Industrial Automation
(Automação de Processos Industriais)

Discrete Event Systems

http://users.isr.ist.utl.pt/~jag/courses/api1213/api1213.html

Slides 2010/2011 Prof. Paulo Jorge Oliveira
Rev. 2011-2013 Prof. José Gaspar
Syllabus:

Chap. 5 – CAD/CAM and CNC [1 week]

... 

Chap. 6 – Discrete Event Systems [2 weeks]

...

Chap. 7 – Analysis of Discrete Event Systems [2 weeks]
Some pointers to Discrete Event Systems

History: http://prosys.changwon.ac.kr/docs/petrinet/1.htm

Tutorial: http://vita.bu.edu/cgc/MIDEDS/
http://www.daimi.au.dk/PetriNets/


Simulators: http://www.informatik.hu-berlin.de/top/pnk/download.html

Bibliography:  
Generic characterization of systems resorting to input / output relations

State equations:

\[
\begin{align*}
\dot{x}(t) &= f(x(t), u(t), t) \\
y(t) &= g(x(t), u(t), t)
\end{align*}
\]

in continuous time (or in discrete time)

Examples?

Figure 1.1. Simple modeling process.
Open loop vs closed-loop (\(\leftrightarrow\) the use of feedback)

![Open-loop and closed-loop systems diagram](image)

**Figure 1.17.** Open-loop and closed-loop systems.

Advantages of feedback?

(to revisit during SEDs supervision study)
Example of closed-loop with feedback

\[ \lambda_{CL}(t) = \begin{cases} \rho & \text{if } x(t) < K \\ 0 & \text{if } x(t) = K \end{cases} \]

**Figure 1.18.** Flow system of Example 1.11 and closed-loop control model.
Discrete Event Systems: Examples

Set of events:

\[ E = \{N, S, E, W\} \]
Discrete Event Systems: Examples

Queueing systems

Set of events:
$E = \{\text{arrival, departure}\}$
Discrete Event Systems: Examples

Computational Systems

Processes Arrival

CPU

Processes Departure
Characteristics of systems with continuous variables

1. State space is continuous

2. The state transition mechanism is *time-driven*

Characteristics of systems with discrete events

1. State space is discrete

2. The state transition mechanism is *event-driven*

Polling is avoided!
Taxonomy of Systems

Figure 1.29. Major system classifications.
Levels of abstraction in the study of Discrete Event Systems

Languages

Timed languages

Stochastic timed languages
Systems’ Theory Objectives

• Modeling and Analysis
• Design and synthesis
• Control / Supervision
• Performance assessment and robustness
• Optimization

Applications of Discrete Event Systems

• Queueing systems
• Operating systems and computers
• Telecommunications networks
• Distributed databases
• Automation
Discrete Event Systems

Typical modeling methodologies

Automata

GRAFCET

Petri nets

Augmenting in modeling capacity and complexity
Automata Theory and Languages

Genesis of computation theory

**Definition:** A language $L$, defined over the alphabet $E$ is a set of *strings* of finite length with events from $E$.

Examples: $E = \{ \alpha, \beta, \gamma \}$

$L_1 = \{ \varepsilon, \alpha \alpha, \alpha \beta, \gamma \beta \alpha \}$

$L_2 = \{ \text{all strings of length 3} \}$

How to build a machine that “talks” a given language?

or

What language “talks” a system?
Operations / Properties of languages

\[ E^* = \text{Kleene-closure of } E: \] set of all strings of finite length of E, including the null element \( \varepsilon \).

Concatenation of \( L_a \) and \( L_b \):

\[ L_a L_b := \left\{ s \in E^* : s = s_a s_b, s_a \in L_a, s_b \in L_b \right\} \]

Prefix-closure of \( L \subseteq E^* \):

\[ \overline{L} := \left\{ s \in E^* : \exists_{t \in E^*} st \in L \right\} \]
Operations / Properties of languages

Example 2.1 (Operations on languages)
Let $E = \{a, b, g\}$, and consider the two languages $L_1 = \{\varepsilon, a, abb\}$ and $L_4 = \{g\}$. Neither $L_1$ nor $L_4$ are prefix-closed, since $ab \notin L_1$ and $\varepsilon \notin L_4$. Then:

\[
\begin{align*}
L_1L_4 &= \{g, ag, abbg\} \\
\overline{L_1} &= \{\varepsilon, a, ab, abb\} \\
\overline{L_4} &= \{\varepsilon, g\} \\
L_1\overline{L_4} &= \{\varepsilon, a, abb, g, ag, abbg\} \\
L_4^* &= \{\varepsilon, g, gg, ggg, \ldots\} \\
L_1^* &= \{\varepsilon, a, abb, aa, aabb, abba, abbabb, \ldots\}
\end{align*}
\]

[Cassandas99]
Automata Theory and Languages

Motivation: An automaton is a device capable of representing a language according to some rules.

Definition: A deterministic automaton is a 5-tuple

\[ (E, X, f, x_0, F) \]

where:
- \( E \) - finite alphabet (or possible events)
- \( X \) - finite set of states
- \( f \) - state transition function \( f: X \times E \rightarrow X \)
- \( x_0 \) - initial state \( x_0 \subset X \)
- \( F \) - set of final states or marked states \( F \subset E \)

[Cassandras93]
Example of an automaton

\((E, X, f, x_0, F)\)

\(E = \{ \alpha, \beta, \gamma \}\)

\(X = \{x, y, z\}\)

\(x_0 = x\)

\(F = \{x, z\}\)

\[\begin{align*}
  f(x, \alpha) &= x & f(x, \beta) &= z & f(x, \gamma) &= z \\
  f(y, \alpha) &= x & f(y, \beta) &= y & f(y, \gamma) &= y \\
  f(z, \alpha) &= y & f(z, \beta) &= z & f(z, \gamma) &= y
\end{align*}\]
Example of a stochastic automata

\[(E, X, f, x_0, F)\]

\[E = \{\alpha, \beta\}\]

\[X = \{0, 1\}\]

\[x_0 = 0\]

\[F = \{0\}\]

\[f(0, \alpha) = \{0, 1\}\]
\[f(0, \beta) = \{\}\]
\[f(1, \alpha) = \{\}\]
\[f(1, \beta) = 0\]

*Figure 2.4. State transition diagram for the nondeterministic automaton of Example 2.7.*
Given an automaton

\[ G=(E, X, f, x_0, F) \]

the **Generated Language** is defined as

\[ L(G) := \{ s \in E^* : f(x_0,s) \text{ is defined} \} \]

*Note: if \( f \) is always defined then \( L(G)=E^* \)

and the **Marked Language** is defined as

\[ L_m(G) := \{ s \in E^* : f(x_0,s) \in F \} \]
Example: marked language of an automaton

\[ L(G) := \{ \varepsilon, a, b, aa, ab, ba, bb, aaa, aab, baa, \ldots \} \]

\[ L_m(G) := \{a, aa, ba, aaa, baa, bba, \ldots \} \]

Concluding, in this example \( L_m(G) \) means all strings with events \( a \) and \( b \), ended by event \( a \).
Automata equivalence:

The automata $G_1$ e $G_2$ are equivalent if

$$L(G_1) = L(G_2)$$

and

$$L_m(G_1) = L_m(G_2)$$
Example of an automata:

Objective: To validate a sequence of events

Figure 2.6. State transition diagrams for digit sequence detector in Example 2.9.
Deadlocks (*inter-blocagem*)

Example:

The state 5 is a *deadlock*.

The states 3 and 4 constitutes a *livelock*.

How to find the *deadlocks* and the *livelocks*?

Need methodologies for the analysis of *Discrete Event Systems*
Deadlock:

in general the following relations are verified

\[ L_m(G) \subseteq \overline{L}_m(G) \subseteq L(G) \]

An automaton G has a deadlock if

\[ \overline{L}_m(G) \subseteq L(G) \]

and is not blocked when

\[ \overline{L}_m(G) = L(G) \]
Deadlock:

Example:

The state 5 is a **deadlock**.

The states 3 and 4 constitutes a **livelock**.

\[ L_m(G) = \{ ab, abgab, abgabgab, \ldots \} \]

\[ L(G) = \{ \varepsilon, a, ab, ag, aa, aab, abg, aaba, abga, \ldots \} \]

\[ (L_m(G) \subset L(G)) \]

\[ \overline{L_m}(G) \neq L(G) \]
Alternative way to detect deadlocks:

Example:

The state 5 is a *deadlock*.

The states 3 and 4 constitutes a *livelock*.
Timed Discrete Event Systems

**Figure 3.10.** The event scheduling scheme.
Petri nets
Developed by Carl Adam Petri in his PhD thesis in 1962.

**Definition:** A marked Petri net is a *5-tuple*

\[ (P, T, A, w, x_0) \]

where:

- **P** - set of places
- **T** - set of transitions
- **A** - set of arcs
- **w** - weight function
- **x_0** - initial marking
Example of a Petri net

\[(P, T, A, w, x_0)\]

\[P=\{p_1, p_2, p_3, p_4, p_5\}\]

\[T=\{t_1, t_2, t_3, t_4\}\]

\[A=\{(p_1, t_1), (t_1, p_2), (t_1, p_3), (p_2, t_2), (p_3, t_3),
(t_2, p_4), (t_3, p_5), (p_4, t_4), (p_5, t_4), (t_4, p_1)\}\]

\[w(p_1, t_1)=1, w(t_1, p_2)=1, w(t_1, p_3)=1, w(p_2, t_2)=1\]
\[w(p_3, t_3)=2, w(t_2, p_4)=1, w(t_3, p_5)=1, w(p_4, t_4)=3\]
\[w(p_5, t_4)=1, w(t_4, p_1)=1\]

\[x_0 = \{1, 0, 0, 2, 0\}\]
Example of a Petri net

\[(P, T, A, w, x_0)\]

\[P=\{p_1, p_2, p_3, p_4, p_5\}\]

\[T=\{t_1, t_2, t_3, t_4\}\]

\[A=\{(p_1, t_1), (t_1, p_2), (t_1, p_3), (p_2, t_2), (p_3, t_3),
(t_2, p_4), (t_3, p_5), (p_4, t_4), (p_5, t_4), (t_4, p_1)\}\]

\[w(p_1, t_1)=1, w(t_1, p_2)=1, w(t_1, p_3)=1, w(p_2, t_2)=1\]

\[w(p_3, t_3)=2, w(t_2, p_4)=1, w(t_3, p_5)=1, w(p_4, t_4)=3\]

\[w(p_5, t_4)=1, w(t_4, p_1)=1\]

\[x_0 = \{1, 0, 0, 2, 0\}\]
Petri nets

Rules to follow (mandatory):

• **Arcs** (directed connections)
  connect **places** to **transitions** and
  connect **transitions** to **places**

• A **transition** can have no **places** directly as inputs (source),
  *i.e. must exist arcs between transitions and places*

• A **transition** can have no **places** directly as outputs (sink),
  *i.e. must exist arcs between transitions and places*

• The same happens with the input and output **transitions** for **places**
Alternative definition of a Petri net

A marked Petri net is a 5-tuple

\((P, T, I, O, \mu_0)\)

where:

- \(P\) - set of places
- \(T\) - set of transitions
- \(I\) - transition input function \(I : T \rightarrow P^\infty\)
- \(O\) - transition output function \(O : T \rightarrow P^\infty\)
- \(\mu_0\) - initial marking \(\mu_0 : P \rightarrow \mathbb{N}\)

Note: \(P^\infty = \text{bag of places}\)
Example of a Petri net and its graphical representation

Alternative definition

$$(P, T, I, O, \mu_0)$$

$P = \{p_1, p_2, p_3, p_4, p_5\}$

$T = \{t_1, t_2, t_3, t_4\}$

$I(t_1) = \{p_1\}$ \hspace{1cm} $O(t_1) = \{p_2, p_3\}$

$I(t_2) = \{p_2\}$ \hspace{1cm} $O(t_2) = \{p_4\}$

$I(t_3) = \{p_3, p_3\}$ \hspace{1cm} $O(t_3) = \{p_5\}$

$I(t_4) = \{p_4, p_4, p_4, p_5\}$ \hspace{1cm} $O(t_4) = \{p_1\}$

$\mu_0 = \{1, 0, 0, 2, 0\}$
Petri nets

The state of a Petri net is characterized by the marking of all places.

The set of all possible markings of a Petri net corresponds to its state space.

How does the state of a Petri net evolve?
Execution Rules for Petri Nets (Dynamics of Petri nets)

A transition \( t_j \in T \) is enabled if:

\[
\forall p_i \in P : \ \mu(p_i) \geq \#(p_i, I(t_j))
\]

A transition \( t_j \in T \) may fire whenever enabled, resulting in a new marking given by:

\[
\mu'(p_i) = \mu(p_i) - \#(p_i, I(t_j)) + \#(p_i, O(t_j))
\]

\#(p_i, I(t_j)) = \text{multiplicity of the arc from } p_i \text{ to } t_j
\#(p_i, O(t_j)) = \text{multiplicity of the arc from } t_j \text{ to } p_i

[Peterson81 § 2.3]
Execution Rules for Petri Nets
(Dynamics of Petri nets)

\[
\mu'(p_i) = \mu(p_i) - \#(p_i,I(t_j)) + \#(p_i,O(t_j))
\]

[Peterson81 § 2.3]
Petri nets

Example of evolution of a Petri net

Initial marking:

$$\mu_0 = \{1, 0, 1, 2, 0\}$$

This discrete event system can not change state.

It is in a *deadlock*!
Petri nets: Conditions and Events

Example: Machine *waits until an order appears* and then *machines the ordered part* and *sends it out for delivery.*

Conditions:
- a) The server is idle.
- b) A job arrives and waits to be processed
- c) The server is processing the job
- d) The job is complete

Events
- 1) Job arrival
- 2) Server starts processing
- 3) Server finishes processing
- 4) The job is delivered

<table>
<thead>
<tr>
<th>Event</th>
<th>Pre-conditions</th>
<th>Pos-conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>b</td>
</tr>
<tr>
<td>2</td>
<td>a, b</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>d, a</td>
</tr>
<tr>
<td>4</td>
<td>d</td>
<td>-</td>
</tr>
</tbody>
</table>
**Petri nets:** Modeling mechanisms

**Concurrence**

**Conflict**
Petri nets: Modeling mechanisms

Mutual Exclusion

Producer / Consumer

Place m represents the permission to enter the critical section

B = one element buffer
Petri nets: Modeling mechanisms

Producer / Consumer with finite capacity

s Readers / t Writers
Discrete Event Systems

Example of a simple automation system modeled using PNs

An automatic soda selling machine accepts 50c and $1 coins and sells 2 types of products: SODA A, that costs $1.50 and SODA B, that costs $2.00.

Assume that the money return operation is omitted.

p₁: machine with $0.00;
t₁: coin of 50 c introduced;
t₈: SODA B sold.
Extensions to Petri nets

Switches [Baer 1973]

Possible to be implemented with restricted Petri nets.
Extensions to Petri nets

Inhibitor Arcs

Equivalent to nets with priorities

Can be implemented with restricted Petri nets?

Zero tests...

Infinity tests...
Extensions to Petri nets

P-Timed nets
Extensions to Petri nets

T-Timed nets

Job arrival
Start of processing
End of processing
Job is delivered

Jobs waits processing
Job is being processed
Job is complete
Server is idle
Extensions to Petri nets

Stochastic nets

Stochastic switches

Transitions with stochastic timings described by a stochastic variable with known pdf
Discrete Event Systems

Sub-classes of Petri nets

State Machine:

Petri nets where each transition has exactly one input arc and one output arc.

Marked Graphs:

Petri nets where each place has exactly one input arc and one output arc.
Discrete Event Systems

Example of DES:

Manufacturing system composed by 2 machines ($M_1$ and $M_2$) and a robotic manipulator (R). This takes the finished parts from machine $M_1$ and transports them to $M_2$.

No buffers available on the machines. If R arrives near $M_1$ and the machine is busy, the part is rejected.

If R arrives near $M_2$ and the machine is busy, the manipulator must wait.

Machining time: $M_1=0.5s$; $M_2=1.5s$; $R_{M1 \rightarrow M2}=0.2s$; $R_{M2 \rightarrow M1}=0.1s$;
Discrete Event Systems

Example of DES:

Variables of

\[ M_1 \quad x_1 \]
\[ M_2 \quad x_2 \]
\[ R \quad x_3 \]

Example of arrival of parts:

\[ a(t) = \begin{cases} 
1 & \text{in} \{0.1, 0.7, 1.1, 1.6, 2.5\} \\
0 & \text{in} \quad \text{other time stamps}
\end{cases} \]

\[ x_1 = \{\text{Idle, Busy, Waiting}\} \]
\[ x_2 = \{\text{Idle, Busy}\} \]
\[ x_3 = \{\text{Idle, Carrying, Returning}\} \]
Discrete Event Systems

Example of DES:

Definition of events:

\begin{align*}
a_1 & \quad \text{loads part in } M_1 \\
d_1 & \quad \text{ends part processing in } M_1 \\
r_1 & \quad \text{loads manipulator} \\
r_2 & \quad \text{unloads manipulator and loads } M_2 \\
d_2 & \quad \text{ends part processing in } M_2 \\
r_3 & \quad \text{manipulator at base}
\end{align*}
Discrete Event Systems

a_1

\[ t \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 2 & 3 & 4 \\
\end{array} \]
Discrete Event Systems

Example of DES:

Events:

- $a_1$ - loads part in $M_1$
- $d_1$ - ends part processing in $M_1$
- $r_1$ - loads manipulator
- $r_2$ - unloads manipulator and loads $M_2$
- $d_2$ - ends part processing in $M_2$
- $r_3$ - manipulator at base