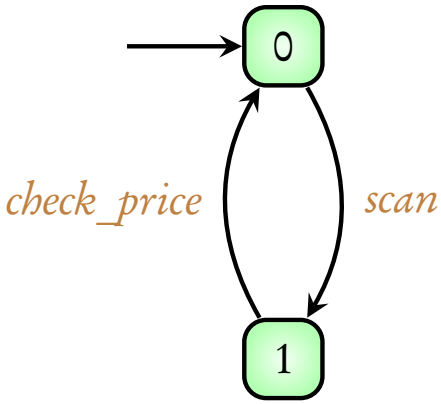
Booking Program (BP)



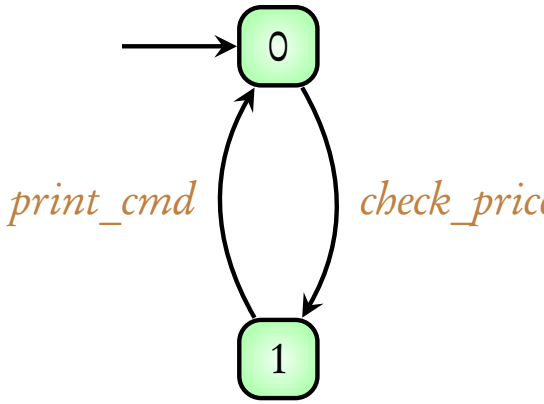
Printer (P)



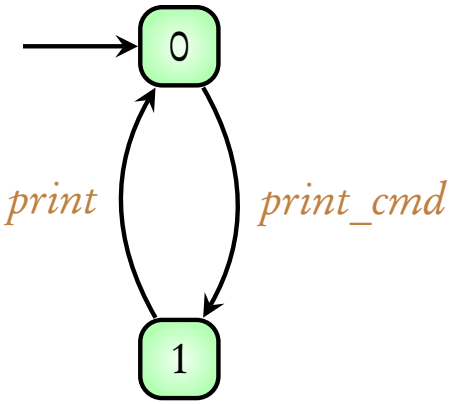
Bar-Code Reader (BCR)

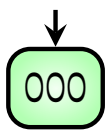


Booking Program (BP)

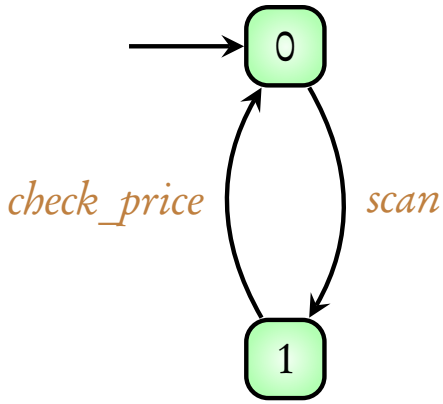


Printer (P)

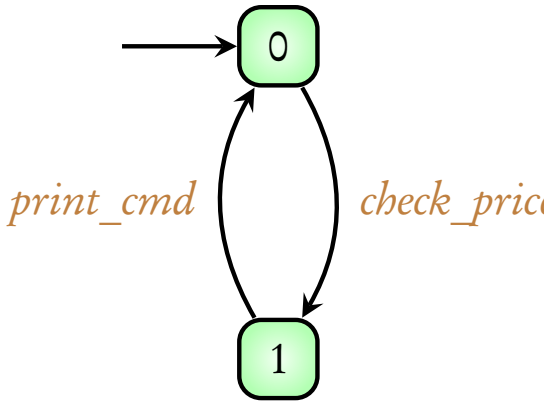




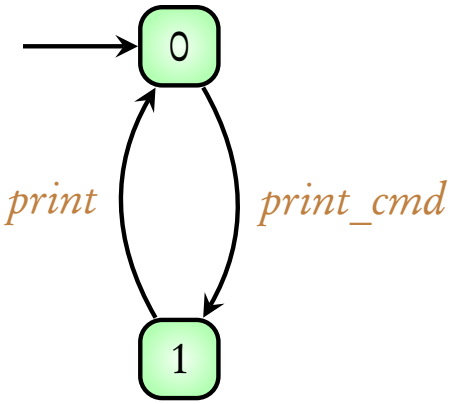
Bar-Code Reader (BCR)



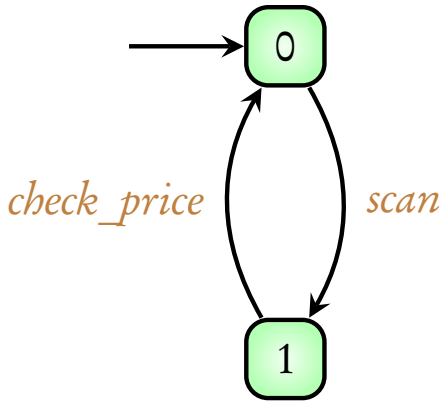
Booking Program (BP)



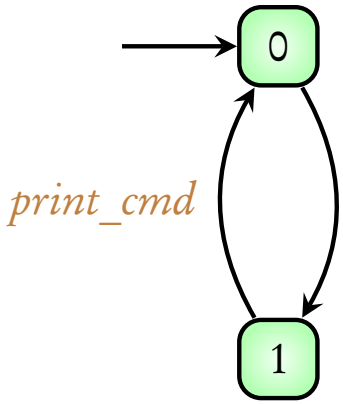
Printer (P)



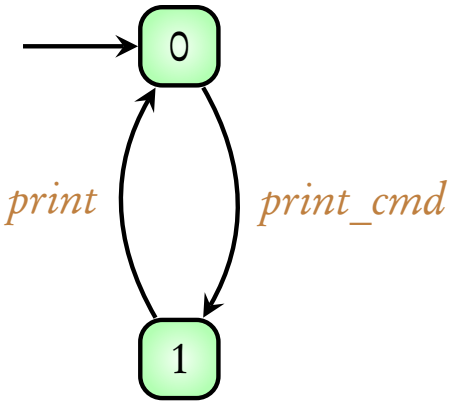
Bar-Code Reader (BCR)

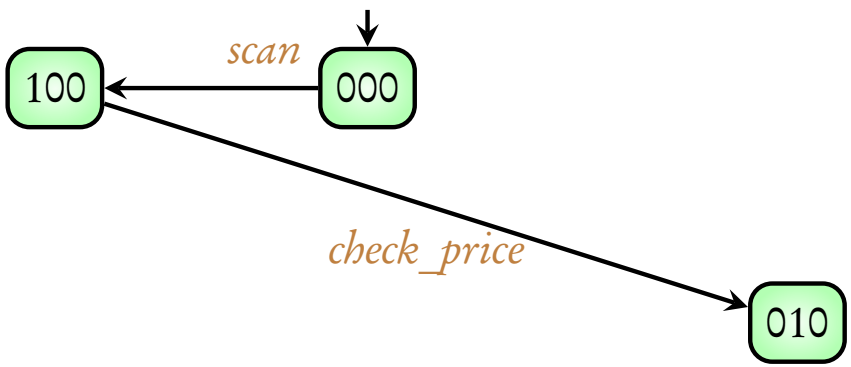


Booking Program (BP)

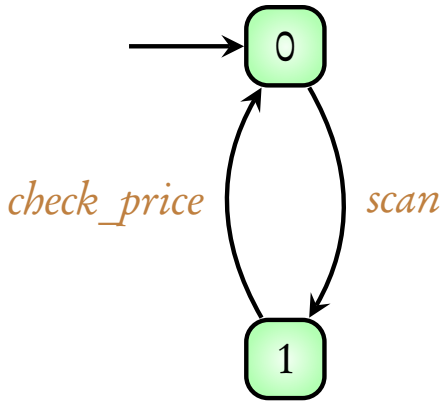


Printer (P)

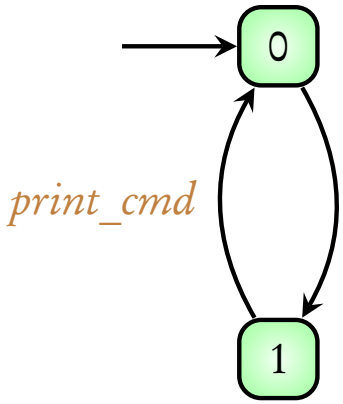




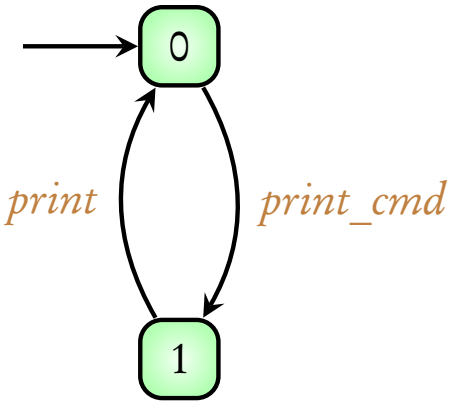
Bar-Code Reader (BCR)

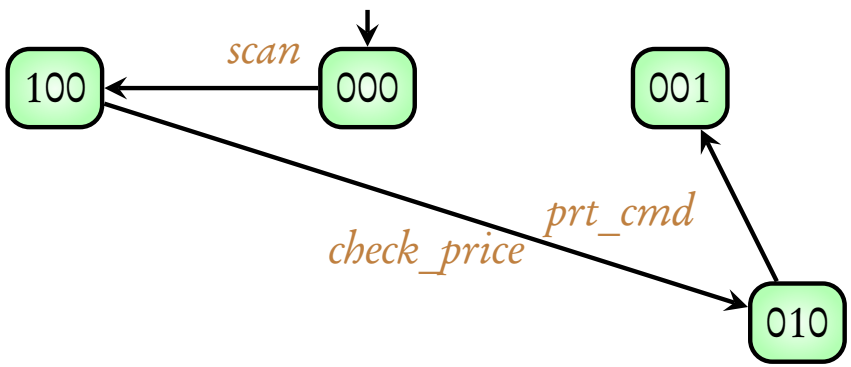


Booking Program (BP)

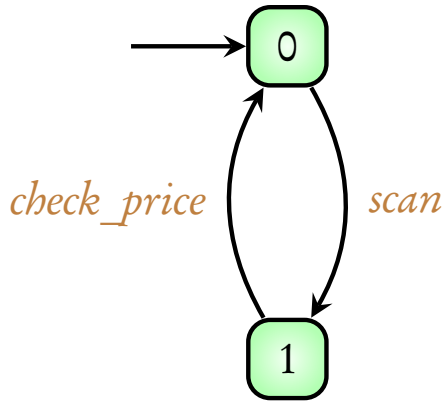


Printer (P)

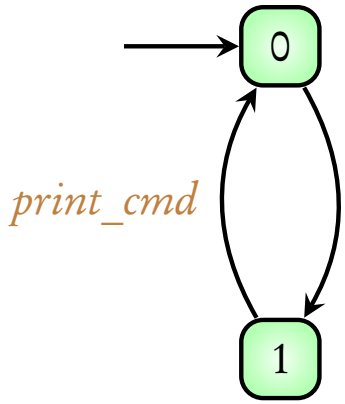




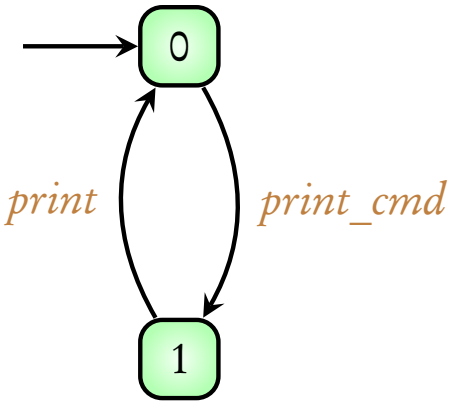
Bar-Code Reader (BCR)

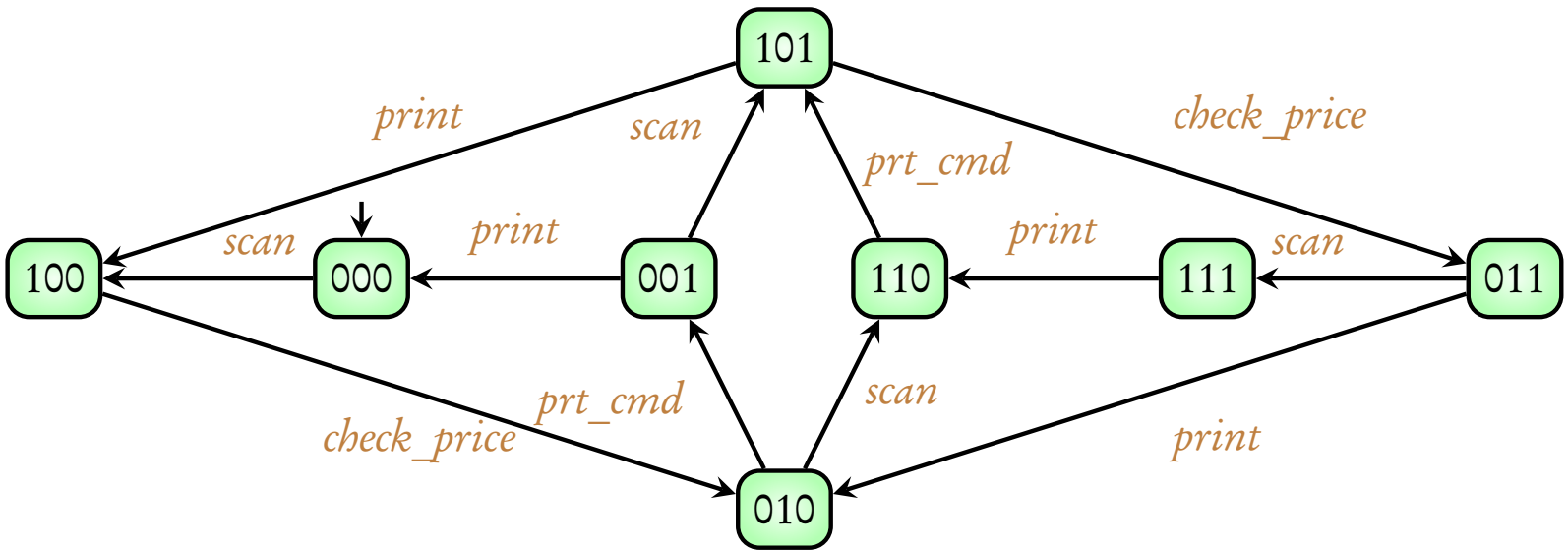


Booking Program (BP)

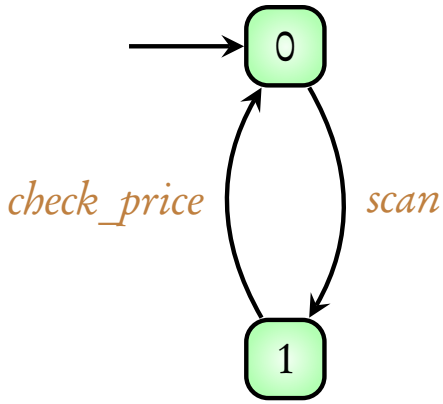


Printer (P)

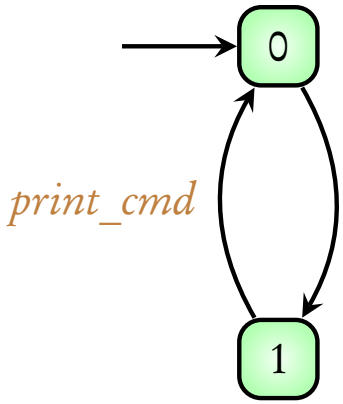




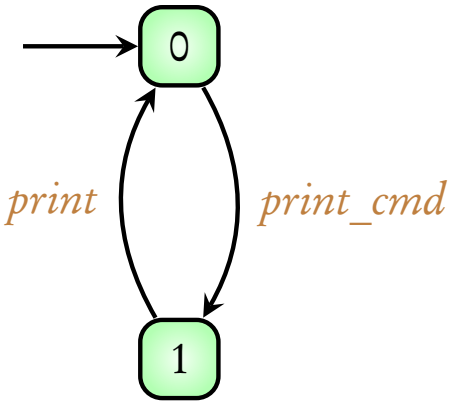
Bar-Code Reader (BCR)



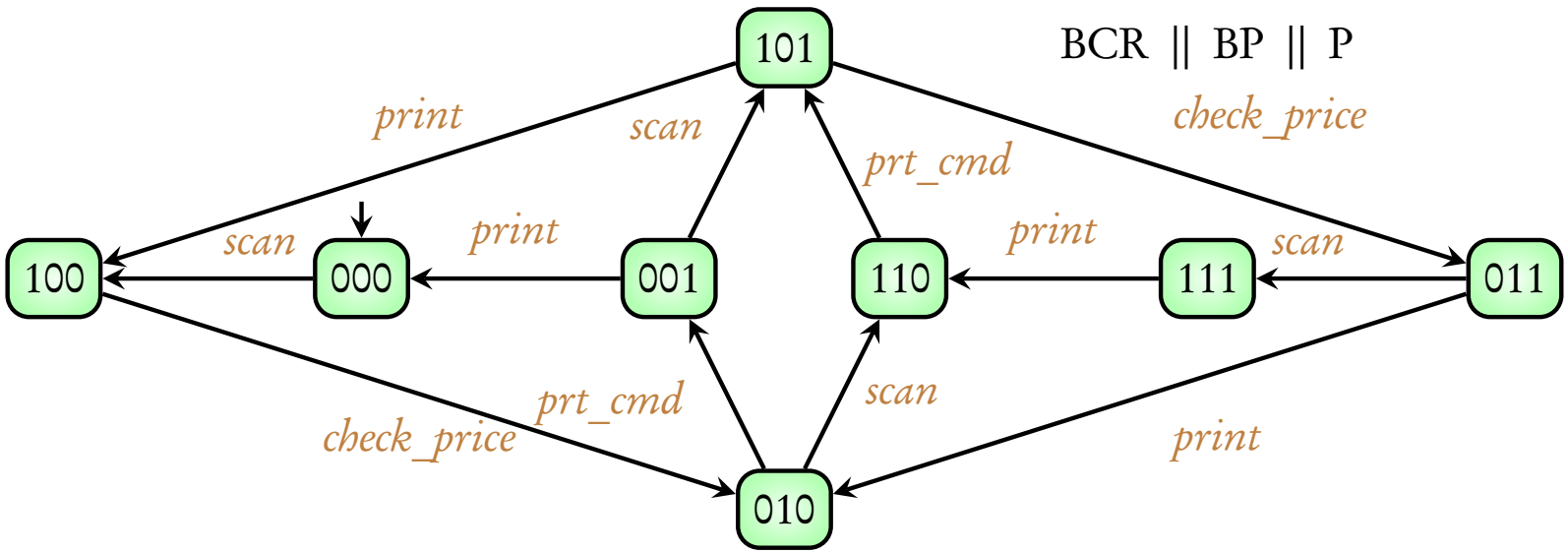
Booking Program (BP)



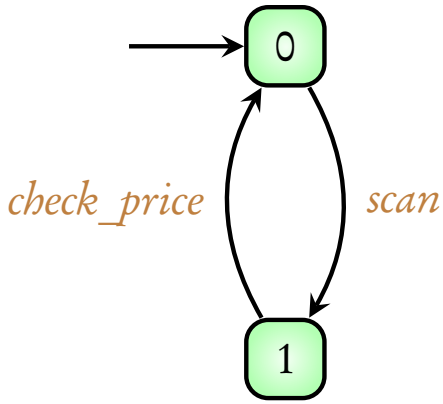
Printer (P)



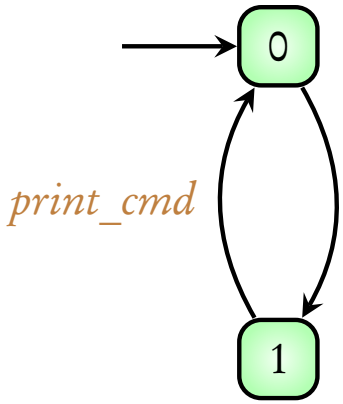
BCR || BP || P



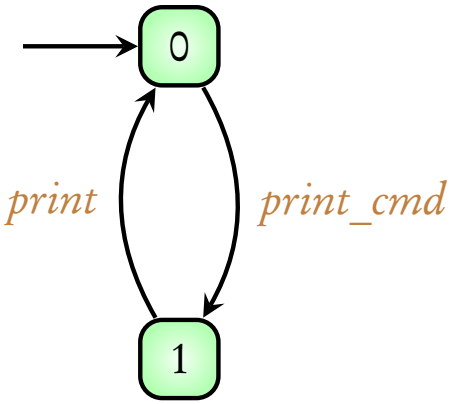
Bar-Code Reader (BCR)



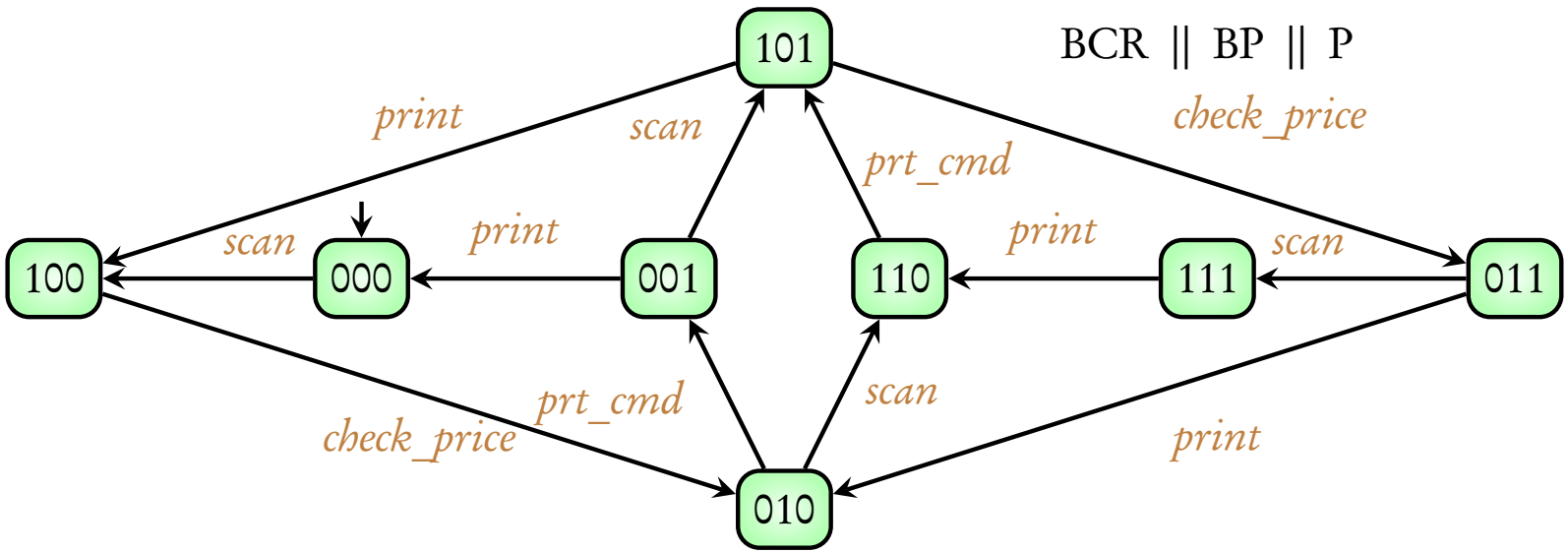
Booking Program (BP)



Printer (P)

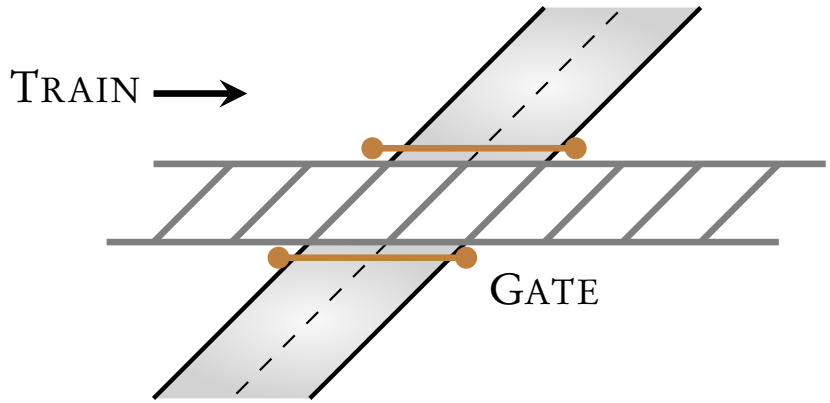


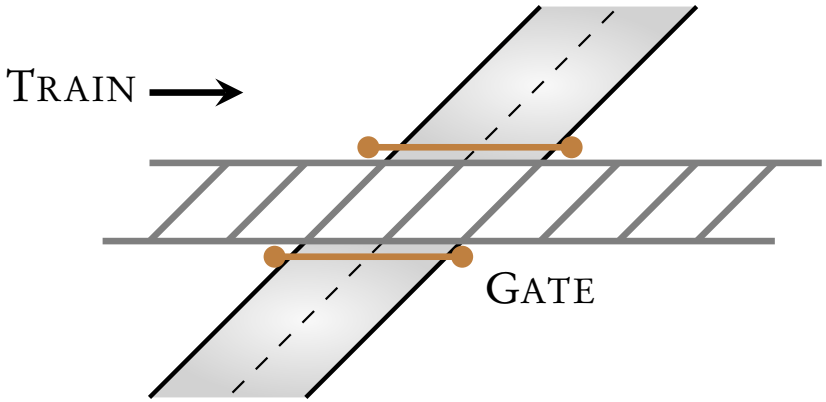
BCR || BP || P

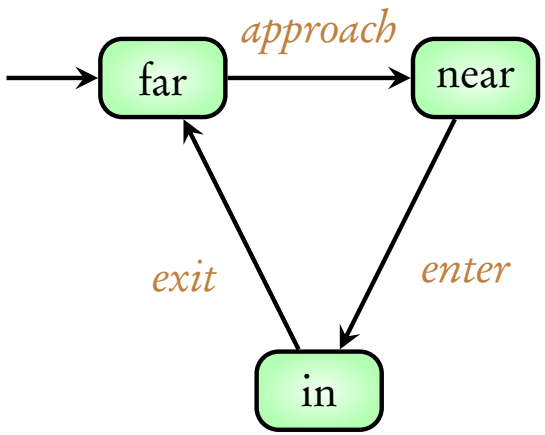


check_price, print_cmd: **Shared** actions (*also called* **handshaking** actions)

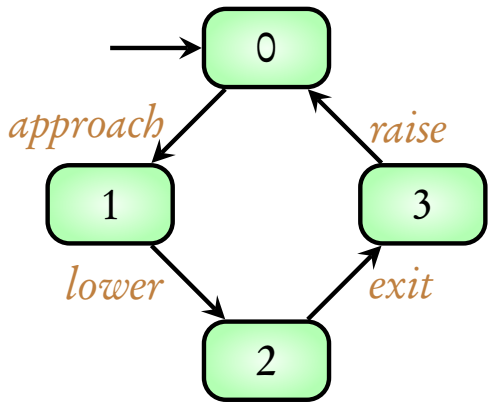
Next example: Train-Gate-Controller


TRAIN
GATE

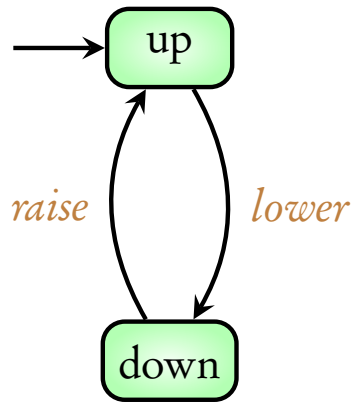




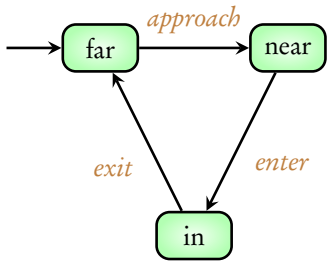
Train



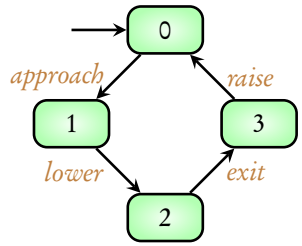
Controller



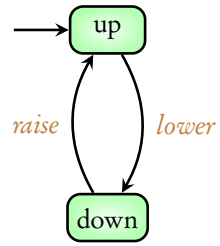
Gate



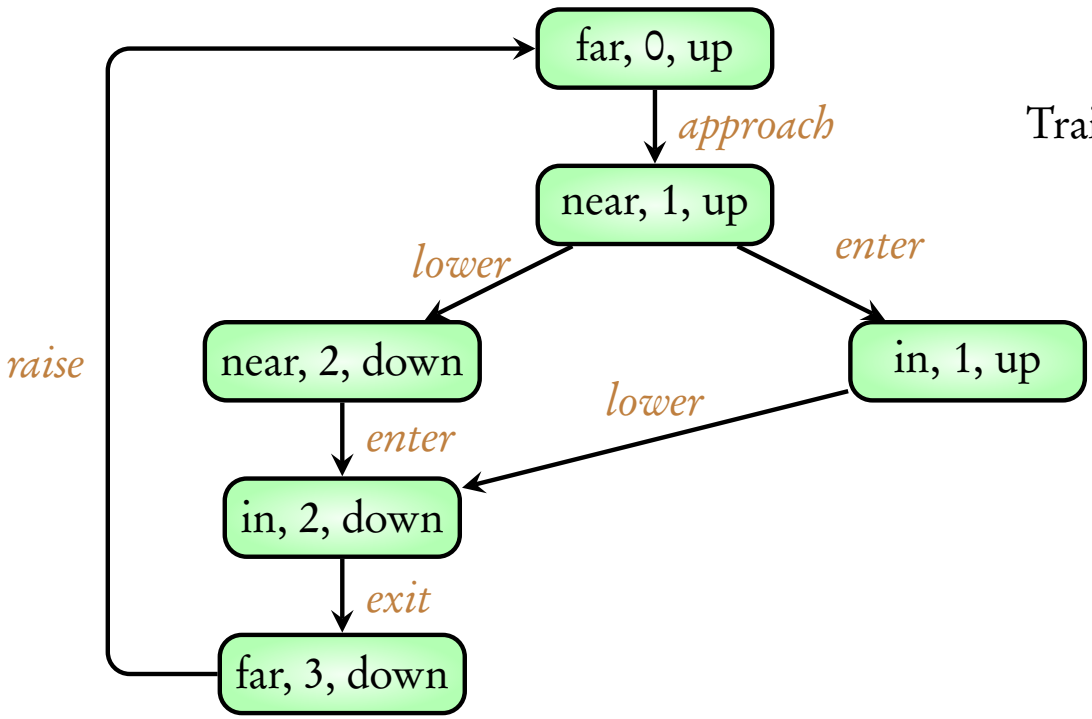
Train



Controller



Gate



Train || Controller || Gate

|| : **Handshake** operator

Independent

Interleaving

$TS_1 \;|||\; TS_2 \;|||\; \ldots \;|||\; TS_n$

Shared variables

$TS(\,PG_1 \;|||\; PG_2 \;|||\; \ldots \;|||\; PG_n\,)$

Mutual Exclusion

Shared actions

$TS_1 \;||\; TS_2$

Reference: Principles of Model Checking, *Baier and Katoen*, MIT Press (2008)
Pages 35 - 53

Week-1: Introduction to model checking

B. Srivathsan

Chennai Mathematical Institute

NPTEL-course

July - November 2015

Summary

- Course Overview
- **Module 1:** Transition systems, Modeling simple sequential programs
- **Module 2:** Modeling sequential hardware circuits
- **Module 3:** Modeling data-dependent programs
- **Module 4:** Modeling concurrent systems

Important concepts: Non-determinism, program graphs, interleaving and handshake operators