Pergamon

Automatica Vol. 30, No. 2, pp. 175–202, 1994

0005-1098/94 $6.00 + 0.00

Survey Paper

Petri Nets for Modeling of Dynamic Systems—A Survey*

RENÉ DAVID† and HASSANE ALLA†

A review of the basic concepts relative to Petri nets and the various classes of derived models shows that Petri nets can be used to model discrete event systems of any kind whatsoever.

Key Words—Petri net; properties; modeling; dynamic systems; discrete event systems; continuous systems.

Abstract—Petri nets enable a discrete event system of any kind whatsoever to be modeled. They present two interesting characteristics. Firstly they make it possible to model and visualize behaviors comprising concurrency, synchronization and resource sharing. Secondly the theoretical results concerning them are plentiful. The aim of this paper is to present the basic concept relative to Petri nets and the various classes of derived models which can be used for dynamic system modeling. The tool enables qualitative and quantitative analysis and its numerous applications have been still further increased by a number of research workers to enable more condensed descriptions, even where the time factor intervenes, such as synchronized, timed, stochastic, colored and continuous models. Each of these models thus has its own specific character and privileged fields of application. Nevertheless, the ordinary Petri net forms a common basis: it may be likened to a 'common language' allowing dialogue between persons of very varied training backgrounds.

1. INTRODUCTION

CARL ADAM PETRI is a contemporary German mathematician who defined a general purpose mathematical tool for describing relations existing between conditions and events (Petri, 1962). This work was conducted in the years 1960–1962. Since then, these nets have resulted in considerable research, some in the United States, notably the work carried out at MIT (Massachusetts Institute of Technology) in the early seventies, but above all in Europe ever since. The European Workshop (rechristened the International Conference in 1989) on the Theory and Applications of Petri Nets, which has been held every year since 1980, is without doubt the international conference where the largest number of results have been presented on the subject. Another one, the International Workshop on Petri Nets and Performance Models, has been held every other year since 1985.

Petri nets (PNs) present two interesting characteristics. Firstly they make it possible to model and visualize behaviors comprising concurrency, synchronization and resource sharing. Secondly the theoretical results concerning them are plentiful. The tool enables qualitative analysis and its numerous applications have been still further added to by a number of research workers to enable more condensed descriptions, including where the time factor intervenes.

The aim of this paper is to present the basic concepts relative to Petri nets and the various classes of derived models which can be used for dynamic system modeling (David, 1991). This paper does not claim to be a survey on the properties and research about Petri nets, but mainly to show people concerned with automatic control what the various kinds of Petri nets are and how they can be used [there are many applications; Silva and Valette (1990) provides a survey of applications for manufacturing systems, for example]. These models will then intuitively be shown through simple examples.

Many papers can be found in Rozenberg (1984–1993) and Memmi (1985). Murata (1989) provides a good survey on properties, analysis and applications of Petri nets. Several books on Petri nets have been published: Peterson (1981), Brams (1983) and Genrich and Lautenbach (1983) are general purpose books on Petri nets where theory takes an important place; some more recent theoretical results are presented in Reutenauer (1990), Silva (1985) devotes an important part to implementation of Petri nets, David and Alla (1989, 1992) present the various kinds of Petri nets which are used for modeling dynamic systems.

The basic notions are illustrated in Section 2. In Section 3 ordinary PNs, their abbreviations and extensions are presented; these models allow qualitative analysis. The PNs in which external synchronization and time intervene are presented in Section 4: they are called non-autonomous PNs (when a PN is neither synchronized nor timed, it may be called

* Received 7 July 1992; received in final form 24 March 1993. This paper was not presented at any IFAC meeting. This paper was recommended for publication in revised form by Editor K. J. Åström. Corresponding author H. Alla. Tel. (33)76826244; Fax (33)76826388; E-mail alla@lag.grenet.fr.

† Laboratoire d'Automatique de Grenoble (Institut National Polytechnique de Grenoble, associated with the Centre National de la Recherche Scientifique), BP 46, 38402 Saint-Martin-d'Hères, France.

autonomous). Concluding remarks are given in Section 5.

2. BASIC NOTIONS

A Petri net comprises two types of nodes, namely places and transitions. A place is represented by a circle and a transition by a bar (certain authors represent a transition by a box). Places and transitions are connected by arcs. The number of places is finite and not zero. The number of transitions is also finite and not zero. An arc is directed and connects either a place to a transition or a transition to a place.

In other words, a PN is a bipartite graph, *i.e.* places and transitions alternate on a path made up of consecutive arcs. It is compulsory for each arc to have a node at each of its ends.

2.1. *Marking*

A PN which is marked is such that every place it contains an integer (positive or zero) of tokens or marks. The number of tokens contained in a place P_i will be called either $M(P_i)$ or m_i. For example in Fig. 1(c) before firing, we have $m_1 = 2$, $m_2 = 1$, $m_3 = m_4 = 0$. The net marking, $\mathbf{M}$, is defined by the vector of these markings, *i.e.* $\mathbf{M} = (m_1, m_2, \ldots, m_n)$. The marking at a certain time defines the state of the PN, or more precisely the state of the system described by the PN. The evolution of the state thus corresponds to an evolution of the marking, an evolution which is caused by firing of transitions, as we shall see.

We shall practically always consider marked Petri nets. We shall call them quite simply Petri nets. On the other hand, we shall specify unmarked PNs when necessary.

2.2. *Firing of a transition*

A transition can only be fired if each of the input places of this transition contains at least one token. The transition is then said to be enabled. The transitions in Fig. 1(a, b and c) (before firing) are enabled because in each case places P_1 and P_2 contain at least one token. This is not the case for the example of Fig. 1(d) in which transition T_1 is not enabled, since P_1 does not contain any tokens.

Firing of a transition T_j consists in withdrawing a token from each of the input places of transition T_j and in adding a token to each of the output places of transition T_j. This is illustrated in Fig. 1. In Fig. 1(b) we note that there are two tokens in place P_3 after firing because there was already one beforehand. In Fig. 1(c) we observe that a token remains in place P_1 after firing.

When a transition is enabled, this does not imply that it will be immediately fired. This only remains a possibility.

The firing of a transition is indivisible. Although the concept of duration does not enter into a PN (if it is neither timed nor synchronized), it is useful to consider that the firing of a transition has a zero duration, in order to facilitate understanding of the concept of indivisibility. It is usually considered that only one firing occurs at a time.

2.3. *First example*: *billiard balls*

The first example is illustrated in Fig. 2. Two billiard balls, A and B, move on the same line parallel to one of the bands. In Fig. 2(a), A moves to the right while B moves to the left. (We assume they have the same speed.) The conditions for the event hitting the balls to occur are satisfied. When this event occurs,

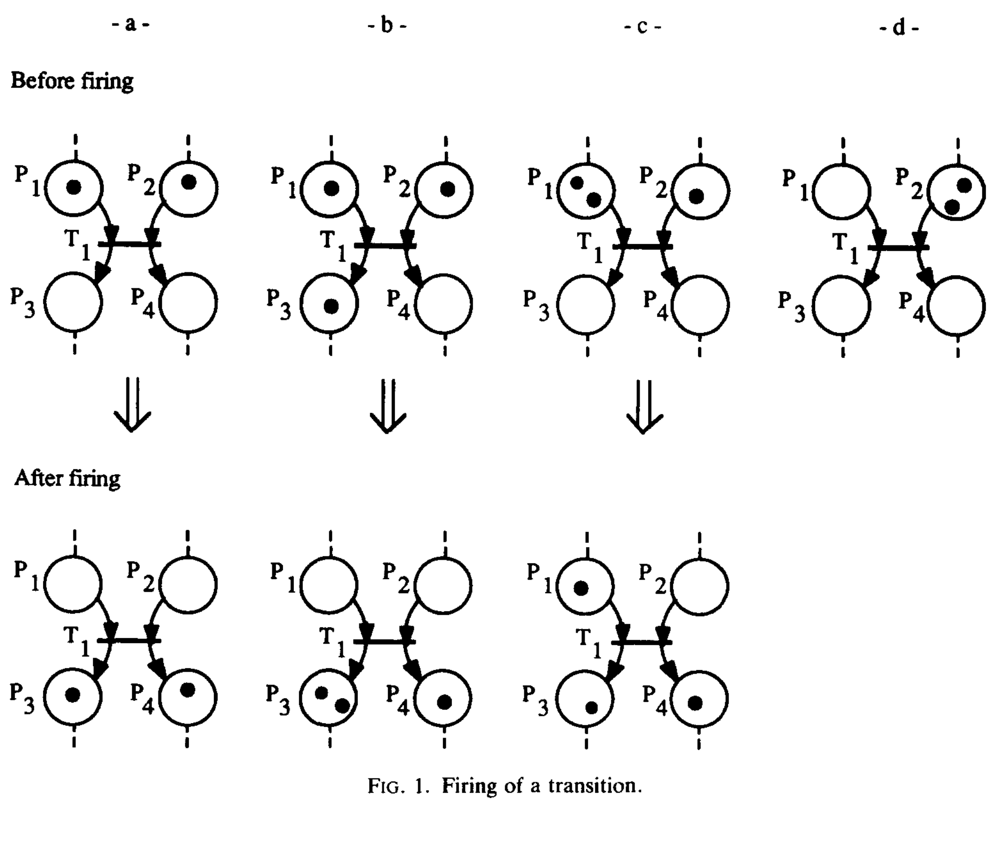


FIG. 1. Firing of a transition.

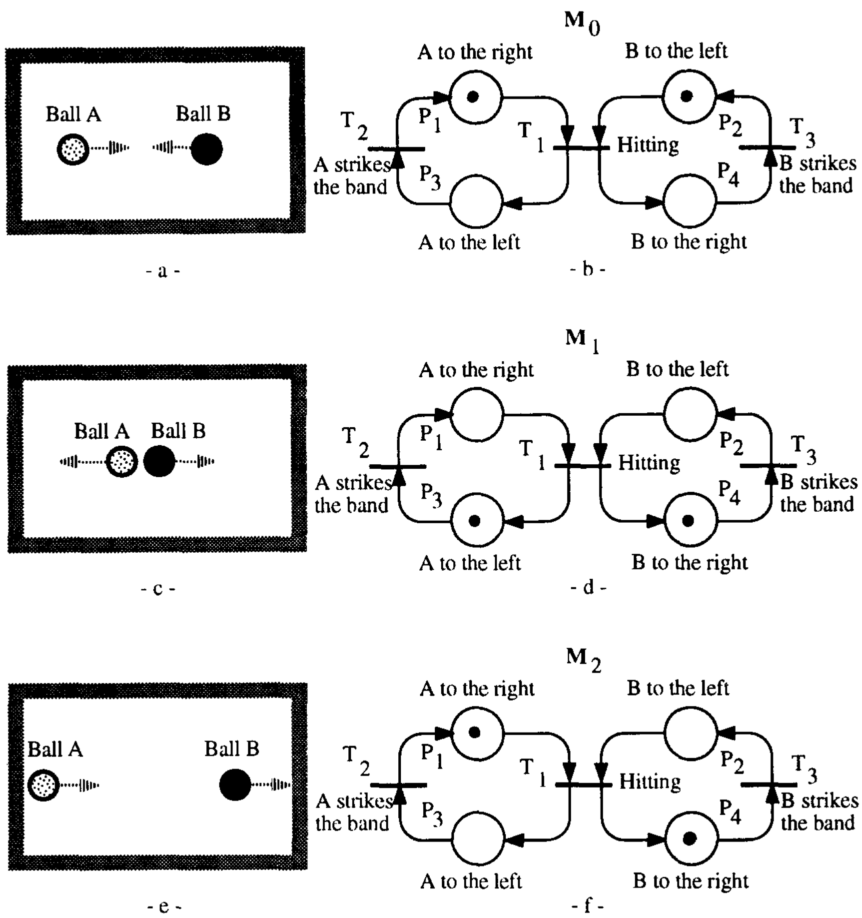


FIG. 2. First example: billiard balls.

the balls set off again in the opposite direction at the same speed [Fig. 2(c)]. A ball which strikes a band sets off in the other direction at the same speed [Fig. 2(e)].

This is modeled by the Petri net in Fig. 2(b, d and e). The event hitting is associated with transition T_1. When places P_1 and P_2 are marked, the conditions for this event to occur are fulfilled. This appears in Fig. 2(b) because transition T_1 is enabled. We do not know when this event will occur, but we know that it will occur (because T_1 is the only enabled transition) for the current marking $\mathbf{M}_0$. After hitting, *i.e.* firing of transition T_1, the marking is $\mathbf{M}_1 = (0, 0, 1, 1)$, which is illustrated in Fig. 2(d). Then, there are two enabled transitions, namely T_2 and T_3. Transition T_2 corresponds to the event A strikes the left band, and T_3 corresponds to the event B strikes the right band. We do not know which of these two events will occur first. If T_2 is fired before T_3, the marking becomes $\mathbf{M}_2 = (1, 0, 0, 1)$ as shown in Fig. 2(f). In Fig. 2(f), only transition T_3 is enabled, then the next event will be B strikes the right band; after firing of T_3 the initial marking $\mathbf{M}_0$ is obtained again.

Concepts illustrated in Fig. 2

1. This is an autonomous PN. That means that neither time nor external synchronization is involved in this model. This is a qualitative description of the system which is observed.
2. The set of reachable markings from $\mathbf{M}_0$ is $^*\mathbf{M}_0 = \{\mathbf{M}_0, \mathbf{M}_1, \mathbf{M}_2, \mathbf{M}_3\}$, where $\mathbf{M}_3 = (0, 1, 1, 0)$. Markings $\mathbf{M}_0$, $\mathbf{M}_1$ and $\mathbf{M}_2$ have already been presented. Marking $\mathbf{M}_3$ can be reached from $\mathbf{M}_1$, if T_3 is fired before T_2.
3. The PN is bounded. This means that for every reachable marking, the number of tokens in every place is bounded. Furthermore, the PN is safe, *i.e.* the marking of every place is either 0 or 1 (Boolean marking).
4. The PN is live. This means that, regardless of the evolution, no transition will become unfirable on a permanent basis.
5. There are two marking invariants. The first one is $M_i(P_1) + M_i(P_3) = 1$. This means that for any reachable marking $\mathbf{M}_i$, the number of tokens in the set of places $\{P_1, P_3\}$ is equal to 1. One can simply write $m_1 + m_3 = 1$. This invariant has a clear physical meaning: ball A may have two states, namely moving to the right and moving to the left, and it is always in one and only one state. Place P_1 is associated with the first state, and P_3 is associated with the second one. The set $\{P_1, P_3\}$ is a conservative component. Similarly, $m_2 + m_4 = 1$. By adding the two minimal marking invariants, $m_1 + m_2 + m_3 + m_4 = 2$ is obtained. This is a new marking invariant. The whole PN is conservative since the last marking invariant contains all the places in the PN.
6. From $\mathbf{M}_0$, the firing sequence $T_1T_2T_3$ causes a

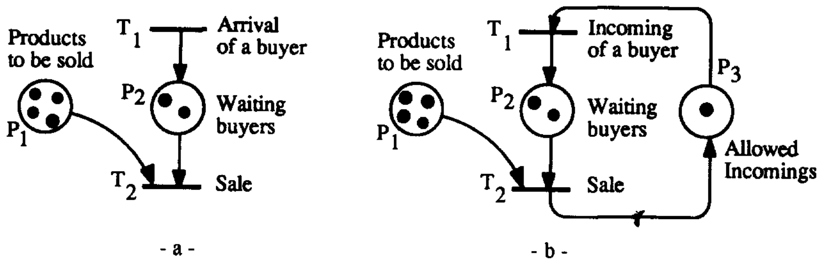


FIG. 3. Second example: products on sale.

return to the initial state. Then this sequence is repetitive. The firing sequence $T_1T_3T_2$ is also repetitive. These sequences are different, but they contain the same number of firings for each transition. There is a firing invariant whose meaning is: if every transition in the set $\{T_1, T_2, T_3\}$ is fired once from a state $\mathbf{M}$, then the corresponding firing sequence causes a return to $\mathbf{M}$.

7. This example illustrates concurrency. When the marking $\mathbf{M}_2 = (0, 0, 1, 1)$ is reached, the firing of T_2 and T_3 are causally independent (*i.e.* concurrent, they may occur in any order).
8. This example illustrates synchronization. Although the balls behave independently of each other for some time, ball A cannot change from 'moving to the right' to 'moving to the left' independently of ball B (and *vice versa*). This synchronization of both changes of direction is illustrated by transition T_1.

2.4. *Second example: products on sale*

Consider the behavior illustrated in Fig. 3(a). There are two places and two transitions. Place P_1 corresponds to the number of products to be sold. In the figure, $m_1 = 4$ means that there are currently four products to be sold. Place P_2 corresponds to the number of buyers who are waiting. Firing of T_2 corresponds to selling a product.

Concepts illustrated in Fig. 3(*a*).

1. T_1 is a source transition, *i.e.* without input place. This transition is always enabled. Its firing corresponds to adding a token to P_2.
2. T_2 is a sink transition, *i.e.* without output place. Its firing corresponds to taking one token away from P_1 and one token from P_2.
3. The PN is unbounded. Each firing of T_1 adds a token to P_2. Since T_1 is always enabled, the marking of P_2 is unbounded.
4. The PN is not live. As a matter of fact, when T_2 has been fired four times, there is no token left in P_1, and T_2 is not enabled. Since P_1 has no input transition, $m_1 = 0$, definitely. After the firing sequence, $S = T_2T_2T_1T_2T_1T_2$, for example, transition T_2 will never be enabled. There is neither a conservative component, nor a repetitive sequence.

In Fig. 3(b), entry of a buyer is not allowed if there are three buyers already waiting.

Concepts illustrated in Fig. 3(*b*).

1. The number of tokens in place P_2 is limited by a complementary place P_3. Due to the marking invariant $m_2 + m_3 = 3$, there is always $m_2 \leq 3$.
2. There is a deadlock. After the firing sequence $S = T_2T_2T_1T_2T_1T_2T_1T_1T_1 = (T_2)^2(T_1T_2)^2(T_1)^3$, there is no transition enabled. The marking $\mathbf{M} = (0, 3, 0)$ has been reached, forever.

2.5. *Third example: two computers use a common memory*

This is illustrated in Fig. 4. Computer CP_1 has three possible states: either it requests the memory (place P_1), or it uses it (P_2), or it does not need it (P_3). Similarly, computer CP_2 has three possible states. When the memory is free (place P_7 marked) and CP_1 requests it, transitions T_1 is enabled [Fig. 4(a)]. If T_1 is fired, then CP_1 uses the memory [Fig. 4(b)]. When CP_1 has finished, transition T_2 is fired, then the marking in Fig. 4(c) is reached (the memory is released, and may be re-used either by CP_1 or by CP_2).

Concepts illustrated in Fig. 4.

1. There is a conflict. The place P_7 is an input of both transitions T_1 and T_4. This is a structural conflict. Now, when there is one token in every place P_1, P_4 and P_7 [Fig. 4(a)], there is an effective conflict between transition T_1 and T_4. As a matter of fact both transitions are enabled, but only one can be fired. If T_1 is fired, then T_4 is no longer enabled, and *vice versa*.
2. Resource sharing. The memory may be used by two computers, but not at the same time (this implies a conflict). One can observe that there is a marking invariant $m_2 + m_5 + m_7 = 1$. This means that if there is a token in P_2, there is no token in P_5 and *vice versa*. This property expresses that the memory cannot be used by both computers at the same time (mutual exclusion).

2.6. *Fundamental equation and invariants*

The marking of a PN at a given moment is a column vector whose ith component is the marking of place P_i at this moment. In order to facilitate writing, we write the markings in the transposed form in the text. We use square brackets to represent a matrix and curly brackets to represent the transposed form. For example for the marking of Fig. 2(b):

$$\mathbf{M}_0 = (1, 1, 0, 0) = [1 \quad 1 \quad 0 \quad 0]^{\mathrm{T}} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \end{bmatrix}.$$

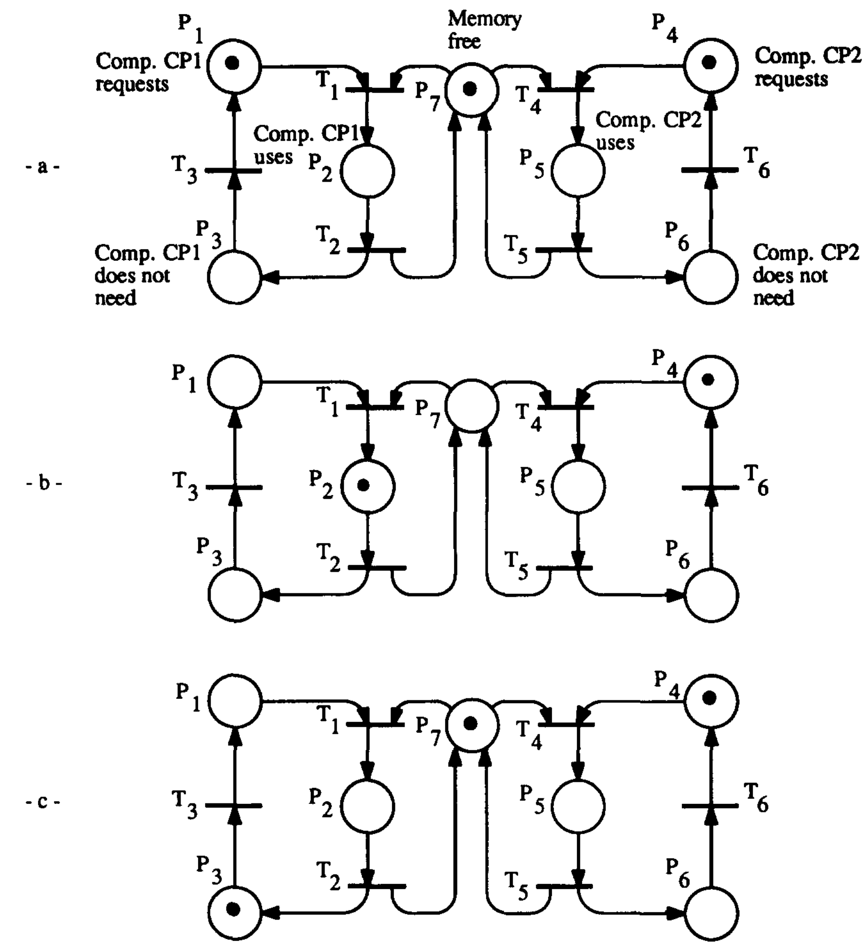


FIG. 4. Third example: two computers use a common memory.

The incidence matrix associated with a Petri net corresponds to its structure (independently of the marking). For example, the incidence matrix associated with the PN in Fig. 2 is:

$$\mathbf{W}=\begin{array}{c} \begin{array}{ccc} T_1 & T_2 & T_3 \end{array} \\ \left[\begin{array}{ccc} -1 & +1 & 0 \\ -1 & 0 & +1 \\ +1 & -1 & 0 \\ +1 & 0 & -1 \end{array}\right] \end{array} \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \end{array}$$

In this matrix, a row is associated with a place, and a column is associated with a transition. Each column corresponds to the marking modification when the associated transition is fired. For example, the first column means that when T_1 is fired, one token is withdrawn from P_1 and from P_2, while one token is added to P_3 and to P_4.

With a firing sequence S is associated a characteristic vector $\mathbf{S}$, whose ith component is the number of times T_i is fired in S. From the marking in Fig. 2(b), one can have, for example, the firing sequences $S_1=T_1$, $S_2=T_1T_2T_3$, or $S_3=T_1T_2T_3T_1T_3$, whose characteristic vectors are $\mathbf{S}_1=(1,0,0)$, $\mathbf{S}_2=(1,1,1)$, $\mathbf{S}_3=(2,1,2)$. A characteristic vector may correspond to several firing sequences; for example $(1,1,1)$ corresponds to both $T_1T_2T_3$ and $T_1T_3T_2$. A warning: not all the $\mathbf{S}$ vectors whose components are positive or zero integers are possible; for example, there is no firing sequence from $\mathbf{M}_0$ corresponding to $(0,1,1)$ since neither transition T_2 nor transition T_3 can be fired before a firing of transition T_1.

If a firing sequence S is applied from a marking $\mathbf{M}_i$, the marking $\mathbf{M}_k$ which is reached is given by the fundamental equation (1) where $\mathbf{W}\cdot\mathbf{X}$ is the normal matrix multiplication:

$$\mathbf{M}_k=\mathbf{M}_i+\mathbf{W}\cdot\mathbf{S} \qquad (1)$$

A vector $\mathbf{X}$ is a P-invariant if $\mathbf{X}^{\mathrm{T}}\cdot\mathbf{W}=0$ (Lautenbach and Schmid, 1974). Such a vector has an interesting property. From $\mathbf{M}_k=\mathbf{M}_0+\mathbf{W}\cdot\mathbf{S}$, where $\mathbf{S}$ is the characteristic vector of a firing sequence S leading from the initial marking $\mathbf{M}_0$ to the reachable marking $\mathbf{M}_k$, we obtain $\mathbf{X}^{\mathrm{T}}\cdot\mathbf{M}_k=\mathbf{X}^{\mathrm{T}}\cdot\mathbf{M}_0+\mathbf{X}^{\mathrm{T}}\cdot\mathbf{W}\cdot\mathbf{S}$. Since $\mathbf{X}^{\mathrm{T}}\cdot\mathbf{W}=0$, it can be deduced that $\mathbf{X}^{\mathrm{T}}\cdot\mathbf{M}_k=\mathbf{X}^{\mathrm{T}}\cdot\mathbf{M}_0$, for any reachable $\mathbf{M}_k$. A P-invariant is a structural property since it does not depend on the marking. However the value $\mathbf{X}^{\mathrm{T}}\cdot\mathbf{M}_0$ depends on the initial marking and it corresponds to a marking invariant which is associated with $\mathbf{X}$.

For example $\mathbf{X}=(1,0,1,0)$ is a P-invariant for the PN in Fig. 2 (independent of the marking). For the initial marking in Fig. 2(b), the corresponding marking invariant is $m_1+m_3=1$ since $\mathbf{X}^{\mathrm{T}}\cdot\mathbf{M}_0=1$. This means that the number of tokens in the set of places $\{P_1, P_3\}$ is constant. A P-invariant corresponds to a vector of weights associated with the places. In $\mathbf{X} = (1, 0, 1, 0)$, the weight 1 is associated with P_1 and P_3 and the weight 0 is associated with both P_2 and P_4. In general the weight associated with a place may be any integer, but usually one is mainly interested by P-invariants whose weights are positives. The set of places having a weight not nil in such an invariant is a conservative component. This means that the weighted number of tokens in this set of places is constant.

A vector $\mathbf{Y}$ is a T-invariant if $\mathbf{W} \cdot \mathbf{Y} = 0$. If a firing sequence S exists from a marking $\mathbf{M}_i$ such that $\mathbf{S} = q\mathbf{Y}$ (where q is a positive integer), then S leads back to $\mathbf{M}_i$. As a matter of fact the marking reached from $\mathbf{M}_i$ by the firing sequence S is given by equation (1). Since $\mathbf{W} \cdot \mathbf{S} = 0$, then $\mathbf{M}_k = \mathbf{M}_i$.

For example $\mathbf{Y} = (1, 1, 1)$ is a T-invariant for the PN in Fig. 2 (independent of the marking). This means that if there is some firing sequence S applicable from $\mathbf{M}_i$, in which each transition T_1, T_2 and T_3 appear the same number of times (because their weights are the same in $\mathbf{Y}$), then firing of S from $\mathbf{M}_i$ leads back to $\mathbf{M}_i$. For the initial marking in Fig. 2(b), both firing sequences $S_1 = T_1T_2T_3$ and $S_2 = T_1T_3T_2$ are such that $\mathbf{S}_1 = \mathbf{S}_2 = \mathbf{Y}$.

It appears from the definitions that the sum of two P-invariants is a P-invariant, and that the sum of two T-invariants is a T-invariant. Then minimal invariants exist from which the others can be constructed by composition.

2.7. *Abbreviations, extensions and particular structures*

Let us divide up Petri nets which can be found in the literature into three main classes: ordinary Petri nets (the basic model), abbreviations and extensions.

In an ordinary Petri net all the arcs have the same weight which is 1, there is only one kind of token, the place capacities are infinite (*i.e.* the number of tokens is not limited by place capacities), the firing of a transition can occur iff every place preceding it contains at least one token, and no time is involved. Up to now only ordinary PNs have been presented.

The abbreviations correspond to simplified representations, useful in order to lighten the graphical representation, to which an ordinary Petri net can always be made to correspond. Generalized PN, finite capacity PN and colored PN are abbreviations. Then they have the same power of description as the ordinary Petri nets.

The extensions correspond to models to which functioning rules have been added, in order to enrich the initial model, enabling a greater number of applications to be treated. Three main subclasses may be considered. The first subclass corresponds to models which have the description power of Turing machines: inhibitor arc PN and priority PN. The second subclass corresponds to extensions allowing modeling of continuous and hybrid systems: continuous PN and hybrid PN. The third subclass corresponds to non-autonomous Petri nets, which describe the functioning of systems whose evolution is considered by external events and/or time: synchronized PN, timed PN, interpreted PN and stochastic PN.

Some particular Petri nets structures have interesting properties, for example event graphs and state graphs which are important from a practical point of view. Both event graphs and state graphs are ordinary PNs.

An event graph is such that every place has exactly one input and one output transition. For example the PN in Fig. 2 is an event graph. Another event graph is presented in Fig. 5(a). In an event graph there is no conflict, since every place has exactly one output transition. In general, there are synchronizations, corresponding to transitions with several input places. Then event graphs are well adapted to modeling systems whose qualitative behavior is deterministic. The event graph in Fig. 2 has an additional property: it is strongly connected. This means that for any pair of nodes u and v (places or/and transitions) there is a path from u to v. For example from P_1 to P_2 there is the path $P_1T_1P_4T_3P_2$. Since there is a path from any node to any other node, there are circuits leading from a node back to itself. Circuits passing at most once by a node are called elementary circuits. In the event graph in Fig. 2 there are two elementary circuits, namely $P_1T_1P_3T_2$ and $P_2T_1P_4T_3$. A strongly connected event graph has the following properties: (1) a P-invariant is associated with every elementary circuit [for example $\mathbf{X}_1 = (1, 0, 1, 0)$ is associated with $P_1T_1P_3T_2$ and $\mathbf{X}_2 = (0, 1, 0, 1)$ is associated with

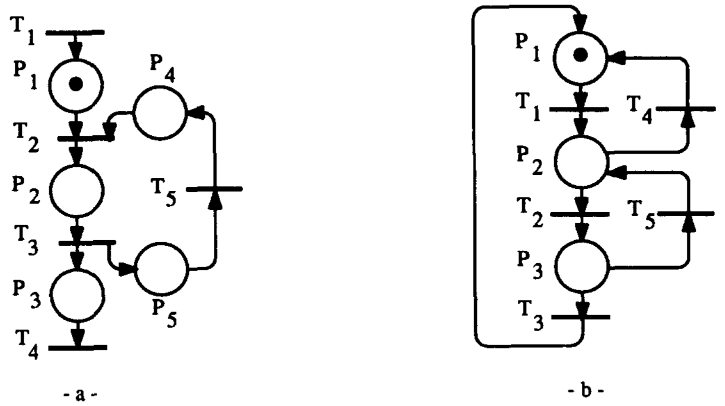


FIG. 5. (a) Event graph; (b) strongly connected state graph.

$P_2T_1P_4T_3$]; (2) there is a single T-invariant with a weight 1 associated with every transition [$\mathbf{Y} = (1, 1, 1)$ for the PN in Fig. 2]; (3) a strongly connected event graph is live if and only if there is at least one token in every elementary circuit.

A state graph is such that every transition has exactly one input and one output place. From the definition it appears that state graphs and event graphs have dual properties. In a state graph there is no synchronization and, in general, there are conflicts. For a strongly connected state graph [Fig. 5(b) for example]: (1) a T-invariant is associated with every elementary circuit; (2) there is a single P-invariant with a weight 1 associated with every place; (3) it is live if and only if there is at least one token in the PN. If a state graph contains exactly one token, it corresponds to a classical state diagram (a state is associated with every place).

3. MODELING BY AUTONOMOUS PNS

The different types of Petri nets do not correspond exactly to the functioning rules which have been previously laid down.

The abbreviations correspond to simplified representations, useful in order to lighten the graphism, but to which an ordinary Petri net (*i.e.* a marked autonomous Petri net functioning according to the rules laid down in Section 2) can always be made to correspond.

The extensions correspond to models to which functioning rules have been added, in order to enrich the initial model, thereby enabling a greater number of applications to be treated.

It follows that all the properties of ordinary Petri nets are maintained for the abbreviations with a few adaptations, whereas these properties are not all maintained for the extensions. However, the main concepts remain suitable.

The three examples in Section 2 correspond to ordinary Petri nets. We shall now present some examples of abbreviations and extensions retaining the autonomous character. In Section 4, non-autonomous Petri nets will be presented (obviously, non-autonomous Petri nets correspond to extensions).

3.1. *Generalized PN (abbreviation)*

A generalized PN is a PN in which weights (strictly positive integers) are associated to the arcs. Figure 6 represents a generalized PN such that the arcs $P_2' \to T_6$ and $T_3 \to P_2$ have the weights 5, and arcs $T_6 \to P_1$ and $P_1' \to T_3$ have the weights 3. All the other arcs, whose weights are not explicitly specified, have a weight 1. When an arc $P_i \to T_j$ has a weight p, this means that transition T_j will only be enabled if place P_i contains at least p tokens. When this transition is fired, p tokens will be taken away from place P_i. When an arc $T_j \to P_i$ has a weight p, this means that when T_j is fired, p tokens will be added to place P_i. Let us now comment on the meaning of the generalized PN in Fig. 6.

Let us assume that two tasks to be performed share the same central unit. Execution in Round Robin consists in performing a part of Task 1, followed by a part of Task 2, and so on. Figure 6 represents the following case: three instructions from Task 1, then five instructions from Task 2, and so on.

For the initial marking indicated [Fig. 6(a)], only transition T_6 is enabled. When it is fired, five tokens are taken away from place P_2', and three tokens are deposited in place P_1. The marking in Fig. 6(b) is then reached. The central unit is ready to execute three instructions from Task 1 (corresponding to three tokens in place P_1).

For the marking in Fig. 6(b), only transition T_1 is enabled. When it is fired, the marking in Fig. 6(c) is then reached. The token in place EX_1 means execution of an instruction from Task 1. Then, firing of T_2 after this execution leads to the marking in Fig. 6(d). Only transition T_1 is enabled, then another instruction from Task 1 will be executed, and so on.

When three instructions from Task 1 have been executed, there is a token in place A_1, a token in place A_2, and three tokens in place P_1'. Then, only transition T_3 is enabled. When it is fired, three tokens are taken away from place P_1', and five tokens are deposited in place P_2. The central unit is ready to execute five instructions from Task 2.

The marking invariants relating to Tasks 1 and 2 are $M(A_1) + M(EX_1) = 1$, and $M(A_2) + M(EX_2) = 1$, respectively. The marking invariant relating to the central unit is such that there are either three tokens in the set of places P_1, EX_1 and P_1', or five tokens in the set of places P_2, EX_2 and P_2'. Thus

$$5M(P_1) + 5M(EX_1) + 5M(P_1') + 3M(P_2) + 3M(EX_2) + 3M(P_2') = 15.$$

All generalized Petri nets can be transformed into ordinary PNs. A number of authors has proposed transformation principles. However this transformation is generally not useful since the properties of ordinary PNs can easily be adapted to generalized PNs. For example the fundamental equation (1) is true for generalized PNs. The incidence matrix of the generalized PN in Fig. 6 is

$$\mathbf{W} = \begin{array}{c} \begin{array}{cccccc} T_1 & T_2 & T_3 & T_4 & T_5 & T_6 \end{array} \\ \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & +3 \\ 0 & +1 & -3 & 0 & 0 & 0 \\ +1 & -1 & 0 & 0 & 0 & 0 \\ -1 & +1 & 0 & 0 & 0 & 0 \\ 0 & 0 & +5 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & +1 & -5 \\ 0 & 0 & 0 & +1 & -1 & 0 \\ 0 & 0 & 0 & -1 & +1 & 0 \end{bmatrix} \end{array} \begin{array}{l} \\ P_1 \\ P_1' \\ EX_1 \\ A_1 \\ P_2 \\ P_2' \\ EX_2 \\ A_2 \end{array}.$$

3.2. *Colored Petri net (abbreviation)*

Several models in which the tokens are identified have been defined. They are called high level Petri nets. Predicate Petri nets and colored PNs (with variants in their definitions) are common high level Petri nets.

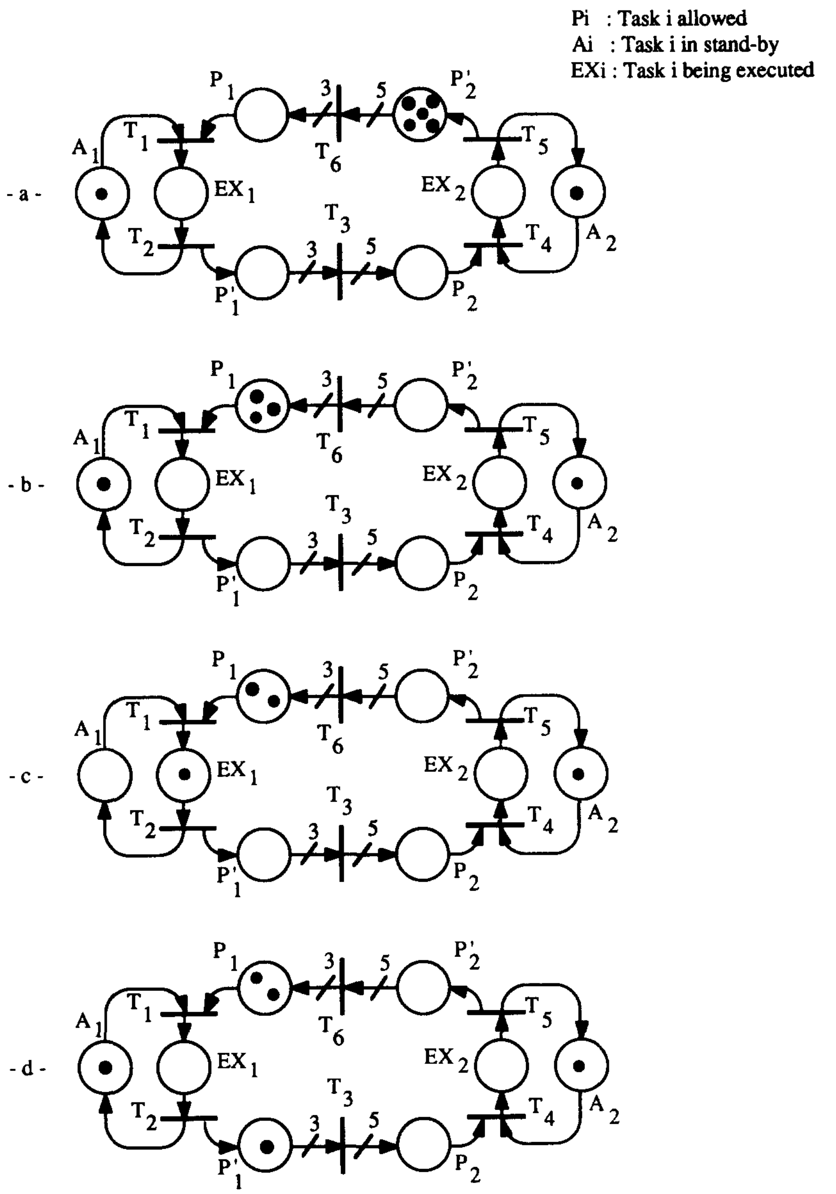


FIG. 6. Generalized Petri net.

The colored PNs comprise tokens to which colors are attributed (Jensen, 1980, 1981). They form a category of nets whose intuitive perception is less clear than for the generalized PNs. They are of great value for the modeling of certain complex systems.

An example illustrating the usefulness of colored PNs is given in Fig. 7. Figure 7(a) shows a 3-cell FIFO (first-in-first-out) system. An object can move from the left side to the right side, without passing over the preceding one. In Fig. 7(a), the object in cell 1 can move to cell 2, since cell 2 is empty. If it shifts, then cell 1 becomes empty and a new object can move into cell 1. In Fig. 7(a), it is also possible for the object in cell 3 to leave the queue.

The behavior of this system can be modeled by the Petri net in Fig. 7(b). In this PN there are two enabled transitions, namely T_1 and T_3. Transition T_1 corresponds to shifting from cell 1 to cell 2. It is enabled since there is a token in P_1 (an object is present in cell 1), and there is a token in P_2' (cell 2 is empty). Firing this transition consists in removing these tokens and in adding tokens in places P_1' (cell 1 is now empty) and P_2 (an object is now present in cell 2). Firing of transition T_3 corresponds to the departure of the object from cell 3: a token is removed from P_3 and added to P_3', because cell 3 becomes empty. If the number of cells was k, the PN would have $2k$ places and $k+1$ transitions (this will be a very big PN if $k = 100$ for example), with a repetitive structure; it is clear in Fig. 7(b) that the partial PN associated with transition T_1 is similar to the partial PN associated with transition T_2. It is then interesting to represent these transitions by a single one, and to associate colors with tokens and firings of transitions.

Figure 7(c) represents a colored PN describing the same system. Place P_{123} in Fig. 7(c) corresponds to the

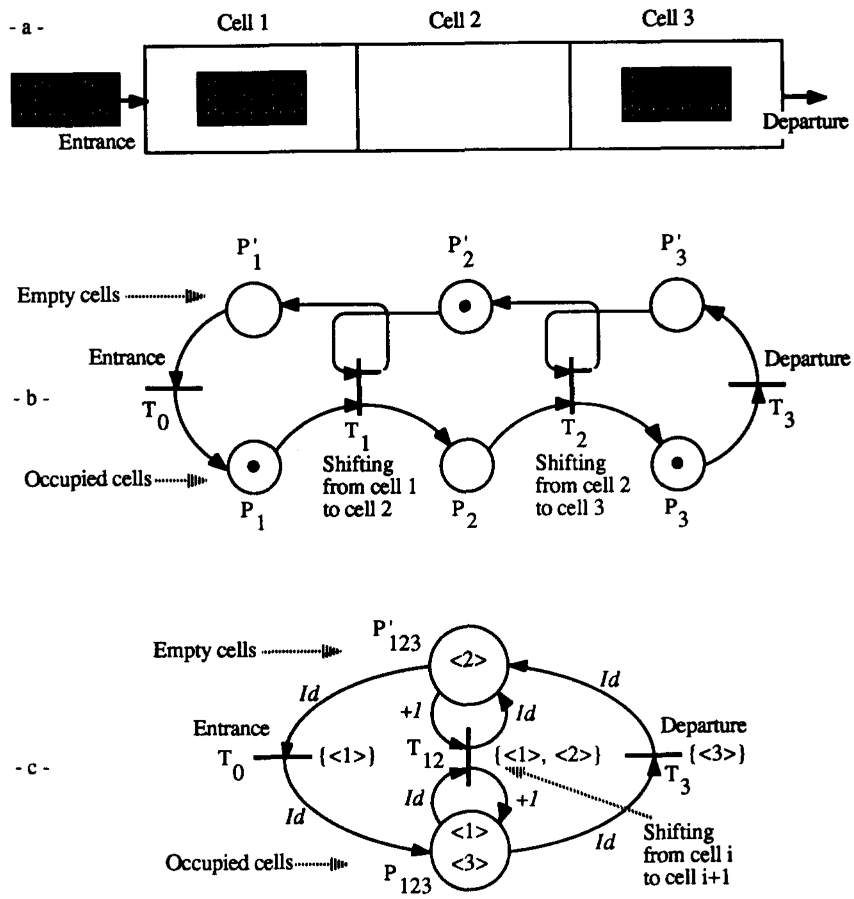


FIG. 7. Coloured Petri net. (a) FIFO system; (b) ordinary PN; (c) coloured PN.

set of places $\{P_1, P_2, P_3\}$ in Fig. 7(b). A colored token $\langle i \rangle$ in P_{123} corresponds to a token (without color) in P_i. Similarly, place P'_{123} in Fig. 7(c) corresponds to the set of places $\{P'_1, P'_2, P'_3\}$ in Fig. 7(b). A token $\langle i \rangle$ in P'_{123} corresponds to a token in P'_i. Now, transition T_{12} in Fig. 7(c) corresponds to the set of transitions $\{T_1, T_2\}$ in Fig. 7(b). Transition T_{12} may be fired with respect to any color in the set $\{\langle 1 \rangle, \langle 2 \rangle\}$: firing with respect to the color $\langle i \rangle$, corresponds to the firing of T_i in Fig. 7(b). With the arcs in Fig. 7(c), are associated functions. The function Id (meaning identity) is associated with several arcs. The function $+1$ is associated with $P'_{123} \rightarrow T_{12}$ and $T_{12} \rightarrow P_{123}$. Consider the transition T_{12}. It is enabled with respect to color $\langle 1 \rangle$, because there is a token of color $\mathrm{Id}\,\langle 1 \rangle = \langle 1 \rangle$ in place P_{123}, and a token of color $+1\langle 1 \rangle = \langle 2 \rangle$ in place P'_{123}. The corresponding firing consists in removing these tokens and in adding a token of color $+1\langle 1 \rangle = \langle 2 \rangle$ to place P_{123}, and a token of color $\mathrm{Id}\,\langle 1 \rangle = \langle 1 \rangle$ to place P'_{123}. If the number of cells was k, the colored PN would have the same graph with the same functions. The only difference is that the transition which is called T_{12} could be fired with respect to any color in the set $\{\langle 1 \rangle, \langle 2 \rangle, \ldots, \langle k-1 \rangle\}$, and that the colors of the tokens would be $\langle 1 \rangle, \langle 2 \rangle, \ldots, \langle k \rangle$.

In a general case, a color may be a n-tuple. For example, if there are two types of objects in the FIFO queue (type a, and type b), the color $\langle a, 2 \rangle$ could correspond to the presence of a type a object in cell 2.

3.3. *Finite capacity Petri net* (*abbreviation*)

A finite capacity PN is a PN in which capacities (strictly positive integers) are associated to places. Firing of an input transition of a place P_i, whose capacity is $\mathrm{Cap}\,(P_i)$ is only possible if firing of this transition does not result in a number of tokens in P_i which exceeds this capacity.

Consider the example in Fig. 7. Only one object can be present in a cell i. Then $M(P_i) \leq 1$, at any time. This is ensured in Fig. 7(b) since there is a marking invariant $M(P_i) + M(P'_i) = 1$. Now, the same system may be represented by the finite capacity Petri net in Fig. 8. In this figure, transitions T_1 and T_3 are enabled, while T_0 is not enabled (although it appears as a source transition) because place P_1 has reached its maximum capacity.*

The transformation of a finite capacity PN into an ordinary PN is quite simple. If place P_i has a finite capacity $\mathrm{Cap}\,(P_i)$, a complementary place is added to P_i, known as place P'_i, whose marking is also

* A Petri net which is both generalized and finite capacity, according to the terminology used in this paper, was called a place/transition net (or P/T net) in Petri's original terminology. The most widely studied net is the subclass of P/T nets which is neither generalized nor finite capacity. In many papers this subclass (here known as ordinary Petri nets) has been called place/transition net, or P/T net, or just Petri net.

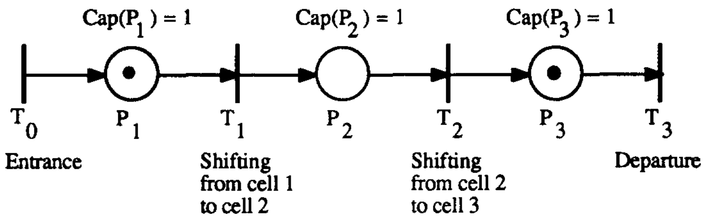


FIG. 8. Finite capacity Petri net.

complementary to the capacity of P_i. That is to say $M(P_i') = \text{Cap}(P_i) - M(P_i)$.

3.4. *Inhibitor arc Petri net* (*extension*)

An inhibitor arc is a directed arc which leaves a place P_i to reach a transition T_j. Its end is marked by a small circle as shown in Fig. 9. The inhibitor arc between P_2 and T_4 means that transition T_4 is only enabled if place P_2 does not contain any tokens. Firing consists in taking away a token from each input place of T_4, with the exception of P_2, and in adding a token to each output place of T_4. The expressions zero test and extended PNs are used by some authors.

The following example is illustrated in Fig. 9(a). An administration lets customers in (their number is not bounded) and then closes the entrance door before starting work. Once they have been served, the customers leave through another door. The entrance

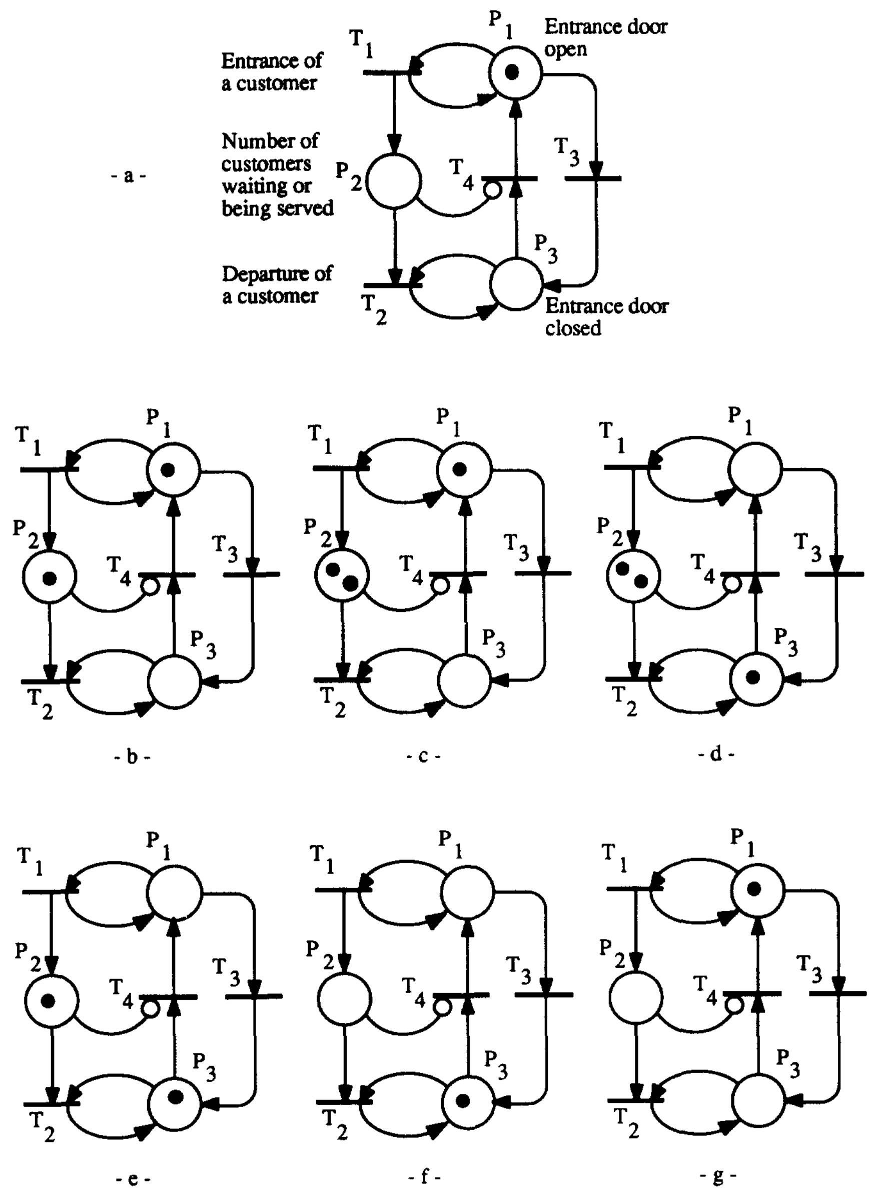


FIG. 9. Inhibitor arc Petri net.

door will only be opened again when all the customers who came in have gone out. The number of tokens in place P_2 represents the number of customers who have come in but not yet gone out, and places P_1 and P_3 represent the state entrance door open and entrance door closed, respectively. We move from one state to the other by the firing of transitions T_3 and T_4.

In Fig. 9(a), transitions T_1 and T_3 are enabled. Firing of transition T_1 corresponds to the entrance of a customer, and leads to the marking in Fig. 9(b). This firing adds a token in place P_2, while place P_1 does not change its marking. Since P_1 is the input and output place of transition T_1, the added token compensates the token taken away. This operation is called reading of place P_1 (*i.e.* firing of T_1 is conditioned by the marking of P_1, but does not affect this marking).

In Fig. 9(b), transitions T_1 and T_3 are still enabled. Firing of transition T_1 leads to the marking in Fig. 9(c) and T_1 and T_3 are still enabled. Assume now that transition T_3 is fired: the marking in Fig. 9(d) is obtained. This means that the administration has closed the entrance door after two customers have entered, and that the service may begin, *i.e.* transition T_2 is enabled.

In Fig. 9(d), only transition T_2 is enabled. As a matter of fact, enabling of T_4 requires a token in P_3 (this is verified), and zero token in P_2 (this is not verified). When T_2 is fired, the marking in Fig. 9(e) is obtained (a customer has been served, and has left), for which only transition T_2 is enabled again. When it is fired, the marking in Fig. 9(f) is obtained.

In Fig. 9(f), transition T_4 is enabled. Both enabling conditions are verified, since there is no token left in P_2. The marking $M(P_2)=0$ means that all the customers who came in have gone out. Firing of transition T_4 corresponds to opening the entrance door. It leads to the marking in Fig. 9(a).

This dynamic system cannot be represented by an ordinary PN, because the number of customers who may come in is not bounded. In the general case, inhibitor arc PNs cannot be transformed into ordinary PNs (inhibitor arc PNs have the computational power of Turing machines). However, if an inhibitor arc PN is bounded, it can be transformed into an ordinary PN.

3.5. *Priority Petri net* (*extension*)

Such a net is used when we wish to make a choice between a number of enabled transitions. It is made up of a Petri net and a partial order relation on the net transitions.For example, a priority PN is obtained if the PN of Fig. 4 is considered and we indicate in addition that transition T_1 has priority over transition T_4. This means that if the marking in Fig. 4(a) is reached, T_1 must be fired.

Priority PNs cannot be transformed into ordinary PNs. They have the computational power of Turing machines. It follows that all priority PNs can be modeled by inhibitor arc PNs, for example.

3.6. *Continuous Petri net* (*extension*)

Their distinguishing feature is that the marking of a place is a real (positive) number and no longer an integer. Firing is carried out like a continuous flow. These nets enable systems to be modeled which cannot be modeled by ordinary PNs, and a suitably close model to be obtained when the number of markings of an ordinary PN becomes too large. The model considered here is an autonomous model, time is not involved. In Section 4.6, timed continuous models will be defined.

An example is given in Fig. 10. A place is represented by a double circle, and a transition by a box (this representation is not really useful here, but it is useful when hybrid PNs are concerned (Section 3.7). The French dressing is obtained by mixing salad oil with vinegar, in the ratio of two. Figure 10(b) represents an initial state where there is 1 l of salad

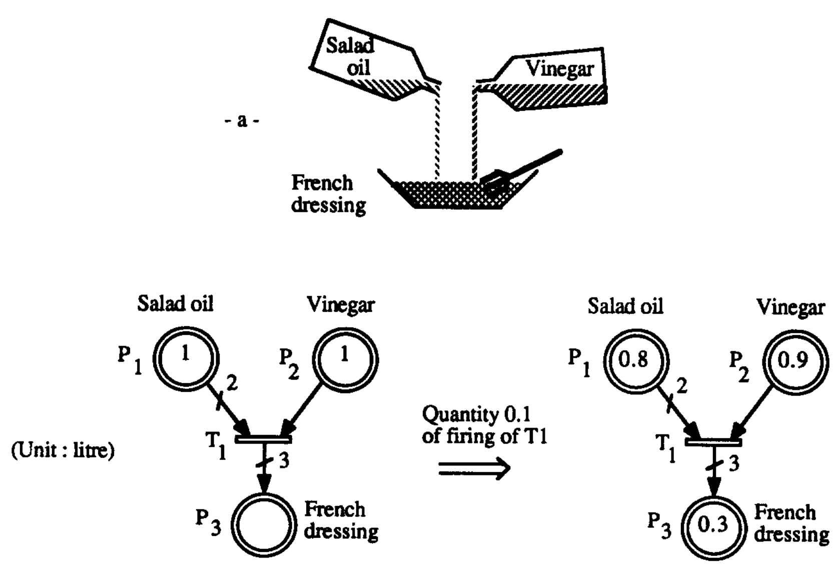


FIG. 10. Continuous Petri net.

oil, and 1 l of vinegar, and no French dressing. Firing of transition T_1 corresponds to the mixing in the appropriate ratio.

Transition T_1 is enabled if $m_1 > 0$, and $m_2 > 0$. In an ordinary PN, one can have one firing, then another firing. In a continuous PN, one can have a quantity of firing which is not an integer (David and Alla, 1990). For example, if the quantity of firing is $x = 0.1$, the marking in Fig. 10(c) is obtained from the marking in Fig. 10(b). The quantity $2x = 0.2$ marks are taken out of place P_1 (since the weight of arc $P_1 \to T_1$ is 2), the quantity $x = 0.1$ marks are taken out of place P_2, the quantity $3x = 0.3$ marks are added to place P_3 (since the weight of arc $T_1 \to P_3$ is 3). The quantity of firing may be any real number x such that $x \leq m_1/2$ (because the weight of arc $P_1 \to T_1$ is 2), and $x \leq m_2$. This quantity may be infinitely small.

One can observe that there are two marking invariants for the continuous PN in Fig. 10(b), namely $m_1/2 + m_3/3 = 0.5$, and $m_2 + m_3/3 = 1$. From the initial marking, there is an infinity of reachable markings: all the markings fulfilling the two marking invariants, and such that every $m_i \geq 0$. Note that the marking $\mathbf{M} = (0, 0.5, 1.5)$ is a deadlock (there is no salad oil left).

3.7. *Hybrid Petri net (extension)*

This is a new model (Le Bail *et al.*, 1991). Such a net contains both discrete places and transitions, and continuous places and transitions.

An example is presented in Fig. 11. Machine A produces coated copper wire, from uncovered copper wire and plastic. The behavior of this machine may be modeled by the continuous PN made of places P_3, P_4 and P_5, and transition T_3 in Fig. 11(b). Now, if machine A breaks down the production is stopped, *i.e.* transition T_3 can no longer be fired. This is modeled by the hybrid PN in Fig. 11(b), in which places P_1 and P_2, and transitions T_1 and T_2, are discrete ones.

In Fig. 11(b), transition T_3 is enabled if $m_3 > 0$, $m_4 > 0$, and there is a token in P_1. Consider a quantity of firing x of this transition: markings of P_3, P_4 and P_5 are modified according to the corresponding weights; and marking of P_1 remains unchanged since there are arcs $P_1 \to T_3$ and $T_3 \to P_1$, with the same weight (reading of P_1).

If machine A breaks down, transition T_2 is fired (note that this implies a priority of T_2 over T_3). There is now a token in P_2, but no token in P_1. Then transition T_3 is no longer enabled.

3.8. *Comments on applications*

Modeling. Modeling by autonomous PNs can be applied to various kinds of systems belonging to the class of discrete events (dynamic) systems (DEDS). A PN application field is communication protocols in computer systems; since concurrency, synchronization and resource sharing can be found in the specification of such systems, PN is a well suited modeling tool (Berthelot and Terrat, 1982; Murata, 1989). Another field of application which became important during the last decade is manufacturing systems (Silva and Valette, 1990); concurrency (two machines working independently), synchronization (a machine is free and a part is ready to be processed by it) and resource sharing (a robot is affected to handling parts for two machines but cannot serve both machines at the same time) are also usual features of these systems.

Up to now continuous PNs have been essentially used as approximations of discrete event systems when the number of reachable markings is a great

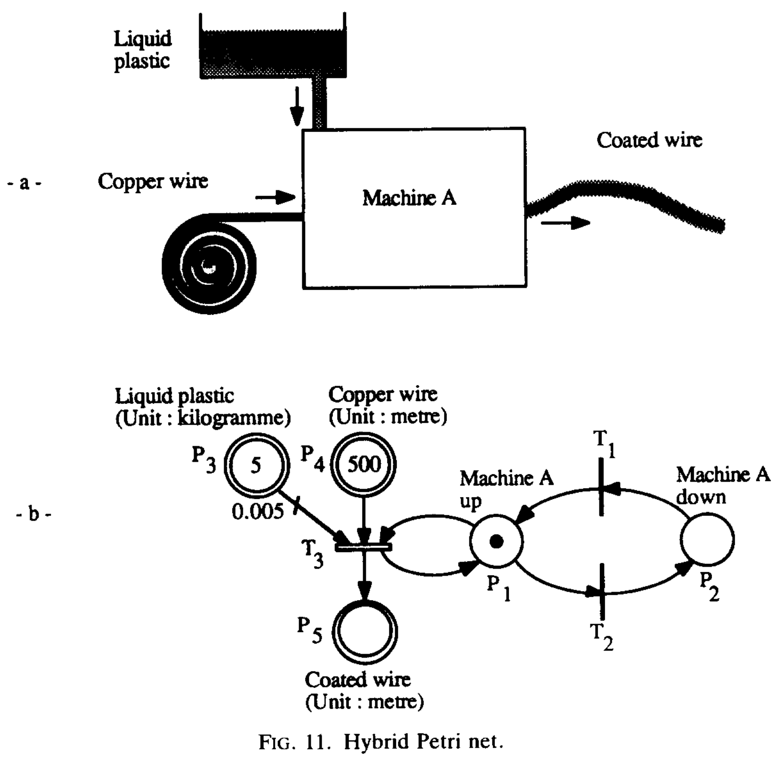


FIG. 11. Hybrid Petri net.

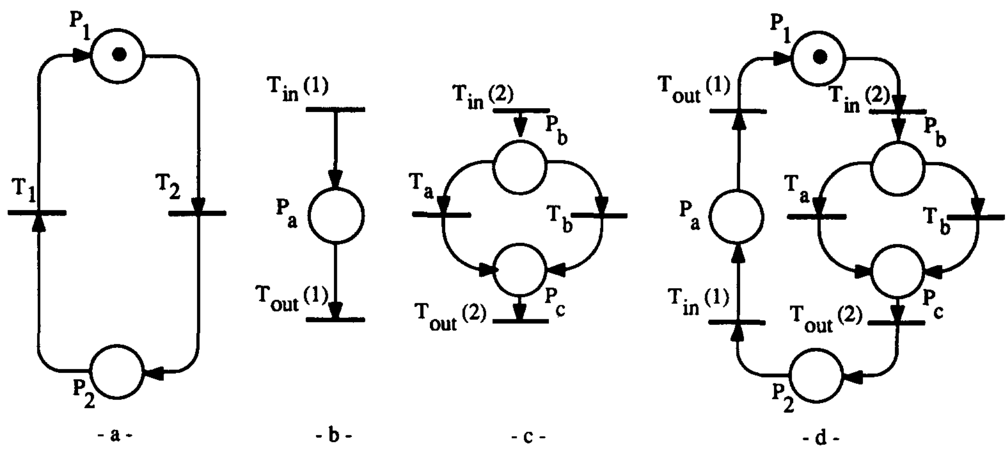


FIG. 12. Modeling by refinement mechanism.

number: for example the parts processed by a machine is modeled as a continuous flow. The model is fairly new and models of continuous systems could be developed (mixing of liquids for example). However the most interesting applications could be found in hybrid systems.

Some systems are hybrid when they are composed of a continuous part and a discrete part. More precisely some components of the state of the system evolve as continuous functions of the time while other components evolve in a discrete way on occurrence of discrete events. This is the case, for example, of the batch production process in the biotechnological industry. Some manufacturing systems can also be considered as hybrid systems, when the flow of parts is approximated by a continuous flow and the state of the resources is modeled in a discrete way. These systems can advantageously be modeled by hybrid PNs.

In the case of real life systems, it is often necessary to use progressive modeling. This can be performed by means of stepwise refinements (Valette, 1979; Zhou and DiCesare, 1993). The refinement mechanism allows the construction of hierarchical structured models. The developed model can be performed relating either to transitions or to places. Figure 12 gives an example of a developed model according to transitions. The approach consists in roughly describing the behavior of the system [Fig. 12(a)] and then in replacing the transitions by PN parts [Fig. 12(d) is obtained from Fig. 12(a) and the PN parts in Fig. 12(b) and (c)]. In Section 4.4, the development according to places will be illustrated by the concept of macrosteps in a grafcet, which is similar.

The first usefulness of PNs is to exhibit how a system works, in an unambiguous way. The paper (Di Mascolo *et al.*, 1991), where kanban systems described by various authors are compared thanks to Petri nets models, provides an illustration of this usefulness. When a model has been obtained, a qualitative analysis allows the basic properties of the model, and thus of the described system to be found.

Seeking properties. When the autonomous model has been obtained, a qualitative analysis may be performed. This analysis consists in searching for the properties of the constructed model, for example liveness, boundedness, deadlock, and so on (Jantzen and Valk, 1979). It is then possible to show that the specifications are fulfilled. There are three main categories of methods for seeking these properties of a PN.

The basic method consists in drawing up the graph of markings or coverability tree (Karp and Miller, 1969). The graph of markings is made up of nodes which correspond to the reachable markings and of arcs corresponding to the firing of transitions resulting in the passing from one marking to another.

The graph of markings for the PN in Fig. 2 is shown in Fig. 13. From $\mathbf{M}_0$ only T_1 can be fired and its firing leads to $\mathbf{M}_1$. From $\mathbf{M}_1$ either T_2 or T_3 can be fired. If T_2 is fired then $\mathbf{M}_2$ is reached, and if T_3 is fired then $\mathbf{M}_3$ is reached. All the firings from these new markings are then considered. From $\mathbf{M}_2$ only T_3 can be fired and its firing leads to $\mathbf{M}_0$. From $\mathbf{M}_3$ only T_2 can be fired and its firing leads to $\mathbf{M}_0$. No new marking has been obtained, then the graph of markings is complete. On the graph in Fig. 13, one can observe that there is no deadlock, the PN is live, it is bounded and even safe. It is also clear that $T_1T_2T_3$ and $T_1T_3T_2$ are repetitive sequences, and so on.

When the PN is not bounded, the graph of markings cannot be drawn up since the number of reachable markings is infinite. In that case a coverability tree can be drawn up. In this tree the

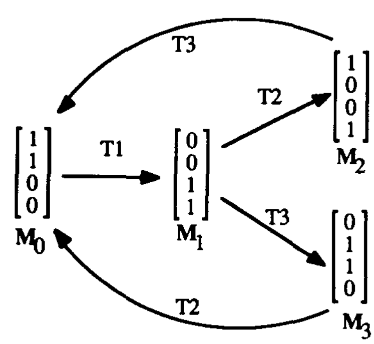


FIG. 13. Graph of markings for the Petri net in Fig. 2.

'marking' (0, 1, ω, 1), for example, represents an infinite set of markings such that $m_1 = 0$, $m_2 = m_4 = 1$, and $m_3 = \omega$ means that m_3 can reach any arbitrarily high value. The number of nodes in a coverability tree is always finite. Deadlock freeness, liveness and boundedness can be found from the coverability tree, but the reachability problem (*i.e.* given any marking, is it reachable?) and the liveness problem cannot be solved by the coverability tree alone. However these properties are decidable (Reutenauer, 1990).

The second class of methods is based on linear algebra. These results are powerful and elegant. They are derived from the fundamental equation (1) and properties given in Section 2.6, and are particularly interesting for obtaining the invariants (Memmi, 1983; Colom and Silva, 1989).

The third class of methods consists in reductions of the PNs (Berthelot and Terrat, 1982). Although the reductions do not provide equivalent PNs, they enable certain properties to be preserved, resulting in an easier analysis by the above methods.

Numerous commercial softwares have been developed with the aim of proving the properties of a PN. These softwares, for example Design/CPN (Meta, 1990) and Eval (Verilog, 1991) are generally made of two parts, namely the editing of the model and the actual analysis. The model can be edited either in textual or graphic form. Colored PNs are often used in order to obtain compact models.

4. MODELING BY NON-AUTONOMOUS PNS

In Section 3, we presented various autonomous PNs which permit a qualitative approach. In this section, we shall present extensions of Petri nets which make it possible to describe not only what 'happens' but also when 'it happens'. These Petri nets will enable systems to be modeled whose firings are synchronized on external events, and/or whose evolutions are time dependent.

4.1. *Synchronized Petri net*

Consider the PN in Fig. 14, representing the states of a motor for example. This is a synchronized Petri net, because the firings of transitions are synchronized on external events (the external events corresponds to a change in state of the external world) (Moalla *et al.*, 1978).

In an autonomous PN, we know that a transition may be fired if it is enabled, but we do not know when it will be fired. In a synchronized Petri net, an event is associated with each transition, and the firing of this transition will occur:

if the transition is enabled,

when the associated event occurs.

In Fig. 14(a), the external event E_1 (start-up order) is associated with transition T_1. For the marking in this figure, $\mathbf{M}_0$, transition T_1 is said to be receptive to event E_1, because it is enabled. It will become firable when event E_1 occurs, and it will be fired immediately (see the corresponding timing diagram).

For the marking $\mathbf{M}_0$, transition T_2 is not enabled. Then it is not receptive to event E_2. Since there is no transition receptive to event E_2, the synchronized PN is not receptive to event E_2. This means that the marking does not change if this event occurs [see Fig. 14(b)]. For the marking $\mathbf{M}_1 = (0, 1)$, the synchronized PN is receptive to event E_2.

In a synchronized PN, a transition is synchronized either on an external event like E_1, or on the 'always occurring event' noted as e. This is the neutral element of the monoid $(E_1 + \cdots + E_p)^*$, where $\{E_1, \ldots, E_p\}$ is the set of external events. In other

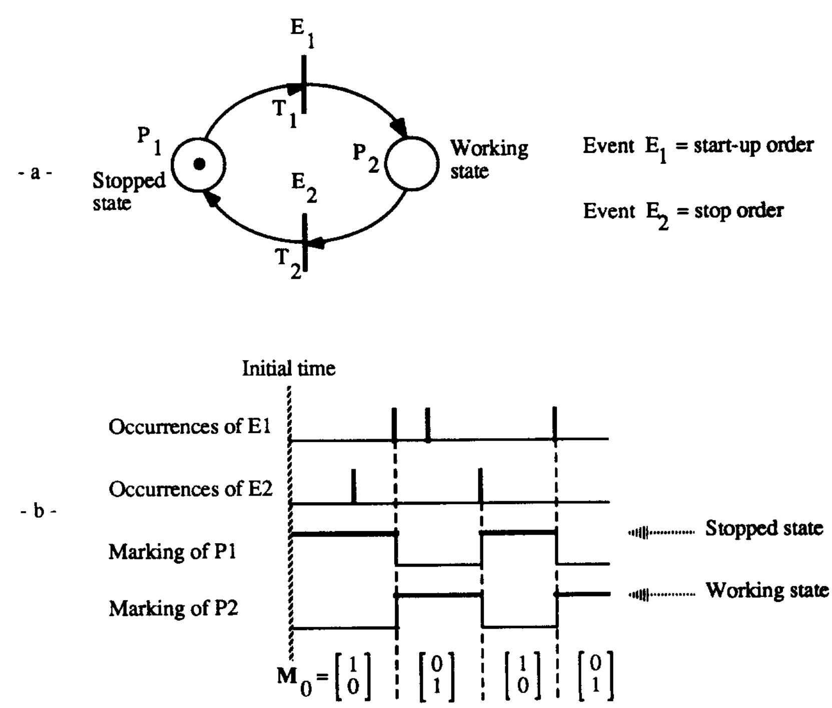


FIG. 14. Synchronized Petri net.

words, e corresponds to a sequence of external events whose length (number of these events) is zero. If a transition is synchronized on event e, it is fired as soon as it is enabled. This is illustrated in Section 4.3.

It is assumed that two external events never occur simultaneously.

4.2. *Timed Petri net*

A timed Petri net enables a system to be described whose functioning is time dependent (Ramchandani, 1973). For example, a certain time may elapse between the start and the end of an operation. If a mark in a certain place indicates that this operation is in progress, a timed PN enables this time to be taken into account. Timed PNs are useful for evaluating the performances of a system. Carlier and Chrétienne (1983) used them for modeling scheduling problems. Hillion and Proth (1989) studied the peformances of job-shop systems. There are two main methods for modeling timing: either the timings are associated with the places (the PN is said to be P-timed), or the timings are associated with the transitions (the PN is said to be T-timed).

4.2.1. *P-timed Petri net.* A timing d_i, possibly of zero value, is associated with each place P_i (Sifakis, 1977). We shall consider the case where d_i is a constant value, but in a general case d_i could be variable.

When a token is deposited in place P_i, this token must remain in this place at least for a time d_i. This token is said to be unavailable for this time. When the time d_i has elapsed, the token then becomes available. Only available tokens are considered for enabling conditions. In most applications, functioning at maximal speed is considered. This means that a transition is fired as soon as it is enabled (except possibly in the event of conflict involving this transition). This is illustrated in Figs 15 and 16.

We consider that there are two pallets (each one carrying a part) which pass in turn through machines A_1 and A_2 [Fig. 12(a)]. Machine A_1 can only treat one part at a time and its service time is 5 time units. Machine A_2 can treat two parts at a time and its service time for a part is 8 time units. The initial state is such that both pallets are in buffer B_1.

A P-timed PN corresponding to this system is presented in Fig. 15(b). Since there are only two parts and machine A_2 is able to serve them both at the same time, there are never any parts waiting in buffer B_2. There is no need to associate a place at buffer B_2. Places P_1, P_2 and P_3 correspond to the parts in buffer B_1 on machine A_1 and on A_2 respectively. A place P_2' complementary of P_2 ensures that there is only one part on machine A_1. The timings $d_2 = 5$ and $d_3 = 8$ are associated with the places P_2 and P_3. The other places have a zero timing.

Figure 15(b) is the initial state $\mathbf{M}_0$, at $t = 0$. The evolution of this PN between $t = 0$ and $t = 13$ is shown in Fig. 16. For $\mathbf{M}_0$, T_1 is firable, but only once. The marking in Fig. 16(a) is reached. The token which has been deposited in P_2 is unavailable for 5 time units, since $d_2 = 5$. One available token remains in P_1, and no transition is enabled. Five time units later, the token in P_2 becomes available [Fig. 16(b)]. Then T_2 is fired [Fig. 15(c)], and transition T_1, which becomes enabled again, is also fired. The marking in Fig. 15(d) is obtained. Five time units later, the token in P_2 becomes available [Fig. 15(e)], and T_2 is fired again [Fig. 15(f)]. And so on.

4.2.2. *T-timed Petri net.* A timing d_j, possibly of zero value, is associated with each transition T_j (Ramchandani; 1973, Chretienne, 1983). The preced-

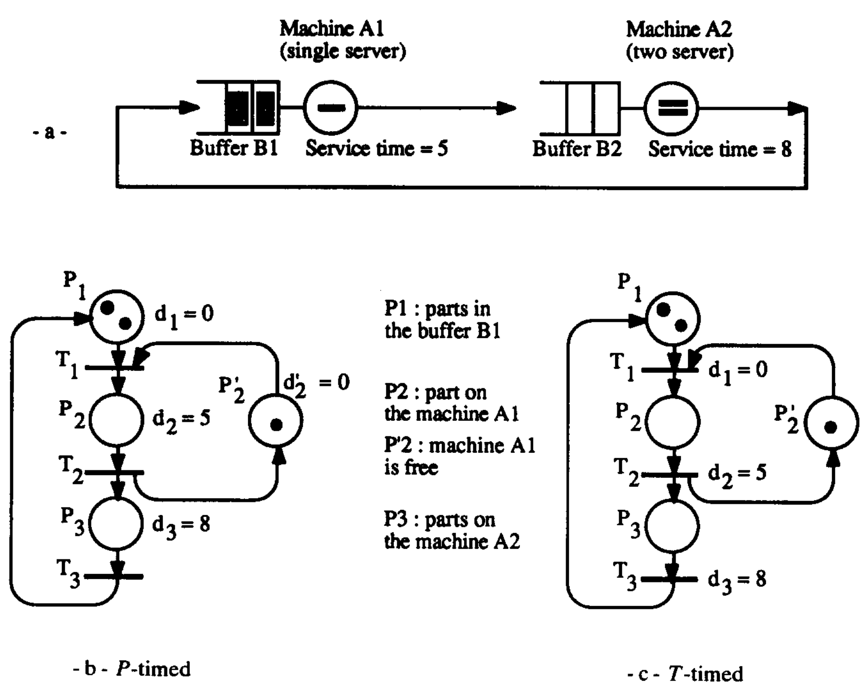


FIG. 15. Timed Petri nets.

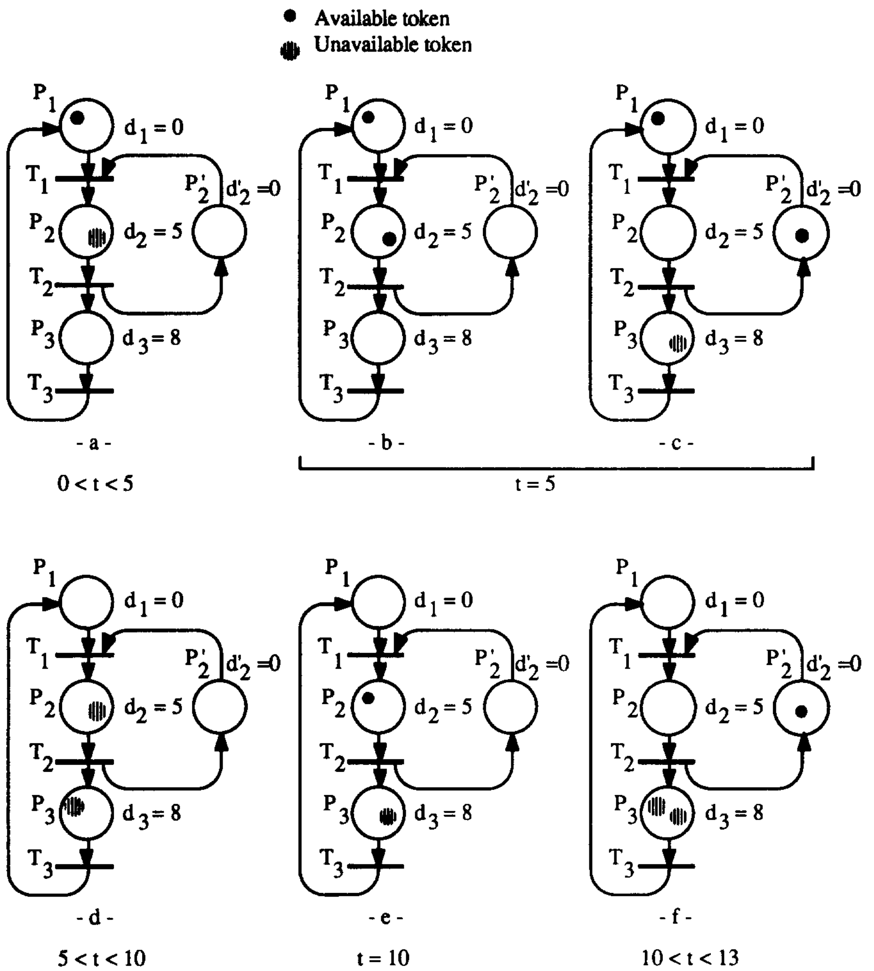


Fig. 16. Marking of a P-timed Petri net.

ing example is modeled by the T-timed PN in Fig. 15(c).

A token can have two states: it can be reserved for the firing of a transition T_j or it can be non-reserved. Only non-reserved tokens are considered for enabling conditions.

In Fig. 15(c), transition T_1 is enabled because all the tokens are non-reserved at the initial time. After this firing, there is one non-reserved token left in P_2, and 1 non-reserved token left in P_1. The transition T_2 is enabled. The token in place P_2 is then reserved for firing of T_2, and this firing will occur five time units later, since $d_2 = 5$.

Depending on the system to be modeled, one of the models (P-timed or T-timed) may be easier to use than the other one. However it is always possible to pass from a P-timed PN to a T-timed PN, and *vice versa* [see Fig. 15(b) and (c), but in the general case the unmarked PN is not the same in both cases].

4.3. *Interpreted Petri nets*

The expression 'interpreted Petri nets' can be applied to various interpretations according to the use wished to be made of it. Interpretations are found adapted to the description of software, hardware, logic controllers, to formal languages and to performance evaluation. The interpreted Petri net model which we shall present here allows modeling of logic controllers and real-time systems (Moalla, 1985).

An interpreted PN exhibits the following three characteristics:

(1) It is synchronized

(2) It is P-timed

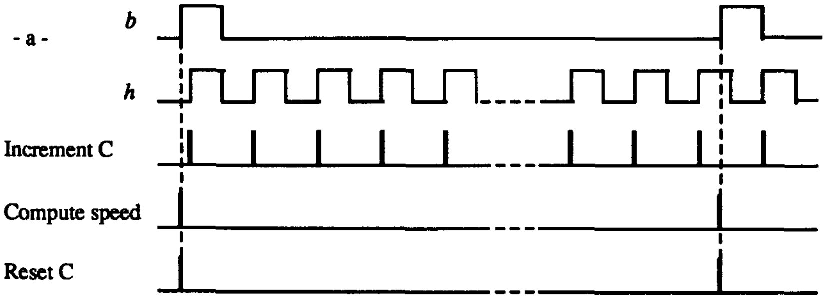
(3) It comprises a data processing part whose state is defined by a set of variables $V = \{V_1, V_2, \ldots\}$. This state is modified by operations $O = \{O_1, O_2, \ldots\}$ which are associated with the places. It determines the value of the conditions (predicates) $C = \{C_1, C_2, \ldots\}$ which are associated with the transitions.

In an interpreted PN, a transition T_j will be fired:

if transition T_j is enabled
and *if* condition C_j is true,
when event E_j occurs.

If transition T_j is enabled and if condition C_j is true, transition T_j is said to be firable on occurrence of E_j. If a token is deposited in a place P_i at instant t, the operation O_i is carried out and the token is unavailable for d_i.

Let us introduce the basic notions through the example shown in Fig. 17. A train passes in front of points B which are equidistant. A signal $b = 1$ is given off when the train passes in front of a point. The number of periods of a clock h is counted between two points, and the speed of the train is calculated



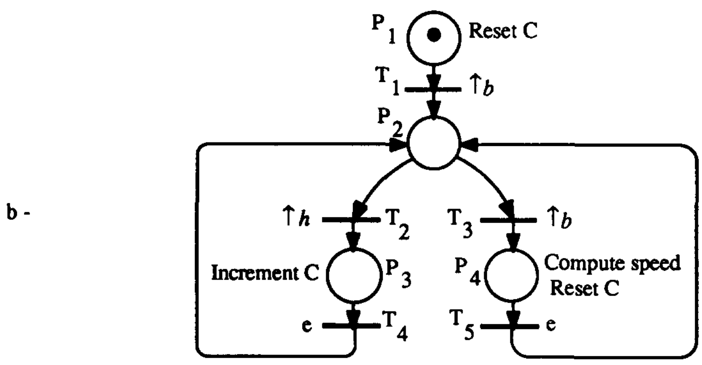


FIG. 17. Interpreted Petri net.

from this number. This functioning is illustrated by the timing diagram in Fig. 17(a).

The interpreted PN in Fig. 17(b) corresponds to the specification. At initial time, there is a token in P_1. Then the operation reset C which is associated with P_1 is executed. There is no timing associated with P_1, then the token in P_1 is available (the timing is zero when it is not explicitly mentioned; in this example the timing is zero for every place). Transition T_1 is enabled.

Transition T_1 is fired when the event $\uparrow b$ occurs (this event means the rising edge of the Boolean variable b). As a matter of fact, there is no condition associated with transition T_1 (the condition is always verified when it is not explicitly mentioned; in this example this is true for every transition). After firing of T_1 there is a token in P_2 (since there is no timing associated with places, in this example, all the tokens are always available). Transitions T_2 and T_3 are enabled.

When the event $\uparrow h$ occurs, transition T_2 is fired, and a token is put into P_3. The operation increment C which is associated with P_3 is executed. Transition T_4 is enabled. Since the event associated with T_4 is the 'always occurring event', e, this transition is immediately fired, and the token is again in place P_2.

One can observe in this example that the marking $\mathbf{M}_1 = (0, 1, 0, 0)$ is stable: it lasts as long as one of the external events $\uparrow b$ or $\uparrow h$ occurs. On the other hand, the marking $\mathbf{M}_2 = (0, 0, 1, 0)$ is unstable: as soon as it is reached, transition T_4 is fired. However, the operation associated with P_3 must be performed, even if the marking of this place is transient.

Then, once transition T_1 has been fired, the circuit $T_2P_3T_4P_2$ counts the edges $\uparrow h$. When the train passes in front of a point B, we have $\uparrow b$ and thus changeover to $\mathbf{M}_3 = (0, 0, 0, 1)$, where the speed is calculated and the impulse counter reset.

Since the external events ($\uparrow b$ and $\uparrow h$) do not occur simultaneously, and the circuits $T_2P_3T_4P_2$ and $T_3P_4T_5P_2$ have no duration, no event $\uparrow b$ or $\uparrow h$ is 'lost', according to the model in Fig. 17(b). In the event of 'apparent simultaneity' of $\uparrow h$ and $\uparrow b$ (the two events may sometimes be so close together that there is no technological means to distinguish which took place first), priority must on all accounts be given to $\uparrow b$ in order to avoid a large error being made. For this, we can add the condition b' to transition T_2. Thus if $\uparrow h$ and $\uparrow b$ occur almost simultaneously, only transition T_3 is fired. In Silva and Velilla (1982) the implementation of interpreted PNs on programmed logic controllers is studied.

4.4. *Grafcet*

Grafcet is a model which has been defined in order to model logic controllers. This model is very close to Interpreted Petri Net, with slight differences. It was defined by one of the work groups making up AFCET (Association Française pour la Cybernétique Economique et Technique) in 1975–1977. Firstly a French standard, in 1987 it became an international one (Publication 848 of the International Electrotechnical Commission entitled Etablissement des diagrammes fonctionnels pour systèmes de commande or Preparation of function charts for control systems).

A grafcet is made of steps (represented by squares) and of transitions (represented by bars) joined by direct links. A step may have two states: it may either

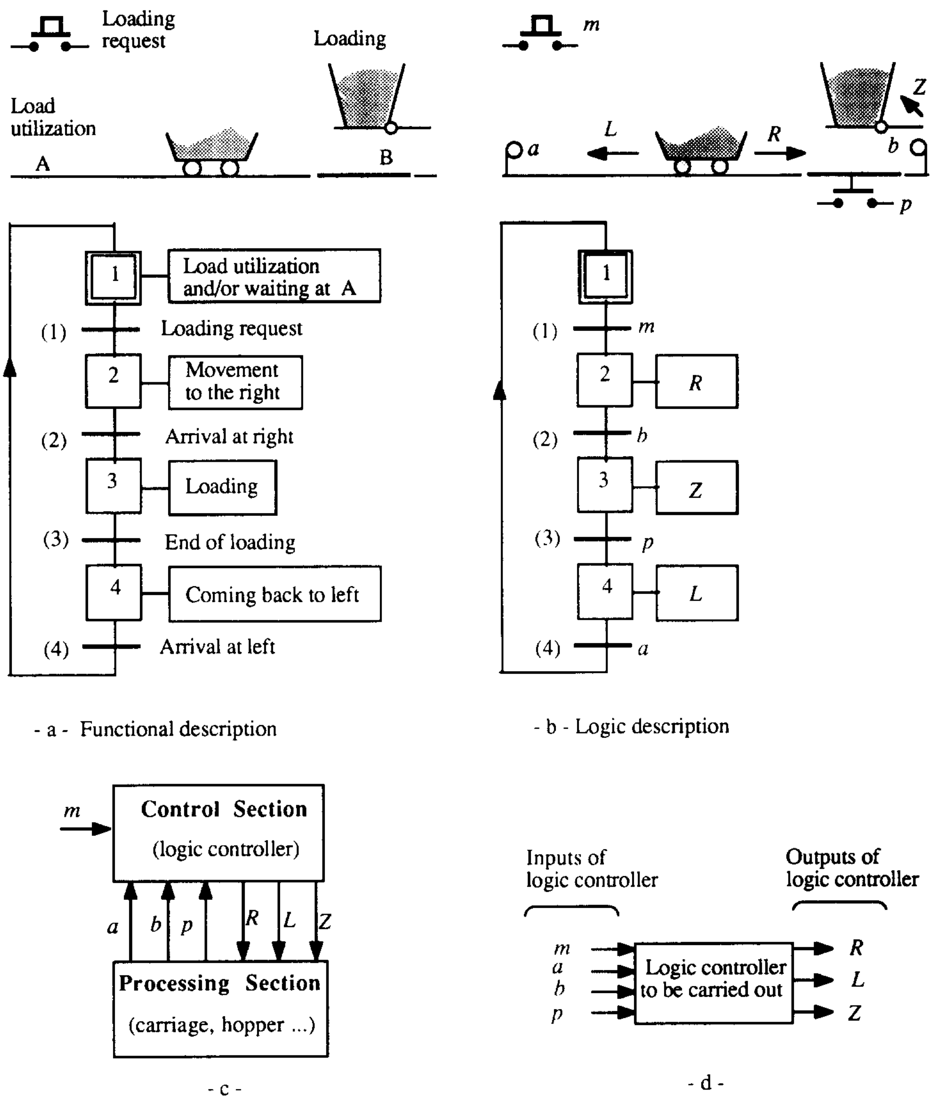


FIG. 18. First example of grafcet: loading of a truck.

be active (this is represented by a token in the step) or inactive. Actions are associated with the steps, these being the outputs of the grafcet. A receptivity, which is a function of the input variables and possibly of the internal state is associated with each transition.

A transition is firable if and only if both the following conditions are met.

(1) All the steps preceding the transition are active (the transition is said to be enabled).

(2) The receptivity of the transition is true.

It is clear that steps, transitions and directed links in a grafcet, look like places, transitions and arcs in a Petri net. A basic difference is that the marking of a grafcet is Boolean (a step is active or inactive) while the marking of a PN is numerical.*

In order to introduce the basic notions of grafcet, let us take a first example (see Fig. 18). A truck may move between points A and B. At A, an operator may ask for the truck to be loaded. The truck proceeds up to point B. Upon arrival, it is loaded by opening a hopper. When loading is complete, the hopper is closed and the truck returns to A where its load is made use of. It will set off again when the operator asks for a fresh loading. In the initial state, the truck is in the standby position at point A. The grafcet defining this functioning are represented in Fig. 18. The 'functional' grafcet is given in Fig. 18(a).

The initial state (represented by a double square) corresponds to the situation where step 1 is active. In this situation, there is no action. Transition (1) is thus the only one enabled and is receptive to the loading request. When the loading request receptivity becomes true, transition (1) becomes firable and is immediately fired. This firing corresponds to the inactivation of step 1 and the activation of step 2. In this new situation where step 2 is active, the action movement to the right is carried out, *i.e.* the movement to the right lasts as long as step 2 remains active. It is the event arrival at right which will end this movement by causing transition (2) to fire. At this point step 3 will become active and the loading action will begin. And so on.

* Nets with Boolean markings have also been defined by C. A. Petri: condition/event nets, or C/E nets.

In the functional description given in Fig. 18(a), the receptivities and actions are indicated using everyday language. The user is free to choose the expressions or symbols which suit him. On the other hand, he must imperatively associate quantities which may cause a change in state with the transitions, and the resulting actions with the steps. The grafcet shown in Fig. 18(a) represents the desired functioning of the system, but is not yet, however, the description of a logic controller. Indeed the entire setup could be envisaged as manual. It could be thought that a worker uses the loading at A. Then when the truck is empty, he pushes it up to B, opens the hopper, *etc.* We have thus described the desired behavior quite independently of the fact that this behavior is ensured or not by a logic controller. In order to describe a logic controller in charge of the truck movements and opening the hopper, we shall associate Boolean variables to the system inputs and outputs. Let m be the Boolean quantity associated with a loading request, a and b the variables associated with the presence of the truck at points A and B respectively, and p a variable of value 1 when the truck is filled. These are the inputs of the logic controller to be described. The outputs are as follows: $R = 1$ when there is movement to the right, $L = 1$ when the truck moves to the left, and $Z = 1$ when the hopper is opened. Using these specifications, the 'logic' grafcet of Fig. 18(b) can be described, from which the logic controller can be implemented. Action R associated with step 2, for example, means that $R = 1$ for all the time that this step is active. The complete system may be broken down into two sections as shown in Fig. 18(c): the processing section (made up of the process to be controlled: truck, hopper) and the control section (logic controller to be implemented, described by a grafcet). The logic controller inputs are m, a, b and p, the first one coming from outside the system and the three others being given by sensors placed on the processing section. The logic controller outputs are R, L and Z which activate the processing section (there may also be outputs to the world outside the system described). Comparison of Fig. 18(b) and (d) clearly shows that the logic controller inputs are associated with the transitions, whereas its outputs are associated with the steps.

Figure 19 presents a second example. There are two graphical representations for a grafcet. Figure 19(a) corresponds to the standardized representation, while Fig. 19(b) gives the original representation which is similar to PN representation. The logic controller inputs and outputs of this example are shown in Fig. 19(c).

On the example of Fig. 19, one can observe that, like in an interpreted PN, there are receptivities which are conditions [Boolean value a for transition (3)], or events [rising edge of the Boolean value m for transition (1)], or both [condition b and event $\uparrow a$ for transition (2)]. Now there are two kinds of actions which are associated with places; level actions A and B which last as long as the corresponding steps are active, and impulse actions C^* which are executed as soon as step 3 changes from the inactive state to the active state (the asterisk is used to show that it is an impulse action). This example also shows that the standardized representation is slightly different from the original one: a step is represented by a square instead of a circle, but in particular a transition has a different representation when it is an input of several places, or/and output of several places.

There are many other things to say about Grafcet, but there is not enough place in this paper. Let us only say that the firing rule is different from a Petri net when there is a conflict. However, in practice, most of the grafcets have behavior which is similar to the behavior of a Petri net. It may be useful to build a

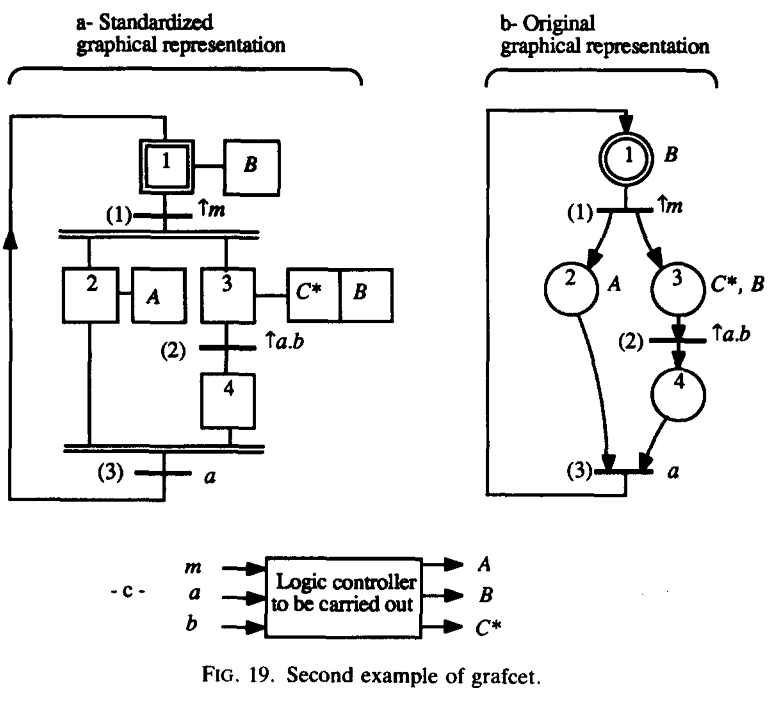


FIG. 19. Second example of grafcet.

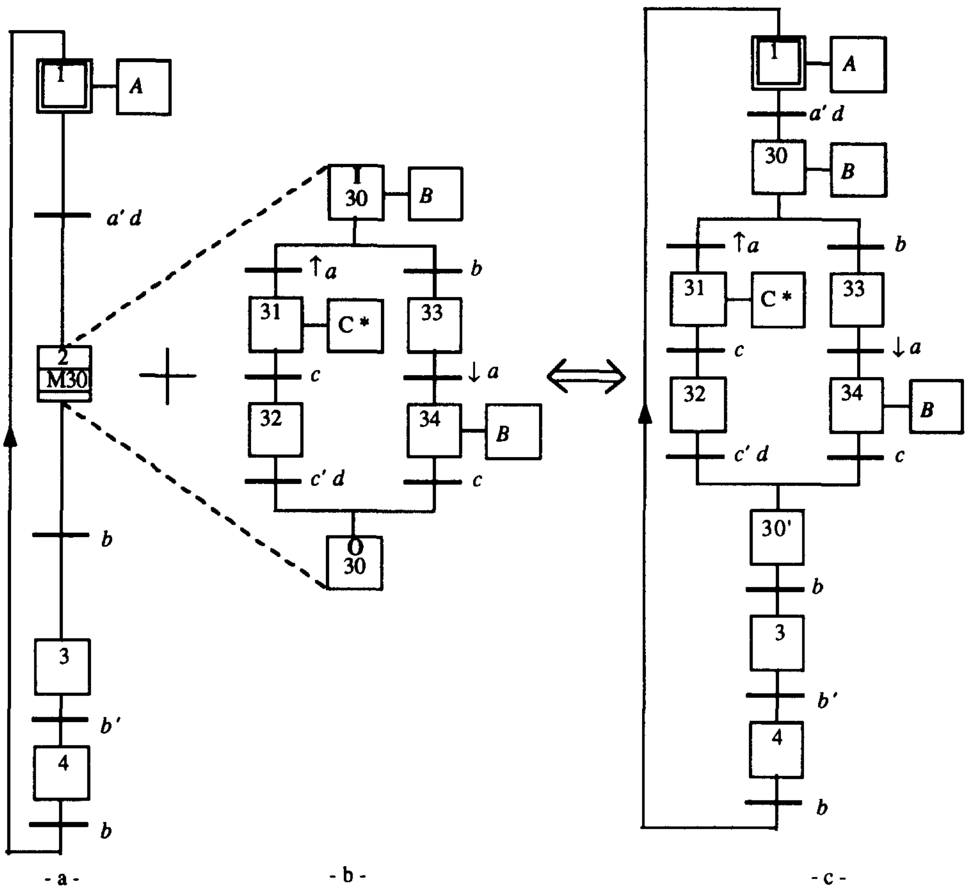


Fig. 20. Macrostep. (a) Grafcet with macrostep; (b) macrostep expansion; (c) equivalent grafcet without macrostep.

grafcet in a hierarchical manner. Let us comment on the concepts of macrosteps and macroactions (David and Alla, 1992).

Macrosteps. The aim of the macrostep concept is to facilitate the description of complex systems. The macrostep makes it possible to lighten the graphical representation of a grafcet by detailing certain parts separately.

The concept of the macrostep is illustrated in Fig. 20. A macrostep is represented by a square divided into three parts by two horizontal lines. A macrostep given as 2/M30 is represented in Fig. 20(a). It represents a grafcet which is detailed elsewhere and which is known as a macrostep expansion. Figure 20(b) is the expansion corresponding to M30. If this macrostep expansion is used to replace macrostep 2/M30, the grafcet of Fig. 20(c) is obtained. The complete set of Fig. 20 (a and b) (grafcet with a macrostep, plus expansion of the macrostep) is strictly equivalent to Fig. 20(c) in which the macrostep has been replaced by its expansion.

A macrostep and its expansion satisfy the following rules:

(1) A macrostep expansion has only one input step (written as **I**) and one output step (written as **O**).
(2) All firings of a transition upstream of the macrostep activate the input step of its expansion.
(3) The output step of the macrostep expansion participates in the enabling of the downstream transitions, in accordance with the structure of the grafcet containing this macrostep.
(4) No directed links either join or leave the macrostep expansion.

Let us now return to Fig. 20 to give further details on certain points. The grafcet of Fig. 20(a) contains a macrostep which is written as 2/M30. Number 2 is the macrostep number, *i.e.* it indicates its position in the grafcet. Symbol M30 relates to the macrostep expansion. Number 30 is found in the expansion to mark the input step, **I**30, and the output step, **O**30, of expansion M30. When the grafcet part corresponding to expansion M30 has been used to replace macrostep 2 [*i.e.* in Fig. 20(c)], the symbols M, **I** and **O** have disappeared. We now have an ordinary grafcet in which every step has a number. During replacement, steps **I**30 and **O**30 have become 30 and 30′ (any other numbering would have been possible, provided that no two grafcet steps have the same number).

In general, a grafcet may contain several macrosteps. Eventually two or more macrosteps may have the same expansion, for example 2/M40 and 5/M40 (David and Alla, 1992).

Macroactions. When describing complex systems, the size of the grafcets may increase so that they become difficult to work out and thus to understand, correct, update, *etc.* Taking the safety devices into

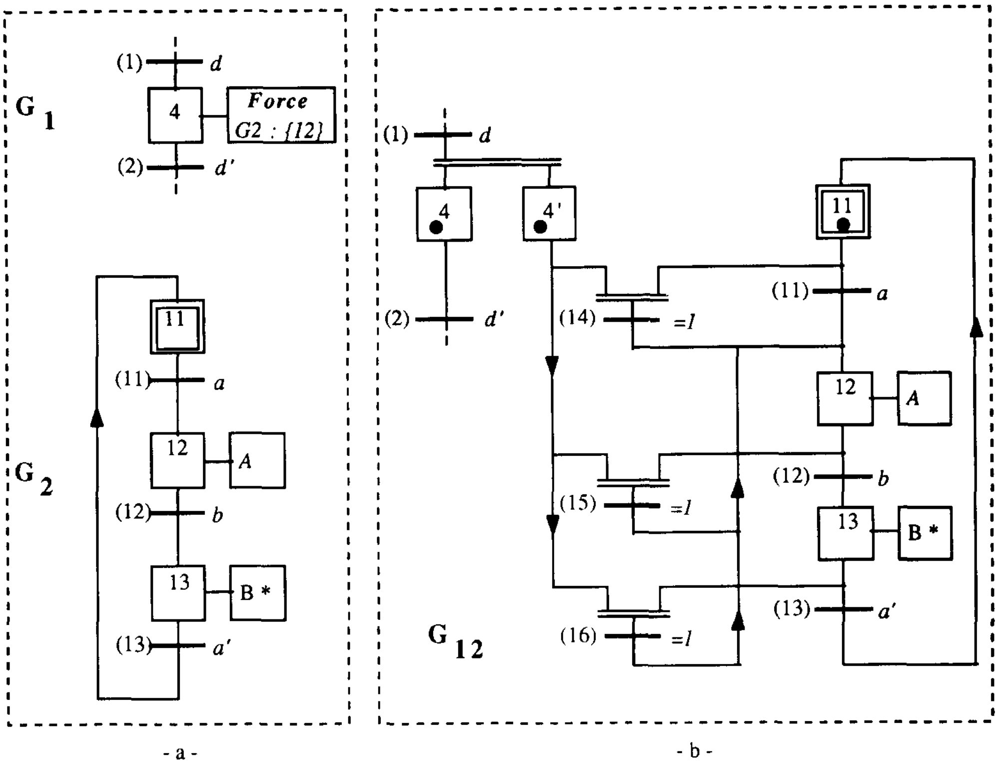


FIG. 21. Illustration of the macroaction force.

account, in particular, is an important reason for increasing complexity if we wish to treat them like other parameters, whereas they have a different role, which is both less frequent (we hope!) and with greater priority. The concept of hierarchy naturally springs to mind. It is easy to imagine that a logic controller (described by a grafcet) has a global influence on another logic controller (described by another grafcet), this being known as a macroaction.

A macroaction may be a level or an impulse action (just like an ordinary action). It is produced by a grafcet G_1 and has an effect on the behavior of a grafcet G_2. A macroaction is thus homogeneous with an action from the point of view of G_1.

The macroaction *force* is illustrated in Fig. 21. *Force* G2: {12}, associated with step 4, means that the grafcet G_2 is put into the situation such that step 12 is active (while the other ones are inactive) when step 4 becomes active. It is an impulse macroaction. The set of the two grafcets G_1 and G_2 of Fig. 21(a) is equivalent to grafcet G_{12} of Fig. 21(b). When step 4 becomes active, step 4′ also becomes active. Whichever step of the right-hand part of the grafcet is active (11, 12 or 13), one of the transitions (14), (15) or (16) is enabled and fired immediately. The figure shows the situation in which step 11 is active and transition (1) has just been fired. This situation is transient. Transition (14) is then fired, thereby inactivating steps 4′ and 11 and activating step 12. Notice that as soon as step 12 has been activated, transition (12) can be fired if $b = 1$ because *force* is an impulse action. On the other hand if the level macroaction *forcing* was performed, the grafcet G_2 would remain in the situation {12} as long as step 4 would remain active. Transition (15) activates and inactivates simultaneously step 12, which changes nothing with respect to this step.

Other macroactions such as *forcing, freezing* and *masking* (level macroactions) may be used (David and Alla, 1992).

Notice that macrosteps and macroactions correspond to abbreviations of an 'ordinary' grafcet, since it is always possible to obtain an equivalent grafcet without any macrostep or macroaction, as illustrated by Figs 20 and 21.

4.5. *Stochastic Petri net*

In a timed PN, a fixed duration is associated with each place or with each transition of the net. Models are obtained which are well adapted for studying systems in which the operating durations are fixed. This is the case, for example, of production systems where the working time of a machine to treat a part is constant. However, phenomena exist which cannot be properly modeled with constant durations. This is the case, for example, of the proper functioning time (between two breakdowns) of a machine. This duration may be modeled by a random variable. Stochastic Petri nets may be used (Florin and Natkin, 1984; Ajmone Marsan *et al.*, 1985). A random time is associated with the firing of a transition. The most commonly used hypothesis is that the timings are distributed according to an exponential law. The marking $\mathbf{M}(t)$ of the stochastic PN is then an homogeneous Markovian process, and thus an homogeneous Markov chain can be associated with every stochastic PN.

Consider a system made up of two computers and

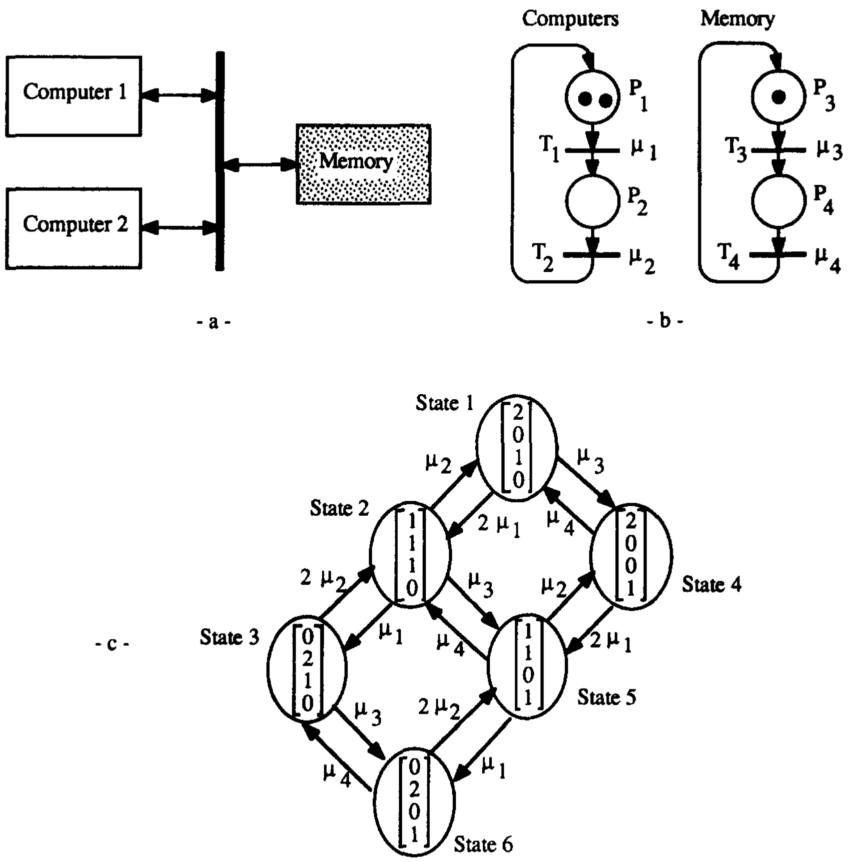


FIG. 22. Stochastic Petri net.

one memory which communicates by means of a bus [Fig. 22(a)]. We call μ_1 and μ_2 the computer breakdown and repair rates, and μ_3 and μ_4 the memory breakdown and repair rates.

A stochastic PN modeling this system is made up of two disconnected parts, as shown in Fig. 17(b). The initial marking, $\mathbf{M}_0 = (2, 0, 1, 0)$, indicates that both computers and the memory are operational. For $\mathbf{M}_0$ there are two enabled transitions, namely T_1 and T_3. The meaning of this PN is as follows. The probability that T_3 fires between times t and $t + \mathrm{d}t$, given it has not yet fired at time t, is $\mu_3 \cdot \mathrm{d}t$. The probability that T_1 fires between times t and $t + \mathrm{d}t$, given there are two tokens in P_1 at time t, is $2\mu_1 \cdot \mathrm{d}t$ (the probability that computer 1 fails between t and $t + \mathrm{d}t$ is $\mu_1 \cdot \mathrm{d}t$; since computer 2 can fail with the same probability, the probability that one of them fails is $2\mu_1 \cdot \mathrm{d}t$). Then transition T_1 will fire after some time d_1 has passed, and d_1 is a random variable distributed according to an exponential law whose rate is $2\mu_1$.

The Markovian process associated with this system is presented in Fig. 22(c). The initial marking $\mathbf{M}_0 = (2, 0, 1, 0)$ corresponds to the state 1 in this figure. If T_1 fires before T_3, then the state 2 corresponds to the marking $\mathbf{M}_1 = (1, 1, 1, 0)$ is reached, and so on.

In the stochastic PNs which has been presented, the tokens are never reserved. Another model can be defined such that tokens are reserved to fire some transitions (this model may be called stochastic timed PN). A stochastic PN and a stochastic timed PN model exactly the same system if there is no effective conflict.

4.6. *Timed continuous Petri nets*

Consider the system in Fig. 23(a). The behavior of this sytem is modeled in Fig. 23(b), assuming that at initial time: the three valves are open, and that there are 87.4 l in tank 1, 54 l in tank 2, while tank 3 is empty. In Fig. 23(b), places P_1, P_2 and P_3 represent the contents of tanks 1, 2 and 3, respectively (at the initial time in the figure).

Transition T_1 expresses that during one second (time unit), 1.3 marks are taken away from P_1, 1 mark is taken away from P_2, and 2.3 marks are added to P_3. This behavior [*i.e.* the instantaneous firing speed $v_1(t)$ is equal to V_1] lasts as long as $m_1 > 0$ and $m_2 > 0$. Then $v_1(t) = 0$, which corresponds to the specification of the system.

Transition T_2 expresses that during one time unit, 0.6 marks are taken away from P_3, when $m_3 > 0$. As long as $v_1(t) = 1$, it is clear that $m_3 > 0$, since $v_1(t) > V_2$. When $v_1(t)$ becomes 0, one has $v_2(t) = V_2$ for some time because $m_3 > 0$. Then $v_2(t) = 0$, when $m_3 = 0$.

This model is called constant speed continuous PN (CCPN) (David and Alla, 1987). It allows modeling either of continuous systems or of discrete event systems when the number of tokens in places is sufficiently large. In Brinkman and Blaauboer (1990),

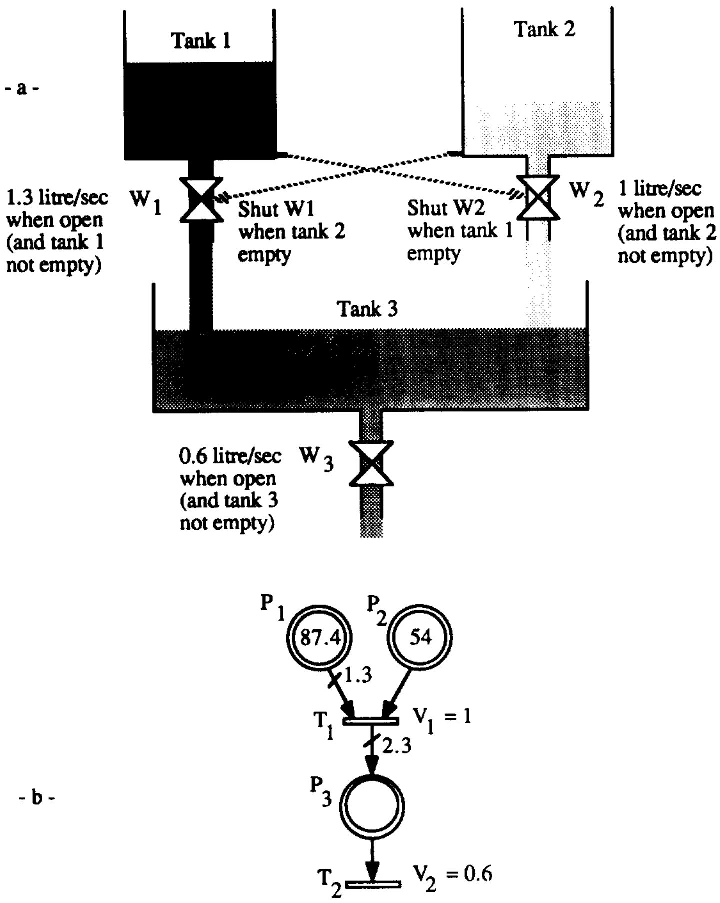


FIG. 23. Constant speed continuous Petri net.

delays were added in the CCPN. Two other timed continuous models have been defined. The variable speed continuous PN (VCPN) (David and Alla, 1990), and the asymptotic continuous PN (ACPN) (Le Bail *et al.*, 1993). These models, in which instantaneous firing speeds depend on the marking, approximate more accurately some discrete event systems when the number of tokens is small (unsaturated production system, for example). Here is an example presenting these models.

The discrete timed PN presented in Fig. 24(a) models a production station with two servers (modeled by transition T_2 and place P_2) and an entrance buffer (modeled by place P_3). The time d_1 associated with T_1 may represent the duration between two arrivals of customers in the entrance buffer. The service time of a server is d_2. The discrete timed PN gives the exact functioning of this system. It is our reference. Figure 24(b) shows the instants of firing of transitions T_1 and T_2 and the marking of place P_3 as a function of time t, which is denoted $m_3(t)$. Quantitative results can be obtained from Fig. 24(b). For example, after time 3, a periodical behavior is reached in which $m_3(t)$ is always equal to 2, and at each time unit, transitions T_1 and T_2 are fired.

From the discrete PN of Fig. 24(a), one can construct either a CCPN or a VCPN or an ACPN. In a CCPN, the instantaneous firing speed of a transition T_j is $v_j(t) = U_j = 1/d_j$ as long as the input places of T_j are not empty (U_j represents the maximal firing speed in a CCPN). When one of the input places is empty, $v_j(t)$ may have a lower value. In any case, the value $v_j(t)$ is piecewise constant. In this example, the CCPN presents the following features. The steady state is reached from the initial time. The firing speed $v_1(t) = 1$ for $t \geq 0$ is a good approximation. The firing speed $v_2(t) = 1$ for $t \geqslant 0$ is excessive at the beginning, since the first firing of T_2 occurs at $t = 3$, but is a good approximation afterwards. The marking $m_3(t) = 0$ is different from the corresponding marking of the discrete model [Fig. 24(b)].

The VCPN is a model taking into account the input place markings for computing instantaneous firing speeds. Consider a very simple example: assume transition T_j has a single input place P_i in a discrete timed PN. For some period of time D, there is sometimes a token in P_i (which is immediately reserved for the firing of T_j), and sometimes there is no token in P_i. Assume the average time of the periods without token is d_j, which is also the average time with a token. Then the average marking of P_i during the period D is $m_{av} = 0.5$. The average time between two firings is $d_{av} = 2d_i = d_i/m_{av}$. Now consider the same place and transition in a continuous timed PN, with $m_i(t) = 0.5$ during the same period of time D. The value $m_i(t)$ is similar to m_{av} in the

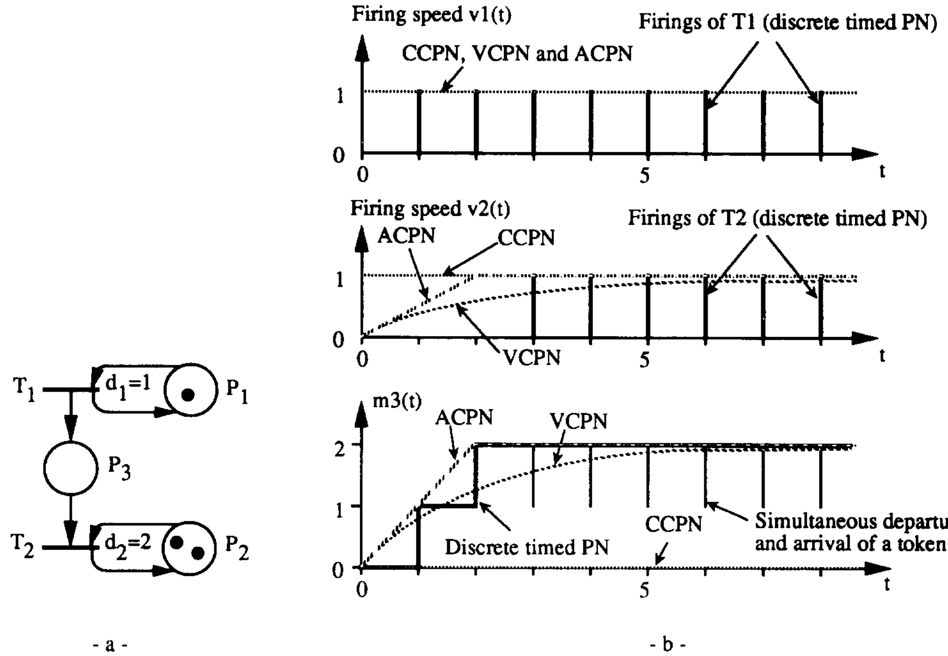


FIG. 24. A production station. (a) Petri net; (b) markings and firing speeds evolution.

discrete PN, and the instantaneous firing speed $v_j(t) = 1/d_{av}$, *i.e.* $v_j(t) = U_j m_i(t) = 0.5 U_j$ is a correct approximation. This is the basic idea.

Let $\{P_1, P_2, \ldots, P_a\}$ be the set of input places of the transition T_j. The instantaneous firing speed of T_j in a VCPN is (when T_j is not involved in a conflict):

$$v_j(t) = U_j \min(m_1(t), m_2(t), \ldots, m_a(t)).$$

It follows that, in the general case, instantaneous speeds and markings are given by differential equations. It can be observed in Fig. 24(b), that this model provides a good approximation of the discrete behavior in the transient and in the stationary state. The asymptotic value of m_1, v_1 and v_2 of this continuous model are exactly the average values of the corresponding discrete timed PN.

A new model called ACPN combines the advantages of the CCPN (allowing very fast simulations because the instantaneous firing speeds are piecewise constant) and the VCPN (good approximation of a discrete timed PN). This is illustrated in Fig. 24(b). Roughly speaking, an ACPN is an asymptotic approximation of a VCPN. This model cannot be explained in this paper. The reader is referred to Le Bail *et al.* (1993).

4.7. *Mixed models*

Mixing of several kinds of non-autonomous PNs may be useful for modeling some systems. Let us present two examples. The first example is a production system controlled by kanbans: it is modeled by a discrete PN which is both timed and synchronized (Di Mascolo *et al.*, 1990). The second example is a power station coupled with a pump storage station: it is modeled by a hybrid PN which is both timed and synchronized.

4.7.1. *Production system controlled by kanbans.* Figure 25(a) represents the control of a production system by kanbans (kanban is a Japanese word which means 'label'). This system is made up of two in series production meshes. Mesh i is made up of the production system PS_i and its buffer of finished products B_i (the parts in a buffer are not arranged: they may be considered to be loose, since they are all identical). The raw parts are in buffer B_0. In order for a part which is in buffer B_{i-1} to enter production system PS_i, it must carry a kanban i ($i = 1, 2$). When it is finished, it is deposited in buffer B_i with its kanban which remains attached to it. When a part is removed from B_i in order to satisfy a downstream request (request from an external customer for B_2, or request from mesh 2 by the arrival of a kanban 2 for B_1), it is separated from its kanban i and it is given a kanban $i+1$ (except if mesh i is the last one). Kanban i is then brought back to the entrance of production system PS_i to be allocated to another part.

The part treatments last 10 units in production system PS_1, and 12 units in system PS_2. In each of the loops, the return of a kanban from the exit to the entrance of the loop lasts one unit. All of the other operations have a zero duration. It is assumed that there are four kanbans for mesh 1, three for mesh 2, and that each production system can process only one part at one and the same time.

The Petri net in Fig. 25(b) is a model of this system. This PN is T-timed: transitions T_3, T_5, T_6 and T_8 are timed, and the timing is $d_j = 0$ for all the other transitions T_j (the timed transitions are represented with a thicker line). Transition T_1 is synchronized on event E_1 (entrance of a part in buffer B_0), and transition T_9 is synchronized on event E_2 (arrival of a request for a part in buffer B_2).

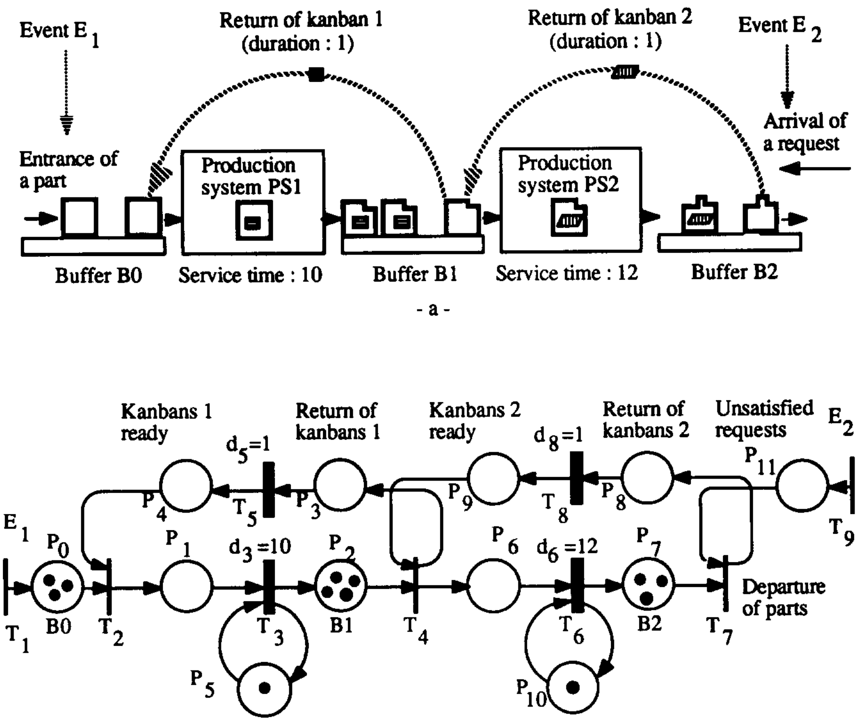


FIG. 25. Production system controlled by kanbans.

The initial marking in Fig. 25(b) is such that there have not been any requests coming from downstream in the system for a long time (P_{11} is empty, the four kanbans of mesh 1 are associated with parts in buffer B_1, and the three kanbans of meash 2 are associated with parts in buffer B_2), and there are three raw parts in buffer B_0.

One can observe that the number of kanbans in a mesh corresponds to a marking invariant: $m_1 + m_2 + m_3 + m_4 = 4$, for mesh 1.

This model is rather simple. However, if production system PS_1, for example, is more complicated, one can substitute a sub-PN corresponding to its working, to the sub-PN made of T_3, P_5 and the arcs between them. If the parts enter the production system by batches, one can model the behavior of the system using a generalized PN.

4.7.2. *Power station coupled with a pump storage station.* The specification of the system, which is modeled in Fig. 26, is as follows. A power station has a constant production of electric power which is 1300 MW. The average consumption is close to this value but is not constant (for example the demand of consumption decreases in the night and increases in daylight). In order to adjust the available production to the demand, the power station is coupled with a pump storage station. Roughly speaking: when the consumption is much lower than 1300 MW, part of the excess electric energy is converted into hydraulic energy by pumping; when the consumption is much higher than 1300 MW, part of the stored hydraulic energy is converted into electric energy. More precisely, the energy transformation has three states: (1) no transformation; (2) electricity production from available hydraulic energy; (3) hydraulic energy production from electricity in excess. When the consumption fits the constant production, there is no transformation and the voltage is roughly 15 kV. Assume that the consumption demand decreases. Since the power production is constant, the voltage increases. When this voltage reaches 16 kV, the system switches to pumping. If the system is in this state, when the demand increases the voltage decreases, and the switch to the no transformation state occurs when the voltage becomes less than 15 kV. Similarly one can switch from the no transformation state to the electricity production state when the voltage becomes less than 14 kV, and *vice versa* when the voltage reaches 15 kV. The maximum hydraulic energy which can be stored is 2000 MW h. The pumping produces an hydraulic power of 80 MW and makes use of 1.3×80 MW of electric power. The electricity production makes use of 100 MW and produces 0.7×100 MW of electricity.

In Fig. 26, the continuous transition T_5 models the constant production of electric power. This production corresponds to its constant speed, *i.e.* $V_5 = 1300$ MW. Since T_5 is a source transition, $v_5(t) = V_5$. The continuous transition T_6 models the variable consumption of electric power. This consumption depends on the time. The maximal consumption is $V_6(t)$, which corresponds to the demand at time t. If the demand is too high, it may be not satisfied, *i.e.* $v_6(t) \leq V_6(t)$.

Firing of transition T_7 corresponds to converting hydraulic energy into electric energy and firing of transition T_8 corresponds to converting electric energy into hydraulic energy. Only one out of the places P_1, P_2 or P_3 can be marked. When P_1 is marked (no

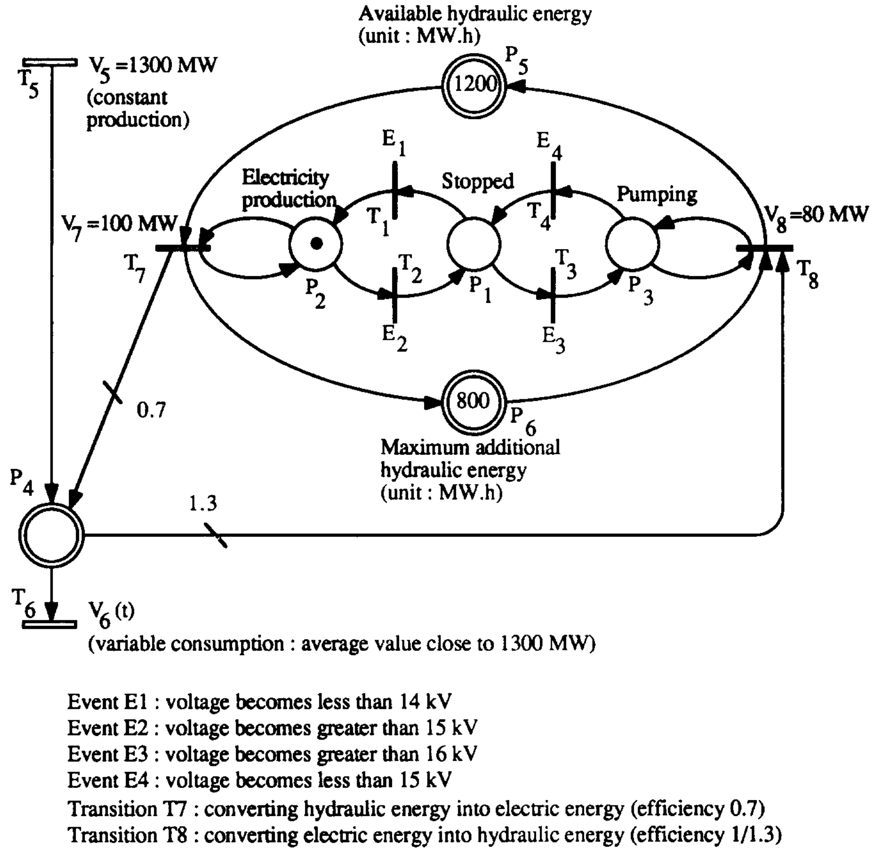


FIG. 26. Power station coupled with a pump storage station.

transformation) neither T_7 nor T_8 is enabled. When P_2 is marked, T_7 is enabled if the marking of P_5 is not nil, and its firing speed is $v_7(t) = V_7$: the hydraulic energy consumed per unit of time (*i.e.* power) is $v_7(t)$, while the released electric power is $0.7v_7(t)$. If P_2 is marked and P_5 is empty then $v_7(t) = 0$. Similarly, when P_3 is marked T_8 is enabled if the marking of P_6 is not nil.

Since the continuous place P_4 corresponding to electric power which is produced and not consumed must remain empty, one has $v_5(t) + 0.7v_7(t) = v_6(t) + 1.3v_8(t)$, at any time. For the marking of Fig. 26, when the voltage becomes greater than 15 kV, transition T_2 is fired on occurrence of this event E_2 (transition T_2 takes priority over T_7), and so on.

One can observe the marking invariant $m_5 + m_6 = 2000$ MW h which is the maximum storage of hydraulic energy.

4.8. *Comments on applications*

Non-autonomous PNs have two main application domains. The synchronized PNs (of which the interpreted PNs and almost similar models form part) enable the evolution of a system subjected to external constraints to be modeled (an important application is the description of controllers and real time systems). The timed and stochastic PNs, which take time into account, have as their natural vocation performance evaluation (data processing systems, production systems, *etc*).

Controllers. Basically, a logic controller is a discrete event system whose aim is to control the behavior of a process which is itself (seen as) a discrete event system, taking into account the state of this process, and other information coming from an operator or from other systems. The interpreted PNs and similar models are well adapted to this modeling. The Grafcet (David and Alla, 1992), whose behavior is inspired from Petri nets provides a clear understanding of the input/output behavior of a logic controller. Control of discrete event systems, taking explicitly into account that some events are controllable and that some others are not controllable, is now well established (Ramadge and Wonham, 1989). Such a control can be performed from a Petri net representation of the behavior of the system (Holloway and Krogh, 1990): Boolean variables, represented by the marking of additional control places, are associated with the various transitions.

Performance evaluation. Performance evaluation is a quantitative analysis which can be performed on PNs whose behavior depends on time, i.e. timed or/and stochastic PNs. These PNs may be discrete, continuous or hybrid, and the timings may be constant or not, but, in any case, their behaviors must be completely defined by timing considerations (deterministic or random).

The performance of the PN in Fig. 25(b), for example, cannot be obtained from this model only, because it is both timed and synchronized, and we have no information about occurrences of the events

E_1 and E_2. Now, if some information is given about the time between two successive occurrences of the events E_1 (and similarly for E_2), deterministic or random with a given distribution, the whole PN is 'timed', and a quantitative analysis is possible.

Some analysis methods allow analytical results to be obtained on timed Petri nets (the timings are deterministic). Sifakis' approach provides a calculation of the periodical functioning at maximal speed of a PN (Sifakis, 1977). It gives a set of inequalities which allow the mean firing frequencies of transitions to be determined in this functioning mode.

The Min–Max algebra approach is able to study the transient behavior of a system modeled by an event graph (Cohen *et al.*, 1989). It gives an analytical expression of the firings of transitions in the form of a transfer function. This latter contains a transient part and a periodical part.

The timed continuous PNs allow determination both of the transient behavior (an approximation of this behavior if the continuous model is an approximation of a discrete event system) and the steady state (see Section 4.6). Continuous PNs have been used to model a beer bottling line (Brinkman and Blaauboer, 1990). Ordinary PNs are used to model the human operators controlling the bottling line and the continuous part models the processing and the transport of the bottles. The use of hybrid PNs has been applied to the modeling for the performance evaluation of a workshop producing electronic components (Alla *et al.*, 1992). These components, which may be either transistors or diodes, are gathered into batches. A batch may be seen as a discrete entity or as a real quantity which approximates the number of parts. The resource machines are modeled as discrete variables and the parts are modeled as flows moving in the production system. The main aim of this study was to perform simulations in order to determine the total duration of the batch production for a given fabrication order sequence.

In the case of stochastic timings, the Markov chain deduced from the stochastic PN can be solved by classical methods. It is possible to compute the probabilities of states in stationary behavior. Performance indexes can be deduced from this, such as the mean markings of places or the mean firing frequencies of transitions.

Analytical methods can be used either when the timings are constant (Sifakis, 1977; Cohen *et al.*, 1989), or when the timings are stochastic with exponential laws (Ramamorthy and Ho, 1980). When the quantitative analysis by analytical methods is not possible (for example mixing of constant and stochastic times), this analysis can be performed thanks to simulation. The simulation time depends on the number of events to take into account in this simulation (this number is always large when randomly distributed times are involved). This simulation can be performed by commercial softwares, for example Design/CPN (Meta, 1990) or Eval (Verilog, 1991).

The advantage in using PNs for specification and evaluation of discrete events sytems, is that they allow a qualitative analysis of the model before the evaluation. The main weakness is that PNs are not a structured modeling tool, as can be a structured language.

5. CONCLUSION

A variety of models have been presented, each one with its own particular features.

The basic model is the autonomous Petri net (ordinary or generalized). It enables a discrete event system of any kind whatsoever to be modeled. When a model has been established for a given system, it can be used to explain 'how it works'. This allows the qualitative validation of a functioning process.

Non-autonomous PNs have two major application categories: to begin with, the synchronized PNs (of which the interpreted PNs form part) enable the evolution of a system subjected to external constraints to be modeled. An important application is the description of controllers and real time systems.

Then, the timed and stochastic PNs take time into account, which allows the quantitative analysis of a functioning process. They have as their natural vocation performance evaluation (data processing systems, production systems, *etc*).

The hybrid PNs may be used when some part can be modeled by a continuous PN (continuous system or approximation of discrete event system), while another needs a discrete modeling. The colored PNs facilitate modeling of large-size systems, certain parts of which have fairly similar functionings. The parameter setting which can be performed from a generic graphic model is very suitable for certain categories of production systems. All these models may be autonomous or not.

Each of these models thus has its own specific character and privileged fields of application. Nevertheless, the Petri net forms a common basis: it may be likened to a 'common language' allowing dialogue between persons of very varied training backgrounds.

Acknowledgements—The authors would like to thank colleagues J. Le Bail and J.-P. Peyrol for their help in preparing this paper.

REFERENCES

Alla, H., J. B. Cavaillé, J. Le Bail and G. Bel (1992). Modélisation par réseaux de Petri hybrides et evaluation de performances de systèmes de production par lots. *Rapport final de contrat MRT,* LAG n° 92-06, March.

Ajmone Marsan, M., G. Balbo, A. Bobio, G. Chiola, G. Conte and A. Cumani (1985). On Petri nets with stochastic timing. *Int. Workshop on Timed Petri Nets,* Torino.

Berthelot, G. and R. Terrat (1982). Petri nets theory for correctness of protocols. *IEEE Trans. Commun.,* **COM-30,** 2497–2505.

Brams, G. W. (1983). Réseaux de Petri: théorie et pratique. Masson, Paris.

Brinkman, P. L. and W. A. Blaauboer (1990). Timed continuous Petri nets: a tool for analysis and simulation of discrete event systems. 1990 *European Simulation Symp.,* Ghent.

Carlier, J. and P. Chrétienne (1983). Modelling scheduling problems with timed Petri nets. *4th European Workshop on Theory and Applications of Petri Nets*, Toulouse.

Cohen, G., P. Moller, J.-P. Quadrat and M. Viot (1989). Algebraic tools for the performance evaluation of discrete event systems. *Proc. IEEE*, **77**, 39–57.

Colom, J. M. and M. Silva (1989). Improving the linearly based characterization of P/T nets. *10th Int. Conf. on Application and Theory of Petri Nets*, Bonn.

Chrétienne, P. (1983). Les réseaux de Petri temporisés. Thèse d'Etat, Paris VI University.

David, R. (1991). Modeling of dynamic systems by Petri nets. *1st European Control Conf.*, Grenoble, pp. 136–147.

David, R. and H. Alla (1987). Continuous Petri nets. *8th European Workshop on Application and Theory of Petri Nets*, Zaragoza, pp. 275–294.

David, R. and H. Alla (1989). *Du Grafcet aux réseaux de Petri*. Hermès, Paris.

David, R. and H. Alla (1990). Autonomous and timed continuous Petri nets. *11th Int. Conf. on Application and Theory of Petri Nets*, Paris, pp. 367–386.

David, R. and H. Alla (1992). *Petri Nets and Grafcet: Tools for Modelling Discrete Event Systems*. Prentice-Hall, London.

Di Mascolo, M., Y. Frein, Y. Dallery and R. David (1991). A unified modeling of kanban systems using Petri nets. *Int. J. Flexible Manufac. Syst.*, **3**, 275-307.

Florin, G. and S. Natkin (1984). Définition formelle des réseaux de Petri stochastiques. *Research Report CNAM*, Paris.

Genrich, H. J. and K. Lautenbach (1983). *Application and Theory of Petri Nets*. Springer, Berlin.

Hillion, H.-P. and J.-M. Proth (1989). Performance evaluation of job-shop system using timed event-graphs. *IEEE Trans. Aut. Control*, **AC-34**, 3–9.

Holloway, L. E. and B. H. Krogh (1990). Synthesis of feedback control logic for a class of controlled Petri nets. *IEEE Trans. Aut. Control*, **AC-35**, 514–523.

Jantzen, M. and R. Valk (1979). Formal properties of place/transitions nets. *Lecture Notes in Computer Science*, Vol. 84(15). Springer, Berlin.

Jensen, K. (1980). How to find invariants for coloured Petri nets. *DAIMI PB* 120. Aarhus University, Denmark.

Jensen, K. (1981). Coloured Petri nets and the invariant method. *Theoretical Comp. Sci.*, **14**, 317–336.

Karp, R. and R. Miller (1969). Parallel program schemata. *J. Comp. Syst. Sci.*, **3**, 167–195.

Lautenbach, K. and H. A. Schmid (1974). Use of Petri nets for proving correctness of concurrent process systems. *Proc. IFIP Cong.* 75, pp. 187–191. North-Holland, Amsterdam.

Le Bail, J., H. Alla and R. David (1991). Hybrid Petri nets. *1st European Control Conf.*, Grenoble, pp. 1472–1477.

Le Bail, J., H. Alla and R. David (1993). Asymptotic continuous Petri nets. *Discrete Event Dynamic Systems: Theory and Applications*, Vol. 2, pp. 235–263.

Memmi, G. (1983). Méthodes d'analyse des réseaux de Petri, réseaux à files. Applications aux systèmes en temps réels. Thèse d'Etat, Paris VI University.

Memmi, G. (Ed.). (1985). Special Issue, Réseaux de Petri. *TSI—Technique et Science Informatiques*, **4**.

Meta (1990). *Design/CPN*. Meta Software Corporation, Cambridge, MA.

Moalla, M. (1985). Réseaux de Petri interprétés et Grafcet. *TSI—Technique et Science Informatiques*, **14**, 17–30.

Moalla, M., J. Pulou and J. Sifakis (1978). Réseaux de Petri synchronisés. *J. RAIRO, Automatic Control Series*, **12**, 103–130.

Murata, T. (1989). Petri nets: properties, analysis and applications. *Proc. IEEE*, **77**, 541–580.

Peterson, J. L. (1981). *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, Englewood Cliffs, NJ.

Petri, C. A. (1962). Kommunikation mit Automaten, Schriften des Rheinisch *Westfalischen Inst. fur Intrumentelle Mathematik and der Universität Bonn*. Translation by C. F. Greene, Applied Data REsearch Inc., Suppl. 1 to Tech. Report RADC-TR-65-337, NY.

Ramadge, P. J. G. and W. M. Wonham (1989). The control of discrete event systems. *Proc. IEEE*, **77**, 81–98.

Ramamorthy, C. V. and G. S. Ho (1980). Performance evaluation of asynchronous concurrent systems using Petri nets. *IEEE Trans. Software Eng.*, **SE-6**, 440–449.

Ramchandani, C. (1973). Analysis of Asynchronous Concurrent Systems by Timed Petri Nets. Ph.D. Thesis. MIT, Cambridge, MA.

Reutenauer, C. (1990). *The Mathematics of Petri Nets*. Prentice-Hall, London.

Rosenberg, G. (Ed.) (1984–1993). *Advances in Petri Nets*. Lecture notes in Comp. Sci., Springer, New York.

Sifakis, J. (1977). Use of Petri nets for performance evaluation. Measuring. In H. Beilner and E. Gelenbe (Eds), *Modelling and Evaluating Computing Systems*, pp. 75–93. North-Holland, Amsterdam.

Silva, M. (1985). *Las redes de Petri en la Automatica y la Informatica*. Ed. AC, Madrid.

Silva, M. and R. Valette (1990). Petri nets and flexible manufacturing. *Advances in PN'89*, *LNCS* 84. Springer, Berlin.

Silva, M. and S. Velilla (1982). Programmable logic controllers and Petri nets. *IFAC Symp. on Software for Comp. Control*, Madrid, pp. 29–34.

Valette, R. (1979). Analysis of Petri nets by stepwise refinements. *J. Comp. Syst. Sci.* **18**, 35–46.

Verilog (1991). Vérifier et évaluer un système avec EVAL. Verilog SA, Toulouse.

Zhou, M. C. and F. DiCesare (1993). *Petri Net Synthesis for Discrete Event Control of Manufacturing Systems*. Kluwer Ac, Boston, MA.