Supervisor specification and synthesis for discrete event systems

F. LIN†, A. F. VAZ† and W. M. WONHAM†

In this paper, a method for the specification of a supervisor for a discrete event system in terms of a legal constraint language which is defined in terms of illegal states and event sequences is proposed. This is a natural manner for the specification of mutual exclusion and synchronization requirements. A procedure for the synthesis of a supervisor which enforces the legal constraint language is also discussed.

1. Introduction

Discrete event systems (DES) are asynchronous state machines that are used to model a variety of situations, such as computer operating systems, electrical power systems, air traffic control, and flexible manufacturing, to name a few. In each of the aforementioned situations, spontaneous events take the system from one state to another. The control problem concerns itself with the enabling and disabling of events in a systematic manner so that the resulting behaviour is acceptable. The automaton which effects this control is termed the supervisor. The supervisor cannot instruct that an event occur; instead it inhibits undesired events. An example of this is a computer and its user. The computer does not command the user to type a sequence of instructions; instead, it inhibits the user's access to certain data sets without the appropriate passwords being given.

When the DES is used to model several interacting processes, two control problems arise: mutual exclusion and synchronization. Mutual exclusion concerns itself with allowing only one process at a time to use a certain resource. Synchronization concerns itself with the coordination of events in the interacting processes. An example of this is a producer–consumer relationship between processes. The consumer must be blocked until a resource has been produced by the producer. Then the producer must be blocked until the consumer has consumed the resource. The cycle can then repeat ad infinitum. Another example is the fair usage of a resource among competing processes. In this case, the supervisor must allow the processes to use the resource in the order of their requests.

The requirements of mutual exclusion and synchronization necessitate a specific supervisor structure. The supervisor must contain some model of the dynamics that it is to enforce. This is termed the quotient structure property (cf. Ramadge and Wonham 1987 and Vaz and Wonham 1986). Mutual exclusion entails the avoidance of states in this model where more than one process would be simultaneously using a resource. Synchronization involves remembering a sequence of events, and avoiding a later invalid sequence which is determined with reference to the former. Since a supervisor is a state machine, the remembering of strings is accomplished by augmenting the process dynamics with additional states (cf. Lin 1984). The supervisor
can then be constructed from such an enhanced process model with an appropriate set of state dependent controls.

This paper is organized in the following manner. In § 2, some results in supervisory control theory are reviewed. The legal constraint language used to specify desired behaviour is detailed in § 3. A procedure for the construction of a generator for the legal constraint language and its optimal approximation are discussed in § 4. The procedures are illustrated in § 5 with a simple example.

2. Preliminaries

Modelling and control of DES have been discussed by many authors (see references). The present paper is set in the framework of Ramadge and Wonham (1987) and related articles (see references). The reader is referred to these sources for certain background material that will be taken for granted in the discussion to follow.

Abstractly, an uncontrolled DES can be taken to be a 5-tuple

\[ G = (\Sigma, Q, \delta, q_0, Q_m) \]

where \( \Sigma = \{\alpha, \beta, \ldots\} \) is a set of event labels, \( Q = \{q, q', \ldots\} \) is a set of states, and \( \delta : \Sigma \times Q \rightarrow Q \), the transition function, is a partial function (defined at each \( q \in Q \) for a subset of the \( \sigma \in \Sigma \)). Events are state transitions \( q \xrightarrow{\sigma} q' \) where \( \delta(\sigma, q) = q' \). The initial state is \( q_0 \in Q \), and the subset of marked states is \( Q_m \subseteq Q \). The latter serve to 'mark' the termination of 'tasks' or sequences of tasks of the underlying physical process that G is intended to model. The language \( L(G) \) generated by G consists of the empty string \( \epsilon \) together with all strings obtained by finite iteration of \( \delta \) starting from \( q_0 \). Let \( \Sigma^* \) denote the set of all finite strings over \( \Sigma \), plus 1. Then \( \delta : \Sigma^* \times Q \rightarrow Q \) is extended in the obvious way to \( \delta : \Sigma^* \times Q \rightarrow Q, \) where \( \delta(1, q) := q \); and then the language generated by G is

\[ L(G) := \{s \in \Sigma^* | \delta(s, q_0) \text{ is defined}\} \]

The strings marked by G make up the sublanguage defined by

\[ L_m(G) := \{s \in \Sigma^* | \delta(s, q_0) \in Q_m \} \]

In general, a language over \( \Sigma \) is any subset of \( \Sigma^* \). The closure \( \overline{K} \) of a language \( K \subseteq \Sigma^* \) is the set of all prefixes (initial segments) of strings in \( K \); and \( K \) is closed if \( K = \overline{K} \). If \( K = \emptyset \) then \( \overline{K} = \emptyset \); otherwise, the empty string \( \epsilon \in \overline{K} \). Clearly \( L(G) \) is closed, but \( L_m(G) \) need not be. We shall always make the natural assumptions that \( L(G) \neq \emptyset \) and that \( L_m(G) = L(G) \).

A control feature is incorporated by the further assumption that the set of event labels is partitioned as

\[ \Sigma = \Sigma_{ue} \cup \Sigma_c \]

The events labelled by elements of \( \Sigma_{ue} \) are uncontrollable, that is, cannot be prevented from occurring when the system G is in the appropriate state; whereas the events labelled by elements of \( \Sigma_c \) are controllable, that is, can be disabled or prevented from occurring, or alternatively enabled or permitted (but not forced) to occur, by an external controlling agent that we call a supervisor. It is evident that by repeatedly enabling or disabling each controllable event according to some rule based on the past behaviour of the DES, a supervisor S will create, or synthesize, a closed sublanguage \( L(S/G) \) of \( L(G) \). A closed sublanguage \( K \subseteq L(G) \) can be synthesized in
such a closed-loop control structure if and only if it is controllable with respect to \( L(G) \) and \( \Sigma_{uc} \) (cf. Ramadge and Wonham 1987), namely

\[
K\Sigma_{uc} \cap L(G) \subseteq K
\]

In general a language \( H \), not necessarily closed, is controllable with respect to \( L(G) \) and \( \Sigma_{uc} \) if \( H \subseteq L(G) \) and

\[
H\Sigma_{uc} \cap L(G) \subseteq H
\]

An arbitrary sublanguage \( E \) of \( L(G) \) may be optimally approximated by the supremal (in the sense of subset inclusion) controllable sublanguage of \( E \), denoted by \( \text{sup } C(E) \). A supervisor \( S \) can then be constructed such that

\[
L(S/G) = \text{sup } C(E)
\]

Therefore, a crucial issue in supervisor synthesis is to specify and compute (a generator for) \( E \) and \( \text{sup } C(E) \).

3. Legal languages

Consider a DES given by \( G = (\Sigma, Q, \delta, q_0, Q_m) \) and a set of forbidden strings \( F \) and a set of forbidden states \( Q_f \). Our objective is to synthesize a supervisor \( S \) so that the resulting SDES \( S/G \) neither generates forbidden strings nor visits forbidden states. This desired behaviour is defined in terms of a legal constraint language \( E \), defined as

\[
E = L(G) - \Sigma^* F \Sigma^* - \langle Q_f \rangle
\]

where

\[
\Sigma^* F \Sigma^* := \{ s : (\exists f \in F)(\exists u, v \in \Sigma^*) s = uvf \}
\]

\[
\langle Q_f \rangle := \{ s : (\exists u, v \in \Sigma^*) s = uw \land \delta(u, q_0) \in Q_f \}
\]

The language \( E \) is defined by removing forbidden words in \( L(G) \). As in jurisprudence, legal behaviour is defined in terms of illegal actions: what is not expressly forbidden is permitted.

In this section, we detail several properties of the language \( E \). The supremal controllable sublanguage of \( E \), \( \text{sup } C(E) \), is found to be characterized in a simple manner if \( L(G) \) satisfies a technical condition, which we define later.

Before stating our results, we make the following definitions. For \( Q_f \subseteq Q \) and \( Q'_f \subseteq Q \), the languages \( E_1 \) and \( E'_1 \) are given by

\[
E_1 := L(G) - \langle Q'_f \rangle \quad \text{and} \quad E'_1 := L(G) - \langle Q_f \rangle
\]

For a string \( f = \sigma_1 \sigma_2 \ldots \sigma_k \in \Sigma^* \), we say that \( \sigma_1 \sigma_2 \ldots \sigma_j \), with \( j \leq k \), is a prefix of \( f \); let \( \text{pre} \ (f) \) denote the set of all such prefixes. By definition, \( 1 \in \text{pre} \ (f) \). Let \( C \text{ pre} \ (f) \) denote the longest prefix of a string \( f \) which has a controllable last symbol; in the case where \( f \in \Sigma_{uc}^* \), \( C \text{ pre} \ (f) = 1 \).

Similarly, for a string \( f = \sigma_1 \sigma_2 \ldots \sigma_k \in \Sigma^* \), we say that \( \sigma_j \sigma_{j+1} \ldots \sigma_k \), with \( j \leq k \), is a suffix of \( f \); let \( \text{suf} \ (f) \) denote the set of all such suffixes. By definition, \( 1 \in \text{suf} \ (f) \).

For two sets of forbidden strings \( F \) and \( F' \), define the languages \( E_2 \) and \( E'_2 \) by

\[
E_2 = L(G) - \Sigma^* F \Sigma^* \quad \text{and} \quad E'_2 = L(G) - \Sigma^* F' \Sigma^*
\]

We now have the following results.
Property 1 (conjunctive characterization)

\[ E = E_1 \cap E_2 \]

Proof

The result follows from a simple set operation.

Property 1 shows that the language \( E \) can be characterized in terms of the languages \( E_1 \) and \( E_2 \). Hence, we first consider the language \( E_1 = L(G) - \langle Q_t \rangle \).

Property 2 (containment)

If \( Q_t \subseteq Q'_t \), then \( E'_1 \subseteq E_1 \).

Proof

The result follows from the definitions.

Property 3 (controllability characterization)

\( E_1 \) is controllable if and only if \((\forall q \in Q - Q_t)(\forall q' \in Q_t)(\forall \sigma \in \Sigma)\delta(q, \sigma) = q' \) implies \( \sigma \in \Sigma_c \).

Proof

If \( E_1 \) is not controllable, namely

\[(\exists s \in E_1)(\exists \sigma \in \Sigma_{uc})s\sigma \in L(G) - E_1\]

then the states \( q = \delta(q_0, s) \) and \( q' = \delta(q, \sigma) \) are defined. Clearly \( q \in Q - Q_t, q' \in Q_t \). Therefore,

\[(\exists q \in Q - Q_t)(\exists q' \in Q_t)(\exists \sigma \in \Sigma)\delta(q, \sigma) = q' \land \sigma \in \Sigma_{uc}\]

This contradicts the hypothesis. The converse is also obvious. \(\Box\)

Property 3 characterizes the controllability of \( E_1 \) in a simple manner: \( E_1 \) is controllable if and only if all the transitions from any 'good state' to any 'bad state' are controllable. This leads to the following simple characterization of the supremal controllable sublanguage of \( E_1 \).

Property 4 (supremal controllable sublanguage)

\[ \sup C(E_1) = L(G) - \langle Q'_t \rangle \]

where

\[ Q'_t := \{ q : (\exists t \in \Sigma_{uc}^*)\delta(t, q) \in Q_t \} \]

Proof

Because \( Q_t \subseteq Q'_t \), by Property 2,

\[ L(G) - \langle Q'_t \rangle \subseteq L(G) - \langle Q_t \rangle = E_1 \]
By Property 3,
\[ L(G) - \langle Q'_t \rangle \text{ is controllable} \]

Therefore,
\[ L(G) - \langle Q'_t \rangle \subseteq \sup C(E_1) \]

For the reverse inclusion, it is equivalent to prove
\[ \sup C(E_1) \subseteq L(G) \]  (1)

and
\[ \sup C(E_1) \cap \langle Q'_t \rangle = \emptyset \]  (2)

By the definition of \( E_1 \), (1) follows. Suppose (2) is false, namely, there exists \( s \in \Sigma^* \), such that
\[ s \in \sup C(E_1) \]  (3)

and
\[ s \in \langle Q'_t \rangle \]  (4)

By the definition of \( \langle Q'_t \rangle \), (4) implies
\[ (\exists u, v \in \Sigma^*) s = uv \land \delta(q_0, u) \in Q'_t \]

By the definition of \( Q'_t \),
\[ (\exists t \in \Sigma^*) \delta(q_0, ut) \in Q'_t \]

hence, \( ut \notin E_1 \). By closure, \( u \in \sup C(E_1) \). For \( t \in \Sigma^* \) and \( ut \in L(G) \), \( ut \in \sup C(E_1) \subseteq E_1 \) since \( \sup C(E_1) \) is controllable. This is a contradiction; hence (2) must hold. \( \square \)

We now examine the properties of \( E_2 = L(G) - \Sigma^*F\Sigma^* \), the language of 'allowed strings'.

**Property 5** (containment)
If \( (\forall f' \in F')(\exists f \in F) f' \in \text{pre}(f) \), then \( E'_2 \subseteq E_2 \).

**Proof**
The result follows from the definitions.

**Property 6** (controllability characterization)
If \( (\forall f \in F) f = C \text{pref}(f) \) then \( E_2 \) is controllable.

**Proof**
If \( E_2 \) is not controllable, namely
\[ (\exists s' \in E_2)(\exists \sigma \in \Sigma_{uc}) s' \sigma \in L(G) \land s' \sigma \notin E_2 \]

then \( s' \in E_2 \) and \( s' \sigma \notin E_2 \) imply there exists a suffix of \( s' \sigma \), denoted by \( s \sigma \), which belongs
to \( F \). Therefore
\[
(\exists s \in \Sigma^*)(\exists \sigma \in \Sigma)s\sigma \in F \land \sigma \in \Sigma_{\text{uc}}
\]
This contradicts the hypothesis. Therefore \( E_2 \) is controllable.

Essentially Property 6 says that if the last symbol of every forbidden string is controllable then \( E_2 \) is controllable. The converse does not necessarily hold unless \( L(G) \) is \( F \)-completable. We say that \( L(G) \) is \( F \)-completable (with respect to a set of prefixes \( F' \)) if
\[
(\forall f' \in F')(\forall s \in \Sigma^*)sf' \in L(G) = > sf \in L(G)
\]
where \( f' \in \text{pre}(f) \); namely, \( L(G) \) is \( F \)-completable (with respect to \( F' \)) if every word in \( L(G) \) that contains \( f' \in F' \) as a suffix can be completed to a word in \( L(G) \) that contains \( f \notin F \).

**Property 7** (characterization of supremal controllable sublanguage)

Let \( F' := \{ f' : (\exists f \in F) f' = C \text{ pre}(f) \} \). If \( L(G) \) is \( F \)-completable (with respect to \( F' \)), then
\[
\sup C(E_2) = L(G) - \Sigma^*F'S^*
\]

**Proof**

By Property 5,
\[
L(G) - \Sigma^*F'S^* \subseteq E_2
\]

By Property 6,
\[
L(G) - \Sigma^*F'S^* \text{ is controllable}
\]

Therefore,
\[
L(G) - \Sigma^*F'S^* \subseteq \sup C(E_2)
\]

For the reverse inclusion, it is equivalent to prove
\[
\sup C(E_2) \subseteq L(G)
\]
\[
\sup C(E_2) \cap \Sigma^*F'S^* = \emptyset
\]

By definition, (5) holds. Suppose (6) is false, namely, there exists \( s \in \Sigma^* \), such that
\[
s \in \sup C(E_2)
\]
and
\[
s \in \Sigma^*F'S^*
\]

By the definition of \( \Sigma^*F'S^* \), (8) implies
\[
(\exists u, v \in \Sigma^*)(\exists f' \in F')s = uf'v
\]
Let \( f \in F \) such that \( f' \in \text{pre}(f) \). By the definition of \( F' \), \( (\exists t \in \Sigma_{\text{uc}}) f = f't \). For such a \( t \), \( uf't \notin E_2 \). Since \( L(G) \) is \( F \)-completable, \( uf't \in L(G) \). This together with \( uf' \in \sup C(E_2) \), \( t \in \Sigma^* \) and \( \sup C(E_2) \) is controllable imply \( uf't \in \sup C(E_2) \subseteq E_2 \). This is a contradiction; hence (6) must hold.

\[\square\]
Now, because both $E_1$ and $E_2$ are closed, by Corollary 6.1 of Wonham and Ramadge (1988),

$$\sup C(E_1 \cap E_2) = \sup C(E_1) \cap \sup C(E_2)$$

Therefore, we have the following result.

**Proposition 1**

If $L(G)$ is $F$-completable, then

$$\sup C(E) = L(G) - \Sigma^*F^*\Sigma^* - \langle Q_f \rangle$$

where $F'$ and $Q_f$ are defined as in Property 4 and Property 7.

**Proof**

The result follows from Property 4, Property 7 and Corollary 6.1 of Wonham and Ramadge (1988).

The above proposition suggests an easy and direct way to compute $\sup C(E)$ when $L(G)$ is $F$-completable. We will take advantage of this fact in the procedure to synthesize a suitable supervisor.

4. **Computation of the legal languages**

To construct a generator for the legal language $E$, we first rewrite $E$ as:

$$E = L(G) - \Sigma^*F^*\Sigma^* - \langle Q_f \rangle = (\Sigma^* - \Sigma^*F^*\Sigma^*) \cap (L(G) - \langle Q_f \rangle)$$

Therefore, a generator for $E$ can be constructed in the following three steps:

- **Step 1.** Construct a generator $S_1$ of the language $L(G) - \langle Q_f \rangle$;
- **Step 2.** Construct a generator $S_2$ of the language $\Sigma^* - \Sigma^*F^*\Sigma^*$;
- **Step 3.** Construct a generator $S_3$ of the language $E$.

We now discuss each step.

**Step 1**

It is easy to show that $L(G) - \langle Q_f \rangle$ is marked by

$$S_1 = (\Sigma, Q - Q_f, \delta | (Q - Q_f), q_0, Q - Q_f)$$

where $\delta | (Q - Q_f)$ denotes $\delta$ restricted to $Q - Q_f$, namely,

$$\delta | (Q - Q_f) : \Sigma \times (Q - Q_f) \rightarrow (Q - Q_f)$$

with

$$\delta | (Q - Q_f)(\sigma, q) = \begin{cases} 
\delta(\sigma, q) & \text{if } \delta(\sigma, q) \in Q - Q_f \\
\text{undefined} & \text{otherwise}
\end{cases}$$

Thus $S_1$ is constructed by restricting the states and transition function on the generator $G$. 

Step 2

Without loss of generality, we assume no string in $F$ is a prefix of another string in $F$, namely

$$(\forall f_i, f_j \in F) f_i \notin \text{pre}(f_j) \forall i = j$$

(9)

Then $\Sigma^* - \Sigma^* F \Sigma^*$ is marked by

$$S_2 = (\Sigma, X - F, \xi | (X - F), x_0, X - F)$$

where

$$X := \cup_{f \in F} \text{pre}(f)$$

$$x_0 := 1$$

and $\xi | X - F$ is defined by the restriction of $\xi$ to $X - F$. The transition function $\xi : \Sigma \times X \rightarrow X$ is defined by $\xi(\sigma, s) := t$ where $t$ is the longest suffix of $s\sigma$ that belongs to $X$. Since $X$ contains the empty string, $\xi$ is well defined.

Proposition 2

$$L(S_2) = \Sigma^* - \Sigma^* F \Sigma^*$$

Proof

The result follows by induction on length of strings. For $s = 1$

$$1 \in \Sigma^* - \Sigma^* F \Sigma^*$$

$$\iff 1 \notin F$$

$$\iff 1 \in X - F$$

$$\iff x_0 \in X - F$$

$$\iff 1 \in L(S_2)$$

Suppose now for all $s$, $|s| = n$

$$s \in L(S_2) \iff s \in \Sigma^* - \Sigma^* F \Sigma^*$$

then for all $s\sigma$

$$s\sigma \in L(S_2)$$

$$\iff s \in L(S_2) \land \xi(s\sigma, x_0) \notin F$$

$$\iff s \in L(S_2) \land \text{suf}(s\sigma) \cap F = \emptyset$$

(because of (9))

$$\iff s\sigma \in \Sigma^* - \Sigma^* F \Sigma^*$$

Therefore

$$L(S_2) = \Sigma^* - \Sigma^* F \Sigma^*$$

The following procedure constructs $S_2$. 

\[ \square \]
Procedure Legal

\{compute generator \(S_2\) for the language \(\Sigma^* - \Sigma^*F\Sigma^*\}\}

begin
\(X := \cup_{f \in F} \text{pre}(f);\)
for all \(s \in X - F\) do
  for all \(\sigma \in \Sigma\) do
    begin
    \(t := 1;\)
    for all \(w \in X\) do
      begin
      if \(w \in \text{suf}(s\sigma)\) and \(|w| > |t|\) then
        \(t := w;\)
      end;
      if \(t \notin F\) then
        \(\xi(\sigma, s) := t\)
      else \(\xi(\sigma, s)\) is undefined
      end;
    \(x_0 := 1;\)
  end.
end.

Step 3

The generator \(S_3\) for \(E\) is the 'meet' of \(S_1\) and \(S_2\), namely
\(S_3 = S_1 \times S_2\)

\[= (\Sigma, (Q - Q_f) \times (X - F), (\delta|((Q - Q_f) \times (X - F)))(\sigma, q, x) = (\delta|((Q - Q_f)\sigma, q, \xi|((X - F)(\sigma, x))\]

if both \(\delta|((Q - Q_f)\sigma, q)\) and \(\xi|((X - F)(\sigma, x))\) are defined. Otherwise, it is undefined.

The above procedure yields a generator for \(E\) provided
\(L(S_1) \cap L(S_2) \neq \emptyset\)

In essence, this procedure enhances the 'good' portion of \(G\) with extra states which allow the supervisor to disable undesired strings. Standard procedures to compute the 'meet' can be found in Hopcroft and Ullman (1979) or Lin and Wonham (1985).

By Proposition 1, if \(L(G)\) is \(F\)-completable, then sup \(C(E) = L(G) - \Sigma^*F\Sigma^* - \langle Q_f \rangle\); hence we can compute sup \(C(E)\) by replacing \(Q_f\) with \(Q'_f\) and \(F\) with \(F'\) in the above three steps. If \(L(G)\) is not \(F\)-completable, an algorithm to compute sup \(C(E)\) can be found in Lin and Wonham (1985). When sup \(C(E)\) is computed, a supervisor \(S\) can be easily constructed based on the generator of sup \(C(E)\) such that \(L(S/G)\) = sup \(C(E)\) as shown in Ramadge and Wonham (1987). In fact, we can construct \(S = (S, \psi)\) as follows:

\(S = (\Sigma, Y, \eta, y_0, Y_m)\) is the generator of sup \(C(E)\)

and \(\psi\) is given by

\[\psi(\sigma, y) = \begin{cases} 1 & \text{if } \eta(\sigma, y) \text{ is defined} \\ 0 & \text{otherwise} \end{cases}\]
5. Example

Consider a system with two users and one resource (as shown in Fig. 1). Each user is modelled by $G_i$ as $(i = 1, 2)$:

![Figure 1]

where the event symbols stand for the following:
- $q_i$: user $i$ requests the resource
- $u_i$: user $i$ uses the resource
- $r_i$: user $i$ releases the resource

The controllable events are given by $\{u_1, u_2\}$. The uncontrolled system (Fig. 2) is modelled by the 'shuffle' $G = G_1 || G_2$. It is desired to impose two constraints on the

![Figure 2]
above dynamics. First, simultaneous use of the resource is forbidden. Second, the users should access the resource in the order of their requests. These constraints can be expressed using exclusion of states and strings in the following manner.

**Mutual exclusion**

Two users cannot use the resource simultaneously. The legal language is

$$E_1 = L(G) - \langle q_9 \rangle$$

The only events entering $q_9$ are $u_1$ and $u_2$ which are controllable. Hence $E_1$ is controllable.

**First-come first-served**

The resource is used by two users according to first-come first-served discipline. The corresponding legal language is

$$E_2 = L(G) - \Sigma^* \{q_1 q_2 u_2, q_1 r_2 q_2 u_2, q_2 q_1 u_1, q_2 r_1 q_1 u_2\} \Sigma^*$$

Since the last event of each of the above forbidden strings is controllable, $E_2$ is controllable.

**Supervisor**

Using the procedures of Section 3, the following supervisor results. $S$ is shown in Fig. 3 and $\psi$ is given in the Table.

```
<table>
<thead>
<tr>
<th>State</th>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_4$</th>
<th>$x_5$</th>
<th>$x_6$</th>
<th>$x_7$</th>
<th>$x_8$</th>
<th>$x_9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$u_2$</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
```
6. Conclusions

A simple method of specifying a legal language with mutual exclusion and synchronization constraints in terms of forbidden strings and states has been considered. The supremal controllable sublanguage for this legal language has been shown to have a simple form if the language $L(G)$ is $F$-completable. This specification and supervisor synthesis method have been illustrated with a simple example. The required supervisor is constructed by enhancing the process dynamics of the DES $G$ to be controlled with extra states and transitions. In this manner, the required specifications can be expressed as the avoidance of a set of 'bad' states in the enhanced model.

REFERENCES


