ALGEBRA OF EVENTS : A MODEL FOR PARALLEL AND REAL TIME SYSTEMS

P. CASPI, N. HALBWACHS
IMAG Laboratory
Grenoble, FRANCE

Abstract: The model presented here differs from the usual models of parallel processing by two aspects: On one hand, it takes fully into account the metric notion of time, thus allowing the description of hard real time systems. On the other hand, it is a pure behavioural model, in the sense that it does not use any abstract machine notion. From a formalization of the notion of event, we show that the behaviour of a logical system may be described, by means of few operators, in a precise and concise way. The algebraic properties of the model are then studied, in order to define some methods for analysing or transforming systems described in this formalism.

INTRODUCTION

Two different notions of time are used in system modeling. In sequential systems, as far as time performance is not considered, the time concept may be reduced to the ordering of actions, or more generally of events occurring during the system life, that is a perfectly known total ordering relationship. In parallel systems, the ordering of events depends on the execution time of the actions. So a precise description of such a system needs the usual metric notion of time. However, since the execution times are generally uncknown, the correctness of parallel systems is commonly required to hold independently of any assumption about the speeds of the involved processors. So, many authors were led to consider the ordering of events in a parallel system as a partial ordering, and to assimilate parallel systems with undeterministic sequential ones. This approach allows to get rid of any metric notion of time, and has led to most of the parallel programs proof techniques. However it does not apply as soon as real time systems are considered. In such systems, the metric notion of time is used not only to compare the performances of several implementations, but also to decide of the adequacy of a system to its specifications.

Another characteristic of many approaches to parallel behaviour modeling (for instance [1],[7]) is the use of an abstract machine model, more or less derived from finite state automata. A behaviour is defined as an equivalence class upon the set of machines, and thus the proof of a system reduces to the proof of the equivalence between the abstract machines representing the specification and the implementation of the system. The drawbacks of such an operational approach for the initial specification process have been pointed out in [3]. In short, the specification language is generally far from being natural, and may lead to overspecification.

In this paper, we present a purely behavioural model for logical, parallel or real time systems, which takes fully into account the real time dependencies between internal and external events of a system. Our notion of time may be viewed as a simple ordering time, as far as purely parallel systems are considered, or as a metric time, assumed to be the global time of an external observer to the system.

In section 1, the basic notions of time and event are defined. An event is represented by an increasing staircase function from time to non negative integers, which counts the number of occurrences of the event during the time. An ordering relationship and a set of operators are provided in section 2, that structure the set of events as an ordered semiring. To illustrate the descriptive power of this algebra, we show (section 3) that finite state machines and Petri net models may be specified by systems of linear equations and inequalities over events. In order to define an effective calculus on such specifications, the algebra is extended in section 4 to become a ring, the elements of which are called pseudoevents. The use of this calculus to real time systems design problems is illustrated in section 5. Section 6 describes a systematic method to get approximate results about descriptions in our model, by means of discrete transforms of pseudoevents. Some nice properties of the algebra, when the time may be considered as discrete, are given in section 7. In conclusion, the extension of the model towards numerical systems is discussed, and open problems are set, the solution of which would greatly increase the capabilities of our calculus. Most proofs have been omitted, but may be found in an extended version of this paper [5].

1. TIME AND EVENTS

1.1 Time

Our notion of time refers to an absolute one, such as perceived by an external observer to the system. 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 $\mathbb{R}$ or $\mathbb{Z}$ of real or integer numbers. Elements of $\mathbb{T}$ are called times or instants when $\mathbb{T}$ is considered as an affine

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

space, and time intervals, delays or durations when the vectorial structure of $\mathbb{T}$ is considered.

1.2 Events

We consider as events the transitions between states that may appear either in a system or in its environment, such as setting a switch, or assigning a new value to a variable. Moreover, an event may occur several times during the period of observation of the system, but, as we deal with discrete systems, the set of occurrences of an event is assumed to be enumerable. At a suitable level of abstraction, 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 times before and after the event occurs. Thus we define an event e to be a finite or infinite increasing sequence of instants, where e(n) denotes the instant of the n-th occurrence of e. We shall furthermore impose that, if the sequence is infinite, it converges towards $+\infty$ in $\mathbb{T}$ with n. This restriction is motivated by algebraic reasons and may be intuitively justified because, in discrete systems, an event may not occur infinitely often in a finite amount of time.

The number of occurrences of an event e will be noted #e. For convenience, we do not prevent an event from having several simultaneous occurrences. The set of events will be noted $E(\mathbb{T})$, or simply E when the choice of $\mathbb{T}$ is irrelevant.

Of course, this definition of events copes with real time behaviour modeling. However, it is also convenient to describe sequential or purely parallel systems: For instance, if L is a language on a vocabulary V, we can associate with each symbol a in V and with each string c in L, the event $\hat{a}$ which is the increasing sequence of the ranks of the symbol a in c.

This representation by means of sequences allows us to equally handle the present, the past and the future of the system. This is close to the point of view adopted, for instance, in the applicative language LUCID [2].

1.3 Counters

An alternative way for handling events consists of using counters. Such counters have appeared useful in describing or programming synchronization between processes [10],[11]. With each event e, we shall associate a counter μ_e, which is an application from $\mathbb{T}$ to $\mathbb{N}$, defined as follows:

$$\forall t \in \mathbb{T}, \quad \mu_e(t) = \max\{ n \mid 1 \leq n \leq \#e \ \& \ e(n) < t \} ,$$

Thus $\mu_e(t)$ measures the number of occurrences of e that have happened strictly before t. μ_e is an increasing, left continuous staircase function on $\mathbb{T}$. Figure 1 pictures the counter of the event e=(1,3,4,6).

Let an event counter be an increasing, left continuous total function from $\mathbb{T}$ to $\mathbb{N}$, which

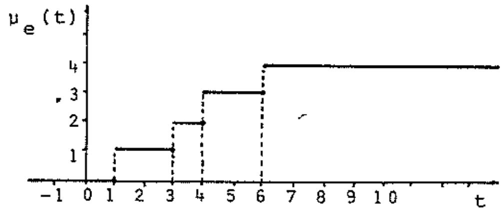

Figure 1

value is zero on some interval $]-\infty, x_0]$, then (using the Church's lambda notation), $\mu = \lambda e.\mu_e$ is obviously a bijection between the set of events and the set of event counters, since:

$$\forall n=1..\#e, \ e(n) = \max\{ t \in \mathbb{T} \mid \mu_e(t) < n\}$$

2. THE ALGEBRA OF EVENTS

A logical system behaviour will be considered as a vector of interrelated events, and a system as a set of such behaviours. In this section, we shall see how to specify such a system by means of few operators over events.

2.1 Primary Events

If $k \in \mathbb{N}^*$, the primary event $\underline{k}$ is, by definition, the event which has exactly k occurrences, simultaneously happening at the instant zero:

$$\forall n=1..k, \ \underline{k}(n)=0$$

$$\mu_{\underline{k}}(t) = \text{if } t\leq 0 \text{ then } 0 \text{ else } k$$

Since the instant zero will generally represent the initial instant in a system life, primary events will be often used to model initial states.

2.2 Ordering over E

For every e,f in E, let

$$e \leq f \iff \#e \leq \#f \ \& \ \forall n=1..\#e, \ e(n) \geq f(n)$$
$$\iff \forall t \in \mathbb{T}, \ \mu_e(t) \leq \mu_f(t)$$

So the (partial) ordering over events coincides with the pointwise ordering over counters. This ordering will be useful, in particular, to represent causality relationships over events.

(E,≤) is a lattice, and we can define the inf and sup operators as follows:

$\forall e,f \ E,$

$\mu_{\inf(e,f)} = \lambda t. \min(\mu_e(t), \mu_f(t))$

$\mu_{\sup(e,f)} = \lambda t. \max(\mu_e(t), \mu_f(t))$

E has a minimum element 0, which is the event which has no occurrences ($\#\underline{0}=0$).

2.3 Sum and Difference of events

The sum of two events e and f is defined to be the event which occurs each time e or f occurs. More precisely, the sequence of occurrences of the event e+f is built by interleaving the sequences of e and f, according to their temporal ordering. This notion can be easily formalized by means of counters, justifying the additive notation:

$$\mu_{e+f} = \lambda t.\ \mu_e(t) + \mu_f(t)$$

The + operation, being obviously commutative and associative, may be generalized to an arbitrary finite number of operands:

$$f = \Sigma_{i=0}^{k} e_i \iff \mu_f = \lambda t. \Sigma_{i=0}^{k} \mu_{e_i}(t)$$

The product of an event e by a natural integer k is the k times iterated sum of e:

$$ke = \Sigma_{i=1}^{k} e$$

The difference over events is only a partial operation, the definition of which results from the definition of the sum:

$$d = e-f \iff e = d+f$$

Note that the difference e-f is defined only if f is a subsequence of e.

2.4 Delay Operators

Let Δ be a delay, then the delay operator D^Δ performs a translation of every occurrence of its operand according to Δ:

$$\mu_{D^\Delta e} = \lambda t. \mu_e(t-\Delta)$$

The exponential notation is justified by the obvious properties that D^0 is the identity operator on E, and that $D^\Delta D^\delta = D^{\Delta+\delta}$ for every delay Δ, δ. The operator D^1 will be noted D.

3. APPLICATION TO BEHAVIOURAL DESCRIPTION

Let us show here that the preceding concepts are well suited to the description of parallel and real time systems, and lead to very concise descriptions of such systems.

3.1 Periodic Events

Let us express that an event e occurs at times 0, Δ, $2\Delta, \ldots, n\Delta, \ldots$. Clearly, e satisfies the following recursive definition:

$$e = D^\Delta e + \underline{1}$$

Similarly, the weaker assumption that e occurs at positive instants, and that two successive occurrences of e are separated by a delay smaller than Δ may be expressed as follows:

$$e < D^\Delta e + \underline{1}$$

3.2 Response Times

Let e be an input event to a system, and s be the output response to e, that is requested to occur within the time interval Δ following each occurrence of e. This can be expressed by:

$$D^\Delta e < s < e$$

These examples point out the usefulness of linear equations and inequalities over events. Evidence for such a fact will also be provided by the following application of our model to the behavioural description of finite state machines, Petri nets, and timed Petri nets.

3.3 Finite State Machine

Let $M = (V, Q, \sigma, q_0)$ a finite state machine, where:

- V is a finite vocabulary
- Q is a finite set of states
- σ is a mapping from Q×Q to V
- $q_0 \in Q$ is the initial state

A behaviour of M is a string $c = a_1 a_2 \ldots a_n \ldots$ of V^* such that there exists a sequence $q_0 q_1 \ldots q_n \ldots$ of states, such that, for every n smaller than the length of c, $\sigma(q_n, q_{n+1})$ exists and is equal to a_{n+1}. In our model, a behaviour of M will be a vector $(\hat{a} \mid a \in V)$ of events, such that $\hat{a}$ is the sequence of the ranks of the symbol a in a string like c.

First, we may describe, for every couple (q,q') of Q×Q, the event "the transition q->q' is performed". Let $e_{qq'}$ be this event. For notational convenience, let $q^\cdot$ (resp. $^\cdot q$) be the set of states q' such that $\sigma(q,q')$ (resp. $\sigma(q',q)$) is defined.

Then, by observing that a state q is left at "instant" n if and only if it was reached at "instant" n-1 and it has some successor state, we get:

- For every state q, such that $q^\cdot \neq \emptyset$,
  $$\Sigma_{q' \in q^\cdot} e_{qq'} = D\Sigma_{q'' \in {}^\cdot q} e_{q''q} + u(q)$$
  where $u(q)$= if $q=q_0$ then $\underline{1}$ else $\underline{0}$

Now, for every a in V, the event $\hat{a}$ happens each "time" a transition q->q' is performed, where $\sigma(q,q')=a$. So:

- For every a in V, $\hat{a} = \Sigma_{\sigma(q,q')=a}\, e_{qq'}$

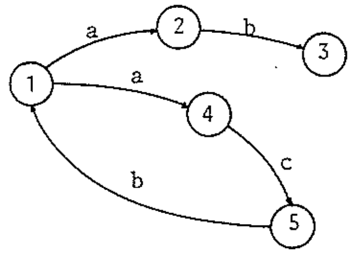

Figure 2

Example: Let us consider the state graph of figure 2. We get:

$$e_{12} + e_{14} = De_{51} + \underline{1} \quad , \quad e_{23} = De_{12}$$

$$e_{45} = De_{14} \quad , \quad e_{51} = De_{45}$$

and:

$$\hat{a} = e_{12} + e_{14} \quad , \quad \hat{b} = e_{23} + e_{51} \quad , \quad \hat{c} = e_{45}$$

from which it follows that:

$$\hat{a} = D^2\hat{c} + \underline{1} \quad \text{and} \quad \hat{b} = D^3\hat{c} + D\hat{c} - \hat{c} + \underline{D}$$

We shall see in section 7 a necessary and sufficient condition for the difference in the last equation to be defined. With this additional condition, the above equations exactly characterize the machine behaviours. Of course, the characterization by means of regular expressions is much simpler, but the same process applies to more complex machines, like communicating systems of [7],[8].

Now, let us see how the model applies to a parallel asynchronous language.

3.4. Petri Nets

Like state machines, Petri nets [9] only use an ordering notion of time. So we shall choose $\mathbb{T} = \mathbb{Z}$ and describe, for each transition of the net, the event "the transition is fired".

Notations: Let P be the set of places, T be the set of transitions. For each place p and each transition a, let us denote:

. $p^\cdot$ (resp. $^\cdot p$), the set of output (resp. input) transitions of p.

. $a^\cdot$ (resp. $^\cdot a$), the set of output (resp. input) places of a.

Let m(p,0) be the initial marking of p, and $\hat{a}$ be the event which happens each time the transition a is fired.

The transitions are fired one at a time, so the marking m(p,n) of the place p at the instant n is:

$$m(p,n) = m(p,0) + \Sigma_{b\in {}^\cdot p}\, \mu_{\hat{b}}(n-1) - \Sigma_{a\in p^\cdot}\, \mu_{\hat{a}}(n)$$

Writting that this marking may not become negative, we get:

$$\forall p \in P, \quad \Sigma_{a\in p^\cdot}\, \hat{a} \leq \Sigma_{b\in {}^\cdot p} D\hat{b} + \underline{m(p,0)} \qquad (1)$$

Now, we can write that at most one transition may be fired at each instant:

$$\Sigma_{a\in T}\, \hat{a} \leq \Sigma_{a\in T} D\hat{a} + \underline{1} \qquad (2)$$

(1) and (2) constitute a system of linear inequalities which characterize the set of correct behaviours of the net.

3.5 Timed Petri Nets

Of course, the preceding characterization of Petri nets may be extended to synchronous real time models such as timed Petri nets [13]. In such nets, a delay $\Delta(p)$ is associated with each place p. The two following rules differenciate timed Petri nets from ordinary ones:

. If a token reaches a place p at the instant t, it becomes unavailable until the instant $t+\Delta(p)$. A transition is enabled if and only if each of its input places contains an available token.

. A transition may not remain enabled during a non null interval of time: It must be either fired or disabled as soon as it is enabled.

The inequality (2) of ordinary Petri nets does not hold for timed nets, since several transitions may be simultaneously fired. Taking the first rule into account, the system (1) becomes:

$$\forall p \in P, \quad \Sigma_{a\in p^\cdot}\, \hat{a} \leq D^{\Delta(p)}\Sigma_{b\in {}^\cdot p}\, \hat{b} + \underline{m(p,0)}$$

The second rule forces every event to be as large as possible, so the above system must become:

$\forall a \in T,$

$$\hat{a} = \inf_{p\in {}^\cdot a}\left(D^{\Delta(p)}\Sigma_{b\in {}^\cdot p}\hat{b} + \underline{m(p,0)} - \Sigma_{c\in p^\cdot - \{a\}}\hat{c}\right)$$

This system of equations characterize the set of correct behaviours of the net only if it does not contain so called "no duration loop", i.e. if it is impossible for a token to participate in the firing of a transition and to come back simultaneously enabling this transition. Otherwise, the set of correct behaviours is only a subset of the solutions of the system of equations: For instance, if the delays associated with both places of the net of figure 3 are zeros, the only equation we get is $\hat{a} = \hat{b}$, though the true behaviour is $\hat{a} = \hat{b} = \underline{0}$, because of the null initial marking.

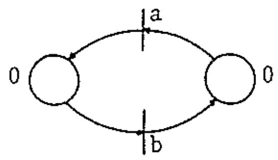

Figure 3

4. PSEUDO EVENTS

In the previous section, we have illustrated the descriptive power of the model. Let us now look for transformation and proof techniques for such descriptions. Starting with an equation such as

$$e = D^\Delta e + \underline{1}$$

the approach taken here consists of giving a sense to the expressions:

$$(1 - D^\Delta)e = \underline{1}$$

and

$$e = \frac{1}{1 - D^\Delta} = \Sigma_{n=0}^{\infty} D^{n\Delta}$$

This is achieved by extending the set of events so as to make the difference a total operator, and by defining an internal product.

Let us first note that, from the definitions of the sum and delay operators over events, the following identity holds for every event e:

$$e = \Sigma_{n=1}^{\#e} D^{e(n)} \underline{1}$$

Our extension of the set of events staightforwardly results from this identity.

4.1 Definition

A pseudo event is a formal series

$$x = \Sigma_{n=1}^{\#x} \bar{x}_n D^{x(n)}$$

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, but in the latter case, the sequence (x_n) converges towards infinity.

The pseudo event 0 is such that #0=0. With each pseudo event x can be associated in a one to one way its counter μ_x defined by:

$$\mu_x = \lambda t.\ \text{if } x=0 \text{ then } 0 \text{ else } \Sigma_{x_n<t} \bar{x}_n$$

The set R of pseudo events is provided with the usual sum and product operators over formal series. (R,+,×) is an integral, commutative ring with neutral elements 0 and $1=D^0$.

A partial order is defined over pseudo events as follows:

$$a \leqslant b \;=\; \forall t \in \mathbb{T},\ \mu_a(t) \leqslant \mu_b(t)$$

(R,⩽) is a lattice, and the sup and inf operators are the corresponding operators on counters.

An event is either 0 or a pseudo event with positive coefficients $\bar{x}_n$. Thus its counter is an increasing function of t. One can see that these definitions are consistent with the previous ones given in sections 1 and 2, with the following loosened notations:

Since, for every pseudo event a and every k in $\mathbb{N}$, ka=ka, we shall omit henceforth to subline the primary pseudo events. Since 1 is the neutral element of the product,it will be omitted in products. So D^Δ will denote the event $D^\Delta 1$. Notice that, with these notations, the expression $D^\Delta a$ may be viewed either as the D^Δ operator applied to a, or as the product of D^Δ by a. More generally, every pseudo event may be viewed as an operator on R.

The product operation, and the above notations justify the first step of the process of formal resolution of the equation $e=D^\Delta e+1$. The second step will be justified by the study of invertibility in R (a pseudo event a has an inverse if and only if there exists a' such that aa'=1).

4.2 Euclidean Division

4.2.1 Proposition: A necessary and sufficient condition for a pseudo event x to have an inverse is that $\bar{x}_1 = \pm 1$. Moreover,

$$\frac{1}{x} = \bar{x}_1\ D^{-x(1)}\ \Sigma_{n\geq 0}\ y^n ,$$

where

$$y = \text{sign}(-\bar{x}_1)\ \Sigma_{n=2}^{\#x}\ \bar{x}_n D^{x(n)-x(1)} ,$$

and y^n denotes the n times iterated product of y by itself.

4.2.2 Corollaries:. Let a=1-e, where e is an event such that e(1)>0, then the inverse of a is an event, since $1/a = \Sigma_{n\geq 0} e^n$.

. A necessary and sufficient condition for the inverse of an event e to be an event is that $e=D^\Delta$ for some Δ. Obviously $1/D^\Delta = D^{-\Delta}$. So $\{D^\Delta \mid \Delta \in \mathbb{T}\}$ is the set of unity elements of the semiring (E,+,×).

4.2.3 Ring norm: Let us recall that an application ν from a ring R to $\mathbb{N}$ is called a ring norm if and only if:
- ν(x) = 0 <=> x = 0
- ν(xy) = ν(x)ν(y)
- x has an inverse if and only if ν(x)=1

So the application ν, which associates with each pseudo event a≠0 the integer $|\bar{a}_1|$, and such that ν(0)=0 is a ring norm on R.

4.2.4 Proposition: R is an Euclidean ring, i.e. for every a,b in R, (b≠0), there exist q,r in R such that a=bq+r and ν(r) < ν(b).

Let us give the division algorithm, which is very close to the polynomials division according to increasing variables powers:

- Step 0: Let $r^{(0)}= a$ and $q^{(0)}= 0$;
- Step k+1: If $|\bar{r}_1^{(k)}|<|\bar{b}_1|$ then stop. Else ,let $x^{(k)}= \bar{r}_1^{(k)}/\bar{b}_1$. If $x^{(k)} \notin \mathbb{Z}$ then go to step α, else let:

$$p^{(k)}=x^{(k)}D^{r^{(k)}(1)-b(1)} ,\ q^{(k+1)}= q^{(k)}+ p^{(k)},$$

$$r^{(k+1)}= r^{(k)}- p^{(k)}b$$

- Step α: Let x be the smallest integer greater than $x^{(k)}$ if $x^{(k)}$ 0, the greatest integer smaller than $x^{(k)}$ otherwise. Let:

$$p= xD^{r^{(k)}(1)- b(1)} ,\ q= q^{(k)}+ p ,\ r= r^{(k)}- pb$$

4.3 Linear Inequalities of Pseudo Events

Our formal calculus is now powerful enough to solve any linear equation. However, behavioural specification in our model makes a very general use of linear inequalities, which are more difficult to handle because of the partial nature of the ordering on R. So, let us examine some properties of this ordering in relation with algebraic operators.

4.3.1 Inequalities and sum: For every a,b,c in R, $a > b \;=\; a+c > b+c$. In other words, the sum and difference operators are order preserving.

4.3.2 Inequalities and product: A great deal of works concerning ordered algebraic structures (see for instance [14]) make the hypothesis that positive product is order preserving, that is to say, that for every a,b,c:

$$a > b \;\&\; c>0 \Rightarrow ac > bc$$

This hypothesis is obviously false in R: For instance $1-D$ is positive, but $(1-D)^2 = 1-2D+D^2$ is not. So let us consider the set **Mon**(R) of order preserving pseudo events:

$$\mathbf{Mon}(R) = \{ x \in R \mid a \in R \;\&\; a > 0 \Rightarrow ax > 0 \}$$

It can be easily shown that **Mon**(R) = E.
Example: Let us consider the two inequalities :

$$x(1-D^{\Delta}) < 1 \qquad (1)$$

$$x < \frac{1}{1-D^{\Delta}} \qquad (2)$$

(1) means that x cannot have two occurrences separated by a delay smaller than Δ (cf.3.1). Since $1/(1-D^{\Delta})$ is an event, we may multiply by it the two members of (1), so (1) implies (2). But the converse is false, because $1-D^{\Delta}$ is not an event: Figure 4 pictures an event satisfying (2) but not (1), with Δ=4.

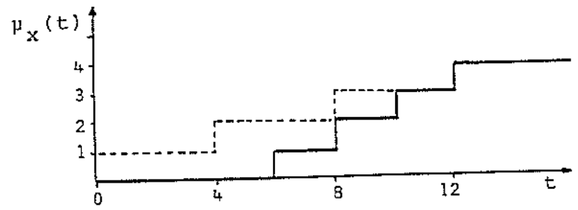

Figure 4

5. APPLICATION TO DESIGN PROBLEMS

In this section, we shall illustrate the use of the calculus on pseudo events on two simple problems.

5.1 First Example

A system receives two strictly periodic sequences of input requests. The former sequence starts from the instant 0, with a 2 seconds period, and the later one starts from the instant 1, with a 4 seconds period. The system is made of n identical processors, each of which takes 7 seconds for processing a request belonging to the former sequence, and 5 seconds for processing a request from the later one. This system may be represented by the timed Petri net of Figure 5.

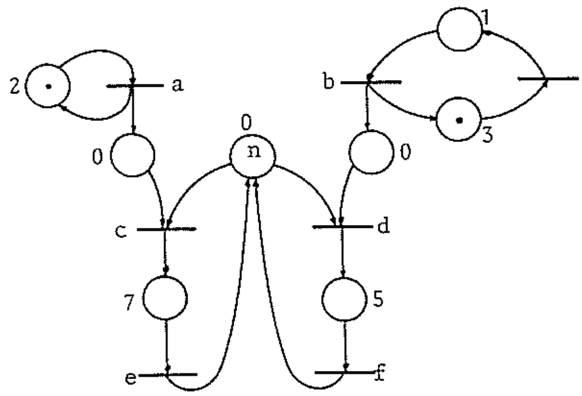

Figure 5

The question is: What is the minimum number of processors needed so as to take into account every request as soon as it happens.

In our model, this problem may be stated as follows: Let $\hat{a}$, $\hat{b}$ be the events respectively associated with input arrivals from each sequence. Let $\hat{c}$, $\hat{d}$ respectively represent the event "an input from the former (resp. later) sequence is taken into account by some processor", and $\hat{e}$, $\hat{f}$ respectively represent the event "a processor ends processing an input from the former (resp. later) sequence". Then:

. The specification of input sequences may be written:

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

. Since a request cannot be taken into account before its arrival, we have:

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

. The processing times of requests are 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, we get:

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

. Finally, the immediate handling requirement provides:

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

Now, (1) reduces to

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

So getting rid of any event variable, the problem may be restated as follows:

"Find the least integer n, such that

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

or "what is the maximum value of the counter of the pseudo event

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

Now, we can perform the division in x, until getting:

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

$-D^7(1-D)/(1-D^4)$ is a periodic pseudo event, the counter of which can easily be shown to have the maximum value 0. Thus, the maximum value of the counter of x is the one of $1+D+D^2+D^4+D^5$, which is 5 (see figure 6). So n=5 is the solution.

Figure 6

5.2 Second Example

Let us consider two processes p_1 and p_2, sharing an exclusive resource. Each process p_i cyclically asks for the resource, uses it during a delay δ_i , then releases the resource and works during a delay Δ_i, (δ_i, $\Delta_i > 0$) , after what it comes back asking for the resource. This system is represented by the net of Figure 7.

Now assume that the resource is very expensive and is required to be permanently used. The problem is: What condition must satisfy the delays δ_1, Δ_1, δ_2, Δ_2, to achieve this requirement?

With the notations of the net, the problem may be stated as follows: Find a necessary and sufficient condition on the delays δ_i, Δ_i so that

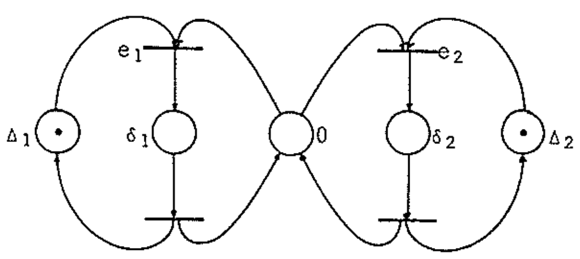

Figure 7

the following system S admits a solution (e_1, e_2) in E×E:

$$S \begin{cases} e_i(1 - D^{\delta_i+\Delta_i}) < 1 \quad , \; i=1,2 \\ e_1(1 - D^{\delta_1}) + e_2(1 - D^{\delta_2}) = 1 \end{cases}$$

Now, since $e_1(1-D^{\delta_1}) + e_2(1-D^{\delta_2}) = 1$ which is an event, then $D^{\delta_1}e_1 + D^{\delta_2}e_2$ must be a subsequence of $e_1 + e_2$. On the other hand, since $\Delta_1 > 0$ and $e_1(1-D^{\delta_1+\Delta_1}) < 1$, $D^{\delta_1}e_1$ has no simultaneous occurrences with e_1, for one can easily show that for every integer n:

$$e_1(n+1) > (D^{\delta_1}e_1)(n) > e_1(n)$$

So $D^{\delta_1}e_1$ (respectively $D^{\delta_2}e_2$) must be a subsequence of e_2 (resp. e_1).

$e_1-D^{\delta_2}e_2$ and $e_2-D^{\delta_1}e_1$ are events and their sum equals 1, so one of them must be equal to 1 and the other to 0. Therefore:

S = S_{12} <u>or</u> S_{21}, where

$$S_{ij} = \{ e_i = \frac{D^{\delta_j}}{1-D^{\delta_1+\delta_2}} \;\&\; e_j = \frac{1}{1-D^{\delta_1+\delta_2}} \;\&\; \frac{D^{\delta_j}(1-D^{\delta_i+\Delta_i})}{1-D^{\delta_1+\delta_2}} < 1 \;\&\; \frac{1-D^{\delta_j+\Delta_j}}{1-D^{\delta_1+\delta_2}} < 1 \}$$

So, a solution satisfying S exists if and only if

$$\frac{1-D^{\delta_1+\Delta_1}}{1-D^{\delta_1+\delta_2}} < 1 \quad \text{and} \quad \frac{1-D^{\delta_2+\delta_2}}{1-D^{\delta_1+\delta_2}} < 1$$

which is equivalent to

$$\delta_1+\Delta_1 < \delta_1+\delta_2 \text{ and } \delta_2+\Delta_2 < \delta_1+\delta_2$$

The final necessary and sufficient condition is

$$\Delta_1 < \delta_2 \text{ and } \Delta_2 < \delta_1$$

6 APPROXIMATE ANALYSIS USING DISCRETE TRANSFORM

In section 5, we have given some examples of the use of the formal calculus in proving properties about behavioural specifications. Of course the proofs performed there may have appeared rather ad hoc, and are not susceptible of systematization. On the other hand, it has been shown in §4.3, that the non monotonicity of the product over pseudo events may give rise to difficult problems in dealing with linear inequalities. In this section, we shall propose a systematic method providing approximate results, even when such difficulties arise.

Our definition of pseudo events by means of formal series of the delay operator D is very close to discrete transform techniques widely used in the field of finite difference equations. Nevertheless, to our knowledge, those techniques never have been applied to inequalities.

6.1 Definition: For every pseudo event $a=\Sigma_{n=1}^{\#a}\bar{a}_n D^{a(n)}$, let us define the function ϕ_a from $\mathbb{R}^+$ to $\mathbb{R}$, by:

$$\phi_a=\lambda x.\Sigma_{n=1}^{\#a}\bar{a}_n x^{a(n)}$$

ϕ_a is generally a partial function, only defined on an interval $[0,r_a[$, where r_a is the convergence radius of the series.

6.2 Theorem: If a is a positive pseudo event, then ϕ_a is positive on the interval $]0,\min(1,r_a)]$. The converse is not true.

6.3 Example of application: Let us come back to example 5.2. We want the system S to have a solution, where

$$S=\begin{cases} e_i(1-D^{\Delta_i+\delta_i})<1, \ i=1,2 \\ e_1(1-D^{\delta_1})+e_2(1-D^{\delta_2})=1 \end{cases}$$

Eliminating e_2, we get:

$$S=\begin{cases} e_1(1-D^{\Delta_1+\delta_1})<1 \\ \dfrac{D^{\delta_2}(1-D^{\Delta_2})}{1-D^{\delta_2}}<e_1\dfrac{(1-D^{\delta_1})(1-D^{\Delta_2+\delta_2})}{1-D^{\delta_2}} \end{cases}$$

Now this system admits a solution e_1 only if there exists a real function ϕ ($=\phi_{e_1}$) such that, for every x in $[0,1[$:

$$\begin{cases} \phi(x)\ (1-x^{\Delta_1+\delta_1})<1 \\ \dfrac{x^{\delta_2}(1-x^{\Delta_2})}{1-x^{\delta_2}}<\phi(x)\ \dfrac{(1-x^{\delta_1})(1-x^{\Delta_2+\delta_2})}{1-x^{\delta_2}} \end{cases}$$

which is equivalent to:
$\forall x\in[0,1[$,

$$F(x)=\frac{x^{\delta_2}(1-x^{\Delta_2})(1-x^{\Delta_1+\delta_1})}{(1-x^{\delta_1})(1-x^{\delta_2+\Delta_2})}<1$$

In the neighbourhood of $x=1$, $F(x)\sim\dfrac{\Delta_2(\delta_1+\Delta_1)}{\delta_1(\delta_2+\Delta_2)}$.

So a necessary condition for the system S to have a solution is:

$$\Delta_1\Delta_2<\delta_1\delta_2$$

It is exactly the result provided by the method of [12] to find permanent behaviours of timed Petri nets. Notice that it is only a necessary condition, since the n.s.c. found in 5.2 was:

$$\Delta_1<\delta_2 \text{ and } \Delta_2<\delta_1$$

7 DISCRETE TIME

All the non real time, and most of the real time digital systems make use of a discrete notion of time. This motivates the investigation of particular properties of $R(\mathbb{Z})$ which is done in this section.

7.1 Discrete Derivatives

7.1.1 Definition: If a $R(\mathbb{Z})$, let us call the derivative of a the pseudo event $a(1-D)$.

This denomination is motivated by the following - obvious, but very useful - proposition, which corresponds to the property of real functions, that a function is increasing if and only if its derivative is positive:

7.1.2 Proposition: A necessary and sufficient condition for a pseudo event a in $R(\mathbb{Z})$ to be an event, is that its derivative is a positive pseudo event.

Example: Let us come back to the example given in 3.3. As announced there, we are now able to express the condition on $\hat{c}$ for $D^3\hat{c}+D\hat{c}-\hat{c}+D$ to be an event, which is:

$$D(1-D)>\hat{c}(1-D-D^3)(1-D)$$

7.2 Linear Inequalities and Fixed Points

Notations: For each a,b in $R(\mathbb{Z})$, let us define:
- $[\![a) = \{x\in R(\mathbb{Z}) \mid a<x\}$
- $(b]\!] = \{x\in R(\mathbb{Z}) \mid x<b\}$
- $[\![a,b]\!] = [\![a) \cap (b]\!]$

7.2.1 Proposition: For each a,b in $R(\mathbb{Z})$, $[\![a)$ (respectively $(b]\!]$, $[\![a,b]\!]$) is a complete inf-closed semilattice (resp. sup-closed semilattice, lattice), i.e. every subset of $[\![a)$ (resp. $(b]\!]$, $[\![a,b]\!]$) has a greatest lower bound (resp. a least upper bound, a least upper bound and a greatest lower bound).

Notice that $R(\mathbb{R})$ does not satisfy this property: For instance, the sequence:

$$\left(x_n=D^{\frac{2n-2}{2n-1}}-D^{\frac{2n-1}{2n}},n\in\mathbb{N}^*\right)$$

is included in $[\![0,1-D]\!]$, but has no least upper bound in $R(\mathbb{R})$.

7.2.2 Proposition: Let us recall that a function f from R to R is said to be latticecontinuous, if and only if, for every subset X of R admitting a least upper bound $\bar{x}$, (resp. a greatest lower bound $\underline{x}$) the set $\{f(x)\ x\ X\}$ admits a least upper bound $\bar{y}$ such that $\bar{y}=f(\bar{x})$ (resp. a greatest lower bound $\underline{y}$ such that $\underline{y}=f(\underline{x})$.

Then, for every Δ in $\mathbb{T}$ and every pair (f,g) of lattice continuous functions, the functions $\lambda x.D^{\Delta}x$, $\lambda x.f(x)+g(x)$, $\lambda x.\inf(f(x),g(x))$, $\lambda x.\sup(f(x),g(x))$ are lattice continuous.

7.2.3 Application: Let us consider a system of linear inequalities in $R(\mathbb{Z})$, of the following form:

$$S=\{\ x(1-e_i)<b_i,\ i=1..n\ \}$$

where all the e_i are events such that $e_i(1)>0$.

Then the set P of solutions of S is the set of pre-fixed points of the function $f_S = \lambda x..\inf_{i=1..n}(b_i + e_i x)$, which is lattice continuous.

On the other hand, from 4.2.2 and 4.3.2, we have:

$$x(1-e_i) \leq b_i \Rightarrow x \leq b_i/(1-e_i)$$

So P is included in $\rrbracket\beta\rrbracket$, with $\beta = \inf_{i=1..n}(b_i/(1-e_i))$. Since $\rrbracket\beta\rrbracket$ is a sup-closed semilattice, if P is not empty, it admits a least upper bound $\bar{\beta}$. By Tarski's fixed point theorem, $\bar{\beta}$ is the greatest fixedpoint of f_S. Furthermore, the sequence $(f_S^i(\beta) \mid i \in \mathbb{N})$ is included in the complete lattice $\llbracket \bar{\beta}, \beta \rrbracket$ and by Kleene's fixed point theorem, it converges towards $\bar{\beta}$. Note that P is generally only included in $\rrbracket\bar{\beta}\rrbracket$. The point is that by this process, we can add to S a new inequality, which is implied by S and may be saturated, since $\bar{\beta} \in P$.

<u>Example:</u>
Let us consider the following system of inequalities:

$$\begin{cases} \dfrac{x}{1-D} \leq \dfrac{1}{1-D^3} \\ x \leq 1 - \dfrac{D^4}{1+D} \end{cases}$$

Neither of the two inequalities may be saturated by x without violating the other. But the system reduces to:

$$X = \frac{x}{1-D} \;,\; X \leq f(X)$$

with $f = \lambda X.\ \inf(\frac{1}{1-D^3}\ ,\ 1 - \frac{D^4}{1+D} + DX)$.

Using the above notations, we get:

$$\beta = \inf(\frac{1}{1-D^3},\ \frac{1}{1-D} - \frac{D^4}{1-D^2}) = \frac{1}{1-D^3}$$

Let us compute the greatest fixed point $\bar{\beta}$ of f, smaller than β. We get:

$$\beta_0 = f^0(\beta) = \beta = 1/(1-D^3)$$

$$\beta_1 = f^1(\beta)$$

$$= \inf(1/(1-D^3),\ 1-D^4/(1+D)+D/(1-D^3))$$

$$= 1 + (D^3+D^7)/(1-D^6) \quad \text{(see Figure 8)}$$

$$\beta_2 = f^2(\beta)$$

$$= \inf(1/(1-D^3),\ 1-D^4/(1+D) + D + (D^4+D^8)/(1-D^6))$$

$$= \beta_1$$

So,

$$\bar{\beta} = 1 + \frac{D^3+D^7}{1-D^6}$$

and the initial system implies:

$$\frac{x}{1-D} \leq 1 + \frac{D^3+D^7}{1-D^6}$$

7.3 Application: Interruption Modeling

As a last illustration of the descriptive

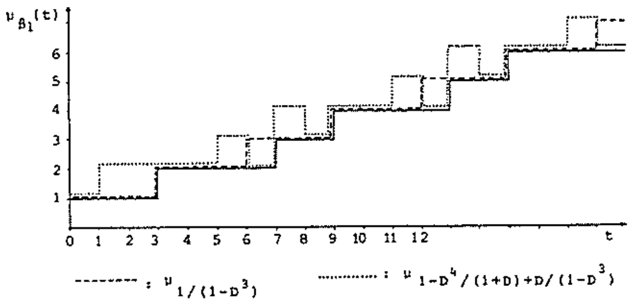

Figure 8

power of our calculus, let us consider the description of a task that needs a delay $\Delta \in \mathbb{N}$, but may be interrupted on every integer instant. The task is assumed to be non reentrent.

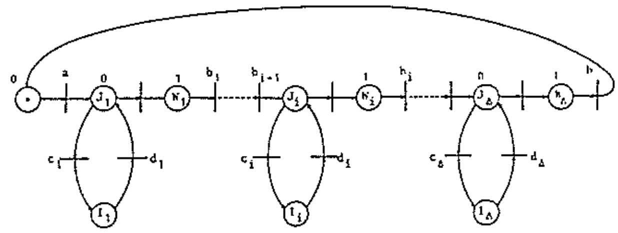

Figure 9

Modeling this task leads to a very complex timed Petri net (Figure 9). In this net, the transition a represents the beginning of the task. When the token reaches a place J_i, the task may be either immediately interrupted by the firing of c_i, then entering the interrupted state I_i until reactivated by the firing of d_i, else continuated for one unit of time in W_i before becoming again interruptible. b $(=b_\Delta)$ represents the end of the task.

<u>Proposition:</u> Let $\hat{a}, \hat{b}, \hat{c}, \hat{d}$ be the four events respectively representing the beginning, the end, the interruption and the reactivation of the task. Then, given $\hat{a}, \hat{c}, \hat{d}$, the event $\hat{b}$ is uniquely determined by the folowing relation:

$$0 \leq \frac{\hat{a} + \hat{d} - \hat{c} - \hat{b}}{1 - D} - \Delta\hat{b} \leq \Delta$$

The proof is rather tedious [5], but completely formal, and the result proved is not trivial and may be used to deal with systems with interruptible tasks in a very simpler way than by means of timed Petri nets.

CONCLUSION

This paper has presented a model for real time and parallel systems, and a set of results allowing, to some extend, the transformation and analysis of the description of these systems in the model. This work must be extended particularly in two directions:

First, the power of the calculus must be increased. We have shown that a great deal of problems involve investigations on systems of linear inequalities. For instance, let us consider two communucating asynchronous processes like in CCS [7]. Assume each process may be described by a system of linear inequalities over its external events. Then the resulting process will be described by the conjunction of the two systems, where the interprocesses communication events have been equalized and eliminated. So we must be able to eliminate a variable from a system of linear inequalities without loosing any information about the remaining variables. Furthermore, many problems, and particularly scheduling problems, may be expressed by linear optimization problems over (pseudo) events. But the partiel nature of the ordering relationship gives raise to a lot of difficult questions in applying linear programming techniques.

Another future extension concerns numerical systems. One way is to combine the results obtained by our calculus with classical techniques of program analysis. Another possibility is to extend the model to deal with variables. This was done in [4] for specification purposes, but the extension of the calculus to such a widened model is far to be obvious.

In spite of these questions, the model presented here seems to us a powerful tool to describe and analyse the behaviour of parallel and real time systems, and a unifying framework for a lot of problems in this field. Of course this approach is not considered as concurrent to the classical state-transition ones, but is expected to lead to complementary results.

REFERENCES

[1] C.André, F.Boeri, "The behaviour equivalence and its applications in Petri nets analysis". Journées d'étude AFCET, schemas de contrôle des systèmes informatiques. Paris (September 1979).

[2] E.A.Ashcroft, W.W.Wadge, "LUCID: A non procedural language with iteration". CACM, vol.20, n°7 (July 1977).

[3] P.Caspi, N.Halbwachs, M.Moalla, "Approche comportementale pour la spécification des systèmes temps réèl". Journée d'étude AFCET, spécification. Toulouse (September 1980).

[4] P.Caspi, N.Halbwachs, An approach to real time systems modeling. R.R.n°253, IMAG Laboratory, Grenoble (June 1981).

[5] P.Caspi, N.Halbwachs, Algebra of events: A model for parallel and real time systems. R.R.n°285, IMAG Laboratory (January 1982).

[6] L.Lamport, "Time clocks, and the ordering of events in a distributed system". CACM, vol.21, n°7 (July 1978).

[7] R.Milner, A calculus of communicating systems. Lecture notes in computer science, n°92. Springer Verlag (1980).

[8] R.Milner, "On relating synchrony and asynchrony", CSR-75-80, Edinburgh University (November 1980).

[9] J.L.Peterson, "Petri nets", ACM Computing Surveys, vol.9, n°3 (September 1977).

[10] D.P.Reed, R.K.Kanodia, "Synchronization with eventcounts and sequencers", CACM, vol.22, n°2 (February 1979).

[11] P.Robert, J.P.Verjus, "Towards autonomous description of synchronization modules". Proc. IFIP Congress, Toronto (1977).

[12] J.Sifakis, "Use of Petri nets for performance evaluation". Measuring,modeling and evaluating computer systems. North Holland Pub. Co. (1977).

[13] J.Sifakis, Le contrôle des systèmes asynchrones: Concepts, propriétés, analyse statique. Thesis, Grenoble University (June 1979).

[14] U.Zimermann, Linear and combinatorial optimization in ordered algebraic structures. Annals of discrete mathematics, n°10, North Holland Pub. Co. (1981).