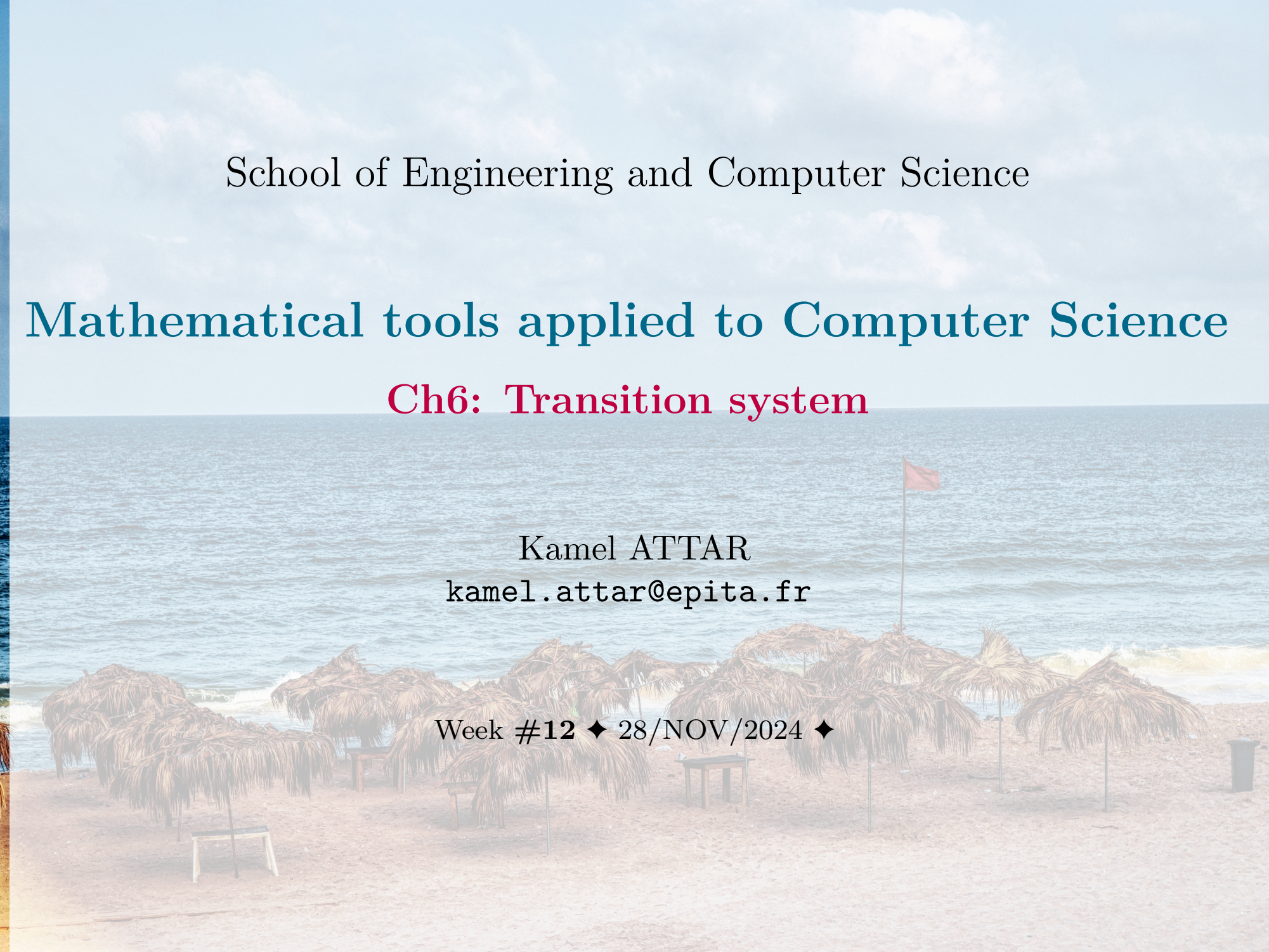


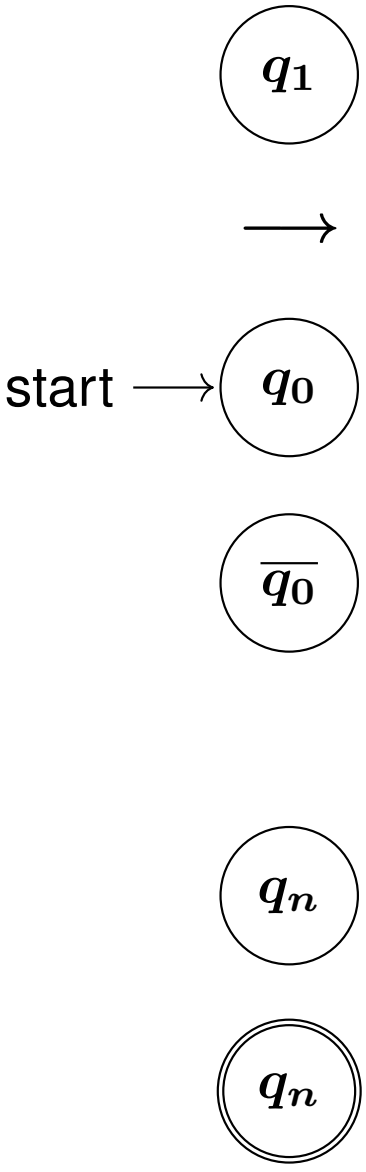
School of Engineering and Computer Science

Mathematical tools applied to Computer Science

Ch6: Transition system

Kamel ATTAR
kamel.attar@epita.fr

Week **#12** ✦ 28/NOV/2024 ✦



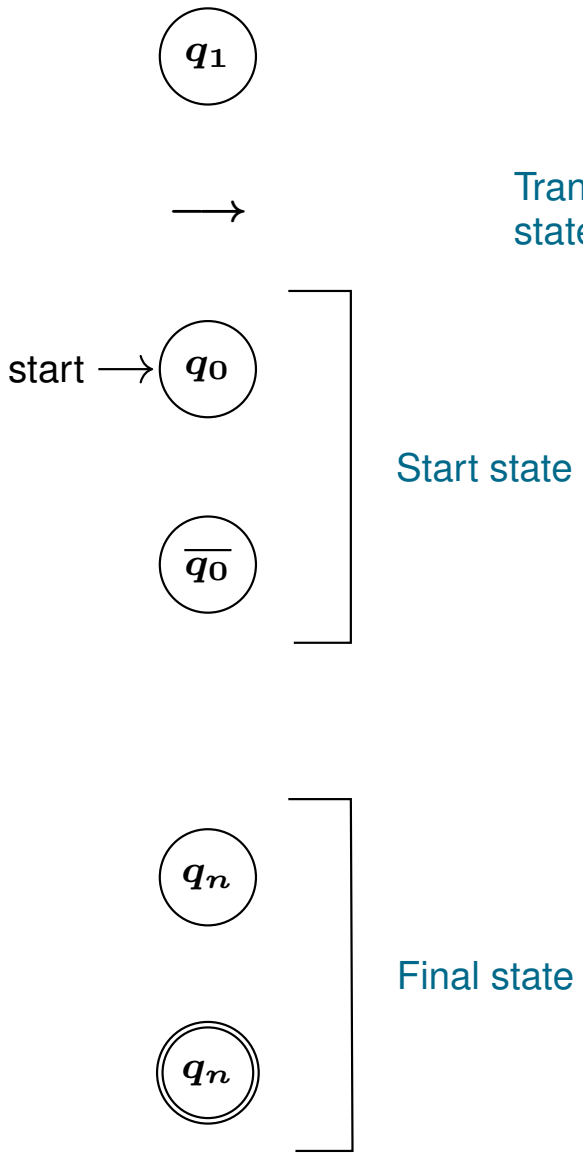
State

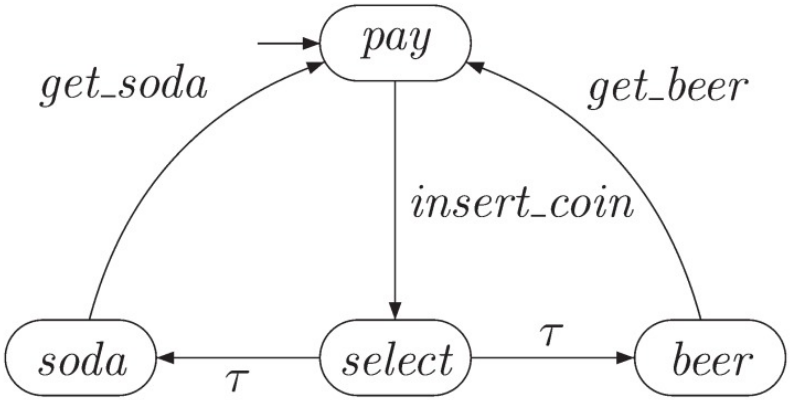
Transition from one state to another

Start state

Final state

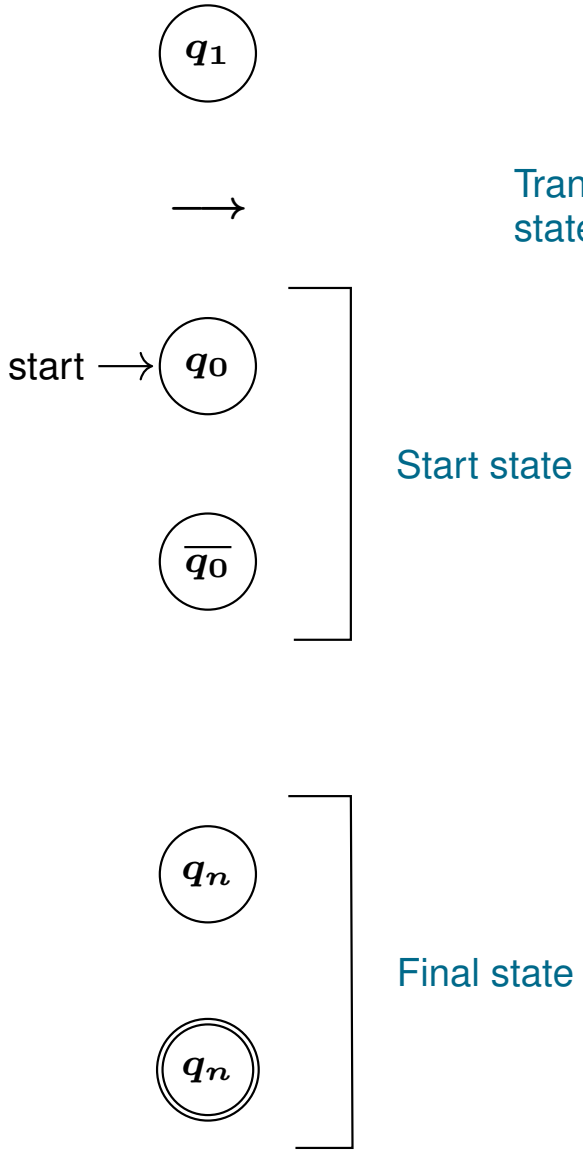
A transition system consists of a set of states and a set of transitions between those states. These transitions are labelled by actions and one state is designated as the initial state.

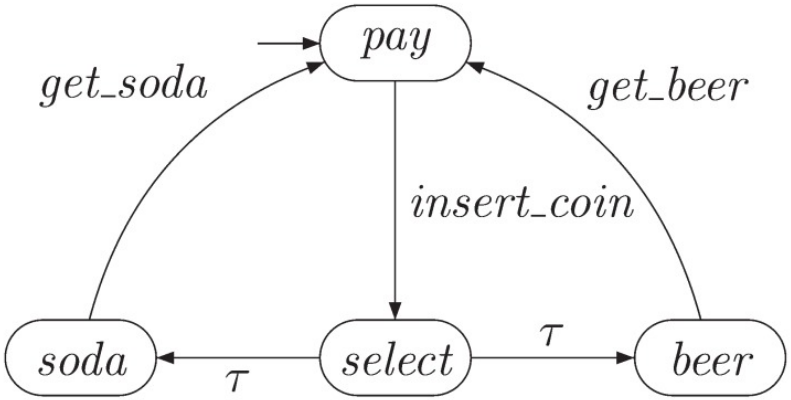




A transition system consists of a set of states and a set of transitions between those states. These transitions are labelled by actions and one state is designated as the initial state.

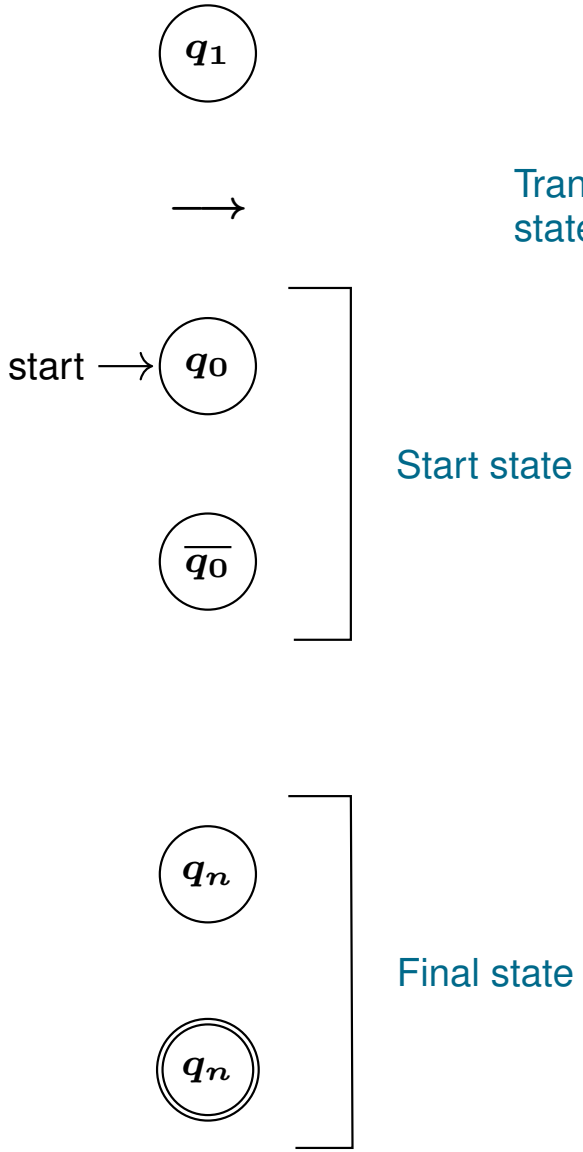
- Model describing the behavior of a system.

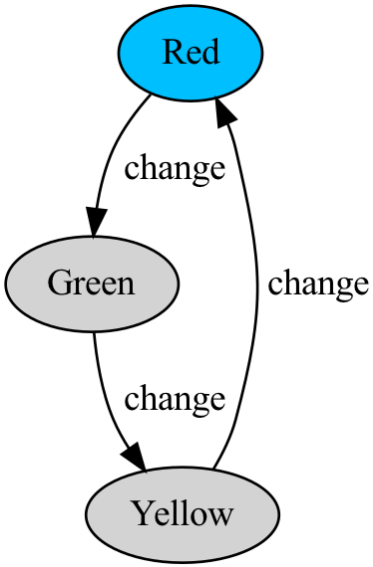


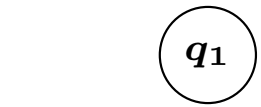


A transition system consists of a set of states and a set of transitions between those states. These transitions are labelled by actions and one state is designated as the initial state.

- Model describing the behavior of a system.
- Directed graphs: vertices = states, edges = transitions.

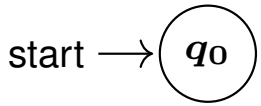






State

Transition from one state to another



Start state

A transition system consists of a set of states and a set of transitions between those states. These transitions are labelled by actions and one state is designated as the initial state.

- Model describing the behavior of a system.
- Directed graphs: vertices = states, edges = transitions.
- **State**: current mode of the system, current values of program variables, current color of a traffic light . . .
- **Transition** or state changes: mode switching, execution of a program instruction, change of color . .

Final state

⋆ Preliminaries ⋆

Definition

A **proposition** is a statement that can be either true or false, but not both.

Example

- “Traffic light is green” is a proposition.
- “The front pad is occupied” is a proposition.
- “Is the front pad?” is not a proposition.

Definition

An **atomic proposition** is one whose truth or falsity does not depend on the truth or falsity of any other proposition.

Example

- All propositions above are atomic propositions.
- “If traffic light is green, the car can drive” is not an atomic proposition.

✯ Formal definition✯

Definition (Transition system (TS))

Tuple $\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$ where:

✯ Formal definition ✯

Definition (Transition system (TS))

Tuple $\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$ where:

- S the set of states,

✯ Formal definition ✯

Definition (Transition system (TS))

Tuple $\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$ where:

- S the set of states,
- Act the set of actions,

✯ Formal definition✯

Definition (Transition system (TS))

Tuple $\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$ where:

- S the set of states,
- Act the set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ the transition relation,

✯ Formal definition ✯

Definition (Transition system (TS))

Tuple $\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$ where:

- S the set of states,
- Act the set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ the transition relation,
- $S_0 \subseteq S$ the set of initial states,

✯ Formal definition ✯

Definition (Transition system (TS))

Tuple $\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$ where:

- S the set of states,
- Act the set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ the transition relation,
- $S_0 \subseteq S$ the set of initial states,
- AP the set of atomic propositions, and

✯ Formal definition ✯

Definition (Transition system (TS))

Tuple $\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$ where:

- S the set of states,
- Act the set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ the transition relation,
- $S_0 \subseteq S$ the set of initial states,
- AP the set of atomic propositions, and
- $L : S \longrightarrow 2^{AP}$ the labeling function.

✯ Formal definition ✯

Definition (Transition system (TS))

Tuple $\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$ where:

- S the set of states,
- Act the set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ the transition relation,
- $S_0 \subseteq S$ the set of initial states,
- AP the set of atomic propositions, and
- $L : S \longrightarrow 2^{AP}$ the labeling function.

  2^{AP}: the powerset (set of all subsets) of AP.

✯ Formal definition ✯

Definition (Transition system (TS))

Tuple $\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$ where:

- S the set of states,
- Act the set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ the transition relation,

  i.e. Transitions have the form (s, α, s') where $s, s' \in S$ and $\alpha \in Act$.
- $S_0 \subseteq S$ the set of initial states,
- AP the set of atomic propositions, and
- $L : S \longrightarrow 2^{AP}$ the labeling function.

  2^{AP}: the powerset (set of all subsets) of AP.

★ Formal definition★

Definition (Transition system (TS))

Tuple $\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$ where:

- S the set of states,
- Act the set of actions,
- $\longrightarrow \subseteq S \times Act \times S$ the transition relation,

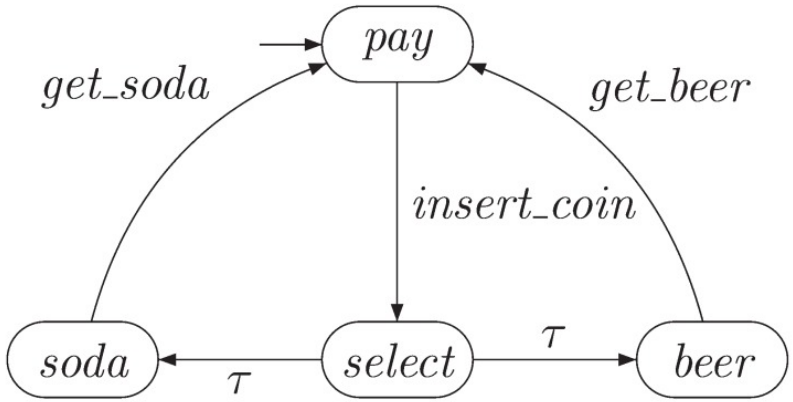
  i.e. Transitions have the form (s, α, s') where $s, s' \in S$ and $\alpha \in Act$.
- $S_0 \subseteq S$ the set of initial states,
- AP the set of atomic propositions, and
- $L : S \longrightarrow 2^{AP}$ the labeling function.

  2^{AP}: the powerset (set of all subsets) of AP.

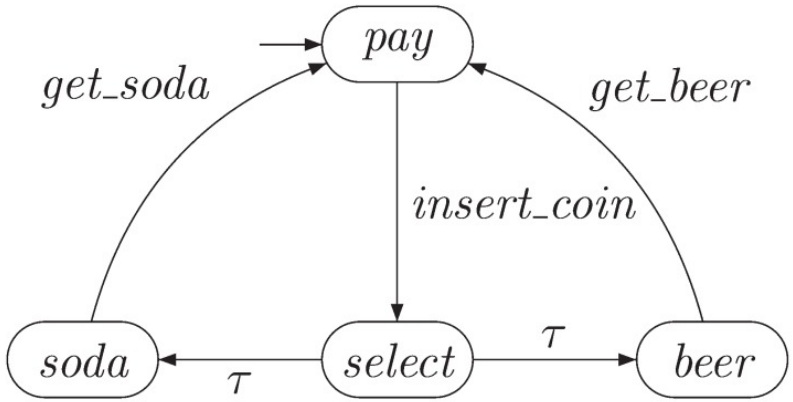
S and Act are either finite or countably infinite

Notation: sometimes we write $s \xrightarrow{\alpha} s'$ instead of $(s, \alpha, s') \in \longrightarrow$.

✯ Transition system for beverage machine✯

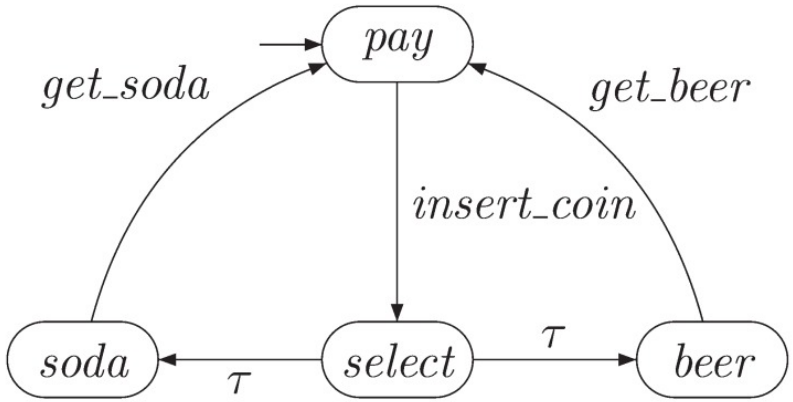


✯ Transition system for beverage machine✯



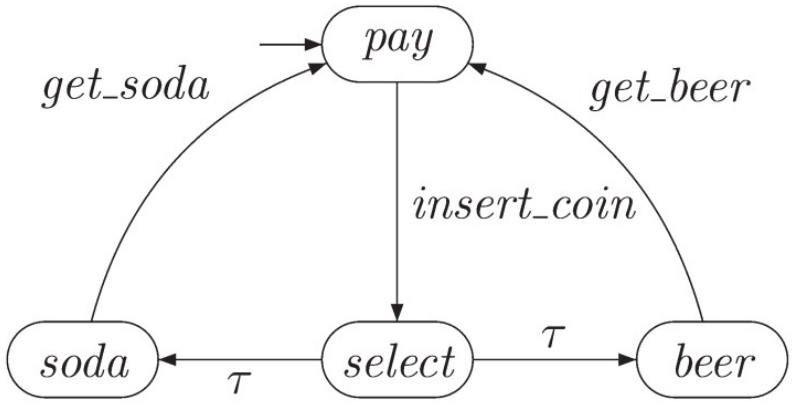
- state space $S = \{pay, select, beer, soda\}$

✯ Transition system for beverage machine✯



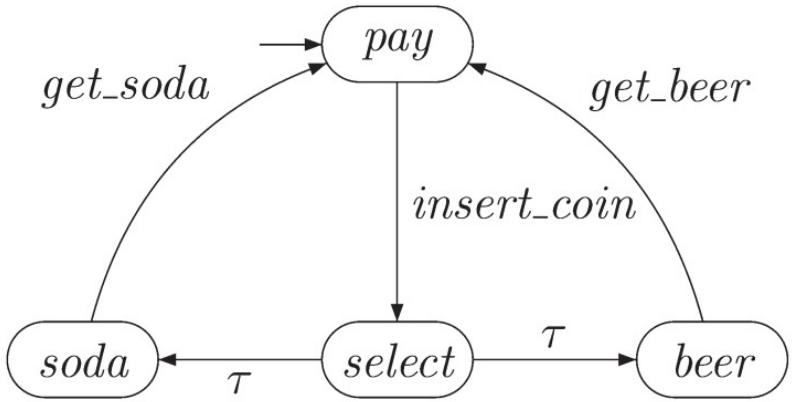
- state space $S = \{pay, select, beer, soda\}$
- set of initial states $S_0 = \{pay\}$

✯ Transition system for beverage machine✯



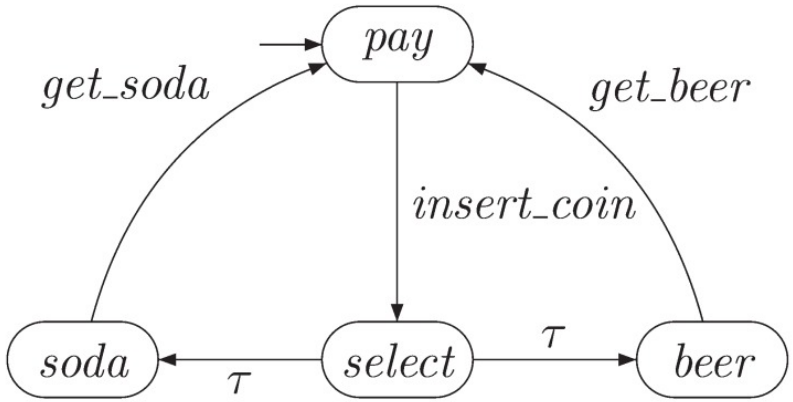
- state space $S = \{pay, select, beer, soda\}$
- set of initial states $S_0 = \{pay\}$
- $Action = \{insert\_coin, get\_beer, get\_soda, \tau\}$,

✯ Transition system for beverage machine✯



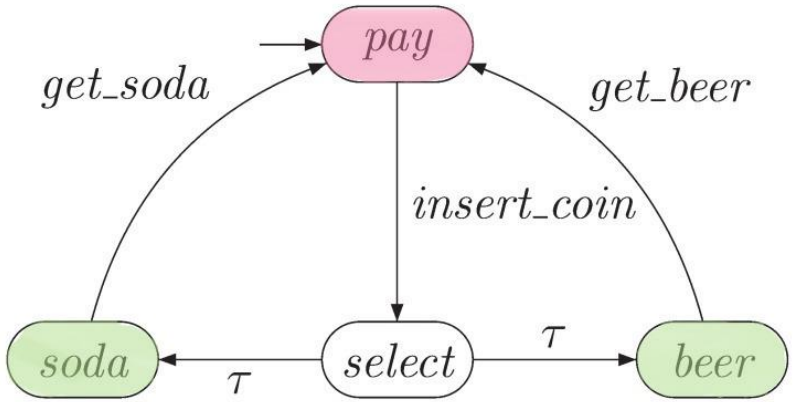
- state space $S = \{pay, select, beer, soda\}$
- set of initial states $S_0 = \{pay\}$
- $Action = \{insert\_coin, get\_beer, get\_soda, \tau\}$,
- Some Transition: $pay \xrightarrow{insert\_coin} select$ and $select \xrightarrow{\tau} beer$

✯ Transition system for beverage machine✯

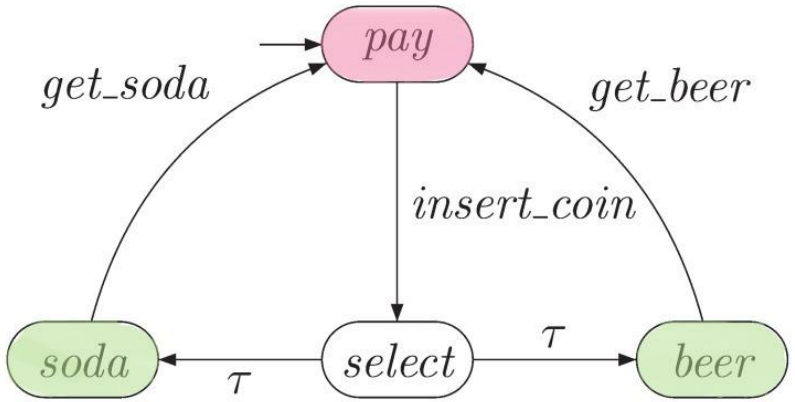


- state space $S = \{pay, select, beer, soda\}$
- set of initial states $S_0 = \{pay\}$
- $Action = \{insert\_coin, get\_beer, get\_soda, \tau\}$,
- Some Transition: $pay \xrightarrow{insert\_coin} select$ and $select \xrightarrow{\tau} beer$

What about the labeling?

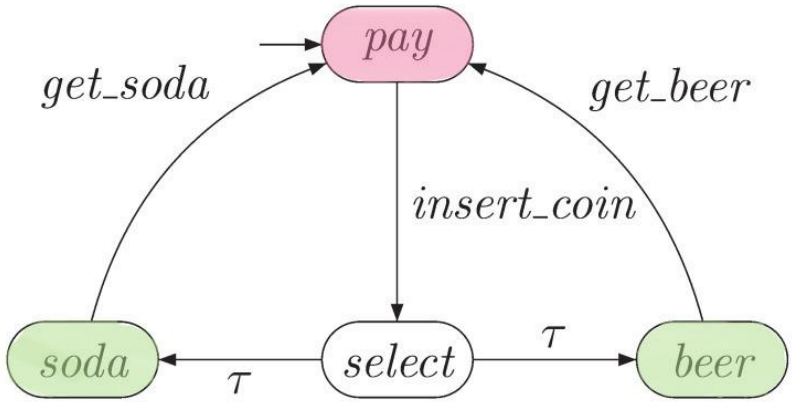


Depends on what we want to model!



Depends on what we want to model!

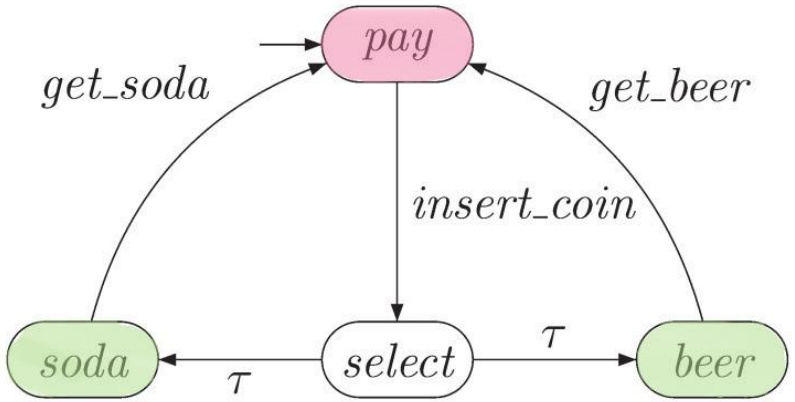
Say the property is “the vending machine only delivers a drink after providing a coin”



Depends on what we want to model!

Say the property is “the vending machine only delivers a drink after providing a coin”

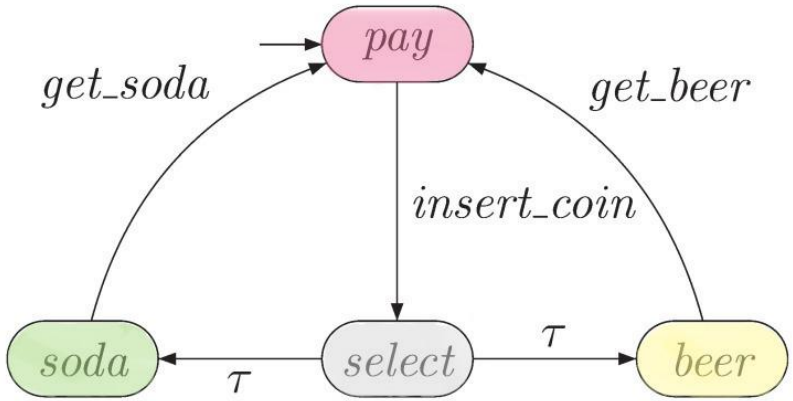
- $AP = \{paid, drink\}$,



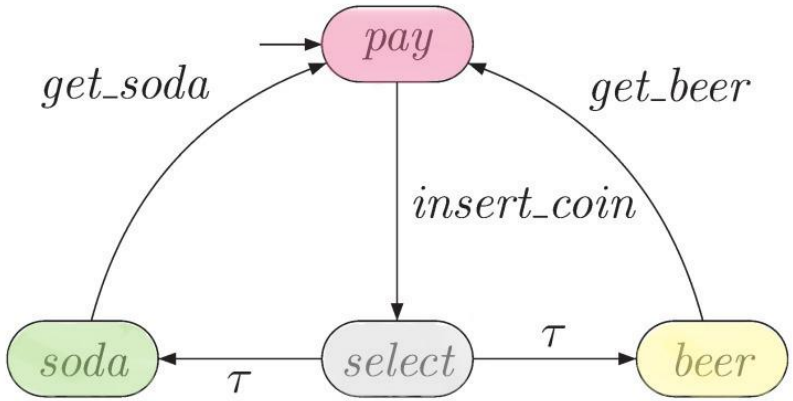
Depends on what we want to model!

Say the property is “the vending machine only delivers a drink after providing a coin”

- $AP = \{paid, drink\}$,
- $L(pay) = paid$, $L(select) = \emptyset$ and $L(soda) = L(beer) = drink$

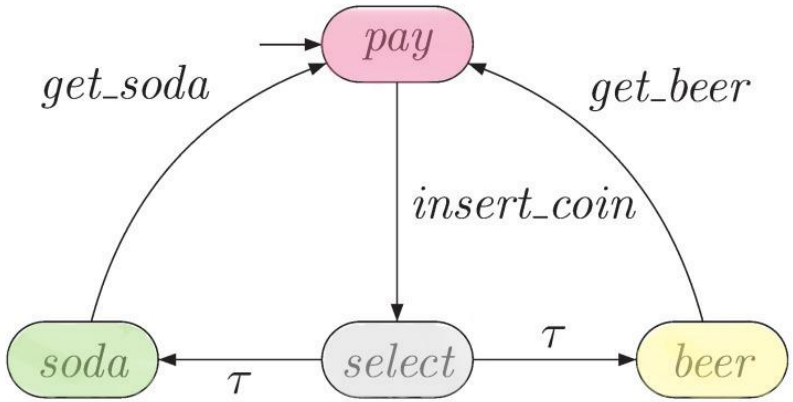


Simple choice: $\forall s,\ L(s) = \{s\}$.



Simple choice: $\forall s,\ L(s) = \{s\}$.

- $AP = S = \{pay, select, beer, soda\}$,
- $L(pay) = pay$, $L(select) = select$, $L(soda) = soda$ and $= L(beer) = beer$



Simple choice: $\forall s,\ L(s) = \{s\}$.

- $AP = S = \{pay, select, beer, soda\}$,
- $L(pay) = pay$, $L(select) = select$, $L(soda) = soda$ and $= L(beer) = beer$

- When the labeling is not important, we often omit it.
- We do the same for actions or simply use internal actions τ.

✎ **Exercise 1.** Draw the graph of a transition system $\mathcal{T}$ with:

$$S = \{s_1, s_2, s_3\}, \quad S_0 = \{s_1\}, \quad R = \{(s_1, s_2), (s_2, s_1), (s_3, s_2)\}$$

✎ **Exercise 2.** Consider the example of a traffic light. Initially the red light is on. After some time, the traffic light switches from the red light to a state where both the red and the yellow light are on. From red/yellow, it switches to green, from green to yellow, and from yellow back to red, and so on. Model the traffic light as transition system.

✎ **Exercise 3.** Create a state transition diagram for the following system. A door can be in one of the following three states:

- Opened
- Closed
- Locked

The door is initially open.

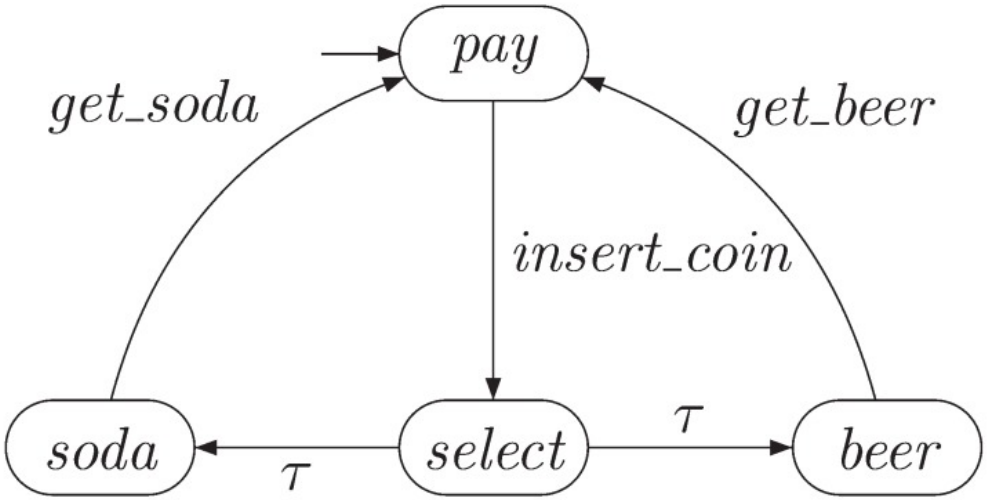
- An open door can be closed if the condition that the doorway is empty is met.
- A closed door can be opened.
- A closed door can be locked.
- A locked door can be unlocked.

✎ **Exercise 4.** State-transition diagrams are also useful for showing the working of algorithms that involve a finite number of states. The following algorithm is for a three-digit combination lock where the correct combination to unlock is '367'. The initial state is Locked, each correct digit changes the state, until the combination unlocks the lock. An incorrect digit returns the lock to the original locked state.

(a) Write an algorithm that describes the above informations.

(b) Model the combination lock as transition system.

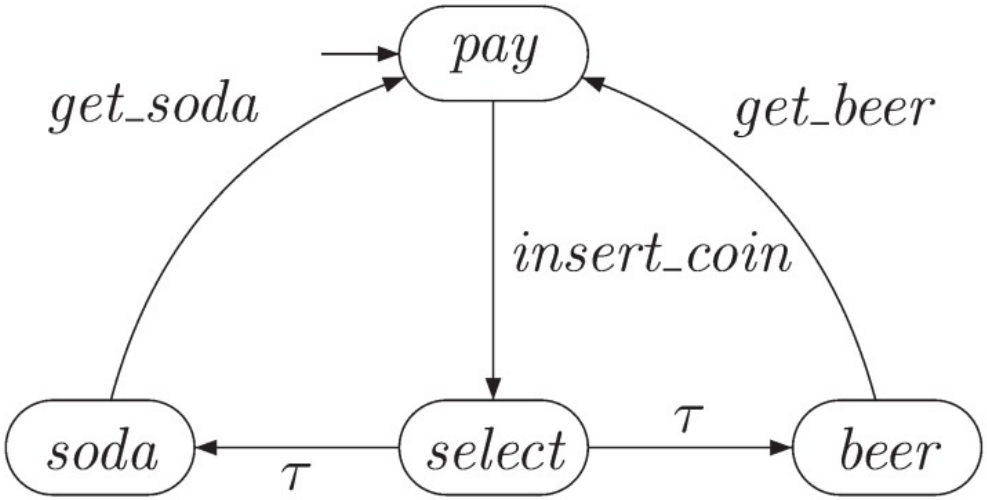
✯ Semantics of TS: non-determinism ✯



When two actions are possible (select), the choice is made non-deterministically!

Also true for the initial state if $|S_0| > 1$.

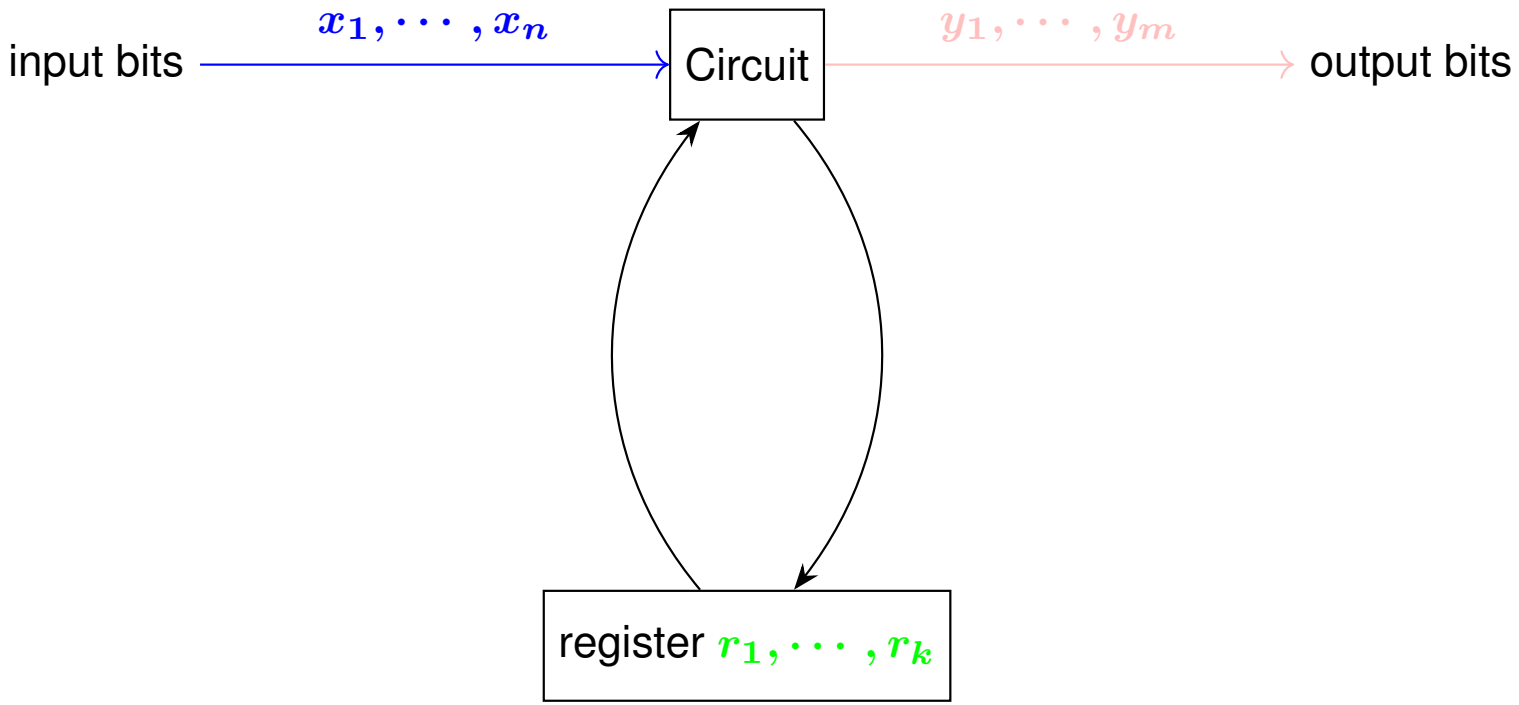
✯ Semantics of TS: non-determinism ✯

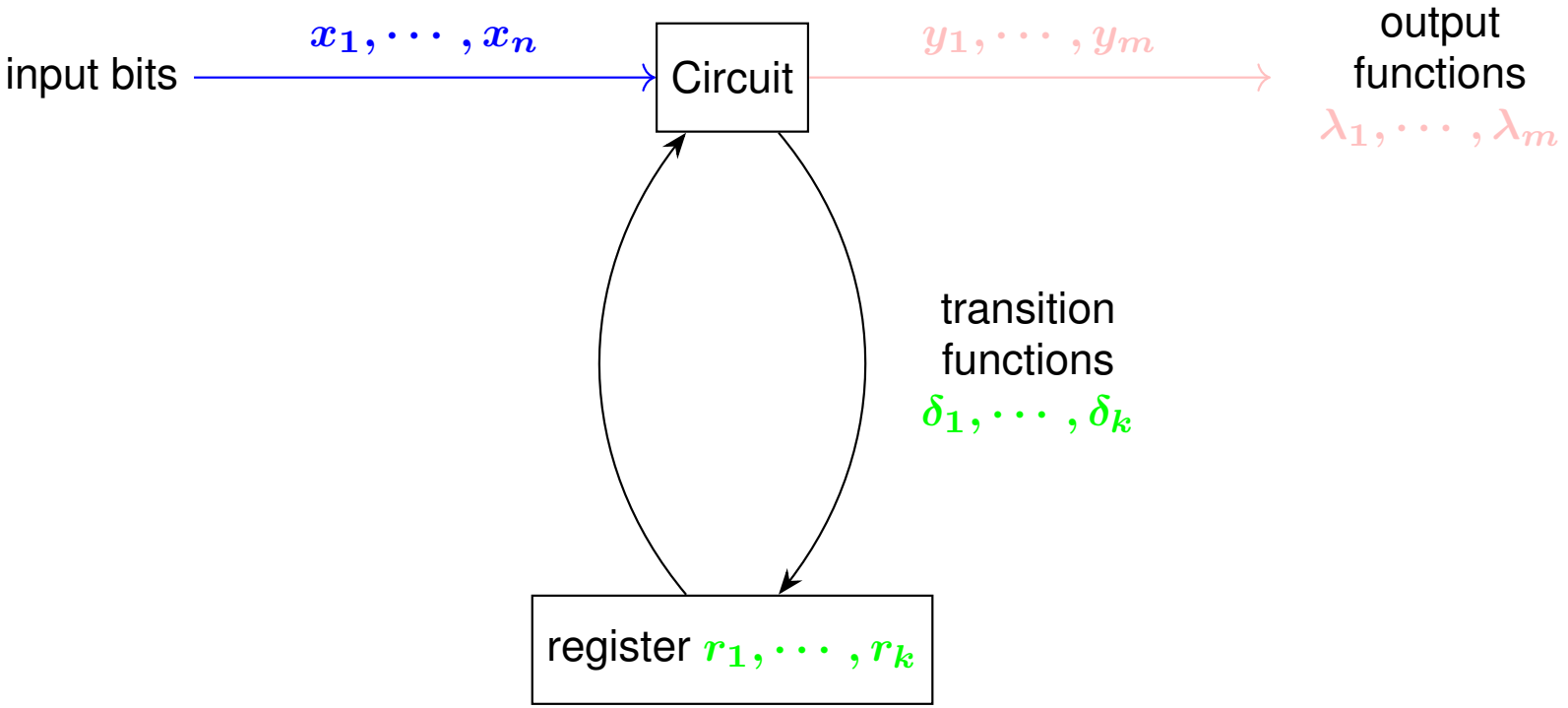


When two actions are possible (select), the choice is made non-deterministically!

Also true for the initial state if $|S_0| > 1$.

- Meaningful to model interleaving of || executions for example.
- Also for abstraction or to model an uncontrollable environment (here, drink choice by the user).


input bits
$x_1, \cdots, x_n$
Circuit
$y_1, \cdots, y_m$
output bits
register $r_1, \cdots, r_k$



inputs values $a_1, \cdots, a_n$ for the input variables $x_1, \cdots, x_n$

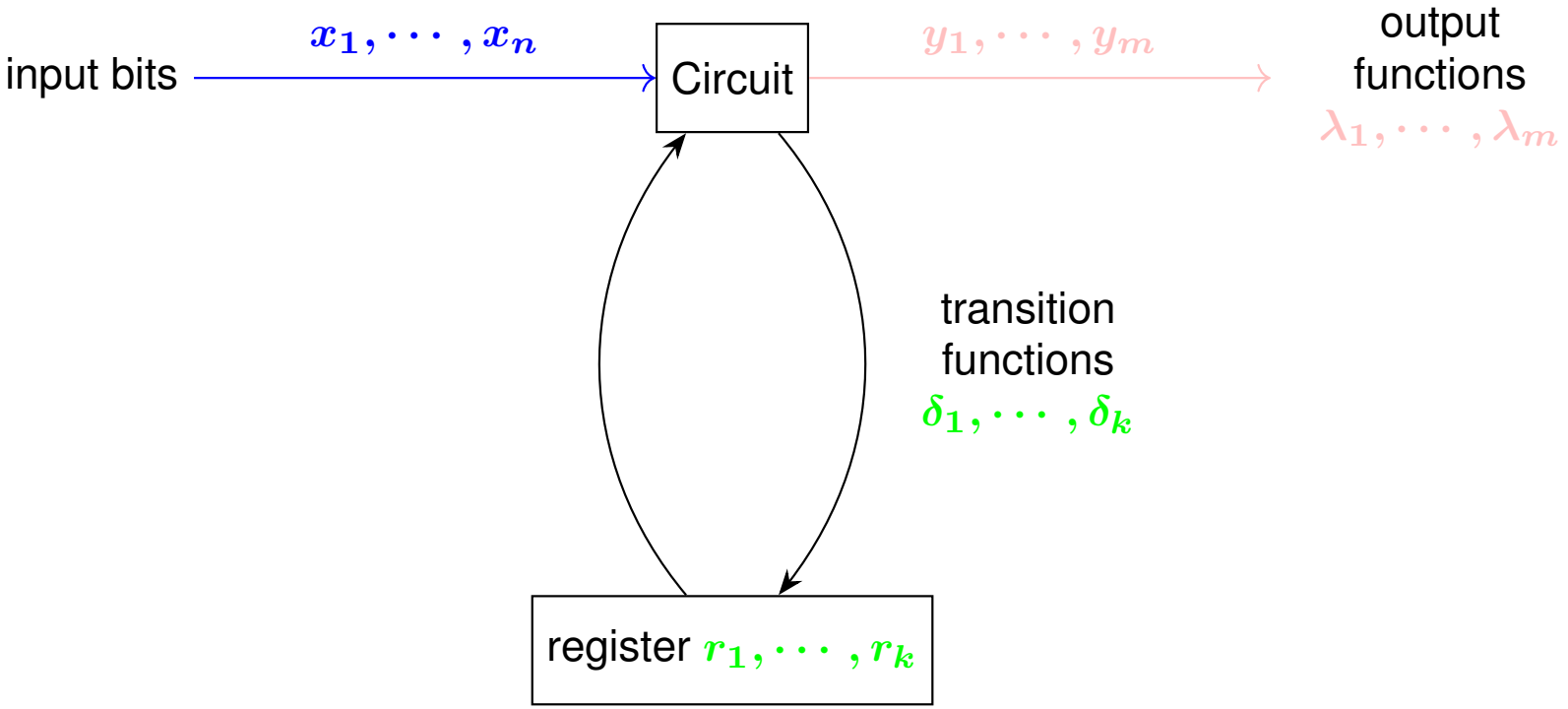
+

current values $c_1, \cdots, c_k$ of the registes $r_1, \cdots, r_k$

$\longmapsto$

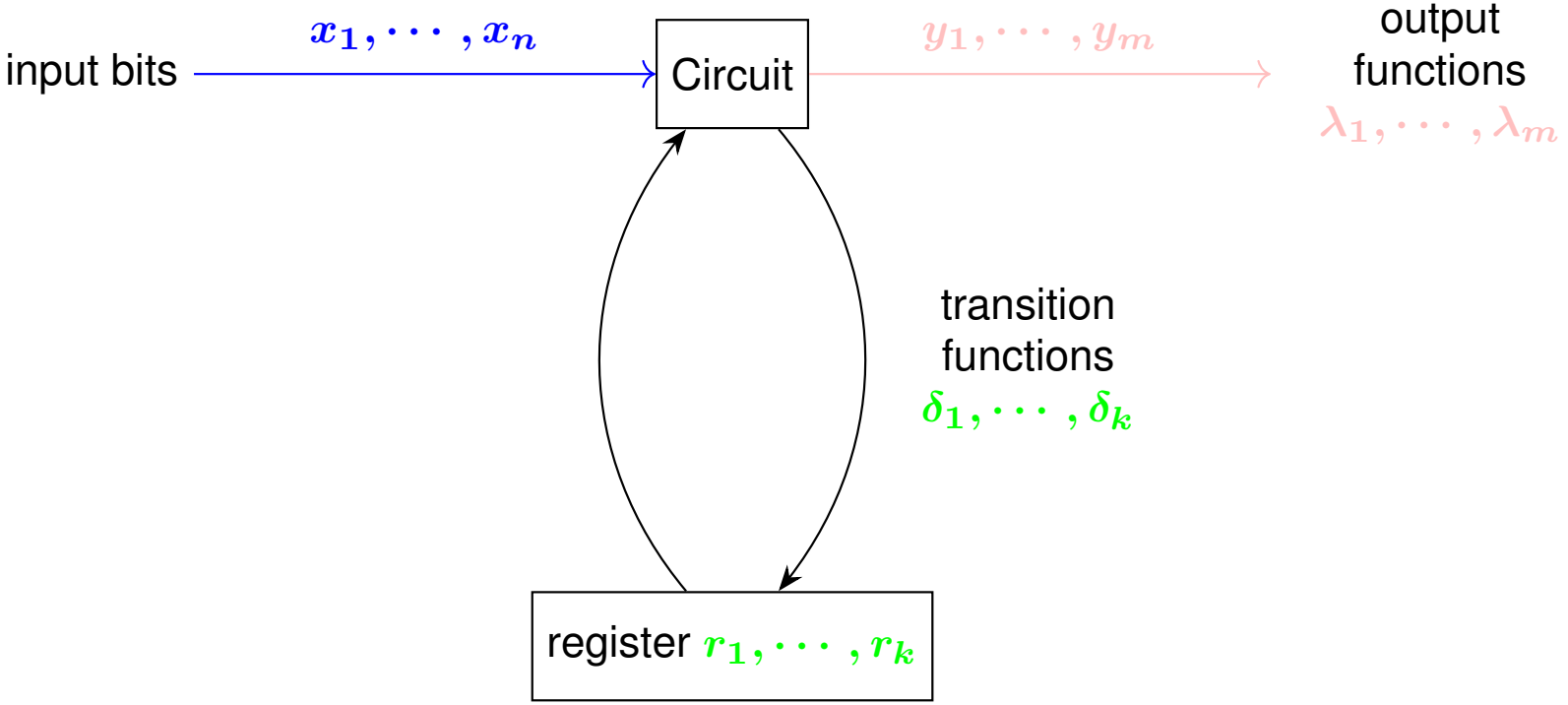
output value $\lambda_i(\cdots)$ for the ouput variable y_i

next value $\delta_j(\cdots)$ for register r_j.



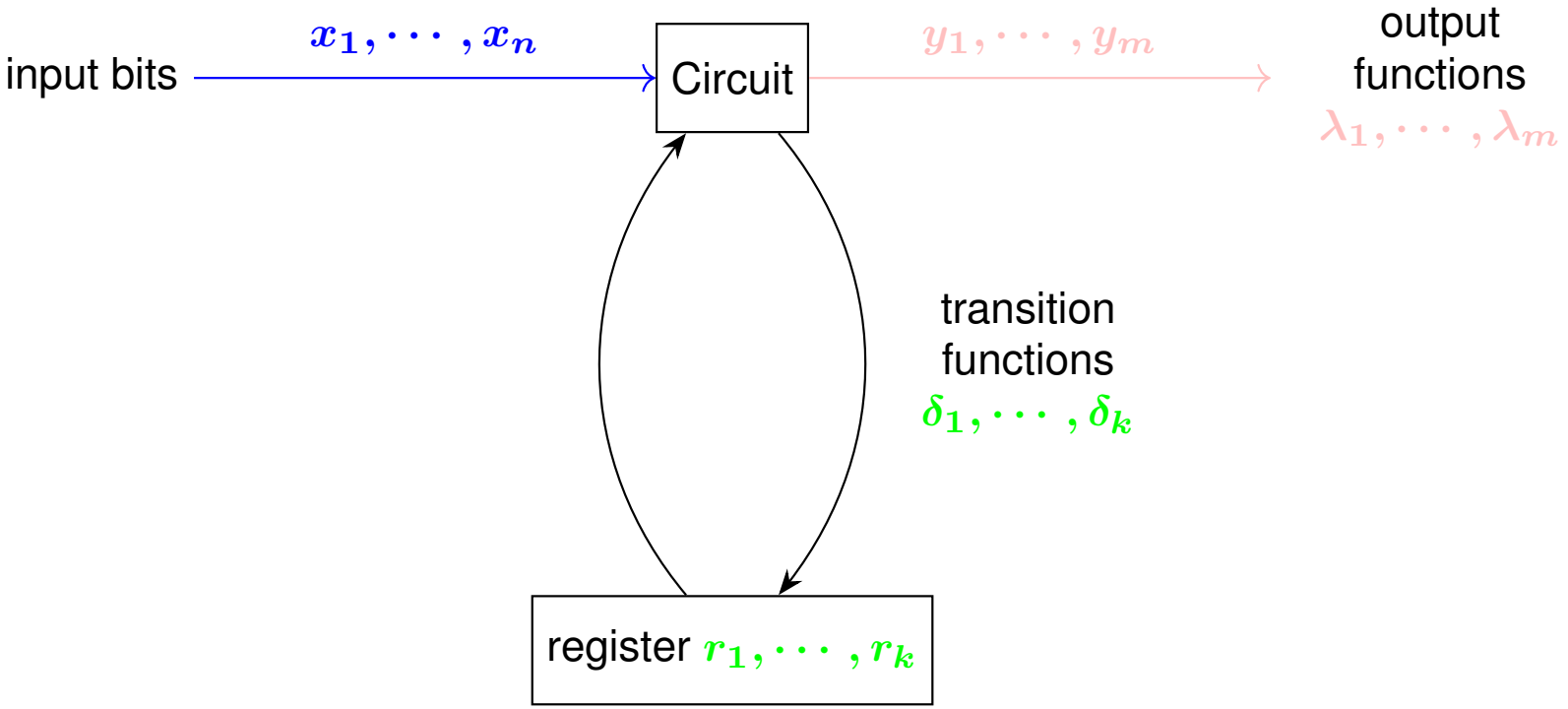
Transition system:

- states: evaluations of $x_1, \cdots, x_n$ and $r_1, \cdots, r_k$



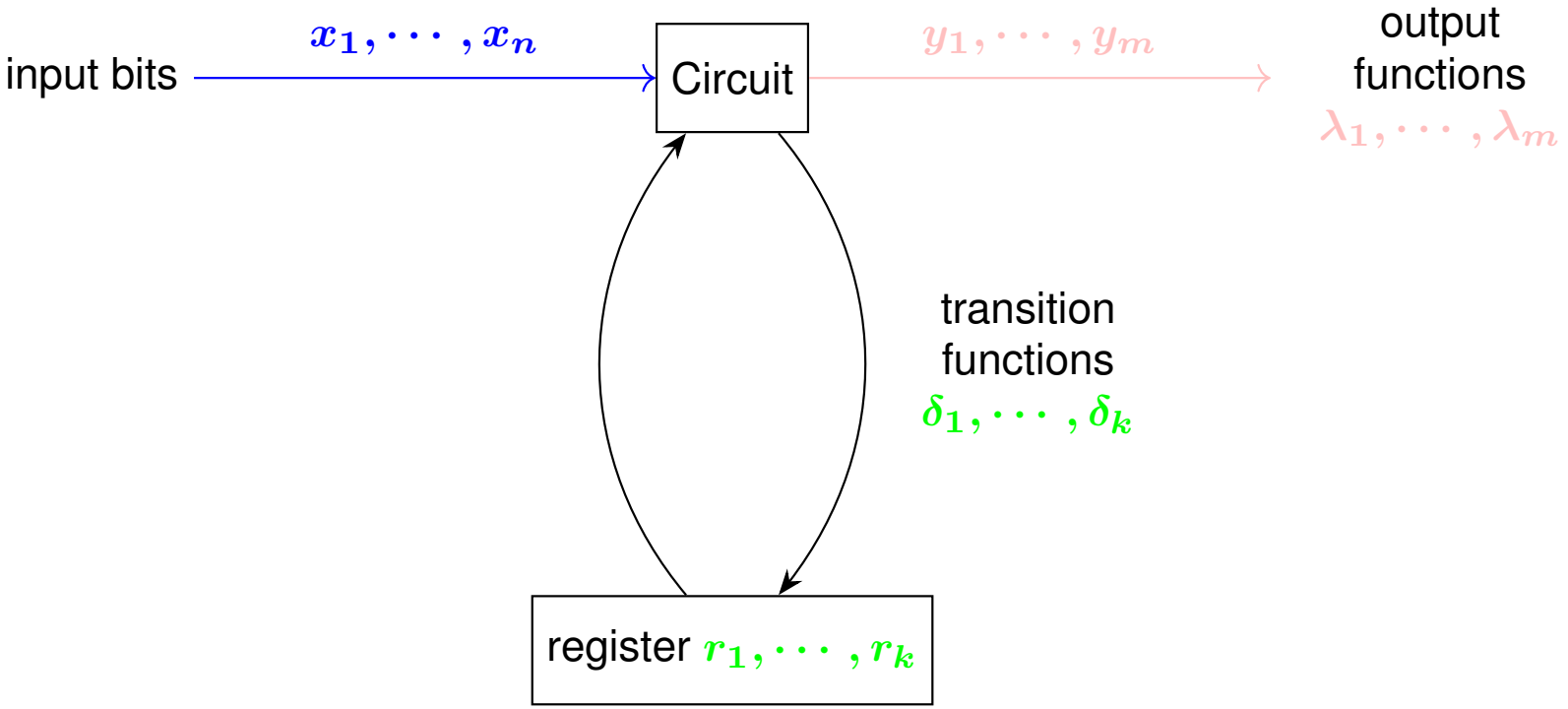
Transition system:

- states: evaluations of $x_1, \cdots, x_n$ and $r_1, \cdots, r_k$
- transitions represent the stepwise behavior



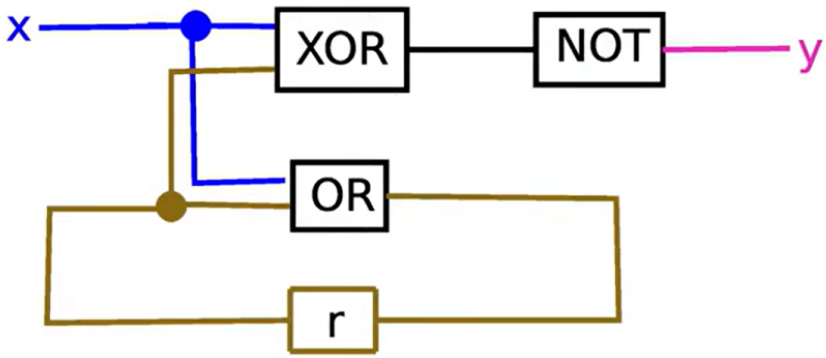
Transition system:

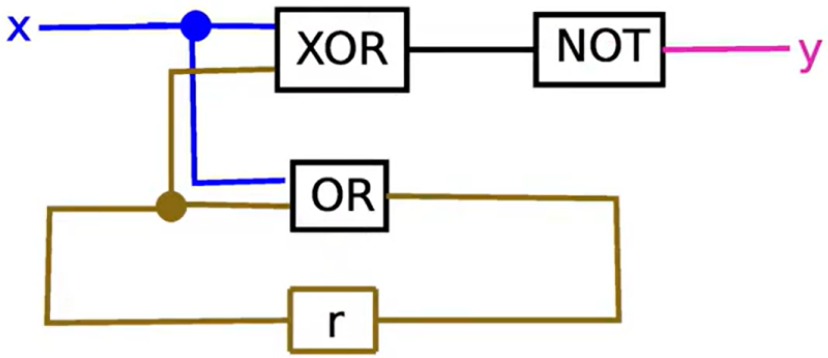
- states: evaluations of $x_1, \cdots, x_n$ and $r_1, \cdots, r_k$
- transitions represent the stepwise behavior
- values of input bits change nondeterministically



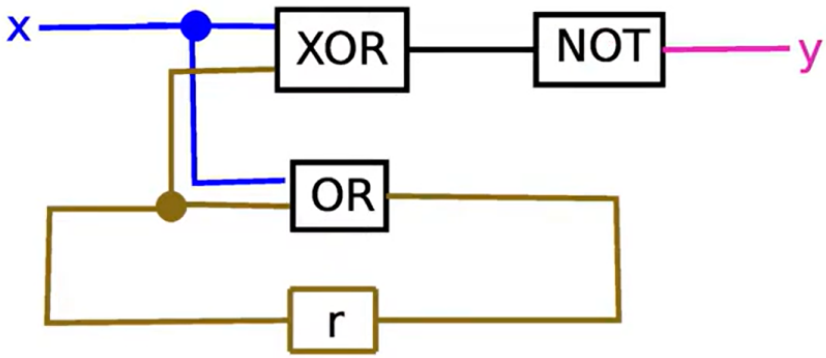
Transition system:

- states: evaluations of $x_1, \cdots, x_n$ and $r_1, \cdots, r_k$
- transitions represent the stepwise behavior
- values of input bits change nondeterministically
- atomic propositions: $x_1, \cdots, x_n$, $r_1, \cdots, r_k$ and $\lambda_1, \cdots, \lambda_m$.

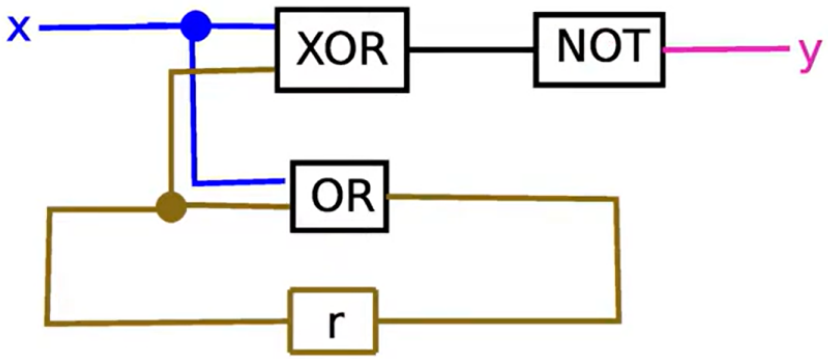

x
XOR
NOT
y
OR
r



- output function: $\lambda_y =$
- transition function: $\delta_r =$



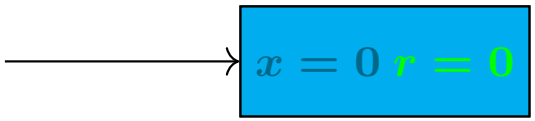
- output function: $\lambda_y = \neg(x \oplus r)$
- transition function: $\delta_r = x \vee r$

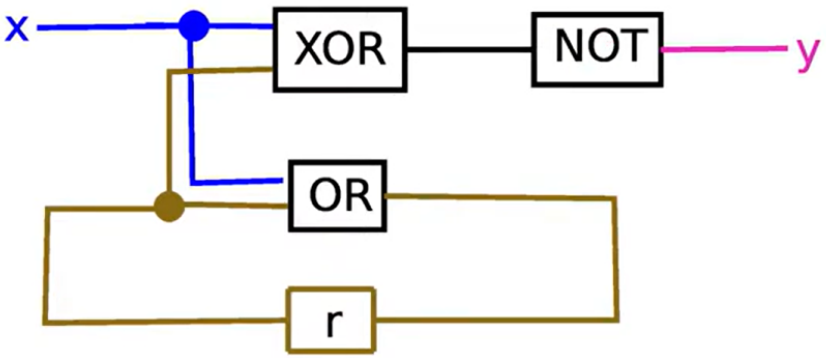


- output function: $\lambda_y = \neg(x \oplus r)$
- transition function: $\delta_r = x \vee r$

transition system

initial register evaluation: $r = 0$

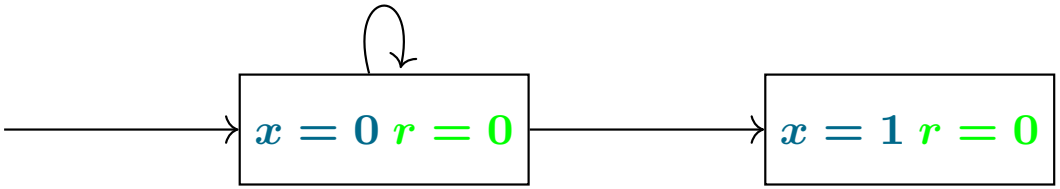


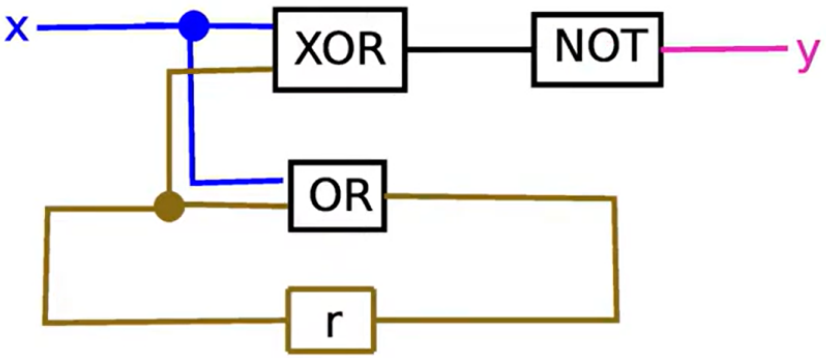


- output function: $\lambda_y = \neg(x \oplus r)$
- transition function: $\delta_r = x \vee r$

transition system

initial register evaluation: $r = 0$

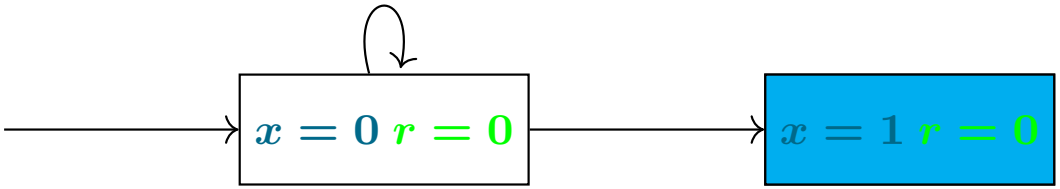


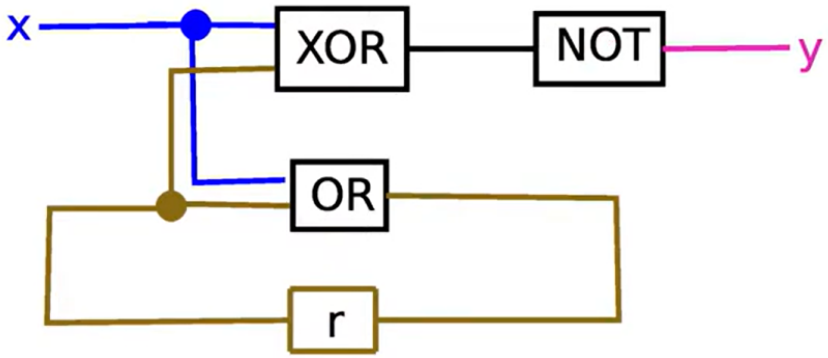


- output function: $\lambda_y = \neg(x \oplus r)$
- transition function: $\delta_r = x \vee r$

transition system

initial register evaluation: $r = 0$

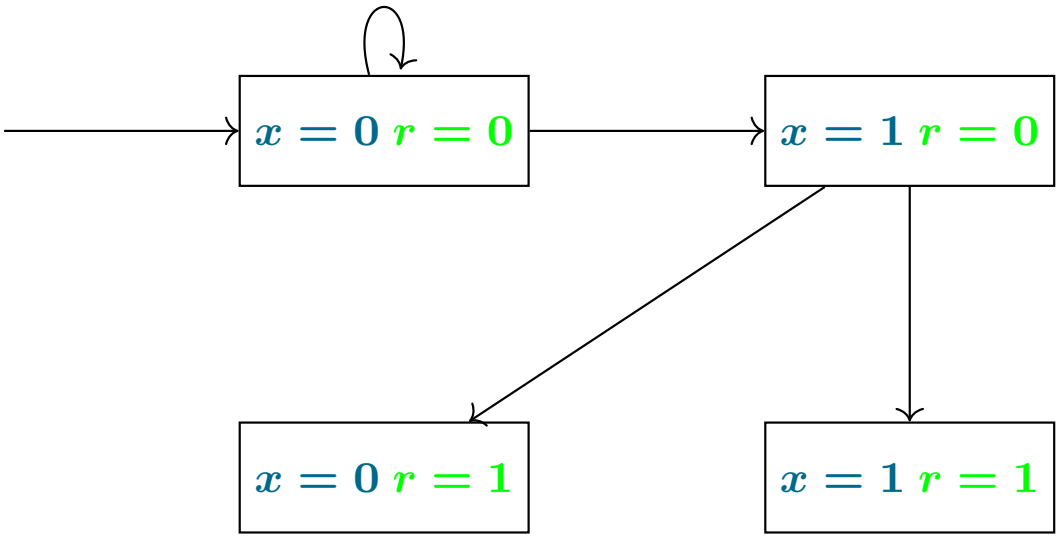


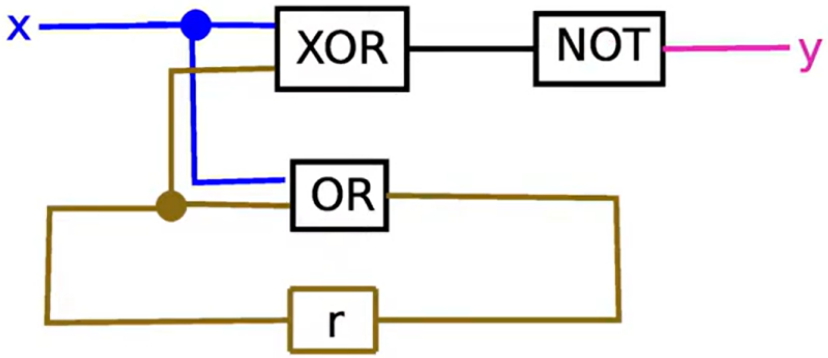


- output function: $\lambda_y = \neg(x \oplus r)$
- transition function: $\delta_r = x \vee r$

transition system

initial register evaluation: $r = 0$

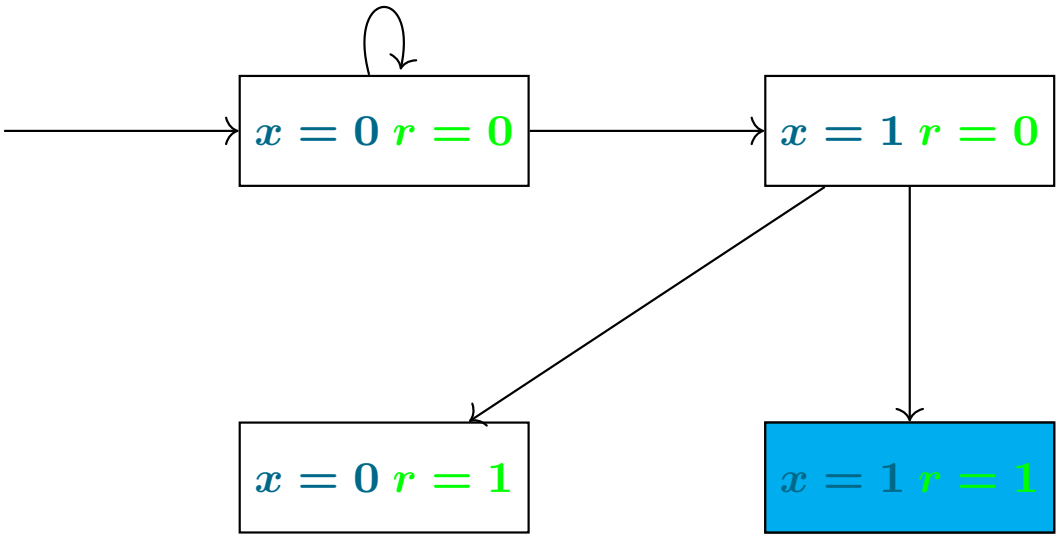


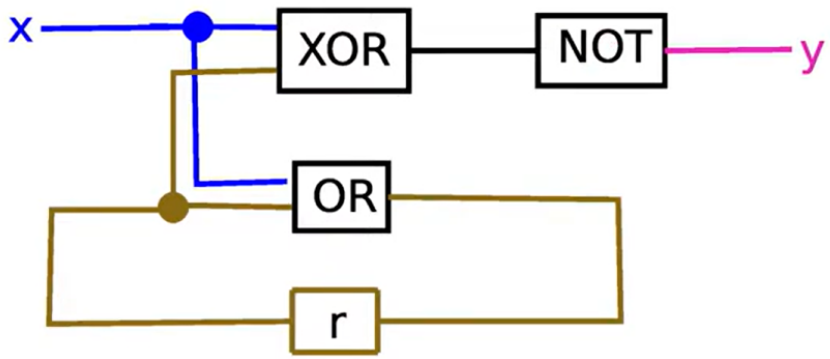


- output function: $\lambda_y = \neg(x \oplus r)$
- transition function: $\delta_r = x \vee r$

transition system

initial register evaluation: $r = 0$

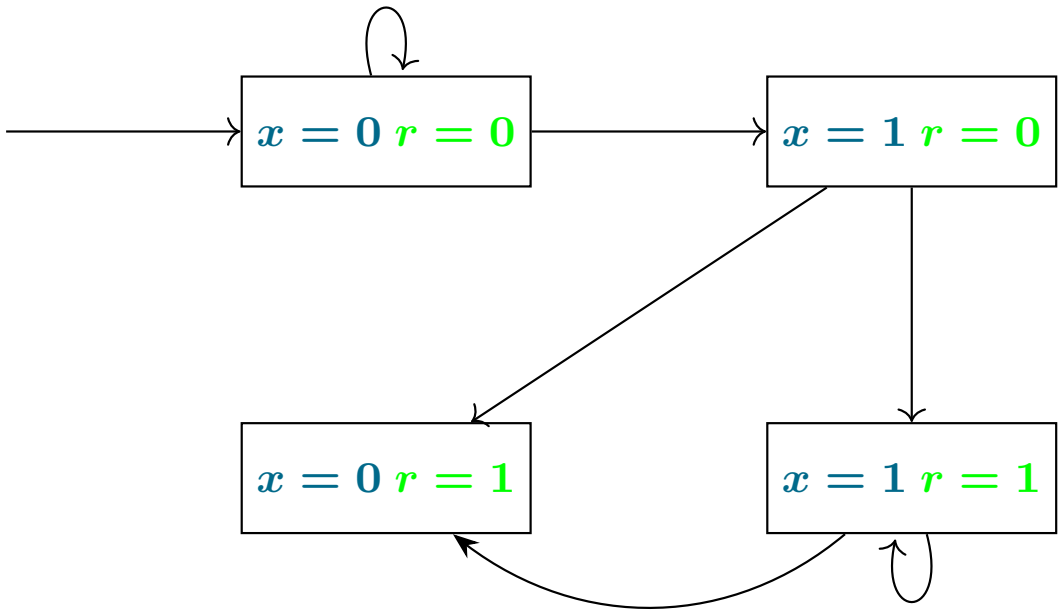


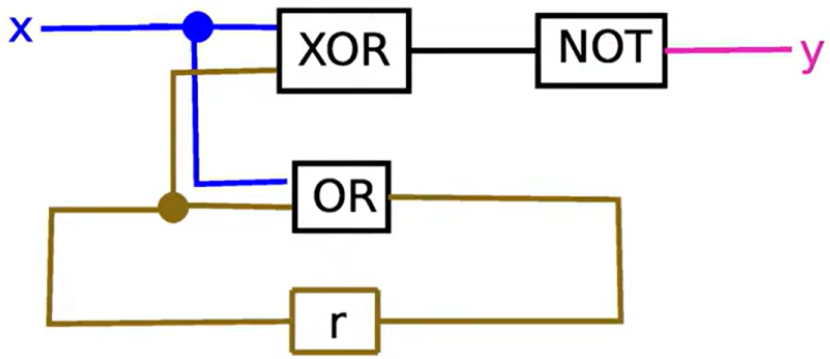


- output function: $\lambda_y = \neg(x \oplus r)$
- transition function: $\delta_r = x \vee r$

transition system

initial register evaluation: $r = 0$

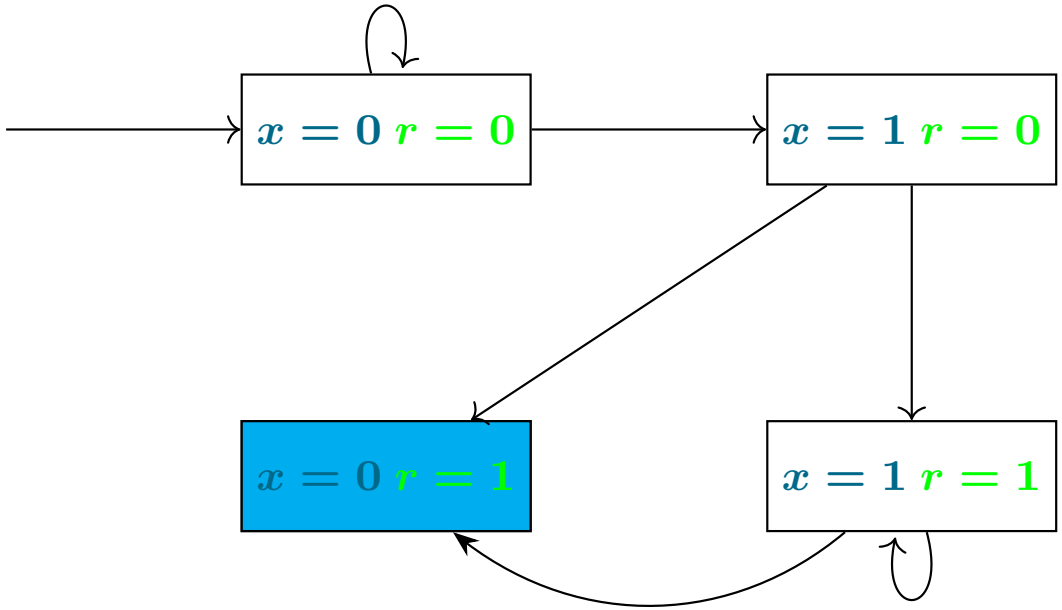


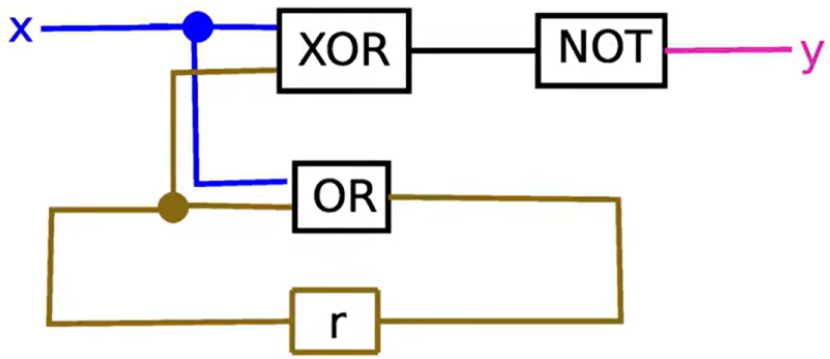


- output function: $\lambda_y = \neg(x \oplus r)$
- transition function: $\delta_r = x \vee r$

transition system

initial register evaluation: $r = 0$

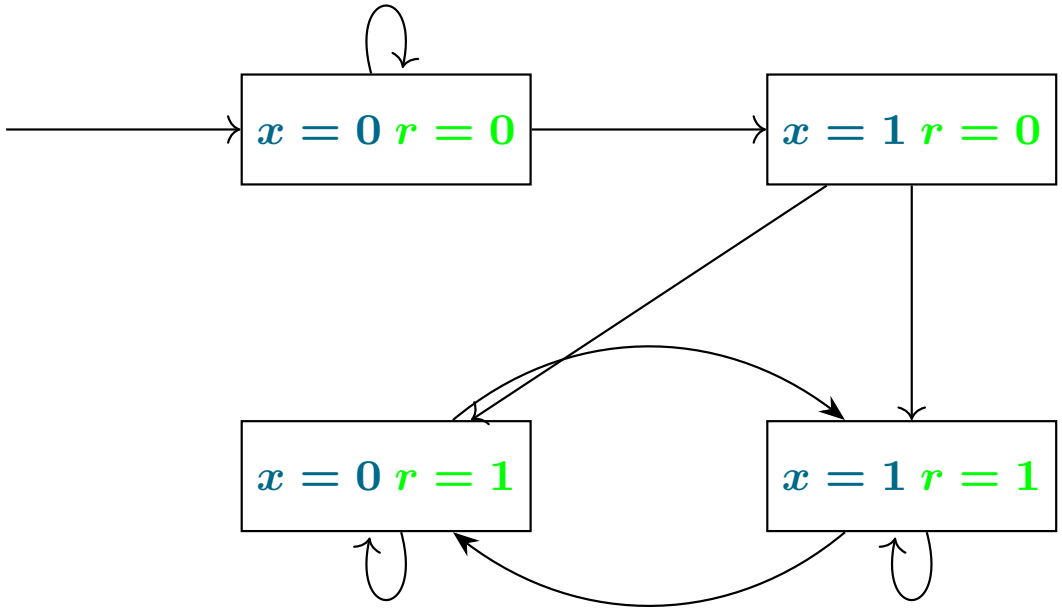


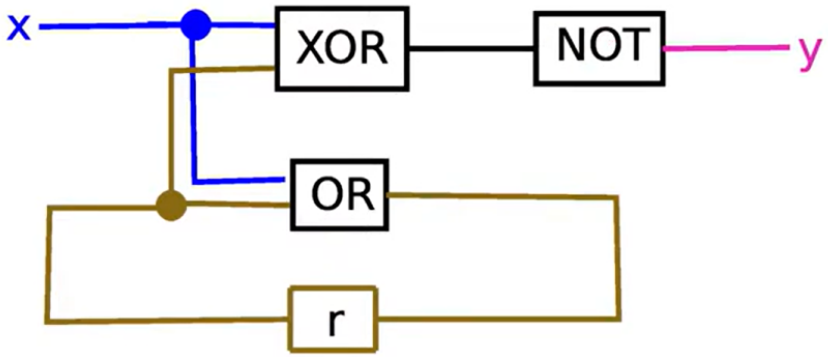


- output function: $\lambda_y = \neg(x \oplus r)$
- transition function: $\delta_r = x \vee r$

transition system

initial register evaluation: $r = 0$

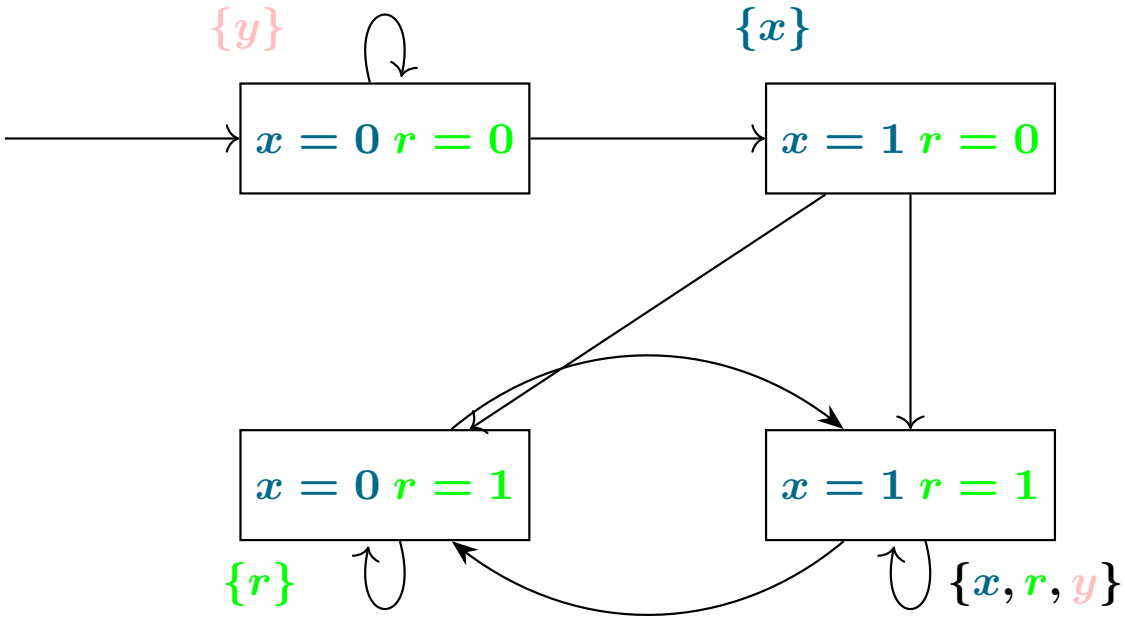


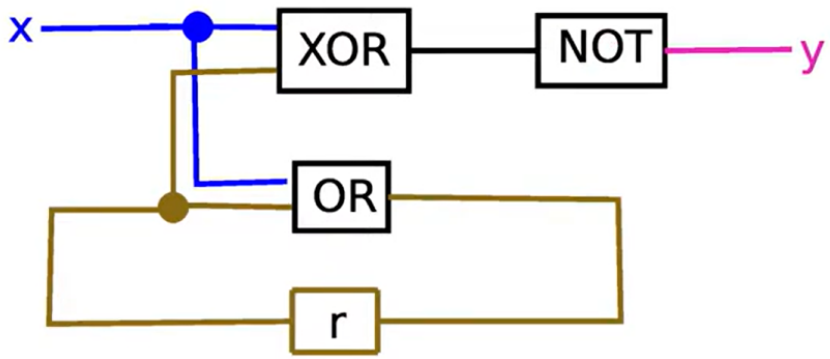


- output function: $\lambda_y = \neg(x \oplus r)$
- transition function: $\delta_r = x \vee r$

transition system

initial register evaluation: $r = 0$

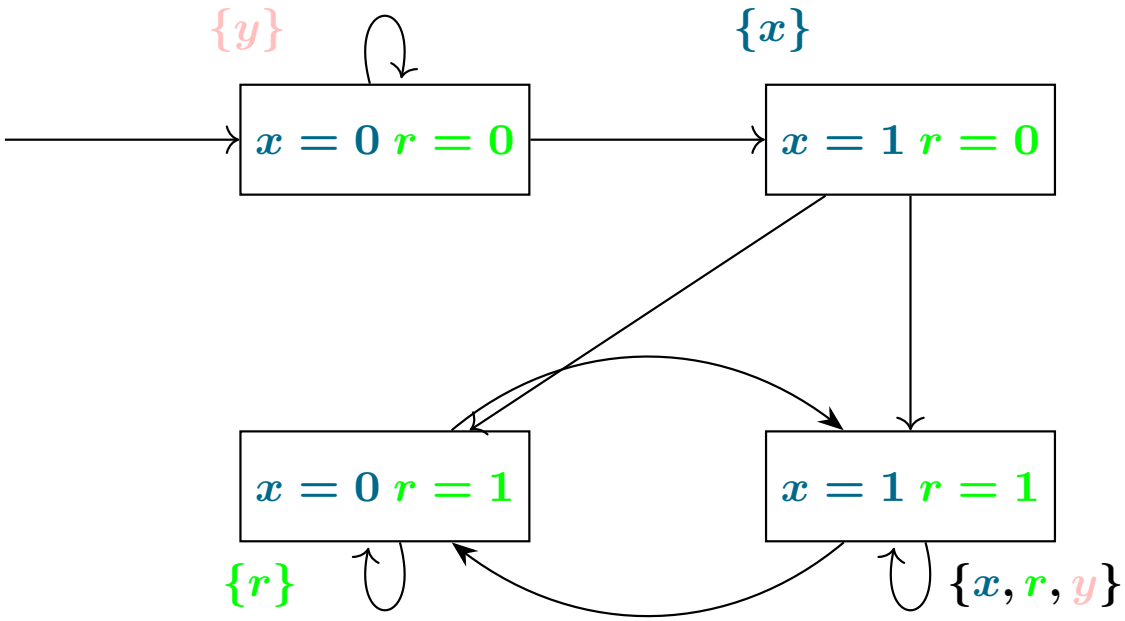




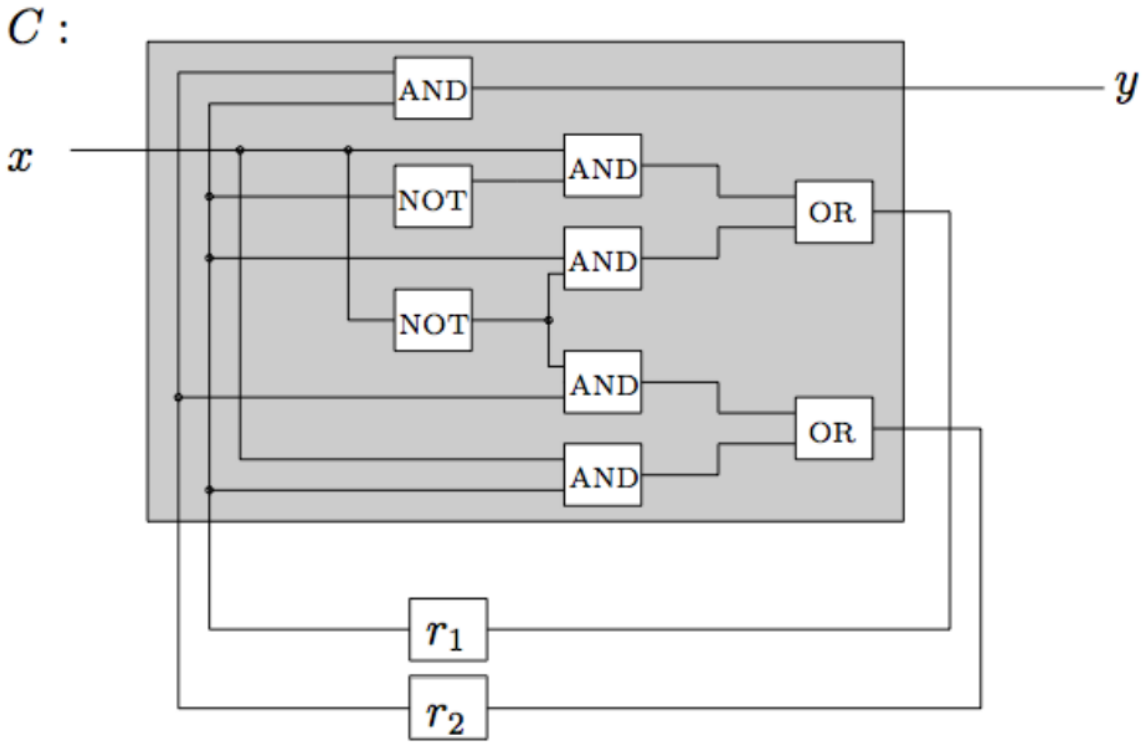
- output function: $\lambda_y = \neg(x \oplus r)$
- transition function: $\delta_r = x \vee r$

transition system

initial register evaluation: $r = 0$ $\quad AP = \{x, r, y\}$



✎ **Exercise 5.** Consider the following sequential hardware circuit:



Give the transition system representation of C.

⭐ Basic concepts: predecessors and successors ⭐

Let $\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$ be TS. For $s \in S$ and $\alpha \in Act$, we define the following sets.

Definition (Direct ($\alpha-$)successors of s:)

$$Post(s, \alpha) = \left\{ s' \in S \mid s \xrightarrow{\alpha} s' \right\}, \quad Post(s) = \bigcup_{a \in Act} Post(s, \alpha).$$

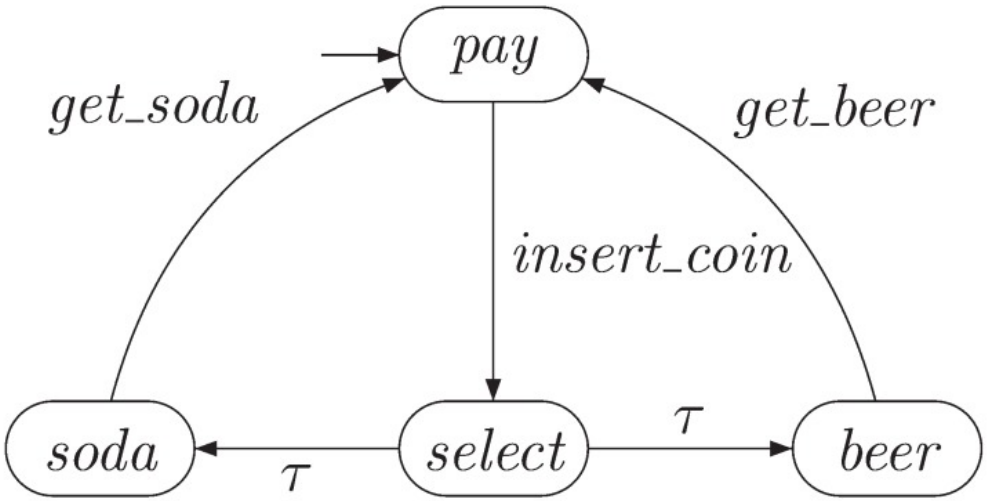
Definition (Direct ($\alpha-$)predecessors of s:)

$$Pre(s, \alpha) = \left\{ s' \in S \mid s' \xrightarrow{\alpha} s \right\}, \quad Pre(s) = \bigcup_{a \in Act} Pre(s, \alpha).$$

Definition (Terminal states)

A state $s \in S$ is called **terminal** or **blocking** iff $Post(s) = \emptyset$.

Remark: For reactive systems, those states should in general be avoided.



- $Post(select) = \{soda, beer\}$
- $Post(beer, \tau) = \emptyset$
- $Pre(select) = \{pay\}$
- $Pre(pay, get\_beer) = \{beer\}$

✎ **Exercise 6.** Consider the example of a simple state-transition diagram for a media player with three buttons:

- stop □, play ▷ and pause ||.
- The initial state of the player is stopped.
- In each state, only the buttons for the other states can be pressed (e.g. in play, only the stop and pause buttons can be pressed).
- Pressing the pause button when the player is stopped does not result in any change to the player

(a) Complete the following table

Current State	Event	Next State (Successor)
Stopped	Press play button	
Stopped	Press pause button	
Play	Press stop button	
Play	Press pause button	
Paused	Press play button	
Paused	Press stop button	

(b) Model the media player as transition system.

✎ **Exercise 7.** Create a state transition diagram for the following system. A burglar alarm has three states:

- Off
- Alarmed (the alarm is switched on)
- Sounding

The possible inputs are: turning the alarm on or off with a key, and sensors on (movement detected which sounds the alarm) or off. The state-transition table that describes the burglar alarm is a follows:

Current State	Key	Sensors	Next State
Off	Off	On	Off
Off	On	Off	Alarmed
Off	On	On	Sounding
Alarmed	Off	Off	Off
Alarmed	Off	On	Off
Alarmed	On	On	Sounding
Sounding	Off	Off	Off
Sounding	Off	On	Off
Sounding	On	Off	Alarmed

✯ Basic concepts: executions ✯

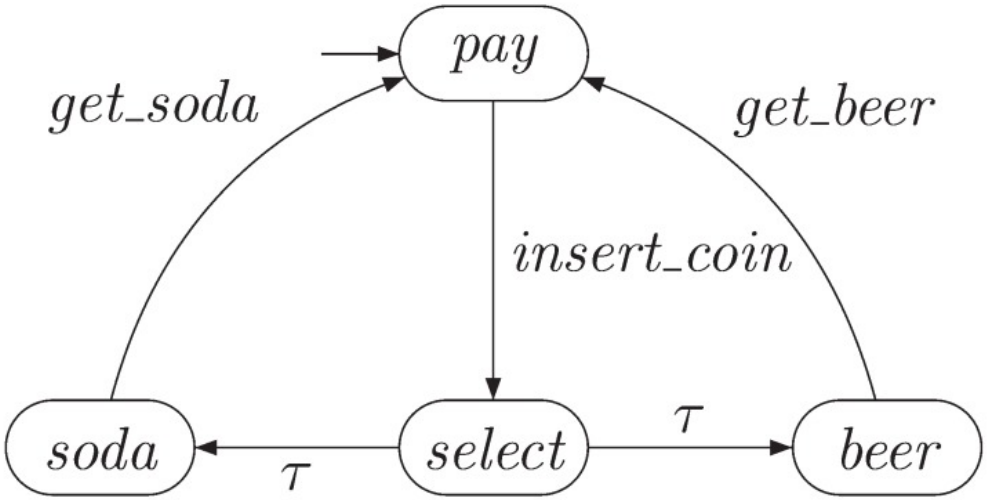
Definition (Executions)

- A finite execution fragment $\rho \in (S \times Act)^{\omega}$ of transition systems TS is a finite, alternating sequence of states and actions: $\rho = s_0 \alpha_1 s_1 \alpha_2 \cdots \alpha_n s_n$ such that $s_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} s_n$.
- An infinite execution fragment $\rho \in (S \times Act)^{\omega}$ of transition systems TS is an infinite, alternating sequence of states and actions: $\rho = s_0 \alpha_1 s_1 \alpha_2 \cdots$ such that $s_i \xrightarrow{\alpha_{i+1}} s_{i+1}$ for all $i \geq 0$.
- ρ is maximal if ρ is infinite or finite and ending in a terminal state i.e. Fragment cannot be prolonged.
- ρ is initial if starts in an initial state i.e. Fragment starting in $s_0 \in S_0$.
- An execution is an initial, maximal execution fragment.

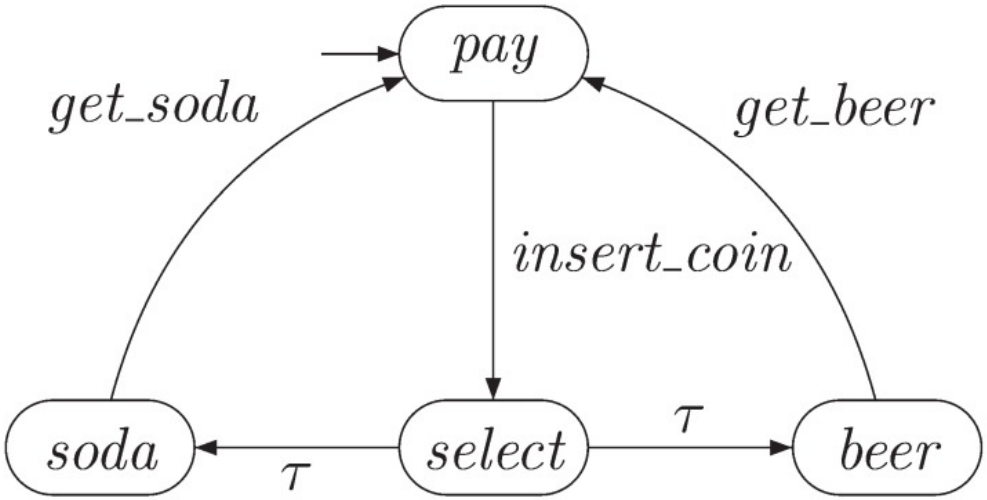
Definition (Reachable states:)

A state s is reachable in TS if s occurs in some execution of TS

$$Reach(\mathcal{T}) = \left\{ s \in S \mid \exists s_0 \in S_0 \wedge s_0 \xrightarrow{\alpha_1} \cdots \xrightarrow{\alpha_n} s_n = s \right\}$$

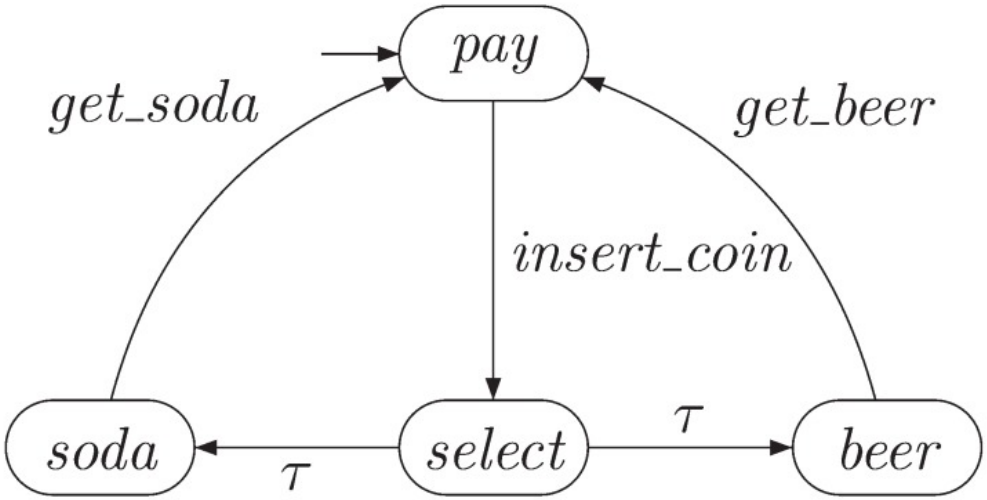


- $\rho_1 = pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} beer \xrightarrow{get\_beer} pay \xrightarrow{insert\_coin} \cdots$
- $\rho_2 = beer \xrightarrow{get\_beer} pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} beer \xrightarrow{get\_beer} \cdots$
- $\rho_3 = pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} soda \xrightarrow{get\_soda} pay$
- $Reach(\mathcal{T}) = S.$



- $\rho_1 = pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} beer \xrightarrow{get\_beer} pay \xrightarrow{insert\_coin} \cdots$

  ρ_1 is an execution.
- $\rho_2 = beer \xrightarrow{get\_beer} pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} beer \xrightarrow{get\_beer} \cdots$
- $\rho_3 = pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} soda \xrightarrow{get\_soda} pay$
- $Reach(\mathcal{T}) = S.$

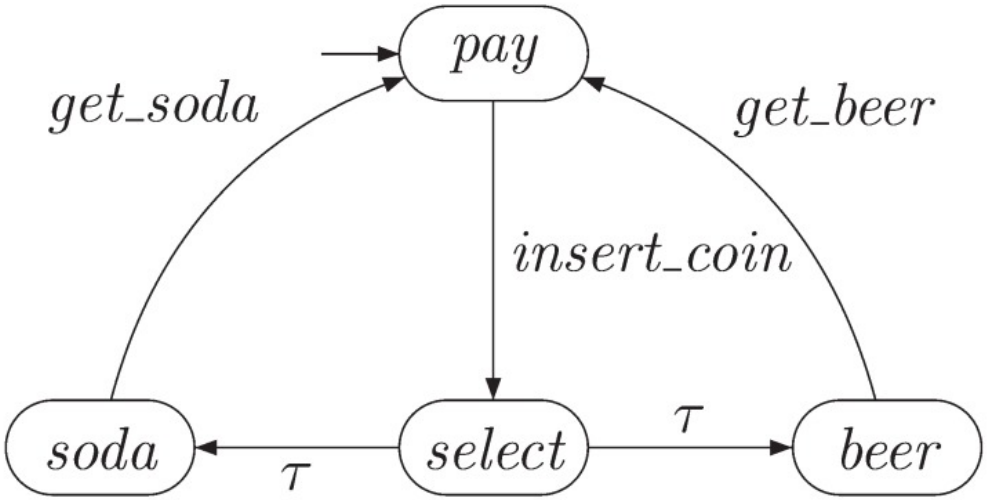


- $\rho_1 = pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} beer \xrightarrow{get\_beer} pay \xrightarrow{insert\_coin} \cdots$

  ρ_1 is an execution.
- $\rho_2 = beer \xrightarrow{get\_beer} pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} beer \xrightarrow{get\_beer} \cdots$

  ρ_2 is not an execution (maximal but not initial).
- $\rho_3 = pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} soda \xrightarrow{get\_soda} pay$

- $Reach(\mathcal{T}) = S$.



- $\rho_1 = pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} beer \xrightarrow{get\_beer} pay \xrightarrow{insert\_coin} \cdots$

  ρ_1 is an execution.
- $\rho_2 = beer \xrightarrow{get\_beer} pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} beer \xrightarrow{get\_beer} \cdots$

  ρ_2 is not an execution (maximal but not initial).
- $\rho_3 = pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} soda \xrightarrow{get\_soda} pay$

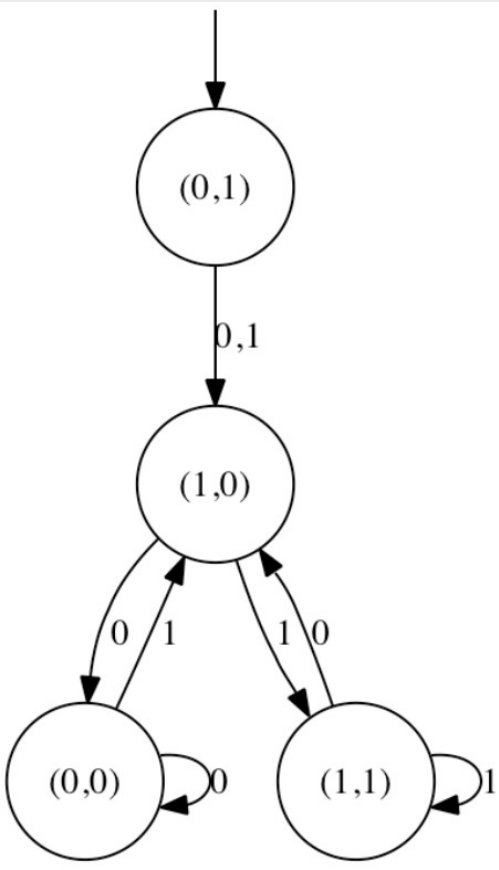
  ρ_3 is not an execution (initial but not maximal).
- $Reach(\mathcal{T}) = S$.

Definition

Let $\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$ be TS

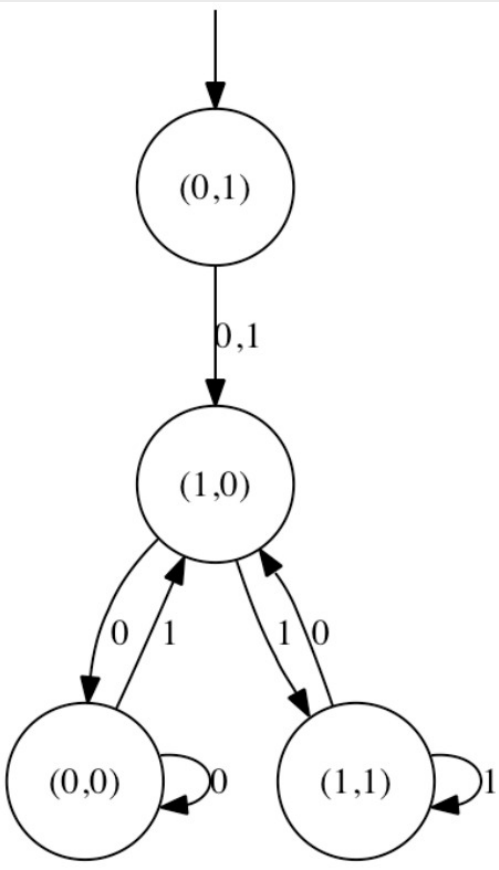
- A sequence of states, either finite or infinite , is a **path fragment** if $\pi = s_0 s_1 s_2 \cdots s_n$, $s_{i+1} \in Post(s_i)$, $\forall i \geq 0$.
- A **path** is a path fragment s.t. $s_0 \in S_0$ and it is
  - either finite with terminal s_n
  - or infinite.
- Denote the set of paths in TS by $Path(TS)$.

Example



- $(0,1)\ (1,0)\ (1,1)\ (1,1)\ \cdots$
- $(1,0)\ (0,0)\ (0,0)\ (1,0)\ \cdots$
- $(0,1)\ (1,0)\ (1,1)$.

Example



A path

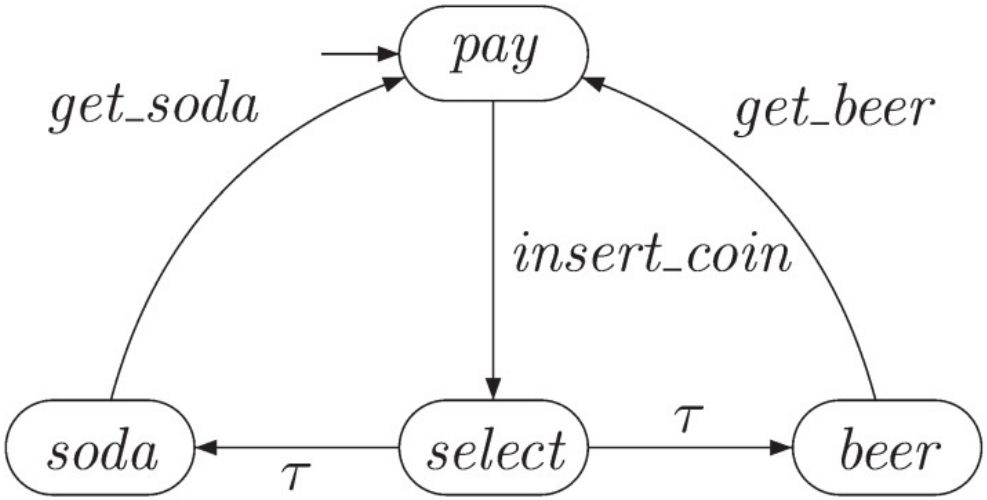
- $(0,1)\ (1,0)\ (1,1)\ (1,1)\ \cdots$

Not a path

- $(1,0)\ (0,0)\ (0,0)\ (1,0)\ \cdots$

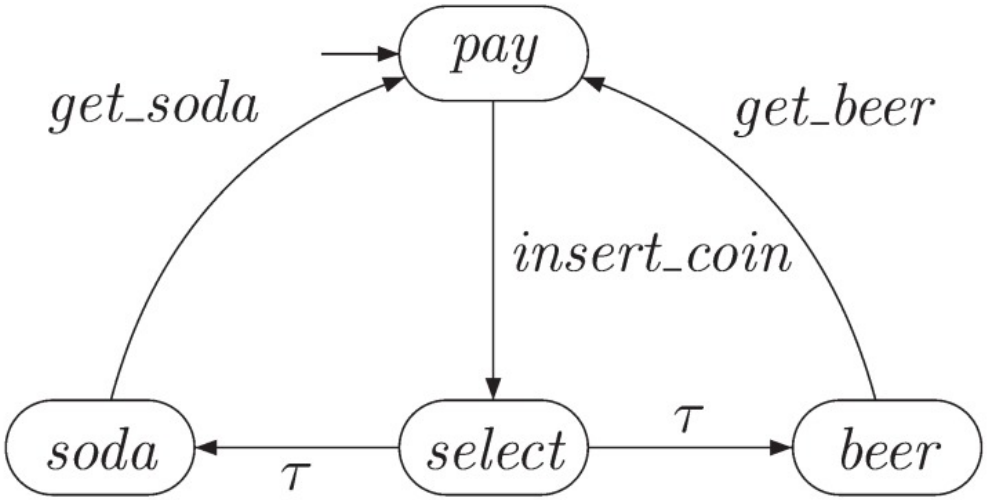
Not a path

- $(0,1)\ (1,0)\ (1,1).$



- Execusion:

- Path :



- Execusion:

$$\rho_1 = pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} beer \xrightarrow{get\_beer} pay \xrightarrow{insert\_coin} \cdots$$

- Path :

$$\pi_1 = pay\ select\ beer\ pay\ \cdots$$

Definition

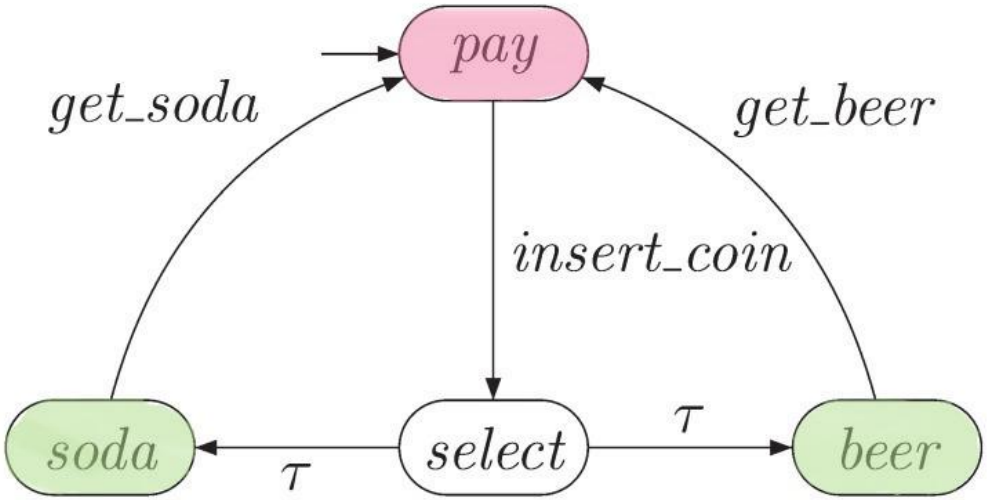
Let $\mathcal{T} = (S, Act, \longrightarrow, S_0, AP, L)$ be TS

- The **trace** of an infinite path fragment $\pi = s_0 s_1 s_2 \cdots$ is defined by

$$trace(\pi) = L(s_0)L(s_1)L(s_2)\cdots$$

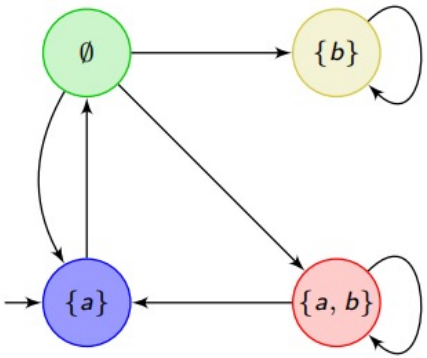
- The set, $Traces(TS)$, of traces of TS is defined by

$$Traces(TS) = \{trace(\pi) : \pi \in Paths(TS)\}$$



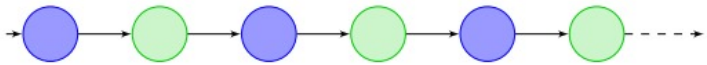
- Execusion: $\rho_1 = pay \xrightarrow{insert\_coin} select \xrightarrow{\tau} beer \xrightarrow{get\_beer} pay \xrightarrow{insert\_coin} \cdots$
- Path : $\pi_1 = pay\ select\ beer\ pay\ \cdots$
- Trace :
  - $AP = \{paid, drink\}$,
  - $L(pay) = paid$, $L(select) = \emptyset$ and $L(soda) = L(beer) = drink$

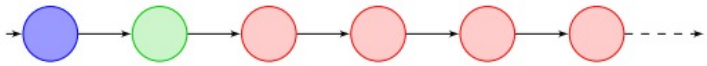
$trace(\pi_1) = L(pay)\ L(select)\ L(beer)\ L(pay)\ \cdots = paid\ \emptyset\ drink\ paid$

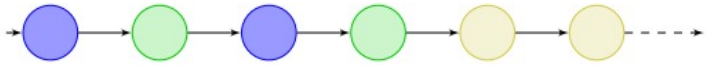


- Notice the added self-loop on 

- Paths:

$\pi_1 =$ 

$\pi_2 =$ 

$\pi_3 =$ 

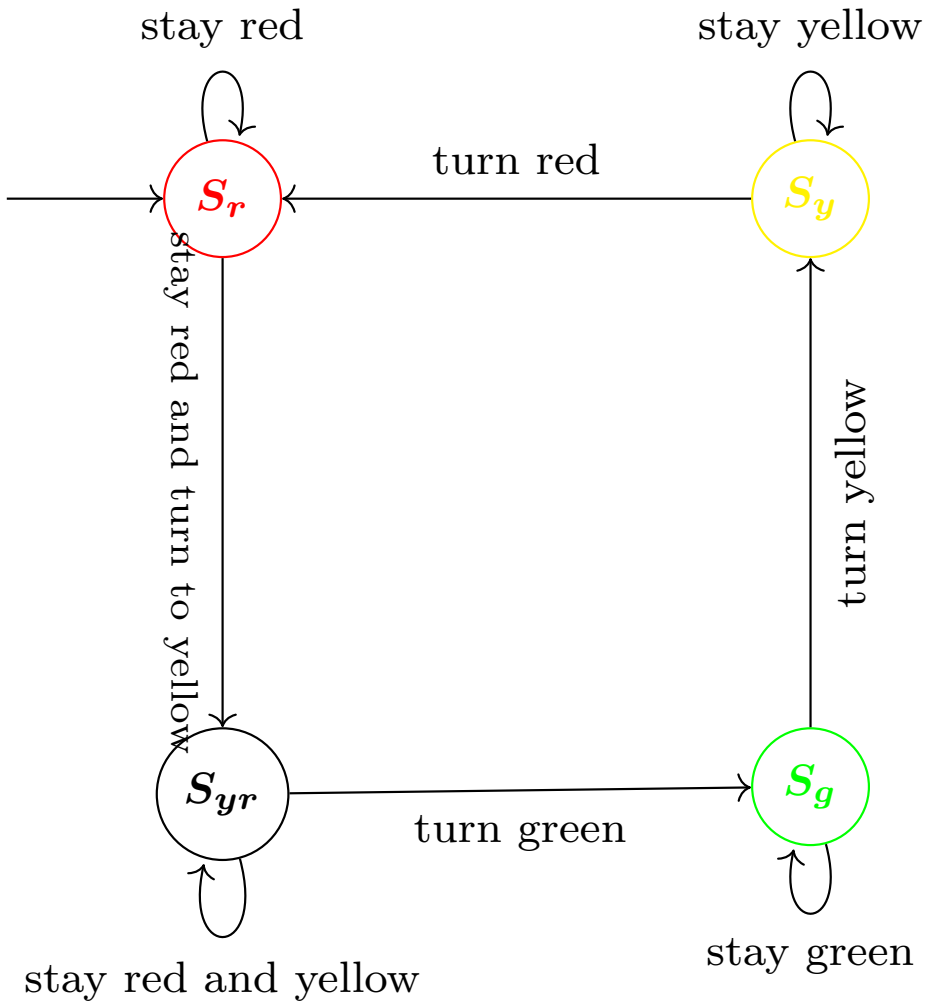
- Corresponding traces:

$$trace(\pi_1) = \{a\}\emptyset\{a\}\emptyset\{a\}\emptyset\ldots = (\{a\}\emptyset)^\omega$$
$$trace(\pi_2) = \{a\}\emptyset\{a,b\}\{a,b\}\{a,b\}\{a,b\}\ldots = \{a\}\emptyset\{a,b\}^\omega$$
$$trace(\pi_3) = \{a\}\emptyset\{a\}\emptyset\{b\}\{b\}\ldots = \{a\}\emptyset\{a\}\emptyset\{b\}^\omega$$

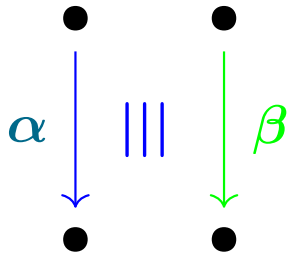
Traces are (infinite) words on alphabet 2^{AP}.

✎ Exercise 8.

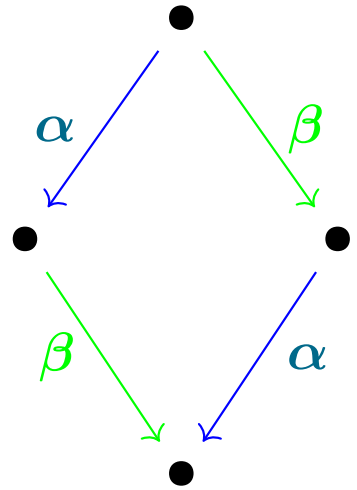
Find three differents execusions, paths and traces of the following digraph of traffic light:



$$effect(\alpha|||\beta) \quad = \quad effect(\alpha;\beta+\beta;\alpha)$$



$\Longleftrightarrow$

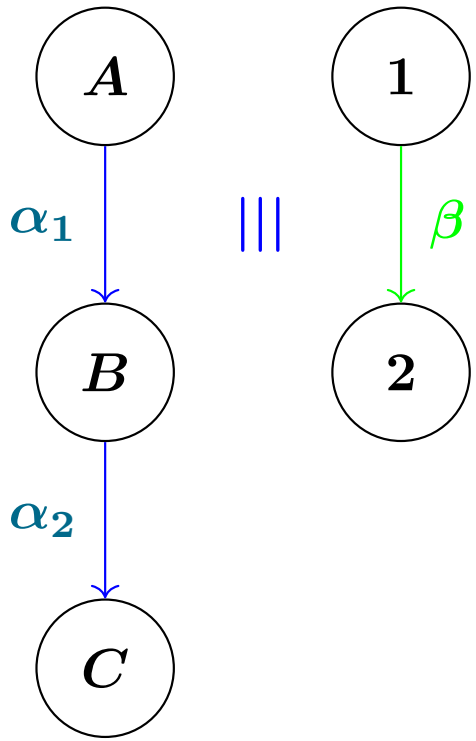


parallel execution of α and β on two processors

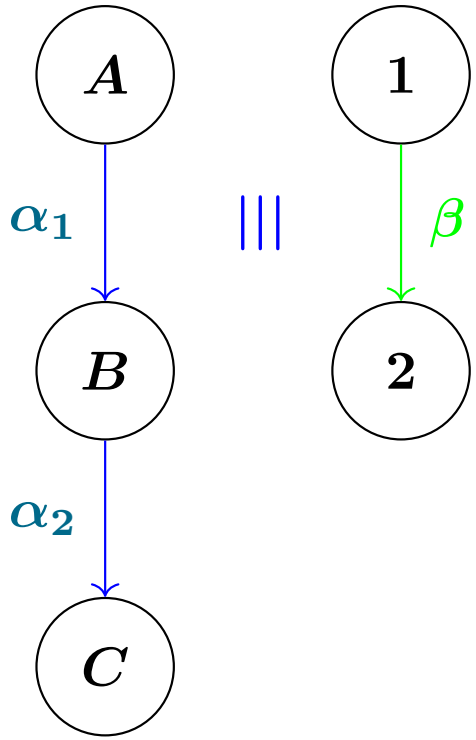
$\approx$

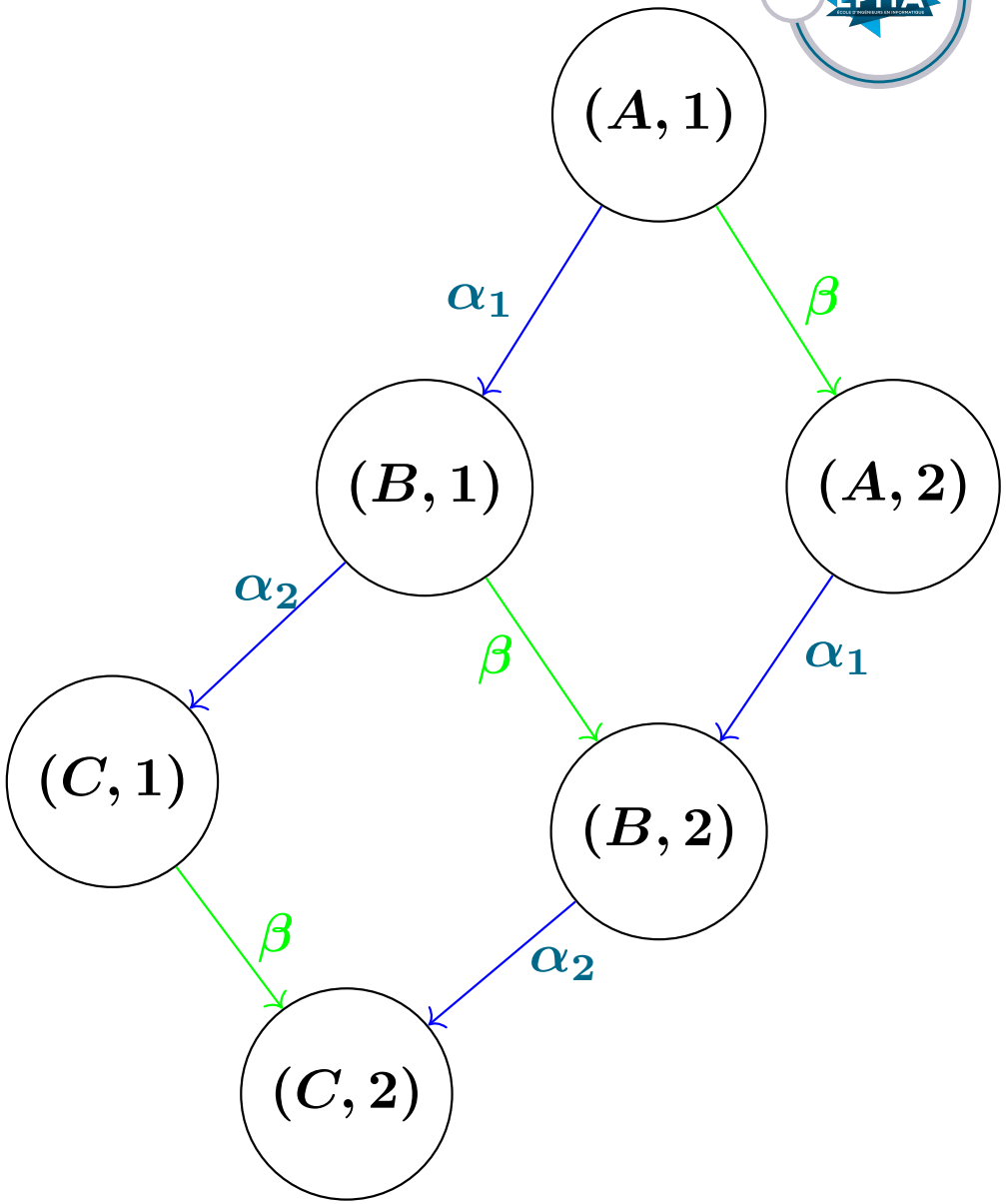
serial execution on a single processor in arbitrary order

Interleaving operator ||| for TS



Interleaving operator ||| for TS





- Let $\mathcal{T}_1 = (S_1, Act_1, \longrightarrow_1, S_{0,1}, AP_1, L_1)$ and $\mathcal{T}_2 = (S_2, Act_2, \longrightarrow_2, S_{0,2}, AP_2, L_2)$ be two TS.

- Let $\mathcal{T}_1 = (S_1, Act_1, \longrightarrow_1, S_{0,1}, AP_1, L_1)$ and $\mathcal{T}_2 = (S_2, Act_2, \longrightarrow_2, S_{0,2}, AP_2, L_2)$ be two TS.
- The transition system $\mathcal{T}_1 ||| \mathcal{T}_2$ is defined by :

$$\mathcal{T}_1 ||| \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \longrightarrow, S_{0,1} \times S_{0,2}, AP_1 \cup AP_2, L)$$

- Let $\mathcal{T}_1 = (S_1, Act_1, \longrightarrow_1, S_{0,1}, AP_1, L_1)$ and $\mathcal{T}_2 = (S_2, Act_2, \longrightarrow_2, S_{0,2}, AP_2, L_2)$ be two TS.
- The transition system $\mathcal{T}_1 ||| \mathcal{T}_2$ is defined by :

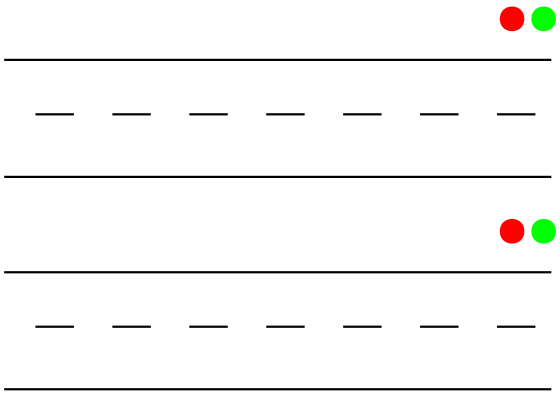
$$\mathcal{T}_1 ||| \mathcal{T}_2 = (S_1 \times S_2, Act_1 \cup Act_2, \longrightarrow, S_{0,1} \times S_{0,2}, AP_1 \cup AP_2, L)$$

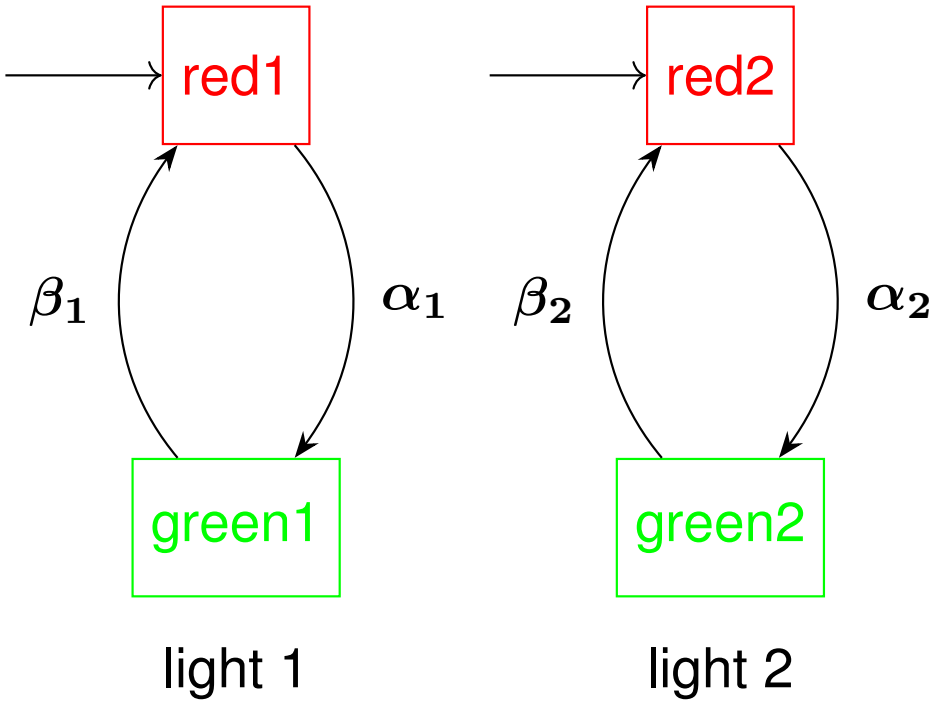
where the transition relation $\longrightarrow$ is given by:

$$\frac{s_1 \xrightarrow{\alpha}_1 s_1'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle} \quad \text{and} \quad \frac{s_2 \xrightarrow{\alpha}_2 s_2'}{\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle}$$

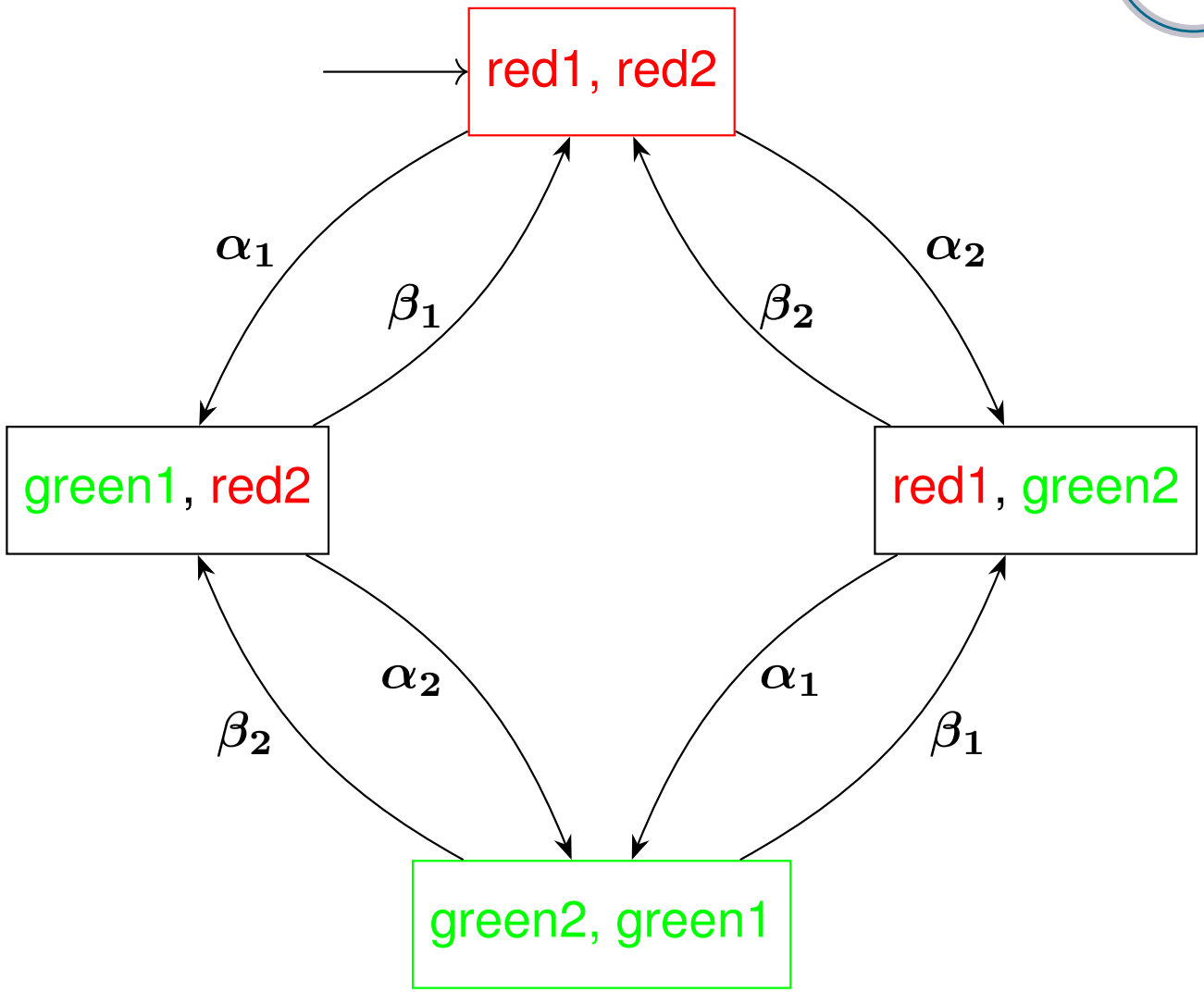
1. If $s_1 \xrightarrow{\alpha}_1 s_1'$ then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1', s_2 \rangle$
2. If $s_2 \xrightarrow{\alpha}_2 s_2'$ then $\langle s_1, s_2 \rangle \xrightarrow{\alpha} \langle s_1, s_2' \rangle$

Interleaving operator ||| for TS

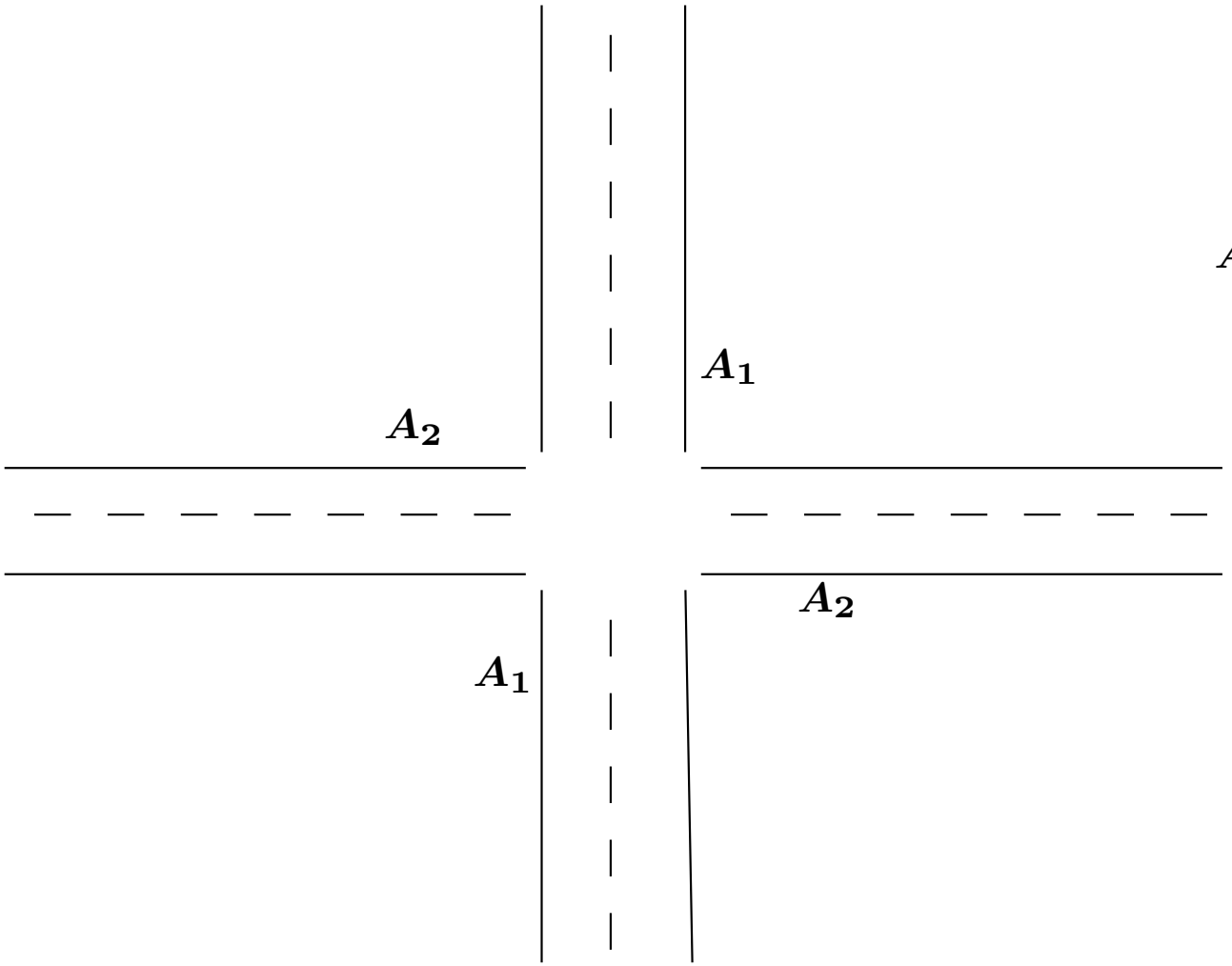


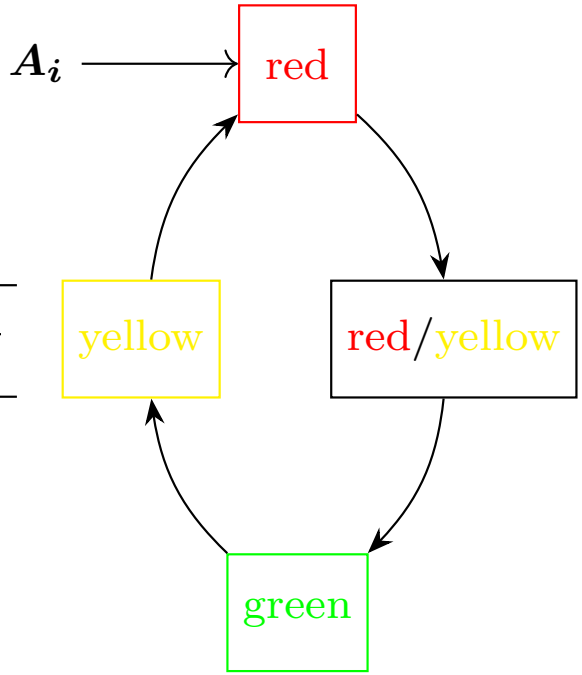


light 1 ||| light 2



✎ **Exercise 9.** Consider the following street junction with the specification of a traffic light as outlined on the right.





1. Choose appropriate actions and label the transitions of the traffic light transition system accordingly.
2. Outline the transition system $A_1||A_2$. Give the transition system representation of a (reasonable) controller that switches the green signal lamps in the following order: A_1, A_2, A_1, A_2, $\cdots$ (Hint: Choose an appropriate communication mechanism.)