Contents

1 Introduction

1.1 Typical Sensors and Actuators

- Sensors
  - acceleration sensor
  - light sensor
  - force sensor
  - temperature sensor
  - video camera
  - pressure sensor
  - angle sensor
  - LIDAR
- Actuators
  - electric motor
  - hydraulic/pneumatic cylinder
  - magnetic valve
  - relay
  - heating
  - piezo actuator
  - pump
  - laser

1.2 Model-Based Design

1.2.1 Process

1. Modeling
2. Design
3. Analysis
4. Deployment

1.2.2 Advantages

- Improvement of the product quality
- Handling complexity
- Shorter development times

1.2.3 Concept of Systems

- **System:** Is a set of interacting or independent components that is distinguished from its environment by a system boundary
- **System boundary:** Describes the exchange of a system with its environment via inputs and outputs
- **Subsystem:** System in a system

1.3 Signal Types

1.3.1 Continuous Signals

$$f : \mathbb{R}_0^+ \to \mathbb{R}$$

1.3.2 Discrete-Time Signal

$$f : \mathcal{D} \to \mathbb{R}$$

where $\mathcal{D}$ is a countable set, e.g. $\mathcal{D} = \{t_1, t_2, \dots\}$ or $\mathcal{D} = \mathbb{N}_0$

1.3.3 Discrete-Value Signal

$$f : \mathbb{R}_0^+ \to \mathcal{D}$$

where $\mathcal{D}$ is a countable set, e.g. $\mathcal{D} = \{0, 1\}$

1.3.4 Discrete-Time and Discrete-Value Signal

$$f : \mathcal{D} \to \tilde{\mathcal{D}}$$

where $\mathcal{D}, \tilde{\mathcal{D}}$ are countable sets

1.4 Systems

	Discrete-State System	Continuous-State System	Hybrid System
States/Inputs/Outputs	discrete	continuous	discrete and continuous
Time variable	t_k (discrete)	t (continuous)	t (continuous)
Input variable	$u(t_k) \in \tilde{\mathcal{D}}_1$	$u(t) \in \mathbb{R}$	$u(t) \in \tilde{\mathcal{D}}_1$ or $\mathbb{R}$
Output variable	$y(t_k) \in \tilde{\mathcal{D}}_2$	$y(t) \in \mathbb{R}$	$y(t) \in \tilde{\mathcal{D}}_2$ or $\mathbb{R}$
State vector	$z(t_k)$	$x(t)$	$x(t)$
Equations	$z(t_{k+1}) = Az(t_k) + bu(t_k)$ $y(t_k) = c^T z(t_k)$	$\dot{x}(t) = Ax(t) + bu(t)$ $y(t) = c^T x(t)$	$\dot{x}(t) = Ax(t) + bu(t)$ $y(t) = c^T x(t)$

1.4.1 Properties

State of a dynamic system

- A state vector x consists of the (smallest) number of variables that need to be specified at the initial time t_0 so that the future behavior is uniquely defined for a given input signal $u(t)$

Static and Dynamic Systems

- **Static System:** No state required
- **Dynamic System:** State required

Time-Invariant and Time-Variant Systems

- **Time-invariant system:** Shift of time does not change the outcome
- **Time-variant system:** Shift of time alters the outcome

Deterministic and Non-Deterministic Systems

- **Deterministic system:** For an initial state $x(0)$ and a given input signal $u(t)$, there exists a unique solution of the state and the output
- **Non-deterministic system:** The evolution of the state and the output is not uniquely determined by the initial state and the input signal

Causal, Acausal, and anticausal Systems

- **Causal sytem:** The output only depends on past and current inputs
- **Acausal system:** The output also depends on future inputs
- **Anticausal system:** The output only depends on future inputs

2 Finite State Automata (FSA)

2.1 Mealy Machine

2.1.1 Definition

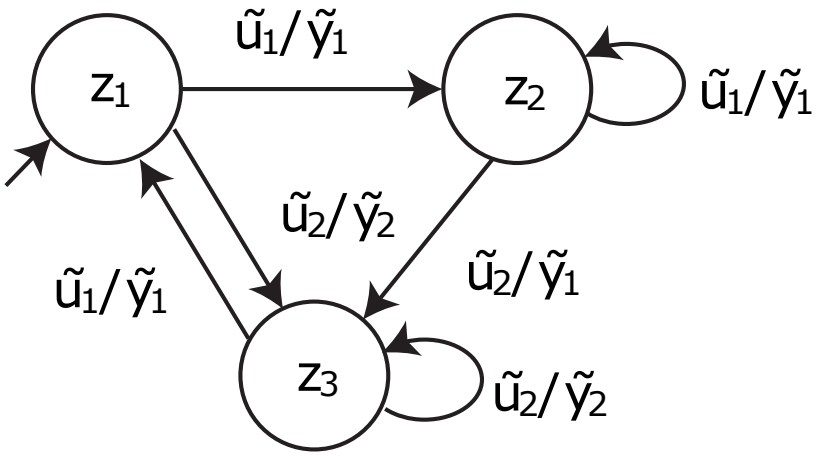
$$A_{mealy} = (\mathcal{Z}, \mathcal{U}, \mathcal{Y}, f, g, z(0))$$

where

$z(0)$	initial state
$\mathcal{Z} = \{z_1, \ldots, z_n\}$	set of states
$\mathcal{U} = \{\tilde{u}_1, \ldots, \tilde{u}_m\}$	set of input symbols (input alphabet)
$\mathcal{Y} = \{\tilde{y}_1, \ldots, \tilde{y}_o\}$	set of output symbols (output alphabet)

$f : \mathcal{Z} \times \mathcal{U} \to \mathcal{Z}$	transition function	$z(k+1) = f(z(k), u(k))$
$g : \mathcal{Z} \times \mathcal{U} \to \mathcal{Y}$	output function	$y(k) = g(z(k), u(k))$

2.1.2 Graphical Representation



2.1.3 State Transition Table

	$(z(k+1), y(k))$ for $u(k)$		
$z(k)$	$\tilde{u}_1$	$\ldots$	$\tilde{u}_m$
z_1	$(z(k+1), y(k))$	$\ldots$	$(z(k+1), y(k))$
z_2	$(z(k+1), y(k))$	$\ldots$	$(z(k+1), y(k))$
$\vdots$	$\vdots$	$\ddots$	$\vdots$

2.1.4 Transition Function and Output Function

Transition Function:

$$f(z(k), u(k)) = \begin{cases} z_1 \text{ for } & ((z(k) = z_{i_1}) \wedge ((u(k) = \tilde{u}_{j_{11}}) \vee (u(k) = \tilde{u}_{j_{12}}) \vee \ldots)) \vee \\ & ((z(k) = z_{i_2}) \wedge ((u(k) = \tilde{u}_{j_{21}}) \vee (u(k) = \tilde{u}_{j_{22}}) \vee \ldots)) \vee \\ & \qquad\qquad \vdots \\ z_2 \text{ for } & \ldots \\ \vdots & \qquad\qquad \vdots \end{cases}$$

where $z_{i_1}, z_{i_2}, \cdots \in \mathcal{Z}$, $\tilde{u}_{j_{11}}, \tilde{u}_{j_{12}}, \ldots, \tilde{u}_{j_{21}}, \cdots \in \mathcal{U}$

Output Function:

$$g(z(k),u(k)) = \begin{cases} \tilde{y}_1 \text{ for } & ((z(k)=z_{i_1}) \wedge ((u(k)=\tilde{u}_{j_{11}}) \vee (u(k)=\tilde{u}_{j_{12}}) \vee \dots)) \vee \\ & ((z(k)=z_{i_2}) \wedge ((u(k)=\tilde{u}_{j_{21}}) \vee (u(k)=\tilde{u}_{j_{22}}) \vee \dots)) \vee \\ & \vdots \\ \tilde{y}_2 \text{ for } & \dots \\ \vdots & \vdots \end{cases}$$

where $z_{i_1}, z_{i_2}, \dots \in \mathcal{Z}$, $\tilde{u}_{j_{11}}, \tilde{u}_{j_{12}}, \dots, \tilde{u}_{j_{21}}, \dots \in \mathcal{U}$

2.2 Moore Machine

2.2.1 Definition

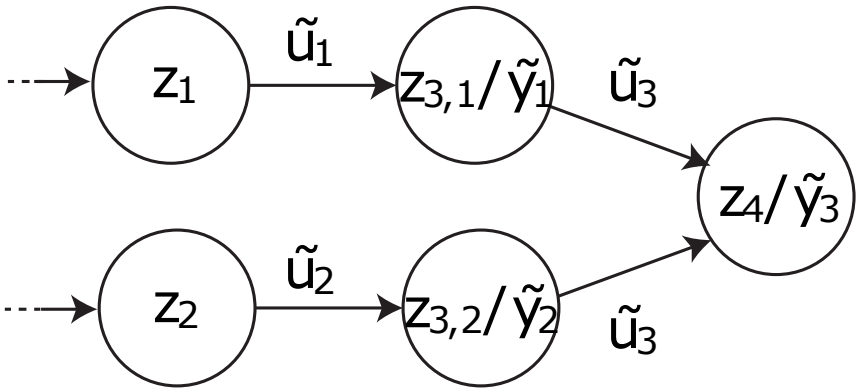
$$A_{moore} = (\mathcal{Z}, \mathcal{U}, \mathcal{Y}, f, g, z(0))$$

where

$z(0)$	initial state
$\mathcal{Z} = \{z_1, \dots, z_n\}$	set of states
$\mathcal{U} = \{\tilde{u}_1, \dots, \tilde{u}_m\}$	set of input symbols (input alphabet)
$\mathcal{Y} = \{\tilde{y}_1, \dots, \tilde{y}_o\}$	set of output symbols (output alphabet)

$f : \mathcal{Z} \times \mathcal{U} \to \mathcal{Z}$	transition function	$z(k+1) = f(z(k), u(k))$
$g : \mathcal{Z} \to \mathcal{Y}$	output function	$y(k) = g(z(k))$

2.2.2 Graphical Representation



2.2.3 State Transition Table

	$(z(k+1))$ for $u(k)$		
$z(k)/y(k)$	$\tilde{u}_1$	$\dots$	$\tilde{u}_m$
$z_1/y(k)$	$z(k+1)$	$\dots$	$z(k+1)$
$z_2/y(k)$	$z(k+1)$	$\dots$	$z(k+1)$
$\vdots$	$\vdots$	$\ddots$	$\vdots$

2.2.4 Transition Function and Output Function

Transition Function:

$$f(z(k),u(k)) = \begin{cases} z_1 \text{ for } & ((z(k)=z_{i_1}) \wedge ((u(k)=\tilde{u}_{j_{11}}) \vee (u(k)=\tilde{u}_{j_{12}}) \vee \dots)) \vee \\ & ((z(k)=z_{i_2}) \wedge ((u(k)=\tilde{u}_{j_{21}}) \vee (u(k)=\tilde{u}_{j_{22}}) \vee \dots)) \vee \\ & \vdots \\ z_2 \text{ for } & \dots \\ \vdots & \vdots \end{cases}$$

where $z_{i_1}, z_{i_2}, \dots \in \mathcal{Z}$, $\tilde{u}_{j_{11}}, \tilde{u}_{j_{12}}, \dots, \tilde{u}_{j_{21}}, \dots \in \mathcal{U}$

Output Function:

$$g(z(k)) = \begin{cases} \tilde{y}_1 \text{ for } & (z(k)=z_{i_1}) \vee (z(k)=z_{i_2}) \vee \dots \\ \tilde{y}_2 \text{ for } & \dots \\ \vdots & \vdots \end{cases}$$

where $z_{i_1}, z_{i_2}, \dots \in \mathcal{Z}$

2.3 Nondeterministic Finite State Automaton

2.3.1 Definition

A nondeterministic finite state automaton A_N has at least one state z_i, which has more than one next state for the same input $\tilde{u}_j$

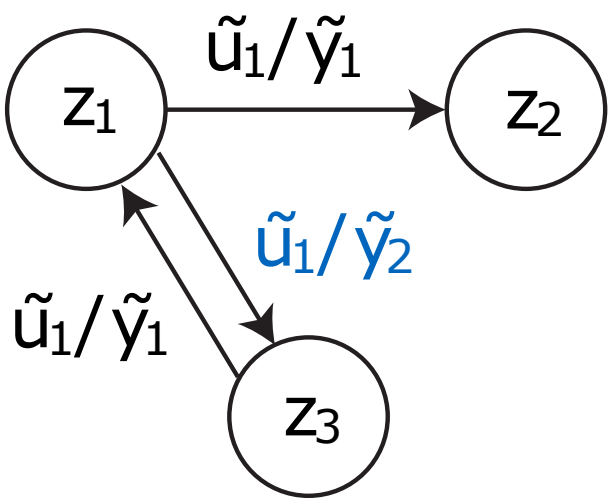
$$A_N = (\mathcal{Z}, \mathcal{U}, \mathcal{Y}, h, z(0))$$

where

$z(0)$	initial state
$\mathcal{Z} = \{z_1, \dots, z_n\}$	set of states
$\mathcal{U} = \{\tilde{u}_1, \dots, \tilde{u}_m\}$	set of input symbols (input alphabet)
$\mathcal{Y} = \{\tilde{y}_1, \dots, \tilde{y}_o\}$	set of output symbols (output alphabet)

$h : \mathcal{Z} \times U \to P(\mathcal{Z} \times \mathcal{Y})$ Potential pairs of next states and output

2.3.2 Graphical Representation



2.4 Events

2.4.1 Input/Output events

Input events:
The j-th event of the i-th input signal is defined as:

$$IE_j^{(i)}$$

Output events:
The j-th event of the i-th output signal is defined as:

$$OE_j^{(i)}$$

2.4.2 Multiple Signals

Input alphabet:
The input symbols based on the input signals are defined as $(i_1 \neq i_2 \neq \dots)$:

$$\begin{aligned}
\tilde{u}_1 &\hat{=} \{IE_{j_1}^{(i)}\} \\
\tilde{u}_2 &\hat{=} \{IE_{j_2}^{(i)}\} \\
&\vdots \\
\tilde{u}_m &\hat{=} \{IE_{j_1}^{(l)}\} \\
&\vdots \\
\tilde{u}_n &\hat{=} \{IE_{j_1}^{(i_1)}, IE_{j_2}^{(i_2)}\} \\
&\vdots \\
\tilde{u}_o &\hat{=} \{IE_{j_1}^{(i_1)}, IE_{j_2}^{(i_2)}, IE_{j_3}^{(i_3)}\} \\
&\vdots \\
&\vdots
\end{aligned}$$

Output alphabet:
The output symbols based on the output signals are defined as $(i_1 \neq i_2 \neq \dots)$:

$$\begin{aligned}
\tilde{y}_1 &\hat{=} \{OE_{j_1}^{(i)}\} \\
\tilde{y}_2 &\hat{=} \{OE_{j_2}^{(i)}\} \\
&\vdots \\
\tilde{y}_m &\hat{=} \{OE_{j_1}^{(k)}\} \\
&\vdots \\
\tilde{y}_n &\hat{=} \{OE_{j_1}^{(i_1)}, OE_{j_2}^{(i_2)}\} \\
&\vdots \\
\tilde{y}_o &\hat{=} \{OE_{j_1}^{(i_1)}, OE_{j_2}^{(i_2)}, OE_{j_3}^{(i_3)}\} \\
&\vdots \\
&\vdots
\end{aligned}$$

2.4.3 Don't care-Event

Any event is allowed for the i-th signal
Input:

$$\{IE_*^{(i)}\} \equiv \{\} \vee \{IE_1^{(i)}\} \vee \{IE_2^{(i)}\} \vee \dots$$

Output:

$$\{OE_*^{(i)}\} \equiv \{\} \vee \{OE_1^{(i)}\} \vee \{OE_2^{(i)}\} \vee \dots$$

2.5 Input/Output Sequences

Input sequence:

$$\bar{u} = (u(t_0), u(t_1), \dots), \qquad \forall k : u(t_k) \in \mathcal{U}$$

Output sequence:

$$\bar{y} = (y(t_0), y(t_1), \dots), \qquad \forall k : y(t_k) \in \mathcal{Y}$$

Input-Output Sequence

$$IOS = (\bar{u}, \bar{y})$$

Set of Input-Output Sequences
Set of all possible input-output sequences

$$SIOS = \{(\bar{u}, \bar{y}) | \bar{y} \text{ is an output sequence of the input sequence } \bar{u} = (u(t_0), u(t_1), \dots), \forall k : u(t_k) \in \mathcal{U}\}$$

2.6 Abstraction

2.6.1 Definition

An FSA A_B is an abstraction of an FSA A_A, if

$$SIOS_A \subseteq SIOS_B$$

2.6.2 Properties

- **Number of states:** Abstracted FSA has fewer states than the original FSA $\rightarrow$ Easier to analyze
- **Liveness:** If all input-output sequences in $SIOS_B$ fulfill a property, then all sequences in $SIOS_A$ fulfill this property
- **Safety:** If no input-output sequence of $SIOS_B$ fulfills a property, then no input-output sequence of $SIOS_A$ fulfills this property

2.7 Simulation

2.7.1 Definition

An FSA A_B simulates an FSA A_A, if there exists a simulation relation $\mathcal{S}_{AB}$, so that

- the initial states are in the simulation relation:

$$(z_A(0), z_B(0)) \in \mathcal{S}_{AB}$$

- for each transition, there exists a transition in the other FSA, so that the new states are within the simulation relation:

$$\forall u(k) \in \mathcal{U}, \forall (z_A(k+1), y_A(k)) \in h_A(z_A(k), u(k))$$
$$\exists (z_B(k+1), y_B(k)) \in h_B(z_B(k), u(k)) :$$
$$(z_A(k), z_B(k)) \in S_{AB} \wedge (z_A(k+1), z_B(k+1)) \in S_{AB} \wedge y_A(k) = y_B(k)$$

2.7.2 Verification

$\mathcal{S}_{AB} = \{(z_{A,i1}, z_{B,i1}), (z_{A,i2}, z_{B,i2}), \dots\}, \qquad z_{A,i1}, z_{A,i2}, \dots \in \mathcal{Z}_A, z_{B,i1}, z_{B,i2}, \dots \in \mathcal{Z}_B$

transition	simulating transition	$(z_A(k), z_B(k)) \in S_{AB}$ $\wedge (z_A(k+1), z_B(k+1)) \in S_{AB}$ $\wedge y_A(k) = y_B(k)$
$(z_A(k), u(k)) \rightarrow (z_A(k+1), y_A(k))$	$(z_B(k), u(k)) \rightarrow (z_B(k+1), y_B(k))$	
$\vdots$	$\vdots$	$\vdots$

2.7.3 Properties

- **Abstraction:** An FSA that simulates another FSA is also an abstraction of that FSA
- **Bisimulation:** An FSA A_B simulates an FSA A_A and vice versa.

3 Petri Nets

3.1 Definition

$$PN = (\mathcal{P}, \mathcal{T}, \mathcal{A})$$

where

$$\begin{aligned} &\mathcal{P} = \{P_1, \ldots, P_{n_p}\} && \text{set of places} \\ &\mathcal{T} = \{T_1, \ldots, T_{n_T}\} && \text{set of transitions} \\ &\mathcal{A} \subseteq \mathcal{P} \times \mathcal{T} \cup \mathcal{T} \times \mathcal{P} && \text{set of arcs} \\ &\mathcal{P} \cap \mathcal{T} = \emptyset \end{aligned}$$

Pre- and post-arcs:

$$\begin{aligned} &\mathcal{PRE} \subseteq \mathcal{P} \times \mathcal{T} && \text{Set of pre-arcs} \\ &\mathcal{POST} \subseteq \mathcal{T} \times \mathcal{P} && \text{Set of post-arcs} \end{aligned}$$

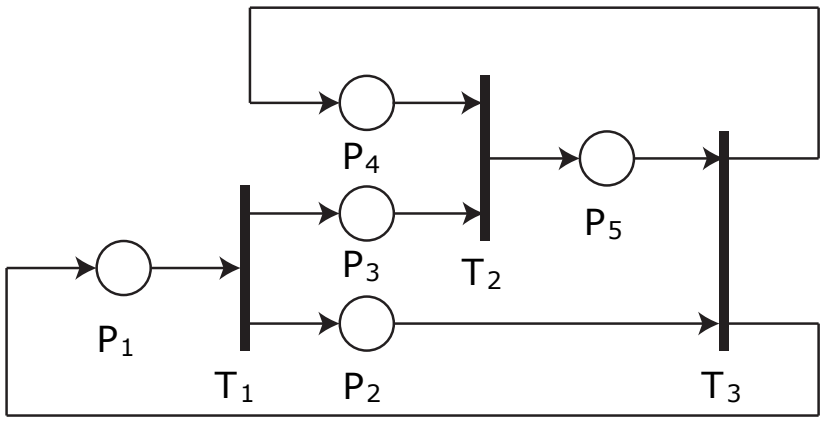
Pre- and post-places:

$$\begin{aligned} &{\bullet T_i} = \{p \in \mathcal{P} | (p, T_i) \in \mathcal{A}\} && \text{Pre-places of } T_i \\ &T_i{\bullet} = \{p \in \mathcal{P} | (T_i, p) \in \mathcal{A}\} && \text{Post-places of } T_i \end{aligned}$$

Pre- and post-transitions:

$$\begin{aligned} &{\bullet P_i} = \{t \in \mathcal{T} | (t, P_i) \in \mathcal{A}\} && \text{Pre-transitions of } P_i \\ &P_i{\bullet} = \{t \in \mathcal{T} | (P_i, t) \in \mathcal{A}\} && \text{Post-transitions of } P_i \end{aligned}$$

3.2 Graphical Representation



3.3 Place/Transition Petri Nets

Notes

This is a summary of the lecture Cyber-Physical Systems of the Technical University Munich. This lecture was presented by Althoff M. in the summer semester 2020. This summary was created by Gaida B. All provided information is without guarantee.