Analyzing Non-deterministic Real-Time Systems with (max,+) Algebra *

Guillaume P. Brat
The University of Texas at Austin
Dept. of Electrical and Computer Engineering
Austin, Texas 78712, USA
gbrat@pine.ece.utexas.edu

Vijay K. Garg
The University of Texas at Austin
Dept. of Electrical and Computer Engineering
Austin, Texas 78712, USA
garg@ece.utexas.edu

Abstract

We describe a hierarchical technique that allows a class of non-deterministic timed Petri nets to be analyzed using the (max,+) algebra of periodic signals. We show that the timing and controllability analysis of such systems is possible via the use of (sup- and) inf-convolution operations within the (max,+) framework. We apply this technique to the verification of timing constraints in an intelligent structural control system and compare our technique to other modeling tools for real-time systems.

1. Introduction

There is a growing body of work showing the usefulness of the (max,+) algebra [1, 3, 4, 13, 7]) to analyze timed event graph models and deterministic real-time systems. The strongest criticism about these approaches is their failure to cope with non-deterministic systems. This work extends the (max,+) algebra of periodic signals [4] to the analysis of non-deterministic real-time systems.

We are aware of only one work that has applied the (max,+) algebra to non-deterministic, timed Petri nets [6]. In that work, the (max,+) algebra is applied to variable routing event graphs in which places are allowed to have many successors under the following restrictions. First, the function allocating tokens to the successors of a given place has to be deterministic. Second, places with multiple successors are not allowed within cycles. Our work aims at lifting those two restrictions, thus facilitating the analysis of non-deterministic systems.

We propose a hierarchical technique that replaces subnets with non-deterministic features by single places with delays that capture the non-deterministic behavior. The result of this reduction is a deterministic Petri net that is analyzable using the (max,+) algebra of signals. This technique applies to a class of real-time systems that are also analyzable by well-known modeling tools such as Modechart [18], HYTECH [17], timed automata [16], Uppaal [19], KRONOS [10], SCR [15], CSR [14], PARAGON [2] and other techniques based on temporal logics. Due to length limitations, we limit our comparison to Modechart which is well known and has been compared to the other above-mentioned techniques.

There are two main advantages to the (max,+) algebra of signals over these existing techniques. First, unlike existing techniques, our algebra does not require the construction of large state spaces to prove timing properties. For example in Modechart, a state consists of a combination of the active modes at a given time and the values of the global variables at that time. This potentially leads to large graphs with sizes that are usually reduced by pruning based on the timing constraints in the modeled system. In the (max,+) algebra, the behavioral information of a system is captured by $N \times N$ closure matrices where N is the number of events in the modeled system. These matrices, which are built using an $O(N^3)$ algorithm, contain the necessary information to compute all time occurrences of all the events in the system. Moreover, this technique tolerates system decomposition which can further reduce the cost of the analysis [5]. These features make the (max,+) algebra of signals less prone to state explosion than other traditional techniques. Second, the analysis techniques available in the (max,+) algebra of signals include algorithms to compute optimal controllers for real-time discrete event systems (DES). Similar work can be done using symbolic methods within the framework of timed automata [11]. However, we are not aware of any efficient implementation of that work yet. Control synthesis is not demonstrated in this paper, but examples are given in [3, 5, 7, 8].

Unfortunately, there are some drawbacks to our technique. The (max,+) algebra of signals applies only to these systems in which events are eventually periodic. Therefore, our technique handles sporadic events in a manner similar

*supported in part by the NSF Grants ECS-9414780, CCR-9520540, TRW faculty assistantship award, a General Motors Fellowship, and an IBM grant

to the way sporadic tasks are handled in scheduling algorithms for periodic tasks; the timing behavior of a sporadic event has to be bound by periodic signals. Note also that HYTECH, SCR, and timed automata have been applied to the verification of hybrid systems, which is not possible at this time with our technique.

The organization of this paper is as follows. Section 2 explains our motivation for extending the current model. Section 3 provides background information on the (max,+) algebra and timed event graphs. Then, Section 4 describes our analysis technique, the role of convolution in capturing non-determinism and our reduction technique. Section 5 explains how to perform controllability analysis of non-deterministic systems. Section 6 describes our implementation. Section 7 details the analysis of an intelligent structural control system originally described in [12]. Finally, Section 8 presents our conclusions and future work.

2. Motivation

As a motivating example, we apply our analysis technique to an intelligent structural control system initially defined in [12] using Modechart for specification and Temporal CCS for verification. The goal of this system is to counteract the effects of earthquakes on buildings. It computes forces that need to be applied to a structure to counter its external excitations.

The system consists of three interacting components. A sensor monitors the state of the system (e.g., accelerations and displacement); a controller computes the appropriate forces needed to counter external excitations of the structure, and an actuator applies these forces to the structure. The algorithm described in [12] is a pulse control algorithm. Pulses are sent to the structure (via the actuator) to control the structure's excitation. The magnitude of each pulse is computed based on the state variables of the system. The pulse control algorithm provides satisfactory performance if the time between pulses is bounded by $T_0/8$ and $T_0/2$ where T_0 is the natural period of the structure (in this case, $T_0 = 290$ time units (t.u.)). Moreover, it is required in [12] that 135 t.u. elapse between two consecutive pulse calculations. The goal of the analysis is to ensure that both conditions are met.

Figure 1 illustrates the timed Petri net (PN) for the whole system. Time is associated with places, i.e, a token cannot participate in the enabling of the downstream transitions of a place until it has spent a minimum amount of time in that place. For readability, zero delays are omitted in Figure 1. Observe that the subnet for the controller (area delimited by dotted lines) includes some non-deterministic choices (transitions t_5 and t_{11} have the same upstream place). It also synchronizes with two deterministic subnets representing the sensor (area with evenly dashed lines) and the actuator

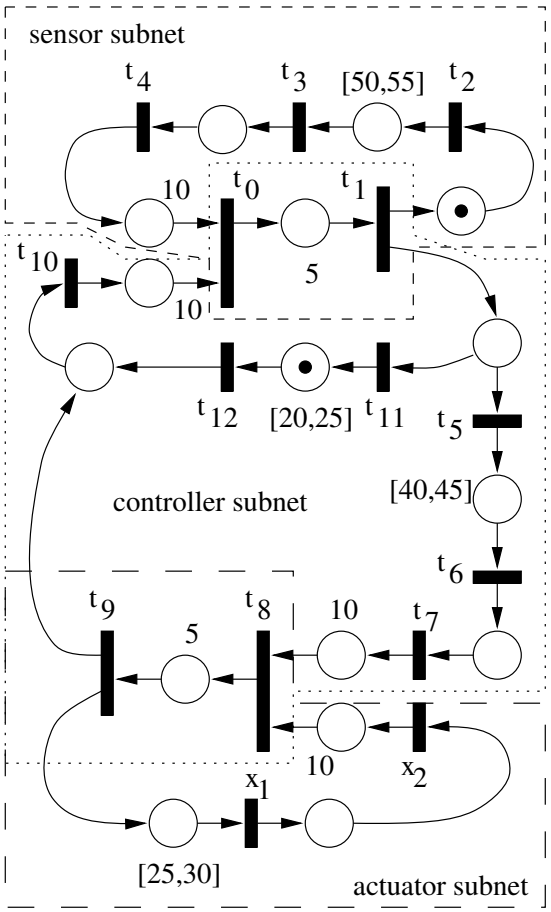

Figure 1. Timed Petri net for an intelligent structural control system

(area with extended dashed lines). The subnet between transitions t_5 and t_9 describes the control actions taken upon getting values from the sensors. The pulse control algorithm computes control values that are then transmitted to the actuator via a synchronization on transition t_8. If the minimum separation time (135 t.u.) constraint on successive calculations of control values is not satisfied, the values acquired by the controller are discarded and the controller goes through an internal updating phase; this is represented by the subnet between transitions t_{11} and t_{12}.

The (max,+) algebra of signals provides an interesting analysis framework to analyze the timing behavior of real-time DES that can be modeled by timed event graphs (TEG) [4, 5]. Unfortunately, TEG belong to a class of deterministic timed PN in which places have exactly one upstream transition and one downstream transition. This is not the case in our example and many other practical real-time systems. Therefore, in this work, we extend the (max,+) algebra of signals so that it applies to non-deterministic timed Petri

nets (which we call non-deterministic timed event graphs).

3. Background

3.1 Introduction to (max,+) algebra

This section is a short introduction to the (max,+) algebra for DES. Further details can be found in [1, 7, 13].

Let $\varepsilon = -\infty$, $e = 0$, and Z be the union of the set of integers and $\{\varepsilon\}$. Let $+$ be the conventional addition on Z given that, for all $a \in Z$, $a + \varepsilon = \varepsilon + a = \varepsilon$. Define two binary operations in Z, $\oplus$ and $\otimes$, such that, for all $a, b \in Z$, $a \oplus b = \max(a, b)$ and $a \otimes b = a + b$. The $\oplus$ operator is commutative, associative, idempotent, and ε is its identity element. The $\otimes$ operator is associative, distributive over $\oplus$, e is its identity element, and ε is absorbing with respect to $\otimes$. $(Z, \oplus, \otimes)$ forms a dioid called the (max,+) algebra.

In [1, 7, 13], the (max,+) algebra has been applied to the modeling of DES. Systems are described by matrices (called delay matrices) that defines delays between events. Therefore, indices in delay matrices correspond to events in DES. Let x be the vector representing the time sequences corresponding to all (say N) events in a DES. Let v be a vector defining the earliest times at which the events can occur. Let A be the delay matrix. Then, x is a solution to the following set of equations:

$$x_i = \bigoplus_{1 \le j \le N} A_{ij} x_j \oplus v_i, \text{ where } 1 \le i \le N.$$

According to [9], this system of equations can be written as

$$x = Ax \oplus v,$$

the least solution of which is $A^* v$ where $A^* = \bigoplus_{k \ge 0} A^k$. The closure matrix, A^*, gives the maximum delay between transitions along infinite paths. The solution $x = A^* v$ gives the earliest times at which the transitions may fire.

We illustrate these concepts with a manufacturing process example (see Figure 2). Upon arrival ($x_1(k)$ is the time of the k^{th} arrival), parts are first set-up in a machine queue, and then worked in order of arrival. Let $x_2(k)$ represent the departure of the k^{th} part from the queue, and $x_3(k)$ its completion time. Each action (i.e, setting the part (s), working it (w), and resetting the machine (r)) takes a certain amount of time. Moreover, the part interarrival time is represented by a. Then, assuming that $r(0) = \varepsilon$, the system is described by the following equations.

$$\begin{aligned} x_1(k) &= x_1(k-1) + a(k-1) + v_1 \\ x_2(k) &= \max\{x_1(k) + s, x_3(k-1) + r(k-1)\} + v_2 \\ x_3(k) &= x_2(k) + w + v_3 \end{aligned}$$

which translates into the following matrix form:

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} a\gamma & \varepsilon & \varepsilon \\ s & \varepsilon & r\gamma \\ \varepsilon & w & \varepsilon \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} \oplus \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}.$$

The backshift operator, denoted by γ, is used when the value of a state variable at index k depends on the value of a state variable at index $k-1$ (as in the expression for x_1 and x_2).

In [4, 5], we define a (max,+) algebra over the set $\mathcal{Z}$ of periodic signals. Signals are finite representations of ultimately periodic, infinite sequences. A signal is defined by two lists of integers capturing the transitory part (or aperiodic finite subsequence) and the period of a periodic sequence. Each element of these lists represents a time difference between two consecutive values of the sequence. For alternative finite representations of infinite sequences see [13]. For example, the sequence

$$X = \{1, 3, 6, 9, 12, \ldots\}$$

is represented by the signal

$$x = ((1, 2); (3)).$$

Note that symbols for signals are in lower case and that the corresponding sequences are the same symbols in upper case.

In [4, 5], delays can vary in a periodic manner over time. Therefore, they are also described by periodic signals (called delay signals). Delay signals are subject to the following restrictions: the sum of their periodic steps is equal to 0 and the values in their underlying sequences cannot be negative. The delay function associated with a given delay, say d, is written as δ_d. Delays in the manufacturing process example are not always constant. For example, on one hand, the interarrival time a varies over time. It alternates between five and seven. Therefore, the associated delay function is δ_a where $a = ((7); (-2, 2))$, which corresponds to the delay sequence $A = \{7, 5, 7, 5, \ldots\}$. On the other hand, The setting time s is constant and equal to one. Therefore, the corresponding delay function is δ_s where $s = ((1); (0))$, which corresponds to $S = \{1, 1, 1, 1, \ldots\}$. The other delays are characterized by $w = ((4); (0))$ and $r = ((); (4, -3, 0, 0, 0))$. The effect of γ on a signal is to increment all indices by one, e.g., $\gamma[((1, 2); (3))] = ((\varepsilon, 1, 2); (3))$. In the manufacturing process example, the equations yield the sequence

$$X_3 = \{5, 10, 17, 22, 29, 37, 42, 47, 53, 58, 66, 71, 77, \ldots\}$$

which represents the times of occurrence for the event x_3. The corresponding signal is

$$x_3 = ((5, 5, 7); (5, 7, 8, 5, 5, 6, 5, 8, 5, 6)).$$

3.2 Timed event graphs

This section gives a brief description of timed event graphs; further details can be found in [1, 20] among others.

An event graph is a Petri net in which places have only one upstream transition and one downstream transition. A timed event graph (TEG) is an event graph in which enabling delays are associated with places. An enabling delay, or simply a delay, is the minimum amount of time (possibly zero) a token has to spend in a place before it can participate in the enabling of the downstream transition of the place. TEGs can be used to describe the temporal behavior of DES. Transitions model events. The temporal behavior is captured by a delay matrix. The entries of these matrices are the delays specified in the places. Therefore, the delay matrix of the manufacturing process example corresponds to the TEG in Figure 2. In our terminology, a place without incoming (outgoing) arcs is a source (sink respectively) place. Similarly, a transition without incoming (outgoing) arcs is a source (sink respectively) transition.

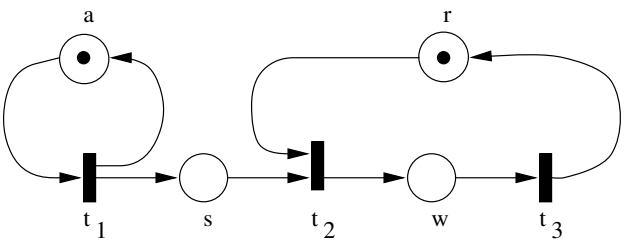

Figure 2. Manufacturing process example.

4. Non-deterministic Timed Event Graphs

This section describes how (max,+) algebra analysis techniques can be extended to a class of timed PNs with non-deterministic constructs.

Our strategy is quite simple. We identify the smallest subnet encompassing the scope of some non-deterministic constructs and replace it by a single place. The delay of this place reflects the possibility of selecting any alternative paths in the replaced non-deterministic subnet. This is done in two steps. First, reduce each alternative path to a single place (Section 4.2). Second, combine all these "alternative" places into a single place using some convolution on their delays (Section 4.3). First, we define what class of Petri nets can be analyzed with our technique.

4.1. Model definition

We now define our non-deterministic extension to TEGs. Figure 3 representing the intelligent structural control system without the actuator illustrates our definitions.

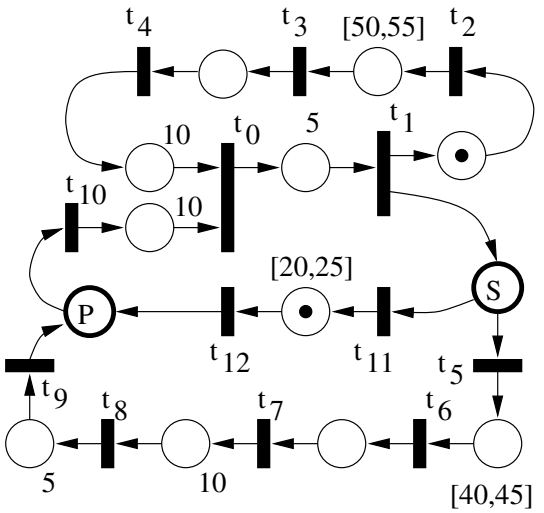

Figure 3. Non-deterministic TEG for the intelligent control system

Definition 1 (Balanced Acyclic TEG)
A balanced, acyclic TEG, say B, is a TEG without any directed cycles such that

- *B has one source transition (called $sourcetr(B)$) and one sink transition (called $sinktr(B)$), and*
- *any two paths between any two transitions have the same number of tokens.*

In Figure 3, there are two balanced, acyclic TEGs. The first one is the path between transitions t_{11} and t_{12}. The second one is defined by the path from transition t_5 to transition t_9.

Definition 2 (Alternative TEG)
An alternative TEG consists of

- *a source place and a sink place with no initial tokens,*
- *a finite set $\mathcal{B}$ of balanced acyclic TEGs ($\mathcal{B} = \{B_1, B_2, \ldots, B_n\}$), and*
- *for all $B_i \in \mathcal{B}$, there are arcs from the source place to $sourcetr(B_i)$ and from $sinktr(B_i)$ to the sink place.*

In Figure 3, there is one alternative TEG (corresponding to most of the controller) whose sink and source places (shown in bold) are named P and S.

Definition 3 (Non-Deterministic TEG)
A non-deterministic TEG is a PN that can be derived from a TEG by finite applications of the following rules:

- *pick a place, and*
- *replace it by an alternative TEG.*

The Petri net described in Figure 3 is a non-deterministic TEG.

Observe that the non-deterministic behavior started in the source place of an alternative TEG ends at its sink place. The absence of connectivity between balanced, acyclic TEGs of a same alternative TEG is motivated by the desire to avoid deadlocks. If a deadlock occurs within an alternative TEG, it results from a deadlock within one of its balanced, acyclic TEGs, and not from interactions between the balanced, acyclic TEGs.

4.2. Reduction of balanced acyclic TEGs

This subsection studies the reduction of TEGs to nets with a single source (sink) transition and a single place with a, possibly, non-zero delay. As described in [4], star-delay operations have to be used to analyze cycles in TEGs. Since the computation of a star-delay operation depends on its input signal, subnets with cycles are not reducible, hence they are excluded from the definition of acyclic TEGs.

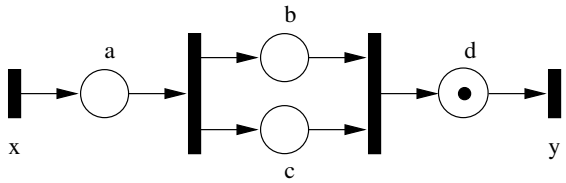

Figure 4. Example of an acyclic TEG

Consider an acyclic TEG with a single source transition and a single sink transition. The structure of this TEG consists of the places, and transitions composing the graph as well as the initial tokens present in places. Such a net can be completely described by a (max,+) expression consisting of max, delay and backshift operations. For example, the net displayed in Figure 4 can be described by the following (max,+) expression where x and y are the input and output signals respectively and a, b, c, and d are delay signals:

$$y = \delta_d \gamma(\delta_b \oplus \delta_c)\delta_a[x]$$

Definition 4 (Reducibility)
An acyclic TEG with a single source transition x and a single sink transition y is reducible if its (max,+) expression can be written

$$y = \delta_a \gamma^i[x] \quad (1)$$

where a is a delay and i is a positive integer.

The interpretation of $y = \delta_a \gamma^i[x]$ is that of a net with a single source transition x, a single sink transition y, and a single place with delay signal a and i initial tokens. Any expression of this form is called a primitive form.

Theorem 1 *Any balanced acyclic TEG is reducible.*

Proof:
First, observe that the composition of two delay operations can be reduced to a single delay operation, i.e., let d and d' be two delay signals and x be an input signal,

$$\delta_d(\delta_{d'}[x]) = \delta_{d \odot d'}[x]$$

where $d \odot d' = \delta_d[d']$.

As shown in [5], for any delay signal d and any input signal x,

$$\gamma\delta_d[x] = \delta_{(\gamma d)}[\gamma(x)].$$

Therefore, the composition of delays and backshift can be reduced to a primitive form. Also,

$$\forall x, y \in \mathcal{Z}, x \oplus y \in \mathcal{Z}, \text{ and}$$

$$\forall x, y \in \mathcal{Z}, \gamma(x \oplus y) = \gamma(x) \oplus \gamma(y).$$

Therefore, the composition of delays and backshift as operands of a maximization can be reduced to a primitive form as long as the numbers of backshifts in the operands match.

■

Note that if the numbers of backshifts are different in operands of a maximization, then the expression cannot be reduced to a primitive form. Consider the expression

$$y = (\delta_a \gamma \oplus \delta_b)[x]$$

where a and b are two delay signals and x is an input signal. Let A, B, and X be the underlying sequences of a, b, and x respectively. For $k \geq 2$,

$$\begin{aligned} Y[k] &= (A[k] + X[k-1]) \oplus (B[k] + X[k]) \\ &= (A[k] + X[k-1] - X[k] + X[k]) \oplus \\ &\qquad (B[k] + X[k]) \\ &= ((A[k] + X[k-1] - X[k]) \oplus B[k]) + X[k] \end{aligned}$$

This shows that $Y[k]$ can not be obtained by adding a delay, independent of X, to $X[k]$. Therefore, this expression is not reducible. For example, if there is an initial token in place b, but not in place c, in the acyclic net in Figure 4, then the net is not reducible.

4.3. Convolution of primitive forms

In this section, we assume that balanced, acyclic TEGs are reduced to single places each characterized by a delay as shown in the previous section.

We now define the cumulative delay $C(d,i)$ at index i of a delay d (with underlying sequence D) as

$$C(d,i) = \sum_{j=1}^{i} D[j].$$

$C(d,i)$ is the cumulative sum of all the delays incurred after i tokens have passed through the place of delay d. We call cumulative delay sequence $C(d)$ the sequence $\{C(d,1), C(d,2), \ldots\}$. It is equivalent to the sequence obtained by applying a delay sequence to a zero input signal. For example, consider the delay signal

$$a = ((7);(-2,2))$$

in the manufacturing example. Its underlying delay sequence is

$$A = \{7,5,7,5,7,5,\ldots\}.$$

The associated cumulative sequence is given by

$$C(a) = \{7,12,19,24,31,36,\ldots\}.$$

Our goal is to eventually replace every alternative TEG by a single place. To characterize the delay of such a place, we examine the cumulative delay sequence within an alternative TEG. Let d be the delay signal associated with an alternative TEG (consisting of two balanced, acyclic TEGs in their primitive forms consisting of delay signals d_1 and d_2) once it has been reduced to a single place. The convolutive sum $C(d,i,j)$, where

$$C(d,i,j) = C(d_1,j) + C(d_2,i-j),$$

is the cumulative delay at index i when the alternative delay d_1 has been selected j times while the first i tokens passed through the alternative TEG. $C(d,i,j)$ is the result of applying one of all the possible decision strategies in the place creating the non-determinism. Obviously, $C(d,i)$ is a function of the $C(d,i,j)$. Since $C(d,i,j)$ is under a convolutive form, we view $C(d,i)$ as the result of applying some type of convolution on the cumulative sums of the delays of the balanced, acyclic TEGs of the alternative TEG. In fact, Section 5 shows that the inf-convolution and the sup-convolution (obtained by taking the inf, and sup respectively, of the $C(d,i,j)$ at each index i) are two convolutions that play a critical role in computing the controllability of lower and upper bound specifications.

Let $\oslash$ be a commutative, associative operator on integers such that the traditional arithmetic addition distributes over $\oslash$, e.g., $\min$ or $\max$. The operator $\oslash$ defines a decision function that determines a path (hence, a delay signal d) in the alternative TEG. The cumulative delay sequence $C(d)$ in the alternative TEG once it is reduced to the delay d from the alternative delays is defined as follows. For any $i \geq 1$,

$$C(d,i) = \oslash_{j=1}^{i} C(d,i,j).$$

The elements D, for $i \geq 1$, of the delay sequence D are given by the following equations:

$$\begin{aligned} D[1] &= C(d,1) \\ D[i] &= C(d,i) - C(d,i-1), \text{ for } i > 1 \end{aligned}$$

The signal d is easily derived from the sequence D as described in [5].

For example, the delay signal $d_1 = ((4);(2,-2))$ has for delay sequence $D_1 = \{4,6,4,6,4,6,\ldots\}$ which yields the cumulative sequence

$$C(d_1) = \{4,10,14,20,24,30,\ldots\}.$$

Similarly, the delay signal $d_2 = ((5);(1,-1))$ has for delay sequence $D_2 = \{5,6,5,6,5,6,\ldots\}$ which yields the cumulative sequence

$$C(d_2) = \{5,11,16,22,27,33,\ldots\}.$$

The cumulative delay sequence of the signal d corresponding to the inf-convolution of d_1 and d_2 is computed by taking the inf-convolution of $C(d_1)$ and $C(d_2)$. Thus, $C(d,1) = \min\{4,5\}$, $C(d,2) = \min\{10, 4+5, 11\}$, $C(d,3) = \min\{14, 10+5, 4+11, 16\}$, and so on. Therefore, it yields

$$C(d) = \{4,9,14,19,24,29,\ldots\}$$

which corresponds to the delay sequence $D = \{4,5,5,\ldots\}$ and the delay signal $d = ((4,1);(0))$.

We now establish the fact that the convolution of periodic signals results in a periodic signal. First, define a pure periodic delay signal as a signal whose underlying sequence has no transitory subsequence (hence, in the signal form, the list for the transitory sequence contains only one step).

Lemma 1 *The convolution of two pure periodic delay signals results in a periodic delay signal.*

Proof: Let $\oslash$ be a commutative and associative operator such that the traditional arithmetic addition distributes over $\oslash$. Without loss of generality, consider two periodic signals x and y the periods of which are of the same length m. Recall that the sum of the periodic steps of a periodic delay signal is zero. Since x and y are pure periodic signals, they are fully characterized by the first m elements of their underlying sequence. Let $P_x = \sum_{i=1}^{m} X[i]$ and $P_y = \sum_{i=1}^{m} Y[i]$.

As shown above,

$$C(d,k) = \oslash_{j=1}^{k} C(d,k,j).$$

It can be shown that, for any $j,k \geq 1$,

$$\begin{aligned} \text{either,} \quad C(d,k+m,j) &= C(d,k,j) + P_x \\ \text{or,} \quad C(d,k+m,j) &= C(d,k,j) + P_y. \end{aligned}$$

Therefore,

$$C(d, k+m) = C(d,k) + P_x \oslash P_y. \quad (2)$$

Since $P_x \oslash P_y$ is constant, the cumulative delay sequence $C(d)$ is periodic, and so are the delay sequence D and the signal d.

■

Theorem 2 *The convolution of two periodic delay signals results in a periodic delay signal.*

Proof: The proof is similar to the proof of Lemma 1. Let $\oslash$ be a commutative and associative operator such that the traditional arithmetic addition distributes over $\oslash$. Assume, without loss of generality, that the signals, say x and y, are homogeneous, i.e., the lengths of their transitory sequence are equal (to n) and the lengths of their period are also equal (to m). An algorithm to homogenize signals is given in [5].

It can be shown that, for any index $k > 2n + m$, the equality (2), in the proof of Theorem 2, holds again. Therefore, the value of the $\oslash$-convolution of x and y at index k is periodic for any $k > 2n + m$.

■

5. Controllability Analysis

Using the (max,+) algebra framework, the controllability of a system can be analyzed. A controllable event is defined to be an event whose occurrences can be delayed (but not hastened). An uncontrollable event is an event that cannot be hastened nor directly delayed. The only means of delaying an uncontrollable event is to delay a controllable event that causes the delay of the uncontrollable event. Controllable events can be specified using a diagonal matrix I_c (of equal size to the transition matrix) in which diagonal elements are either ε when the event is uncontrollable or e when the event is controllable. The remaining elements of I_c are equal to ε.

If y is a vector of signals defining the control signals for all events in a system, $I_c y$ defines the valid control signals (a control signal is valid if it applies to a controllable event). Moreover, if A is the transition matrix of the system and v the initial firing times for every transition, then $A^*(I_c y \oplus v)$ represents the earliest occurrence times of the events when the system is under the control law defined by I_c and y. Thus, if y is a vector of signals specifying the desired timing behavior of the events in the system, one can compute the controllability of y with respect to the system defined by the matrices A and I_c and the initial vector v by comparing the value of $A^*(I_c y \oplus v)$ to y.

5.1. Lower and upper bound specifications

Assume that y is a vector defining a lower bound specification, i.e., the occurrence times of the events in the system are greater or equal to y. Then,

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

is a controllability test for the lower bound specification y.

Denote the inf-convolution operator by $\sqcap$. Since $\sqcap$ provides the smallest value of all the possible convolutions, it is trivial to show that, for any convolution operator $\oslash$,

$$A^*_{\oslash}(I_c y \oplus v) \geq A^*_{\sqcap}(I_c y \oplus v)$$

where $A_{\oslash}$ is the transition matrix resulting from applying $\oslash$ to the delays of the non-deterministic subnet. Therefore, a lower bound specification y is controllable, if

$$A^*_{\sqcap}(I_c y \oplus v) \geq y.$$

The convolution $\sqcup$ defined by the sup operator can also be used to compute the controllability of an upper bound specification. Given an upper bound specification y,

$$A^*_{\sqcup}(I_c y \oplus v) \leq y.$$

6. Implementation

Our goal is to automate all phases of the analysis process. We are therefore extending our tool (maxplus), which implements the (max,+) algebra of periodic signals with algorithms to compute convolution operations on delay signals, an algorithm to reduce a single input-single output, acyclic TEG to a single delay, and an algorithm to check the reducibility of a subnet

6.1. Convolution operations

It is easy to derive from the proof of Lemma 1 that the length of the period of the signal resulting from the convolution of two pure periodic signals is the lowest common multiple of the lengths of their period. Moreover, the length of the transitory sequence is bounded by the size of the period. Therefore, we can derive the algorithm in Figure 5 where m is the common length of the periods of the two input signals, and *trans* and *prev* are integer arrays of size m. The function *convolutivesum* takes for inputs two periodic signals and an index. It computes the convolutive sums of the signals at the specified index. The complexity of *convolutivesum* is $O(k)$ where k is a given index. Therefore, the complexity of *convolution* is $O(m^2)$.

Using results from the proof of Theorem 2, the *convolution* algorithm can be modified to handle any periodic delay

```
convolution(signal x,y): signal z
{   1. previous := 0; cumul := 0;
    2. for i from 1 to 2m do
         current := convolutivesum(x,y,i) - previous;
         if (i ≤ m)
              trans[i] := current - cumul;
              cumul += trans[i];
         if (i > m)
              per[i] := current - cumul;
              cumul += per[i];
         previous += current;
    3. z := (trans;per); }
```

Figure 5. Algorithm for the convolution of periodic signals

signal without changing its overall complexity: make *trans* an integer array of size $2n$, *per* an integer array of size m, and change the tests of both *if* statements to reflect those new bounds. Still, the modified algorithm does not handle the presence of initial tokens in balanced, acyclic TEGs.

Assume that there are two balanced, acyclic TEGs corresponding to two signals x and y of underlying sequences X and Y respectively that contain n_x and n_y initial tokens respectively. It is obvious that the first $n_x + n_y$ steps of the signal z corresponding to the convolution of x and y have to result from the convolution of the first n_x steps of x and the first n_y steps of y. The remaining steps can be computed by replacing the first n_x steps of x by their sum and respectively the first n_y steps of y by their sum, and by computing the convolution of the modified signals.

6.2. Reducibility of balanced, acyclic TEGs

There are two important steps in the reduction of balanced, acyclic TEGs to single places. First, one needs to check if the subnets are reducible (i.e., no loops and a balanced number of tokens in concurrent paths). Second, the reduction itself has to take place. This can be performed by using the algebraic simplification rules discussed in Section 4.2. We now describe an algorithm to check the reducibility of a balanced, acyclic TEG.

Detecting the presence of a cycle within a subnet is an easy operation that can be performed in $O(|V| + |E|)$ operations using a depth-first search algorithm. E is the set of edges in the subnet, and V is the set of vertices. Detecting an unbalanced number of tokens in concurrent activities can also be performed using a depth-first search algorithm (with $O(|V| + |E|)$) complexity) given in Figure 6. The idea is as follows. Consider the weighted, directed graph where

- the vertices are the transitions of the acyclic TEG,
- there is an arc from a vertex to another if there is a place connecting the corresponding transitions in the TEG, and
- the weight $w(a)$ of that arc a is the number of tokens initially present in that place.

Assume that *Cnt* is an array of integers, whose elements are initialized to -1, indexed by the vertices of the graph. Define the function $w : V \times V \rightarrow N$ (where N is the set of natural number) that associates its weight to an arc. Algorithm 6

```
reducible(vertex u, int tokens): boolean
{
    1. if (Cnt[u] = −1)
       then Cnt[u] := tokens;
            for each vertex v successor of u do
                 if not reducible(v, Cnt[u]+w(u,v));
                 then return FALSE;
    2. if (Cnt[u] ≠ tokens)
       then return FALSE;
    3. return TRUE;
}
```

Figure 6. Algorithm for checking reducibility

starts with the source transition and 0 as arguments.

7. Example

In this section, we apply our analysis technique to the intelligent structural control system example defined in [12]. Our goal is to verify that the time between control pulses is bounded by $T_0/8$ and $T_0/2$ where T_0 is 290 t.u. and that consecutive calculations of the actuator values are separated by at least 135 t.u.

Figure 3 illustrates the non-deterministic TEG for the sensor and the controller. The actuator needs to synchronize with the controller. However, the timing constraints in the actuator net are such that it cannot affect the behavior of the controller. Therefore, we need not model the actuator when analyzing the timing of the sensor and the controller. This makes all the balanced, acyclic TEGs reducible. Note that all timing values are integer constants that correspond to constant signals. The constant c corresponds to the signal $((c); (0))$. Intervals represent ranges of possible delays.

The balanced, acyclic TEG for the calculation of the pulse can be reduced to a single place with a cumulative delay ranging from 55 to 60 t.u. However, the resulting PN is still not analyzable. The alternative TEG must be replaced by a single place whose delay d is the result of some

convolution operation on the delays in the balanced, acyclic TEGs. Once this is done, we obtain the reduced TEG in Figure 7.

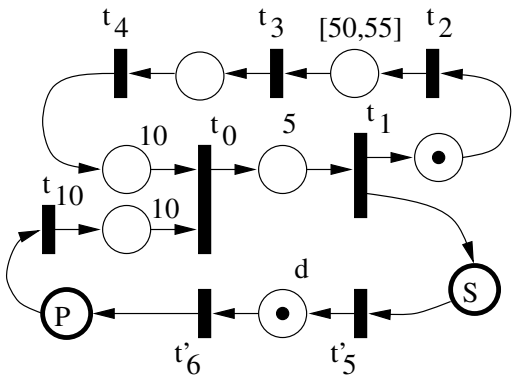

Figure 7. Reduced TEG for the intelligent structural control system

Using the inf- and sup-convolutions to compute lower and upper bounds on the timing behavior in the alternative TEG, we find that

$$((20);(0)) \leq d \leq ((25,35);(0))$$

which yields

$$((0,85);(65)) \leq t'_6 \leq ((0,130);(75)).$$

The lower bound $(((0,85);(65)))$ corresponds to a decision function in place S that always selects t_{11} as its downstream transition. Since the goal is to generate pulses, t_5 should be selected at some point in time. Therefore, inf-convolution is not a desirable decision function in S. Moreover, the upper bound $(((0,130);(75)))$ shows that the constant selection of t_5 as the output of S results in a separation time between calculations as low as 75, which violates our requirements. Clearly, deciding the output of S requires some control.

Computing a decision function for S cannot be done automatically in this case. However, we can observe that the sum of the minimum cycling time in the updating cycle (65) and the minimum cycling time in the pulse calculation cycle (70) is 135. Therefore, a decision function alternating between the updating cycle (65 to 70 t.u. because of the synchronization with the sensor cycle) and the calculation cycle (70 to 75 t.u.) of the controller should result in a separation time of at least 135 t.u. between successive pulse calculations. Let u and c be the delay signals (and U and C their underlying sequences) in the updating cycle and the calculation cycle respectively of the controller. Define delay d such that its underlying sequence D is as follows:

$$\forall i > 0 : D[i] = \begin{cases} U[(i+1)/2] & \text{if } i \text{ is odd} \\ C[i/2] & \text{otherwise} \end{cases}$$

Then,

$$d \in [((20);(35,-35)),((25);(35,-35))]$$

which yields the following bounds on D:

$$\{20,55,20,55,\ldots\} \leq D \leq \{25,60,25,60,\ldots\}.$$

Using the new definition of delay d, we obtain

$$((0,120);(35,100)) \leq t'_6 \leq ((0,130);(40,105))$$

Passes through the calculation loop correspond to the even-indexed events. Therefore, the separation time between two calculations is always greater than or equal to 135.

We now show how to compute the timing occurrences of events in the actuator. Figure 8 illustrates the TEG for the actuator. The synchronization between the controller and the actuator is handled as follows: the actuator is modeled as a stand-alone TEG and its synchronization with the controller via a non-zero initial value for transition x_3. In general, initial values are constant, and therefore, they impact only the first values of the event signals. However, as shown in [4] and Section 5, non-constant initial values serve as synchronization points at all indices in the event signals. An initial value v for an event signal x guarantees that $x \geq v$. Therefore, making the signal corresponding to transition t_8 the initial value for transition x_3 actually enforces the synchronization between the controller and the actuator. Let A be the transition matrix of the actuator subnet. Let the initial vector v be such that $v[0] = v[1] = v[2] = e$ and $v[3] = t_8$. Then, the event signals for x_i $(i = 0,\ldots 3)$ are given by A^*v.

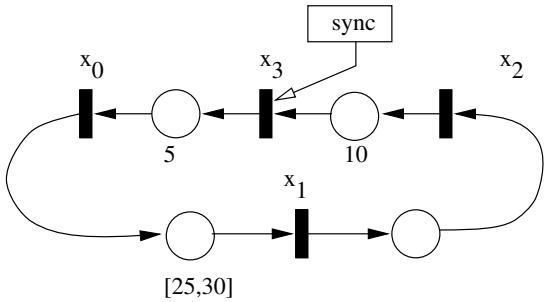

Figure 8. Event graph for the actuator of the intelligent structural control system

The value of transition t_8 can be computed from the value of transition t'_6. Let T_8 be the underlying sequence corresponding to transition t_8. Then,

$$\forall i > 0 : T_8[i] = T'_6[2i] - 5.$$

Using this result, we find that

$$x_1 = [((75);(135)),((80);(145))]$$

We can now check the correctness of the system. Given that X_1 is the underlying sequence of signal x_1, the system is correct if the following inequation is verified:

$$\forall i > 0 : T_0/8 \leq X_1[i+1] - X_1[i] \leq T_0/2.$$

Since $T_0 = 290$, the inequation becomes

$$\forall i > 0 : 37 \leq X_1[i+1] - X_1[i] \leq 145.$$

Observe that the value $X_1[i+1] - X_1[i]$ is the value of the elements (or steps) in the period and transitory sequence of signal x_1. Therefore, the proof of correctness is reduced to verifying that the value of any step in signal x_1 is between 37 and 145, which is the case.

8. Conclusions

In this work, we have illustrated how the (max,+) algebra of signals can be used to verify the timing correctness of real-time discrete event systems. In doing so, we introduced a hierarchical analysis technique that allows the use of the (max,+) algebra on non-deterministic Petri nets. This allowed us to model and verify the correctness of an intelligent structural control system [12]. Future work will focus on extending the analysis capabilities for non-deterministic systems and on defining composition techniques to further reduce the cost of analysis.

References

[1] F. Baccelli, G. Cohen, G. Olsder, and J.-P. Quadrat. *Synchronization and Linearity: an Algebra for Discrete Event Systems*. John Wiley and Sons, 1992.

[2] H. Ben-Abdallah, I. Lee, and J.-Y. Choi. A graphical language with formal semantics for the specifications and analysis of real-time systems. In *Proceedings of the 16th IEEE Real-Time Systems Symposium*, pages 276–286, Raleigh Durham, North Carolina, December 1995.

[3] G. P. Brat and V. K. Garg. A (max,+) algebra for non-stationary periodic timed discrete event systems. In *Proceedings of the International Workshop on Discrete Event Systems*, pages 237–242, Cagliari, Italy, August 1998.

[4] G. P. Brat and V. K. Garg. A max-plus algebra of signals for the supervisory control of real-time discrete event systems. In *Proceedings of the 9th Symposium of the International Federation of Automatic Control on Information Control in Manufacturing*, Nancy-Metz, France, June 1998.

[5] G. P. Brat and V. K. Garg. A max-plus algebra of signals for the supervisory control of real-time discrete event systems. Technical Report TR-PDS-1998-001, Department of Electrical and Computer Engineering, University of Texas at Austin, USA, January 1998.

[6] D. D. Cofer and V. K. Garg. A timed model for the control of discrete event systems involving decisions in the max-plus algebra. In *Proceedings of the 31st IEEE Conference on Decision and Control*, pages 3363–3368, Tuczon, Arizona, 1992.

[7] D. D. Cofer and V. K. Garg. A max-algebra solution to the supervisory control problem for real-time discrete event systems. In *Lecture Notes in Control and Information Sciences 199: 11th International Conference on Analysis and Optimization of Systems*, pages 283–289, 1994.

[8] D. D. Cofer and V. K. Garg. Supervisory control of real-time discrete event systems using lattice theory. In *Proceedings of the 33rd IEEE Conference on Decision and Control*, pages 978–983, December 1994.

[9] G. Cohen, P. Moller, J.-P. Quadrat, and M. Viot. Dating and counting events in discrete event systems. In *Proceedings of the 25th IEEE Conference on Decision and Control*, pages 988–993, 1986.

[10] C. Daws and S. Yovine. Two examples of verification of multirate timed automata with KRONOS. In *Proceedings of the 16th IEEE Real-Time Systems Symposium*, pages 66–75, Pisa, Italy, Dec. 1995.

[11] E.Asarin, O. Maler, and A.Pnueli. Symbolic controller synthesis for discrete and timed systems. In A. N. P. Antsaklis, W. Kohn and S. Sastry, editors, *Hybrid Systems II*, pages 1–20. LNCS 999, Springer-Verlag, 1995.

[12] W. M. Elseaidy, R. Cleaveland, and J. W. B. Junior. Verifying an intelligent structural control system: A case study. In *Proceedings of the 15th IEEE Real-Time Systems Symposium*, pages 271–275, San Juan, Puerto Rico, December 1994.

[13] S. Gaubert. *Théorie Linéaire des Systèmes dans les Dioïdes*. PhD thesis, Ecole des Mines de Paris, Paris, 1992.

[14] R. Gerber, I. Lee, and A. Zwarico. Communicating Shared Resources: A model for distributed real-time systems. In *Proceedings of the 12th IEEE Real-Time Systems Symposium*, pages 68–78, Santa Monica, California, December 1989.

[15] C. Heitmeier, A. Bull, C. Gasarch, and B. Labaw. SCR: A toolset for specifying and analyzing requirements. In *Proceedings of the Tenth Annual Conference on Computer Assurance (COMPASS '95)*, pages 109–122, Gaithersburg, Maryland, June 1995.

[16] C. Heitmeier and N. Lynch. The generalized railroad crossing: A case study in formal verification of real-time systems. In *Proceedings of the 15th IEEE Real-Time Systems Symposium*, pages 120–131, San Juan, Puerto Rico, December 1994.

[17] T. A. Henzinger, P.-H. Ho, and H. Wong-Toi. HYTECH: The next generation. In *Proceedings of the 16th IEEE Real-Time Systems Symposium*, pages 56–65, Raleigh Durham, North Carolina, December 1995.

[18] F. Jahanian and D. Stuart. A method for verifying properties of Modechart specifications. In *Proceedings of the 9th IEEE Real-Time Systems Symposium*, pages 12–21, Los Alamitos, CA, July 1988.

[19] K. G. Larsen, P. Petterson, and Wang Yi. Compositional and symbolic model-checking of real-time systems. In *Proceedings of the 16th IEEE Real-Time Systems Symposium*, pages 76–87, Pisa, Italy, Dec. 1995.

[20] T. Murata. Petri nets: Properties, analysis, and applications. *Proceedings of the IEEE*, 77(1):541–580, 1989.