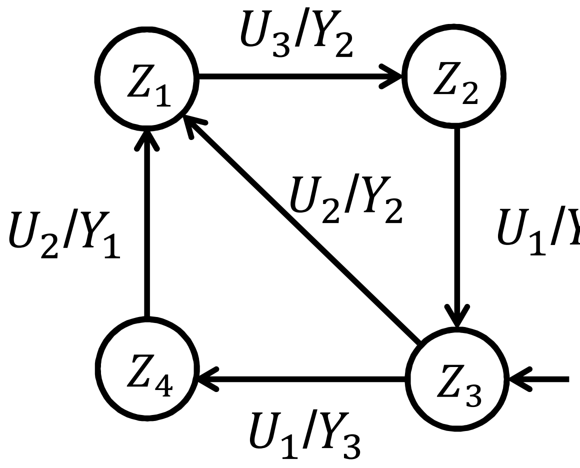
Logic Control

Prof. Dr. Ping Zhang

WS 2017/2018

Overview of the course

- Introduction
- Modeling of logic control systems
  - Boolean algebra
  - Finite state automata
  - Petri nets, SIPN
- Analysis of logic control systems
- Design of logic control systems
- Verification and validation
- Online diagnosis of logic control systems
- Implementation of logic control systems
  - PLC
  - Programming languages (IEC 61131-3)
  - Automatic code generation
- Distributed control (optional)



Finite State Automata (FSA)

- The automata describe state transitions.
- FSA are suitable for the description of **dynamic systems with discrete signals.**
- Key elements in a finite state automaton:

$$\mathrm{A} = (Z, U, Y, f, g, z_0)$$

$Z = \{Z_1, Z_2, \cdots, Z_{n_z}\}$: the set of states

$U = \{U_1, U_2, \cdots, U_{n_u}\}$: the set of inputs

$Y = \{Y_1, Y_2, \cdots, Y_{n_y}\}$: the set of outputs

f: the state transition function

g: the output function

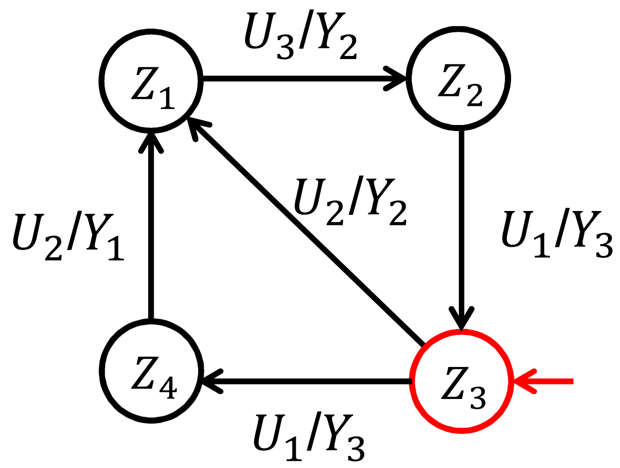
z_0: the initial state

Finite State Automata (FSA)

- An automaton can be described in different ways:
  - State transition diagram (in German: Automatengraf)
  - Update table (in German: Automatentabelle)
  - State and output equations (in German: Automatengleichungen)

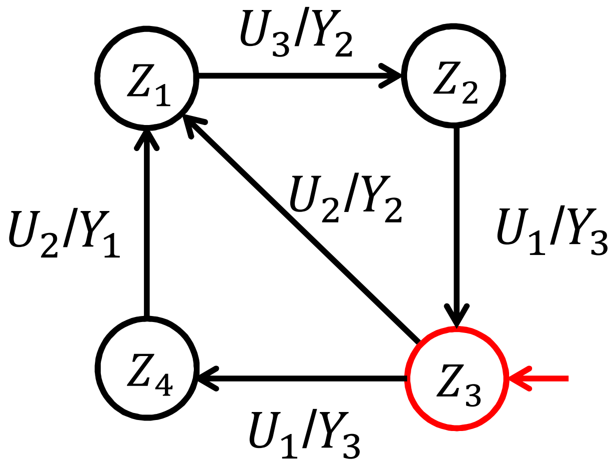
Finite State Automata (FSA)

- **State transition diagram**
  - **Directed** graph
  - The nodes denote the **states**.
  - The branches denote the **state transitions**.
  - The branches are labeled by the corresponding **inputs** and **outputs**.
  - The **initial state** is marked by an arrow.



Finite State Automata (FSA)

- **Update table**



Next state	Output	Current state	Input
Z_4	Y_3	Z_3	U_1
Z_1	Y_2	Z_3	U_2
Z_1	Y_1	Z_4	U_2
Z_2	Y_2	Z_1	U_3
Z_3	Y_3	Z_2	U_1

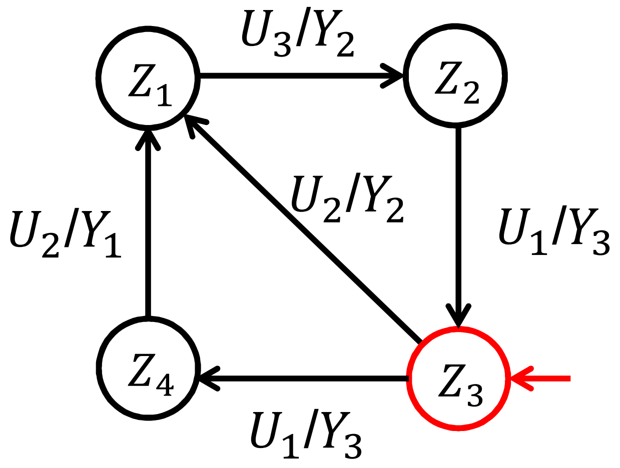
Each row denotes a state transition.

Current state	(Next state, Output)		
	U_1	U_2	U_3
Z_1	$-$	$-$	(Z_2, Y_2)
Z_2	(Z_3, Y_3)	$-$	$-$
Z_3	(Z_4, Y_3)	(Z_1, Y_2)	$-$
Z_4	$-$	(Z_1, Y_1)	$-$

Each cell denotes a state transition.

State and output equations

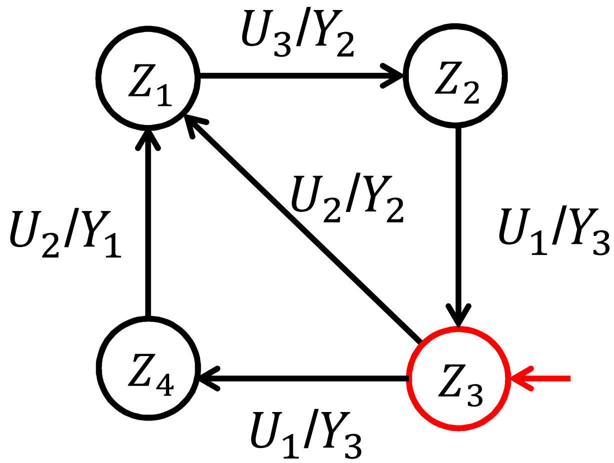
$$z(k+1) = f(z(k), u(k))$$
$$y(k) = g(z(k), u(k))$$
$$z(0) = Z_3$$



$$z(k+1) = f\big(z(k), u(k)\big) = \begin{cases} Z_1, & \text{if } [(z(k) = Z_3) \vee (z(k) = Z_4)] \wedge (u(k) = U_2) \\ Z_2, & \text{if } (z(k) = Z_1) \wedge (u(k) = U_3) \\ Z_3, & \text{if } (z(k) = Z_2) \wedge (u(k) = U_1) \\ Z_4, & \text{if } (z(k) = Z_3) \wedge (u(k) = U_1) \end{cases}$$

$$y(k) = g\big(z(k), u(k)\big) = \begin{cases} Y_1, & \text{if } (z(k) = Z_4) \wedge (u(k) = U_2) \\ Y_2, & \text{if } [(z(k) = Z_1) \wedge (u(k) = U_3)] \vee [(z(k) = Z_3) \wedge (u(k) = U_2)] \\ Y_3, & \text{if } [(z(k) = Z_2) \vee (z(k) = Z_3)] \wedge (u(k) = U_1) \end{cases}$$

Finite State Automata (FSA)



- **Input sequence**

  $\{U_1, U_2, U_3, U_1, U_2, U_3, U_1, U_1, U_2, \dots\}$

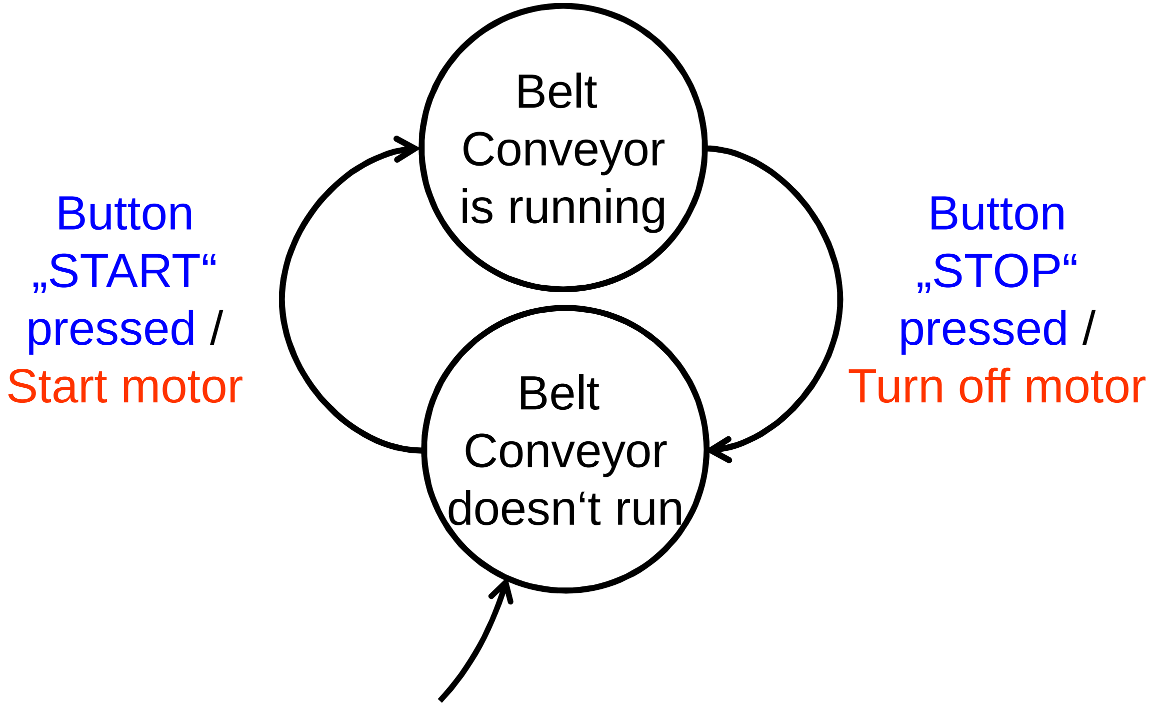
- **State trajectory**

  $\{Z_3, Z_4, Z_1, Z_2, Z_3, Z_1, Z_2, Z_3, Z_4, Z_1, \dots\}$

- **Output sequence**

  $\{Y_3, Y_1, Y_2, Y_3, Y_2, Y_2, Y_3, Y_3, \dots\}$

Example: Belt conveyor



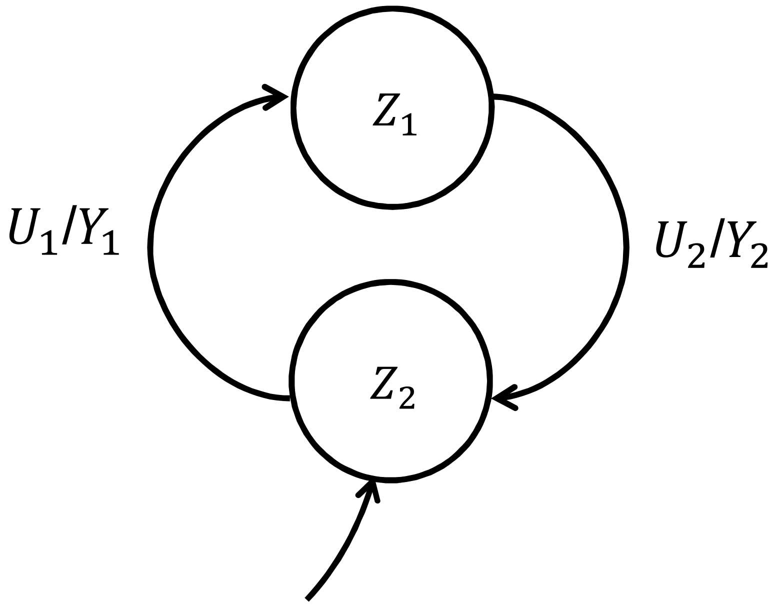
Finite State Automata (FSA)

States, inputs and outputs in the above automaton

State	Description
Z_1	The belt conveyor is running
Z_2	The belt conveyor doesn't run

Input	Description
U_1	Press button "START"
U_2	Press button "STOP"

Output	Description
Y_1	Start motor
Y_2	Turn off motor

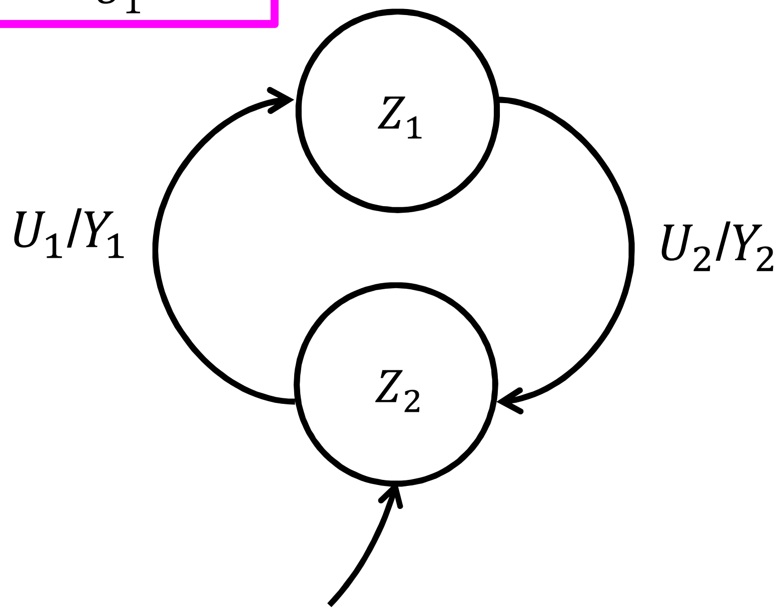


Finite State Automata (FSA)

- Update table

Next state	Output	Current state	Input
Z_2	Y_2	Z_1	U_2
Z_1	Y_1	Z_2	U_1

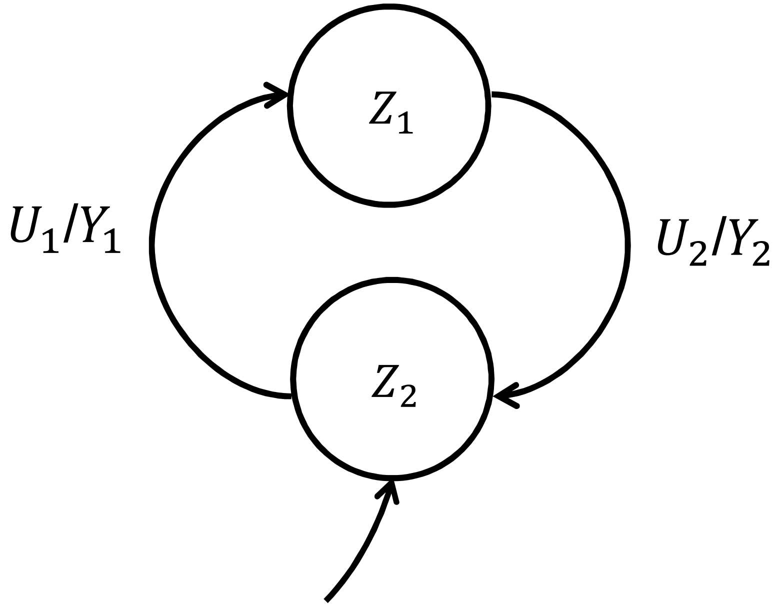
Current state	(Next state, Output)	
	U_1	U_2
Z_2	(Z_1, Y_1)	$-$
Z_1	$-$	(Z_2, Y_2)



Finite State Automata (FSA)

- State and output equations

$$z(k+1) = f(z(k), u(k))$$
$$y(k) = g(z(k), u(k))$$
$$z(0) = Z_2$$



$$z(k+1) = f\big(z(k), u(k)\big)$$
$$= \begin{cases} Z_1, & if\ (z(k) = Z_2) \wedge (u(k) = U_1) \\ Z_2, & if\ (z(k) = Z_1) \wedge (u(k) = U_2) \end{cases}$$

$$y(k) = g\big(z(k), u(k)\big)$$
$$= \begin{cases} Y_1, & if\ (z(k) = Z_2) \wedge (u(k) = U_1) \\ Y_2, & if\ (z(k) = Z_1) \wedge (u(k) = U_2) \end{cases}$$

Automata with/without input/output

Autonomous automaton: no input and output signals

Semi automaton: with inputs, but no outputs

Automaton with inputs and outputs: with both inputs and outputs

Moore automaton:

The current output is only influenced by the current state, but **not by the current input**.

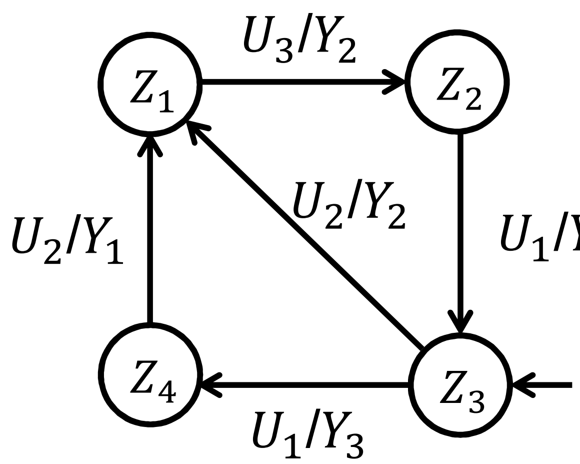
$$y(k) = g(z(k))$$

Mealy automaton:

The current output is influenced by both the current state and the current input.

$$y(k) = g(z(k), u(k))$$

Discussion: Is the following automaton a Mealy automaton or a Moore automaton?



Deterministic vs. nondeterministic Automata

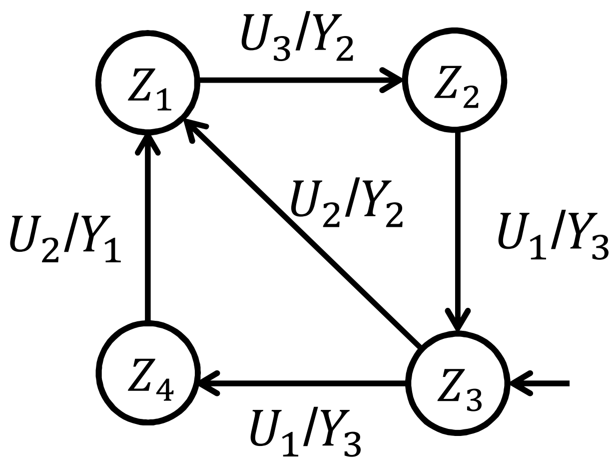
Deterministic Automaton:

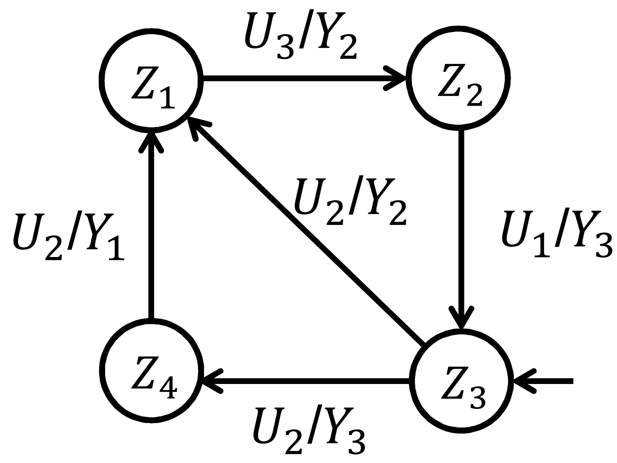
For a given current state and a given input, both the output and the next state are **uniquely** determined.

Nondeterministic Automaton:

For a given current state and a given input, the next state or the output is **not uniquely** determined.

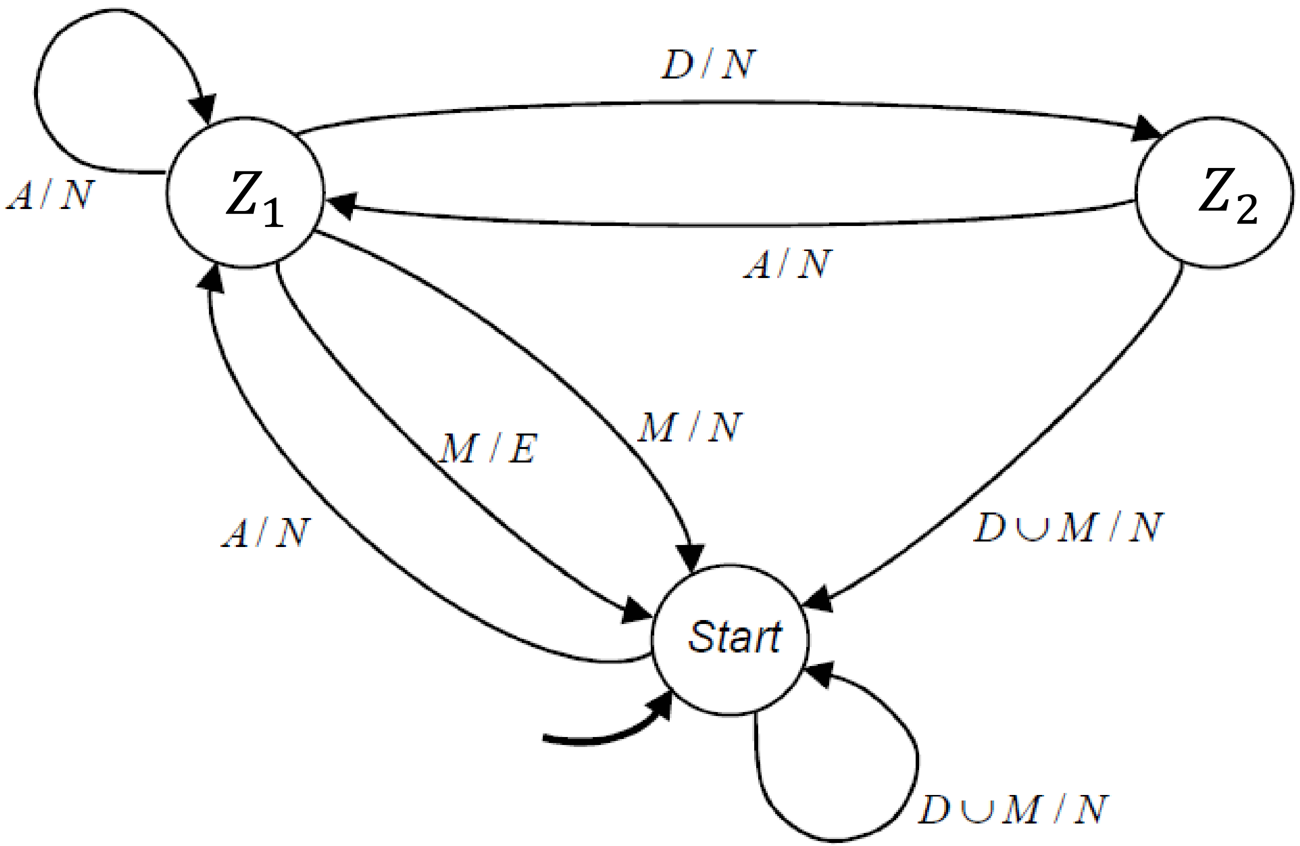
Discussion: Are the following automata a deterministic automaton or a nondeterministic automaton?





Deterministic vs. nondeterministic Automata

Discussion: Are the following automata a deterministic automaton or a nondeterministic automaton?

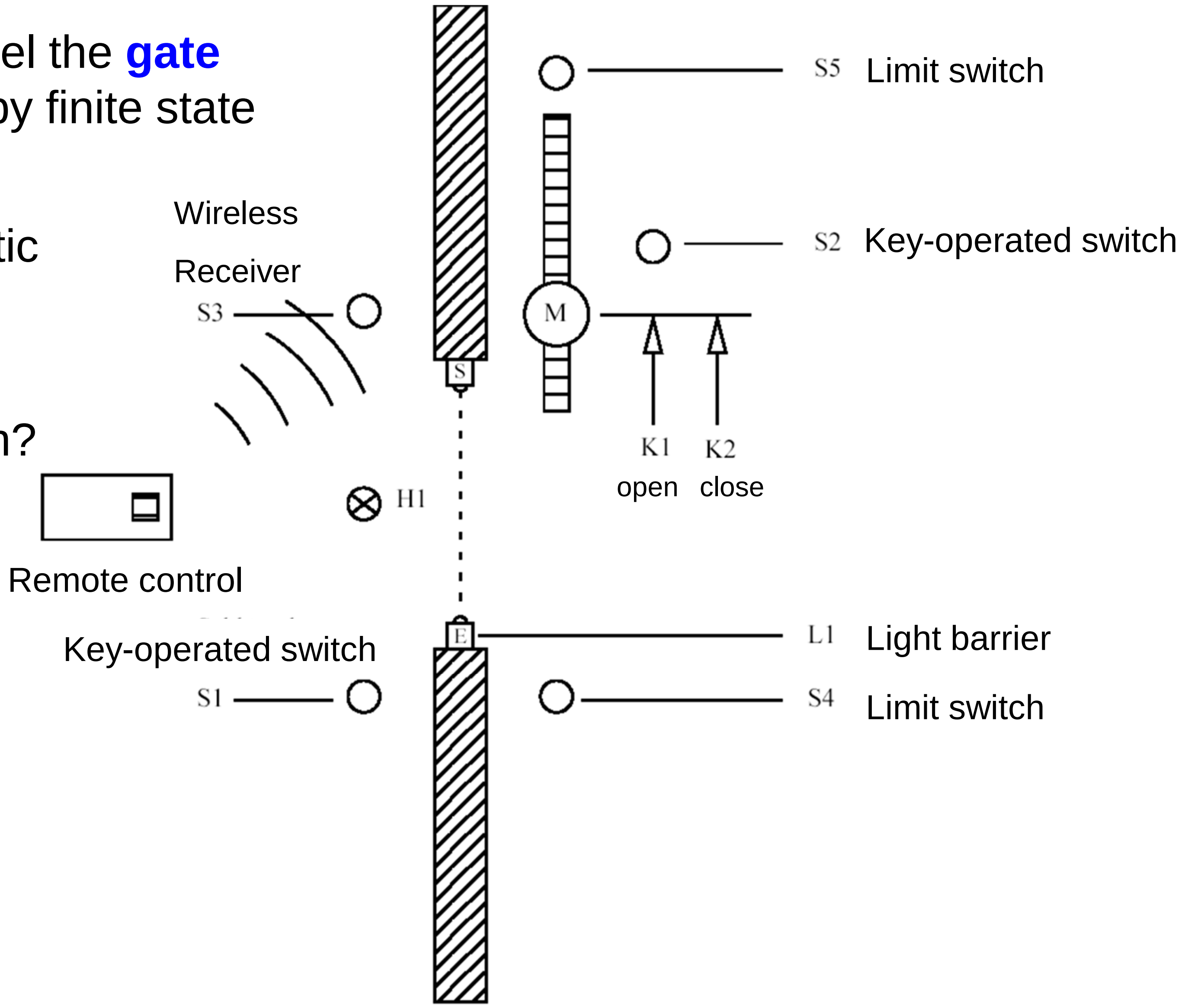


Modeling by FSA: Gate control system

Example 1: Model the **gate control** system by finite state automata.

Is it a deterministic automaton?

Is it a Moore or Mealy automaton?



Modeling by FSA: Gate control system

Specifications:

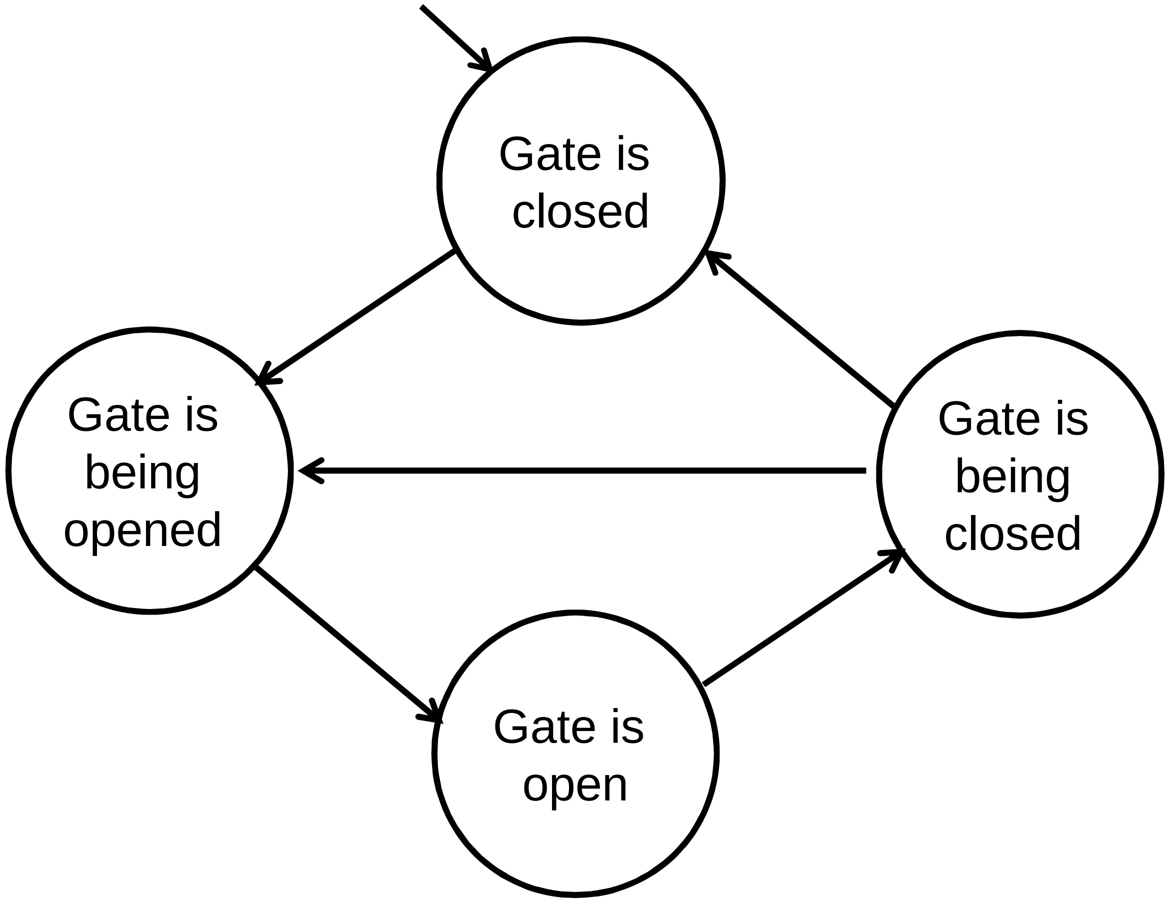
- The gate can be opened and closed by controlling the electric motor. The rotary direction of the electric motor (**clockwise rotation** or **anti-clockwise rotation**) is controlled by two contactors.
- The gate can be opened from outside or inside by the **key-operated switches**, or by a **remote control unit**.
- For safety reasons, a **light barrier** is mounted. If the light barrier senses an object, the gate should not be closed.
- **Two limit switches** notify the states of the gate (i.e. „gate is opened“ or „gate is closed“).
- The completely opened gate should be automatically closed after **20 seconds waiting time**.
- A **red flashing light** should notify the opening and closing of the rolling gate on both sides.

Modeling by FSA: Gate control system

Signals	I/O	Symbol	Logic assignment
Key-operated switch (outside)	I	S1	Operated S1=1
Key-operated switch (inside)	I	S2	Operated S2=1
Wireless receiver	I	S3	Code received S3=1
Limit switch (gate opened)	I	S4	Gate is opened S4=1
Limit switch (gate closed)	I	S5	Gate is closed S5=1
Light barrier	I	L1	Interrupted L1=0
Flash light	O	H1	Light on H1=1
Contactor (opening gate)	O	K1	Contactor activated K1=1
Contactor (closing gate)	O	K2	Contactor activated K2=1

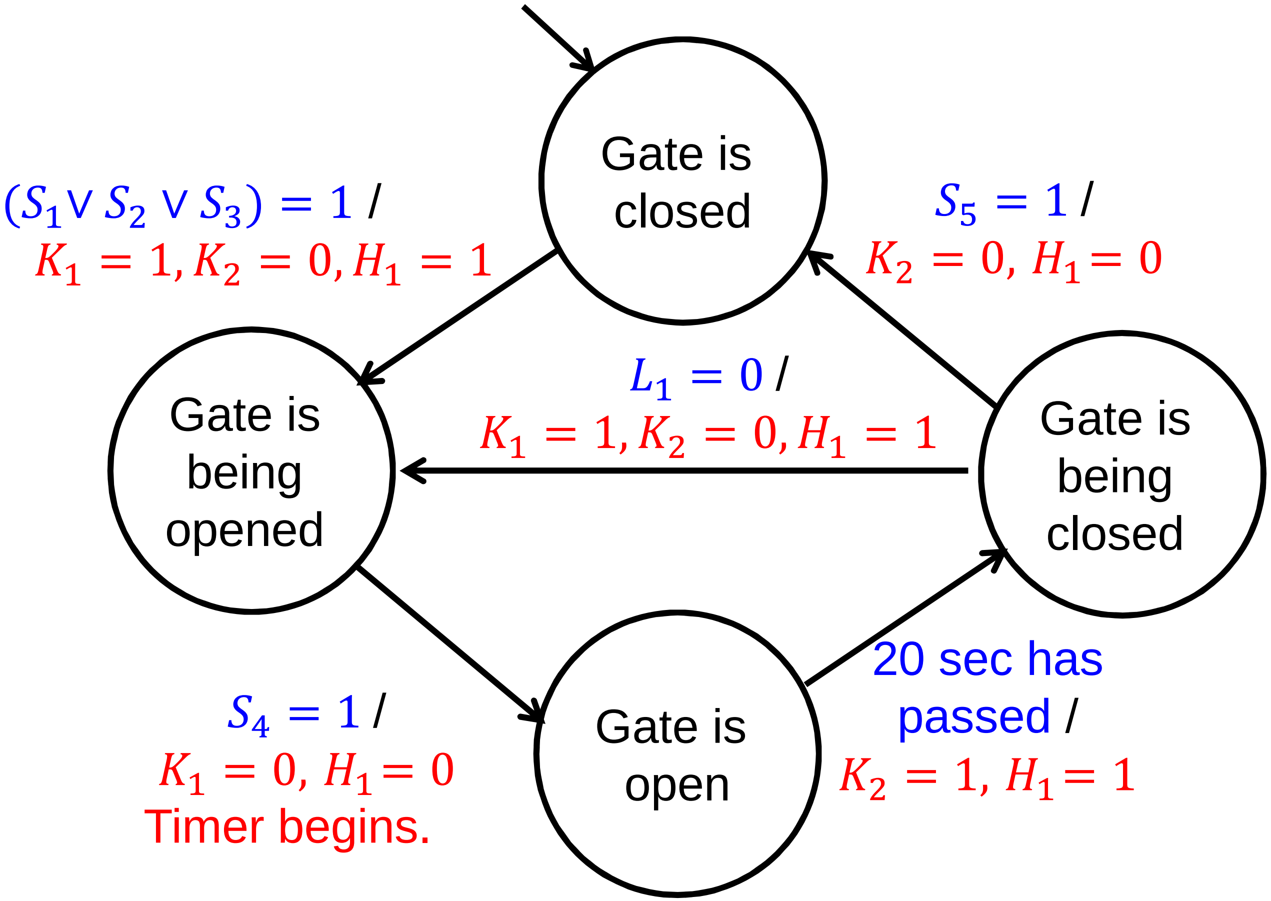
Modeling by FSA: Gate control system

State transition diagram of the gate control system



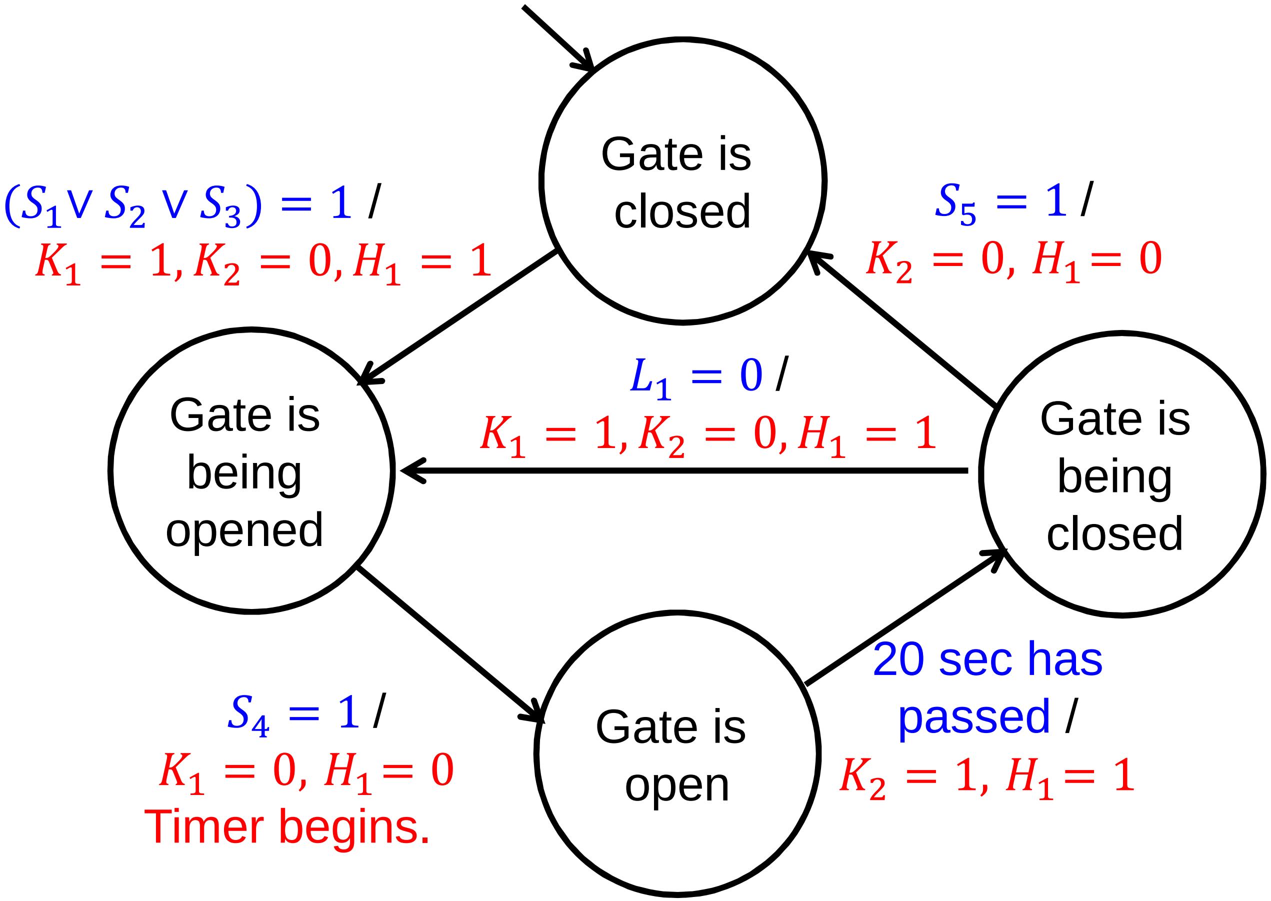
Modeling by FSA: Gate control system

State transition diagram of the gate control system



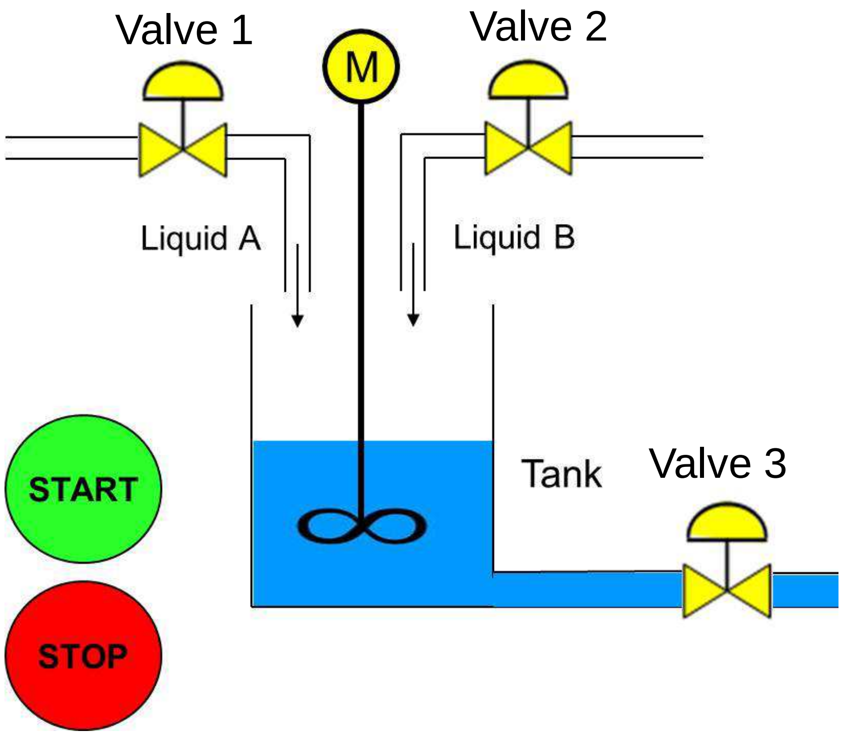
Modeling by FSA: Gate control system

State transition diagram of the gate control system



Deterministic Automaton, Mealy Automaton

Example 2: Mixing tank

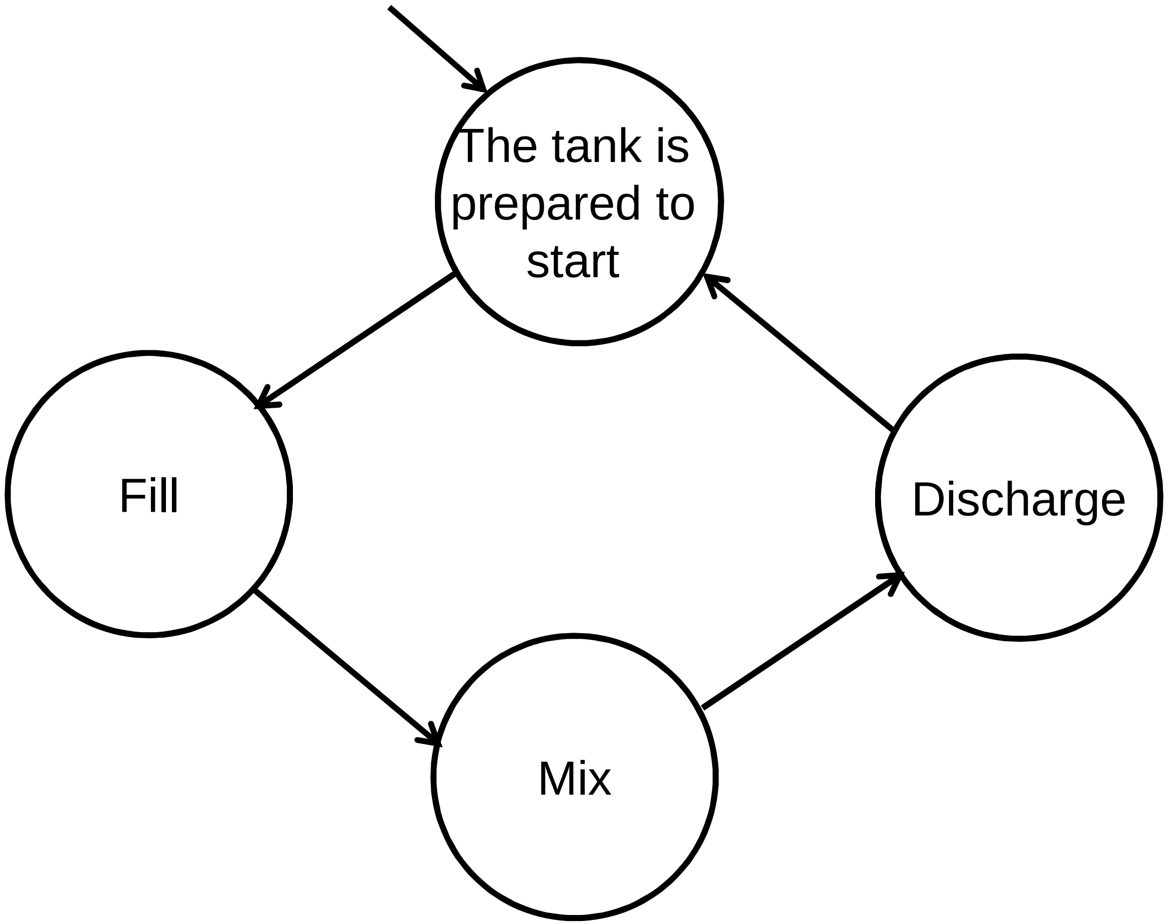


Specifications:

1. If the **button START** is pressed, **open Valve 1** and **Valve 2** to fill in, respectively, the liquid A and B.
2. When **the tank level reaches Level 2**, **close** both **Valve 1** and **Valve 2** and **start** the **motor** of the mixer.
3. After 10 minutes, **turn off** the **motor** of the mixer and **open Valve 3**.
4. When **the tank is empty**, **close Valve 3**.
5. If the **button STOP** is pressed at any time, stop the process immediately.

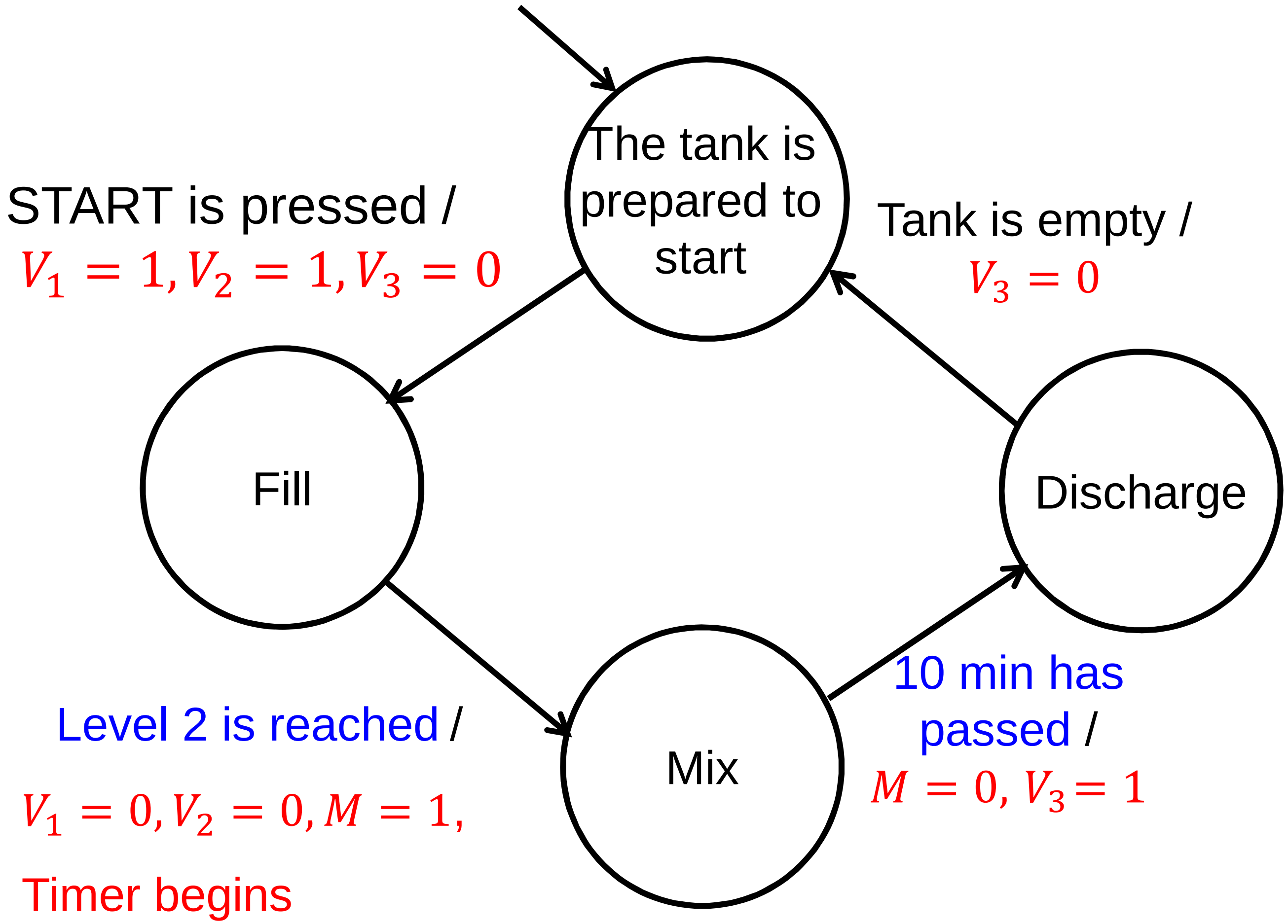
Modeling by FSA: Mixing tank

Model the mixed tank system by considering Specifications 1-4.



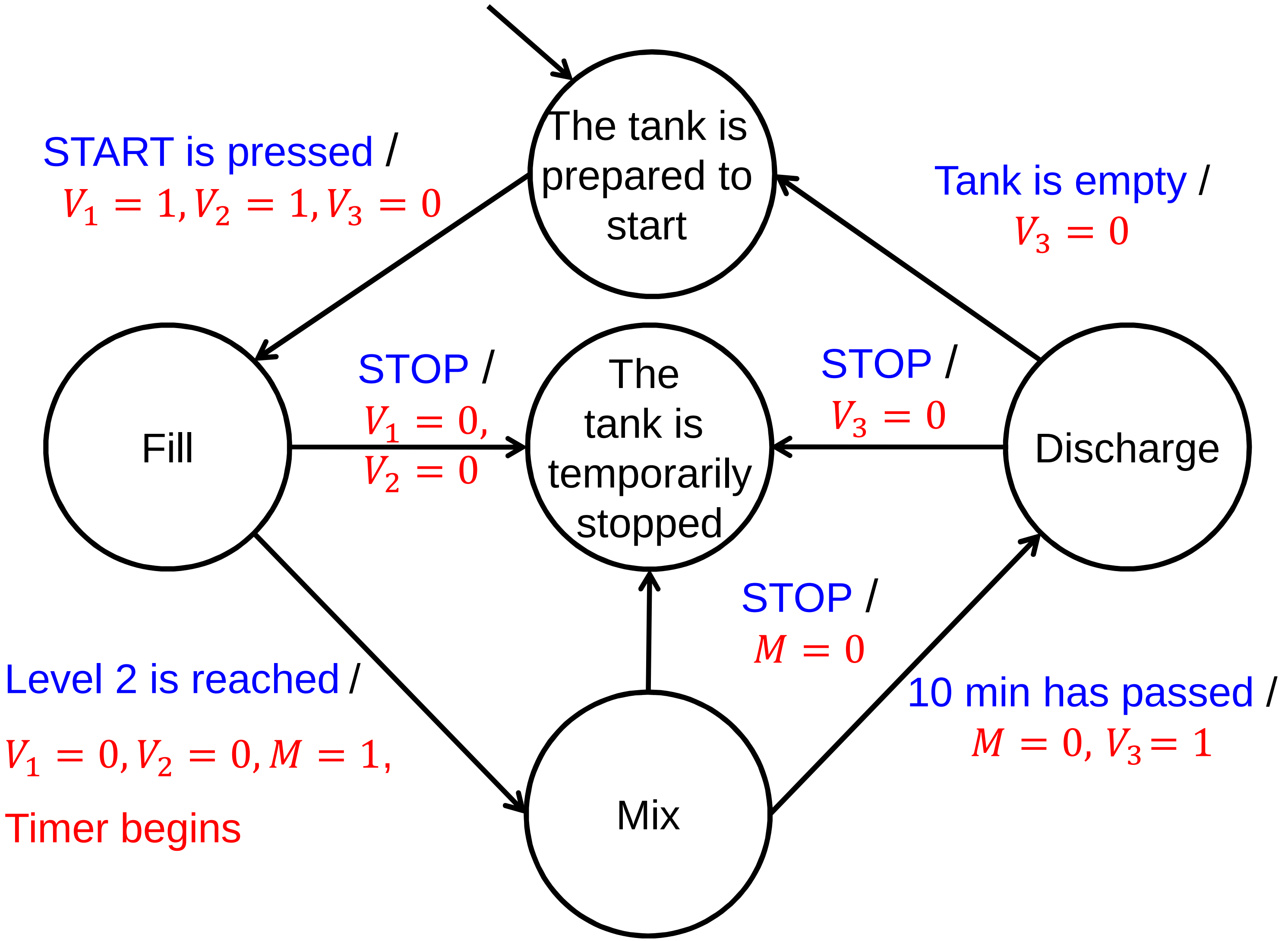
Modeling by FSA: Mixing tank

Model the mixed tank system by considering Specifications 1-4.



Modeling by FSA: Mixing tank

Model the mixed tank system by considering Specifications 1-5.



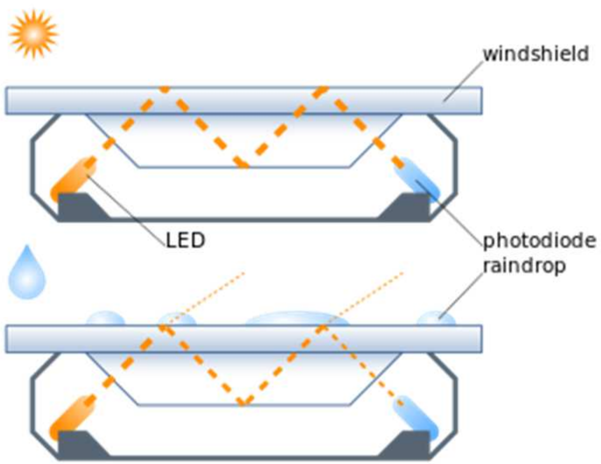
Example 3: Rain sensor

Working principles of rain sensors:

- An infrared light is emitted.
- Rain is detected based on how much light is reflected back.
- There may be other factors that influence the reflection of the light, such as dust, snow slush, etc.

→ false alarms: There is no rain, but the rain sensor indicates rain.

→ miss detections: There is rain, but the rain sensor indicates no rain.



Modeling by FSA: Rain sensor

States, inputs and outputs of the rain sensor system

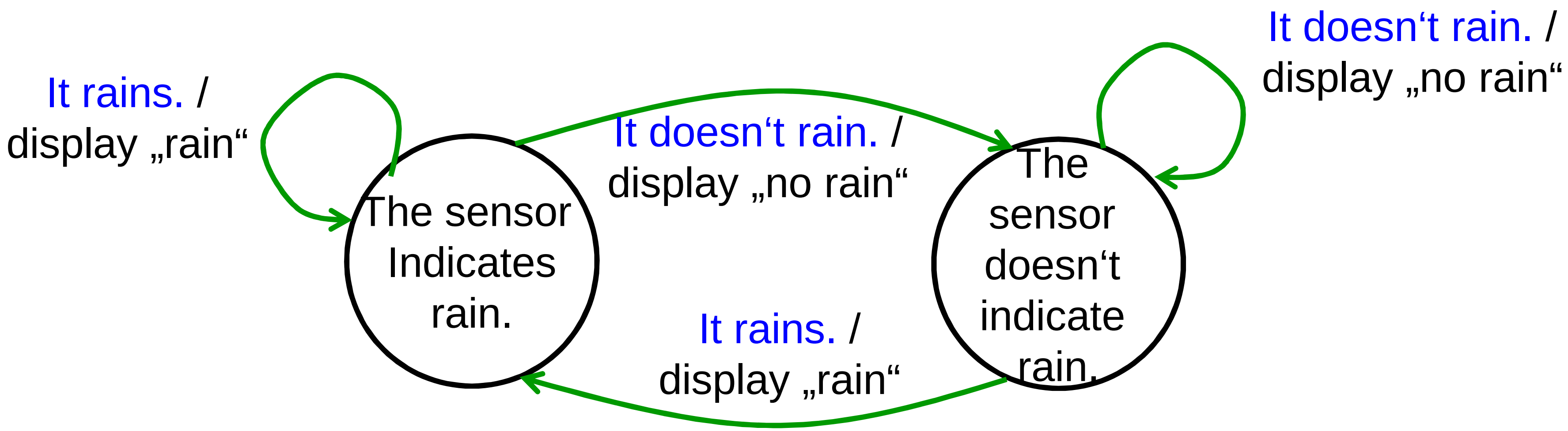
State	Description
Z_1	The rain sensor doesn’t indicate rain.
Z_2	The rain sensor indicates rain.

Input	Description
U_1	It doesn’t rain.
U_2	It rains.

Output	Description
Y_1	Display “no rain”
Y_2	Display “rain”

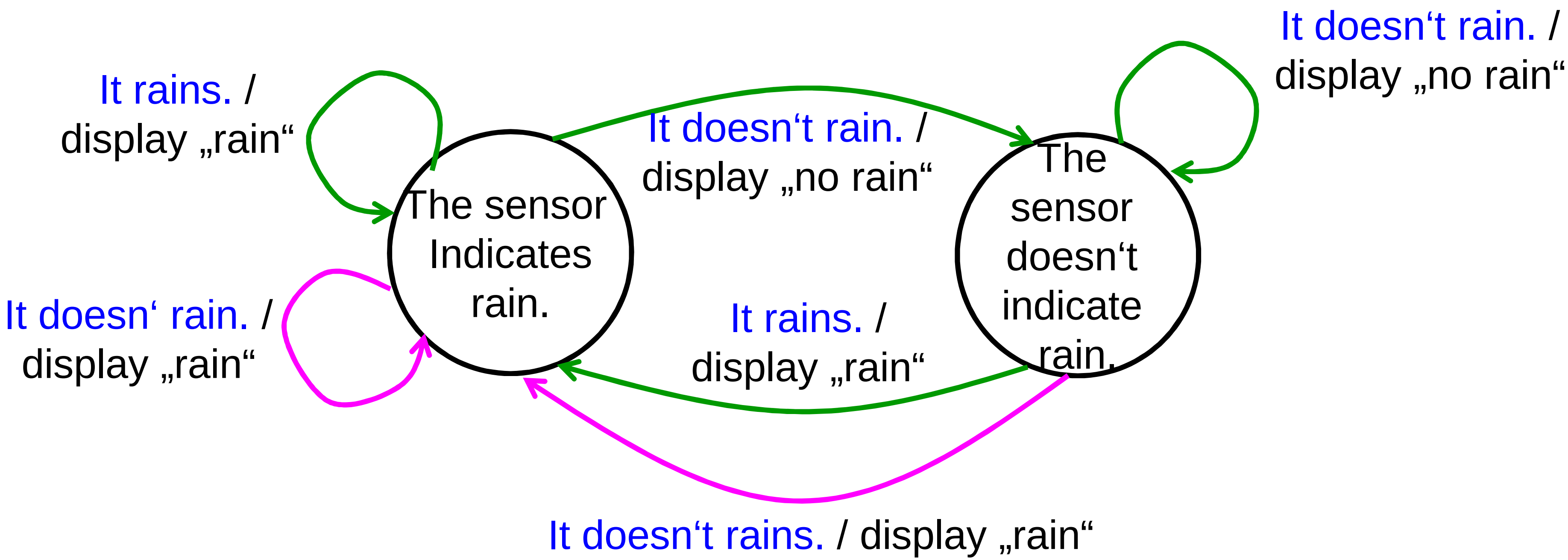
Modeling by FSA: Rain sensor

State transition diagram



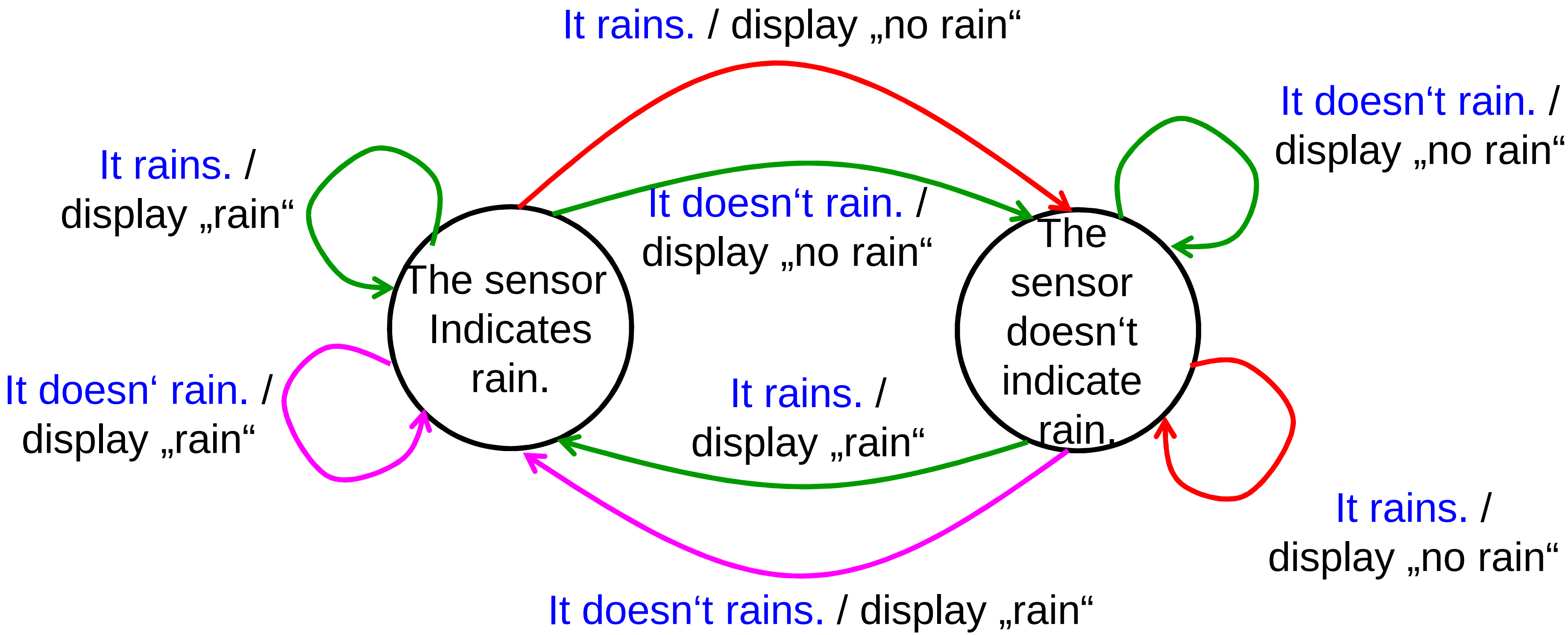
Modeling by FSA: Rain sensor

Take into account false alarms



Modeling by FSA: Rain sensor

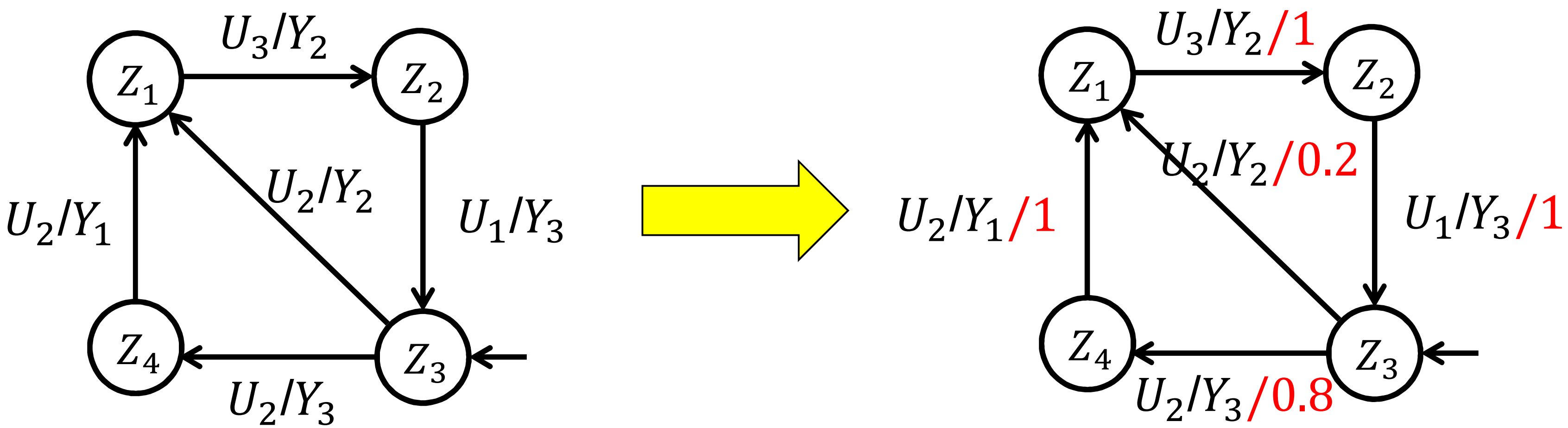
Take into account both false alarms and miss detections



Nondeterministic Automaton!

Stochastic Automata

- The state transition is regarded as a stochastic process.
- Some information like the probability of the state transition is known.
- The labels on the arcs of the automata are further extended to include such information.



$$\text{Prob}\{z(k+1) = Z_4, y(k) = Y_3 | z(k) = Z_3, u(k) = U_2\} = 0.8$$
$$\text{Prob}\{z(k+1) = Z_1, y(k) = Y_2 | z(k) = Z_3, u(k) = U_2\} = 0.2$$
$$\text{Prob}\{z(k+1) = Z_1, y(k) = Y_1 | z(k) = Z_4, u(k) = U_2\} = 1$$

Stochastic Automata

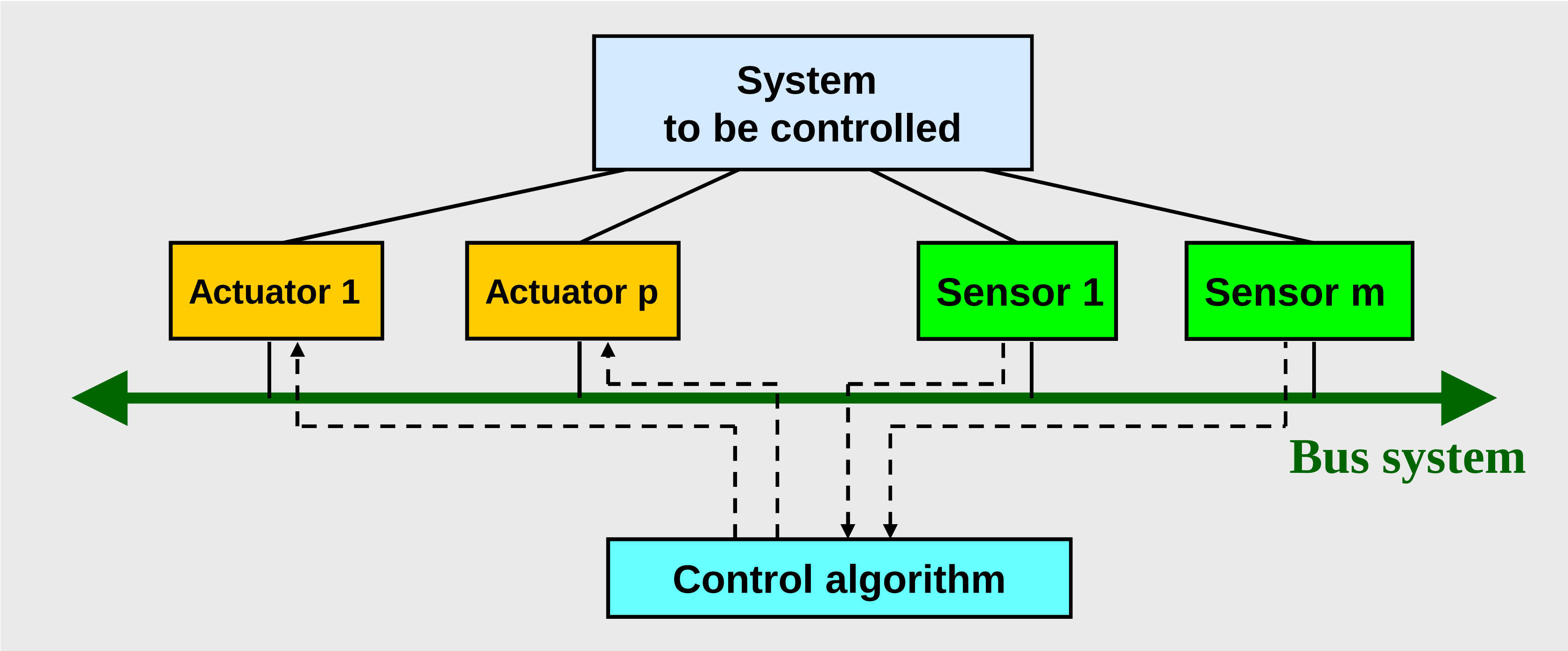
A special kind of stochastic automata: The state transition is governed by a **Markov chain**

$$\mathrm{Prob}\{z(k+1)=Z_{k+1}|z(k)=Z_k, z(k-1)=Z_{k-1},\cdots,z(0)=Z_0\}$$
$$=\mathrm{Prob}\{z(k+1)=Z_{k+1}|z(k)=Z_k\}$$

The probability that the next state $z(k+1)$ is Z_{k+1} depends **only** on the current state $z(k)$, but not on the past states $z(k-1), z(k-2),\cdots,z(0)$.

Stochastic Automata

Example: Description of packet loss in networked control systems

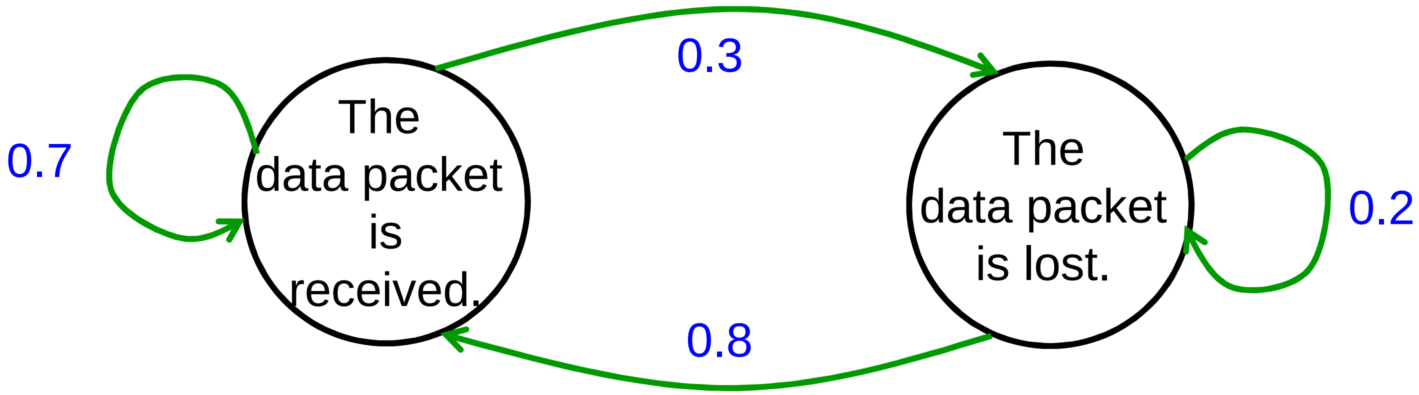


Network Quality of Service:

- Delay and jitter
- **Packet loss** and packet error
- Quantization error, …

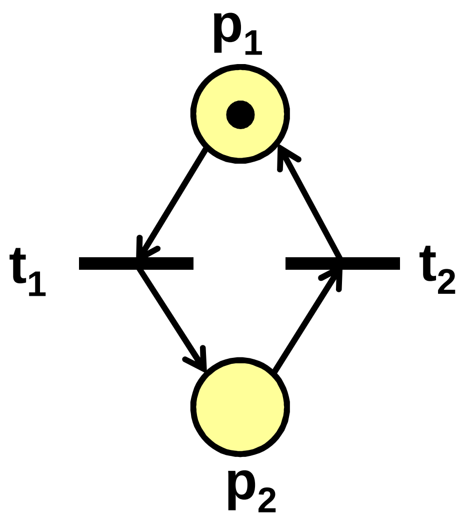
Stochastic Automata

The packet loss in the communication channel is often described by an **autonomous automaton**, whose state transition is governed by a **Markov chain**.



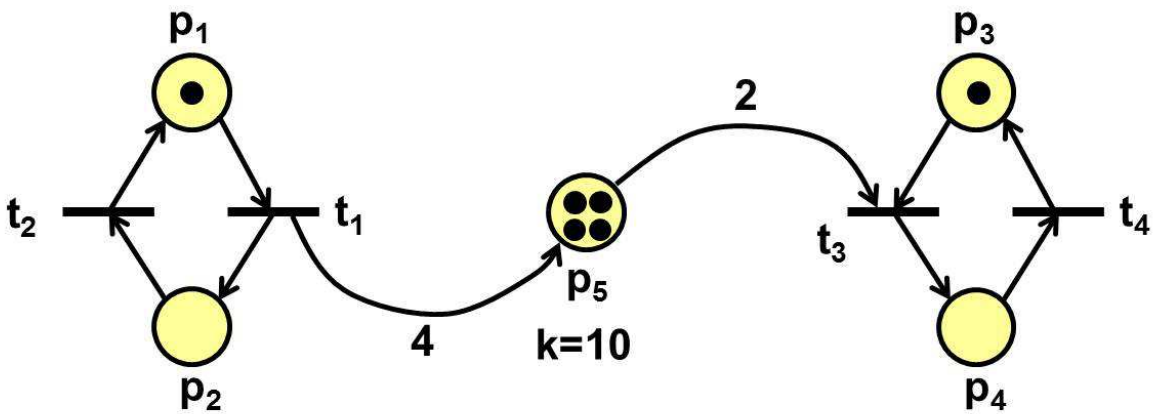
Overview of the course

- **Introduction**
- **Modeling of logic control systems**
  - Boolean algebra
  - Finite state automata
  - **Petri nets**, SIPN
- Analysis of logic control systems
- Design of logic control systems
- Verification and validation
- Implementation of logic control systems
  - PLC
  - Programming languages (IEC 61131-3)
  - Automatic code generation
- Distributed control (optional)

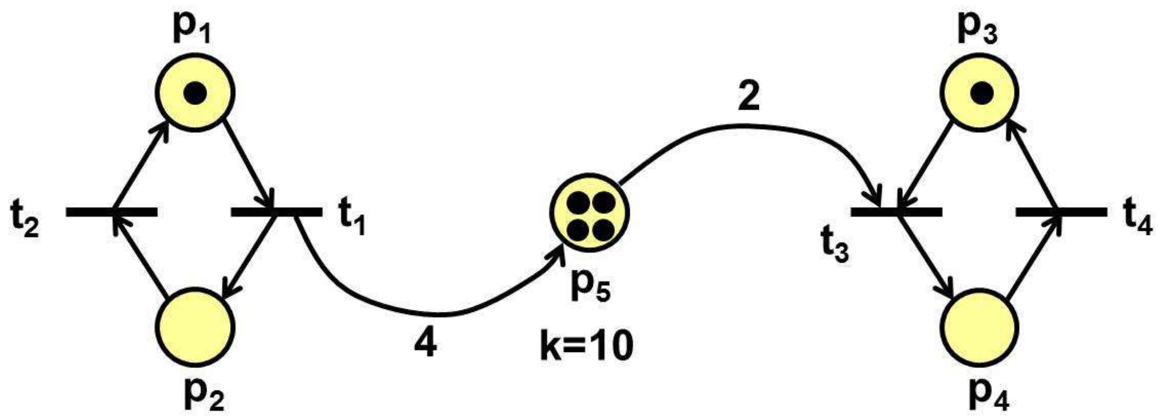


Petri nets (PN)

- Proposed by Carl Petri in 1961
- Petri nets are suitable for the description of **dynamic systems with discrete signals**, especially **concurrent processes**.



Petri nets



- One of the basic forms of Petri nets: **place transition net**
- Two types of nodes in a place transition net: **places** and **transitions**.
- **Directed arcs**: either from a place to a transition or from a transition to a place.
- Each transition has **pre-place(s)** and **post-place(s)**.
- Each place contains a number of **tokens**. The maximal number of tokens that can be put in one place is called the **capacity** of the place.
- The distribution of tokens in the petri net is called the **marking**.

Petri nets

- Tokens are moved by the **firing of transitions**. → system dynamics
- If a transition fires, then tokens will be removed from all its pre-places and all its post-places will receives tokens. The number of tokens that are removed from / added to one place is decided by the **weight** of the directed arc that is connected to that place.
- **Firing conditions** of a transition (i.e. the transition is activated / enabled):
  - Each **pre-place** of the transition has enough tokens.
  - Each **post-place** of the transition has enough capacity to receive the token.
- By the firing of transitions, the marking may change.
- Interpretation from the control perspective: **A marking corresponds to a state of the dynamic system**.
- If several transitions are enabled at the same time, it is assumed that these transitions can only fire individually and successively, but not simultaneously.