A DYNAMIC THEORY OF DISCRETE-EVENT-SYSTEMS

P. Moller

May 1987
WP-87-44

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
A-2361 Laxenburg, Austria

FOREWORD

This paper shows the mathematical problems which appear when a dynamic approach is taken to discrete-event-systems.

A. Kurzhanski
Program Leader
System and Decision Sciences Program.

CONTENTS

1 Introduction to Discrete Event Systems.

1) Simulation of Discrete Event Systems.

The expression "Discrete event system" (DES) first appeared in the engineering litterature, in the 1970's, and was related to simulation tools (see [SIM1] to [SIM4]). Some problems seemed to be beyond the reach of analytical mathematical models, and the reasons for this feeling were of two kind:

Problems of combinatorial complexity were involved, which had no explicit solution.

Complex systems involved several subsystems of different nature, which could hardly be merged into a global analytical model.

For instance, in a manufacturing plant, "classical" system theory and control theory are used to model the behaviour of the various machines and robots. The flow of products beeing manufactured and are travelling from one machine to another, appears as a system of "higher level". It would be impossible to analyse this flow of objects by a quantitative description of their state during the evolution of the system (analytical description of the shape of parts beeing produced, movements of the machines etc..).

Existing mathematical tools dealing with these problems were basically those of operationnal research, but they provided few help. Thus the answer of the engineering community was to develop software packages to do the required simulation. A lot of pacakges are now available commercially (see ref. [SIM2]).

What does distinguish the discrete event system simulation from other simulation techniques ?

The basic Idea is the intoduction of event driven simulation.

This means that no continuous time description of the state of the system is used, but this state is only considered at those instants when events happen: this implies that the next instant, which will be considered during the performance of a simulation, is computed from the present result of the simulation, instead of being chosen independently.

In the litterature, one can find the following informal definition of an event:

> "Within the framework of discrete event simulation languages, an event has a number of interrelated meanings, all based on the fact that the time during simulation is not related to continuous real world time, but is forced to jump discontinuously from event to event"
>
> "The events in a simulation are those particular times when something happens or should have happened."

Independence assumption.

Since one of the purpose of DES is to have a "high level" description of the flow of items in a system, it is generally assumed that the nature of the events (or of the objects they happen on) are not considered; only the relations which bundle the events and the objects are taken into account.

This assumption has some drawbacks: there are numerous examples where this assumption does not hold, for instance if the purpose of the study is to evaluate the simple algorythm:

```
Get integer  x,
Repeat
     x = 2x + 1
      if x is prime then x = 1000.x
                    Else x = x + 3
Until x ≥ 10^6.
```

Clearly, the number of iterations depends on the value of x and it is not possible to estimate it whithout monitoring the value of x. Furthermore, the length required to perform the test "x is prime" is obviously depending on x if any algorythm is used to perform it.

More generally, this assumption does not hold in all computer programs which contain "result depending branching". One solution to get rid of this problem is obvious: if the buffer contains different values, they are related to different states of the system. Unfortunatly, in the previous example, the number of states which ahve to be considered is infinite.

We shall discuss in part 2 this fundamental distinction which has to be made between a system with a finite number of states and a system which can be caracterized by a finite dimensional information.

2 Examples of discrete event systems.

We have mentionned that one of the first purposes of discrete event simulation was to study macro models of temporal relationship between simple systems.

This kind of problems basically appears in the the following domains:

Manufacturing systems: The problem is to model the flow of parts in a workshop, but also the trajectory of machines from one task to another. The complexity of the system increases furthermore if some machines are capable of several different tasks.

Multiprocessor computing systems: With the intoduction of multiprocessor computers, the complexity of the possible behaviours of these computers has dramaticaly increased and an adequate theory to model the flow of information is needed more than ever.

Real time computing systems: The introduction of computing systems which react to an anvironnement which "does not wait" for the results of computations has increased the need for models taking into account the physicall duration of tasks and not only their logical nature.

Theory of networks: The problems in networks are a mixture of two previous ones: there is a need to model the flow of information, the network is subject to "real time" constraints.

In all these systems, two kind of questions are of importance:

Logical validation of the behaviour: if some constraints on the behaviour of the system are given, (which are generally safety constraints), how to prove that these constraints will not be violated ?

Performance evaluation of systems: Here, performance means speed. It is a major concern in manufacturing and computing systems to improve the "production rate" of the system. The optmisation problem of the throughput speed is generally an np complmete cornbinatorial problem. There is very few chance that any system technique may help to break this n p complete problem, but it can provide efficient tools to evaluate the performance of a given configuration, or of given heuristsic optimisation rules, in order to compare them.

All these problems share a common fact: some information about the dynamics of the system is known. By dynamics of the system, we mean precedence constraints on the tasks which enable to partially order these tasks, and temporal constraints which will rule the evolution of the system in real time: duration of tasks and external time constraints on these tasks.

3 Hidden Markov Models and identification of discrete event systems.

Recently, it appeared that some problems can be interpreted as identification problems of a discrete event system.

Suppose that we have a complex system which is described by some model and in which some kind of measurements of the steady state behaviour is available.

Any breakdown of some part of the system will result in a change of the steady state behaviour. In this framework, three questions are of interest:

Estimation of the sensitivity of the system to breakdowns, in order to reduce this sensitivity.

Diagnostic on the level of operation of a system; if this cannot be checked directly, it can be estimated from the perturbations in some outputs.

Estimation of dynamic relations between the breakdowns, in order to see if there is any "hidden model" ruling these breakdowns, eventually linking them wit external events.

The first two questions can be viewed as identification problems in the classical sense; it is only needed to monitor the changes in the behaviour of the system.

Clearly, only the last question is realy relevant to discrete event systems, when it is needed to model the dynamic relationship between events such as breakdowns.
This kind of approach is intensively used in another field which is not obviously related to discrete event systems: pattern recognition and speach recognition.

Patterns, or signals are represented as systems which have discrete changes in their dynamics; they switch from one steady state behaviour to another and the information to be retrieved is contained in these switches. It is important to note that these transition define a discrete event dynamic behaviour, and that the states are steady state behaviours of lower level dynamic systems.

It is probably not relevant to use tempral measurements to study the patterns in a static image, but it is certainly of great importance in the analysis of human speach, where the relative durations of the steady state behaviours are of great importance.

Again, the idea arises that discrete event systems can be higher level systems describing the relation between low level systems.

All the existing approaches to these problems use "hidden Markov Models". An introduction to this theory and its application to speach recognition can be found in [RETO3].

If the dynamics ruling these events are not truly stochastic, it may not be adequate to use Markov models to study this problem, but what could be used instead ?

Thus, it appears necessary to give a definition discrete event systems and a mathematical formalism for these sytstems, in order to provide a mathematical framework for the identification problem of the dynamics of the events.

4 Uncertainty and underterminism.

The problem of undeterminism in DES is an importnat issue: In discrete event systems, it appears necessary to distinguish two notions of undeterminism, which we shall call strong underterminism and probabilistic uncertainty, and which we shall informaly define as follows:

Strong underminism occurs when several possible behaviours can occur when the system is in a given state, and when the system contains no infromation about the choice which will be made among the set of next posible states. In other words, ther is an under specification of the system.

Probabilistic uncertainty means that some coefficients in the dynamics of the system are not known exactly, but may be given by a probabilistic distribution. For instance, the duration of certain task may be given through a probabilistic distribution, or this may be the case for the set of tasks which can be started when a system is in a given state.

The second case, probabilistic uncertainty is addressed by stochastic models. In this paper, we shall not stress on the probabilistic theories, but look into detail at the first case, when no probabilistic information is available, to find out how it can be dealt with.

This first case is called undeterminism in computer science, but in the control and system science vocabulary, the best term would be "incomplete specification".

The problem of data dependent switching in computer programs can be viewed as a problem of strong undeterminism, if the current values of the algorythm are not taken into account in the DES model.

A lot of authors reduce the problem of strong "undeterminism" to a problem of "uncertainty" by supposing that a "random decision maker" generates a choice among the possible next states of the system. This enables the use of stochastic models. Attempts have even been made to use this technique in the sequencers of some parallel computers (for instance prototypes of arrays of transputers), to decide where processes are allocated. This raises an important question:

Is this an efficient method for making the system work, or is it an artefact to justify the use of the stochastic models which are being used to study the system ?

One could even consider that the existance of certain uncertainty in the coefficients in the system is used as an excuse to justify the introduction of probabilistic decision models to get rid of the strong undeterminism.

Moreover, it is very questionable to use Gaussian distributions or any of the usual probabilistic distributions to model the distributions of the durations. For instance in a computer, the duration of a calculation which contains no value related decision is completly deterministic, and measured in microseconds; the time necessary to restart the system from a crash is counted in minutes and the time needed to repair any damadged part may be counted in hours. It seems not very reasonable to handle all these parameters in the same probabilistic distribution, which would be supposed to give a description of the duration of the calculation.

5 Towards a formal definition of discrete event systems.

Behind any work on discrete event systems, there is mathematical theory which is used to analyse the "real world".

Even computer simulation programs refer to a background mathematical theory; the present trend in mathematical models used in simulation of DES is presently queuing networks (see ref.[QUNE1] to [QUNE3]).

In any scientific domain, the tools and their applications should be distinguished. We have mentionned in the previous paragraphs to which problems discrete event systems theory is related, but our purpose is to identify a common mathematical background for these systems.

The existing mathematical theories can be classified in the following way:

- Graphic tools.
- Queuing networks and stochastic models.
- Algebraic automata theory.
- New discrete maths theories.....

Some authors still believe that a purely mathematical theory of these systems is not possible (ref. [OADE1] and [OADE2]) and that no analytical approach can be useful without the use of some simulation.

Our purpose is to identify a new theory which would give an adequate framework to study DES. This theory should stress on the "event driven" modelling of systems, which is fundamental in the DES approach, and be dynamic.

In part 2, we are going to investigate how the notions of events, event sequences and discrete event systems, can be formalised. We shall compare these notions whith the notion of states and state transitions which are the fundaments of automata theory. A DES will be viewed as a an operator on event sequences.

Since all theories are somehow related to graphic models, which are the most convenient approaches for getting qualitative results, we are going to link the various graphic tools with our new formalism in part 4, by associating an operator on event sequences with each kind of node. It will appear that the most explicit graphic theory is given by temporized Petri nets; we shall give a brief introduction on Petri nets.

In part 3, we are going to investigate the problem of causality in our theory of DES. The notion of causality is crucial if one wants to define a dynamic theory. It will appear that several notions of precedence can be introduced, thus leading to different notions of causality.

In part 5, we are going to show that the first framework proposed is too restrictive to handle efficiently an example of dynamic allocations problem. This will lead us to give a generalized definition of DES.

2 Event-sequences

1- Events, time and state of the system.

Before considering the dynamics of a discrete-event-system, the first problem is to give a mathématical description of events. We are going to examine several possible representations of the events and how they are related to existing theories.

Time scale.

We shall first give ourselves a set T of "times". This will be a subset of the extended real line, such that every subset of T has a least bound and an upper bound, making T a complete lattice. For instance:

$T = [-\infty, +\infty]$
$T = [0, +\infty]$
$T = Z$ or $Z \cup \{-\infty, +\infty\}$
$T = N$ or $N \cup \{+\infty\}$

The elements of T will be called "dates"; a closed interval $[t_1, t_2]$ of T will be called a period. The upper bound of T will be denoted $+\infty$ and its least bound $-\infty$.

System and events.

Let us view a system as a finite set S of elements. On each element of the system, events can occur. We have to distinguish the kind of events which are possible, and the actual sequence of events occuring during the evolution of the system. By "kind of events" we mean the set of possble behaviours of the element; several events of the same kind can occur successively on one element.

For the sake of notations, it is more convenient to introduce a set K of all kind of events and to define a function:

$\Lambda: S \times K \rightarrow \{0,1\}$

Such that:

$\Lambda(s, k) = 1$ if and only if events of kind k are possible on element s
$\Lambda(s, k) = 0$ otherwise.

An event can be described by a triple (t,s,k) where:

t is a date in the absoulute-time scale

s is an element of the system

k is a kind of event.

Discrete structure of the events.

In order to introduce discrete-event-systems, which shall put some constraints on the events: We shall assume that the set of events of a certain kind occuring on one element of the system is countable, which is even a stronger assumption than discrete.

This means that every event is completly described by a 4-uple (t,s,k,n) where:

t is a date in the time-scale,

s is an element of the system,

k is a kind of event,

n is an integer which is the number of the event.

It is important to note that in practical problems, the time scale is completly defined only when an "initial instant" is specified; for every event-sequence, an initial vlaue for the numerotation has to be defined.

The evolution of the system can be described by defining a function:

$\Phi : T \times S \times K \times Z \rightarrow \{0,1\}$

Such that:

$\Phi(t,s,k,n) = 1$ if and only if the event of kind k wearing number n occurs on element s at time t

$\Phi(t,s,k,n) = 0$ otherwise.

This gives us the most general description of the set of events, but the dynamic structure is hidden. In order to simplify our notations, we shall assume that only one kind of event can take place on any element of the system, thus we need only to consider a function:

$\Phi : T \times S \times Z \rightarrow \{0,1\}$

This description cancels the dynamic structure of the events. A basic assumption which has to be made in order to make the dynamics appear is to assume that the numerotation of events on an elemnt, which induces a "logic precedence", is coherent with the "temporal precedence"; this means that:

Φ (t,s,n) = 1 and Φ (t',s,n') = 1 implies that:

$t \geq t'$ and $n \geq n'$ or $t \leq t'$ and $n \leq n'$.

Instantaneous events.

We shall also assume that events are instantaneous. Thus we have to introduce another notion, if we want to model a process which has a duration: We shall call it a task.

A task is defined by two events: the beginning of the task and the end of the task. A task can start n one element and finish on another element of the system.

State representation:

We have chosen point of view of considering the events occuring in the system, whithout defining the state of the system. Most of works on discrete-events systems start by a description of the set of states of the systems. Then, events are modelled as transitions between one state and another. This is the point of view which is taken in the two following theories:

-Markovian models.

-Automata theory.

In the Markovian model, the system can switch from one state to a new one out of a set of possible next states. Each new state can be reached from the initial one with a given probability. The model is then used to give the probability for the system to be in a given state at a given date.

Recall that the state may be a very complicated mathematical objects; for instance, in the hidden Markov models used in speach recognition, the state is a dynamic system in the usual sense, and the events are switchings from one dynamic system to another.

In the Automata model, all possible next states are considered, whithout any consideration of probabilities. Thus, this theory appears as an enumeration tool of all possible behaviours of the system.

Thus, if our definition of a discrete-event-system is taken as a starting point, it is straightforward to define the states as being "the systemas it is between events". We could give the following formal but useless definition of the set states: it is the subset of $(TXSXZ)^2$ of all couples of triples $((t,s,n),(t',s',n'))$ such that $\Phi\ (t,s,n) = 1$ and $\Phi\ (t',s',n') = 1$ and $t < t'$. Sometimes, an equivalence relation can be given on the previous set in order to reduce the set of states.

Anyway, in often apperas that this construction leads to a huge (even infinite) set of states, as we are going to show:

Discussion on state-representation.

Let us consider the following example:

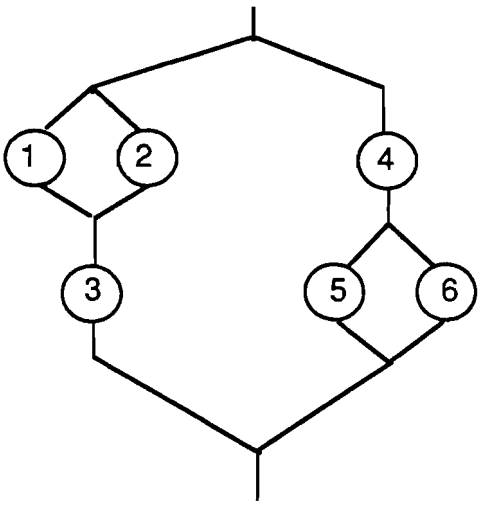


Fig1: Six task system.

This flowchart shows the logic ordering between 6 tasks in a system. Their would be 2^6 possible states for this system, not all of which can be reached.

If we study only the reachable states, we can reduce this number to 24 states, which is still a large number, when compared to the relativly simple system considered. These reachable states can be described by the following state-transition diagram:

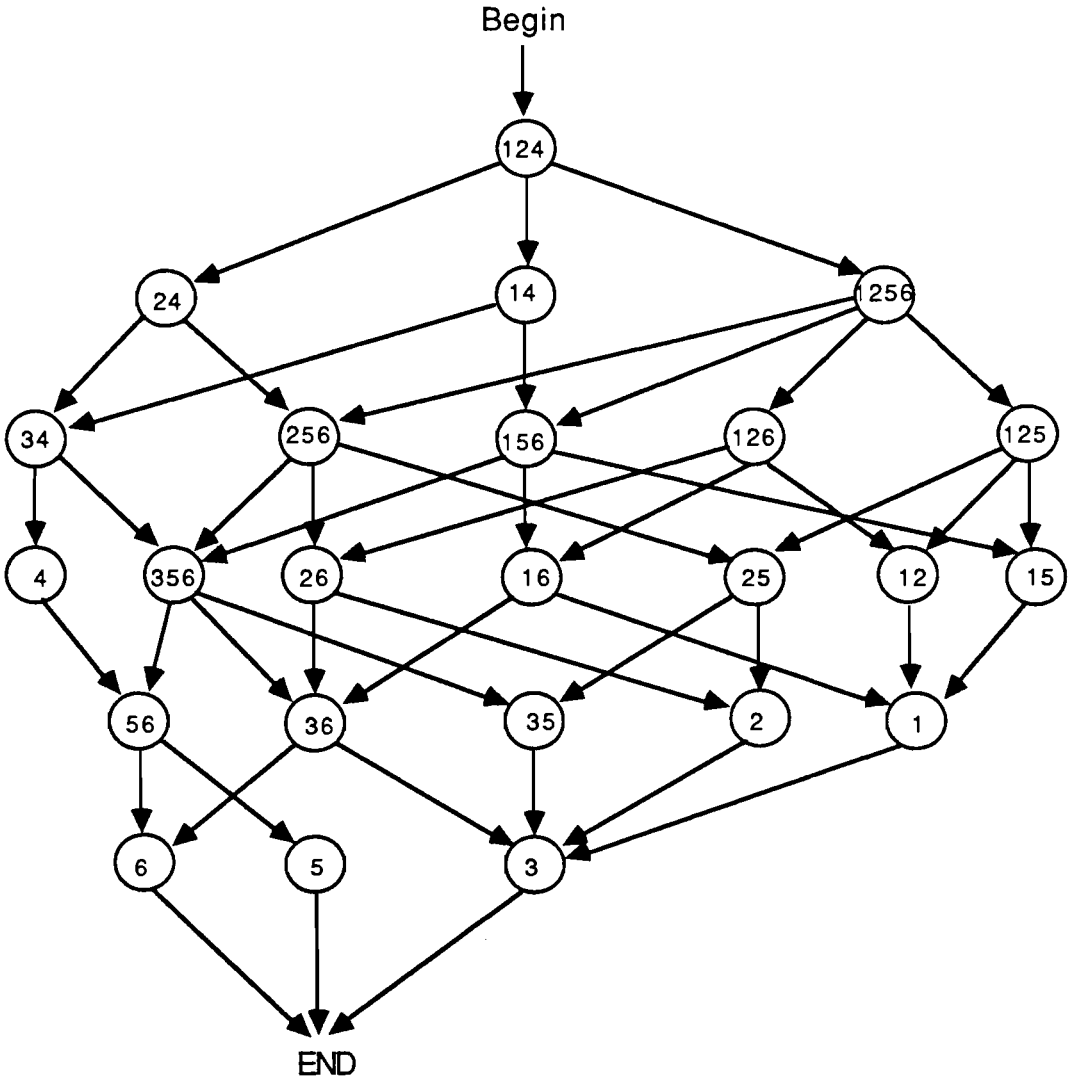


Fig2: State-transition diagram.

The study of the close-loop behaviour of this system would still increase the number of states to consider.

Another example is going to show us that the set of states may be infinite, even if the system is "finite" in the following sense:

- The system consists of a finite set of elements.
- A finite number of events are possible.
- The time is measured by integers.

Suppose that a Machine M1 sends parts to a machine M2 through a buffer B, and suppose that M1 produces 2 items per unit of time, but M2 operates only 1 item by unit of time.

Obviously, the number of items in the buffer will raise to $+\infty$, thus the number of states is not finite. Nevertheless, the state of the system can be represented by a one-dimensional measurement: the number of items in the buffer, and it appears that this is obviously a better description of the state of a system than the enumeration of all its possible states.

Conclusion.

In all of these examples, the set of possible states of the system is much larger than the number of "elements" of th esystem. Thus, it may be wiser, or more efficient to consider the behaviour of the elemnts, by trying to give a quantitative measurement of their behaiviour, rather than eneumerate all the possible states.

For this purpose, we are now going to look more into details how the sequences of events can be described.

2- Dating and counting functions.

From now on, we shall take $T = Z \cup \{-\infty, +\infty\}$ as the time scale.

Let us consider only one element of the system and one kind of event, and let's try to find a mathematical description of the event-sequence of this kind occuring on this element. We shall study later how different sequences can be linked.

We have assumed (in order to have a discrete-event system) that the events can be enumerated by a sequence of integers whose order is coherent with the time scale; this means:

If $n < p$, event number n takes place before or at the same date as event p.

We can consider that each elements of the system defines a local clock, which generates a "local time" which is measured by the events. The number of an even taking place is the date defined by this local clock.

In computer science, this local time is often called "logic time" , since it it is associated with the precedence of the events. This notion of "logic precedence" may be confusing:

In a program, a partial ordering is defined on the tasks which have to be performed. The implementation of the program on a machine must be consistant with this order: if a calculation α logically preceeds β, then α must physically be performed before β. The converse is not always true: if the calculation α is performed before β, this does not imply that α needs to be performed before β; this order may only result from a choice of implementation.

This shows that the notion of logic preceedence is weaker than the ordering of events which we have introduced.

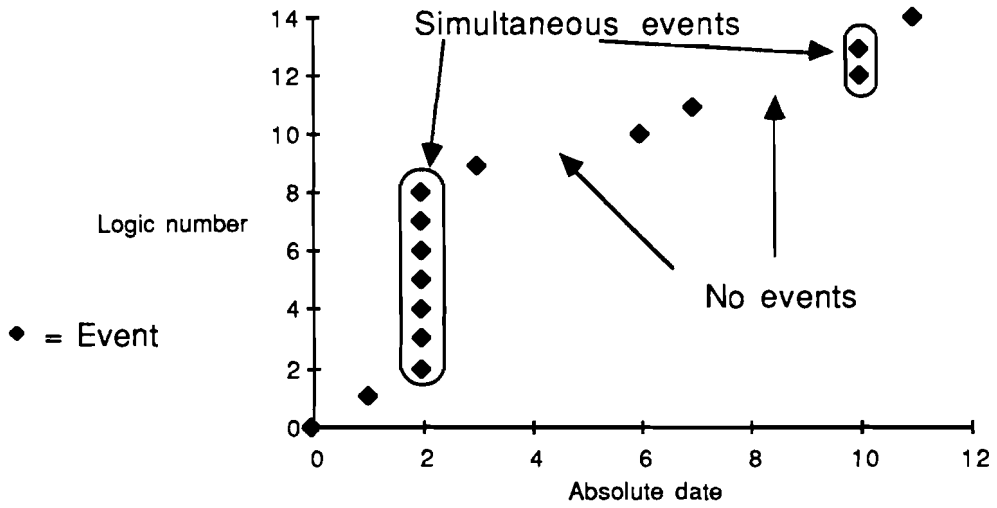
Each event is described by two measurements:

- Its local date or number.
- Its absolute date.

The set of those events associated with one element of the system will be called an Event-Sequence.

The Event-Sequence thus appears as a two-dimensional information.

Let us consider an example:



This chart shows us a sequence of 15 events:

Event number 0 occurs at absolute date 0
Event number 1 occurs at absolute date 1
Events number 3 to 9 occur at date 2
Event number 10 occurs at date 3
Event number 11 occurs at date 6 and so on..

The set-theory tells us that it is a "relation", which could be formally defined by

$t\ R_S\ n$ if and only if

$\Phi(t,s,n) = 1$

Where s is the elemnt of the system which is considered.
Φ is the function introduced in paragraph 1.

Unfortunatly, there are no efficient tools for computations involving. Our problem is to code efficiently such a bidimensional information, in order to make calculations.

In [EVSE1], the authors introduce a representations of this relation as a formal series in 2 commutative variables. An event sequence will be coded as a series:

$$\Sigma_{p,q \in Z}\ b(p,q)\gamma^p.\delta^q$$

Where b(p,q) is a Boolean coefficient which is equal to 1 if and only if an event wearing number p takes place at time q.

The main interests of this approach, is that, under strong assumptions on the dynamics of the system, whicht we shall not discuss here, the authors can factorize the series representing the event-sequence in a very compact way.

This representation has some major drawbacks, because it leads to new algebraic problems which have not been extensively studied yet:

- It is necessary to introduce the following cancellation constraints, to suppress events which are meaningless:
  $\gamma^p.\delta^q + \gamma^r.\delta^s = \gamma^p.\delta^q$ if $p \geq r$ and $q \leq s$

- The formal variables γ and δ commute and there are very few results available on formal series in commutative variables.

One important result on non-commutative variables, the Kleene-Schützenberger theorem has been recently extended by the authors in a way which applies to these series in γ and δ; we shall give this theorem in the last part of this paper.

Avoïding two-dimensional information is possible by transforming this relation between absolute dates and event-numbers into functions. Since we are considering a discrete time scale, our functions will be sequences.

The dating function (or dating sequence) associated with the event sequence is defined as a function D mapping the set of integers Z into the time scale T, such that:

D(n) = absolute date of event number n, if it exists.

This definition is extend to all integers in the following way:

If n is strictly smaller than the least defined event number, $D(n) = -\infty$
If n is strictly larger than the greatest defined event number, $D(n) = +\infty$

This means that if the first event wears number 8, we define pseudo-events whith numbers $p<8$ which have occured at the origin of times, namely $-\infty$.

If the last event wears number 12, we create pseudo-events with numbers starting from 13 and which do not occur in finite time, which means that their date is $+\infty$.

The dating sequence is non-decreasing, thus it is possible to define a quasi-inverse by the formula:

$$C(t) = \sup \{ n , D(n) \leq t\}$$

This is called the couning function or sequence, which generalises the numbering of events; this function is defined at every date, not only those when an event occurs. Clearly, C(t) is the largest number of the events which have occured not later than t. It should be stressed that this number refers to the numerotation of events, not to the amount of past events. Both notions differ if the numerotations does not start at n=1.

It is easy to check that this counting sequence is also non-decreasing, and that the dating sequence can be retrieved from it by the following formula:

$$D(n) = \inf \{ t, C(t) \geq n\}$$

Other quasi-inversions would have been possible, which would have led to slightly different counting or dating sequences.

The information carried by these two sequences is exactly equivalent. The relationship between both have extensively been studied by P.CASPI and N.HALBWACHS in [EVSE5..7]; they have shown that calculation on these sequences is possible in the usual real-numbers algebra, after applying a Laplace-transform. Unfortunatly, to get a complete calculus, they have to introduce sequences with no physical meaning. Furthermore, this approach gives a convenient calculus on dating sequences and counting sequences, but completly cancels the complexity of the information that they carry.

A slightly different approach was taken by G. COHEN et al. . in [EVSE1..4]. They have started from a dynamic study of the dating sequences, then shown that these sequences can be coded as formal series in one variable. If the time scale is discrete, the counting function is also a sequence and can also be coded as a formal series in another formal variable. This has led the authors to an attempt to code the inbeded information, which is 2-dimensional as we have seen, by a series in two variables as we have previously-mentionned.

3-Mixing different event-sequences.

It is straightforward to describe the whole system by using several event-sequences, each of these event-sequences being described by one of the tools we have just introduced: dating or counting sequence, 2-D formal representation.

Let us compare this with the use of formal languages, which could be used to describe the sequences of transitions of the system (events) from one state to another. In a discrete-event system, let us condider the set K of all possible events as a formal alphabet.

K^*, the free monoïd generated by K, is the set of all finite strings constructed with the alphabet K.

The string $\alpha.\beta.\beta.$ means that event α takes place, followed by β and by β again.

A language is a subset of K^* and can be seen as a formal series with variables in K and boolean coefficients.

At first glance, there is no notion of real-time in this formalism, but there is an implicit time which is related to the length of strings and the notion of precedence which clearly appears in a strng:

Let us define the date of completion of a sequence, which is modelled as a string of k^*, as the number of letters contained in this sting. This clearly defines an absolute-time scale.

The local scale asociated with an event α is given by counting the occurences of the letter α in the string. For instance, if the string $\alpha.\beta.\alpha$ belongs to the language describing the system, we understand that at absolute date 3, events of kind α have occured twice.

Thus, the information contained in a formal language can be interpreted with our formalism of event-sequences. Is the converse true ?

The difficulty appears when attempting to model that two events α and β happen simultaneously.

In the formalisim of languages and automata, if a language contains both the one-letter strings α and β, the meaning is "α or β can be the first event" and not "α and β take place initially".

The strings $\alpha.\beta$ or $\beta.\alpha$ neither have the requested meaning: they show that α and β may be completed successively at absolute time 2, but not that they occur simultaneously at time 1. Several ways are possible to bypass this difficulty:

The first solution is to enlarge the set of possible events to create events standing for the simultaneous occurence of elementary events. Thus, the mathematical object remains a language, but the increase of events to be considered is dramatic: if K initially contains k kind of events, then the number of possble combinations of simultaneous events is 2^k !

This increase of the number of possible events, is obviously related to the large number of states needed to model a system in a state representation:

A language is the enumeraion of all possible sequences of state-transitions of an automaton, and if the set of possible behaviours is increased, when simultaneous events are taken into account, the number of states to consider needs generally also to be increased.

To avoid this dramatic increase in the size of the set of events, a solution would be to introduce a temporal coefficient associated with each sequence. This would lead to a formal series:

$$\Sigma\, \tau(\omega).\omega$$

Were ω is a string in the set K^* and $\tau(\omega)$ an absolute date related to the sequence of events modelled by ω. $\tau(\omega)$ may have several meanings:

- First date at which ω may be performed
- Date at which ω may be performed in the worst case.
- Average time needed to perform ω.

et caetera..

We shall not discuss at this point what the meaning of $\tau(\omega)$ should be, because no answer can be given to this question before the dynamics of the system have been introduced. Nevertheless, if the formal series contains the strings:

$5.\gamma.\alpha.\beta$ and $6.\gamma.\alpha.\beta.\alpha.\beta$

It should be clear that both events α and β have their second occurence between date t=5 and date t=6, thus if the time scale consists of integers, these events necesseraly have taken place simultaneously. But, since α and β have an equivalent status as events, for reasons of symetry, the series should also contain the term $6.\gamma.\alpha.\beta.\beta.\alpha$.

Thus it appears that if several events happen at the same date, the variables which stand for these events should commute in the term representing these simultaneous events.

Since information about temporal precedence is already carried in the coefficient, it apperas that there is no need to manipulate non-commutative variables !

In the case of two possible kind of events α and β, we need only to introduce a series:

$$\Sigma_{n,m}\, \tau(n,m)\alpha^n.\beta^m$$

Where $\tau(n,m)$ is some information about the date of completion of n events of kind α and m events of kind β (again, we shall not discuss exactly which information this should be).

Again, we can see that is sufficient to consider only the two series:

$$\Sigma_n\, \tau(n)\alpha^n \text{ and } \Sigma_m\, \tau(m)\beta^m$$

Associated with event α and to vent β. They carry all the necessary information, and are exactly equivalent to the dating sequence associated with α and β.

As a conclusion, the use of formal languages and their related automata is possible to describe the behaviour of a discrete-event system with several different kind of events, but it has two drawbacks:

No explicit mention of real-time in the classical theory of Automata.
Exponential increase of the number of events to be modelled.

An attempt has been made in a previous IIASA- working-paper to compute languages with temporal coefficients.

The best way to model the behaviour of such a system is the use as set of event-sequences to describe its behaviour. These can be represented by their counting functions or the dating functions.

If COHEN et al. succeed in constructing a 2-D calculus, this might be the best representation of the event-sequences in a system.

4- Conclusion: First definition of a discrete-events system.

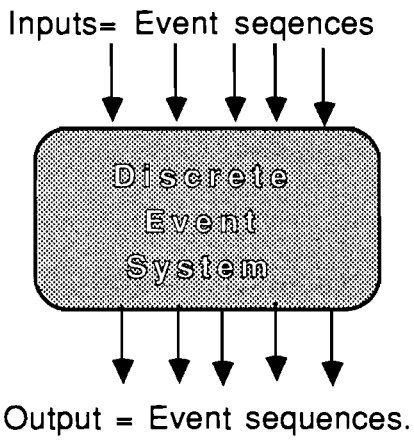
We are now going to make a first attempt to formalize the notion of discrete-event system

The very idea of "systems analysis" is to study links between inputs and outputs. From the previous study, it is straightforward to consider a discrete-event system as a transformation between event-sequences.

3 Systems, control and causality.

1- Discussion on the nature of inputs and outputs.

The conclusion of the second part has led us to describe a discrete-events system as an operator between event sequences.



Undeterminism and uncertainty.

As we have mentionned previously, it is useful to distinguish two notions of undeterminism which we have called (strong) underterminism and (probabilistic) uncertainty.

Recall that:

Strong-underminism means that several possible behaviours are possible when the system is in a given state; the system contains no information about the choice which will be made among the set of next possible states.

We have already mentionned that the first method to get rid of "strong undeterminism" is to suppose that a "random decision maker" generates a choice among the possible next states of the system. This reduces the undeterminism to "probabilistic uncertainty". If a probabilistic distribution is given on the choices of next events, it induces a probabilistic distribution on the dates of the possible next events. Thus, stochastic techniques can be used.

Another approach would be to enumerate all possible behaviours. This is done in automata theory. Unfortunatly, the number of possible behaviours may increase exponentially when several subsystems are connected; furthermore, this approach cancels the fact that the decision has to be made somewhere;Nnevertheless, it is a usuful approach to the problem of validation of the logic behaviour of a system, but it is nearly worthless in the study of its performance.

A variation of the previous approach is to view the system as a "relation" between inputs and outputs. If *In* is the set of all possible inputs and *Out* the set of all possible outputs, then the system can beviewed as a function

Ψ: *In* X *Out* -> {0,1}

such that

$\Psi(u,v) = 1$ if input u and output v are possible together.

$\Psi(u,v) = 0$ if input u and output v are impossible together.

This is a very convenient algebraic trick to get rid of the problem of undeterminism, but it does not give much useful tools, basically because the complexity of this kind of relations is much higher than the complexity of the inputs or outputs. There exists a theory of relations between inputs and outputs, when they are languages generated by automata (see [ALAT2] and [ALAT4]).

If we want to have a "dynamic system", the sets *In* and *Out* should not have an equivalent status in the above definition; thus we have to add the following constraint on the relation Ψ, namely that for every input u, there is at least one output v such that:

$\Psi(u,v) = 1$.

Even if there is no "physical output" produced by the system, this creates a phenomenon which can be represented by the void output-sequence, whose dating sequence is:

$$D(n) = +\infty \quad \forall\ n$$

The relation Ψ can also be viewed as a multi-valued function from *In* into *Out*.

It is a single valued function if and only if the system is deterministic and the equation $\Psi(u,v) = 1$ has a unique solution v for every u.

2- Controlled discrete-event system.

We shall take another approach, much more in the spirit of system theory, namely to consider the decisions as a particular set of inputs. In the language of control theory, it means that the decisions are made "open loop".

This leads us to the following distinctions:

Among the inputs, there are two classes of inputs which play a different role:

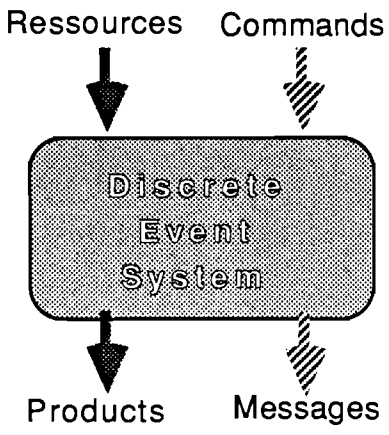
-Arrival of ressources needed by the system; these can be
Datas for a computing system.
Parts or raw materials in a manufacturing system.
Messages in a communicating network.
etc..

- Commands acting on the way the system manipulates these ressources:
Allocation of routines to a processor in a computing system.
Allocation of a task to a machine in a manufacturing system.
Choice of a path in a communicating network.
Activation or inhibition of a subsystem.
etc..

Among the outputs, we may need to distinguish

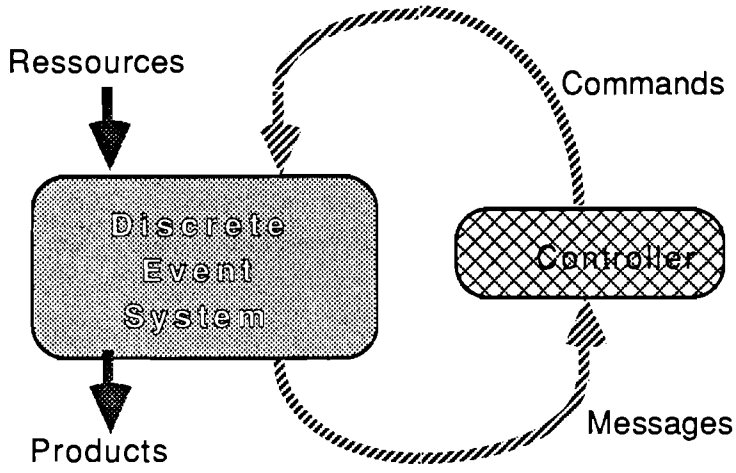
-Outputs of ressources, or products of the system.

-Outputs of messages.



What is the use of these messages ?

They give information to the "outside world" on the state of the system, and these messages may be used to generate the commands. Thus we could define a controled discrete-event-system as a structure of the following kind:



Among the messages, there should be requests for commands. This means that the system sends messages to the controller to ask for a command which will solve a conflict between various possible next behaviours.

This makes sense only under certain constraints; for instance, it seems natural to suppose that a command is sent only after the request message has been received. This constraint can be understood as a "causality constraint" on the controller.

This problem of causality will be studied later and will raise other important questions. First, we shall show that the controller is also a discrete-event systems. For this, we need only to show that the messages and commands (particular kind of messages) can be interpreted as event sequences.

Suppose that K is the set of all possible messages. We clearly have a description of the flow of messages by considering, for every k in K, the sequences of events defined as "emission of a message of kind k".

As a conclusion, the systems and controllers can both be viewed as discrete-event-systems, with all inputs and outputs being event-sequences.

Such an approach has been taken by P.J. RAMADGE and W.M. WONHAM in [CADE3] to [CADE6], but they used the formalism of automta theory to describe the behaviour of the sytem they wanted to control. The controls have an influence on the system by blocking or allowing a subset of events, thus restricting the set of possible behaviours. These controls are generated from a controller which gets inputs from the system, which can be viewed as messages in our model.

3- Causality in a discrete event system.

Let us firts recall the formal definition of causality introduced by Nerode. We shall enounce it in the case of discrete inputs, without considering what kind of system it is applied to.

Let S be a system whose inputs are sequences U(n) and whose outputs are sequences V(n). The system is causal if and only if the following condition is true:

Let U and U' be two input seqences which are equal upt to an indice n,:

$\forall\, p \leq n,\ U(p) = U'(p)$

Then, if V and V' are the related output sequences,

$\forall\, p \leq n\ V(p) = V'(p)$

Obviously the condition means that the outputs only depend on the past inputs. But the notion of past can be associated with real-time or with logic precedence in the inputs. Thus, in the case of discrete -event systems, two questions arise:

1)Which measurement of precedence should be taken into account to define causality ? Should it be related to the absolute time or should it be the logic ordering of events ?

2) How to define causality for "strongly non-deterministic" systems ?

Let us first consider the first question; recall that we have introduced two kind of representations of the event-sequences: counting sequences and dating sequence.

If we use the counting sequences to describe the inputs and outputs, we have inputs which are ordered according to the absolute time. Applying the definition of Nerode to these sequences gives us a notion of causality, which we shall call absolute-time-causality.

If we use the dating seqeunces to describe the inputs and outputs, we get another definition, which we shall call logic-numbering-causality.

These two notions are not equivalent, as it will appear from the very simple following examples:

Both examples have only one input U and one output V and the time scale T is the set of integers.

Let the relation between the input counting sequence CU and the output counting sequence CV is defined as:

$$\forall t \in T \qquad CV(t) = CU(t-3) + 1$$

This relashionship is obviously causal (and even "strictly causal") when expressed through the counting sequences. If we translate it into the relation between dating sequence, we get:

$$\forall n \in Z \qquad DV(n) = DU(n+1) + 3$$

Which is not causal in the sense of Nerode. This situation can occur when modelling the following phenomenon:

Buffer with no Initial value

Datas are sent from the input to the output through a buffer which contains no initial value, and which is keeping always one value when the system is operating. The duration of the process of sending out the value in the buffer and getting the new value has a duration of 3 time units.

We can also give an example of the converse situation: Suppose the relationship between inputs U and outputs V is given through their dating sequences:

$$\forall n \qquad DV(n) = DU(n) - 1$$

This relation is causal in the sense of Nerode, (though it makes no physical sense to model outputs which take place before the inputs they are related to). Obviously, the relation between the counting sequences is not causal:

$$\forall t \qquad CV(t) = CU(t+1)$$

Thus, the real-time-causality and the logic-time-causality are not equivalent; which one should be requested?

The real-time causality should obviously be requested, since it seems obvious that no phenomenon may have an influence on the past. On the other hand, the use of event-driven models is fundamental in the idea of discrete-event systems, and they lead to a representation of event-sequences by the dating sequences. It would be disappointing to give up the notion of causality in this case.

If we look closer at our definition of counting sequences, we see that C(t) is not the amount of events which have happended at time t, but their largest number, relatively to a numerotation of these events.

The numerotation of a sequence of events is completly defined by the logic ordering of the events and by an "initial value"; an arbitrary number must be given to one of the events, in order to define completly the numerotation. Obviously, changing this "initial value" from p to q increases all numeros by q-p. The definition of causality should be independent of this arbitrary choice.

The renumerotation has no effect on the definition of real-time-causality but it may allow us to introduce a less constraigning definition of logic-number-causality.

Recall that we have also to deal with strong undeterminism. To give a general definition of the two possible causalities, we shall consider a discrete-event system as a relation between inputs and outputs.

4- A formal definition of causality.

Let U be a set of vectors of p input event sequences.
Let V be a set of vectors of q output event sequences.
Let Ψ be a relation:

Ψ: U X V -> {0,1}

Such that the equation

$\Psi(u_1,.... u_p, v_1,..... v_q) = 1$

has always at least one solution $(v_1,..... v_q)$ for every input $(u_1,.... u_p)$.

1) A system is "real-time-causal" if and only if the following condition is satisfied:

Here every event sequence y is represented by its counting sequences Cy, and the system is given as a relation between p inputs and q outputs.

Let $(u_1,.... u_p)$ and $(u'_1,.... u'_p)$ be two inputs such that:

$Cu_i(t) = Cu'_i(t)$ for every i and everu date $t \leq t^\circ$

Then there exists two outputs $(v_1,..... v_q)$ and $(v'_1,..... v'_q)$ such that:

$Cv_j(t) = Cv'_j(t)$ for every j and everu date $t \leq t^\circ$

and

$\Psi(u_1,.... u_p, v_1,..... v_q) = 1$ and $\Psi(u'_1,.... u'_p, v'_1,..... v'_q) = 1$

2) A system is "logic-order-causal" if and only if the following constraint is satisfied:

Here, every event-sequence y is represented by its dating sequence Dy. The system is guven as a relation between p inputs and q outputs.

There exist p+q strictly increasing functions $\rho_1.....\rho_p$ and $\tau_1...\tau_q$ mapping Z into Z such that, if we define:

$(u'_i)(n) = (u_i)(\rho_i(n)) \quad \forall\ i= 1..p\ ,\ \forall\ n \in Z$
$(v'_j)(n) = (v_j)(\tau_j(n)) \quad \forall\ j= 1..q\ ,\ \forall\ n \in Z$

The new relation denoted ϑ between the inputs u and the outputs v given by

$$\vartheta(u_1,\ldots u_p, v_1,\ldots v_q) = \Psi(u'_1,\ldots u'_p, v'_1,\ldots v'_q)$$

satisfies:

If t $(u_1,\ldots u_p)$ and $(u'_1,\ldots u'_p)$ are two inputs such that:

$$Du_i\ (n) = Du'_i\ (n) \quad \text{for every i and everu integer } n \leq n°$$

Then there exists two outputs $(v_1,\ldots v_q)$ and $(v'_1,\ldots v'_q)$ such that:

$$Dv_j\ (n) = Dv'_j\ (n) \quad \text{for every j and everu integer } n \leq n°$$

and

$$\vartheta(u_1,\ldots u_p, v_1,\ldots v_q) = 1 \quad \text{and} \quad \vartheta(u'_1,\ldots u'_p, v'_1,\ldots v'_q) = 1$$

5 Comments.

We have adressed the problem of undeterminism by considering that if to input sequences are identical up to an date t or a logical number n, they should be related to possible outputs which are identical up to date t or logical number n. These logical numbers may be redefined by the following aretefact:

The strictly increasing functions $\rho_1\ldots\rho_p$ and $\tau_1\ldots\tau_q$ are renumerotations of the event-sequences. In the second definition, causality is defined modulo this renumerotation. We have not allowed any changing in the absolute time scale in the definition of real-time causality, since a change of orgin in the absolute time would affect the whole system equally.

According to this definition, the first example (the buffer) is both real-time causal and logic-order causal. To show this, one only needs to increase all the numbers of the outputs by one unit.

Both definitions are not equivalent, since the second example is still not real-time causal. Futhermore, it should not be real-time causal by any definition, because the output depends on the real-time future of the output.

This last statement seems obvious, but in part 5 we are going to show an example where the system could be optimised, only if some present knowledge about the future inputs is taken into account. This will lead us to reconsider the notion of causality, and to reconsider the very definition of a discrete-event-system.

6 Conclusion on the need of a notion of causality.

The notion of causality is crucial if one wants to have a dynamic approach to discrete-event-systems: a system can be conisdered as dynamic if and only if its evolution dfrom any state is determined by its past.

As we have seen, the definition of "past "in our approach is not obvious, since several time-scales have to be taken into account. The most natural approach "in the spirit" of event-driven models would be to define the "past" according to the logic ordering, but this leads to the most of mathematical troubles.

Furthemore, it will appear in the foloowing chapter, when we shall study graphic representations, that some systems lead naturally to static descriptions, which can hardly be translated into dynamic ones. This situation is similiar to the one encountered in the study of electrical networks, which are rules by the Kirchof's laws which can hardly be translated into dynamic equations. As mentionned by M.FLIESS in [CAUS1], there is also a basic problem of causality to solve.

In [CAUS1], the autor takes the approach to define inputs in an abstract way: they are the algebraic generators of the differential field of all possible behaviours of the system; thus causality is defined from the point of view of calculus: the causes are those terms which allow to compute the others.

Propbably, this approach must be taken in the case of discrete-event systems; it first needed to investigate which algebra has to be used for calculus. To investigate this question, we are going to have a look at the equations which arise when one studies the basic graphic representations of discrete-event-systems.

Part 4
Graphic description of systems and related operators on event-sequences.

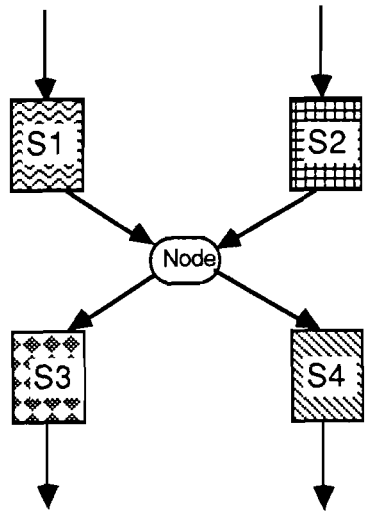
1- Various nodes in a graph.

Behind most of the simulation models, stochastic models and generally any analitcal models of discrete-event-systems, there is a graphic representation of the system.

A wide family of graphs may be used:

- Flowsharts
- Pert graphs,
- State diagrams,
- Transition diagrams,
- Petri nets....

In order to compare these graphic tools, it appears necessary to first investigatethe meaning of a node in such a graph:



Clearly, the rectangles stand for machines, buffers, processors or any more complex subsystem. The important question is: What is going on at the node ?

The answer to this question is not the same in all garphic representations. There appear to be two possible meanings:

The logical "AND" or the logical exclusive "OR".

In the previous example, the following meanings are possible:

S1 and S2 must both have produced an item to activate the node.(AND)
The node can get items from S1 or S2, or both. (OR)
The node sends items to both S3 and S4. (AND)
The nodes sends an item to only one of the susystems S3 and S4. (OR)

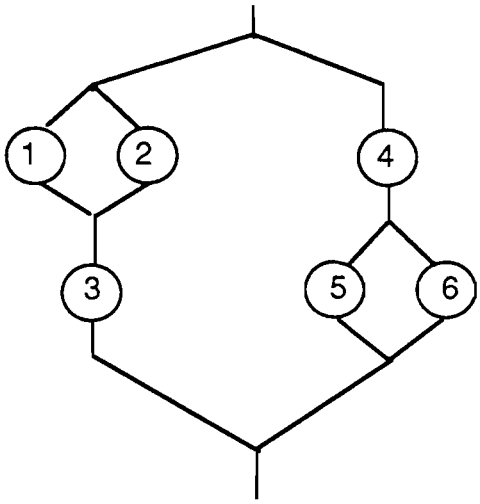
Most of graphic models take into account only one of these logical operators:

Pert graphs, flowcharts deal only with "AND" nodes. Therfore they are tools to model synchronization problems and precedence problems.

State diagrams, transition diagrams.. deal only with "OR" nodes, thus are tools to enumerate the possible behaviours of the system.

The notion of "strong-undeterminism" appears here: the state-diagrams are essentially used to model this strong-undeterminism, whereas a graphic model containing only "AND" nodes does not contain any strong undeterminism.

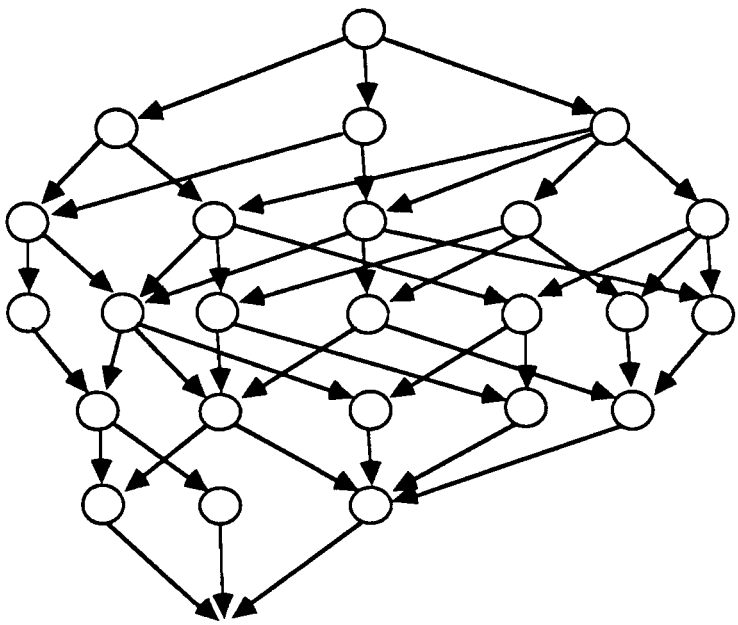
We have seen in part 2 that a system can be studied from both point of views. Recall that our system was gieven by its flowshart:



In this diagram, the nodes all stand for "AND" operations:

Initially, tasks 1 and task 2 and task 4 are started.
Task 3 starts when task 1 and task 2 are finished.
When task 4 is finished, task 5 and task 6 start.
The operation is finished when all of tasks 3, 5 and 6 are accomplished.

We have shown that the enumeration of all possible working configurations of the system leads to a state-diagram:



We have already discussed the increase of the number of states compared with the number of tasks modelled.

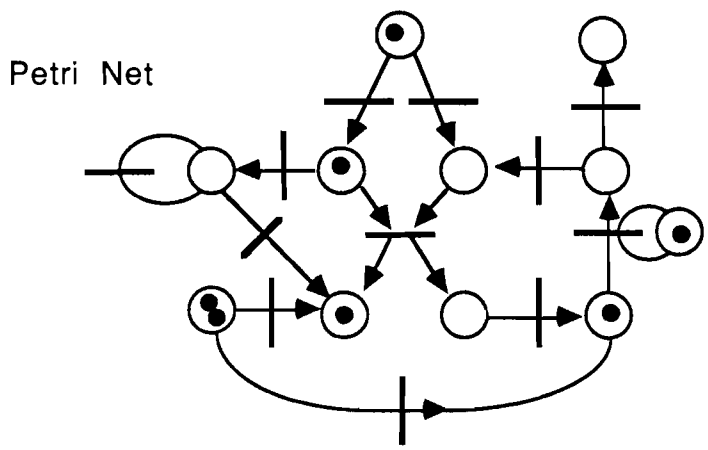
Tthis example to shows that all the sysnchronizations involved in the original system can be hidden in order to get this state diagram, where all nodes stand for the logical "OR".

2- Petri nets.

The use of Petri nets is becomming common in the modelling of discrete-event systems. The success of these models can be partly explained by the fact that they include both "AND" and "OR" nodes.

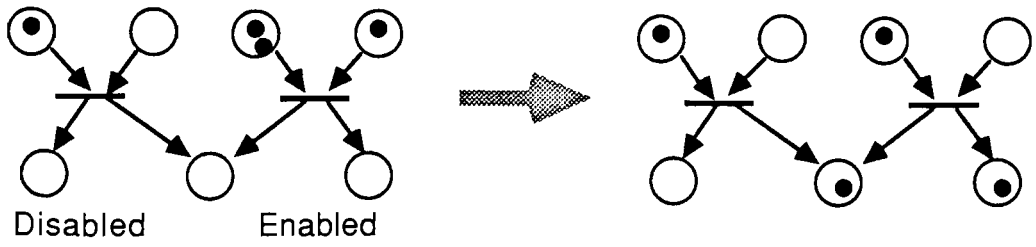
The rules given in Petri nets can handle both logical operators, which enables a description of the systems in which the distinction between synchronization (AND) and strong-undeterminism (OR) clearly appears.

Let us briefly describe Petri nets. They are directed digraphs with two kind of knodes: transitions which are represented by bars and places which are represented by circles.



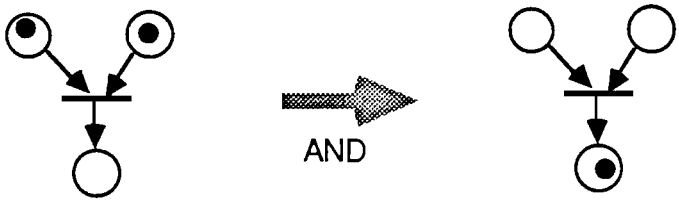
Tokens are circulating in this graph according to the following rule:

A transition can be activateded if and only if, all its input places contain at least one token (or a required number of tokens). Then, these tokens can be erased and will produce one (ore a specified number) token in all the output places of the transition.

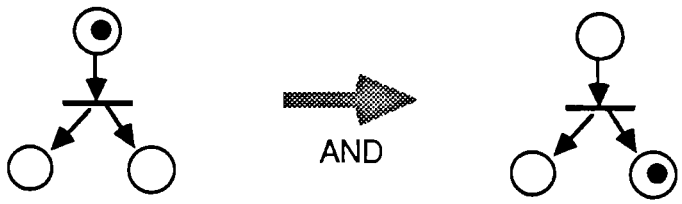


It is important to understand that a system is modelled by both the graph and the initial distribution of tokens. Different distributions of tokens in the same graph can represent quite different systems.

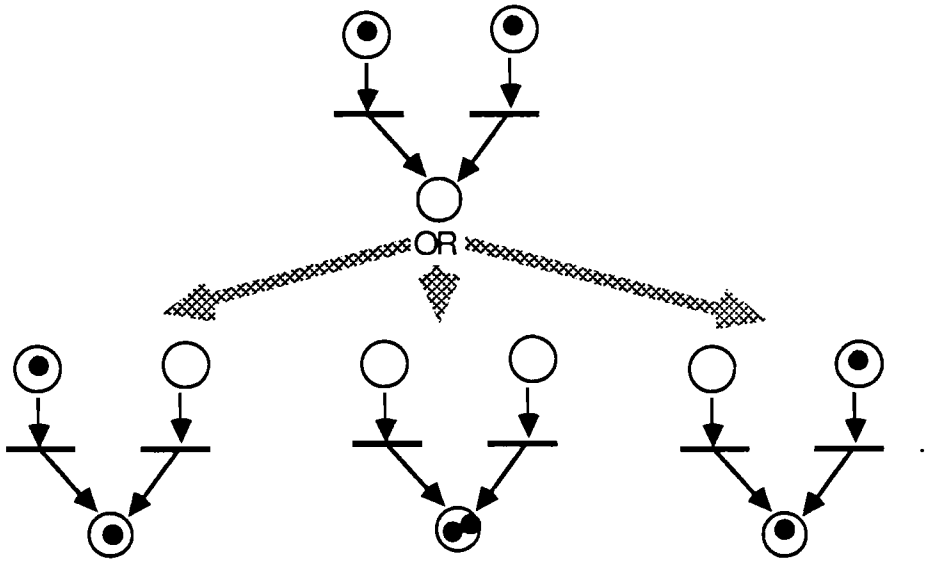
The basic logical operators are modelled by the following nodes:



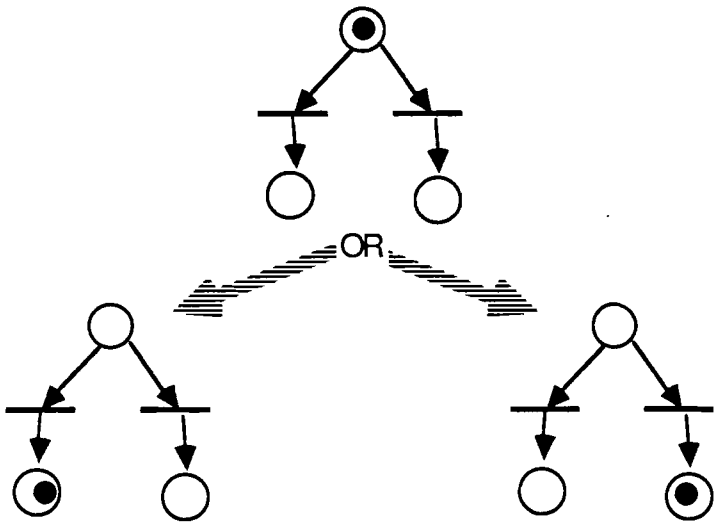
The previous node will be called the "backward AND"; there is also a "forward AND":



The "OR" operator are modelled in the following way:



This situation also models a "backward or", but there is also a converse "forward or":

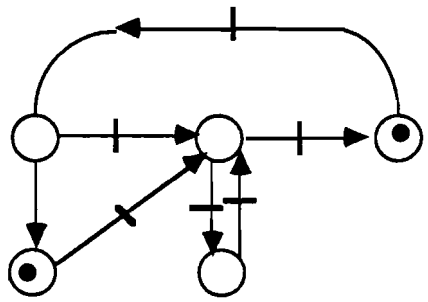


If we suppose that the tokens move automatically, only the last situation (forward OR) contains the strong-undeterminism as we have defined it. There is no informationin the graph which decides where the token is going to move.

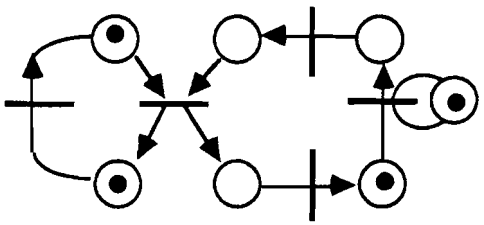
We have seen that most of graphic models belong to two subclasses: those containing only "OR" nodes and those containing only "AND" nodes. This distinction, in the case of Petri nets, leads to the introduction of the following two sub-classes: event-graphs and state-machines; in both these classes, each transition consumes only one token in each of its input places, and produces only one token in each of its output places.

Event graphs are Petri nets such that:
There is 1 or 0 transition before any place and after any place.

State machines are Petri nets such that:
There is 1 or 0 place before any transition and after any transition.



State Machine



Event Graph

State-machines are generalizations of finite automata, whereas timed event-graphs can be viewed as generalization of Pert-graphs.

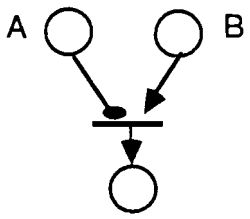
Petri Nets have been introduced mainly as logical validdation tools; most of the results available are concerned with testing of statements like the following examples:

- An infinite behaviour of the system is possible.
- A given transition can be activated an infinite number of times.
- Two given places cannot contain tokens simultaneously.
- The number of tokens in a given place is bounded
- There is no dead-end position of the tokens.

et caetera..

These questions have generally obvious answers in the case of event-graphs or of state-machines, but not in the general case.

The power of Petri nets can be increased by adding the following constraint, which is called a "blocking arc":



In this situation, the transition cannot be activated if a token is in A, whatever happens in B. A is called a "blocking place".

With this new constraint, the power of modelling of these Petri nets can be proved to be equivalent to the power of TURING machines. It can also be shown that if the possible number of tokens in a blocking place is not bounded, the system cannot be reduced to a Petri net whithout blocking place.

The reader interested in Petri nets as modelling tools may consult the book by J. PETERSON [PETRI1]. A lot of examples are given in this Book, which show that Petri nets are tools which are very powerful and can model a very large variety of systems. It also appears that there is are two major drawbacks:

- The size of the networks are generally very large, even for simple problems, making it nearly impossible to apply this tool to real-world systems.

- The modelling is generally not local: a modification of only one small part of a system may result in a reconfiguration of the whole system.

From these two remarks, it appears that Petri nets are essentially usuful to study small examples, in order to analyse the different problems which may arise in a system.

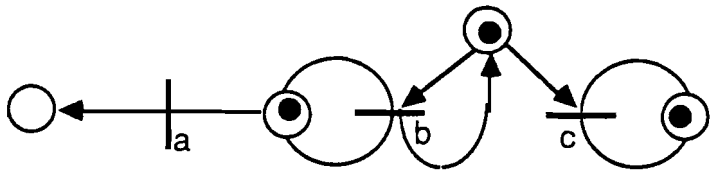
3 Real time and Petri nets.

The theory of Petri nets was first developped for the study of pure-logic problems. No notion of real-time was introduced. There was only a notion of precedence, which is worth discussing because it rises an important question about the use of these networks to study parallel systems.

In all the studies related to the logic behaviour of a Petri net, the assumption is made that an "invisible hand" (similar to the one that is supposed to operate on the market) moves the tokens, by moving them one by one, or one at a time. This appears obviously in the definition of the language associated with the network:

Each transition is labelled by a letter in a formal alphabet. The strings in the language are obviously associated with the sequences of firings of transitions.

Let us take an example:



The set of possible behaviours of this system is defined by the language of all strings which can be written:

$b^n a^r c^m a^s$
where n,r,m,s are positive integers and r+s = 1.

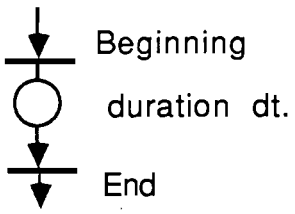
As we have seen in part 2, the precedence and the length of the string induce a notion of real-time, but it is impossible to model the fact that a and c can be performed simultaneously. to solve this, one could enlarge the alphabet by adding symbols for all sets of simultaneously possible transition-crossings, or introducing real-time constraints in the model.

Temporal measurements have initially been intyroduced by RAMCHANDANI in [PETRI2]. An extensive study of timed-Petri-nets has been done by CHRETIENNE in [PETRI3].

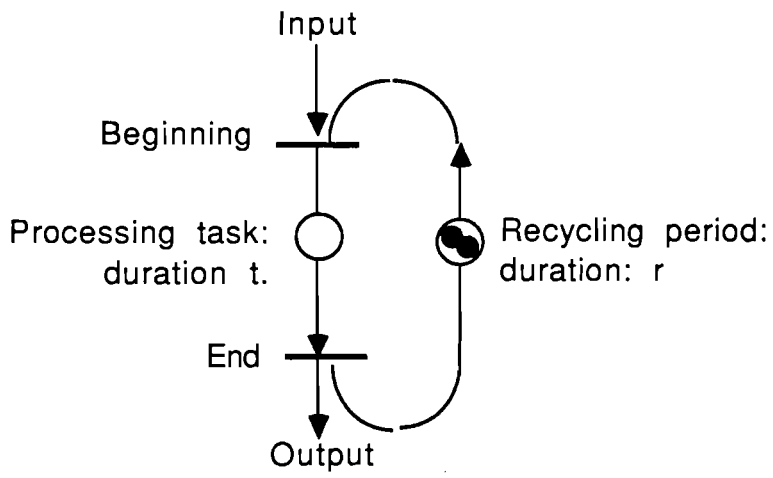
The "Real time" constraints can be introduced in these models in the following way:

- The crossing of a transition is instantaneous.
- With each place is associated a minimal stay of the tokens.

Obviously, it is not a limitation to suppose that transitions are instantaneous, since a task with a duration can be modelled by the structure:

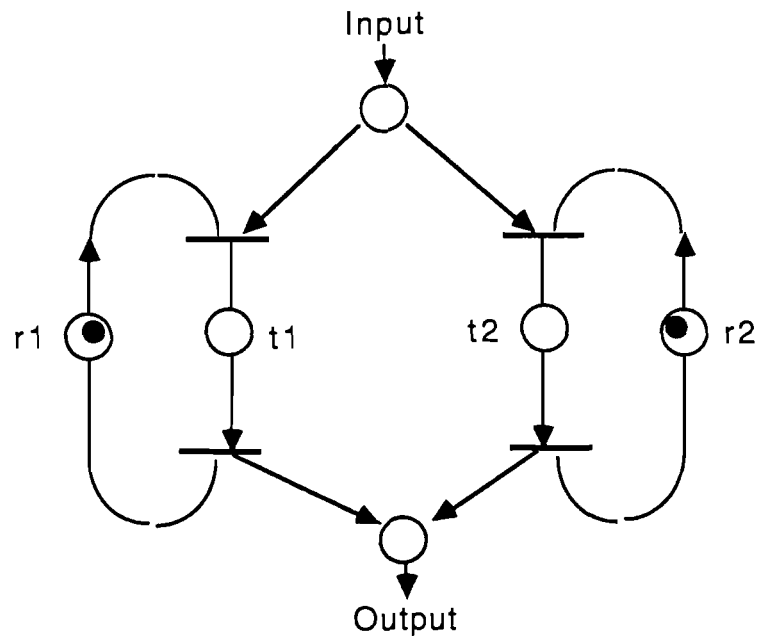


Let us look at an example. Suppose that items are being operated by two identical machines, which have a recycling time of r time-units and perform their tasks in t time units:



From this example, it appears that tokens may be considered as inputs and outputs to this system.

It also appears that some kind of strong-undeterminism can be handled without the explicit use of "OR" node. This is possible here because, since both machines are equivalent, the choice which is made among them has no influence on the further behaviour of the sysem. If the machines needed to be distinguished, the model would have been:



Obviously, if r1=r2 and t1=t2, both paths between input and output are equivalent, thus it is possible to contract this graph into the previous one.

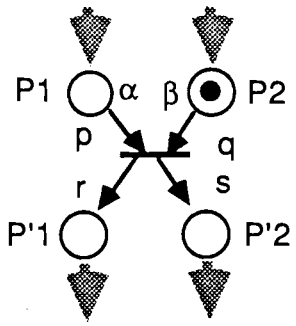
According to our definition of events, it is straightforward to study the following events:

- the crossings of transitions.
- the arrivals of tokens in places.
- the departure of tokens out of places.
- the arrival of tokens from inputs.
- the output of tokens.
- the decision for solving strong-undeterministic situations.

We can now study how the various event-sequences are related by the nodes which can be found in a Petri net.

4 The AND node.

Let consider te following example:



The firing of the transition requires
p tokens in place P1
q tokens in place P2
and produces
r tokens in place P'1
q tokens in place P'2

Arriving tokens have to stay a minimal period of
α in P1 before being available
β in P2 before being available.

There is already a token in place P2. This situation is strongly deterministic if we suppose that the transition is fired as soon as possible.

As we have mentionned, the numerotation of the events is only defined when an "inital value" is produced. First, we need to introduce the event-sequences describing this small system:

DT is the dating sequence associated with the crossings of the transition.
D1 will be the dating sequence associated with the arriving of tokens in P1
D2 will be the dating sequence associated with the arriving of tokens in P2
D'1 will be the dating sequence associated with the arriving of tokens in P'1
D'2 will be the dating sequence associated with the arriving of tokens in P'2

The origin of numbering will be set in the following way: the crossing of the transition wearing number 0 will produce:
Tokens number 0 to r-1 in P'1
Tokens number 0 to s-1 in P'2
And be enabled by the arrival of
Tokens number 1-p to 0 in P1
Tokens number 2-q to 0 in P2
This means that the already available token in P2 wears number 1-q.

We now can write the relations between all these dating sequences. The initial conditions on the numbering imply that:

$$DT(0) = D'1(0) = D'1(1) = \ldots\ldots\ldots = D'1(r-1)$$
$$= D'2(0) = D'2(1) = \ldots\ldots\ldots = D'2(s-1)$$
$$= \text{Max}\,(D1(0) + \alpha\ ,\ D2(0) + \beta)$$

And this can be extended to the n-th transition by the following equations:

$$DT(n) = D'1(n.r) = D'1(n.r+1) = \ldots\ldots\ldots = D'1((n+1).r-1)$$
$$= D'2(n.s) = D'2(n.s+1) = \ldots\ldots\ldots = D'2((n+1).s-1)$$
$$= \text{Max}\,(D1(n.p) + \alpha\ ,\ D2(n.q) + \beta)$$

Thus it appears that these event-sequences are linked by dynamic relations. It is worth noticing that the presence of a token in P2 has been cancelled by the choosen numerotation.

A causality problem arises in this event-driven approach:
if r > p, then, according to this numerotations, the values of the output dating sequence D'1 depends on the "future" of the values of the input dating sequence D1 !

This cuasality problem can be avoided by supposing that p=q=r=s=1; this is the case in event-graphs. A complete theory of the dynamic system describing a timed-event-graph has been developped by COHEN et al.. in references [EVSE2] [EVSE3] and [EVSE4].

Their model has not been extended to more general timed Petri nets yet, mainly because of the problem of causality which appears as soon as the condition p=q=r=s=1 does not hold in a transition. Therefore we shall study this causality problem in detail in chapter 4.

Equations describing this node could also be written for the counting sequences. With each dating sequence DX we associate a counting sequence CX.

We have to specify an absolute date for one of the events, in order to set an "initialcondition". For instance, we can assume that the firing of the transition wearing number 0 takes place at absolute date 0. Then:

$CT(0) = 0 = C1(-\alpha) = C2(-\beta)$
$C'1(0) = r-1$
$C'2(0) = s-1$

And at any date t, the following relations hold:

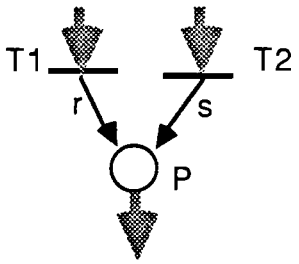
$CT(t) = \min (Int (C1(t-\alpha)/p) , Int (C1(t-\beta)/q))$
$C'1(t) = r.(CT(t) +1) -1$
$C'2(t) = s.(CT(t) +1) -1$

Where *INT* (x) is the (minorating) integer part of x.

Obviously, there is no problem of causality in these dynamic relations, but they use quite weired operations. These can be notably simplified if p=q=r=s=1. COHEN et al. have shown that in the cae of timed-event graphs, both representations are strictly equivalent.

5 The OR node.

To study the or operator, it is important to separate the "backward Or", which induces no strong-undeterminism from the "forward or".



The place P receives r tokens each time T1 is activated, and receives s tokens each time T2 is activated.

The equations describing this "bacwards OR" node would be obvious if the counting sequences would effectively model the number of arriving tokens in the plave P. We would have:

$CP(t) = r.C1(t) + s.C2(t)$

Unfortunatly, this does not hold if these counting sequences refer to an arbitrary numerotation. To get this relation, we have to fix some iniatal values of the counting sequences and give an absolute date for some reference events. The previous relation can be obtained with the following conditions:

The time scale is redefined by supposing that the arrival of token number 0 in the place P takes place at absolute date 0, and that no token with number >0 has arriven at this date. Then, we have to define the numerotations relative to the activations of transitions:

The last activation of T1 before date 0 wears number 0.
The last activation of T2 before time 0 wears number 0.

No dynamic structure appears here, since all movements of tokens through the transitions are supposed to be instantaneous. Dynamic relations appear if we look at possible outputs of the place P, described by their counting sequence C'P. If the requested stay in P lasts α time-units:

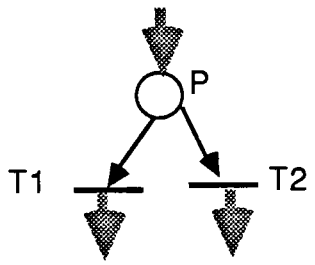
$$C'P(t) = CP(t- \alpha)$$

This counting sequence C'P is only a majoration of the actual output sequences out of the place P, since if this node is a part of a larger system, the tokens may be blocked in P by other constraints.

Under the same initial values on the numerotations, the equation describing this node in the event-driven domain, when event-sequences are represented by their dating sequences is:

$$DP(n) = Min_{p+q=n} \max(D1(p) + D2(q))$$

We now are going to look at the strongly-undeterministic "forward OR" node.



Clearly, in this example, there is no single possible behaviour; we can only express a constraint linking the event-sequences related to T1 and T2.

There is no difficulty to compute the maximum number of possible firings of T1 or T2, which are equal, since both are obtained by allocating all the arriving tokens to one transition only. This does not create any new information.

Another approach is to write down a relation linking the event-sequences associated with the transitions and the output of the place P.

With "initial values" similar to the previous case, we obviously get a constraint on the counting sequences which is similar to the equation obtained in the previous case:

$$r.C1(t) + s.C2(t) \leq CP(t- \alpha)$$

Where CP(t) is the counting sequence describing the arrival of tokens in the place P, and α is the minimal stay in this place.

Equality holds if the decision , or control, defining the use of any available token is made not later than α time units after the arrival of a token.

The related inequation on dating sequences is:

$$Min_{p+q = n} \max(D1(p) + D2(q)) \geq DP(n)$$

6 Controled "forward node".

In the previous part of this paper, we have proposed another way of managing the strong-undeterminism, which is more in the spirit og the theory of systems. The solution would be to consider that there are other inputs which decide where the available token is allocated.

This idea seems very simple, but it will appear that it is not so simple to decide how the system reacts to these commands. Let us study this to the previous node, which will be described by the following event-sequences:

-The arrival of tokens in the place P, which will be modelled by its counting and dating sequences CP and DP.

-The requests to send the token to T1, which will be modelled by its counting and dating sequences CU1 and DU1.
-The requests to send the token to T1, which will be modelled by its counting and dating sequences CU1 and DU1.
- The requests to send the token to T2, which will be modelled by its counting and dating sequences CU2 and DU2.
- The activation of transition T1, which will be modelled by its counting and dating sequences CT1 and DT1.
- The activation of transition T2, which will be modelled by its counting and dating sequences CT2 and DT2.

The initial values of these sequences will be set such that:

Token number 0 arrives in P at absolute date 0.
At absolute dtae 0, the last firing of T1 had number 0.
At absolute time 0, the last firing of T2 had number 0.
At absolute time O, the last commands had both number 0.

This is not sufficent to describe the behaviour of the system, because there remains an ambiguity on the reactions of the controlled node to control messages, and there is no rule in the theory of Pertl nets to answer this question. In his Doctoral THesis, P.CHRETIENNE (see ref) has examined the problem of defining a controled timed Petri net. He has shown that several possible responses to this control are possible.

For instance, the controler decides the path taken by a token only after its arrival in the place and generates the command afterwards.

This rule seems very natural; implicitly, the system sends a request for a command to the controller each time when a token arrives. Thus the event-sequence described by CP and DP must also be viewed as an output of the system. Since the command can only be sent after the arrival of the token, the following constraint on the command must hold:

$$CU1(t) + CU2(t) \leq CP(t)$$

If it takes a fixed delay λ to compute this command, we get:

$$CU1(t) + CU2(t) = CP(t-\lambda)$$

If the "reaction-time" of the system is modelled by the time needed to compute the command, it is possible to suppose that transitions react instantaneously to these commands. Then, it is necessary that the command is sent only when the token is available, thus $\lambda \geq \alpha$.

Then, we get a very simple input-output relation:

$$CT1(t) = CU1(t)$$
$$CT2(t) = CU1(t)$$

In this relation, the dependance from the outputs on the arrival of tokens is hidden, because it becomes as a constraint on the controller, and we consider the system "open loop", whithout modelling the controller.

Obviously, this is not a very statisfying result, because the problem of undeterminism is rejected to the controller!

The only way to get a better model is to explicitely describe the algorythm which generates the command. A lot of choices are possible; let us consider an example:

The controller sends alternatively a token to T1 and a token to T2. Arrival token number 1 being sent to T1. This gives the following relations:

$$CT1(t) = CU1(t) = In\,(CP(t-\lambda)/2) + 1$$
$$CT2(t) = CU2(t) = In\,(CP(t-\lambda)/2)$$

Another approach would be to suppose that the commands are sent independently of the arrival of tokens. The question rises again to define exactly the response of the system to such inputs. Several choices are possible:

The last (in the sense of real-time) command received is taken into account, whith priority given to T1 in case of conflict.

The command is chosen according to the choice which has been mostly requested, with a priority in case of ex-aeco.

Other rules are possible. Most of these rules share a common fact: it is very difficult to translate them simply into equations on the event-sequences.

7- Conclusion on the algebra of events.

From the previous examples, it appears that there is no obvious choice for the set of operations which would be natural on the set of event-sequences. COHEN et al. have shown in [EVSE2] that in the case of timed-event-graphs, the dynamics of the system, which is necesseraly strongly deterministic, can be represented by dynamic equations in the (max,+) algebra, but their model can hardly be generalized to all Petri nets.

It also appears that if a control is modelled, to decide the allocation of tokens in a strongly-undeterministic node, quite complicated operations may be necessary.

For this reason, it seems that a generalized description of a controlled timed-Petri-net is only possible at the level of a calculus on operators, as suggested by P. CASPI and N.HALBWACHS in [EVSE7]. A system will be viewed as built up with a set of operators, which can be linked by composition. This has been studied by these authors, and they have introduced a functional calculus to describe these operators.

We shall give an imporatnt result on th ecomplexity of this operator calculus in part 6, but first we are going to have a closer look to a dynamic allocation problem.

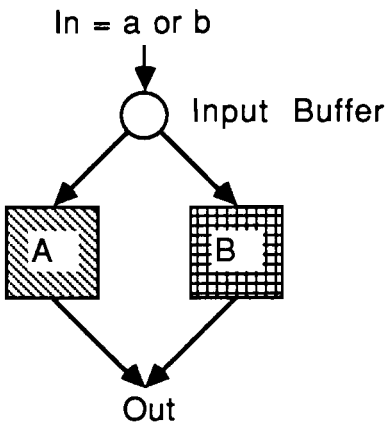
Part 5
A non-causal optimisation problem and its consequences.

1- Dynamic allocation.

We have already shown that the definition of causality in the event-driven representation of a system is not straightforward. When the system is modelled by the counting sequences.

Let us consider the very simple following example:

Two machines A and B are processing inputs which can be of nature a or b.



The processing durations are not equal:

A needs 2 time units to process a
A needs 3 time units to process b
B needs 4 time units to process a
B needs 6 time units to process b
A and B have a reset time of 1 time unit.

This system gets a finite input sequence which is a mixture of a and b. Clearly, we are in a strongly undeterministic situation since the allocation of the inputs to the machines is not determined.

Our aim is to design a controller which will decide where the parts should be allocated, in order to end the processing of the input sequences as soons as possible.

If this input sequence is known in advance, we clearly have a combinatorial problem, but this is not a "dynamic" problem, and certainly not a "causal" problem. Let us compare some simple decision algorythms to do this allocation, when only the available inputs waiting in the buffer are known to the controller.

A first decision scheme would be to apply the following rules:

If A and B are idle and one inout only is available, send it to A.
If A and B are idle and several inputs are available, start A and B
with the rule: if a and b are available, a goes to B and b goes to A.
If one machine is idle and an input is available, start processing it
with the rule: if a and b are available, A idle takes b, Bidle takes a.

This scheme is obviously derives from the following rules:

If no knowledge of the future is available, there is no reason to wait.
A is always quicker so it should have priority.
If possible, b parts should be processed by the quickest machine: A.

Let us test this scheme by doing a small simulation of the reaction of tis system to a sample input sequence:

We us e the following notation:
Sx means starts to process x.
Px means processes x
Ex means ends processing x.
I means idle.

Date	Input	Buffer	A	B	Output	Product
0	a	a	Sa	I	-	-
1	b	b	Pa	Sb	-	-
2	-	-	Ea	Pb	a	a
3	a	a	Sa	Pb	-	a
4	a	a	Pa	Pb	-	a
5	-	a	Ea	Pb	a	aa
6	-	a	Sa	Pb	-	aa
7	b	b	Pa	Eb	b	aab
8	-	b	Ea	Sb	a	aaba
9	a	a	Sa	Pb	-	aaba
10	-	-	Pa	Pb	-	aaba
11	-	-	Ea	Pb	a	aabaa
12	-	-	I	Pb	-	aabaa
13	a	a	Sa	Pb	-	aabaa
14	b	b	Pa	Eb	b	aabaab
15	end	b	Ea	Sb	a	aabaaba
16						
....	-	-	I	Pb	-	aabaaba
20						
21	-	-	I	Eb	b	aabaabab
End.						

This simulation clearly shows that the allocation procedure was mislead by the arrival of the first input a, which was allocated to the quickest machine and has induced the slowest machine to always operate the input b, for which it is the least efficient.

From this analysis, it its straightforward to try another allocation scheme, which aim would be to avoid allocating parts b to the slowest machine B.

This was already done in the previous example by the last rules, but the problem arose because the quickest machine was busy when the slowest part b arrived.

Thus, the first rule needs to be changed, in order to have parts a be sent to B, in order to keep the quickest machine A available, in case a part b arrives next.

Let us try the following rules:

If A and B are idle and both inputs are available,
a goes to B and b goes to A.
If A and B are idle and only a is available, send it to B
If A and B are idle and only b is available, send it to A
If one machine is idle and an input is available, start processing it
with the rule: if a and b are available, A idle takes b, B idle takes a.

This leads to the following simulation:

Date	Input	Buffer	A	B	Output	Product
0	a	a	I	Sa	-	-
1	b	b	Sb	Pa	-	-
2	-	-	Pb	Pb	-	-
3	a	a	Pb	Pa	-	-
4	a	aa	Eb	Ea	ab	ab
5	-	aa	Sa	Sa	-	ab
6	-	-	Pa	Pa	-	ab
7	b	b	Ea	Pa	a	aba
8	-	b	Sb	Pa	-	aba
9	a	a	Pb	Ea	a	abaa
10	-	a	Pb	Sa	-	abaa
11	-	-	Eb	Pa	b	abaab
12	-	-	I	Pa	-	abaab
13	a	a	Sa	Pa	-	abaa
14	b	b	Pa	Ea	a	abaaba
15	end	b	Ea	Sb	a	abaabaa
16						
....	-	-	I	Pb	-	abaabaa
20						
21	-	-	I	Eb	b	abaabaab
End.						

The result is not improven, the time needed to process all the inputs is stil 21 time units.

A closer observation shows that in both cases, time was lost by starting to process a part a on machine A at date 13, though a part b was going to arrive at date 14.

It would have been wiser to wait until date 14, to allocate the part b to machine a, because the time won in using the quickest machine is larger than the time lost in delaying the process.

This analysis is possible only whence the whole sequence of inputs is known. In this case, we can improve the whole processing time by applying the following scheme:

Date	Input	Buffer	A	B	Output	Product
0	a	a	I	Sa	-	-
1	b	b	Sb	Pa	-	-
2	-	-	Pb	Pb	-	-
3	a	a	Pb	Pa	-	-
4	a	aa	Eb	Ea	ab	ab
5	-	aa	Sa	Sa	-	ab
6	-	-	Pa	Pa	-	ab
7	b	b	Ea	Pa	a	aba
8	-	b	Sb	Pa	-	aba
9	a	a	Pb	Ea	a	abaa
10	-	a	Pb	Sa	-	abaa
11	-	-	Eb	Pa	b	abaab
12	-	-	I	Pa	-	abaab
13	a	a	I	Pa	-	abaa
14	b	ab	Sb	Ea	a	abaaba
15	end	a	Pb	Sa	-	abaaba
16	-	-	Pb	Pa	-	abaaba
17	-	-	Eb	Pa	b	abaabab
18	-	-	I	Pa	-	abaabab
19	-	-	I	Ea	a	abaabab

End.

2- Discussion on the previous example.

We have shown that the two heuristic allocation schemes, which do not take into account any knowledge on the future inputs do not give an optimal proceeding time.

Such a scheme, wich we have decided to view as a contoler in our system approach, would obviously be Real-Time-Causal (RTC).

Of course, we have not proved that the optimal performance could not be achieved by a RTC-controller. It is certainly possible, for a given input sequence, to construct a RTC controller which will be optimal for this specific input, but certainly not for all inputs.

Obviously, a good allocation scheme would require some knowledge of the future. A knowledge of all the future is certainly not needed:

In the previous example, at any date, it is only necessary to know what inputs will arrive during the next 7 time units, since no task lasts longuer than this duration.

Anyway such a controller would not be causal.

This seems to imply this allocation problem should be studied globally, that is by considering the whole sequence of inputs. This is obviouslty not a "dynamic" approach to the problem.

Nevertheteless, the whole procedure of acquiring information is generally dynamic:

At every instant, some information about the future inputs is known, and this information is updated at the next instant, generally by increasing this information. We are going to use this idea to construct a more general framework to study such control problems for discrete-event-systems (DES).

3- Two-level discrete-event-systems.

Let us define a sequence of event-sequences (SES) as a function mapping the time-scale T into the set of event-sequences.

Clearly, such a SES can be represented as a sequence of dating-sequences, or a sequence of counting sequences. We shall use the following notations:

If U is a SES,

CSU will be its sequence of counting seuquences.
DSU will be its sequence of dating sequences.

The meaning of U can be defined through its representations CSU and DSU:

CSU(t)(t') is the least majorant of the numbers of events which will have happened at date t',
according to the knowledge available at date t.

DSU(t)(n) is the date at which event number n will occur,
according to the knowledge available at date t.

The knowledge of the past, is exact, thus:

if $t > t°$, for any $t' \leq t°$ $\quad CSU(t)(t') = CSU(t°)(t')$

This condition means that the number of events which have occured at any date t' before date t° will remain the same at any date t posterior to date t°.
This condition is not so easy to express on the dating sequences.

A few remarks can be made on this new approach: the real-time and the logic time are not equivalent any more in the representation of the system. Clearly, at the higher level of sequences of event-sequences, it makes only sense to consider the knowledge available as a function of the real time.

This may also be a way of solving the various causality problems which have appeared in the case of logic time, since at this higher level of modelling, only the past in the "real-sense" is taken into account. Furthermore, the previous problem of designing a controller which uses some information on the next arrivals of inputs can obviously be adressed much more efficiently in this framework.

4-Strong undeterminism and the dynamics of knowledge.

The meaning which have been associated with the sequences of inputs and outputs need to be defined more precisely if there is some strong undeterminism in the process which creates them: several dates may be possible for the same outputs. Two of them are of interest, if no probability distribution is given, as we have assumed in the strong-undeterministic case:

The worst or latest possible output.
The best or quickest possible output.

Note that the worst possible case is always obtained when the controller takes no decision and is blocking the whole system. Thus some optimality in the decision algorythm should be assumed: the system is not delayed whthout reason when a decision can be made.

Choosing one of these meanings has some consequences on the dynamics of the SES:

If a SES U models the worst case; let us consider the event-sequence at time t. At time t+1, all past events, which occured not later than date t, will remain unchanged in U(t+1), but for the next events, the prediction can only be "more optimistic", since at time t, the worst case was considered. Thus:

$CSU(t+1)(t') \geq CSU(t)(t')$ for any t'
and equality holds for $t' \leq t$.

Conversly:

$DSU(t+1)(n) \leq DSU(t)(n)$ for any n.

Thus CSU is a non-decreasing sequence of non-decreasing sequences, and DSU is a non-increasing sequence of non-decreasing sequences.

If the SES U models the best possible event-sequence, we have the opposite constraint: the knowledge at time t+1 of the future events can only be more pessimistic, thus:

$CSU(t+1)(t') \leq CSU(t)(t')$ for any t'
and equality holds for $t' \leq t$.

Conversly:

$DSU(t+1)(n) \geq DSU(t)(n)$ for any n.

In this case, CSU is a non-increasing sequence of non-decreasing-sequences and DSU is a non-decreasing sequence of non-decreasing sequences.

TYhe use of these macro objects which are SES thus appears as another way of coping with strong undeterminism. The system is represented by the dynamic evolution of the upper and lower bounds of the dating and counting sequences associated with its inputs and outputs, and thes should have a strongly-deterministic behaviour.

We have shown that these sequences have a structure; the next step is now to define the operations on theses sequences of event-sequences. These should obviously be constructed from the operations on event sequences.

If an addition is defined on the dating sequences, and is denoted $\oplus$, then obviously, the addition on the sequences of dating-sequences should be:

$(DSU \oplus DSV)(t) = DSU(t) \oplus DSV(t)$

Conversly, the same should be required for counting sequences.

This brings us back to the basic dilemma of the choice of a set of operations on the event-sequences, which we have already extensively studied. We still are not ready to give an ultimate answer, but some hints will be given from the results of the following chapter.

It will appear that it is not neccessary to define a product on event-sequences, becasue the wright algebra to be considered is the on of operators, in which their is a natural product: the composition of operators. Only the summ is needed, in order to induce a summ on operators.

Part 6
A representation theorem.

1- Operations on systems and hypothsesis.

In the chapter dealing with graphic representations, we have shown that there is no straightforward choice for the operations on event-sequences, wheteher they are represented by their dating sequences or by their counting sequences. In the chapter 5, we have even shown that it may even be necessary to introduce sequences of event-sequences, which will have other operations.

Independtly of the choice which can be made on the mathematical structure of the inputs and outputs of systems, and of the operations on these inputs and outputs, it is possible to show that the usual operations on systems can be represented by a matrix calculus in an adequate set of operators.

Let us assume that all inputs and outputs are column-matrixes with coefficients in a monoïd D, whose neutral element will be denoted ε; in fact elements in D are sequences of elements in a simpler monoïd.

In the one-dimensional case, a system will be viewed as an operator λ acting on D:

$$v = \lambda.u$$

means that v is the output produced by the input u.

We shall denote H(D) the set of operators which are monoïd-homomorphisms, that is, the set of operators λ which satisfy:

$$\lambda.(a \oplus b) = \lambda.a \oplus \lambda.b$$

Recall that a semi-ring S is a structure $(S,\oplus, .)$ where:

S is a set.
$\oplus$ is an internal operation named "addition".
. is an internal operaion named "product".
$\oplus$ is associative and commutative.
. is associative and left and right distributive over $\oplus$, which means:
$a.(b \oplus c) = a.b \oplus a.c$ for every a, b, c in S
$(a \oplus b).c = a.c \oplus b.c$ for every a, b, c in S.
$\oplus$ has a neutral element which we shall denote ε
. has a right and left-neutral element which we shall denote e.

Some authors use the word "semi-ring" only for the case when the "addition" is cancelletive, which allows the semi-ring to be extended to a ring. Thus the structure we have defined is often reffered as a dioïd, which is a monoïd with a second operation.

The set H(D) has obviously a semi-ring structure, when the following operations are defined:

$(S \oplus S')(x) = S(x) \oplus S'(x)$ for every x in D.
$S.S'(x) = S(S'(x))$

The distributivity holds because we consider only homomorphisms.
The element ε in H(D) is the function identicaly equal to the ε of D.
The element e in H(D) is the identity function.

If D is also a semi-ring, then any element a in D can be identifyed to the homomorphism:

$x \rightarrow a.x$

By analogy with group theory, such homorphisms will be called "internal".

We shall work with a subset of H(D), CH(D) such that:

For every operator α in CH(D) and any element b in D, the implcit equation

$$x = \alpha.x \oplus b$$

has a canonical solution

$$x = \alpha^*.b$$

which can be identified with the series:

$$\alpha^*.b = b \oplus \alpha.b \oplus \alpha^2.b \oplus \alpha^3.b \oplus \alpha^4.b \oplus .....$$

And defines an operator α^* in CH(D).

This hypothesis generally holds for internal operators, when D is semi-ring. A general discussion on this can be found in [Eilenberg].

It always holds for H(D), when D is a complete lattice, the addition being the "mean" of the lattice.

In usual system-theory, the set D is a set of formal series (Z-transforms) and $a^*.b = b/(1-a)$ is always defined for internal operators, which are also series, though $a^*.b$ may be a series with poles.

The use of this star operation will become obvious, when the natural operations on systems are defined. We shall study this only in the one-dimentional case, but all our results will begeneralizable to multi-dimensional inputs and outputs, thanks to one of the theorems we are going to prove.

Which are the natural operations on systems ?

Sequence of operators S and S': This new operator is S.S'.

S and S' beeing in parallel. This defines $S \oplus S'$.

Feedback of operators; this defines $S^*.S$.

2- Rationnal operations and closure.

The summ, product and star operation in a semi-ring are called the rationnal operation.

If A is a given subset in a semi-ring, the smallest subset of the semi-ring which contains A and is stable under the rationnal operations is called the rationnal closure of A, and will be denoted A^*.
A set A is rationnaly closed if and only if $A^* = A$.

It is easy to check that, if A and B are subsets of the semi ring,

$A \supseteq B \Rightarrow A^* \supseteq B^*$

$(A^*)^* = A^*$

In the previous paragraph, the assumptions that we have made on CH(D) iimply that it is rationnaly closed.

All these notions of star operations, closure etc.. have extensively been studied in automata theory, and can be found in [Eilenberg]. Classical automata theory is essentially concerned with the study of formal series, which define the set D, and internal operators, which are products by other formal series. Rationnal series are then defined as those belonging to the closure of the set of polynoms.

An essentilat result in this theory is the KIEENE-SCHÜTZENBERGER theorem, which shows that the coefficients of a rationnal series can be computed by a matrix-calculus.

This result is known, in a simpler form, as the realisation theorem in usual system-theory (see Reutenauer).

Unfortunaltly, this result cannot be applied to more general non-internal operators, as those which may appear in the modelling of discrete-event systems. Neither does this apply if the set of elementary operators contains elements which commute.

We are going to study the following problem:

Let S be a rationnaly closed semi-ring.
Let E be any subset of S, containing ε and e. E^* is the rationnal closure of E.
$M_{p,q}(E)$ is the set of (p,q) matrixes with coefficients in E.
$M_{p,q}(S)$ is obviously a rationnaly closed semi-ring.
$M_{p,q}(E)^*$ is the closure of $M_{p,q}(E)$ in $M_{p,q}(S)$.

We shall prove that

$M_{p,q}(E)^*$ and $M_{p,q}(E^*)$ are equal.

and

All elements in $M_{p,q}(E)^*$ can be represented by matrixes with coefficients in E.

3 The realisation theorem and its proof.

We shall first prove the following technical lemma:

Lemmma:

When the star operation is applied to a square matrix which is split into 4 blocs, we have the following formula:

$$\begin{array}{|c|c|}\hline a & b \\ \hline c & d \\ \hline \end{array}^* = \begin{array}{|c|c|}\hline a^* \oplus a^*.b.(c.a^*.b \oplus d)^*.c.a^* & a^*.b.(c.a^*.b \oplus d)^* \\ \hline (c.a^*.b \oplus d)^*.c.a^* & (c.a^*.b \oplus d)^* \\ \hline \end{array}$$

It is important to note that this result holds whatever the nature of blocs is: they may be scalars or again be matrixes. Only a and d need to be square.

Demonstration:

Let M be the whole matrix. M* is then defined as the canonical solution of the equation:

$$X = M.X \oplus E$$

Where the unknown matrix X is of same size as M, and E is the identity matrix.

$$X = \begin{array}{|c|c|}\hline x & y \\ \hline z & t \\ \hline \end{array} \qquad E = \begin{array}{|c|c|}\hline e & \\ \hline & e \\ \hline \end{array}$$

This equation is equivalent to the system:

$$x = a.x \oplus b.z \oplus e$$
$$z = c.x \oplus d.z$$
$$y = a.y \oplus b.t$$
$$t = c.y \oplus d.t \oplus e$$

By using the star operation to solve implicit equations in the above system, we get:

$$y = a^*.b.t$$
$$t = c.a^*.b.t \oplus d.t \oplus e$$
$$t = (c.a^*.b \oplus d)^* \quad \text{et } y = a^*.b.(c.a^*.b \oplus d)^*$$
$$x = (b.d^*.c \oplus a)^*$$
$$z = d^*.c.(b.d^*.c \oplus a)^*$$

Another approach to the equation would yeld:

$$x = a^*.(b.z \oplus e)$$
$$z = c.\ a^*.(b.z \oplus e) \oplus d.z$$
$$z = (c.a^*.b \oplus d)^*.c.a^* \quad \text{et} \quad x = a^* \oplus a^*.b.\ (c.a^*.b \oplus d)^*.c.a^*$$

Thus, during this demonstration, we have also shown the following formulas:

$$(b.d^*.c \oplus a)^* = a^* \oplus a^*.b.\ (c.a^*.b \oplus d)^*.c.a^*$$
$$(c.a^*.b \oplus d)^*.c.a^* = d^*.c.(b.d^*.c \oplus a)^*$$

Now, let us consider a subset E of the semi-ring D, such that:

$$D \supseteq E^* \supseteq E \supseteq \{\varepsilon,e\}^* = \{\varepsilon,e\}$$

Recall that $M_{n,p}(E)$ is the set of (n,p) matrixes with coefficients in E.

We shall compare the closure $M_{n,n}(E)$ in the semi-ring $M_{n,n}(D)$, which we shall denote $M_{n,n}(E)^*$, with the set $M_{n,n}(E^*)$.

Theorem:

$$M_{n,n}(E^*) = M_{n,n}(E)^*$$

Demonstration:

Firts, we shall prove the following result by induction on n:

$$M_{n,n}(E)^* \subset M_{n,n}(E^*)$$

The iduction hypothesis at order p will be:

For any semi-ring D, every part E of D containing $\{\varepsilon,e\}$ and any integer $n \leq 2^p$

$$M_{n,n}(E)^* \subset M_{n,n}(E^*)$$

According to the lemma:

$$A \in M_{2,2}(E) \Rightarrow A^* \in M_{2,2}(E^*)$$

Since E^* is a sub-semiring of D containing E,

$$A \in M_{2,2}(E)\ ,\ B \in M_{2,2}(E) \Rightarrow A \oplus B \in M_{2,2}(E^*)$$
$$A \in M_{2,2}(E)\ ,\ B \in M_{2,2}(E) \Rightarrow A.B \in M_{2,2}(E^*)$$

Thus we have proved the result for p=1.

If $2^p=n$, $2^{p+1}= 2.n$. Let us assume that the result holds at order p.

$$M_{2.n,2.n}(E) = M_{2,2}(M_{n,n}(E))$$

Thus by using the induction hypothesis at order 1 in $M_{2,2}(D)$ with E beeing replaced by $M_{n,n}(E)$, we get:

$$M_{2.n,2.n}(E)^* = M_{2,2}(M_{n,n}(E))^* \quad C \quad M_{2,2}(M_{n,n}(E)^*)$$

Now by applying the induction hypothsesi at order p, we get the resulta at order p+1:

$$M_{2.n,2.n}(E)^* \quad C \quad M_{2,2}(M_{n,n}(E)^*) \; C \; M_{2,2}(M_{n,n}(E^*)) = M_{2.n,2.n}(E^*)$$

To show the converse inclusion,

$$M_{n,n}(E^*) \; C \; M_{n,n}(E)^*$$

we shall need the following result

Theorem 2 (Realisation theorem):

Let D be a R-closed semi-ring, E a subset D; let G and H be two subsets of E. We shall only assume that:

E, G and H all contain ε and e.

Then, an element x belongs to the closure E^* of E if and only if it can be written:

$$x = C.A^*.B$$

with

$C \in M_{1,n}(H)$, $A \in M_{n,n}(E)$ et $B \in M_{n,1}(G)$

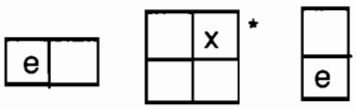
and n is an integer wich depends on x.

We shall call this a (H,E,G)-representation of x.
n will be called the dimension of the representation.

Demonstration:

Let R be the subset of all elements which have a (H,E,G)-representation. We only need to show that R is rationnaly closed and that it contains E.

R contains E because every elemnt of E can be written

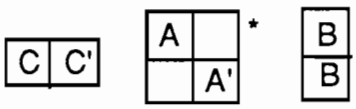
$x =$ 

(empty terms in the matrix have coefficient ε)

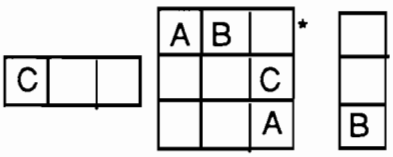
Now let us show that R is R-closed. If

$x = C.A^*.B$
$y = C'.A'^*.B'$

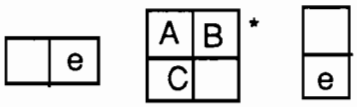
$x \oplus y = C.A^*.B \oplus C'.A'^*.B'$

$=$ 

$x.y = (C.A^*.B).(C'.A'^*.B')$

$=$ 

$x^* = (C.A^*.B)^*$

$=$ 

Thus R is R-closed and contains E. We have proved that $R=E^*$, thus the realisation theorem is proved.

Demonstration of theorem 1 (end):

We still have to show the inclusion:

$$M_{n,n}(E^*) \subset M_{n,n}(E)^*$$

We shall prove this in the case of n=2, but our proof can obviously be extended to any dimension.

Let X be a matrix with coefficients in E^*. All its coefficients have $(\{\varepsilon;e\},E,\{\varepsilon,e\})$-representations. By creating additional coefficients equal to ε, we can enlarge the smallest rtepresentation, in order to suppose that all elemnts in the matrix have representations of the same deimension p.

$$X = \begin{array}{|c|c|}\hline C_1.A_1^*.B_1 & C_2.A_2^*.B_2 \\ \hline C_3A_3^*.B_3 & C_4.A_4^*.B_4 \\ \hline \end{array}$$

$$= \begin{array}{|c|c|c|c|}\hline C_1 & C_2 & & \\ \hline & & C_3 & C_4 \\ \hline \end{array} \cdot \begin{array}{|c|c|c|c|}\hline A_1 & & & \\ \hline & A_2 & & \\ \hline & & A_3 & \\ \hline & & & A_3 \\ \hline \end{array} \cdot \begin{array}{|c|c|}\hline B_1 & \\ \hline & B_2 \\ \hline B_3 & \\ \hline & B_4 \\ \hline \end{array}$$

If we look at the blocs of size (2,2) in this matrix, we see that it is $a(M_{2,2}(E), M_{2,2}(E), M_{2,2}(E))$ representation of the matrix X thus by applying theorm 2, we get:

$$X \in (M_{2,2}(E))^*$$

This completes the proof of theorem 1.

References.

Simulation of discrete-event systems.

[SIM1] J. REITMAN.
Computer simulation Applications.
Discrete-Event Simulation for Synthesis and Analysis of Complex Systems.
Wiley-Interscience, 1971.

[SIM2] R.F. GARZIA, M.R. GARZIA, B.P. ZEIGLER, Discrete-event simulation.
IEEE-Spectrum, December 1986.

[SIM3] J.P. BEVANS,
First, choose an FMS simulator.
American Machinist, pp 143-145, May 1982.

[SIM4] B.P. ZEIGLER,
Multifaceted Modeling and Discrete Event Simulation.
Academic Press, 1984.

Mathematical approaches based on event-sequences.

[EVSE1] G. COHEN, P. MOLLER, J.P. QUADRAT, M. VIOT,
Dating and counting in Discrete Event Systems.
25rd IEEE conf. on Decision and Control, Athens, Greece, December 1986.

[EVSE2] G. COHEN, P. MOLLER, J.P. QUADRAT, M. VIOT,
Linear System Theory for Discrete Event Systems.
23rd IEEE conf. on Decision and Control , December 1984.

[EVSE3] G. COHEN, D. DUBOIS, J.P. QUADRAT, M. VIOT,
A linear-System-Theoretic view of Discrete-Event Processes
and its use for Performance evaluation in Manufacturing.
IEEE Trans. on Automatic Control , 1985.

[EVSE4] G. COHEN, P. MOLLER, J.P. QUADRAT, M. VIOT,
Une théorie linéaire des systèmes à évènements discrets.
Rapport I.N.R.I.A N° 362 , Le Chesnay , Janvier 1985.

[EVSE5] P. CASPI, N. HALBWACHS,
An approach to real-time-systems modelling.
Rapport de recherche n° 253, IMAG Grenoble, Juin 1981.

[EVSE6] P. CASPI, N. HALBWACHS,
Algebra of Events: A model for parallel and real-time systems.
Rapport de recherche n° 285, IMAG Grenoble, Janvier 1982.

[EVSE7] P. CASPI, N. HALBWACHS,
A functional model for describing and reasoning about time behaviour of computing systems.
Acta Informatica n°22, pp 595-627, Springer Verlag,1986.

Control approaches to discrete-event-system.

[CADE1] S.B. GERSHWIN, R.R. HILDEBRANT, R. SURI, S.K. MITTER,
A Control Theorist's perspective on recent trends in manufacturing systems.
MIT Lab for Info. & Dec. Systems Teport LIDS-P-1408, August 1985.

[CADE2] S.B. GERSCHWIN,
Opportunities for control in manufacturing systems.
IEEE Trans. on Automatic Control, Vol. AC-30, p 833, 1985.

[CADE3] P.J. RAMADGE,
Control and supervision of Discrete event processes.
Ph. D. Thesis, Department of electrical engineering, University of Toronto, 1983.

[CADE4] P.J. RAMADGE, W.M. WONHAM, On the supremal controlable sublangage of a given langage.
23rd IEEE conf. on Decision and Control , Las Vegas, Nevada, December 1984.

[CADE5] P.J. RAMADGE, W.M. WONHAM,
On modular synthesis of supervisory control for discrete-event processes.
in "International conf. on computers, Systems & signal " Bangalore, India, December 1984.

[CADE6] P.J. RAMADGE, W.M. WONHAM,
Proceedings of the Seventh international conference on
analysis and optimisation of systems, Antibes, France,1986.
In Lecture Notes in Control and information sciences, Springer-Verlag, 1986.

Other approaches to DES.

[OADE1] Y.C. HO, C. CASSANDRAS,
Computing costate variables for Discrete-Event Systems.
Proc. IEEE Conf on Decision and Control, Albuquerque, pp. 697-700, 1980.

[0ADE2] Y.C. HO, C. CASSANDRAS,
A new approach to the analysis of discrete-event systems.
Automatica, vol.19, pp 149-168, 1983.

[OADE3] A. HAURIE, P. L'ECUYER,
Approximation and bounds in Discrete -event Programming,
IEEE Trans. on Automatic Control, vol. AC-31, n°3, March 1986.

Petri nets.

[PETRI1] J.L. PETERSON,
Petri Net Theory and the modelling of systems.
PRENTICE HALL, 1981.

[PETRI2] C. RAMCHANDANI,
Analysis of asynchronous concurrent systems by Petri nets.
PhD Thesis, MIT, Cambridge Mass. Project MAC TR 120, February 1974.

[PETRI3] P. CHRETIENNE,
Les réseaux de petri temporisés.
Thèse , Université Pierre et Marie Curie (Paris VI) , 1983.

[PETRI4] J. SIFAKIS,
Use of Petri nets for performance evaluation. Mesuring, modelling and evaluating computer systems,
NORTH HOLLAND PUB. CO., 1977.

[PETRI5] C.V. RAMAMOORTHY, G.S. HO,
Performance evaluation of asynchronous concurrent systems using Petri nets.
IEEE Trans. on Software Eng., 6, n° 5 , 1980.

[PETRI6] M.K. MOLLOY,
Performance analysis using stochastic Petri nets.
IEEE Trans. on Comput., Vol. C-31, pp 913-917, September 1982.

[PETRI7] M. AYMONE-MARSAN, G. CHIOLA, G. CONTE,
A class of generalized stochastic Petri nets
for the performance evaluation of complex systems.
ACM Trans; on Comp. Syst., vol.2, pp 93-122, May 1984.

[PETRI8] R.R. RAZOUK, C.V. PHELPS,
Performanceanalysis using timed Petri nets.
in Proc. Conf. Parallel Processing, pp126-129, August 1984.

[PETRI9] M.A. HOLLIDAY, M.K. VERNON,
A generalized timed Petri net fot performance evaluation.
in Proc. Int. Workshop on Timedd Petri Nets, July 1985.

Various works on Manufacturing systems.

[MANS1] J.L. CASTI,
Mafacturing as a system-determined science.
IIASA Working Paper WP-86-43, October 1986.

[MANS2] R.A. CUNNINGHAM-GREEN,
Describing industrial processes and approximating their steady-state behaviour.
Op. Research Quaterly, 13, pp. 95-100, 1982.

[MANS3] S.B. GERSCHWIN, A. RAMAKRISHNAN, Y. CHOONG,
Short term Production Scheduling of an automated manufacturing system.
IBM Journal on Res. & Dev. , n° 29, pp 392-400, 1985.

[MANS4] J. B. CAVAILLE, D. DUBOIS,
Heuristic methods based on mean-value analysis for flexible manufacturing systems performance evaluation.
21st IEEE conf. on Decision and Control, Orlando, Florida, 1982.

[MANS5] E.G. COFFMAN, Editor.
Computer and Job-Shop scheduling theory.
WILEY, New-York, 1976.

Related topics.

[RETO1] R. MILNER
A Caculus of communicating systems.
Springer Verlag, 1980.

[RETO2] J.L. BAER.
A survey of some theoretical aspects of multiprocessing.
ACM Comput. Surv., vol.5, pp 31-80, March 1983.

[RETO3] L.R. RABINER, B.H. JUANG,
An introduction to Hidden Markov Models,
IEEE ASSP Magazine, January 1986.

Algebraic automata theory and discrete-mathematics.

[ALAT1] J.E. HOPCROFT, J.D. ULLMANN,
Introduction to Automata theory, languages and computation.
Addison-Wesley, Reading, Massachusetts, 1979.

[ALAT2] S. EILENBERG
Automata, languages and Machines.
Volume A and B.
Academic Press 1974.

[ALAT3] P. MOLLER
Introducing real time in the algebraic theory of finite Automata.
IIASA working paper 86-49 September 1986.

[ALAT4] D. ZEILBERGER A combinatorial approach to matrix algebra.
Discrete Mathematics 56, pp 61-72, NORTH-HOLLAND, 1985.

Some of the numerous papers on queuing networks..

[QUNE1] A. THOMASIAN, P.F. BAY,
Analytic queueing network models for parallel processing of task systems.
IEEE Trans. on computers, vol. C-35, pp 1045-1054, December 1986.

[QUNE2] P. HEIDELBERGER, K.S. TRIVEDI,
Analytic queuing model for programs with internal concurrency.
IEEE Trans. on Comp. , vol. C-32, pp 73-82, January 1983.