DISCRETE NETS AND THEIR APPLICATION TO SYSTEMS ANALYSIS

Alexander Petrenko

October 1982
WP-82-109

INTERNATIONAL INSTITUTE FOR APPLIED SYSTEMS ANALYSIS
2361 Laxenburg, Austria

ABSTRACT

To assist in the design and the analysis of complex discrete systems exhibiting concurrancy, formal techniques are needed which comprehend a hierarchical representation of such systems and a rigourous analysis of their properties. This paper presents the main axiom and definitions of uninterpreted models named discrete nets, and demonstrates by examples the usefulness of proposed formalization.

CONTENTS

DISCRETE NETS AND THEIR APPLICATION TO SYSTEMS ANALYSIS

Alexander Petrenko

INTRODUCTION

A complex discrete system consists of a large number of components (subsystems) which function concurrently or parallel in time. The processes of subsystems are not completely independent, they rather frequently communicate, exchanging information and sharing resources of the system. To analyse such a system it is necessary to have an appropriate formal language which allows the system to be modelled on a different level of abstraction in order to establish various properties of the system. For the analysis of discrete systems with asynchronous processes so called Petri nets, introduced by C.A. Petri in 1962 [1], have attracted great attention due to their simplicity and ability to describe and analyse concurrent systems. Petri nets are utilized to model the following aspects of discrete systems: events, and conditions, and their

relationships. It is usually assumed when using Petri nets that there are some conditions valid at a particular moment, these conditions can cause some new events within the system. New events can change states of some subsystems, changing conditions, and these changes can proceed concurrently. It has been established that Petri nets can be successfully applied in the analysis of a rather broad class of systems [2] but at the same time it became obvious that the original model of C.A. Petri possesses some drawbacks: inability to represent some real situations and processes, and complexity of the resulting distribution. Some authors have proposed several modifications to Petri nets [3-7]. All these improvements are in fact modifications of the original definition of the firing rule in such a way that the condition of the event is formulated as a logical function of primary conditions, which define the state of the system. But there is rather a broad range of problems which do not allow the conditions of events to be represented in the form of a simple logical function. In this paper we try to introduce a general model for description and analysis of discrete systems which allows the class of represented systems to be broadened, and to get a description at an acceptable level of complexity.

The paper is organized as follows. First we introduce some useful definitions (Section I) about bag theory and state machines. Then the structure and dynamics of a general model named discrete net is defined (Section II). In Section III some properties of discrete nets are introduced that are useful for systems analysis. Section IV contains some particular cases of discrete nets. Section V presents some examples of applications of discrete nets.

I PRELIMINARY DEFINITIONS.

Definition 1.1 The bag B over the set X is determined by the pair: $X, X \rightarrow N$, where N is the set of non-negative integers. The mapping $X \rightarrow N$ is called the *occurrances function* $\#(x,b)$. If $\#(x,B) > 0$, then $x \in B$ and $x \notin B$, if $\#(x,b) = 0$.

A bag A over the set X is a *subbag* of a bag B over the set X (denoted by $A \subseteq B$) , if $\#(x,A) \leq \#(x,B)$ for all $x \in X$.

Two bags over the same set X are *equal* $(A = B)$ if $\#(x,A) = \#(x,B)$ for all $x \in X$.

A bag A is *strictly contained* in a bag $B (A \subset B)$, if $A \subseteq B$ and $A \neq B$.

The *bag space* X^n is the set of all bags $B \in X^n$ over the set X such that $\#(x,B) \leq n$ for all $x \in X$. The set X^∞ is the set of all bags over the set X .

Definition 1.2 The following operations are defined on bags A and B over the same set X :

Bag union $A \cup B$:

$$\#(x, A \cup B) = \max(\#(x,A), \#(x,B)) \ .$$

Bag intersection $A \cap B$:

$$\#(x, A \cap B) = \min(\#(x,A), \#(x,B)) \ .$$

Bag sum $A + B$:

$$\#(x, A+B) = \#(x,A) + \#(x,B) \ .$$

Bag difference $A - B$:

$$\#(x, A - B) = \#(x,A) - \#(x, A \cap B) \ .$$

Bag union, intersection and sum are commutative and associative. The following properties are held.

$$A \cap B \subseteq A \subseteq A \cup B \ ,$$
$$A - B \subseteq A \subseteq A + B \ ,$$
$$|A \cup B| \leq |A + |B| \ ,$$
$$|A + B| = |A| + |B| \ ,$$

where $|A| = \sum_{x \in A} \#(x,A)$.

The relations and operation introduced above will also be utilized when bags are represented as vectors.

Definition 1.3

A *state machine* A is a six-tuple $A = (K,R,L,\delta,\gamma,\kappa_0)$,

where

K is a finite set of states,

R is a finite input alphabet,

L is a finite output alphabet ,

$\delta : K \times R \rightarrow K$ is the output function ,

$\gamma : K \times R \rightarrow L$ is the output function,

$\kappa_0 \in K$ is the initial state .

We assume that a state machine does not change its state and produces empty output ϕ under any input which does not belong to its input alphabet. A state machine can be represented in forms of state diagrams or flow tables.

II. DISCRETE NETS.

Definition 2.1 A *discrete net* C is a five-tuple $C = (P,T,I,O,\mu)$, where

$P = \{p_1,p_2,\cdots,p_n\}$ is a finite set of places,

$T = \{t_1,t_2,\cdots,t_m\}$ is a finite set of transitions, $P \cap T = \phi$,

$I : T \to P^\infty$ is the input function,

$O : T \to P^\infty$ is the output function,

$\mu : P \to M^\infty$ is the marking function (marking),

M - is the finite set of markers $M_i \in M$, including one special empty marker ϕ.

A place p_i is an input place of a transition t_j if $p_i \in I(t_j)$; p_i is an output place of t_j , if $p_i \in O(t_j)$.

A discrete net structure has a graphical representation in a form of a bipartite directed multigraph G , which consists of two types of nodes: p -nodes represent places, shaped as circles and t -nodes represent transitions shaped as rectangles. Directed arcs connect the places and the transitions; from a node p_i to a node t_j there is exactly $\#(p_j,I(t_j))$ arcs and from t_j to $p_i - \#(p_i,O(t_j))$. In a graph G p -nodes are weighted by $\mu_i = \mu(p_i)$ - a bag of markers, assigned to a place p_i by a marking μ, which can be represented by an n -vector $\mu = (\mu_1,\mu_2,\cdots,\mu_n)$, where $n = |P|$. The above defined graph will be called a ***discrete net graph***.

By a ***marked*** discrete net we name a net $C = (P,T,I,O,\mu^0)$ where μ^0 is an initial marking.

The *execution* of a discrete net is considered in discrete time, when a current marking is changing (the function μ changes its values) due to

firings of transitions which are controlled by the distribution of markers resided in places.

Let t_i be a transition of a discrete net with $I(t_i)$ as a set of input and $O(t_i)$ -output places. We enumerate all input arcs : $1,2,\ldots,n_i; n_i = |I(t_i)|$; a n_i -vector formed from markers of the set M is an input character ρ^i of a transition t_i . Output characters λ^i are defined similar to input ones to be m_i-vectors, $m_i = |O(t_i)|$. Sets of all admitted for t_i input and output characters are the input R_i and output L_i alphabets of a transition t_i , respectively. These characters will be represented also as n -vectors. Let $\mu_{j\rho^i}$ be a bag of markers from ρ^i corresponding to all input arcs between t_i and $p_j \in I(t_i)$, here $|\mu_{j\rho^i}| = \#(p_j, I(t_i))$ and $\mu_{j\rho^i} = 0$ if $p_j \notin I(t_i)$. An n -vector $\mu_{\rho^i} = (\mu_{1\rho^i}, \mu_{2\rho^i}, \cdots, \mu_{n\rho^i})$ is the marking of an input character ρ_i of a transition t_i , in the same way we define $\mu_{\lambda^i} = (\mu_{1\lambda^i}, \mu_{2\lambda^i}, \ldots, \mu_{n\lambda^i})$ -the marking of an output character λ^i of a transition t_i .

Definition 2.2 A transition t_i in a marked discrete net $C = (P, T, I, O, \mu')$ is ***enabled*** if there is at least one input character $\rho^i \in R_i$ called ***enabling*** character such that its marking μ_{ρ^i} satisfies the following $\mu_{\rho^i} \subseteq \mu'$, that is $\mu_{j\rho^i} \subseteq \mu'(p_j)$ for all $p_j \in I(t_i)$.

The execution of a discrete net is the shifting of markers within a net caused by firing an enabled transition t_i which performs a mapping $R_i \to L_i$. An algorithm of this mapping we will define by means of a state machine $A_i = (K_i, R_i, L_i, \delta_i, \gamma_i, \kappa_0^i)$.

Definition 2.3 A transition t_i with a state $\kappa_c^i \in K_i$ may *fire* whenever it is enabled. Firing an enabled transition t_i results in a new marking $\mu' \to \mu''$ defined by

$$\mu'' = \mu' - \mu_{\rho_e^i} + \mu_{\lambda_b^i}$$

and in a new state of transition t_i (a state machine A_i) $\kappa_c^i \to \kappa_d^i$, where $\mu_{\rho_e^i} \subseteq \mu'$, $\lambda_b^i = \gamma_i(\kappa_c^i, \rho_e^i)$ and $\kappa_d^i = \delta_i(\kappa_c^i, \rho_e^i)$.

In this case the marking μ'' is said to be *immediately reachable* from μ' . If μ' is immediately reachable from μ and μ'' is immediately reachable from μ' , then we say that μ'' is *reachable* from μ . The *reachability set* $D(C,\mu)$ of a discrete net with marking μ is defined to be all markings reachable from μ .

It might be seen from definition 2.2 that an enabled transition t_i can have more than only one enabling character $\{\rho_{j_1}^i, \rho_{j_2}^i, \cdots, \rho_{j_l}^i\} \subseteq R_i, l > 1$.

Definition 2.4 An enabled transition t_i is said to have an *internal conflict* , if

$$\mu_j \subset \sum_{r=1}^{l} \mu_{\rho_{jr}^i}.$$

The set of enabling input characters defines all possible changes of a marking when one particular transition fires. All these changes are realized only if this transition does not have any internal conflicts, otherwise the transition fires indeterminately. As the most general case we assume that the enabling character is being selected arbitrarily, but we will keep in mind the chance to postulate for some particular systems a partial ordering relation on sets R_i which enables us to resolve internal conflicts within a discrete net.

Let $t_i, t_j \in T$ be two transitions of a discrete net $C = (P,T,I,O,\mu)$, which are enabled by one marking μ and have sets of enabling characters $\rho_1^i, \rho_2^i, \cdots, \rho_{l_i}^i$ and $\rho_1^j, \rho_2^j, \cdots, \rho_{l_j}^j$, respectively.

Definition 2.5 Two transitions t_i and t_j are in ***conflict*** under the marking μ, if there are ρ_r^i, $(1 \le r \le l_i)$, $\rho_q^j (1 \le q \le l_j)$ that

$$\mu \subset \mu_{\rho_r^i} + \mu_{\rho_q^j}.$$

Between two transitions t_i, t_j being in conflict only one fires if $\sum_{r=1}^{l_i} \mu_{\rho_r^i} \subseteq \mu$ and $\sum_{q=1}^{l_j} \mu_{\rho_q^j} \subseteq \mu$. We assume as earlier that if the firing transition is being selected arbitrarily.

Definition 2.6 A marking μ of a discrete net $C = (P,T,I,O,\mu)$ is ***deadlock*** if it does not enable any transition.

The execution of a discrete net holds whenever it becomes a deadlock marking.

III PROPERTIES OF DISCRETE NETS

Definition 3.1 A place $p_i \in P$ of a marked discrete net $C = (P,T,I,O,\mu)$ is called:

(a) k_{M_j} ***−bounded*** if for all $\mu' \in D(C,\mu)$ and a marker $M_j \in M$, $\#(M_j, \mu'(p_j)) \le k$;

(b) k ***−bounded*** if for all $\mu' \in D(C,\mu)$ and all $M_j \in M$, $\#(M_j, \mu'(p_i)) \le k$

(c) M_j ***−bounded*** if it is k_{M_j} - bounded and $k = 1$.

Definition 3.2 A discrete net is called:

(a) k_{M_j} *−bounded* if all its places are k_{M_j} -bounded.

(b) k *−bounded* if all its places are k -bounded,

(c) *safe* if it is k -bounded and $k = 1$.

Definition 3.3 A transition $t_i \in T$ of a marked discrete net $C = (P,T,I,O,\mu)$ is *active* under the marking μ if there is $\mu' \in D(C,\mu)$ which enables this transition, and otherwise *passive* under the marking μ .

If a marked discrete net has some passive transitions under the initial marking μ^0 then all those can be removed from the net without having influenced its execution.

Definition 3.4 A transition $t_i \in T$ of a marked discrete net $C = (P,T,I,O,\mu)$ is called :

(a) *live* if it is active under each marking $\mu' \in D(C,\mu)$,

(b) k -live, if there is a firing transition sequence initiated by μ in which t_i occurs at least k times $(k \geq 1)$.

Definition 3.5 A marked discrete net $C = (P,T,I,O,\mu)$ is called:

(a) *live* , if all its transitions are live under the marking μ ,

(b) k -live if all its transitions are k -live under the marking μ .

Definition 3.6 A subset $P_{inv} \subseteq P$ of places of a marked discrete net $C = (P,T,I,O,\mu)$ is called an *invariant* if for all $\mu' \in D(C,\mu)$,

$$\sum_{p_j \in P_{inv}} \mu'(p_j) = \sum_{p_j \in P_{inv}} \mu(p_j).$$

The properties of discrete nets introduced above will be used to establish characteristics of modelled discrete systems.

IV SUBCLASSES OF DISCRETE NETS

Various properties of discrete net structures allow us to distinguish several subclasses of discrete nets.

Definition 4.1 A discrete net $C = (P,T,I,O,\mu)$ is called:

(a) ***simple*** if for all $p_i \in P$ and $t_j \in T$, $\#(p_i,I(t_j)) \leq 1$ and $\#(p_i,O(t_j)) \leq 1$;

(b) $loop-free$, if for all $t_i \in T, I(t_i) \cap O(t_i) = \phi$;

(c) $p-connected$ if for all $t_i \in T$,

$$|I(t_i)| = 1 \quad and \quad |O(t_i)| = 1;$$

(d) $t-connected$ if for all $p_i \in P$,

$$|\{t_j \,|p_i \in I(t_j)\}| = 1 \quad and \quad |\{t_j \,|p_i \in (t_j)\}| = 1.$$

In a simple discrete net bags $I(t_i)$ and $O(t_i)$ are sets and in its graph there are no multiple arcs.

A loop-free net does not have any circles with one transition and one place, because no single place can be both an input and output place for the same transition.

Each transition of p -connected nets has got exactly one input and one output place; these nets can easily represent conflicts by a place with several outputs but cannot model the concurrency or waiting which characterize synchronization. T -connected nets are duals of p -connected nets from graph-theoretical point of view, they can model syn-

chronization and concurrency but cannot represent conflicts, because no single place is shared by several transitions.

Additional classes of discrete nets follow from the properties of state machines which define transitions.

Definition 4.2 A discrete net $C = (P,T,I,O,\mu)$ is called:

(a) *combinational* if all its transitons are defined by state machines with any one state ($|K_i| = 1,\ i = 1,2,\cdots,m$) ;

(b) *binary* if the set of markers M contains only one nonempty marker along with the empty one.

If we assume $M = \{0,1\}$ be the set of markers of a binary combinational net then state machines of transitions are in fact systems of Boolean functions. One particular case when all those Boolean functions are logical AND operations, corresponds to the definition of the original Petri net model [1]. There are also a number of extended Petri nets [6] such as nets with zero-testing exclusive OR transitions, and with other input and output (with respect to transitions) logic. It is easy to see that all those modifications differ only in a type of boolean functions system, which define transitions; in other words they are included into the class of binary combinational nets. An extended Petri model known as coloured Petri net [4] is covered by the combinational discrete nets in which all state machines do not have memory (having only one state) and can be represented in a form of multi-valued logical functions. The formal proof of the above relationships between different net models is a subject for separate publication, here we only show by examples that discrete net models can represent a broader class of real systems than Petri nets and

their modifications do.

VI EXAMPLES OF MODELLING USING DISCRETE NETS.

One of the most difficult design problems of a concurrent system with parallel processes (subsystems) is organization of cooperation between parallel processes because they share information and/or resources of the system. The asynchronous nature of concurrent processes dictates the necessity of a special synchronization mechanism to handle waiting and deadlock problems. A variety of synchronization problems has been proposed in the literature to illustrate the types of relations which can arise between cooperating processes and mechanisms for solving them. Among these is the dining philosophers problem [8]. It concerns five philosophers who alternatively think and eat. They are seated at a round table with a large number of Chinese foods, between each philospher is one chopstick. To eat Chinese food everyone needs two chopsticks (from left and right sides). The problem is that if all philosophers pick up the chopstick on their left side and then wait for the chopstick on their right they will wait forever and starve (but the system would be deadlocked).

To model this system we build a discrete net. First we introduce three places $P = \{p_1,p_2,p_3\}$ with the interpretation: p_1 represents thinking philosophers, p_2 -eating ones, and p_3 -represents free chopsticks. The dynamics of the system is described by two transitions t_1-take chopsticks, t_2 -put chopsticks back with $I(t_1) = (p_1,p_3,p_3)$, $O(t_1) = p_2$, $I(t_2) = p_2$, $O(t_2) = (p_1,p_3,p_3)$. Markers e_1,e_2,e_3,e_4,e_5 represent philo-

sophers, l_1,l_2,l_3,l_4,l_5 -chopsticks (l_1 -the one, which lies between e_1 and e_5 and so on) , thus $M = \{e_1,e_2,e_3,e_4,e_5,l_1,l_2,l_3,l_4,l_5\}$. The discrete net graph is represented in Figure 1. We describe the transitions with two state machines A_1 and A_2 (table 1 and 2), each of them has five input, five output characters, and one state. As an initial marking of this discrete net we can assume

$\mu^0(p_1) = (e_1,e_2,e_3,e_4,e_5)$, $\mu^0(p_2) = \phi$, $\mu^0(p_3) = (l_1,l_2,l_3,l_4,l_5)$.

This net is t -connected, therefore transitions t_1 and t_2 cannot be in conflict , then operations "take chopsticks" and "put chopsticks" are deterministic and predictable regardless of the time parameters of the system. As it follows from table 2 the transition t_2 does not have internal conflict, that means philosophers do not compete when putting chopsticks back on the table. It is obvious that $\mu(p_1) \subseteq \mu^0(p_1)$ and $\mu(p_3) \subseteq \mu^0(p_3)$ for all $\mu \in D(C,\mu^0)$, also $\mu_0 \subset \sum_{i=1}^{5} \mu_{p_i^1}$, therefore the transition t_1 has an internal conflict. A sum of any two input characters with the same marker $l_j (j = 1,2,\cdots,5)$ strictly contains the marking μ^0 thus any neighbors never eat at the same time. The set (p_1,p_2) is an invariant that is every philosopher either eats or thinks.

It is worth mentioning that the introduced net has only two transitions and three places, meanwhile the description of this system in a form of original Petri net requires 10 transitions and 15 places. This shows the possibility of more concise presentation of a modelled system by means of discrete nets.

Let us consider a slight modification of the just considered system. We put the following restriction on the synchronization mechanism: the first and third philosophers must start eating alternately, that is the third philosopher starts after the first one, then again the first and so on. If a philosopher (first or third) is not allowed to start eating then he must think and wait until another has started. This system is modelled by another discrete net (Figure 2) which differs from the previous one (Figure 1) by the transition t_1 with $O(t_1) = (p_1, p_2, p_3, p_3)$. The state machine of t_1 has already two but not one states κ_1 and κ_2 , introducing into consideration the past history of the system, that is, information above preceding philosophers having taken chopsticks (table 3). As one can see from this table the transition t_1 under the same marking puts markers in different places depending (regarding) on its state. This effect illustrates the fact that discrete nets offer greater possibilities for modelling of discrete systems than Petri nets. As one could learn from the above examples, a discrete net is in fact an uninterpreted model which opens the way to hierarchical description of systems.

CONCLUSION

In this paper we have introduced a formal model for complex discrete systems which might be considered as a generalization of existing Petri nets exceptions. The usefulness of the model was illustrated by examples. The proposed methodology allows the modelling of a rather broad class of problems: from communication protocols, synchronization problems, to organization of production schemata. Based on the introduced axioms and definitions, the further investigation to develop

proposed methodology is the establishment of relationships betwen separate properties of discrete nets and construction of formal procedures for checking of systems properties.

REFERENCES

1. Petri, C.A. : Kommunikation mit Automaten, Ph.d. dissertation, University of Bonn, 1962.
2. Peterson, J. : Petri nets, - Computing Surveys, 1977, V.9, No.3, pp 223-252.
3. Genrich, H.J., Lautenbach, K. : System modelling with high-level Petri nets, in Theor. Computer Science, 1981, V.13, pp 106-136
4. Schiffers, M., Wedde, M. : Analyzing program solutions of coordination problems by CP - nets, in Mathematical foundations in Computer Science, 1978, No.64, pp 462-473.
5. Petrenko, A.F. : On the specification and verification of protocols using Petri nets, in Proceedings of the ICCC - 80, Atlanta, 1980, pp 385-390.
6. Agerwala, T., Flynn, M. : Comments on capabilities, limitations and "correctness" of Petri nets, in Proc. First Ann. Sympos. of Computer Architecture, A.C.M., 1973, pp 81-86.
7. Noe, J., Nutt, G. : Macro E-nets for representation of parallel systems, IEEE Trans. on Computers, 1973, V.22, No.8, pp 718-727.
8. Dykstra, E. : Cooperating Sequential Processes, in Programming Languages, N.Y. Academic Press, 1968, pp 43-112.

A_1

ρ^1	$e_1 l_1 l_2$	$e_2 l_2 l_3$	$e_3 l_3 l_4$	$e_4 l_4 l_5$	$e_5 l_5 l_1$
λ^1	e_1	e_2	e_3	e_4	e_5

Table 1

A_1

ρ^2	e_1	e_2	e_3	e_4	e_5
λ^2	$e_1 l_1 l_2$	$e_2 l_2 l_3$	$e_3 l_3 l_4$	$e_4 l_4 l_5$	$e_5 l_5 l_1$

Table 2

A_1

ρ^1	$e_1 l_1 l_2$	$e_2 l_2 l_3$	$e_3 l_3 l_4$	$e_4 l_4 l_5$	$e_5 l_5 l_1$
κ_1	$\kappa_2, \phi e_2 \phi \phi$	$\kappa_1, \phi e_2 \phi \phi$	$\kappa_1, e_3 \phi l_3 l_4$	$\kappa_1, \phi e_4 \phi \phi$	$\kappa_1, \phi e_5 \phi \phi$
κ_2	$\kappa_2, e_1 \phi l_1 l_2$	$\kappa_2, \phi e_2 \phi \phi$	$\kappa_1, \phi e_3 \phi \phi$	$\kappa_2, \phi e_4 \phi \phi$	$\kappa_2, \phi e_5 \phi \phi$

Table 3

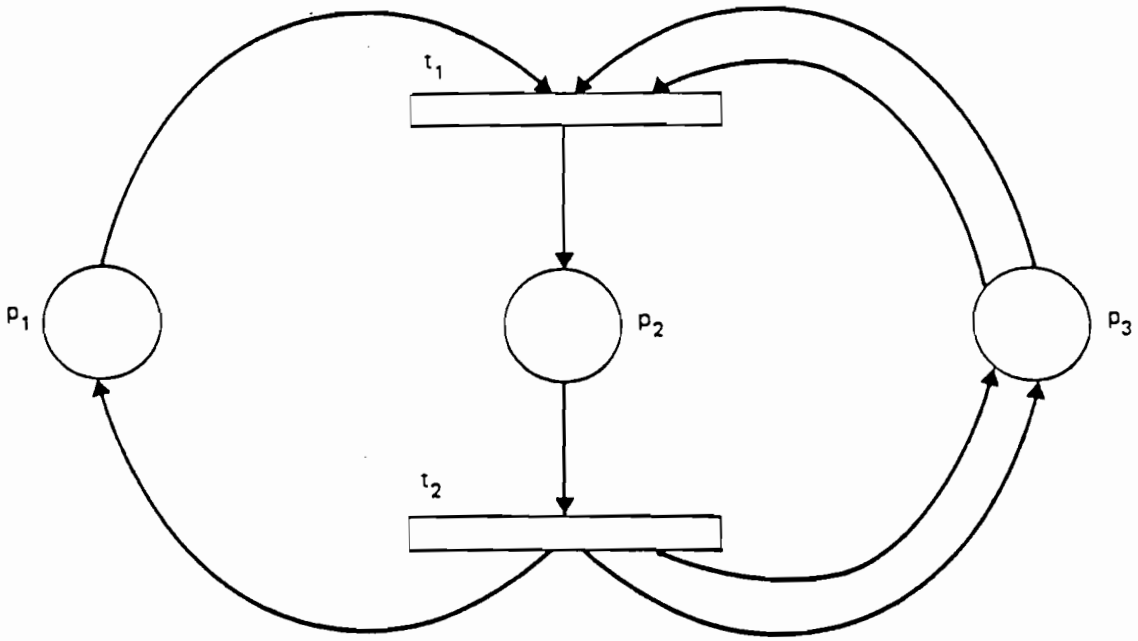

Figure 1

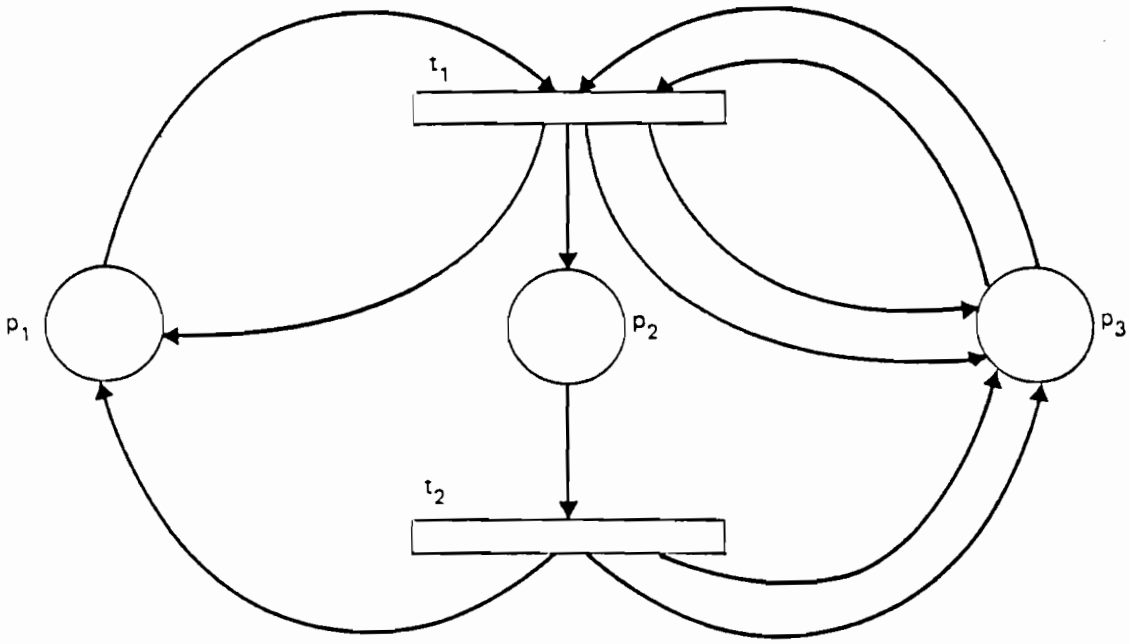

Figure 2