Discrete Event Systems Modelling and Supervisory Control

Abhishek Das, Minali Gupta and Dr. S.K. Nagar

Department of Electrical Engineering,
Indian Institute of Technology (BHU), Varanasi, UP, India.

Abstract

Discrete-event models are generally used to describe systems where coordination and control is required to ensure the orderly flow of events, or to avoid the occurrence of certain chains of events. Reduced Supervision and Hierarchical Supervision are methods of obtaining a supervisor in which control is divided or we can say that the control is distributed. In this paper the Problem of Guide Way, small factory problem and Big factory problem are taken as examples to explain the procedure to obtain supervisory controller using DES by TCT software. It is shown that the specifications are controllable and non-blocking.

Keywords: Discrete Event Systems, Decentralized Supervision Finite Automata, Supervisory Control, Hierarchical Control, TCT software.

1. Introduction

“A Discrete Event System is a dynamic system that evolves in accordance with the abrupt occurrence, at possibly unknown intervals, of physical events [1]. DESs arise in the domains of manufacturing systems, robotics, communication systems, logistics etc., and presently applying the theory of DES in the fields of power system network problems has become an area of active and growing research. The theory was initiated and developed by P.J.Ramadge and W.M.Wonham and subsequently been extended by other researchers.

Supervisory control of DES is an effective analytical tool for automation and control of DES [2]. The supervisor gives the list of events that must be enabled or disabled at a particular state such that the plant behaviour meets the required specification.
Hierarchical structure can be described as a division of control action and the concomitant information processing according to the scope [5]. Hierarchical consistency is the main issue that is focused for achieving a hierarchical control.

In decentralized modular synthesis [5] the overall supervisory task is divided into two or more than two subtasks. Each of the latter is solved using the results of the commands of TCT Software, and the resulting individual sub supervisors are run concurrently to implement a solution of the original problem.

2. Supervisory Control (Automation)

The supervisory control problem for a discrete event system is formulated by modelling the plant as well as its control logic (specifications) as some DES. To solve the supervisory control problem, it is necessary to show that a controller which forces the specification to be met exists.

2.1 Modelling of DES [1]

The plant model is obtained as a DES which specifies the set of states (including initial and marker states which can be desired states in many applications), the set of events, and the transition function of the system. Formally, a DES is represented by an automaton \(G=(Q, \Sigma, \delta, q_0, Q_m)\) in which \(Q\) is a finite set of states viz: \(Q=\{q_0, q_1, q_2, q_3, q_4, q_5\}\) is shown in Fig. 1. with \(q_0 \in Q\) as the initial state and \(Q_m \subseteq Q\) being the marker (desired) states. \(\Sigma\) is a finite set of event (\(\sigma\)) which is referred to as an alphabet viz: \(\Sigma=\{a, b\}\) is shown in Fig. 1. and finally \(\delta(q, \sigma)\) is a transition mapping \(\delta: Q \times \Sigma \rightarrow Q\) is shown in Fig. 2. which gives the next state after the occurrence of an event(\(\sigma\))

\[\text{Fig. 1: Automata Diagram.}\]

\(G\) plays the role of the plant and together with its states, events and transition operator (mapping) model a physical process. The set \(\Sigma^*\) represents all the strings which can be found with plant's event set \(\Sigma\). \(G\) is called generator as it generates a set of language \(L(G)\). \(L(G)=\{s \mid s \in \Sigma^*, \delta(q_0, su)\text{ is defined}\}\). The set of string which lead to the marker stage is called marked language \(L_m(G)\). \(L_m(G)=\{s \mid s \in L(G), \exists u \in \Sigma^*, \delta(q_0, su) \in Q_m\}\) where \(u\) is a prefix of \(s\) which is an initial event sequence of \(s\), i.e. if \(r\) and \(s\) are any strings in \(\Sigma^*\) then \(u\) is a prefix of \(s\). The mathematical description of the plant can be represented as an FSM (Finite State Machine). The generator \(G\) is deterministic in the sense that if an event occurs the transition leads the system to
unique system state. Here we are dealing with the deterministic systems only. A discrete-event system is said to be non-blocking if $Lm(G) = L(G)$. It means that there always exists a sequence of events which takes the plant from any state to marker state. Shuffle product is the procedure to combine two independent DESs into a single new process described by a new automaton $G_3 = G_1 \| G_2$ and is required when it is necessary to consider several independent and asynchronous process simultaneously. The synchronization process defines new states for $G_3$ as ordered pairs of $G_1$ and $G_2$. The initial and marker states of $G_3$ are defined simultaneously [3].

2.2 Non-blocking Supervisor and controllable Specifications

A discrete-event plant must be controlled based on some specifications (requirement behavior logic). By adjoining controller structure to the plant, it is possible to vary the language generated by the closed loop system within certain limits. The desired performance of such a controlled plant will be specified by stating its generated language must be in some specification language. It is often possible to meet the specifications in a maximally permissive way.

The control of discrete-event systems is performed by enabling and disabling of events in the plant. The supervisor controls the behavior of the plant by enabling and disabling events, hence affecting the actual event sequences. Any string of events which occurred in the system is the input to the supervisor and its output is an enable/disable map.

Suppose $G=\langle Q, \Sigma, \delta, q_0, Q_m \rangle$, be nonempty DES representing the plant which must be controlled. $\Sigma = \Sigma_c \cup \Sigma_u$ where $\Sigma_c$ denotes the set of controllable events and $\Sigma_u$ is the set of uncontrollable events. A possible set of events which includes some controllable events and all uncontrollable events is called control pattern $\gamma$. Set of all control patterns, which is a set of sets, is defined as $\Gamma = \{ \gamma \in \mathcal{Pwr}(\Sigma) \mid \gamma \supseteq \Sigma_u \}$. A supervisory control for the plant $G$ is any function $V : L(G) \rightarrow \Gamma$. The pair $(G, V)$ is written $V/G$ to suggest "$G$ under supervision of $V$". The supervisor which has been designed first determines in which state the system is working and then sends a list of events $\Gamma$ which must be disabled in that particular state, as a control signal to the plant. It is a DES synthesized using specifications in such a way that guarantees the required behavior of the plant.

The marked behavior of $V/G$ is $Lm(V/G) = L(V/G) \cap L_m(G)$ i.e. the strings terminated to marker states in $V/G$ are exactly the strings of $L_m(G)$ that survived under supervision of $V$. The supervisor $V$ is non-blocking for $G$ if $\overline{Lm(G)} = L(G)$. The supervisory control of a discrete-event system enforces the controllable and non-blocking behavior of the plant that is admissible by the given specification. The optimal solution to the supervisory control problem is the supremal controllable sublanguage (of the specification language). The DES representing the supremal supervisor typically has a large state size. Its state size is of order the product of state sizes of the plant and specification (plant control logic) DES models. The state size of the supremal supervisor can be reduced without affecting controlled behavior of the closed-loop system [4].
3. Hierarchical Control of DES

In this type of control a modular structure is obtained by defining the effectiveness of the control depending upon the hierarchy of the controller. This reduces complexity and provides a focused controller action.

Hierarchical control strategy attempts to structure a complex DES problem through decomposing it into smaller sub problems and reassembling their solutions into hierarchical levels. In such a pyramidal hierarchical structure, a system is modeled as a multi-level hierarchy in which high-level dynamics are obtained via abstraction from the low-level models, and command and information channels connect high-levels and low-levels. A two level hierarchy consisting of a High Manager, High World, Low Operator, Low World and the flow of command, control, report, plan and advise are shown in Fig. 3.

4. Illustrative Examples

4.1 Problem of Guide Way

Station X and Station Y are connected by a single one-way track from X to Y. Track consists of four sections, with spotlight (*) and detectors (!) installed at various section junctions. Two vehicles (ex- two trains) T1,T2 use the guide way simultaneously. Four detectors are also installed in this guide way to observe the movement of the vehicles (trains). The track is divided into four parts: P1, P2, P3, P4.

<table>
<thead>
<tr>
<th>δ</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>q1</td>
<td>q5</td>
</tr>
<tr>
<td>q1</td>
<td>q5</td>
<td>q2</td>
</tr>
<tr>
<td>q2</td>
<td>q2</td>
<td>q3</td>
</tr>
<tr>
<td>q3</td>
<td>q4</td>
<td>q5</td>
</tr>
<tr>
<td>q4</td>
<td>q5</td>
<td>q5</td>
</tr>
<tr>
<td>q5</td>
<td>q5</td>
<td>q5</td>
</tr>
</tbody>
</table>

Fig. 2: Transition Table.
4.1.1 Control objectives
The main aim of our control objective is to prevent two vehicles from colliding.

4.1.1.1 Control specification
- Control of spotlights must ensure that T1 and T2 never travel on the same section of the track simultaneously.
- This means that C’s are subject to mutual exclusion of the state pairs (i,i), i=1,2,3,4.
- Controllable events are odd numbered: the unobservable events are {13, 23}. The controllable events are the events which are in our hand which means that these events can be disabled when required.

4.1.2 DES modes for guide way

Fig. 5: DES model of Guide Way.

4.1.3 Steps required constructing supervisor using TCT package
T1= Create (T1, [mark 5], [tran [0,11,1], [1,13,2], [2,10,3], [3,15,4], [4,12,5]]) (6, 5)
T2= Create (T2, [mark 5], [tran [0,21,1], [1,23,2], [2,20,3], [3,25,4], [4,22,5]]) (6,5)
T= Sync (T1, T2) (36, 60)
Blocked_events= None
E= Mutex (T1, T2, [[1,1], [2,2], [3,3], [4,4]]) (30,40)
N= Supnorm (E, T, Null [13,23]) (26,32)
NO= Project (N, Null [13,23]) (20, 24)
VO= Project (T, Null [13,23]) (25,40)
KO= Supcon (VO, NO) (20, 24)
KODAT= Condat (VO, KO) Controllable
PIKO= Selfloop(KO, [13, 23]) (20, 64)
True= Nonconflict(T, PIKO)
K= Meet (T, PIKO) (26,32)

The supervisor action of KO can be read from the tabulated transition structure or from the transition graph and is the following (where tsi stands for ’track section i’): if T2 starts first (event 21), it must enter ts4 before T1 may start (event 11: disabled by light #1). T1 may then continue into ts3 (event 10), but may not enter ts4 (event 15: disabled by light #3) until T2 enters station B (event 22). Light #2 is not used. In fact, switching light #2 to red would mean disabling event 13 or 23; but these events are
unobservable, while KO is normal. If all events are observable, supervision could be based on \( E \), allowing \( T_1 \) to start when \( T_2 \) has entered \( t_{s2} \). But then \( T_1 \) must halt at light \#2 until \( T_2 \) has entered \( t_{s4} \).

### 4.2 Small Factory Problem

\[
\begin{array}{c|c}
\beta_1=10 & \beta_2=20 \\
\alpha_1=11 & \alpha_2=21 \\
\lambda_1=12 & \lambda_2=22 \\
\mu_1=13 & \mu_2=23 \\
\end{array}
\]

Fig. 6(a): MACHi

Fig. 6(b): MACHi Spec.

Small factory operates as follows [3] as shown in Fig. 5(a) and the event specification (Spec) as shown in Fig. 5(b). Initially the buffer is empty. With the event \( \alpha_1=11 \)MACH1 takes a work piece from an infinite input bin and enter W. subsequently MACH1 either breaks down and enters D(event \( \lambda_1=12 \)), or successfully completes its work cycle, deposits the work piece in the buffer, and returns to I(event \( \beta_1=10 \)). MACH2 operates similarly, but takes its work piece from the buffer and deposits it when finished in an output bin. If a machine breaks down, then on repair it returns to I (event \( \mu \))

The informal specifications for admissible operation as follow:
1. The buffer must not overflow or underflow
2. If both machines are broken down, then MACH2 must be repaired before MACH1

Specifications are shown in Fig. 7.

4.2.1 Small Factory problem

\[
\text{MACH1} = \text{Create(MACH1,[mark 0],[tran [0,11,1],[1,10,0],[1,12,2],[2,13,0]])} (3,4)
\]
MACH2=Create(MACH2,[mark 0],[tran [0,21,1],[1,20,0],[1,22,2],[2,23,0]]) (3,4)
FACT=Sync(MACH1,MACH2) (9,24)
BUFSPEC=Create(BUFSPEC,[mark 0],[tran [0,10,1],[1,21,0]]) (2,2)
BUFSPEC1=Selfloop(BUFSPEC,[11,12,13,20,22,23]) (2,14)
BRSPEC=Create(BRSPEC,[mark 0],[tran [0,13,0],[0,22,1],[1,23,0]]) (2,3)
BRSPEC1=Selfloop(BRSPEC,[10,11,12,20,21]) (2,13)
SPEC=Meet(BUFSPEC1,BRSPEC1) (4,22)
FACTSUP=Supcon(FACT,SPEC) (12,24)
CONSUP=Condat(FACT,FACTSUP) Controllable.
Control data are displayed as a list of supervisor states where disabling occurs, together with the events that must be disabled there.

**Table 1: Control Data.**

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0: 21</td>
<td>1: 21</td>
<td>8: 11</td>
<td>10: 13</td>
</tr>
<tr>
<td>2: 11</td>
<td>3: 21</td>
<td>11: 11</td>
<td></td>
</tr>
</tbody>
</table>

FACTSR=Supreduce(FACT,FACTSUP,CONSUP)
FACTSRC=Condat(FACT,FACTSR) Controllable.
Control data are displayed as a list of supervisor states where disabling occurs, together with the events that must be disabled there.

**Table 2: Control Data.**

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0: 21</td>
<td>1: 11</td>
<td>2: 13</td>
</tr>
</tbody>
</table>

**Fig. 10:** Reduced order supervisory control for SMALL FACTORY problem.

### 4.3 Decentralized Supervision: Big Factory

The big factory is as described below. The machines operate in parallel to feed a buffer with capacity 3; a third machine empties buffer.

**Fig. 11:** Block diagram for Big Factory.
The plant models are drawn which consists of various machines i.e. MACHI1, MACHI2 and MACHI3.

4.3.1 DES Modelling
A Plant DES model Big Factory consists of components machine1 (MACHI1), machine2 (MACHI2) and machine 3 (MACHI3) and buffer specification BUFE3. Each component is modeled as DES. Then DES models of plant components are synchronized to form the plant model.

4.3.2 Labeling of Events

<table>
<thead>
<tr>
<th>Events</th>
<th>Labels</th>
</tr>
</thead>
<tbody>
<tr>
<td>MACHI1 takes a work piece and hence enters working state</td>
<td>11</td>
</tr>
<tr>
<td>If MACHI1 completes work cycle and return to I</td>
<td>10</td>
</tr>
<tr>
<td>MACHI2 takes a work piece and enters working state</td>
<td>21</td>
</tr>
<tr>
<td>MACHI2 completes work cycle and return to I</td>
<td>20</td>
</tr>
<tr>
<td>MACHI1 breaks down and enter D</td>
<td>12</td>
</tr>
<tr>
<td>MACHI2 breaks down and enter D</td>
<td>22</td>
</tr>
<tr>
<td>MACHI3 takes a work piece and enters working state</td>
<td>31</td>
</tr>
<tr>
<td>MACHI3 completes work cycle and return to I</td>
<td>30</td>
</tr>
<tr>
<td>When MACHI3 breaks down, it enter D</td>
<td>32</td>
</tr>
<tr>
<td>MACHI1 repaired and returns to ideal state I</td>
<td>13</td>
</tr>
<tr>
<td>MACHI2 repaired and returns to ideal state I</td>
<td>23</td>
</tr>
<tr>
<td>When MACHI3 is repaired, it returns to ideal state</td>
<td>33</td>
</tr>
</tbody>
</table>

4.3.3 The informal specification
1. Buffer must not overflow or underflow
2. MACHI1 and MACHI2 are repaired on order of breakdown
3. MACHI3 has priority of repair over MACHI1 and MACHI2

To formalize specifications, we construct a DES as shown -
It should be noted that MACHI3 has always the priority of repair over MACHI1 and MACHI2. So whenever all machines broke down MACHI3 must be repaired before the other two. This has been shown in the specification transition diagram of BR3.

4.3.4 DES Simulation for Big Factory (TCT)
The Monolithic supervision of Big Factory is as designed,

\[
\text{MACHI1} = \text{Create}(\text{MACHI1}, [\text{mark } 0], [\text{tran } [0, 11, 1], [1, 10, 0], [1, 12, 2], [2, 13, 0]]) (3, 4)
\]

\[
\text{MACHI2} = \text{Create}(\text{MACHI2}, [\text{mark } 0], [\text{tran } [0, 21, 1], [1, 20, 0], [1, 22, 2], [2, 23, 0]]) (3, 4)
\]

\[
\text{MACHI3} = \text{Create}(\text{MACHI3}, [\text{mark } 0], [\text{tran } [0, 31, 1], [1, 30, 0], [1, 32, 2], [2, 33, 0]]) (3, 4)
\]

The synchronous reachable product of MACHI1 and MACHI2

\[
\text{MACHI12} = \text{Sync}(\text{MACHI1}, \text{MACHI2}) (9, 24)
\]

The reachable synchronous product of all machines is given by

\[
\text{BFACT} = \text{Sync}(\text{MACHI12}, \text{MACHI3}) (27, 108)
\]

\[
\text{BUF3} = \text{Create}(\text{BUF3}, [\text{mark } 0], [\text{tran } [0, 10, 1], [0, 20, 1], [1, 10, 2], [1, 20, 2], [1, 31, 0], [2, 10, 3], [2, 20, 3], [2, 31, 1], [3, 31, 2]]) (4, 9)
\]

\[
\text{BUF3SL} = \text{Selfloop}(\text{BUF3}, [11, 12, 13, 21, 22, 23, 30, 32, 33]) (4, 45)
\]

\[
\text{BR12} = \text{Create}(\text{BR12}, [\text{mark } 0], [\text{tran } [0, 12, 1], [0, 22, 1], [1, 12, 2], [1, 13, 0], [1, 22, 3], [1, 23, 0], [2, 23, 1], [3, 13, 1]]) (4, 8)
\]

\[
\text{BR12SL} = \text{Selfloop}(\text{BR12}, [10, 11, 20, 21, 30, 31, 32, 33]) (4, 40)
\]

\[
\text{BR3} = \text{Create}(\text{BR3}, [\text{mark } 0], [\text{tran } [0, 13, 0], [0, 23, 0], [0, 32, 1], [1, 33, 0]]) (2, 4)
\]

\[
\text{BR3SL} = \text{Selfloop}(\text{BR3}, [10, 11, 12, 20, 21, 22, 30, 31]) (2, 20)
\]

Combining the specification language into their intersection, we define BSPECM

\[
\text{BUBRSL} = \text{Meet}(\text{BUF3SL}, \text{BR12SL}) (16, 148)
\]
BSPECM=Meet(BUBRSL,BR3SL) (32,248)
The monolithic supervisor is obtained as, BFACTSUP= Supcon (BFACT, BSPECM) (96,302)
BFACTCON=Condat(BFACT,BFACTSUP) Controllable. (CONTROL DATA)

Table 4: Control Data.

|--------------|------|------|------|------|------|------|-------|-------|

The BFACTSUP obtained is optimal.

5. Results and Conclusions
The supervisory controllers using DES theory for developing reduced supervisory control theory problem for problem of guide way and decentralized supervision of Big Factory are obtained by TCT software. The reduced supervision has been obtained for the various problems.

References