A Supervisor Design Procedure for Discrete Event Systems in A Temporal Logic Framework with Applications *

Dan Ionescu and Jing-Yue Lin
Department of Electrical Engineering, University of Ottawa, Ottawa, Ontario, Canada K1N 6N5

Abstract. The aim of this paper is to introduce a design procedure for the synthesis of supervisors for discrete event systems (DES). The processes taking place in a DES are modeled through temporal logic formulas, which consider an initial state, a triggering event and a labeling function. For the development of the synthesis procedure a composition rule of two discrete event systems described as two sets of formulas is deduced. Considering a plant model and the specification formulas for the closed loop systems, a procedure and algorithm for designing the supervisor is given. It is shown that the above technique is equivalent to DES verification. Some examples illustrate the validity of the supervisor synthesis approach.

Keywords. Discrete event systems; temporal logic models; reachability; validity; verification of discrete event systems; discrete event systems synthesis.

1 Introduction

Theoretical and practical procedures regarding the analysis and synthesis of discrete event systems (DES) have recently received a great deal of attention from the research community (Denham 1988; Inan and Varaiya 1988; Lin and Ionescu 1991a; Ramadge and Wonham 1987; Wonham 1988) activating in control, communications, computers and other fields. A set of approaches have been considered as a source for the development of abstract models and for the investigation of system properties. Among them one can mention the approaches originated from abstract automata theory, Petri nets, boolean algebra, temporal logic , etc. Temporal logic is a formalism that has been proposed in computer science by Manna, Pnueli (1983), and others (Clarke *et al.*, 1986; Galton, 1987; Manna and Wolper, 1984; Pnueli, 1981), for software verification, particularly for the analysis and synthesis of concurrent programs, operating systems, and distributed systems. Recently, it has been applied into the control problems of DESs by Fusaoka *et al.* (1983), Thistle and Wonham (1986), Lin and Ionescu (1990, 1991b), and so on. Real time systems have been specified in a real time temporal logic framework by Ostroff (1989).

However, there is little mention about a procedure through which having given a DES and a set of specifications one can obtain the set of rules describing a supervisor (controller) such that the specifications are fulfilled.

The motivation behind this paper is from the need of a

theoretically sound procedure through which, given a system described by a set of temporal logic formulas and its specifications one can determine another set of temporal logic formulas such that the closed loop system, obtained by combining the above two sets of formulas, satisfies the specifications.

This paper is organized as follows. The following section defines a temporal logic model (TLM) built as a set of temporal logic formulas.

In Section 3, reachability analysis of the TLM is carried out and a TLM is formulated as a $\Sigma-$algebra. A method for TLM composition is then developed. This will lead naturally in Section 4 to a synthesis procedure for a supervisor which is proven equivalent to the verification of discrete event systems.In Section 5, an example, concerning a communication protocol is shown. Finally, conclusions are given.

2 Discrete Event Dynamical Behavior and Temporal Logic Models

Dynamical behavior of DES consists of the tracing of occurrences of events as related to the system evolution, which at its turn sets conditions for future events.

In order to describe the dynamical behavior of DES, an event structure is defined as follows: An event structure V is defined by a 3-tuple $V = (\mathrm{S}, \mathrm{E}, f)$ where S is the set of states; E is the set of events, each event can be viewed as a transition from a state to another state; f is a mapping from $\mathrm{E} \times \mathrm{S}$ into S such that $\forall s \in \mathrm{S}, \forall e \in E_s, f(e,s)$ is defined, where E_s is the set of events which are fired in the state s.

Now,it is obvious that for any $s \in \mathrm{S}$, there is an event $e \in E_s$ given by $s' = f(e,s)$, if and only if e is enabled.

An occurrence of a transition in a DES must be described by a set of conditions which have to be true, in some time interval, in order for an action to take place in the system. Thus, a possible model for describing the dynamics of DES, can be built using temporal logic formulas.

Temporal logic is an extension of the ordinary logic to include the notion of time, providing five modal operators: $\Box$ (henceforth), $\Diamond$ (eventually), $\bigcirc$ (next), $\mathcal{U}$ (until) and $\mathcal{P}$ (precedes). Applying the above temporal operators on formulae, leads to new formulae. The rules of combining them in temporal logical formulae can be found for example in Ostroff(1989) or Galton (1987).

Let, therefore, F be a set of logic formulae, and let F^* denote the subsets of F. Then the following is the definition of a temporal logic model for DES.

A temporal logic model (TLM) is a 3-tuple $M = (V, \mathrm{F}^*, s_0, l)$ where $V = (\mathrm{S}, \mathrm{E}, f)$ is an event structure; F^* is the set of all subsets of F; s_0 is an initial state; and $l : \mathrm{E} \times \mathrm{S} \rightarrow \mathrm{F}^*$ is a labelling function, associating to every pair (e,s) the set of formulae that hold in (e,s).

With the labelling function l, a temporal logic model is described by a set of formulae. A generic dynamical formula $l(e,s)$ has the following form:

$$\Box[\delta = e \wedge x = R \Rightarrow (\bigcirc x) = Q]$$

which is related to the transition f by $s = x$, and $s' =$

$f(e,s) = \bigcirc x$.

The formulae are evaluated with respect to an interpretation. Thus, one can affirm that a state formula is any well-formed first order logic formula constructed over the variables and propositions, which can be evaluated over a single state to yield a truth value, and temporal formula is a formula constructed from state formulae to which one applies temporal operators.

From these observation one obtains that a DES can be represented by one or many TLMs.

A state of a TLM would then be given by:

$$\{[\vee_{1\leq i\leq N}(x_i = P)] \wedge [\wedge_{1\leq i<j\leq N}\neg(x_i = P \wedge x_j = P)]\}$$

where P is a value of state x, N is the dimension of the state, $x_1, x_2, \cdots, x_N$ are all the components of the state, and $\neg$ is the logical negation.

3 Reachability and Composability Properties of TLM

As mentioned before a DES evolution is triggered by firing an event $e \in E_{s_0}$ in a state s and a new state s' will occur. That is to say that, s' is immediately reachable from s. It means that one can immediately get to state s' from state s. The state s' will be the starting point for a new evolution (root) if another event $e' \in E_{s_0}$ will be fired, and so on. In this way the set of states reachable from s_0 is the set $R(M, s_0)$ defined by the following implication:

If $s \in R(M, s_0)$ and $s' = f(e, s)$ for some $e \in E_s$, then $s' \in R(M, s_0)$ where $s_0 \in R(M, s_0)$.

The same remark from above can be used to extend the next-state function to map a state and a sequence of events into a new state.

Based on the above, one can interpret a sequence of events $e_1e_2\cdots e_k$, and state s, the state $s' = f(e_1e_2\cdots e_k, s)$ as the result of firing first e_1, then e_2, and so on, until e_k is fired.

The sequence of reachable states allow now to describe the state trajectory of any DEDS. This trajectory can be expressed as transition graph (Lin and Ionescu 1993). There is a straightforward relationship between the reachability of states and the validity of formulas in a TLM. For a given TLM, the reachability of states is equivalent to the validity of the corresponding dynamical formulas. The statement above is based on s' being reachable from s if and only if $l(e, s)$, belongs to $\mathbf{F}^*$ and on the definition of validity.

A composition mechanism will be introduced based on homomorphism composition in abstract algebra. The structure of the above homomorphism will imitate the $\Sigma-$ homomorphism of a Σ-algebra developed by Henessy (1988). It will be called a $\Sigma-$ homomorphism of TLMs,

Considering, after Henessy,A a set, called the carrier, Σ a set of formal functional symbols, Σ_A a set of functions $\{f_A, f \in \Sigma\}$ such that if $arity(f) = n$ (then f_A is a function from $A^n \rightarrow A$, then using the definition of a $\Sigma-$algebra, and the relation defining the dynamical formula $l(e, s)$ (Section 2) the following result is obvious.

A TLM is a $\Sigma-$algebra $(\{\mathbf{S}, \mathbf{E}, \mathbf{F}^*\}, \{f, l, s_0\})$ with the carrier $\{\mathbf{S}, \mathbf{E}, \mathbf{F}^*\}$, the binary operations $f : \mathbf{E} \times \mathbf{S} \rightarrow \mathbf{S}$, and $l : \mathbf{E} \times \mathbf{S} \rightarrow \mathbf{F}^*$, and the nullary operation $s_0 \in \mathbf{S}$.(Lin

and Ionescu 1993).

Considering now two TLMs $M = (\{S, E, F^*\}, \{f, l, s_0\})$ and $M' = (\{S', E', F^{*'}\}, \{f', l', s_0'\})$ a $\Sigma-$homomorphism from M to M' is then given by a three functions: $g : E \rightarrow E'$, $k : F^* \rightarrow F^{*'}$, and $h : S \rightarrow S'$ such that $h(s_0) = s_0'$, and for all $e \in E$, $l'(g(e), h(s)) = k(l(e, s))$ and $f'(g(e), h(s)) = h(f(e, s))$.

Then, for a given $\Sigma-$ homomorphism of TLMs $(g, k, h) : M \rightarrow M'$ $t = f(e, s)$ in M, it follows that $h(t) = f'(g(e), h(s))$ in M' if $t = f(e, s)$ in M .

It is now clear that if s is a reachable state of M, then $h(s)$ is a reachable state of M'.

The composition operation of two or more DES, in terms of two TLMs viewed as $\Sigma-$algebras, can be then introduced:

For two TLMs M and M' defined as above the composition of M and M', denoted by $M \| M'$, is the concurrent operation of M and M' in which selected events, e.g. $e \in E$ and $e' \in E'$, are synchronized and form a synchronized event pair (e, e') such that, for all $s \in S$ and $s' \in S'$, $f((e, e'), s) \stackrel{\text{def}}{=} f(e, s)$ and $f'((e, e'), s') \stackrel{\text{def}}{=} f'(e', s')$, and $l((e, e'), s) \stackrel{\text{def}}{=} l(e, s)$ and $l'((e, e'), s') \stackrel{\text{def}}{=} l'(e', s')$.

Using the definition of projections of homomorphisms and that of a homomorphism one obtains a result which relates the reachability of states in the composite system and its components. A state s in $\bar{M} = M \| M'$ is reachable if and only if both $\Psi(s)$ is reachable in M and $\Psi'(s)$ is reachable in M' hold, where (Φ, Ψ) and (Φ', Ψ') are the projections from $\bar{M} = M \| M'$ to M and M' respectively, i.e. $\Phi : \bar{S} \rightarrow S$, $\Psi : \bar{E} \rightarrow E$, $\Phi' : \bar{S} \rightarrow S'$, and $\Psi' : \bar{E} \rightarrow E'$.

With these, a procedure for constructing a set of rules which will correct a given DES behavior upon some specified satisfactory behavior can be derived.

Let M denote the TLM of the given DES and M_s denote the TLM of the supervisory DES. The two systems operate concurrently and the control action is achieved by the synchronization of the selected events in the supervised DES with the events in the supervisory DES. In other words, some events of the given DES are prevented from occurring unless the state of the supervisory DES allows the corresponding synchronized event to occur.

Thus, for a given DES, whose temporal logic model M is known, a synthesis procedure for the supervisory DES is as follows.

A Synthesis Procedure for DES:

(i): Specify the required behavior of the closed-loop system $\overline{M}$ by temporal logic formulae;

(ii): Find the set of states which should not be reached by the closed-loop system according to (i). Denote the set by R_u;

(iii): Construct the M_s such that

$$R(M, s_0) \times R(M_c, q_0) = R(M, s_0) \times Q - R_u \qquad (5.1)$$

(iv) Synchronize the controlled events in the plant and in the controller; and compose the plant M and the controller M_c.

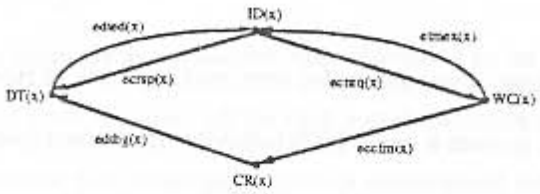

Figure 1: The diagram of the protocol.

4 An Example

As an application, we consider an example of the call processing of the packet layer protocol in the packet-switched data communication network. In this system, two packet layer protocol entities, for instance, transport end-points, call each other to implement the user services by exchanging packet protocol data units (Halsa:88). The call processing can be described by temporal logic models that specify how events cause changes of states and which event can occur at a particular state, as shown in Figure 1.

Let the local variable x represent the states of the protocol entities. The system has four states: $ID(x)$ which means that there is no connection established (idle); $WC(x)$ which stands for waiting for a call connected packet; $CR(x)$ which means connection established and ready for data transfers; $DT(x)$ which stands for data transmission.

There are six events in the system. At state $ID(x)$, there are two events which can occur: $ecnrq(x)$, the connection request; and $ecrsp(x)$, the call response. At state $WC(x)$, there are two events can occur: $eccfm(x)$, the connection confirmed; and $etmex(x)$, the call connected timer expires. At state $CR(x)$, one event can occur: $edtbg(x)$, the data transmission begins. At state $DT(x)$, one event can occur: $edted(x)$, the data transmission ends.

The set of formulas of the specifications for the system are given as follows. *Dynamics*

$$\Box[ecnrq(x) \wedge ID(x) \Rightarrow WC(\bigcirc x)] \qquad (7.6.P1)$$

$$\Box[ecrsp(x) \wedge ID(x) \Rightarrow DT(\bigcirc x)] \qquad (7.6.P2)$$

$$\Box[eccfm(x) \wedge WC(x) \Rightarrow CR(\bigcirc x)] \qquad (7.6.P3)$$

$$\Box[etmex(x) \wedge WC(x) \Rightarrow ID(\bigcirc x)] \qquad (7.6.P4)$$

$$\Box[edtbg(x) \wedge CR(x) \Rightarrow DT(\bigcirc x)] \qquad (7.6.P5)$$

$$\Box[edted(x) \wedge DT(x) \Rightarrow ID(\bigcirc x)] \qquad (7.6.P6)$$

Initial condition

$$ID(x) \qquad (7.6.P7)$$

The simulation result of the uncontrolled system is shown in Figure 2 and its reachability graph in Figure 3.

Let $EQ(x, y)$ represent that x is identical to y. The required behavior of the two entities system is given as follows:

$$\Box[CR(x) \Rightarrow \neg EQ(x,y) \wedge \neg CR(y) \wedge \neg WC(y)] \quad (7.6.CL1)$$

$$\Box[DT(x) \Rightarrow \neg EQ(x,y) \wedge \neg WC(y)] \qquad (7.6.CL2)$$

$$\Box[WC(x) \Rightarrow \neg EQ(x,y) \wedge \neg CR(y) \wedge \neg DT(y)] \quad (7.6.CL3)$$

where $\neg$ is the negation.

The supervisor developed following the above synthesis procedure has to block the states shown in Figure 4 from occurring.

For describing the supervisor we use local variable symbols p, q, and r . q is assigned values 1 and 0 representing

The state transitions of the uncontrolled system are:

```
The root:  s0=(ID,ID)
e1=(ecnrq 1)   --> s1=(WC,ID)
e3=(ecrsp 1)   --> s3=(DT,ID)
e2=(ecnrq 2)   --> s2=(ID,WC)
e4=(ecrsp 2)   --> s4=(ID,DT)

The root:  s1=(WC,ID)
e5=(eccfm 1)   --> s5=(CR,ID)
e6=(etmex 1)   --> s6=(ID,ID)=s0
e7=(ecnrq 2)=e2 --> s7=(WC,WC)
e8=(ecrsp 2)=e4 --> s8=(WC,DT)

The root:  s2=(ID,WC)
e10=(ecnrq 1)=e1 --> s10=(WC,WC)=s7
e9=(ecrsp 1)=e3 --> s9=(DT,WC)
e11=(eccfm 2)  --> s11=(ID,CR)
e12=(etmex 2)  --> s12=(ID,ID)=s0

The root:  s3=(DT,ID)
e13=(edted 1)  --> s13=(ID,ID)=s0
e15=(ecnrq 2)=e2 --> s15=(DT,WC)=s9
e14=(ecrsp 2)=e4 --> s14=(DT,DT)

The root:  s4=(ID,DT)
e17=(ecnrq 1)=e1 --> s17=(WC,DT)=s8
e16=(ecrsp 1)=e3 --> s16=(DT,DT)=s14
e18=(edted 2)  --> s18=(ID,ID)=s0

The root:  s5=(CR,ID)
e19=(edtbg 1)  --> s19=(DT,ID)=s3
e21=(ecnrq 2)=e2 --> s21=(CR,WC)
e20=(ecrsp 2)=e4 --> s20=(CR,DT)

The root:  s7=(WC,WC)
e22=(eccfm 1)=e5 --> s22=(CR,WC)=s21
e23=(etmex 1)=e6 --> s23=(ID,WC)=s2
e24=(eccfm 2)=e11 --> s24=(WC,CR)
e25=(etmex 2)=e12 --> s25=(WC,ID)=s1

The root:  s8=(WC,DT)
e27=(eccfm 1)=e5 --> s27=(CR,DT)=s20
e28=(etmex 1)=e6 --> s28=(ID,DT)=s4
e26=(edted 2)=e18 --> s26=(WC,ID)=s1

The root:  s9=(DT,WC)
e31=(edted 1)=e13 --> s31=(ID,WC)=s2
e29=(eccfm 2)=e11 --> s29=(DT,CR)
e30=(etmex 2)=e12 --> s30=(DT,ID)=s3

The root:  s11=(ID,CR)
e33=(ecnrq 1)=e1 --> s33=(WC,CR)=s24
e32=(ecrsp 1)=e3 --> s32=(DT,CR)=s29
e34=(edtbg 2)  --> s34=(ID,DT)=s4

The root:  s14=(DT,DT)
e35=(edted 1)=e13 --> s35=(ID,DT)=s4
e36=(edted 2)=e18 --> s36=(DT,ID)=s3

The root:  s20=(CR,DT)
e37=(edtbg 1)=e19 --> s37=(DT,DT)=s14
e38=(edted 2)=e18 --> s38=(CR,ID)=s5

The root:  s21=(CR,WC)
e41=(edtbg 1)=e19 --> s41=(DT,WC)=s9
e39=(eccfm 2)=e11 --> s39=(CR,CR)
e40=(etmex 2)=e12 --> s40=(CR,ID)=s5

The root:  s24=(WC,CR)
e43=(eccfm 1)=e5 --> s43=(CR,CR)=s39
e44=(etmex 1)=e6 --> s44=(ID,CR)=s11
e42=(edtbg 2)=e34 --> s42=(WC,DT)=s8

The root:  s29=(DT,CR)
e46=(edted 1)=e13 --> s46=(ID,CR)=s11
e45=(edtbg 2)=e34 --> s45=(DT,DT)=s14

The root:  s39=(CR,CR)
e47=(edtbg 1)=e19 --> s47=(DT,CR)=s29
e48=(edtbg 2)=e34 --> s48=(CR,DT)=s20
```

Figure 2: Simulation result of the uncontrolled system.

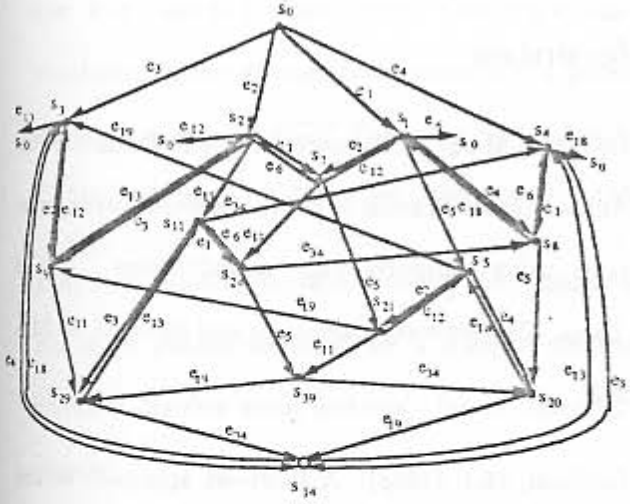

Figure 3: Reachability graph of the uncontrolled system.

According to the specifications of the required behavior

```
s6 should not been reached!
s9 should not been reached! So event e9 should be disabled at state s2 !
s15=(DT,WC)=s9 So event e15 should be disabled at state s2 !
s17=(WC,DT)=s8 So event e17 should be disabled at state s4 !
s21 should not been reached! So event e21 should be disabled at state s4 !
s22=(CR,WC)=s21 So event e22 should be disabled at state s7 !
s24 should not been reached! So event e24 should be disabled at state s7 !
s33=(WC,CR)=s24 So event e33 should be disabled at state s11 !
s41=(DT,WC)=s9 So event e41 should be disabled at state s11 !
s39 should not been reached! So event e39 should be disabled at state s21 !
s43=(CR,CR)=s39 So event e43 should be disabled at state s24 !
s42=(WC,DT)=s8 So event e42 should be disabled at state s24 !
s6 may not be a root!
s9 may not be a root!
s21 may not be a root!
s24 may not be a root!
s39 may not be a root!

Then the set of states which should not be reached are:

s6=(WC,DT) s9=(DT,WC) s21=(CR,WC) s24=(WC,CR) s39=(CR,CR)
```

Figure 4: The warnings and suggestions given by the simulator.

if an entity is in state CR or not; p and r are assigned non-negative integer values representing the number of entities which are in state WC or state DT. Let $ADD1(n)$ stand for that n is increased by 1 and $MIN1(n)$ for that n is decreased by 1. Then, the specifications of the controller are as follows.

Dynamics of the Controller

$$\Box[ecnrq(x) \wedge EQ(q,0) \Rightarrow ADD1(\bigcirc p) \wedge EQ(\bigcirc q,0)] \qquad (7.6.C1)$$

$$\Box[etmcx(x) \wedge \neg EQ(p,0) \Rightarrow MIN1(\bigcirc p)] \qquad (7.6.C2)$$

$$\Box[eccfm(x) \wedge EQ(p,1) \wedge EQ(q,0) \Rightarrow EQ(\bigcirc p,0) \wedge EQ(\bigcirc q,1)] \qquad (7.6.C3)$$

$$\Box[ecrsp(x) \wedge EQ(q,1) \wedge EQ(r,0) \Rightarrow EQ(\bigcirc q,1) \wedge EQ(\bigcirc r,1)] \qquad (7.6.C4)$$

$$\Box[edtbg(x) \wedge EQ(q,1) \wedge EQ(r,1) \Rightarrow EQ(\bigcirc q,0) \wedge EQ(\bigcirc r,2)] \qquad (7.6.C5)$$

$$\Box[edted(x) \wedge \neg EQ(r,0) \Rightarrow MIN1(\bigcirc r)] \qquad (7.6.C6)$$

Initial Condition of the Controller

$$EQ(p,0) \wedge EQ(q,0) \wedge EQ(r,0) \qquad (7.6.C7)$$

```
Now the state transitions of the controled system are:

The root:  S0 =(ID,ID,0,0,0)
e1=(econrq 1)   --> S1 =(WC,ID,1,0,0)
e2=(econrq 2)   --> S2 =(ID,WC,1,0,0)

The root:  S1 =(WC,ID,1,0,0)
e5=(eccfm 1)   --> S5 =(CR,ID,0,1,0)
e7=(econrq 2)=e2 --> S7 =(WC,WC,2,0,0)

The root:  S2 =(ID,WC,1,0,0)
e10=(econrq 1)=e1 --> S10 =(WC,WC,2,0,0)=S7
e11=(eccfm 2)   --> S11 =(ID,CR,0,1,0)

The root:  S5 =(CR,ID,0,1,0)
e20=(ecrsp 2)=e4 --> S20 =(CR,DT,0,1,1)

The root:  S7 =(WC,WC,2,0,0)
e23=(etmex 1)=e6 --> S23 =(ID,WC,1,0,0)=S2
e25=(etmex 2)=e12 --> S25 =(WC,ID,1,0,0)=S1

The root:  S11 =(ID,CR,0,1,0)
e32=(ecrsp 1)=e3 --> S32 =(DT,CR,0,1,1)

The root:  S20 =(CR,DT,0,1,1)
e37=(edtbg 1)=e19 --> S37 =(DT,DT,0,0,2)

The root:  S32 =(DT,CR,0,1,1)
e45=(edtbg 2)=e34 --> S47 =(DT,DT,0,0,2)=S37

The root:  S37 =(DT,DT,0,0,2)
e35=(edted 1)=e13 --> S48 =(ID,DT,0,0,1)
e36=(edted 2)=e18 --> S49 =(DT,ID,0,0,1)

The root:  S48 =(ID,DT,0,0,1)
e18=(edted 2)=e18 --> S50 =(ID,ID,0,0,0)=S0

The root:  S49 =(DT,ID,0,0,1)
e13=(edted 1)=e13 --> S51 =(ID,ID,0,0,0)=S0

The reachability Set:
 S0 =(ID,ID,0,0,0) S1 =(WC,ID,1,0,0) S2 =(ID,WC,1,0,0)
S5 =(CR,ID,0,1,0) S7 =(WC,WC,2,0,0) S11 =(ID,CR,0,1,0)
S20 =(CR,DT,0,1,1) S32 =(DT,CR,0,1,1) S37 =(DT,DT,0,0,2)
S48 =(ID,DT,0,0,1) S49 =(DT,ID,0,0,1)
```

Figure 5: The simulation result of the controlled system.

The controller designed above ensures that the closed-loop system has the required behavior. The simulation result of the closed-loop system is given in Figure 5 and its reachability graph is shown in Figure 6. It shows that there are two paths from state S_0 to state S_7 and from state S_0 to state S_{37}, respectively.

5 Conclusions

This paper has introduced a procedure for the synthesis of process supervisors for discrete event systems. A temporal logic model has been defined by an event structure for a set of formulae, an initial state, and a labelling function. In reachability analysis of the DES, a relationship between initial and validity has been given.

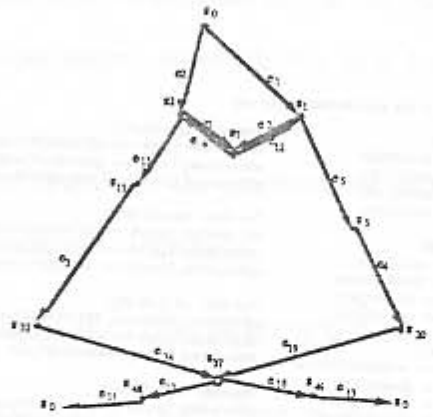

Figure 6: The reachability graph of the controlled system.

Temporal logic models have been viewed as Σ−algebras together with the Σ−homomorphisms between them, and a composition operator has been then defined to provide a basis for a theory of composition of a number of TLMs. Control action has been achieved by the composition of a controller with the plant. A supervisor synthesis procedure has been proposed through initial properties and imposed by synchronization of events in the controller with the selected events in the plant.

References

Clarke, E.M., E.A. Emerson, and A.P. Sistla (1986). Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Trans. Programming Languages and Systems*, 8, 244-263.

Denham, M.J. (1988). A Petri-net approach to the control of discrete-event systems. In M.J. Denham and A.J. Laub (Ed.), *Advanced Computing Concepts and Techniques in Control Engineering*, NATO ASI

Series, Vol. F47, Springer-Verly, Berlin, pp.191-214.

Fusaoka, A., H. Seki, and K. Takahashi (1983). A description and reasoning of plant controllers in temporal logic. *Proc. 8th Int. Joint Conf. on Artificial Intelligence*, 405-408.

Galton, A. (1987). *Temporal Logics and Their Applications*, Academic Press, London.

Hennessy, M. (1988). *Algebraic Theory of Processes*, The MIT press, Massachusetts.

Inan, K., and P. Varaiya (1988). Finitely recursive process models for discrete event systems. *IEEE Trans. Automat. Contr.*, **33**, 626-639.

Lin, J.-Y., and D. Ionescu (1993). A initial synthesis procedure for discrete event systems in a temporal logic approach, submitted to *IEEE Trans. on Syst. Man, and Cyber.*

Lin, J.-Y., and D. Ionescu (1992). Verifying a class of nondeterministic discrete event systems in a generalized temporal logic, *IEEE Trans. on Syst. Man, and Cyber.*, vol. 22, no 6, Nov.-Dec.,1992, pp 1461-1469.

Lin, J.-Y., and D. Ionescu (1991a). Asymptotic behavior of output feedback for a class of nondeterministic discrete event systems. *Int. J. Control*, to appear.

Lin, J.-Y., and D. Ionescu (1991b). A temporal logic approach to the control of discrete event systems. *Proc. the 1991 American Control Conference*, Boston, Massachusetts, June 26-28, 1991. pp.2934-2935.

Manna, Z., and A. Pnueli (1983). Verification of concurrent programs: A temporal proof system. *Foundations of Computer Science IV*, Mathematical Centre Tracts 159, Mathematish Centrum, Amsterdam, pp.163-255.

Manna, Z., and P. Wolper (1984). Synthesis of communicating processes from temporal logic specifications. *ACM Trans. Programming Languages & Systems*, **6**, 68-93.

Ostroff, J.S. (1989). *Temporal Logic for Real-Time Systems*. John Wiley, New York.

Pnueli, A. (1981). The temporal semantics of concurrent programs. *Theoret. Comput. Sci.*, **13**, 45-60.

Ramadge, P.J., and W.M. Wonham (1987). Supervisory control of a class of discrete event process. *SIAM J. Contr. Optimiz.*, **25**, 206-230.

Thistle, J.G., and W.M. Wonham (1986). Control problems in a temporal logic framework. *Int. J. Control*, **44**, 943-976.

Wonham, W.M. (1988). A control theory for discrete-event systems. *Advanced Computing concepts and Techniques in Control Engineering*. ed. M.J. Deham and A.J. Laub. Springer-Verly, Berlin. pp.129-169.

F. Halsall, *Data Communications, Computer Networks and OSI*, Addison-Wesley, New York, 1988.