

# Non-stochastic Analysis of Manufacturing Systems using Timed Event Graphs

**Tod Amon**

Southwest Texas State University  
San Marcos, TX 78666

**Henrik Hulgaard**

University of Washington  
Seattle, WA 98195

## **Abstract**

Using automated methods to analyze the temporal behavior of manufacturing systems has proven to be essential and quite beneficial. Popular methodologies include Queueing networks, Markov chains, simulation techniques, and discrete event systems (such as Petri nets). These methodologies are primarily stochastic. Performance evaluation mandates results which are probabilistic in nature (such as the average rate of part deliveries) and relies on probabilistic inputs (such as the probability of breakdown, or the distributions associated with a manufacturing process). This paper examines non-stochastic analysis, which we argue can be useful for verifying *correct* operation. We model manufacturing systems using timed event graphs which are similar to decision free Petri nets augmented with timing information, and present an example that demonstrates the efficacy of non-stochastic analysis.

**Keywords:** *Non-stochastic Timing Analysis, Timed Event Graphs, Timing Verification, Non-deterministic Bounded Delays*

**Note:** *This paper was typeset and edited and appeared in a slightly modified form in the SME Journal of Manufacturing Systems, May/June 1996.*

## Introduction

The majority of research involving the formal analysis of temporal issues in manufacturing systems has focussed on stochastic techniques. Simulation, Queuing networks, Markov chains, and Petri nets form the basis of many of the most widely known and used methods. Performance evaluation, which is concerned with answering questions such as “What are the average part delivery times” or “What might the effect of a machine breakdown be?” is, it could be argued, stochastic by its very nature. It is thus not surprising that so much emphasis has been placed on stochastic methodologies. Algebraic techniques have been developed for analyzing transient behavior [6], in which it is claimed that the “deterministic approach may sometimes be more adequate than assuming exponential service times.” The algebraic methods have also been extended to stochastic systems [5]. Given the nature and quantity of prior research, one might be tempted to conclude that non-stochastic analysis using non-deterministic delays is non-essential and nonsensical. The primary objective of this paper is to refute this assertion.

We believe that non-stochastic analysis is *very relevant* to the manufacturing domain. Stochastic analysis may be more suitable for analyzing efficiency and utilization, but non-stochastic techniques are needed for verifying *correct* operation. For example, we might allow a recently glued part to proceed down a production line, and yet require it to be fully dry by the time a later machine operates upon it. Given bounds on processing and drying times we would like to prove that the part will always be dry. Non-stochastic analysis can provide temporal guarantees which might be needed in order

to optimize the manufacturing process. Non-stochastic analysis can also be used to provide insight into the dynamic nature of the manufacturing system. Knowing the average amount of time it takes to produce a part does not provide any information regarding whether consecutive deliveries could potentially be quite close together or quite far apart. Non-stochastic verification can be used to prove that a behavioral model of a system is invariant with respect to a model transformation. This allows simplified models to be used when more complex analysis methodologies will be employed.

Non-determinism is necessary, given that it is precisely the uncertainty present in manufacturing systems which has led to simulation and stochastic methodologies being so popular. Although in some cases it might make sense to assume that processing times are fixed, it might also make sense to assume that they vary due to differences in the parts being processed, the temperature of the machine, etc. Modeling, by its very nature, relies on abstractions of real world systems. Non-determinism is a very powerful modeling tool, because it enables many different abstractions for real-world complexities to be employed.

We follow this introduction with a description of the “Time Separation of Events” (TSE) problem and algorithm which determines tight (non-stochastic) bounds on separation times given a description of a concurrent system and lower and upper bounds on delays (e.g., processing and transportation times). The TSE problem framework is used as a basis for the example presented in the subsequent section, which demonstrates the relevance of non-stochastic analysis to manufacturing systems. Conclusions and limitations are then discussed.

## Time Separation of Events

We consider a class of Discrete Event Dynamic Systems called timed event graphs (TEGs), which are in many ways similar to decision free Petri nets annotated with timing information. Many of the modeling techniques used for Petri nets can thus be used for TEGs. Descriptions of Petri nets and their use and relevance to manufacturing are quite prevalent in the literature, and we refer readers to Murata's tutorial [14] for further information. Our work differs in that it is aimed at non-stochastic analysis and allows non-deterministic delays. In this regard, it can be viewed as an extension of the deterministic algebraic techniques [6].

The remainder of this section contains a formal description of TEGs and the TSE problem and solution. Some readers may want to proceed to the example section for a more informal description of TEGs and how they are used to model manufacturing systems.

### Problem Formalization

Let  $G' = \langle E', R' \rangle$  denote a timed event graph (TEG) composed of

- a finite set of events  $E'$ , the vertices of the graph.
- a finite set of rule templates  $R'$ , the edges of the graph.

Each edge is labeled with two objects, the delay range  $[d, D]$  (integers with  $0 \leq d \leq D$ ), and a non-negative occurrence index offset  $\varepsilon$ . The set of events,  $E'$ , always contains a unique event, *root*, which is used to specify the startup behavior of the system. We can represent possible executions of a TEG explicitly by *unfolding* the graph to create an infinite directed

graph  $G = \langle E, R \rangle$ . The set of event *occurrences*  $E$  can be defined recursively from the basis clause  $E = \{root_0\}$  and the inductive clause: if  $u_k \in E$  and  $u \xrightarrow{[d,D],\varepsilon} v \in R'$ , then  $v_{k+\varepsilon} \in E$ . The set of edges  $R$  is defined as:

$$R = \left\{ u_k \xrightarrow{[d,D]} v_{k+\varepsilon} \mid u \xrightarrow{[d,D],\varepsilon} v \in R' \text{ and } u_k, v_{k+\varepsilon} \in E \right\} .$$

A simple TEG and a part of its unfolding is shown in Figure 1. The TEG has  $E' = \{root, a, b\}$  and  $R' = \{root \xrightarrow{[0,0],0} a, root \xrightarrow{[0,0],0} b, a \xrightarrow{[4,10],1} a, a \xrightarrow{[1,2],0} b, b \xrightarrow{[1,6],1} a, b \xrightarrow{[5,20],1} b\}$ .

A *timing assignment*,  $\tau$ , maps event occurrences to global time, thus  $\tau(v_k)$  is the time of the  $k^{\text{th}}$  occurrence (in the unfolded graph) of event  $v$ . The delay information in  $R$  restricts the set of possible timing assignments:

$$\begin{aligned} \tau(v_k) &\geq \max \left\{ \tau(u_{k-\varepsilon}) + d \mid u_{k-\varepsilon} \xrightarrow{[d,D]} v_k \in R \right\} \\ \tau(v_k) &\leq \max \left\{ \tau(u_{k-\varepsilon}) + D \mid u_{k-\varepsilon} \xrightarrow{[d,D]} v_k \in R \right\} . \end{aligned}$$

The constraints on  $\tau(v_k)$  embody the underlying semantics of a timed event graph's execution. Informally, an event occurs when it observes that all of its incident events have occurred. The observation of an event is delayed with respect to its occurrence by a delay chosen non-deterministically from within the specified delay range  $[d, D]$  for the incident edge. Formally, the semantics are based on the infinite set of inequalities represented via the unfolded graph. Due to the repetitive nature of the system, these inequalities can be described using a folded TEG. The delay values for the edges describe the temporal constraints which the system meets and the  $\varepsilon$  values describe the unfolding.

The problem we address, informally, is to determine what additional temporal constraints can be inferred from the infinite set of inequalities. More

formally, given two events,  $t^{from}$  and  $t^{to}$  in  $E'$ , and a separation in occurrence index  $\beta$ , we determine the largest  $\delta$  and the smallest  $\Delta$  such that  $\forall k \geq \max(0, \beta)$ :

$$\delta \leq \tau(t_k^{to}) - \tau(t_{k-\beta}^{from}) \leq \Delta.$$

For example, to determine the bounds on the time separation between two consecutive  $a$  transitions in Figure 1, we would set  $t^{from} = t^{to} = a$  and  $\beta = 1$ , and consider the bounds on  $\tau(a_k) - \tau(a_{k-1})$ .

Our model is fairly restrictive, in that it contains no conditional behavior (choice) and the TEG must be strongly-connected (except for the root portion). Extensions for handling non-strongly connected graphs are discussed in [11] and choice in [9]. The sum of the  $\varepsilon$  values on every cycle in a well-formed graph must be positive (this eliminates any potential for deadlock).

### Problem Solution

In this section, we give an overview of the TSE algorithm (details can be found in [3, 10]). We address only the problem of finding the maximum separation,  $\Delta$ , since the minimum separation,  $\delta$ , can be obtained from the analysis of  $\tau(t_k^{from}) - \tau(t_{k-(-\beta)}^{to}) \leq -\delta$ .

Let  $\Delta_\alpha$  be the strongest bound for the separation problem given a particular occurrence index  $\alpha$ , i.e.,  $\tau(t_\alpha^{to}) - \tau(t_{\alpha-\beta}^{from}) \leq \Delta_\alpha$ . We can determine  $\Delta_\alpha$  by applying a simple algorithm [13] to a finite portion of the unfolded TEG created by only including the vertices for which there is a path to either  $t_\alpha^{to}$  or  $t_{\alpha-\beta}^{from}$ . Let  $G_\alpha = \langle E_\alpha, R_\alpha \rangle$  denote this graph. The algorithm takes time linear in the size of  $G_\alpha$ .

Since we are interested in the bound on the separation in time of two events over *all* occurrences, we would need to apply this algorithm an infinite number of times to be certain that we have found the pair of events that exhibit the worst-case behavior, i.e.,  $\Delta = \max\{\Delta_\alpha \mid \alpha \geq \max(0, \beta)\}$ . As the TEG is a repetitive system, presumably the  $\Delta_\alpha$  values will eventually reach a steady state, for example,  $\Delta_{\alpha+1} = \Delta_\alpha$  for large  $\alpha$ . Unfortunately, the behavior of the  $\Delta_\alpha$  values can be non-monotonic and periodic, and might even start out periodic and then later stabilize to a constant value (see [12]). We believe no simple termination criteria exists for this approach.

Instead, we formulate the simple algorithm in terms of an algebra that will permit us to *implicitly* analyze the infinite graph. We introduce an algebraic structure  $(\mathcal{F}, \oplus, \otimes, \bar{0}, \bar{1})$  which is similar to the  $(\max, +)$ -algebra [4, 7]. Each element in  $\mathcal{F}$  is a piecewise-linear, monotonically non-decreasing function represented by a set of pairs. The singleton set,  $\{\langle l, w \rangle\}$ , represents the function  $f(x) = \min(x + l, w)$ . In general, the set  $\{\langle l_1, w_1 \rangle, \langle l_2, w_2 \rangle, \dots, \langle l_n, w_n \rangle\}$  corresponds to the function

$$f(x) = \max \{ \min(x + l_i, w_i) \mid 1 \leq i \leq n \}.$$

Function maximization,  $f = g \oplus h$ , is defined as  $f(x) = \max(g(x), h(x))$ , and function composition,  $f = g \otimes h$ , is defined as  $f(x) = h(g(x))$ . The elements  $\bar{0}$  and  $\bar{1}$  are the identity elements for function maximization and composition, respectively. We associate a function,  $f_r$ , with each edge  $u_{k-\varepsilon} \xrightarrow{[d, D]} v_k$  in  $G_\alpha$ :

$$f_r = \begin{cases} \{\langle D - m(u_{k-\varepsilon}) + m(v_k), 0 \rangle\} & \text{if } v_k \rightsquigarrow t_{\alpha-\beta}^{from} \\ \{\langle D - m(u_{k-\varepsilon}) + m(v_k), \infty \rangle\} & \text{if } v_k \not\rightsquigarrow t_{\alpha-\beta}^{from} \end{cases}$$

where  $m(v_k)$  is the longest path from  $v_k$  to  $t_{\alpha-\beta}^{from}$  using the lower delay bounds of the edges:

$$m(v_k) = \begin{cases} \max \{d(h) \mid \text{all paths } v_k \xrightarrow{h} t_{\alpha-\beta}^{from}\} & \text{if } v_k \rightsquigarrow t_{\alpha-\beta}^{from} \\ 0 & \text{if } v_k \not\rightsquigarrow t_{\alpha-\beta}^{from} \end{cases}$$

where  $d(h)$  is sum of the  $d$  values of the edges on the path  $h$ . Using function composition and function maximization, we can create a function  $F_{[v_k]}$  defined recursively as

$$F_{[v_k]} = \bigoplus \left\{ F_{[u_{k-\varepsilon}]} \otimes f_r \mid u_{k-\varepsilon} \xrightarrow{f_r} v_k \in R_\alpha \right\}$$

where  $F_{[root_0]} = \bar{\mathbf{1}}$  and  $\Delta_\alpha$  is computed by  $\Delta_\alpha = F_{[t_\alpha^{to}]}(0) - m(t_\alpha^{to})$ . We can perform this construction in segments (because function composition is associative) by introducing matrices whose elements are functions. By using  $(\oplus, \otimes)$  matrix multiplication we can write  $(F_{[t_\alpha^{to}]}) = \mathbf{RST}$ , where  $\mathbf{R}$  is a row-vector relating  $root_0$  to a set of event occurrences in  $E_\alpha$ ,  $\mathbf{S}$  is a square matrix relating these event occurrences to the same events one occurrence later, and  $\mathbf{T}$  is a column-vector with functions relating these event occurrences to  $t_\alpha^{to}$ .

As an example, to find the maximum separation from  $t_{\alpha-\beta}^{from} = a_1$  to  $t_\alpha^{to} = a_2$  in Figure 1, we construct the finite acyclic graph  $G_2$ . The construction, which is depicted in Figure 2, produces  $F_{[a_2]} = \{\langle 22, 25 \rangle, \langle 24, 24 \rangle\}$  and we get  $\Delta_2 = F_{[a_2]}(0) - m(t_\alpha^{to}) = 24 - 0 = 24$ . This result confirms what a simple analysis would indicate. If  $\tau(root_0) = 0$  then we could have  $\tau(a_2) = 2 + 20 + 6 = 28$  and  $\tau(a_1) = 4$ , and this particular execution would maximize  $\tau(a_2) - \tau(a_1)$ .

For a strongly connected TEG, the  $m$ -values eventually follow a periodic pattern [4, 8]. By making  $\mathbf{T}$  represent a sufficiently large portion of the

unfolded TEG,  $\mathbf{S}$  is identical for all subsequent unfoldings, making it possible to reuse  $\mathbf{S}$ . If  $\Delta_{\alpha 0}$  can be found from  $\mathbf{R T}$ , we can compute  $\Delta_{\alpha 0+1}$  from  $\mathbf{R S T}$ , and  $\mathbf{R S}^n \mathbf{T}$  computes  $\Delta_{\alpha 0+n}$  for any  $n \geq 0$  (this is a slight over simplification). The maximum over all  $\Delta_{\alpha 0+n}$  is

$$\bigoplus_{n \geq 0} \mathbf{R S}^n \mathbf{T} = \mathbf{R} (\mathbf{I} \oplus \mathbf{S} \oplus \mathbf{S}^2 \oplus \mathbf{S}^3 \oplus \dots) \mathbf{T} = \mathbf{R S}^* \mathbf{T}.$$

A matrix closure algorithm [1] can be used to compute  $\mathbf{S}^*$  because  $(\mathcal{F}, \oplus, \otimes, \bar{0}, \bar{1})$  forms a closed semiring. The closure takes  $O(n^3)$  time, where  $n$  is the dimension of  $\mathbf{S}$ . Note that the algorithm is guaranteed to terminate *and* find the tightest possible bound. Complete details regarding the TSE algorithm can be found in [10] which explores other application areas, such as asynchronous circuit synthesis. It should be noted that for reasonably sized examples (less than 100 events) execution times for the TSE algorithm implemented on a SPARC II are typically a few CPU seconds.

## Modeling Manufacturing Systems: An Example

In a manufacturing context, the events in a timed event graph are typically used to represent the start and completion points of machine processing or part transportation. The edges contain delay information, such as bounds on the processing time or transportation time. The execution model for timed event graphs implicitly models synchronization, in that an event cannot occur (e.g., a machine start processing) until all incident events have been observed (e.g., all needed parts have been transported).

This model subsumes previously studied fixed delay models, and the non-determinism allows natural variations in temporal behavior to be specified

and accounted for. Admittedly, some systems may not be realistically boundable (e.g., there is always a small probability of encumbering further delay). For these systems, our algorithm reports on the behavior that would be guaranteed, assuming the delays remain within the specified bounds. Our model is certainly a simplistic one, containing only synchronization and delay. Many complexities present in real manufacturing systems would need to be handled via abstraction or by more sophisticated models. There is a definite need for additional research if more complex systems are to be analyzed. In the remainder of this section, we present an example which demonstrates that non-stochastic timing verification is necessary and useful. Although several interesting modeling issues are discussed via our example, the Petri net literature contains many articles which describe how to use Petri nets to model manufacturing systems (e.g., [2], [15]) and how to automate the synthesis of these net models (e.g., [16]). Our algorithm can be used to analyze a variety of models developed using these techniques.

### **Example**

A simple production line is shown in Figure 3. Incoming parts arrive at  $a$  and eventually exit at  $g$ . Two circular tracks are used to transport the parts on dedicated carts/pallets. The event  $c$  represents the location where parts are moved from the first track onto the second. The diagram indicates the initial behavior, i.e., we start with one empty cart on the first track, and two empty carts on the second. Because we assume an infinite supply of parts, the number of carts has a direct impact on production capability, as it limits the rate at which materials can be processed.

Bounds for the machine processing times and the transportation times for the system are shown in Table 1. The processing time for  $M_2$  is fixed, whereas  $M_1$  and  $M_3$  are variable. The transportation times are either fixed or variable depending upon whether or not any intersections must be crossed (i.e., we have chosen to use conservative bounds on transportation times, rather than introduce additional complexities). In order to simplify our example, all activities that do not appear in Table 1 are assumed to take no time (e.g., loads, unloads, and the transfer).

The labels associated with the machining operations in Figure 3 will become events in the TEG used to model this system. Figure 4 contains the complete TEG for this system. The TEG and the graphical depiction in Figure 3 share important similarities but are also quite different. Some of the edges in the TEG are used to formally specify several important characteristics of our example which we have not yet described. Like Figure 3, however, the TEG does consist of two separate cycles joined at the transfer point  $c$ . The cycle from  $a$  to  $b$  to  $c$  back to  $a$  models the first track, and the cycle from  $c$  to  $g$  back to  $c$  models the second. The edges on the two cycles are annotated with delay bounds specified in Table 1. For example,  $M_1$  has a minimum processing time of 2 and a maximum processing time of 4. This results in the edge  $a \xrightarrow{[2,4],0} b$  which formally specifies the inequality:  $\tau(a_k) + 4 \geq \tau(b_k) \geq \tau(a_k) + 2$ . Note that there are two empty carts on the second track and thus  $\varepsilon = 2$  for the edge from  $g$  to  $c$  in order to properly constrain the behavior. Thus, event  $c_k$ , which represents the  $k^{\text{th}}$  transfer of a part from the first track to the second, cannot occur before event  $g_{k-2}$ , which represents the  $(k - 2)^{\text{th}}$  completion of a part by  $M_3$ .

The self-loop at  $a$  is used to model acquisition of incoming parts, and might constrain the performance of the system if additional carts are placed on the two tracks. We assume there is only one vehicle, not shown in Figure 3, fetching raw materials for  $M_1$  to process. Thus event  $a$ , which represents the start of  $M_1$ , can occur only if parts have been delivered, and there is an empty cart to put them on. We now discuss two important issues which explain the additional edges present in the TEG. These issues involve modeling techniques that are well known in the Petri net literature, but each will be discussed in some detail for completeness.

First, we must address the issue of what type of manufacturing process each machine represents. Typically machines are restricted from operating upon other parts until they finish with the current one. An exception would be a manufacturing process akin to transportation, e.g., a part moving through a furnace on a conveyor belt. In our example, machines work on parts in a sequential manner, and thus there are edges from the three machine completion events ( $b$ ,  $e$ , and  $g$ ) back to the corresponding events for the start of machine processing ( $a$ ,  $d$ , and  $f$ ). For example, the edge  $b \xrightarrow{[0,0],1} a$  requires that  $\tau(a_k) \geq \tau(b_{k-1})$  which indicates that  $M_1$  cannot start work on a new part (i.e., event  $a_k$  cannot occur) until it is finished with the previous one (indicated by the occurrence of event  $b_{k-1}$ ). Nonzero entries for the delay bounds could be used if the machines could not work on new parts immediately after processing a part (e.g., they needed to cool down). Although not present in our example, an event corresponding to the completion of machine processing would have multiple outgoing edges if the process produced multiple parts which were routed to different places.

Second, we must worry about a very subtle issue: the possibility that carts pass one another due to non-deterministic transportation delays. If multiple carts are on a single track, we would like to indicate that a fast (non-delayed) cart essentially *piles up* behind a slower (delayed) cart and does not overtake it. This perspective is consistent with the fact that the non-determinism in our model for transportation is not due to different cart speeds but rather that carts can sometimes be delayed. Formally, we need to ensure that  $\tau(t_k) \geq \tau(t_{k-1})$  for any event  $t$  and occurrence  $k$ . This can easily be accomplished by the addition of self-loops with  $[0, 0]$  delay bounds. A self-loop appears at event  $c$  in our model, in order to explicitly prevent passing. We could add self-loops for every event, however they are not always needed. In fact, if the TSE algorithm can prove that  $\tau(t_k) - \tau(t_{k-1}) \geq 0$  for all  $k$ , then we know that passing is not possible, and a self-loop at event  $t_k$  would not be needed. Of course this must be proven whenever the model is changed, and it may be easier to simply include the self-loops. This analysis, however, might be very beneficial in helping to reduce models which are going to be subjected to more complicated analytical techniques such as a stochastic analysis. If the self loop has no affect on the behavior of the system, it's removal should not affect the stochastic analysis, except to decrease its complexity.

### **Analysis**

One benefit of non-stochastic analysis is that a guarantee with regards to system behavior is obtained. This guarantee might allow the manufacturing system to be optimized, in that additional equipment might have been added to the system in order to account for the temporal uncertainty. For example,

with respect to our example in Figure 3, let us assume that  $M_3$  needs to operate on parts leaving  $M_1$  within a bounded interval of time, say, at most 50 time units after leaving  $M_1$ . Perhaps  $M_1$  applies a chemical which must not evaporate before the part enters  $M_3$ , or  $M_1$  heats up the part which must remain above a specific temperature before entering  $M_3$ , etc. There are many sequential manufacturing processes which possess temporal constraints. In our example, we will assume that it is the upper bound which is critical; more formally, the manufacturing process requires  $\tau(f_k) - \tau(b_k) \leq 50$ . Other manufacturing systems might require a lower bound. In reality, situations such as this (where it might be difficult to determine if the constraint will be met) are often explicitly avoided due to a lack of understanding regarding the temporal behavior of the system. Typically the production line is instead modified, perhaps at some loss of efficiency, in order to accommodate the temporal constraints. Our analysis permits more efficient production lines to be designed, and allows unnecessary equipment to be removed (e.g., if equipment is added in order to ensure constraints which will always be satisfied).

Let us consider the temporal constraint present in our example system. If there were only two carts in the entire system (one on each track) then it can easily be seen that the constraint will always be met. We do not need the TSE algorithm for such a simple argument. Adding up the upper bounds on the path from  $b$  to  $f$  yields 35 ( $25 + 1 + 4 + 5$ ) but we must answer a potentially difficult question: “How long might we wait at the transfer point  $c$ ?” With one cart on each track, we merely need consider the previous point in time when both carts were at  $c$ . A large wait will occur if the cart on

the first track moved quickly:  $2 + 2 + 25$  (we use 25 because this was our assumption above) and the other slowly:  $1 + 4 + 5 + 14 + 15$ , thus the maximal wait in this case is 10 ( $39 - 29$ ), and thus  $\tau(f_k) - \tau(b_k) \leq 45 \leq 50$ . If we add a second cart to the second track, the maximum wait at  $c$  might go down, and the system should still operate correctly.

Let us assume that management wishes to increase production, which can be accomplished in our example by simply adding more carts to the system. A simulation study could be used to determine the benefits that would accrue from increasing the number of carts on the first and second tracks. Table 2 presents average values of  $\tau(g_k) - \tau(g_{k-1})$  obtained from simulations of the example system with varying numbers of carts on the two tracks and uniform random distributions for the delays. Each simulation ran for 10,000 time units and required approximately two seconds of CPU time.

From this analysis, we can conclude that adding additional carts to the the two tracks can improve production capacity, but only until a different bottleneck appears. In our example, it is the processing time required by  $M_3$ , i.e., [12,14], which explains the average of 13 which is obtained given sufficient numbers of carts on the two tracks. This analysis, however, does not indicate whether or not our production line will operate correctly given the temporal constraint which must be observed. Table 3 summarizes the results of the TSE algorithm which is used to determine an upper bound on the maximum separation in time between  $b$  and  $f$ , i.e., we report the smallest  $\Delta$  such that in any execution of the system, for all event occurrences  $k \geq 0$ ,  $\tau(f_k) - \tau(b_k) \leq \Delta$ . Each analysis required approximately .2 seconds of CPU time.

Some of this analysis agrees with our intuitive understanding of the system. If there is only one cart on the first track, we expected the delay to be at most 45 if there was only a single cart on the second track, and less if there were additional carts. One might also expect that increasing the number of carts on the first track could cause problems, because additional parts would be processed by  $M_1$  (and end up at  $b$ ) even though the number of carts on the second track is limiting production capacity. Thus, the maximum separation might potentially be too great, because we essentially have increased the duration of the maximum wait at the transfer point  $c$ .

Note, however, that the complex dynamic interaction that is present in the system often defies intuitive understanding. For example, if there are two carts on the first track, the only situation in which our upper bound is met is if there are three carts on the second track. We can gain confidence in the result by comparing these results to those obtained from simulation. In Table 4 we report the maximum observed separations for  $b$  to  $f$  obtained from simulating the system using uniform random distributions for the delays. Each simulation ran for 10,000 time units and required approximately two seconds of CPU time.

Note that these bounds are not always equal to those found via analysis. This is to be expected – simulation can not provide absolute guarantees as can be provided by our algorithm. It is also evident that increasing the number of carts on the tracks can affect the maximum separation, even if the production capacity is not affected by the increase.

The analysis for our example indicates that, fortunately, production capacity can be maximized, and our upper bound constraint satisfied, if there

are two carts on the first track and three carts on the second. If our analysis had indicated that maximizing production would result in the constraint always being violated, we could have used the TSE algorithm to explore possible modifications to our system. For example, one potentially important (yet simple) solution would be to simply require that we do not release empty carts all at once into the system. We could modify the initial behavior (specified using the *root* portion) to perform this study. Of course the temporal uncertainty in the system might not allow such a solution, and we might need to rely on the more drastic methodology of inserting delay. Other interesting studies could also be performed. We might wonder about the characteristics of part delivery at  $g$  (stochastic analysis could tell us the average delivery rate, but provides little insight into the minimum and maximum separation times between consecutive deliveries), or the affects of changing processing or transportation times. We could also examine separation times that are not consecutive, e.g.,  $\tau(g_k) - \tau(g_{k-100})$ , in order to ascertain whether the system contains any self-stabilizing behavior.

## Conclusion

We believe that non-stochastic analysis is useful for gaining insight into the complex temporal behavior of manufacturing systems. Although some aspects of behavior are intuitive, even relatively simple systems such as those we have described (which consist only of synchronization and delay), often exhibit unexpected behavior which cannot easily be understood. Informal analysis is particularly ineffective in analyzing these systems, and simulation cannot provide absolute guarantees. The large amount of parallelism

present in many systems makes informal analysis extremely difficult (if not impossible) and of course the parallelism exists in order to maximize efficiency. Formal analysis tools such as the TSE algorithm, which can be used to determine exact (tight) bounds on separation times between arbitrary system events, are thus quite useful. The bounds may be needed in order to verify the correct operation of a manufacturing system containing temporal constraints. Non-stochastic analysis is not nonsensical. It may very well be less important than the variety of techniques used to evaluate system performance, but we believe that it is essential, and that the TSE algorithm demonstrates both its efficacy and its efficiency.

## References

- [1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading, MA, 1974.
- [2] R. Y. Al-Jaar and A. Desrochers. Performance evaluation of automated manufacturing systems using generalized stochastic petri nets. *IEEE Transactions on Robotics and Automation*, 6(6):621–639, December 1990.
- [3] T. Amon, H. Hulgaard, S. M. Burns, and G. Borriello. An algorithm for exact bounds on the time separation of events in concurrent systems. In *Proc. International Conf. Computer Design (ICCD)*, pages 166–173, October 1993.
- [4] F. Baccelli, G. Cohen, G. J. Olsder, and J.-P. Quadrat. *Synchronization and Linearity*. Wiley Series in Probability and Mathematical Statistics. John Wiley and Sons, 1992.
- [5] F. Bacelli, G. Cohen, G.J. Olsder, and J.P. Quadrat. *Algebraic and stochastic analysis of timed discrete event systems*. Wiley, 1991.
- [6] G. Cohen, P. Moller, J.P. Quadrat, and M. Viot. Algebraic tools for the performance evaluation of discrete event systems. *Proc. of the IEEE*, 77(1):39–58, January 1989.
- [7] R. A. Cuninghame-Green. *Minimax Algebra*. Number 166 in Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, 1979.
- [8] S. Gaubert and C. Klimann. Rational computation in dioid algebra and its application to performance evaluation of discrete event systems. In *Algebraic computing in control*, Lecture Notes in Computer Science # 165. Springer Velag, 1991.
- [9] H. Hulgaard and S. M. Burns. Bounded delay timing analysis of a class of CSP programs with choice. In *International Symposium on Advanced Research in Asynchronous Circuits and Systems*, November 1994.
- [10] H. Hulgaard, S. M. Burns, T. Amon, and G. Borriello. An algorithm for exact bounds on the time separation of events in concurrent systems. *IEEE Transactions on Computers*. To appear 4th quarter of 1995.

- [11] H. Hulgaard, S. M. Burns, T. Amon, and G. Borriello. Practical applications of an efficient time separation of events algorithm. In *Proc. International Conf. Computer-Aided Design (ICCAD)*, pages 146–151, November 1993.
- [12] H. Hulgaard, S. M. Burns, T. Amon, and G. Borriello. An algorithm for exact bounds on the time separation of events in concurrent systems. Technical report, University of Washington, Department of Computer Science and Engineering, February 1994. TR #94-02-02 (available via anonymous ftp: `cs.washington.edu:tr/1994/02/UW-CSE-94-02-02.P.Z`).
- [13] K. L. McMillan and D. L. Dill. Algorithms for interface timing verification. In *1992 IEEE International Conference on Computer Design: VLSI in Computers and Processors*, October 1992.
- [14] T. Murata. Petri nets: properties, analysis, and applications. *Proceedings of the IEEE*, 77(4):541–580, April 1989.
- [15] Y. Narahara and N. Viswanadham. A petri net approach to the modelling and analysis of flexible manufacturing systems. *Annals of Operations Research*, 3:449–472, 1985.
- [16] M. Zhou, F. DiCesare, and A. Desrochers. A hybrid methodology for synthesis of petri net models for manufacturing systems. *IEEE Transactions on Robotics and Automation*, 8(3):350–361, June 1992.

## **Author's Biographies**

Tod Amon received his Ph.D. in Computer Science from the University of Washington, Seattle, in 1993. He is currently an Assistant Professor in the Department of Computer Science at Southwest Texas State University. Prior to attending graduate school he was a Senior Software Engineer in a Computer Aided Manufacturing department within General Dynamics. He was the Project Manager for two separate projects which emphasized production-line analysis via simulation and analytical methodologies. He received an NSF Research Initiation Award in 1994 to support his efforts and interest in the temporal analysis of concurrent systems.

Henrik Hulgaard received the M.S. degree in electrical engineering from the Technical University of Denmark in 1990 and the M.S. degree in computer science from the Department of Computer Science and Engineering, University of Washington, Seattle, in 1992. He is presently a Ph.D. candidate in computer science at the University of Washington. His research interests include formal verification, asynchronous circuits, computer aided design, and timing verification.

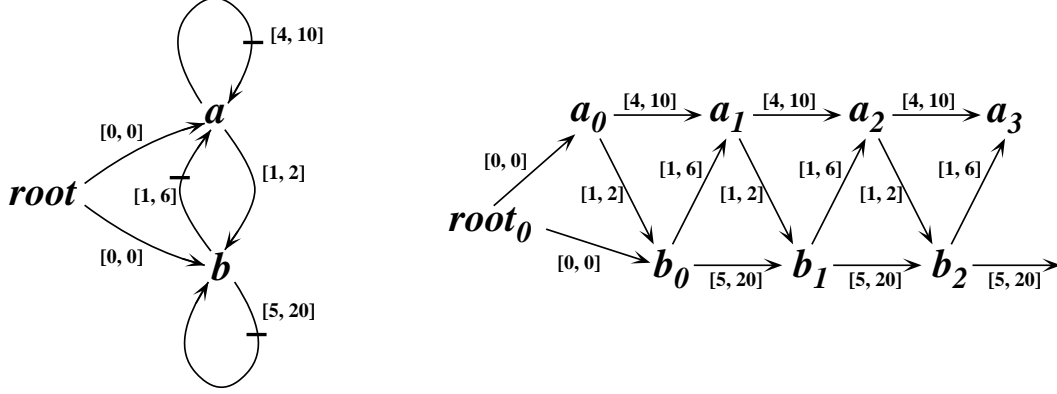


Figure 1: A timed event graph and part of its unfolding.

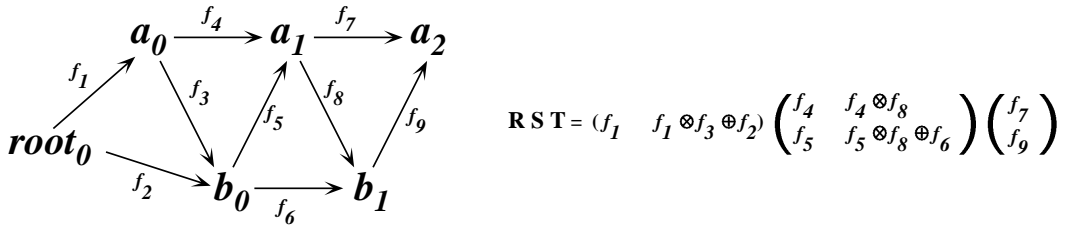


Figure 2: Construction of  $F_{[a_2]}$  using matrices.

Activity	Time required	
	minimum	maximum
arrival frequency for new parts	5	10
processing on $M_1$	2	4
move from $M_1$ to transfer point	15	25
move empty cart to $M_1$	2	2
move from transfer point to $M_2$	1	1
processing on $M_2$	4	4
move from $M_2$ to $M_3$	2	5
processing on $M_3$	12	14
move empty cart to transfer point	10	15

Table 1: Processing and transportation times for the example system.

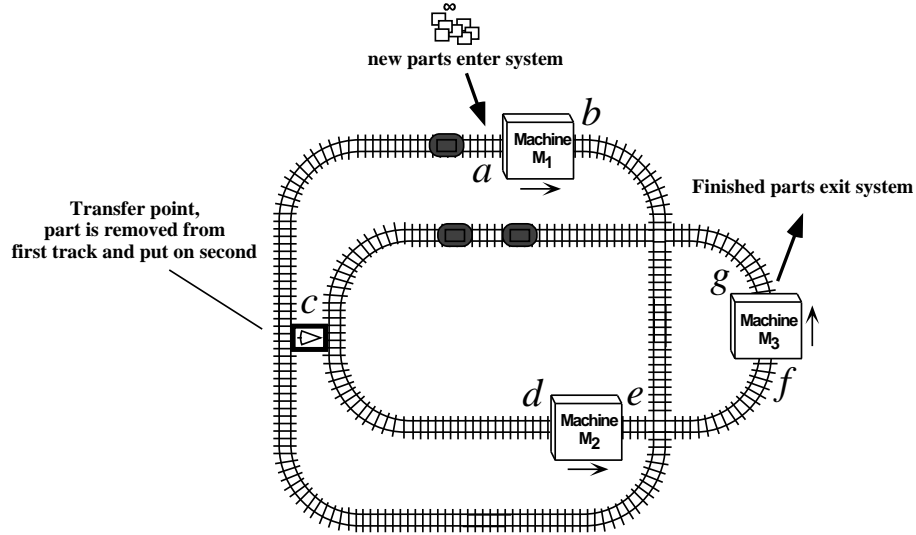


Figure 3: Example manufacturing system.

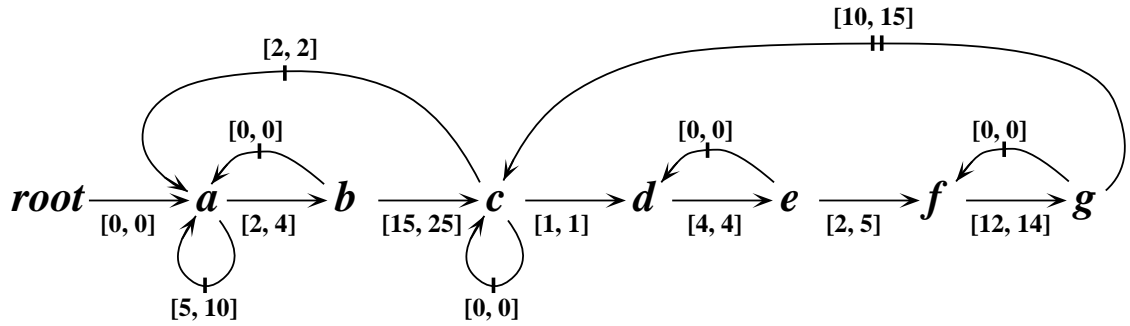


Figure 4: TEG for the example manufacturing system.

Carts on first track	Carts on second track			
	1	2	3	4
1	33	24	24	24
2	33	17	13	13
3	33	17	13	13

Table 2: Average values of  $\tau(g_k) - \tau(g_{k-1})$  obtained from simulation.

Carts on first track	Carts on second track			
	1	2	3	4
1	45	35	35	35
2	84	54	46	56
3	123	79	66	71

Table 3: Upper bounds on  $\tau(f_k) - \tau(b_k)$  obtained from the TSE algorithm.

Carts on first track	Carts on second track			
	1	2	3	4
1	43	35	35	35
2	81	47	41	54
3	117	72	55	67

Table 4: Upper bounds on  $\tau(f_k) - \tau(b_k)$  obtained from simulation.