

Control of Event Separation Times in Discrete Event Systems

Darren D. Cofer
Honeywell Technology Center
3660 Techology Dr.
Minneapolis, MN 55418
cofer_darren@htc.honeywell.com

Vijay K. Garg¹
Dept. of Elec. and Comp. Engineering
University of Texas at Austin
Austin, TX 78712-1084
garg@ece.utexas.edu

Abstract

The class of timed discrete event systems which can be modelled by automata known as timed event graphs are structurally related to finite state machines. Consequently, supervisory control problems for these timed DES can be addressed using methods similar to those developed for their untimed counterparts. When the desired behavior takes the form of minimum separation times between events, it also can be expressed as a timed event graph. Supervised behavior is then defined by the synchronous operation of the plant and specification automata. Controllability and the existence of optimal behaviors can be evaluated in this framework.

1. Introduction

Discrete event systems (DES) which are subject to synchronization constraints in time can be modelled by automata known as timed event graphs [2]. A timed event graph (TEG) is a Petri net in which a delay or processing time is associated with each place and such that forks and joins occur only at its transitions. Each transition in the graph corresponds to an event in the system. Therefore, an event will only occur after all of the processes (places) connecting it to predecessor events are completed.

An autonomous TEG (one having no input transitions) will execute a fixed behavior which may be described by the vector sequence of transition firing times corresponding to the occurrence times of all events. Suppose that some of the events are controllable, meaning that their occurrence may be arbitrarily delayed. Given a desired behavior of the system (such as a set or range of acceptable firing time sequences) we wish to determine whether there exist control actions which will restrict the system to that behavior. If this is not possible, we wish to determine a subset of the desired behavior which can be realized and is in some sense optimal.

By using a max-algebra model for timed DES, this problem

becomes similar to the supremal controllable sublanguage problem for untimed DES [9]. The max-algebra approach allows a TEG to be described by a linear equation. Furthermore, controllable behavior may be characterized by an invariance condition on the lattice of time sequences [5]. This is a departure from previous max-algebra control work which focused on the input-output behavior of systems.

In this paper we consider specifications which are in the form of minimum desired separation times between events in the system. Such a specification can also be expressed as one or more TEGs. We show that questions of controllability and optimal behavior can be addressed by considering the synchronous operation of the plant and specification TEGs. Thus, for this type of problem we follow an automata-based approach, in contrast to [5] where the methods were analogous to formal language theory. A preliminary version of this work appears in [4]. Proofs and additional details may be found in [3].

2. Timed Event Graphs

The dynamic behavior of a TEG can be studied using max-algebra techniques [2, 3]. Consider a TEG $G = (T, A)$ where T is a set of N transitions and A is an $N \times N$ matrix of delay functions at places connecting the transitions. Then the sequence x_i of firing times of transition t_i satisfies

$$x_i = \max\{\max_{1 \leq j \leq n} \{A_{ij}x_j\}, v_i\} \quad (1)$$

where v_i specifies an earliest starting time for t_i . The absence of a place connecting t_j to t_i is denoted by $A_{ij} = \varepsilon$, which represents a delay of $-\infty$ and hence no restriction.

Let $\mathcal{S}$ be the set of sequences over $\mathcal{R} \cup \{\pm\infty\}$. Using the shorthand of max-algebra where $\oplus$ denotes pointwise maximization in $\mathcal{S}^N$, (1) can be written

$$x = Ax \oplus v. \quad (2)$$

Assuming there are no other constraints and since transitions fire as soon as they are enabled, the actual behavior of the system is given by the least solution to (2) which is $x = A^*v$,

¹Research supported in part by NSF grant CCR-9110605, a General Motors Fellowship, and an IBM grant

where

$$A^* = \bigoplus_{i \geq 0} A^i.$$

Borrowing notation from untimed automata we denote the behavior of G by $L(G)$.

For a TEG with constant delay times, the delay functions are of the form $a\gamma^m$, where a represents unary addition of a constant delay to every term in the sequence and γ is the index backshift function ($\gamma x(k) = x(k-1)$). Each of $m < M$ tokens in the initial marking of a place causes a backshift since the $(k-m)$ th token to enter the place will be the k th to depart. We define the *degree* of a delay function of this form by $\deg(a\gamma^m) \equiv m$.

If we are interested in the individual terms of the sequence x it is convenient to rewrite (2) as the recurrence equation

$$x(k) = A_0^*(A_1 x(k-1) \oplus \dots \oplus A_M x(k-M) \oplus v(k)). \quad (3)$$

Here the delay matrix has been decomposed into matrices of delay functions having the same degree so that

$$A = A_0 \oplus A_1 \gamma \dots \oplus A_M \gamma^M. \quad (4)$$

Timed event graphs are structurally very similar to finite state machines (FSM) which are often used to model untimed DES. Both are directed graphs with weighted edges. In a timed event graph the nodes correspond to events and the edges to process delays, while the nodes in a finite state machine represent the system states and the edges correspond to events. The weight of an edge in a TEG specifies a minimum separation time between the events it connects and therefore imposes an order on their execution times. The edge weight in a FSM is just the event label. The structures are dual in the sense that a FSM models nondeterministic choice but not synchronization while a TEG models synchronization but not choice.

As a result of their common structure both TEGs and FSMs are governed by the same equation (2). The only difference is the definition of the operators in the underlying algebra [3]. It is this structural and algebraic similarity which suggests that control of timed DES modelled by TEGs may be studied using techniques developed for FSMs.

3. Supervisory Control of Timed Event Graphs

Suppose that some events $T_c \subseteq T$ are controllable, meaning that their transitions may be delayed from firing until some arbitrary later time. The delayed enabling times $u_i(k)$ are provided by a supervisor, with $u_i(k) = -\infty$ for t_i uncontrollable. Then the supervised system is described by $x = Ax \oplus v \oplus u$.

Now let Y be a set of desired behaviors (sequences) to which we would like to restrict the system. We say that Y is controllable if enabling the controllable transitions at any times allowed by Y is sufficient to guarantee that the resulting behavior belongs to Y . To compute the effect of uncontrollable events, let I_c denote the matrix having the identity function

on diagonal elements i for which $t_i \in T_c$ and ε elsewhere. Then for any desired sequence $y \in Y$ the supervisor provides firing times $u = I_c y$. This results in actual behavior $x = A^*(I_c y \oplus v)$. Since the desired behavior must be invariant under uncontrollable actions, we formalize the definition of controllability as follows [3, 5].

Definition 1 A set of sequences $Y \subseteq \mathcal{S}^N$ is controllable with respect to A , v , and T_c if

$$A^*(I_c Y \oplus v) \subseteq Y. \quad (5)$$

Sets of sequences in $\mathcal{S}^N$ may be ordered by inclusion to form a lattice. The controllability of a set of behaviors is determined by inequation (5) on this lattice. Likewise, the controllability of untimed DES modelled by FSMs is determined by an inequation on the lattice of languages. In the untimed case if a language is not controllable we can find its supremal controllable sublanguage. An uncontrollable set of sequences for a TEG is shown in [5] to have an extremal controllable subset and superset. Furthermore, these extremal controllable behaviors can be found using the same fixed point results for lattices which are used to study untimed DES [7].

In [5] the desired behaviors considered were specified in terms of bounds on the occurrence times of events. Suppose instead that the desired system behavior is specified by minimum separation times that must be maintained between events. For example, we may want to delay the departure of an airplane until all of its connecting flights have been on the ground for some minimum time. In a communication network such constraints could be used to restrict traffic admitted to the network.

For this type of constraint, the set of desirable behaviors is of the form

$$Y = \{x \in \mathcal{S}^N \mid x \geq Sx\} \quad (6)$$

where $x_i \geq S_{ij}x_j$ specifies delay functions connecting pairs of events. Rather than determining the controllability of the set Y associated with specification S , we determine the existence of controllable sequences belonging to the set. We specialize the definition of controllability for a single sequence y to be

$$A^*(I_c y \oplus v) = y$$

and formally state the problem as follows:

Problem: Given a plant $G_p = (T, A)$ and a specification of minimum event separation times S , find a supervisor for the specification.

In [1] upper and lower bounds on event separation times were found. Our interest is to determine whether a specified limit on event separation times can be achieved given the control available. In [5] a language-based approach was taken to this problem. Given a set of the form (6) the supremal controllable sequence satisfying the minimum separation constraints and less than a fixed upper bound is shown to exist.

It is possible to instead use an automata-based approach to examine this problem. The minimum required separation times S essentially form another TEG in the same way as the system or plant delay times. Thus the supervisor which provides the delayed enabling times to the controllable events in the plant is another TEG, defined by $x = Sx \oplus v$.

4. Control by Synchronization

Often the desired behavior for an untimed DES will be specified by another FSM. The specification FSM is made to serve as a supervisor for the plant FSM by operating the two in synchrony [9]. This means that the plant and the supervisor will execute an event only if it is enabled in both machines. Thus, the supervisor may prevent or disable events which otherwise would have been permitted in the plant. A supervisor is said to be complete if it does not disable any uncontrollable events. These concepts are extended to untimed Petri nets in [8].

Synchronous composition of timed event graphs can be defined in a similar manner. Assume that we are given two event graphs defined by delay matrices A_1 and A_2 . Transitions which are common to both graphs fire only when they are enabled in both graphs. Transitions which appear in only one of the graphs fire when enabled by their own local predecessors, as before. An important observation is that synchronization may be used to realize a control policy for a TEG since some of its events will be delayed if they are synchronized to events in another TEG.

In the max-algebra representation, the construction of the synchronous composition of two systems is straightforward. If the respective event sets T_1 and T_2 are not identical, we assume that events are mapped into their union with A_1 and A_2 permuted and augmented with empty rows and columns as needed. Then the synchronous behavior of the plant and the supervisor, denoted $G_1 \parallel G_2$, is governed by

$$x = (A_1 \oplus A_2)x \oplus v$$

where $v = v_1 \oplus v_2$. The sequence of firing times $L(G_1 \parallel G_2)$ is then given by $(A_1 \oplus A_2)^* v$.

Note that as for the synchronous composition of FSMs, this is a completely general and symmetric construction. We can consider the synchronous operation of any number of TEGs in any order. In fact, the decomposition in (4) of a system delay matrix A into the collection $\{A_i\}$ based the number of initial tokens is an example.

It is easy to see that synchronous composition results in event times which are greater than or equal to those of the individual systems operating independently.

Theorem 1 $L(G_1 \parallel G_2) \geq L(G_1) \oplus L(G_2)$.

This follows directly from the definition and the fact that

$$(A_1 \oplus A_2)^* v \geq A_1^* v \oplus A_2^* v.$$

Clearly synchronous composition of TEGs may introduce new circuits. Thus there is the potential to introduce a deadlock (a circuit containing no tokens) that was not present before. This condition is checked by examining A_0 , the subgraph of places in the synchronized graph with no initial tokens. If there exists a permutation of the rows and columns of this matrix that makes it strictly upper triangular then it is circuit-free. Equivalently, if there exists $n \leq N$ such that $(A_0)^n = \varepsilon$ then there is no deadlocked circuit.

Now consider the synchronous composition of a plant $G_p = (T, A)$ and a supervisor $G_s = (T, S)$. In defining synchronization of TEGs we have implicitly assumed that all events in the plant are controllable. If this is actually the case, the problem becomes trivial since $y = (A \oplus S)^* v$ satisfies the minimum separation time specification $y \geq Sy$ and y is a controllable sequence. In fact, there is no sequence less than y which satisfies these requirements.

We must now account for the effect of the uncontrollable events. The specification is a legitimate supervisor for the plant if it does not *directly* delay the occurrence of uncontrollable events. That is, the specification may require uncontrollable event t_i to occur d seconds after any other event t_j , but this delay cannot be imposed directly; only through the action of controllable events and delays within the plant. This is analogous to the requirement that a FSM supervisor not disable any uncontrollable events.

To verify this we must check if the synchronous composition of the plant and supervisor is changed by deleting from the supervisor all incoming edges to uncontrollable events. Let I_c be defined as before. Then $\hat{G}_s = (T, I_c S)$ removes from the supervisor graph all incoming edges to events which are uncontrollable in the plant. We may now sum up this condition with the following definition.

Definition 2 A supervisor $G_s = (T, S)$ is complete with respect to plant $G_p = (T, A)$ and controllable event set T_c if the synchronous composition of the supervisor and plant does not depend upon any delay of uncontrollable events imposed directly by the supervisor; that is,

$$\begin{aligned} L(G_p \parallel \hat{G}_s) &= L(G_p \parallel G_s) \\ \text{or } (A \oplus I_c S)^* v &= (A \oplus S)^* v. \end{aligned}$$

If the completeness condition holds, then the specification is implemented simply by means of the synchronous composition of the plant and the specification event graphs. The resulting behavior is controllable and meets the separation time requirements.

Theorem 2 If supervisor S is complete with respect to A and T_c then $y = (A \oplus S)^* v$ is a controllable sequence and $y \geq Sy$. Furthermore, y is the least such sequence.

Sufficient conditions for the completeness of a specification can also be computed using the recurrence form (3).

Example 1 Consider the following plant and supervisor

$$A = \begin{bmatrix} \varepsilon & \varepsilon & 1\gamma \\ \varepsilon & \varepsilon & 1\gamma \\ 2 & 1 & \varepsilon \end{bmatrix} \quad S = \begin{bmatrix} \varepsilon & 1 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ \varepsilon & 3 & \varepsilon \end{bmatrix}$$

with t_1 controllable in the plant. Letting $D = A \oplus S$ and $\hat{D} = A \oplus I_c S$ we have

$$D_0 = \begin{bmatrix} \varepsilon & 1 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ 2 & 3 & \varepsilon \end{bmatrix} \quad \hat{D}_0 = \begin{bmatrix} \varepsilon & 1 & \varepsilon \\ \varepsilon & \varepsilon & \varepsilon \\ 2 & 1 & \varepsilon \end{bmatrix}.$$

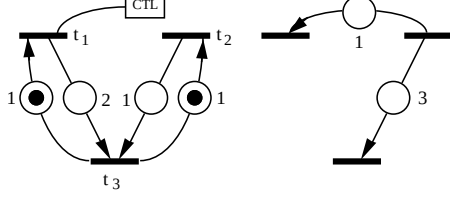


Figure 1: Plant (left) and complete supervisor (right)

Since $D_0^* = \hat{D}_0^*$ and $D_1 = \hat{D}_1$ the sequences generated are identical and the supervisor is complete. For further insight, looking at the graph in Fig. 1 we see that the desired separation of 3 between t_2 and uncontrollable event t_3 is achieved by imposing a delay of 1 between t_2 and controllable event t_1 . Therefore the sequence $(A \oplus S)^*v$ is controllable. $\square$

Suppose that the specification may be decomposed as a collection of k TEGs whose separation times must all be satisfied. Equivalently, the desired behavior may be specified in a modular fashion. In either case we have an overall specification of the form

$$S = S_1 \oplus \dots \oplus S_k.$$

It turns out that the component specifications may be evaluated individually to make a valid judgement regarding the completeness and, therefore, the controllability of the whole specification.

Theorem 3 *If each of the specifications in the collection $\{S_i\}$ is complete then $\bigoplus_i S_i$ is complete.*

The next theorem gives us a componentwise check for completeness which is both necessary and sufficient.

Theorem 4 *A specification S is complete for any initial condition if and only if*

$$S \leq (A \oplus I_c S)^*. \quad (7)$$

To check completeness we only need to verify

$$[S]_{ij} \leq [(A \oplus I_c S)^*]_{ij}$$

for all i and j such that $t_i \notin T_c$ and $[S]_{ij} > \varepsilon$. For all other i and j the condition is trivially satisfied.

5. Optimal Controllable Behavior

If the given specification for event separation times is not complete, we would like to find a complete specification which is optimal in some sense. Assuming that it is important to achieve at least the event separation times given by S , an optimal specification would be the least $\hat{S} \geq S$ which is complete. Unfortunately, we can run into problems in both the existence and uniqueness of such an optimal specification.

If there is no path in the plant from a controllable transition to any uncontrollable transition which must be delayed in the specification S , then there is no $\hat{S} \geq S$ which is complete. We characterize the existence of such paths as follows.

Definition 3 *A transition t_i is structurally controllable if it is reachable from some controllable transition; that is, there exists $n < N$ and $t_j \in T_c$ such that $[A^n]_{ij} > \varepsilon$. $[A^n]_{ij}$ is called a controllable path.*

We may now state a sufficient condition for the existence of a complete specification greater than or equal to S .

Theorem 5 *Suppose S is not complete. Let U denote the set of transitions for which the completeness condition (7) fails to hold. Then a complete specification $\hat{S} \geq S$ exists if the following two conditions are satisfied:*

1. *Every transition in U is structurally controllable.*
2. *For each $t_i \in U$ with $[S]_{ij} \equiv g > \varepsilon$ there exists a controllable path f such that $\deg(f) \leq \deg(g)$.*

If every uncontrollable event t_i which requires a delay from some t_j is reachable from a controllable event t_c (condition 1) then the result follows by adding the appropriate delay from t_j to t_c . Condition 2 is necessary since tokens can be added to the delay path but excess tokens cannot be removed.

Even when complete supervisors greater than the given specification exist, there may not be a minimum complete supervisor. This is because the set of complete supervisors is not closed under minimization. Therefore, we propose the following procedure for generating a minimal complete supervisor; that is, one such that no supervisor less than it is complete. The procedure incorporates the existence criteria above.

Procedure

1. Determine if the specification S is complete.
 $\forall t_i \notin T_c$, let $V_i = \{t_j \mid [S]_{ij} > [(A \oplus I_c S)^*]_{ij}\}$
 $U = \{t_i \mid V_i \neq \emptyset\}$
 Complete if $U = \emptyset$ (Theorem 4).
2. Verify structural controllability for all $t_i \in U$.
 $W_i = \{f \mid f \text{ is a controllable path for } t_i\}$
 If $W_i = \emptyset$ then t_i is not structurally controllable (condition 1 of Theorem 5).
3. Compute delays to be added to S .
 $\forall t_j \in V_i$, with $g \equiv [S]_{ij}$
 - (a) Select $f \in W_i$ such that $\deg(f) \leq \deg(g)$.
 (If none, condition 2 of Theorem 5 fails.)
 - (b) Compute minimum h such that $fh \geq g$ and set $[\hat{S}]_{cj} = [\hat{S}]_{cj} \oplus h$.

By its construction, the procedure finds a complete supervisor $\hat{S} \geq S$ if one exists. Minimality is achieved by the computation of h in step 3(b). Strictly speaking, it turns out that the resulting supervisor may not, in fact, be minimal. It is possible that a delay added in one step may be made unnecessary by a delay added in a parallel path at a later step. Thus, by deleting the unnecessary delay we could

obtain a supervisor less than that computed by our procedure. For practical purposes this is of little consequence since the resulting behavior will be the same in either case.

It is possible that any delays h with $\deg(h) = 0$ added in step 3(b) could introduce deadlocked circuits. This situation can be handled in two ways. After using the procedure to find the new supervisor, check $(A \oplus I_c \hat{S})_0$ to find any deadlocks and identify the added delays which caused them. Then rerun the procedure for the affected transitions, with the restriction that different controllable paths must be chosen in step 3(a).

Alternatively, deadlocks can be avoided on-line by making the following changes to step 3(a). Where possible, give preference to the selection of controllable paths such that $\deg(f) < \deg(g)$. This ensures that $\deg(h) \geq 1$, avoiding the possibility of deadlock. In those cases where $\deg(h) = 0$ cannot be avoided, it necessary to find a controllable path using t_c such that $[(A \oplus I_c \hat{S})_0^*]_{jc} = \varepsilon$. If this cannot be done, deadlock is inevitable.

As a final consideration, suppose that condition 2 of Theorem 5 fails to hold in step 3(a). A further attempt at finding a complete supervisor may be made with the following modification to step 3:

- For t_j failing this condition, replace t_j by the set of its immediate predecessors $\{t_p\}$ in the plant A .
For each predecessor t_p , let $g \equiv [S]_{ij}[A]_{jp}$.

Now if step 3(a) is successful for the modified set of transitions, the separation time required by $[S]_{ij}$ can be met.

To see why this works, consider the case where t_j has a single predecessor t_p and the required separation is $x_i \geq g(x_j)$. Then assuming condition 2 can be satisfied for t_p and letting $g_p = gA_{jp}$, we can find delay h_p and a controllable path such that $fh_p \geq g_p$. Thus we have

$$\begin{aligned} x_i \geq fh_p(x_p) &\geq g_p(x_p) \\ &= gA_{jp}(x_p) \\ &= g(x_j). \end{aligned}$$

Example 2 To illustrate the results of this section, we consider a manufacturing work cell which uses automated guided vehicles (AGV) for transporting parts between stations in the cell. The control objective is to prevent collisions of the AGVs by ensuring sufficient separation between certain events. The logical behavior of this system is studied in [6]. A timed event graph model for the work cell is shown in Figure 2. Two different types of unfinished parts arrive at the input parts stations to be picked up by AGVs. The parts are transported to two different work stations for machining and then moved to a another work station for final machining and assembly. The finished product is then transported to the completed parts station for packaging.

The cell is described by the synchronous composition of 11 subsystems:

- 2 input parts stations M, N
- 3 work stations U, V, W
- 5 AGVs A, B, D, E, F

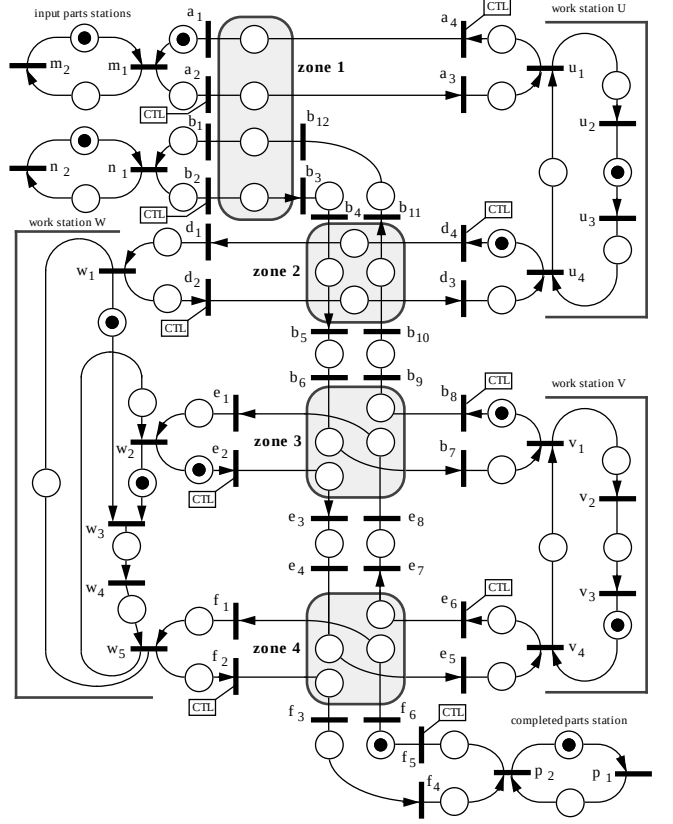


Figure 2: Manufacturing work cell.

- 1 completed parts station P .

Each of these subsystems contains some internal events and shared events. The shared events are those that are synchronized with other subsystems. For example, AGV A is described by the equation

$$x_A = Ax_A \oplus A_M x_M \oplus A_U x_U \oplus v_A$$

where x_A denotes the occurrence times of the internal events for A while x_M and x_U are events shared with subsystems M and U .

The coupled equations for all of the subsystems can be collected into a single equation of the form $x = Ax \oplus v$ where A is

$$\begin{bmatrix} M & & & & & & & & & & \\ & M_A & & & & & & & & & \\ & N & B_N & & & & & & & & \\ A_M & & A & & & & A_U & & & & \\ & B_N & B & & & & & B_V & & & \\ & & & D & & & D_U & D_W & & & \\ & & & & E & & & E_V & E_W & & \\ & & & & & F & & F_V & F_W & F_P & \\ & U_A & & U_D & & & U & & & & \\ & & V_B & & V_E & & & V & & & \\ & & & W_D & W_E & W_F & & & W & & \\ & & & & & P_F & & & & & P \end{bmatrix}$$

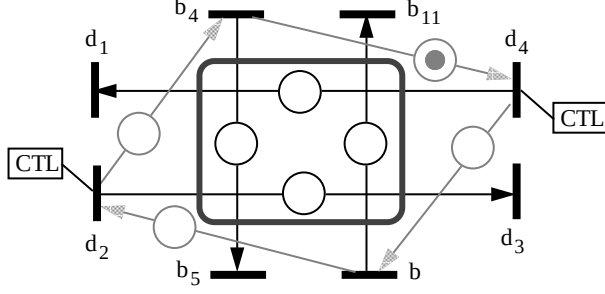


Figure 3: Specification for AGVs in Zone 2.

Due to space restrictions, the AGVs must pass through several common zones on the workfloor. Only one AGV may occupy a zone at any time; otherwise, there will be a collision.

Events in the AGV subsystems correspond to the passage of the vehicles through various points on their paths. Certain of these events are controllable (as indicated in Figure 2), meaning that the AGV may be delayed at that point. To prevent collisions, we specify minimum separation times between the admittance of each AGV into a zone. Zone 2 and its separation time specifications are shown in Figure 3. The required separation times are specified by two delay matrices for each zone. For zone 2 we have

$$\begin{aligned} x_B &\geq S_3 x_D \\ x_D &\geq S_4 x_B \end{aligned}$$

The control objective is to impose the minimum delays at the controllable events consistent with meeting the specification for each zone.

The first step is to check whether the specifications are complete. By Theorem 3 we may consider each specification separately. Using Theorem 4 we obtain the following results.

Zone 1:	$S_1 = A_B$	complete
	$S_2 = B_A$	not complete at b_{12}
Zone 2:	$S_1 = A_B$	not complete at b_4 and b_{10}
	$S_2 = B_A$	complete
Zone 3:	$S_1 = A_B$	not complete at b_6
	$S_2 = B_A$	not complete at e_8
Zone 4:	$S_1 = A_B$	not complete at e_4
	$S_2 = B_A$	not complete at f_6

Now we use Theorem 5 to determine if an optimal supervisor can be found for those specifications which fail completeness. In every event in question, there is a structurally controllable path which satisfies the degree condition of the theorem. It is a simple matter now to compute the additional delay arcs required to form the revised complete specifications $\hat{S}_2, \hat{S}_3, \hat{S}_5, \hat{S}_6, \hat{S}_7$, and $\hat{S}_8$. These specifications are therefore controllable, satisfy the original separation time re-

quirements to avoid collisions, and impose the least possible delays to accomplish this. As the last step, we compute $(A \oplus S)_0^*$ to ensure that no deadlock conditions have been introduced by the new supervisors. $\square$

6. Conclusion

When the desired behavior of a timed event graph is given in terms of minimum separation times between events, the specification may be expressed as another timed event graph. We have defined the synchronous composition of TEGs and shown that this may be used to control the plant by delaying certain events. We have introduced an appropriate notion of completeness to characterize specifications which may legitimately function as supervisors for a particular plant. When a specification fails to be complete we have shown that the set of complete specifications greater than the desired specification is not well-behaved in that it may be empty or may fail to have a unique minimum element. In this case we have outlined a procedure to generate a minimal complete specification.

References

- [1] T. Amon, H. Hulgaard, S. M. Burns, G. Borriello, "An algorithm for exact bounds on the time separation of events in concurrent systems," *IEEE Int. Conf. on Computer Design*, October 1993.
- [2] F. Baccelli, G. Cohen, G. J. Olsder, J. P. Quadrat, *Synchronization and Linearity*, Wiley, New York, 1992.
- [3] D. D. Cofer, *Control and Analysis of Real-Time Discrete Event Systems*, Ph.D. Thesis, Dept. of ECE, Univ. of Texas at Austin, May 1995.
- [4] D. D. Cofer, V. K. Garg, "Supervisory Control of Timed Event Graphs," in *Proc. 1994 IEEE Int. Conf. Sys. Man & Cyb.*, San Antonio, TX, pp. 994-999, Oct. 1994.
- [5] D. D. Cofer, V. K. Garg, "Supervisory Control of Real-Time Discrete Event Systems Using Lattice Theory," in *Proc. 33rd Conf. Dec. Ctl.*, Orlando, FL, pp. 978-983, Dec. 1994 (also to appear in *IEEE Trans. Auto. Ctl.*).
- [6] L. E. Holloway, B. H. Krogh, "Efficient synthesis of control logic for a class of discrete event systems," in *Proc. 1989 Amer. Ctl. Conf.*, Pittsburgh, PA, June 1989.
- [7] R. Kumar, V. K. Garg, *Modeling and Control of Logical Discrete Event Systems*, Kluwer, 1995.
- [8] R. Kumar, L. E. Holloway, "Supervisory Control of Petri Net Languages," *Proc. of the 31st Conf. on Decision and Control*, Tuscon, AZ, pp. 1190-1195, Dec. 1992.
- [9] P. J. Ramadge, W. M. Wonham, "The control of discrete event systems," *Proceedings of the IEEE*, vol. 77, no. 1, pp. 81-98, 1989.