Computer Controlled Systems II – Diagnosis

Linear and nonlinear state space models
Automata and Petri net models

Katalin Hangos

University of Pannonia
Faculty of Information Technology
Department of Electrical Engineering and Information Systems

hangos.katalin@virt.uni-pannon.hu

Apr 2018

Lecture overview

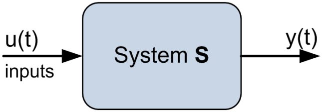
1. Linear and nonlinear state space models
   - Signals and systems
   - Input-output mapping
   - Continuous and discrete time state space models
   - Discrete event systems
2. Modelling for diagnosis
3. Automata models
4. Petri net models
   - Description forms
   - Operation (dynamics) of Petri nets
   - Parallel and conflicting execution steps
   - Solution of Petri net models - reachability graph
   - Coloured Petri Net models

System

System (**S**): acts on signals

$$y = \mathbf{S}[u]$$

- inputs ($u \in \mathcal{U}$) and outputs ($y \in \mathcal{Y}$)
- abstract operator ($\mathbf{S} : \mathcal{U} \rightarrow \mathcal{Y}$)



Input-output modeling

- Measurable variables
  - Data: measuring it for $[t_0, \;\; t_f]$
  - Input variables can be manipulated

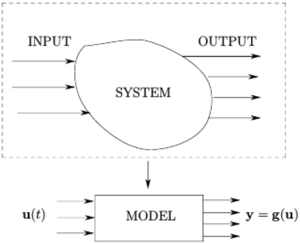
$$\{u_1(t), u_2(t), \ldots, u_p(t)\} \quad t_0 \geq t \geq t_f$$

  - Output variables can be directly measured

$$\{y_1(t), y_2(t), \ldots, y_m(t)\} \quad t_0 \geq t \geq t_f$$

  - Notation:

$$\boldsymbol{u}(t) = [u_1(t), u_2(t), \ldots, u_p(t)]^T$$
$$\boldsymbol{y}(t) = [y_1(t), y_2(t), \ldots, y_m(t)]^T$$



- Mathematical relationship

$$\left.\begin{array}{rcl} y_1(t) & = & g_1(u_1(t), \;\; \ldots, u_p(t)) \\ & \vdots & \\ y_m(t) & = & g_m(u_1(t), \;\; \ldots, u_p(t)) \end{array}\right\} \quad \boldsymbol{y} = \boldsymbol{g}(\boldsymbol{u})$$

State Space

Definition (State equations)

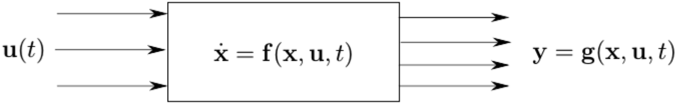
The set of equations required to specify the state $\mathbf{x}(t)$ for all $t \geq t_0$ given $\mathbf{x}(t_0)$and the function $\mathbf{u}(t)$, $t \geq t_0$, are called state equations.

Definition (State space)

The state space of a system, denoted by $\mathcal{X}$, is the set of all possible values that the state may take.

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t), t), \quad \mathbf{x}(t_0) = \mathbf{x_0} \qquad \text{(state equation)}$$

$$\mathbf{y}(t) = \mathbf{g}(\mathbf{x}(t), \mathbf{u}(t), t) \qquad \text{(output equation)}$$



Linear and Nonlinear Systems

Definition (Linear mapping)

The function $\boldsymbol{g}$ is said to be linear if and only if

$$\boldsymbol{g}(\alpha_1 \boldsymbol{u}_1 + \alpha_2 \boldsymbol{u}_2) = \alpha_1 \boldsymbol{g}(\boldsymbol{u}_1) + \alpha_2 \boldsymbol{g}(\boldsymbol{u}_2)$$

Linear state space model

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}(t)\boldsymbol{x}(t) + \boldsymbol{B}(t)\boldsymbol{u}(t)$$
$$\boldsymbol{y}(t) = \boldsymbol{C}(t)\boldsymbol{x}(t) + \boldsymbol{D}(t)\boldsymbol{u}(t)$$

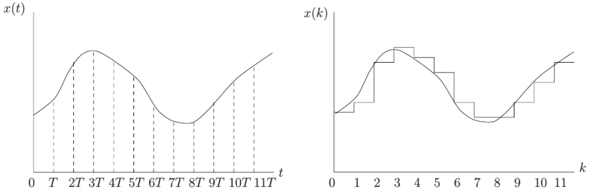
Linear time-invariant state space model

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{B}\boldsymbol{u}(t)$$
$$\boldsymbol{y}(t) = \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t)$$

Discrete-Time Systems

Why?

- Digital computers operate in a discrete-time fashion, it has an internal discrete-time clock.
- Many differential equations of continuous-time models can only be solved numerically using a computer.
- Some systems are inherently discrete-time, e.g. economic models based on quarterly recorded data, etc.



Important: Discretization of time does not imply the discretization of the state space!

Discrete-time state space models

- Nonlinear

$$x(k+1) = f(x(k), u(k), k), \qquad x(0) = x_0$$
$$y(k) = g(x(k), u(k), k)$$

- Linear

$$x(k+1) = A(k)x(k) + B(k)u(k), \qquad x(0) = x_0$$
$$y(k) = C(k)x(k) + D(k)u(k)$$

- Linear time-invariant

$$x(k+1) = Ax(k) + Bu(k), \qquad x(0) = x_0$$
$$y(k) = Cx(k) + Du(k)$$

Discrete time linear state space models

$$x(k+1) = \Phi x(k) + \Gamma u(k) \qquad (state\ equation)$$
$$y(k) = Cx(k) + Du(k) \qquad (output\ equation)$$

given initial condition $x(0)$;
vector valued signals

$$x(k) \in \mathcal{R}^n \ , \ y(k) \in \mathcal{R}^p \ , \ u(k) \in \mathcal{R}^r$$

system parameters:

$$\Phi \in \mathcal{R}^{n \times n} \ , \ \Gamma \in \mathcal{R}^{n \times r} \ , \ C \in \mathcal{R}^{p \times n} \ , \ D \in \mathcal{R}^{p \times r}$$

(Not necessarily) equidistant ($t_k - t_{k-1} = \Delta h$)

$$x(k) = x(t_k) \ \ , \ \ u(k) = u(t_k) \ \ , \ \ y(k) = y(t_k)$$

Continuous-State and Discrete-State Systems

Continuous The state space $\mathcal{X}$ is a continuum

Discrete The state space $\mathcal{X}$ is a discrete set

Hybrid Some variables are discrete, some are continuous

Discrete event systems

Characteristic properties:

- the *range space* of the signals (input, output, state) is **discrete**: $x(t) \in \mathbf{X} = \{x_0, x_1, ..., x_n\}$
- *event*: the occurrence of change in a discrete value
- *time is also **discrete***: $T = \{t_0, t_1, ..., t_n\} = \{0, 1, ..., n\}$

Only the **order of the events** is considered

- description of sequential and parallel events
- **application area**: scheduling, operational procedures, resource management

Discrete event systems – discrete time state space models

Generalization of discrete time linear state space models

$$x(k+1) = \Psi(x(k), u(k)) \qquad (\textit{state equation})$$
$$y(k) = h(x(k), u(k)) \qquad (\textit{output equation})$$

with given initial condition $x(0)$ and nonlinear state Ψ and output function h.

Discrete event system:

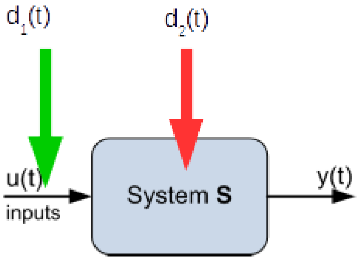
1. discrete time with non-equidistant sampling
2. the range space of the signals is discrete
3. event: change in the discrete value of a signal

System and its signals – revisited

System (**S**): acts on signals

$$y = \mathbf{S}[u, d]$$

- inputs ($u \in \mathcal{U}$), *disturbances ($d \in \mathcal{D}$)* and outputs ($y \in \mathcal{Y}$)
- abstract operator ($\mathbf{S} : \mathcal{U} \rightarrow \mathcal{Y}$)



Fault modelling

A fault/failure changes the dynamic behaviour of the ***nominal (fault-free) system*** that are described by

- an external non-measurable (directly observable) signal - a *disturbance*
- *modifying the model structure or parameters*

Fault indicator*: a (static) non-measurable (directly observable) variable* χ_{F_i} that is

- 0 when there is no fault F_i
- $\neq 0$ in the presence of F_i

Example: sensor with additive fault
Algebraic model equation: $v^m = v + \chi \cdot E$
$v, E \in \mathcal{Q}$, $v^m \in \mathcal{Q}_e$, $\chi \in B_{-1} = \{-1, 0, 1\}$

Automaton - abstract model: $\mathbf{A} = (Q, \Sigma, \delta; \Sigma_O, \varphi)$

- **Set of states**: Q
- **finite alphabet** of the input tape: $\Sigma = \{\#; a, b, ...\}$
- **State transition function**: $\delta : Q \times \Sigma \to Q$
- *Set of initial and final states*: $Q_I, \; Q_F \; \subseteq \; Q$
- **finite alphabet** of the output tape: $\Sigma_O = \{\#; \alpha, \beta, ...\}$
- **Output function**: $\varphi : Q \to \Sigma_O$

Graphical description: weighted directed graph

- **Vertices**: states (Q)
- **Edges**: state transitions (δ)
- **Edge weights**: input symbols (Σ)

Operation of automata

Given

- Initial state: $q_0 \in Q_I \subseteq Q$
- The content of the input tape: $S = [\sigma_1, \sigma_2, ..., \sigma_n]$, $\sigma_i \in \Sigma$

Compute

- Final state: if $q_f \in Q_F \subseteq Q$, then the automaton **accepts** the input
- The content of the output state: $S_O = [\zeta_1, \zeta_2, ..., \zeta_n]$, $\zeta_i \in \Sigma_O$

Automata - discrete event systems

	Automaton model	Discrete event state space model
State space	Q	$\mathcal{X} \in \mathbb{Z}^n$
Input u	string from Σ	discrete time discrete valued signal
Output y	string from Σ_O	discrete time discrete valued signal
State equation	$q(k+1) = \delta(q(k), u(k))$	$x(k+1) = \Psi(x(k), u(k))$
Output equation	$y(k) = \varphi(x(k))$	$y(k) = h(x(k), u(k))$

Overview - Petri nets: modelling and dynamics

Petri net - abstract description: $\mathbf{PN} = (P, T, I, O)$

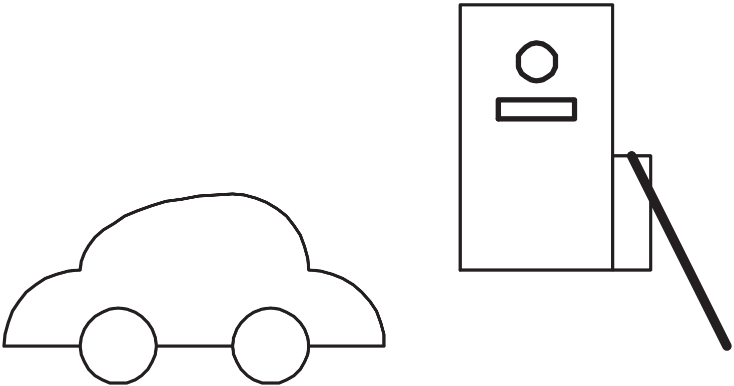
Static description (structure)

- set of **places (conditions)**: P
- set of **transitions (events)**: T
- **Input (pre-condition) function**: $I : T \to P^{\infty}$
- **Output (consequence) function**: $O : T \to P^{\infty}$

Graphical description: bipartite directed graph

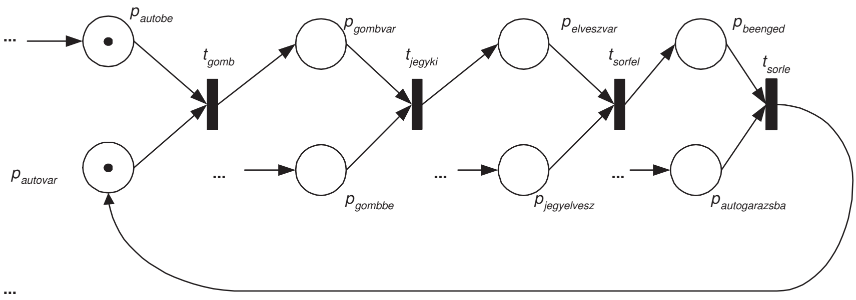
- **Vertices**: places (P) and transitions (T) (partitions)
- **Edges**: input and output functions (I, O)

Introductory example: Garage gate



Example: garage gate – 1

Petri net model - graphical description



Example: garage gate – 2

Petri net model - formal description

Places (states; inputs):

$$P = \{p_{autovar}, p_{gombvar}, p_{elveszvar}, p_{beenged} \; ; \; p_{autobe}, p_{gombbe}, p_{jegyelevesz}, p_{autoga}$$

Transitions:

$$T = \{t_{gomb}, t_{jegyki}, t_{sorfel}, t_{sorle}\}$$

Input function:

$$I(t_{gomb}) = \{p_{autobe}, p_{autovar}\} \quad , \quad I(t_{jegyki}) = \{p_{gombbe}, p_{gombvar}\}$$
$$I(t_{sorfel}) = \{p_{jegyelvesz}, p_{elveszvar}\} \quad , \quad I(t_{sorle}) = \{p_{beenged}, p_{autogarazsba}\}$$

Output function:

$$O(t_{gomb}) = \{p_{gombvar}\} \quad , \quad O(t_{jegyki}) = \{p_{elveszvar}\}$$
$$O(t_{sorfel}) = \{p_{beenged}\} \quad , \quad O(t_{sorle}) = \{p_{autovar}\}$$

The state of Petri nets

Marking function: marking points (**tokens**)

$$\mu : \mathbf{P} \to \mathcal{N} \quad , \quad \mu(p_i) = \mu_i \geq 0$$
$$\underline{\mu}^T = [\mu_1, \mu_2, \ldots, \mu_n] \quad , \quad n = |\mathbf{P}|$$

Transition **fires** (operates): when its pre-conditions are "true" (there is a **token** on its input places)

$$\underline{\mu}^{(i)}[t_j > \underline{\mu}^{(i+1)}$$

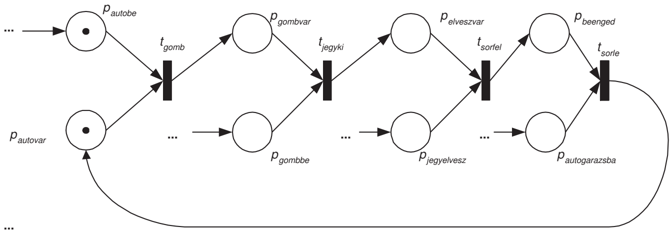
after firing the consequences become "true"

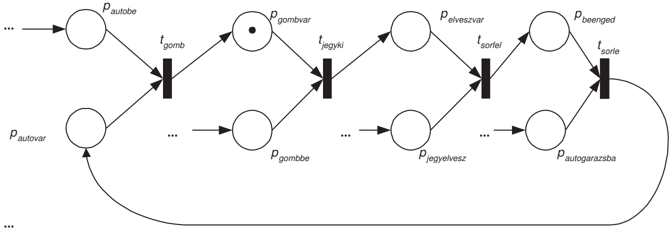
Firing (operation) sequence

$$\underline{\mu}^{(0)}[t_{j0} > \underline{\mu}^{(1)}[t_{j1} > ...[t_{jk} > \underline{\mu}^{(k+1)}$$

Example: garage gate – 3

One operation step





Example: garage gate – 4

Formal description of an operation step

Marking vector

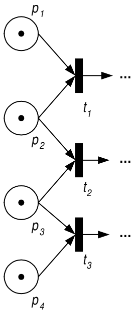
$$\underline{\mu}^T = [\mu_{autovar}, \mu_{gombvar}, \mu_{elveszvar}, \mu_{beenged} \; ; \; \mu_{autobe}, \mu_{gombbe}, \mu_{jegyelevesz}, \mu_{autogarazsba}]$$

Operation (firing) of transition t_{gomb}

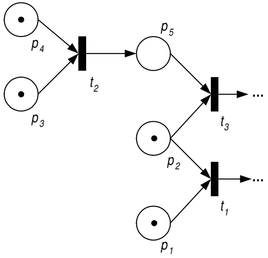
$$\underline{\mu}^{(1)}[t_{gomb} > \underline{\mu}^{(2)}$$
$$\underline{\mu}^{(1)} = [1,\ 0,\ 0,\ 0\ ;\ 1,\ 0,\ 0,\ 0]^T$$
$$\underline{\mu}^{(2)} = [0,\ 1,\ 0,\ 0\ ;\ 0,\ 0,\ 0,\ 0]^T$$

Parallel events

More than one enabled (fireable) transition:
concurrency (independent conditions), conflict, confusion



a,



b,

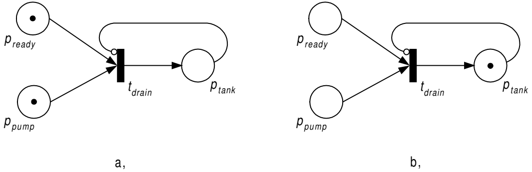
Conflict resolution

Using **inhibitor edges**:

- priority given by the user
- test edges

Other solutions:

- capacity of the places



The solution problem

Abstract problem statement

Given:

- a *formal description* of a discrete event system model
- *initial state(s)*
- *external events*: system inputs

Compute:

- the sequence of *internal (state and output) events*

The solution is **algorithmic**! **The problem is NP-hard**!

Petri net models – reachability graph

Solution: marking (systems state) sequences

reachability graph (tree) (weighted directed graph)

- *vertices*: markings
- *edges*: if exists transition the firing of which connects them
- *edge weights*: the transition and the external events

Construction:

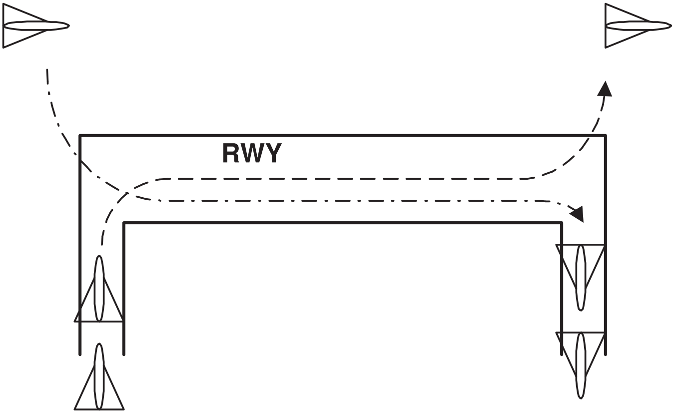
1. *start*: at the given initial state (marking)
2. *adding a new vertex*: by firing an enabled transition (with the effect of inputs!)

May be NP-hard (in conflict situation or non-finite operation)

Generalized Petri net models

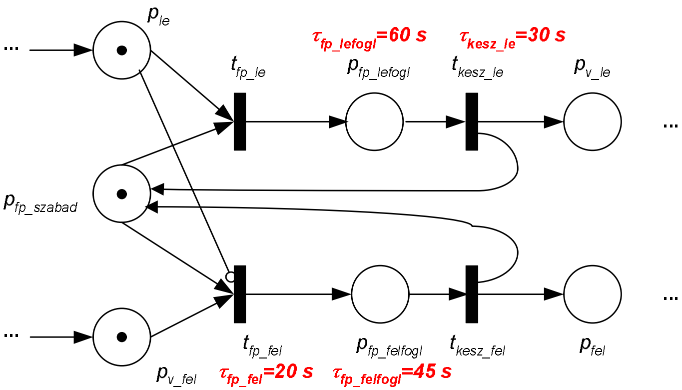
- **Hierarchical Petri nets**
- **Timed Petri nets**: using inscriptions
  - clock: built in (or special "source" place)
  - firing time to transitions
  - (waiting time for places)
- **Coloured Petri nets**: using inscriptions
  - tokens have discrete value ("colour")
  - colour set to places
  - discrete functions to the transitions and arcs

Simple example: Runway



Petri net model of a runway – 3

Timed Peri net model



Petri net model of a runway – 4

Coloured Peri net model: "insriptions"

Edge fucntion: a_{felki} : **if** $val(p_{fp\_lefogl}) =$ " $\uparrow$ " **then** "*true*"

$a_{fel} = val(p_{fp\_lefogl})$, $val(p_{fel}) = a_{fel}$

Colour set: $C_{felle} = \{ \uparrow , \downarrow \}$