FROM PROGRAMMABLE LOGIC CONTROL (PLC) TO DISCRETE EVENT SYSTEMS (DES)

C. G. Cassandras
Dept. of Manufacturing Engineering
Boston University
Boston, MA 02215

cgc@bu.edu

http://vita.bu.edu/cgc
OUTLINE

• A MOTIVATING EXAMPLE
• DES MODELING: STATE AUTOMATA (STATE MACHINES)
• SUPERVISORY CONTROL
• ‘RAPID’ RECONFIGURATION TECHNIQUES
A MOTIVATING EXAMPLE

- **START** initiates cycle with **LS1** and **LS3** ON
- **SOL1** activated, piston 1 moves right
- **LS2** triggered, **SOL1** deactivated, piston 1 moves left
- Time delay of 1 sec.
- **SOL2** activated, piston 2 moves right
- **LS2** triggered, **SOL1** deactivated, piston 2 moves left
- **STOP** resets everything to rest
EXAMPLE – PLC APPROACH

Cylinder 1 control: \( SOL1 = (SOL1 + \text{START} \cdot LS1 \cdot LS3) \cdot \overline{LS2} \cdot \overline{STOP} \)

NOTE: Upon deactivation of SOL1, the system looks IDENTICAL as at rest, yet it must know to activate the DELAY function.
Solution: Introduce some memory...

\[
DONE = (DONE + LS2) \cdot \overline{LS4} \cdot \overline{STOP}
\]

\[
TIMER = LS1 \cdot \overline{DONE}
\]

\[
SOL2 = (SOL2 + LS1 \cdot LS3 \cdot \overline{DELAY}) \cdot \overline{LS4} \cdot \overline{STOP}
\]
EXAMPLE – LADDER DIAGRAM

Cylinder 1 control

Cylinder 1 cycle DONE

Delay

Cylinder 2 control
Define:

- **System STATES**
  (e.g., “System at rest”, “Piston 1 moving right”)

- **EVENTS** causing state transitions
  (e.g., “LS2 triggered”, “Timer activated”)

---

**EXAMPLE – DES APPROACH**

Christos G. Cassandras
- Boston University

CODES Lab.
**EXAMPLE – DES APPROACH**

**STATES**

<table>
<thead>
<tr>
<th>S1</th>
<th>Rest (LS1, LS3 ON)</th>
</tr>
</thead>
<tbody>
<tr>
<td>S2</td>
<td>Piston 1 moves right</td>
</tr>
<tr>
<td>S3</td>
<td>Piston 1 moves left</td>
</tr>
<tr>
<td>S4</td>
<td>Timer ON</td>
</tr>
<tr>
<td>S5</td>
<td>Piston 2 moves right</td>
</tr>
<tr>
<td>S6</td>
<td>Piston 2 moves left</td>
</tr>
</tbody>
</table>

**EVENTS**

<table>
<thead>
<tr>
<th>E1</th>
<th>START</th>
</tr>
</thead>
<tbody>
<tr>
<td>E2</td>
<td>LS2 triggered</td>
</tr>
<tr>
<td>E3</td>
<td>LS1 triggered</td>
</tr>
<tr>
<td>E4</td>
<td>TIMER activated</td>
</tr>
<tr>
<td>E5</td>
<td>LS4 triggered</td>
</tr>
<tr>
<td>E6</td>
<td>LS3 triggered</td>
</tr>
<tr>
<td>RESET</td>
<td>STOP</td>
</tr>
</tbody>
</table>

---

Christos G. Cassandras
- Boston University

CODES Lab.
Two main elements in a *Discrete Event System*:

1. **STATE** - the status of some system component
2. **EVENT** - instantaneous action causing a state transition

**STATE TRANSITION DIAGRAM:**

**STATE TRANSITION FUNCTION:**

\[ X' = f(X, E) \]

**NOTE:** Only *FEASIBLE* events at state \( X \) are considered
**AUTOMATON**: \((E, X, \Gamma, f, x_0)\)

- **\(E\)**: Event Set
- **\(X\)**: State Space
- \(\Gamma(x)\): Set of *feasible* or *enabled* events at state \(x\)
- **\(f\)**: State Transition Function
  
  \(f : X \times E \rightarrow X\)

  (undefined for \(e \in \Gamma(x)\))

- **\(x_0\)**: Initial State

\[ \{e_1, e_2, \ldots\} \rightarrow f(x, e) = x' \rightarrow \{x_1, x_2, \ldots\} \]
Add a *Clock Structure* $V$ to the Automaton: $(E, X, \Gamma, f, x_0, V)$

where:

$$V = \{ v_i : i \in E \}$$

and $v_i$ is a *Clock or Lifetime* Sequence:

$$v_i = \{ v_{i1}, v_{i2}, K \}$$

one for each event $i$.

Need an *internal mechanism* to determine NEXT EVENT $e'$ and hence NEXT STATE $x' = f(x, e')$. 
HOW THE TIMED AUTOMATON WORKS...

CURRENT STATE

\(x \in X\) with feasible event set: \(\Gamma(x)\)

CURRENT EVENT

\(e\) that caused transition into \(x\)

CURRENT EVENT TIME

\(t\) associated with \(e\)

Associate a

\(CLOCK\ VALUE/RESIDUAL\ LIFETIME\ y_i\)

with each feasible event \(i \in \Gamma(x)\)
HOW THE TIMED AUTOMATON WORKS...

NEXT/TRIGGERING EVENT $e'$:

$$e' = \arg \min_{i \in \Gamma(x)} \{ y_i \}$$

NEXT EVENT TIME $t'$:

$$t' = t + y^*$$

where: $y^* = \min_{i \in \Gamma(x)} \{ y_i \}$

NEXT STATE $x'$:

$$x' = f(x, e')$$
Detemine new \textit{CLOCK VALUES} $y'_i$
for every event $i \in \Gamma(x)$

$$y'_i = \begin{cases} 
  y_i - y^* & i \in \Gamma(x'), i \in \Gamma(x), i \neq e' \\
  v_{ij} & i \in \Gamma(x') - \{\Gamma(x) - e'\} \\
  0 & \text{otherwise}
\end{cases}$$

where: $v_{ij}$ = new lifetime for event $i$
STATE : No. of parts in workcenter \{0,1,\ldots,K\}

EVENTS :  
- part ARRIVALS (from conveyor belt)
- part DEPARTURES (from machine)

No new part allowed if no KANBAN available (Belt OFF until KANBAN is available)

When part leaves, KANBAN returned to available pool

KANBAN pool
TIMED AUTOMATON - AN EXAMPLE

\[ E = \{a, d\}, \quad X = \{0, 1, K, \ldots, K\} \]

\[ \Gamma(x) = \{a, d\}, \quad \text{for all } x > 0 \]

\[ \Gamma(0) = \{a\} \]

\[ f(x, e') = \begin{cases} 
  x + 1 & e' = a, \ x < K \\
  x - 1 & e' = d, \ x > 0 
\end{cases} \]

Given input: \[ v_a = \{v_{a1}, v_{a2}, K\}, \quad v_d = \{v_{d1}, v_{d2}, K\} \]
$x_0 = 0$

$e_1 = a$

$x_1 = 1$

$e_2 = a$

$x_2 = 2$

$e_3 = a$

$x_3 = 3$

$e_4 = d$

$x_4 = 2$

$t_0$

$t_1$

$t_2$

$t_3$

$t_4$

$a$

$d$

$a$

$d$

$a$

$d$

$a$

$d$
• Same idea with the Clock Structure consisting of *Stochastic Processes*

• Associate with each event *i* a *Lifetime Distribution* based on which $v_i$ is generated

**Generalized Semi-Markov Process (GSMP)**

• In a simulator, $v_i$ is generated through a pseudorandom number generator
• Events may be CONTROLLABLE or UNCONTROLLABLE

• *Supervisory control*:

  ENABLE/DISABLE controllable events so as to meet desired specifications (avoid deadlock, illegal states, etc.)

In the simplest case, all events are assumed *observable*. 
SUPERVISORY CONTROL -- EXAMPLE

MACH1 → BUFFER → MACH2

CONTROL $v_1$

CONTROL $u_1$

CONTROL $v_2$

CONTROL $u_2$

Idle → Work

Down → Work → Down

1 → 0

$r_1$, $s_1$, $f_1$, $d_1$, $r_2$, $s_2$, $f_2$, $d_2$
• MACH1 can only start when BUFFER is **empty**.
• MACH2 can only start when BUFFER is **full**.
• MACH1 cannot start when MACH2 is **down**.
• If both MACH1 and MACH2 are **down**, then MACH2 is repaired first.
From a SUPERVISOR perspective, model is reduced to 6 states!
THE "QUOTIENT" SUPERVISOR
CONVENTIONAL SIMULATION ANALYSIS

- Repeatedly change parameters/operating policies
- Test different conditions
- Answer multiple WHAT IF questions
WHAT IF...

- Parameter $p_1 = a$ were replaced by $p_1 = b$
- Parameter $p_2 = c$ were replaced by $p_2 = d$

Performance Measures under all WHAT IF Questions

ANSWERS TO MULTIPLE “WHAT IF” QUESTIONS AUTOMATICALLY PROVIDED

Christos G. Cassandras
- Boston University

CODES Lab.
TYPICAL OPTIMIZATION OF COMPLEX SYSTEMS: REPEATED TRIAL AND ERROR

1. SEARCH SPACE
2. SIMULATE SYSTEM
3. ESTIMATE PERFORMANCE
4. CHOOSE NEW POINT
5. SIMULATE SYSTEM
6. ESTIMATE PERFORMANCE
7. CHOOSE NEW POINT
8. etc…etc…etc
CONCURRENT SIMULATION FOR OPTIMIZATION

Christos G. Cassandras
CODES Lab.
- Boston University

SEARCH SPACE

SIMULATE SYSTEM

CONCURRENT SIMULATION

ESTIMATE PERFORMANCE

CHOOSE NEW POINT

RAPID LEARNING
THE CONSTRUCTABILITY PROBLEM

\[ \omega \]
\[ u_1 \to \text{DES} \to \{ e_k^1, t_k^1 \} \]
\[ u_2 \to \text{DES} \to \{ e_k^2, t_k^2 \} \]
\[ \vdots \]
\[ u_m \to \text{DES} \to \{ e_k^m, t_k^m \} \]

OBSERVED

CONSTRUCTED
CONSTRUCTABILITY: AN EXAMPLE

PROBLEM:
- How does the system behave under different choices of $K$?
- What is the optimal $K$?

Reject if Workcenter content $> K$
CONSTRUCTABILITY: AN EXAMPLE

CONTINUED

CONCURRENT SIMULATION APPROACH:

- Choose any \( K \)
- Simulate (or observe actual system) under \( K \)
- Apply Concurrent Simulation to \textit{LEARN} effect of all other feasible \( K \)

Reject if Workcenter content > \( K \)
CONSTRUCTABILITY: AN EXAMPLE

NOMINAL SYSTEM: \( K = 3 \)

PERTURBED SYSTEM: \( K = 2 \)

AUGMENTED SYSTEM
However, if roles of NOMINAL and PERTURBED are reversed, then things get a little trickier…

\[ \Gamma(0) = \{a\} \subset \Gamma(1) = \{a, d\} \]