AN APPROACH TO REAL TIME SYSTEMS MODELING

P. CASPI , N. HALBWACHS

IMAG Laboratory, BP.53
38041 GRENOBLE FRANCE

ABSTRACT

A mathematical model is proposed, allowing a description of real time digital systems, suitable for specification, documentation and validation purposes. This model makes use of an absolute time, as it is perceived by an external observer to the system. The basic concepts of events and variables are defined by means of sequences, so as to permit the whole history of a system to be globally handled. The descriptive power of this model is illustrated by examples. Then we deal with design and validation problems, restricting ourselves to a class of systems without interpretation, i.e. which can be described only by means of events. The algebraic structure of the set of events is studied, and the application of formal calculus techniques is outlined.

INTRODUCTION

In this paper, real time discrete systems are considered in a rather restrictive sense: In ordinary systems, only functional and temporal ordering relationships between input/output variables are relevant for the user, and time appears only as a speed criterion that allows to compare different functionally equivalent systems. In real time systems, time relationships between external variables are also important in order to decide of the adequacy of the system to its requirements. For instance, many logical or numerical automatic control systems, signal and picture processing systems, that were formerly realized using analog or discrete technologies, are now implemented on (multi-) mini or microcomputers, for evident reasons of cost, versatility, computing power and reliability. In most of these systems, fundamental time constraints appear (frequencies, response times...).

In the same way as for ordinary computer systems, the design of these systems leads to problems of specification, description, and validation.

This work was supported by C.N.R.S. under grant ATP-"Parallélisme, communication, synchronisation".

In ordinary systems, these problems have been approached by means of a mathematical formalization. Many authors and the experience over the past 15 years have stressed the advantages of the mathematical language - such as universality, precision, ability to formal derivation - and successful applications to software specification [1], programming languages semantics definition [5,9], and program proving are available. However, those approaches cannot be straightforwardly applied to our real time systems, since, in their philosophy, a great deal of efforts have been spent in order to withdraw the time from the models.

These considerations have led us to look for convenient mathematical descriptions of real time systems. Our first objective being the specification of such systems, we have defined tools adapted to behavioural description of events and variables, allowing to denote every occurrence of an event, and every assignment to a variable along the whole history of the system (sections 1 and 2). These tools are well suited to input-output descriptions of systems, and thus may be expected to avoid these overspecifications that would arise from descriptions in terms of buffers, tasks, processes, clocks...

In section 3, a formal calculus over events is outlined. Events are formalized there as formal series, and algebraic operators are defined on such series, leading to very concise descriptions, and allowing, to some extend, formal transformations and deductions on these descriptions.

1. BASIC CONCEPTS

1.1 Time

Our notion of time refers to an absolute one, such as perceived by an external observer to the system. Clearly, at the description level, the problem of the relative times measured by several subsystems clocks in a distributed system, such as studied in [6], does not arise. We shall generally model the set $\mathbb{T}$ of times by the set $\mathbb{R}$ of real numbers, achieved with minimum and maximum elements $-\infty$ and $+\infty$ (whose usefulness will appear later in the paper). In many cases, $\mathbb{Z}$, the achieved set of relative integers can be considered. Elements of $\mathbb{T}$ are called indifferently times, instants, durations, delays..

1.2 Events

We consider as events the transitions between states that may appear in a system or in its environment, such as setting a switch, or assigning a new value to a variable. Moreover, an event can occur several times during the period of observation of the system. As we deal with discrete systems, the set of occurrences of an event is assumed to be enumerable.

At a suitable level of abstraction, as we look for a mathematically tractable definition, we can decide that an occurrence of an event has no duration, and can be viewed as a cut in the time line that separates the periods before and after the event occurs. As we have adopted an observationnal point of view, it will be convenient not to be restricted by causality limitations, and thus to be able to equally handle the past, the present and the future of the system. For these reasons, we define an event e to be an increasing mapping from $\mathbb{N}$ to $\mathbb{T}$, where e(n) denotes the instant of the n-th occurrence of e.

For some reasons that will appear later in the modeling of variables, we impose furthermore that every event e must be such that $e(0)=-\infty$. Note that if the event e has only a finite number n of occurrences, it will be such that $e(m)=+\infty$ for every $m>n$, for an occurrence whose time is $+\infty$ will never occur.

This modeling in terms of sequences has already been applied, for instance in the applicative language LUCID [2].

Example: Specification of periodic events: In many automatic control and signal processing systems, periodic inputs or outputs are required. A periodic event e, with period Δ, can easily be specified in our model by stating that:

$$\forall n \in \mathbb{N}^*,\ e(n+1) = e(n)+\Delta$$

However, a strictly periodic output can be rather difficult to realize on a digital computer. One can wonder whether this is not an overspecification due to the use of informal language. It may occur that weaker requirements should be equally convenient, such as the following ones:

. Maximum Period: The time interval between two successive occurrences of e must be smaller than Δ:

$$\forall n \in \mathbb{N}^*,\ e(n+1) < e(n)+\Delta$$

. Mean Period: e must happen once and only once in each time interval of duration Δ, counted from an initial instant t_o:

$$\forall n \in \mathbb{N}^*,\ (n-1)\Delta+t_o < e(n) < n\Delta+t_o$$

1.3 Variables

Since we deal with discrete systems, variables may be considered as cells whose values change at discrete times. As for events, it will be convenient to encapsulate the whole history of a variable. So we define a variable X to be a couple $(x,\hat{x})$, where:

. x is a sequence of values, i.e. an application from $\mathbb{N}$ to D_x, the domain of the variable X;

.$\hat{x}$ is an event, called assignment to X.

The interpretation is: At the instant $\hat{x}(n)$, X takes the value x(n). x(0) is the initial value of X, taken at the instant $\hat{x}(0)=-\infty$. Every value may be the undefined value ω, which belongs to every domain D_x.

Example: Specification of response times: When an input variable X and an output variable Y are linked by a functional relationship (Y=g(X)), a common requirement in real time systems consists of defining a maximum response time, say δ, between the input of a value of X and the corresponding output of Y. However, such a requirement can be found in specification documents in the case where X and Y are periodic (in one sense defined above), with different periods. Such a requirement may be given, at least, three interpretations:

. Y must be computed at the same frequency as X, and there is a one to one correspondence between their values:

$$\forall n \in \mathbb{N}^*,\ y(n) = g(x(n)) \text{ and } \hat{x}(n) < \hat{y}(n) < \hat{x}(n)+\delta$$

. Y may be computed at a smaller frequency than X, but each output of Y corresponds to an input of X not older than δ:

$$\forall n \in \mathbb{N}^*,\ \exists m \in \mathbb{N}^* \text{ such that}$$
$$y(n) = g(x(m)) \text{ and } \hat{x}(m) < \hat{y}(n) < \hat{x}(m)+\delta$$

. Y must be computed at a greater frequency than X, and each input of X causes the corresponding output of Y in a delay shorter than δ (this situation appears especially when g involves other variables than X):

$$\forall n \in \mathbb{N}^*,\ \exists m \in \mathbb{N}^* \text{ such that}$$
$$y(m) = g(x(n)) \text{ and } \hat{x}(n) < \hat{y}(m) < \hat{x}(n)+\delta$$

2 DESCRIPTION TOOLS

With the basic notions of events and variables, as defined above, one can describe, using mathematical language, the behaviour of any discrete real time system, or specify the set of correct behaviours of such a system to be implemented. Nevertheless, in order to provide an effective description language, we define now some tools which seem to be of general usage for the sake of natural description. Of course, we do not aim here at dogmatically identifying some closed set of general purpose primitives, all the more as it is the advantage of the mathematical language to allow its user to freely define the particular extensions that seem well adapted to a specific problem.

2.1 Counters

An alternative way for handling events consists of using counters instead of times. Such counters have appeared useful in describing and programming synchronization between processes [7,8]. We shall associate, with each event e, a counter μ_e , which is an application from $\mathbb{T}$ to $\mathbb{N}$, defined as follows:

$$\forall t \in \mathbb{T},\ \mu_e(t) = \text{Card}\ \{ n \in \mathbb{N}^* | e(n) < t \}$$

Thus, $\mu_e(t)$ measures the number of occurrences of e that have happened strictly before t. μ_e is an increasing, left continuous integer function on $\mathbb{T} - \{-\infty\}$. Such a function will be called a left counter. Similarly a right counter (right continuous) $\mu_e{}^+$ may be defined as:

$$\forall t \in \mathbb{T},\ \mu_e^+(t) = \text{Card}\{n \in \mathbb{N}^* \mid e(n) < t \}$$

Obviously, there is a bijection between the set of events and the set of left or right counters since:

$$\forall n \in \mathbb{N},\ e(n) = \inf \{ t \in \mathbb{T} | \mu_e(t) > n \}$$

$$= \min \{ t \in \mathbb{T} \mid \mu_e^+(t) > n \}$$

According to these definitions, the event e occurs at the instant t if and only if $\mu_e(t) < \mu_e^+(t)$.

Some compositions of the already defined functions provide new interesting ones:

2.2 Last occurrence functions

$\Theta_e = e \circ \mu_e$ and $\Theta_e^+ = e \circ \mu_e^+$ are increasing functions from $\mathbb{T}$ to $\mathbb{T}$. $\Theta_e(t)$ (respectively $\Theta_e^+(t)$) provides the instant of the last occurrence of e that precedes strictly (resp. loosely) t. These functions can be useful for description purposes. However, they cannot be used in order to define events, since several events may correspond to the same Θ function, in the case of events with multiple simultaneous occurrences. If we restrict ourselves to events without simultaneous occurrences, the set $\{I, e, \mu_e^+, \Theta_e^+\}$, where I denotes the identity function on $\mathbb{N}$, is closed with respect to functional composition, as shown in figure 1.

2.3 Current value functions

Concerning variables, it may be interesting in some applications, to handle the current value of a variable X as a function of time. This is achieved by means of the functions $\tilde{x} = x \circ \mu_{\hat{x}}$ and $\tilde{x}^+ = x \circ \mu_{\hat{x}}^+$. Both functions completely define the variable X, when $\hat{x}$ has no simultaneous occurrences. This condition is clearly satisfied in sequential systems, but can be false in parallel ones, where several processes can try to assign the same variable at the same time.

	I	e	μ_e^+	Θ_e^+
I	I		μ_e^+	
e	e		Θ_e^+	
μ_e^+		I		μ_e^+
Θ_e^+		e		Θ_e^+
x	x		$\tilde{x}^+$	
$\tilde{x}^+$		x		$\tilde{x}^+$

Figure 1

2.4 Filtering of an event by a condition

The following operation is of very common usage: We define a condition C to be a variable with truth values, i.e. such that $D_c = \{\text{true,false}\}$. Then the filtering of an event e by a condition C provides an event, noted e/C (e/C^+), which occurs each time e occurs when C (C^+) is true:

$\forall t \in \mathbb{T}$,

$$\mu_{e/C}(t) = \text{Card}\ \{n \in \mathbb{N}^* \mid e(n) < t \text{ and } \tilde{c}(e(n)) = \text{true}\}$$

$$\mu_{e/C^+}(t) = \text{Card}\ \{n \in \mathbb{N}^* \mid e(n) < t \text{ and } \tilde{c}^+(e(n)) = \text{true}\}$$

2.5 Analysis example

This example illustates the application of the proposed tools in handling what happens in an asynchronous distributed system. The example is derived from a fault tolerant multiple computer, proposed by the french SFENA Company, for aircraft control purposes [3].

The six computers are organised as a ring structure such that each computer can broadcast its results to its three successors in the ring. A continuous variable V is sampled by three redundant sensors K_1, K_2, K_3, connected to three computers C_1, C_2, C_3 (fig.2).

Periodically, each sensor K_i samples the variable V and sends it to its associated computer. Let X_i (i=1..3) be the corresponding variables.

Computers connected to a sensor emit the sample received from the sensor, and the other computers emit the mean value of the samples received from other computers. Let Y_i (i=1..6) be the result broadcasted by computer C_i.

The system is totally asynchronous, since each computer and sensor has its own clock. All the periods are theoretically equal, but, since clocks are different, they can in fact differ slightly. However, it is possible to know an upper bound Δ of the period. In the absence of failure, the involved variables may be described as follows:

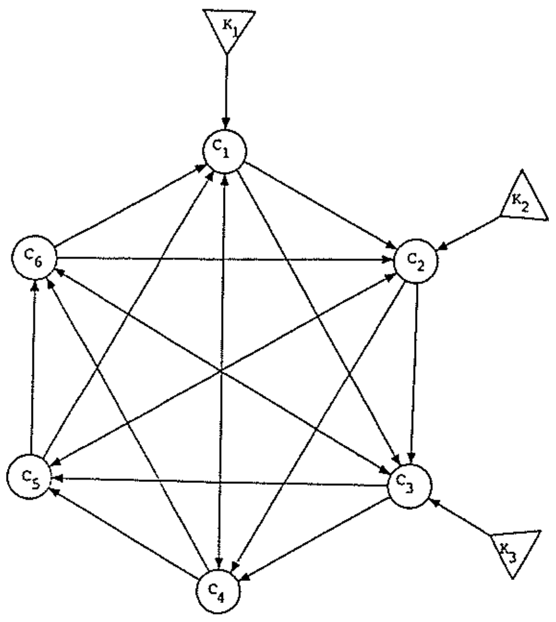

Figure 2

$\forall t \in \mathbb{T}$, $\forall i=1..3$, $\forall j=4..6$

$\tilde{x}_i(t) = v(\Theta_{\hat{x}_i}(t))$ and $\tilde{y}_i(t) = \tilde{x}_i(\Theta_{\hat{y}_i}(t))$

$$\tilde{y}_j(t)= \frac{1}{3}[\ \tilde{y}_{j-1}(\Theta_{\hat{y}_j}(t))+\tilde{y}_{j-2}(\Theta_{\hat{y}_j}(t))+\tilde{y}_{j-3}(t)(\Theta_{\hat{y}_j}(t))]$$

It is then possible to express $\tilde{y}_6(t)$ (using functional composition):

$$\begin{aligned}\tilde{y}_6 = [\ & \tfrac{1}{3}(\tilde{v}\circ\Theta_{\hat{x}_3}\circ\Theta_{\hat{y}_3}) + \\ & \tfrac{1}{9}(\tilde{v}\circ\Theta_{\hat{x}_1}\circ\Theta_{\hat{y}_1} + \tilde{v}\circ\Theta_{\hat{x}_2}\circ\Theta_{\hat{y}_2} + \tilde{v}\circ\Theta_{\hat{x}_3}\circ\Theta_{\hat{y}_3})\circ\Theta_{\hat{y}_4} + \\ & \tfrac{1}{9}(\tilde{v}\circ\Theta_{\hat{x}_2}\circ\Theta_{\hat{y}_2} + \tilde{v}\circ\Theta_{\hat{x}_3}\circ\Theta_{\hat{y}_3})\circ\Theta_{\hat{y}_5} + \tfrac{1}{27}(\tilde{v}\circ\Theta_{\hat{x}_1}\circ\Theta_{\hat{y}_1} + \\ & \tilde{v}\circ\Theta_{\hat{x}_2}\circ\Theta_{\hat{y}_2} + \tilde{v}\circ\Theta_{\hat{x}_3}\circ\Theta_{\hat{y}_3})\circ\Theta_{\hat{y}_4}\circ\Theta_{\hat{y}_5}\ ]\circ\Theta_{\hat{y}_6}\end{aligned}$$

In this expression, terms like $[v\circ\Theta_{\hat{x}_2}\circ\Theta_{\hat{y}_2}\circ\Theta_{\hat{y}_4}\circ\Theta_{\hat{y}_5}\circ\Theta_{\hat{y}_6}](t)$ stand for "the value of V at the instant of the last occurrence of $\hat{x}_2$, preceding the instant of the last occurrence of $\hat{y}_2$, preceding ..., preceding the instant of the last occurrence of $\hat{y}_6$ preceding t.

Using this expression, we can compute an upper bound of the error $\tilde{v}(t) - \tilde{y}_6(t)$, that can be useful for setting a voting threshold. This can be done by replacing $\tilde{v}(t)$ by $\alpha.t$, where α is an upper bound of the derivative of $\tilde{v}$, and by remarking that, from the assumption made upon the maximum period of all events, all the θ functions appearing here satisfy $t-\Delta < \Theta(t) < t$. This yields:

$$\tilde{v}(t) - \tilde{y}_6(t) < \frac{34}{9}\ \alpha\ .\Delta$$

This result might certainly be obtained by other methods. However, it is obtained here in a very straightforward manner and this seems due to the fact that our formalism allows a very exact representation of what happens in that distributed asynchronous system.

3 FORMAL CALCULUS OVER EVENTS

In the remainder of the paper, we shall deal with design problems for real time systems described in our model, such that proofs of properties, static behavioural analysis, or description transformations. Because of space limitations, we can only outline our algebraic model, more thoroughly developped in [4]. We shall henceforth restrict ourselves to logical systems, i.e. without variables. A behaviour of such a system is then a set of inter-related events.

3.1 Pseudo Events

The process of describing sequences by means of formal series is classical, for instance in the field of discrete transform techniques applied to the solution of finite difference equations.

Let us define a pseudo event to be a formal series:

$$x = \Sigma_{i=1}^{\#x}\bar{x}_n D^{x_n}$$

where

. $(\bar{x}_n)$ is a sequence of non null relative integers;
. (x_n) is a strictly increasing sequence of instants;
. Both sequences have the same length $\#x$, which can be finite or infinite. If $\#x$ is infinite, it is assumed that the sequence of instants x_n converges towards infinity. $\#x=0$ defines the pseudo event 0.

With each pseudo event x can be associated in a one to one way its counter μ_x defined by:

$$\forall t \in \mathbb{T},\ \text{if } x=0 \text{ then } \mu_x(t) = 0 \quad \text{else } \mu_x(t) = \Sigma_{x_n<t}\ \bar{x}_n$$

Now, let us consider the set E of events satisfying the following property:

$e \in E \iff$ e may only have a finite number of occurrences in a finite amount of time.

Then, with each e in E, we can associate a pseudo event x(e), such that:
. $(x(e)_n)$ is the sequence of the finite instants of occurrences of e;
. $\bar{x}(e)_n$ is the number of occurrences of e happening at the instant $x(e)_n$. So $\bar{x}(e)_n \in \mathbb{N}^*$.

Example:
$e= (-\infty,0,3,3,4,\infty,..) \quad = \quad x(e)= D^0+ 2D^3+ D^4$

The correspondence being one to one, we shall henceforth confuse e and x(e), and consider E to be the subset of pseudo events x such that $(\bar{x}_n) \subset \mathbb{N}^*$

(or equivalently, such that μ_x is increasing).

The set R of pseudo events is provided with the usual sum and product operations over formal series. Intuitively, the sum of two events of E is their disjunction ($\mu_{e+f} = \mu_e + \mu_f$) and the product by D^Δ delays an event by Δ units of time. So D^o is the unity of the product. It will be noted 1 and omitted in products. The general product will be useful in formal derivations.

A classical result about formal series is that a pseudo event x has an inverse 1/x (i e. such that $x.(1/x)=1$) if and only if $\bar{x}_1 = \pm 1$.

Finally, let us define the following ordering relationship over pseudo events:

$$\forall x,y \in R, \quad x \leq y \iff \forall t,\ \mu_x(t) < \mu_y(t)$$

If x and y are events, $x<y$ means that, for every integer n, the n-th occurrence of x happens always after the n-th occurrence of y.

<u>Example: Periodic events</u>: Let us come back to the example given in 1.2. We get a much more concise specification of each case:

- Strict period: $e = D^\Delta e + D^{t_o}$ (1)

  From (1), we can deduce $e(1-D^\Delta) = D^{t_o}$, and further, if $\Delta > 0$:

$$e = \frac{D^{t_o}}{1 - D^\Delta}$$

- Maximum period: $e > D^\Delta e + D^{t_o}$ (2)

  Note that (2) implies $e > D^{t_o}/(1-D^\Delta)$ but the converse is false, because $1-D^\Delta$ is not an event and it can be shown [4] that the product by a pseudo event x is order preserving if and only if x is an event.

- Mean period: $D^\Delta f < e < f$, with $f = D^\Delta f + D^{t_o}$ (3)

  (3) implies $f = D^{t_o}/(1-D^\Delta)$ and so:

$$\frac{D^{t_o+\Delta}}{1-D^\Delta} < e < \frac{D^{t_o}}{1-D^\Delta}$$

Now, let us apply the formal model outlined above to a more significant example.

3.2 Example

A system receives two strictly periodic sequences of input requests. The former sequence starts from the instant 0, with a 2 seconds period, the later starts from the instant 1, with a 4 seconds period. The system is made of n identical processors. The processing of a request belonging to the former sequence lasts for 7 seconds, while a request from the later sequence needs a 5 seconds processing.

This system may be formalized in our model as follows: Let $\hat{a}$, $\hat{b}$ be the events respectively associated with input arrivals from the former and the later sequences. Let $\hat{c}$, $\hat{d}$ respectively represent the event "an input from the former (resp. later) sequence is taken into account by some processor". Finally, let $\hat{e}$, $\hat{f}$ respectively represent the event "a processor ends the processing of an input from the former (later) sequence". Then:

- The specification of input sequences may be written as:

$$\hat{a} = D^2\hat{a} + 1 \text{ and } \hat{b} = D^4\hat{b} + D \qquad (1)$$

- As a request cannot be taken into account before its arrival, we get:

$$\hat{c} < \hat{a} \text{ and } \hat{d} < \hat{b} \qquad (2)$$

- The processing times of requests is specified as follows:

$$\hat{e} = D^7\hat{c} \text{ and } \hat{f} = D^5\hat{d} \qquad (3)$$

- As a request may only be taken into account when there exists an idle processor, and since $\hat{c}+\hat{d}$ represents the event "a processor begins working" and $\hat{e}+\hat{f}+n$ represents the event "a processor becomes idle", we have:

$$\hat{c} + \hat{d} < \hat{e} + \hat{f} + n \qquad (4)$$

Now consider the problem of determining the minimum number of processors, n, needed so as to be able to take into account every incoming request as soon as it arrives (i.e. each input has to occur at a time when there is an idle processor). This immediate handling requirement provides:

$$\hat{c} = \hat{a} \text{ and } \hat{d} = \hat{b} \qquad (5)$$

Now the equations (1) provide:

$$\hat{a} = \frac{1}{1-D^2} \text{ and } \hat{b} = \frac{D}{1-D^4}$$

Then, all the event variables may be eliminated, and the problem may be restated as follows:

"Find a mimimum integer n so that:

$$\frac{1 - D^7}{1 - D^2} + \frac{D - D^6}{1 - D^4} < n \text{"}$$

Then, by equalizing the denominators of the two fractions:

$$\frac{1 + D + D^2 - D^6 - D^7 - D^9}{1 - D^4} < n$$

The left hand side of this last inequality is a pseudo event, x, which becomes periodic (with period 4) after an initial delay. We want to find the maximum value of its counter function. Performing the polynomial division, according to increasing powers of D, of its numerator by its denominator, we get successively:

$$x = 1 + \frac{D + D^2 + D^4 - D^6 - D^7 - D^9}{1 - D^4}$$

$$= 1 + D + \frac{D^2 + D^4 + D^5 - D^6 - D^7 - D^9}{1 - D^4}$$

$$= 1 + D + D^2 + \frac{D^4 + D^5 - D^7 - D^9}{1 - D^4}$$

$$= 1 + D + D^2 + D^4 + \frac{D^5 - D^7 + D^8 - D^9}{1 - D^4}$$

$$= 1 + D + D^2 + D^4 + D^5 + \frac{-D^7 + D^8}{1 - D^4}$$

The remainder $-D^7+D^8$ is $D^7(-1+D)$ and the degree of $(-1+D)$ is smaller than 4. So, $(-D^7+D^8)/(1-D^4)$ is a periodic pseudo event, the counter of which can easily be shown to have the maximum value zero. Thus the maximum value of the counter of x is the one of $1+D+D^2+D^4+D^5$, which is 5 (cf. figure 3). So n=5 is the solution.

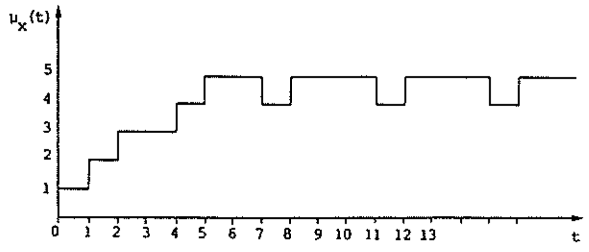

Figure 3

CONCLUSION

We have studied the problem of specifying and reasoning about behaviours of real time digital systems. After having given an original definition of the basic notions of events and variables, we have described a set of tools suitable for giving precise behavioural specifications of such systems.

Some rationales are given concerning these tools, that bring a deeper insight into the description problems, such as the advantage of being able to name every occurrence of an event, or every value of a variable, during the whole history of a system, and to define the kind of continuity satisfyed by the functions of time involved in a description.

Then, we have defined a mathematical structure, which seems to be adequate for handling systems of events. There is a great deal of open problems, in order to increase the power of the outlined calculus. In particular, optimization problems need some kind of linear programming techniques, which are far from being easy to adapt to partially ordered sets.

Nevertheless, we believe that our formalization is a suitable one for handling real time problems, and that the difficult questions met here are intrinsic of any investigation in this field. Even though the class of problems we are able to solve using this model is, until now, quite limited, it seems to be a promising way in the real time systems analysis field, which is far to be thoroughly investigated, in spite of its increasing importance.

REFERENCES

1. ABRIAL J.R., "Z: A specification language". Proc. International Conference on Mathematical Studies of Information Processing, Kioto (Japan), August 1978.

2. ASHCROFT E.A., WADGE W.W., "Lucid: A non procedural language with iteration". CACM, Vol.20, n°7, July 1977.

3. BILLOIR T., Communication at the IEEE Workshop on the Validation of Fault Tolerant Computers and Systems, Luray (Virginia), September 1980.

4. CASPI P., HALBWACHS N., "Algebra of events: A model for parallel and real time systems", Proc. International Conference on Parallel Processing, Bellaire (Michigan), August 1982.

5. HOARE C.A.R., "Consistent and complementary formal theories of the semantics of programming languages". Acta Informatica, 3, 1974.

6. LAMPORT L., "Time, clocks, and the ordering of events in a distributed system". CACM, Vol.21, n°7, July 1978.

7. REED D.P., KANODIA R.K., "Synchronization with eventcounts and sequencers". CACM, Vol.22, n°2, February 1979.

8. ROBERT P., VERJUS J.P., "Towards autonomous descriptions of synchronization modules". Proc. IFIP Congress, Toronto, 1977.

9. TENNENT R.D., "The denotational semantics of programming languages". CACM, Vol.19, n°8, August 1976.