Discrete Event Decision & Control
F.L. Lewis
Automation & Robotics Research Institute (ARRI)
The University of Texas at Arlington

Organized and invited by
Xiao Wendong

Thanks to
Tham Chen
Khong

F.L. Lewis, Fellow IEEE, Fellow IFAC
Moncrief-O’Donnell Endowed Chair
Head, Controls and Sensors Group
Automation & Robotics Research Institute (ARRI)
The University of Texas at Arlington

Discrete Event Control & Decision-Making

http://ARRI.uta.edu/acs

Springer-Verlag 2006
ARRI Intelligent Material Handling (IMH) Cell
3 robots, 3 conveyors, two part paths

**EXAMPLE**

Layout of the IMH Cell

Routing and Dispatching Decisions are Needed

**Rule-Based Controller**

- **Discrete event system**

  - IF Part A enters
    - AND Fixture 1 is available
    - AND Robot 1 is available
  THEN Robot 1 pick up part A
    - AND put into fixture 1

- **Tasks just completed**
- **Resources available**
- **Task sequencing**
- **Resource assignment**

**TWO TYPES of information**-

- IF (tasks just completed) AND (resources available)
  THEN (perform next task)
How to find the rules? Expert system?
Problems with:
- Redundant rules
- Inconsistent rules

How to implement the controller
Problems with:
- Real-time task sequencing
- Dynamic resource assignment
- Blocking phenomena

---

**Task flow information in Semiconductor Wafer Processing**

<table>
<thead>
<tr>
<th>Task</th>
<th>TT1</th>
<th>M1</th>
<th>R1</th>
<th>M2</th>
<th>R2</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Task3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

1. Job sequencing

<table>
<thead>
<tr>
<th></th>
<th>xport</th>
<th>mill</th>
<th>insert</th>
<th>grind</th>
<th>move</th>
<th>smooth</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Resource Assignment

<table>
<thead>
<tr>
<th></th>
<th>TT1</th>
<th>TT2</th>
<th>R1</th>
<th>R2</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>xport</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mill</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>grind</td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polish/smooth</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>move</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>insert</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>High polish</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

---

**Wireless Sensor Networks**

ARRI Distributed Intelligence & Autonomy Lab - DIAL

- Small mobile Sensor
- Sensor-Dan Popa
- Unattended Ground Sensors

**Mission Programming for Distributed Networks**

- Mission Programming and Execution
- Programmable Missions

**Deploy & Program**

- R1
- UGS2
- UGS4
- R3
- UGS1
- UGS5
- UGS3

Testbed containing MICA2 network (circle), Cricket network (triangle), Sentry robots, Garcia Robots & ARRI-bots

Dan Popa
**Fast Programming of Missions**

<table>
<thead>
<tr>
<th>Mission 1</th>
<th>Task sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task 1</strong></td>
<td>S1m1</td>
</tr>
<tr>
<td><strong>Task 2</strong></td>
<td>S2m1</td>
</tr>
<tr>
<td><strong>Task 3</strong></td>
<td>R1m1</td>
</tr>
<tr>
<td><strong>Task 4</strong></td>
<td>R2m2</td>
</tr>
<tr>
<td><strong>Task 5</strong></td>
<td>R1m3</td>
</tr>
<tr>
<td><strong>Task 6</strong></td>
<td>R1m4</td>
</tr>
<tr>
<td><strong>Task 7</strong></td>
<td>R1m5</td>
</tr>
<tr>
<td><strong>Task 8</strong></td>
<td>R2m6</td>
</tr>
<tr>
<td><strong>Task 9</strong></td>
<td>R1m7</td>
</tr>
<tr>
<td><strong>Task 10</strong></td>
<td>R1m8</td>
</tr>
<tr>
<td><strong>Task 11</strong></td>
<td>S2m9</td>
</tr>
<tr>
<td><strong>output</strong></td>
<td>y1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mission 2</th>
<th>Task sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Task 1</strong></td>
<td>S1m1</td>
</tr>
<tr>
<td><strong>Task 2</strong></td>
<td>R1m2</td>
</tr>
<tr>
<td><strong>Task 3</strong></td>
<td>R1m3</td>
</tr>
<tr>
<td><strong>Task 4</strong></td>
<td>R2m4</td>
</tr>
<tr>
<td><strong>Task 5</strong></td>
<td>R1m5</td>
</tr>
<tr>
<td><strong>Task 6</strong></td>
<td>R1m6</td>
</tr>
<tr>
<td><strong>Task 7</strong></td>
<td>R1m7</td>
</tr>
<tr>
<td><strong>Task 8</strong></td>
<td>R2m8</td>
</tr>
<tr>
<td><strong>Task 9</strong></td>
<td>R1m9</td>
</tr>
<tr>
<td><strong>Task 10</strong></td>
<td>S2m10</td>
</tr>
<tr>
<td><strong>output</strong></td>
<td>y2</td>
</tr>
</tbody>
</table>

**Discrete Event Supervisory Control**

- Fast programming of multiple missions
- Real-time event response
- Dynamic assignment of shared resources

**Objective:**

Develop new DE control algorithms for decision-making, supervision, & resource assignment

Apply to manufacturing workcell control, C&C systems, & inetworked systems

- Patent on Discrete Event Supervisory Controller
- New DE Control Algorithms based on Matrices
- Implemented on Intelligent Robotic Workcell
- Implemented on Wireless Sensor Networks
- Internet- Remote Site Control and Monitoring

**WSN is also a Reentrant Flow Line!**

**Routing and Dispatching Decisions are Needed**

**FEEDBACK CONTROL**

<table>
<thead>
<tr>
<th>Commands</th>
<th>Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start jobs</td>
<td>Workcell / WSN</td>
</tr>
<tr>
<td>Assign resources</td>
<td>Jobs done</td>
</tr>
<tr>
<td>Sensors</td>
<td>Resources available</td>
</tr>
</tbody>
</table>

**Events**

- Parts in targets detected

**DE Control Algorithms**

- patented
- new based on Matrices
- implemented on workcell
- implemented on Wireless Sensor Networks
- Internet- Remote Site Control and Monitoring

**History**

- Darwin – Natural Selection of Species
- Volterra – Population Balance
- Adam Smith - Economics
- James Watt – Steam Engine
Discrete event controller

- User interface: mission planning, resource allocation, priority rules
- Rule-based real time controller
  - Controller state monitoring logic: $T = F_u \otimes F_v \otimes F_c \otimes F_r \otimes F'_u \otimes F'_v \otimes F'_c \otimes F'_r$
  - Job start logic: $v_s = S_v \otimes x$
  - Resource release logic: $r_u = S_r \otimes x$
  - Task complete logic: $y = S_y \otimes x$

- Wireless Sensor Network
- Sensor readings
- Commands: Start jobs, Assign resources, Job vector - means jobs 1 and 4 have just completed
- Outputs: Resources available, Resource vector - means resources 2 and 3 are available
- Decision-making
- Events parts in targets detected

The Secret: multiply = AND & addition = OR

New Matrix Formulation for Supervisory Control

DE Model State Equation:
$$\bar{x} = F_u \bar{v}_c + F_r \bar{r}_c + F_u \bar{u} + F_D \bar{u}_D$$

Compare with $x_{k+1} = Ax_k + Bu_k$

- Task sequencing matrix – by Mission Planner
- Resource assignment matrix – by onsite Leader
- Resources available
- Targets / parts in
- Decision/Command input

Job Start Equation:
$$v_s = S_v x$$

Resource Release Equation:
$$r_s = S_r x$$

Product Output Equation:
$$y = S_y x$$

Compare with $y_k = Cx_k$

Binary mathematics - LOGIC not real numbers
Meaning of Matrices

\[
F_v \quad \text{Prerequisite jobs} \quad \text{Resources required} \quad F_r
\]

Next job

Task Sequencing Matrix

Steward

Conditions fulfilled

Task Start Matrix

Resource Requirements Matrix

Kusiak

Conditions fulfilled

Release resource

Resource Release Matrix

Next job


e.g. hierarchical HTFN task planners

Task Sequence

Reorder tasks to get lower diagonal matrix = causal

Two 1’s in a row = assembly

Construct Job Sequencing Matrix \( F_v \)

Used by Steward in Manufacturing
Task Sequencing

Contains same information
as the Bill of Materials
(BOM)

Mission/Task 1

Prerequisite jobs

Next jobs

Mission/Task 2

Construct Resource Requirements Matrix \( F_r \)

Used by Kusiak in
Manufacturing
Resource Assignment

Contains information
about factory resources

Next jobs

Prerequisite resources
Assign the resources – shop foreman

Assembly Tree

Two 1’s in a row = assembly

Multiple 1’s in a col. Means Shared resource

Guaranteed Performance due to DEC Structure

DEC Gives a Rule base that is:
- Consistent
- Nonredundant

Can add code to guarantee:
- No blocking phenomena

Discrete Event Controller

How does it work?

The secret:
- addition = OR
- multiply = AND
OR / AND Matrix Algebra

\[ \bar{x} = F_v \bar{V}_c + F_r \bar{r}_c \]

Example

\[ \bar{x} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} = (1 \land \bar{a}) \lor (0 \land \bar{b}) \lor (1 \land \bar{c}) \]

\[ x = (1 \land \bar{a}) \lor (0 \land \bar{b}) \lor (1 \land \bar{c}) \]

\[ x = (1 \land \bar{a}) \land (0 \land \bar{b}) \land (1 \land \bar{c}) \]

\[ x = (0 \lor a) \land (1 \lor b) \land (0 \lor c) \]

\[ x = a \land c \quad \text{Don't care about b} \]

Easy to implement OR/ AND algebra in MATLAB

Matrix multiply

\[ C = AB \]

for i= 1,I
  for j= 1,J
    c(i,j)=0
    for k= 1,K
      c(i,j)= c(i,j) .OR. ( a(i,k) .AND. b(k,j) )
    end
  end
end

Advantages of the Matrix Formulation

- Formal rigorous framework
- Complete DE dynamical description
- Relation to known Manufacturing notions
- Relation to other tools- Petri Nets, MAX-Plus
- Easy to design, change, debug, and test
- Formal deadlock analysis technique
- Easy to apply any conflict resolution (dispatching) strategy
- Optimization of resources
- Easy to implement in any platform (MATLAB, LabVIEW, C, C++, visual basic, or any other)

Relation to Max-Plus Algebra

\[ \bar{x} = F_v \bar{V}_c + F_r \bar{r}_c + F_u \bar{u} + F_D \bar{u}_D \quad \text{State equation} \]

\[ V_s = S_v x \]

\[ r_s = S_r x \]

\[ y = S_y x \quad \text{Output equations} \]

Define diagonal timing matrices. Then max plus is

\[ x' = S_v T_v F_v x + S_r T_r F_r r \quad \text{OPS. IN MAX-PLUS ALGEBRA} \]

Can also include nonlinear terms- correspond to decisions
Relation to Petri Nets

\[
\bar{x} = F_v \bar{v} + F_r \bar{r} + F_u \bar{u} + F_D \bar{u}_D
\]

State equation

\[
V_s = S_v x
\]

Output equations

\[
r_s = S_r x
\]

\[
y = S_y x
\]

\[x= \text{transition vector} – \text{fire the rules}\]

\[v, r = \text{place vectors} – \text{store the current situation (tokens)}\]

Task flow information

1. Job sequencing

- Every row (task) is a path in the PN

<table>
<thead>
<tr>
<th>Task</th>
<th>TT1</th>
<th>M1</th>
<th>R1</th>
<th>M2</th>
<th>R2</th>
<th>M4</th>
<th>…</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task1</td>
<td>xport</td>
<td>mill</td>
<td>insert</td>
<td>grind</td>
<td>move</td>
<td>smooth</td>
<td></td>
</tr>
<tr>
<td>Task2</td>
<td>xport</td>
<td>grind</td>
<td>move</td>
<td>High polish</td>
<td>xport</td>
<td>grind</td>
<td></td>
</tr>
<tr>
<td>Task3</td>
<td>xport</td>
<td>polish</td>
<td>insert</td>
<td>mill</td>
<td>xport</td>
<td>mill</td>
<td></td>
</tr>
</tbody>
</table>

2. Resource Assignment

<table>
<thead>
<tr>
<th>Resource Assignment</th>
<th>TT1</th>
<th>TT2</th>
<th>R1</th>
<th>R2</th>
<th>M1</th>
<th>M2</th>
<th>M3</th>
<th>M4</th>
</tr>
</thead>
<tbody>
<tr>
<td>xport</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>mill</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>grind</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Polish/smooth</td>
<td>1</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>move</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>insert</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>High polish</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Complete DE Dynamical Description

DE state equation

\[
\vec{x} = F_v \vec{v}_c + F_r \vec{r}_c + F_u \vec{u}_c + F_D \vec{u}_D
\]

Activity Completion Matrix

\[ F = [F_u \ F_v \ F_r \ F_y] \]

Activity Start Matrix

\[ S = [S_u^T \ S_v^T \ S_r^T \ S_y^T]^T \]

PN incidence matrix

\[ M = S^T - F = [S_u^T - F_u, S_v^T - F_v, S_r^T - F_r, S_y^T - F_y] \]

Marking transition equation

\[
m(t+1) = m(t) + M^T x = m(t) + [S^T - F]^T x
\]

\[
m = \begin{bmatrix} u^T & v^T & r^T & y^T \end{bmatrix}^T
\]

Include Process Times in Places

Split up marking vector

\[ m(t) = m_a(t) + m_p(t) \]

Add tokens

\[ m_p(t+1) = m_p(t) + S^T x(t) \]

Wait for process durations

\[ T = [O, vtimes^T, rtimes^T, O]^T \]

\[ T_{pend}(t+1) = \text{diag}(m_p(t)\{T_{pend}(t) - t_{sample}\}) + \text{diag}(S^T x(t))T \]

Take away tokens

\[ m_a(t+1) = m_a(t) - F x(t) \]

Allows easy MATLAB simulation of DE systems

1. Rules- fire transitions

\[
\vec{x} = F_v \vec{v}_c + F_r \vec{r}_c + F_u \vec{u}_c + F_D \vec{u}_D
\]

2. Add tokens to places

\[ m_p(t+1) = m_p(t) + S^T x(t) \]

3. Wait until jobs finish

Duration time counting routine

4. Take tokens from places

\[ m_a(t+1) = m_a(t) - F x(t) \]

5. Find updated vectors for DE state equation

\[ m(t+1) = m_a(t+1) + m_p(t+1) \]

Controller

System model

PN Marking Transition Eq.
**Fast Programming of Missions / Tasks**

### Mission / Task 1 - Job sequence

<table>
<thead>
<tr>
<th>Mission 1</th>
<th>notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 1</td>
<td>$u^1$</td>
<td>UGS1 launches chemical alert</td>
</tr>
<tr>
<td>Task 1</td>
<td>$S4m^1$</td>
<td>UGS4 takes measurement</td>
</tr>
<tr>
<td>Task 2</td>
<td>$R1gS2^1$</td>
<td>R1 goes to UGS2</td>
</tr>
<tr>
<td>Task 3</td>
<td>$R2gA^1$</td>
<td>R2 goes to location A</td>
</tr>
<tr>
<td>Task 4</td>
<td>$R3sS3^1$</td>
<td>R3 takes measurement</td>
</tr>
<tr>
<td>Task 5</td>
<td>$R4sS4^1$</td>
<td>R4 retrieves UGS2</td>
</tr>
<tr>
<td>Task 6</td>
<td>$R5sS5^1$</td>
<td>R5 takes measurement</td>
</tr>
<tr>
<td>Task 7</td>
<td>$R6sS6^1$</td>
<td>R6 listens for interrupts</td>
</tr>
<tr>
<td>Task 8</td>
<td>$R7sS7^1$</td>
<td>R7 goes to location R</td>
</tr>
<tr>
<td>Task 9</td>
<td>$R8sS8^1$</td>
<td>R8 takes measurement</td>
</tr>
<tr>
<td>Task 10</td>
<td>$R9sS9^1$</td>
<td>R9 deploys UGS2</td>
</tr>
<tr>
<td>Task 11</td>
<td>$S1m^1$</td>
<td>S1 takes measurement</td>
</tr>
</tbody>
</table>

**Output**: $y_1^1$ Mission 1 completed

### Mission 2 - Job sequence

<table>
<thead>
<tr>
<th>Mission 2</th>
<th>notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input 1</td>
<td>$u^2$</td>
<td>UGS3 batteries are low</td>
</tr>
<tr>
<td>Task 1</td>
<td>$S5m^2$</td>
<td>UGS5 takes measurement</td>
</tr>
<tr>
<td>Task 2</td>
<td>$R1gS3^2$</td>
<td>R1 goes to UGS3</td>
</tr>
<tr>
<td>Task 3</td>
<td>$R2gA^2$</td>
<td>R2 goes to location A</td>
</tr>
<tr>
<td>Task 4</td>
<td>$R3sS4^2$</td>
<td>R3 takes measurement</td>
</tr>
<tr>
<td>Task 5</td>
<td>$R4sS5^2$</td>
<td>R4 listens for interrupts</td>
</tr>
<tr>
<td>Task 6</td>
<td>$R5sS6^2$</td>
<td>R5 charges UGS3</td>
</tr>
<tr>
<td>Task 7</td>
<td>$R6sS7^2$</td>
<td>R6 deploys UGS2</td>
</tr>
<tr>
<td>Task 8</td>
<td>$R7sS8^2$</td>
<td>R7 takes measurement</td>
</tr>
<tr>
<td>Task 9</td>
<td>$R8sS9^2$</td>
<td>R8 docks the charger</td>
</tr>
<tr>
<td>Task 10</td>
<td>$R9sS10^2$</td>
<td>R9 takes measurement</td>
</tr>
<tr>
<td>Task 11</td>
<td>$S2m^2$</td>
<td>S2 takes measurement</td>
</tr>
</tbody>
</table>

**Output**: $y_2^2$ Mission 2 completed

### Simulation Results

- **Resources**
- **Mission/Task 1 jobs**
- **Mission/Task 2 jobs**

- **Event traces**
  - Up means task in progress
  - Down ⊗ means resource in use

**Simulation 2** – change mission/task priority

**PN Equivalent of Missions/Tasks**

**Mission 1 matrices**

$$F_1 = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

**Mission 2 matrices**

$$F_2 = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$
Discrete event controller
Implementation

Rule-based real time controller

Job start logic
\[ V_J = S_J \otimes x \]

Resource release logic
\[ r_s = S_r \otimes x \]

Task complete logic
\[ y = S_y \otimes x \]

Job start logic

Decision-making

Sensor readings

Tasks completed \( v_t \)
Resource released \( r_s \)
Mission completed \( r_m \)

User interface:
mission planning, resource allocation, priority rules

Remote Site Programming and DEC

- Fast programming of multiple missions
- Real-time event response
- Task sequencing
- Dynamic assignment of shared resources

Results of LabVIEW Implementation on Actual Workcell

Compare with MATLAB simulation!
We can now simulate a DE controller and then implement it, Exactly as for continuous state controllers!!

Man/Machine User Interface

Intelligent Robot Workcell

LabVIEW

USA/Mexico Internetworked Control
Shared resources → circular waits

- Initial resources available

Critical subsystems cannot get full

Shared Resources - which jobs to dispatch first?

LBFS always works – but too conservative

Need Supervisors to limit WIP in Critical Systems

ARRI Intelligent Material Handling (IMH) Cell
3 robots, 3 conveyors, two part paths

But gives negative marking!
Cannot fire both.
Simulation results

Event time trace

Deadlock Analysis- easy with matrix DEC

Find circular waits

\[ G_W = S_j F_r \]  

(in AND/OR algebra)

i.e. if \( g_{ij} = 1 \) then resource \( j \) waits for resource \( i \)

Use string algebra to find a matrix \( C \), wherein each column represents a circular wait.

Find critical siphons

\[ Sc = \begin{bmatrix} C_i^T S F_r \wedge C_r^T F_r \end{bmatrix} \]

where \( \wedge \) denotes the element-by-element matrix 'and' operation.

CS cannot get empty

To avoid deadlock, Critical Siphons cannot become empty.
compute the Critical System for each CS using

\[ \text{Crit. Sys.} = PC \wedge \overline{S}_c \]

Critical systems cannot take up all resources

\( P \) = binary basis for p-invariants

Deadlock Avoidance Dispatching  MAXWIP Policy

\[ \bar{x} = F_v \bar{v}_c + F_r \bar{r}_c + F_D \bar{u} + F_D \bar{u}_D \]

Deadlock Avoid Software Selects Dispatching Input \( u_D \)

Discrete event controller

Deadlock avoidance software here at top
- Decision-making

Sensor readings

Command

Resource release logic

Task complete logic

Sensor output \( u \)

Start tasks \( v_z \)

Start resource release \( r_z \)

Mission completed

Plant commands

Rule-based real time controller

Dispatching rules

Wireless Sensor Network

Sensor readings

Mission completed

Sensor output \( u \)

Output \( y \)

Task completed \( v_x \)

Resource released \( r_x \)

Start resource release \( r_z \)

Controller state monitoring logic

Decision-making
Simulation results

Event time trace without deadlock avoidance

Event time trace with deadlock avoidance