

Petri Nets: Properties, Analysis and Applications

TADAO MURATA, FELLOW, IEEE

Invited Paper

This is an invited tutorial-review paper on Petri nets—a graphical and mathematical modeling tool. Petri nets are a promising tool for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic.

The paper starts with a brief review of the history and the application areas considered in the literature. It then proceeds with introductory modeling examples, behavioral and structural properties, three methods of analysis, subclasses of Petri nets and their analysis. In particular, one section is devoted to marked graphs—the concurrent system model most amenable to analysis. In addition, the paper presents introductory discussions on stochastic nets with their application to performance modeling, and on high-level nets with their application to logic programming. Also included are recent results on reachability criteria. Suggestions are provided for further reading on many subject areas of Petri nets.

I. INTRODUCTION

Petri nets are a graphical and mathematical modeling tool applicable to many systems. They are a promising tool for describing and studying information processing systems that are characterized as being concurrent, asynchronous, distributed, parallel, nondeterministic, and/or stochastic. As a graphical tool, Petri nets can be used as a visual-communication aid similar to flow charts, block diagrams, and networks. In addition, tokens are used in these nets to simulate the dynamic and concurrent activities of systems. As a mathematical tool, it is possible to set up state equations, algebraic equations, and other mathematical models governing the behavior of systems. Petri nets can be used by both practitioners and theoreticians. Thus, they provide a powerful medium of communication between them: practitioners can learn from theoreticians how to make their models more methodical, and theoreticians can learn from practitioners how to make their models more realistic.

Historically speaking, the concept of the Petri net has its origin in Carl Adam Petri's dissertation [1], submitted in 1962

to the faculty of Mathematics and Physics at the Technical University of Darmstadt, West Germany. The dissertation was prepared while C. A. Petri worked as a scientist at the University of Bonn. Petri's work [1], [2] came to the attention of A. W. Holt, who later led the Information System Theory Project of Applied Data Research, Inc., in the United States. The early developments and applications of Petri nets (or their predecessor) are found in the reports [3]–[8] associated with this project, and in the Record [9] of the 1970 Project MAC Conference on Concurrent Systems and Parallel Computation. From 1970 to 1975, the Computation Structure Group at MIT was most active in conducting Petri-net related research, and produced many reports and theses on Petri nets. In July 1975, there was a conference on Petri Nets and Related Methods at MIT, but no conference proceedings were published. Most of the Petri-net related papers written in English before 1980 are listed in the annotated bibliography of the first book [10] on Petri nets. More recent papers up until 1984 and those works done in Germany and other European countries are annotated in the appendix of another book [11]. Three tutorial articles [12]–[14] provide a complementary, easy-to-read introduction to Petri nets.

Since the late-1970's, the Europeans have been very active in organizing workshops and publishing conference proceedings on Petri nets. In October 1979, about 135 researchers mostly from European countries assembled in Hamburg, West Germany, for a two-week advanced course on General Net Theory of Processes and Systems. The 17 lectures given in this course were published in its proceedings [15], which is currently out of print. The second advanced course was held in Bad Honnef, West Germany, in September 1986. The proceedings [16], [17] of this course contain 34 articles, including two recent articles by C. A. Petri; one [18] is concerned with his axioms of concurrency theory and the other [19] with his suggestions for further research. The first European Workshop on Applications and Theory of Petri Nets was held in 1980 at Strasbourg, France. Since then, this series of workshops has been held every year at different locations in Europe: 1981, Bad Honnef, West Germany; 1982, Varenna, Italy; 1983, Toulouse, France; 1984, Aarhus, Denmark; 1985, Espoo, Finland; 1986, Oxford, Great

Manuscript received May 20, 1988; revised November 4, 1988. This work was supported by the National Science Foundation under Grant DMC-8510208.

The author is with the Department of Electrical Engineering and Computer Science, University of Illinois, Chicago, IL 60680, USA.
IEEE Log Number 8926700.

0018-9219/89/0400-0541\$01.00 © 1989 IEEE

Britain; 1987, Zaragoza, Spain; 1988, Venice, Italy; and 1989, Bad Honnef, West Germany (planned). The distribution of the proceedings of these workshops is limited to mostly the workshop participants. However, selected papers from these workshops and other articles have been published by Springer-Verlag as *Advances in Petri Nets* [20]–[25]. The 1987 volume [24] contains the most comprehensive bibliography of Petri nets [26] listing 2074 entries published from 1962 to early 1987. The “recent publications” section of *Petri Net Newsletter* [27] lists short abstracts of recent publications three times a year, and is a good source of information about the most recent Petri net literature.

In July 1985, another series of international workshops was initiated. This series places emphasis on timed and stochastic nets and their applications to performance evaluation. The first international workshop on timed Petri nets was held in Torino, Italy, in July 1985; the second was held in Madison, Wisconsin, in August 1987; the third is to be held in Kyoto, Japan, in December 1989; and the fourth is planned in Australia in 1991. The proceedings of the first two workshops [28], [29] are available from the IEEE Computer Society Press.

The above is a brief history of Petri nets. Now, we look at some application areas considered in the literature. Petri nets have been proposed for a very wide variety of applications. This is due to the generality and permissiveness inherent in Petri nets. They can be applied informally to any area or system that can be described graphically like flow charts and that needs some means of representing parallel or concurrent activities. However, careful attention must be paid to a tradeoff between modeling generality and analysis capability. That is, the more general the model, the less amenable it is to analysis. In fact, a major weakness of Petri nets is the complexity problem, i.e., Petri-net-based models tend to become too large for analysis even for a modest-size system. In applying Petri nets, it is often necessary to add special modifications or restrictions suited to the particular application. Two successful application areas are performance evaluation [28]–[50] and communication protocols [51]–[62]. Promising areas of applications include modeling and analysis of distributed-software systems [63]–[71], distributed-database systems [72]–[75], concurrent and parallel programs [76]–[92], flexible manufacturing/industrial control systems [93]–[100], discrete-event systems [101]–[103], multiprocessor memory systems [30], [104], [105], data-flow computing systems [106]–[108], fault-tolerant systems [109]–[114], programmable logic and VLSI arrays [115]–[120], asynchronous circuits and structures [121]–[129], compiler and operating systems [130], [131], office-information systems [132]–[135], formal languages [136]–[142], and logic programs [143]–[150]. Other interesting applications considered in the literature are local-area networks [151]–[153], legal systems [154], human factors [155], [156], neural networks [157], [158], digital filters [159]–[161], and decision models [162].

The use of computer-aided tools is a necessity for practical applications of Petri nets. Most Petri-net research groups have their own software packages and tools to assist the drawing, analysis, and/or simulation of various applications. A recent article [163] provides a good overview of typical Petri-net tools existing as of 1986. Some of these tools and their applications are discussed in details in references [164] through [170].

The rest of this paper consists of the following topics. Section II discusses informally the transition enabling and firing rule with and without capacity constraints. Several introductory modeling examples are given in Section III to illustrate modeling capabilities and concepts such as conflict (choice or decision), concurrency, synchronization, etc. Section IV describes behavioral or marking-dependent properties that can be studied using Petri nets. Section V presents three methods of analysis: the coverability tree, matrix equations, and reduction techniques. Section VI is concerned with subclasses of Petri nets and their analysis. In-depth analysis and synthesis methods are given in Section VII for one of the subclasses known as marked graphs. Structural or marking-independent properties are discussed in Section VIII. Section IX presents an introduction to timed nets, stochastic nets, and high-level nets, together with their applications. Concluding remarks are given in Section X.

II. TRANSITION ENABLING AND FIRING

In this section, we give the only rule one has to learn about Petri-net theory: the rule for transition enabling and firing. Although this rule appears very simple, its implication in Petri-net theory is very deep and complex.

A Petri net is a particular kind of directed graph, together with an initial state called the *initial marking*, M_0 . The underlying graph N of a Petri net is a directed, weighted, bipartite graph consisting of two kinds of nodes, called *places* and *transitions*, where arcs are either from a place to a transition or from a transition to a place. In graphical representation, places are drawn as circles, transitions as bars or boxes. Arcs are labeled with their weights (positive integers), where a k -weighted arc can be interpreted as the set of k parallel arcs. Labels for unity weight are usually omitted. A *marking* (state) assigns to each place a nonnegative integer. If a marking assigns to place p a nonnegative integer k , we say that p is *marked with k tokens*. Pictorially, we place k black dots (tokens) in place p . A marking is denoted by M , an m -vector, where m is the total number of places. The p th component of M , denoted by $M(p)$, is the number of tokens in place p .

In modeling, using the concept of conditions and events, places represent conditions, and transitions represent events. A transition (an event) has a certain number of *input* and *output places* representing the pre-conditions and post-conditions of the event, respectively. The presence of a token in a place is interpreted as holding the truth of the condition associated with the place. In another interpretation, k tokens are put in a place to indicate that k data items or resources are available. Some typical interpretations of transitions and their input places and output places are shown in Table 1. A formal definition of a Petri net is given in Table 2.

Table 1 Some Typical Interpretations of Transitions and Places

Input Places	Transition	Output Places
Preconditions	Event	Postconditions
Input data	Computation step	Output data
Input signals	Signal processor	Output signals
Resources needed	Task or job	Resources released
Conditions	Clause in logic	Conclusion(s)
Buffers	Processor	Buffers

Table 2 Formal Definition of a Petri Net

A Petri net is a 5-tuple, $PN = (P, T, F, W, M_0)$ where:

$P = \{p_1, p_2, \dots, p_m\}$ is a finite set of places,
 $T = \{t_1, t_2, \dots, t_n\}$ is a finite set of transitions,
 $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs (flow relation),
 $W: F \rightarrow \{1, 2, 3, \dots\}$ is a weight function,
 $M_0: P \rightarrow \{0, 1, 2, 3, \dots\}$ is the initial marking,
 $P \cap T = \emptyset$ and $P \cup T \neq \emptyset$.

A Petri net structure $N = (P, T, F, W)$ without any specific initial marking is denoted by N .

A Petri net with the given initial marking is denoted by (N, M_0) .

The behavior of many systems can be described in terms of system states and their changes. In order to simulate the dynamic behavior of a system, a state or marking in a Petri nets is changed according to the following *transition (firing) rule*:

- 1) A transition t is said to be *enabled* if each input place p of t is marked with at least $w(p, t)$ tokens, where $w(p, t)$ is the weight of the arc from p to t .
- 2) An enabled transition may or may not fire (depending on whether or not the event actually takes place).
- 3) A firing of an enabled transition t removes $w(p, t)$ tokens from each input place p of t , and adds $w(t, p)$ tokens to each output place p of t , where $w(t, p)$ is the weight of the arc from t to p .

A transition without any input place is called a *source transition*, and one without any output place is called a *sink transition*. Note that a source transition is unconditionally enabled, and that the firing of a sink transition consumes tokens, but does not produce any.

A pair of a place p and a transition t is called a *self-loop* if p is both an input and output place of t . A Petri net is said to be *pure* if it has no self-loops. A Petri net is said to be *ordinary* if all of its arc weights are 1's.

Example 1: The above transition rule is illustrated in Fig. 1 using the well-known chemical reaction: $2H_2 + O_2 \rightarrow 2H_2O$. Two tokens in each input place in Fig. 1(a) show that two units of H_2 and O_2 are available, and the transition t is enabled. After firing t , the marking will change to the one shown in Fig. 1(b), where the transition t is no longer enabled. □

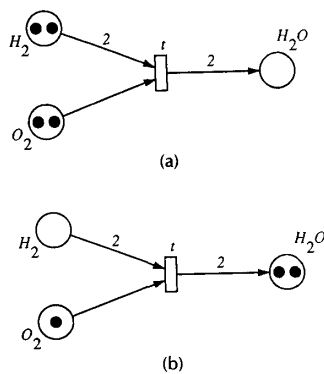


Fig. 1. Example 1: An illustration of a transition (firing) rule: (a) The marking before firing the enabled transition t . (b) The marking after firing t , where t is disabled.

For the above rule of transition enabling, it is assumed that each place can accommodate an unlimited number of tokens. Such a Petri net is referred to as an *infinite capacity net*. For modeling many physical systems, it is natural to consider an upper limit to the number of tokens that each place can hold. Such a Petri net is referred to as a *finite capacity net*. For a finite capacity net (N, M_0) , each place p has an associated capacity $K(p)$, the maximum number of tokens that p can hold at any time. For finite capacity nets, for a transition t to be enabled, there is an additional condition that the number of tokens in each output place p of t cannot exceed its capacity $K(p)$ after firing t .

This rule with the capacity constraint is called the *strict transition rule*, whereas the rule without the capacity constraint is called the *(weak) transition rule*. Given a finite capacity net (N, M_0) , it is possible to apply either the strict transition rule to the given net (N, M_0) or, equivalently, the weak transition rule to a transformed net (N', M'_0) , the net obtained from (N, M_0) by the following *complementary-place transformation*, where it is assumed that N is pure.

Step 1: Add a complementary place p' for each place p , where the initial marking of p' is given by $M'_0(p') = K(p) - M_0(p)$.

Step 2: Between each transition t and some complementary places p' , draw new arcs (t, p') or (p', t) where $w(t, p') = w(p, t)$ and $w(p', t) = w(t, p)$, so that the sum of tokens in place p and its complementary place p' equals its capacity $K(p)$ for each place p , before and after firing the transition t .

Example 2: Let us apply the strict transition rule to the finite-capacity net (N, M_0) shown in Fig. 2(a). At the initial marking $M_0 = (1 \ 0)$, the only enabled transition is t_1 . After firing t_1 , we have $M_1 = (2 \ 0)$, where only t_2 and t_3 are enabled. M_1 changes to $M_2 = (0 \ 0)$ after firing t_2 , or to $M_3 = (0 \ 1)$ after firing t_3 . Continuing this process, it is easy to draw the (reachability) graph shown in Fig. 2(c), which shows all possible markings and all possible firings at each marking. Now, let us see how the net (N, M_0) shown in Fig. 2(a) is transformed by the complementary-place transformation into the net (N', M'_0) shown in Fig. 2(b). The first step is to add the two complementary places p'_1 and p'_2 with their initial markings $M'_0(p'_1) = K(p_1) - M_0(p_1) = 2 - 1 = 1$, and $M'_0(p'_2) = K(p_2) - M_0(p_2) = 1 - 0 = 1$. The next step is to add new arcs between each transition t and some complementary places, so as to keep the sum of tokens in each pair of places p_i and p'_i the same and equal to $K(p_i)$, $i = 1, 2$, before and after firing t . For example, since $w(t_1, p_1) = 1$, we have $w(p'_1, t_1) = 1$. Similarly, $w(t_3, p'_1) = w(p_1, t_3) = 2$ and $w(p'_2, t_3) = w(t_3, p_2) = 1$, since firing t_3 removes two tokens from p_1 and adds one token in p_2 (we draw the two-weight arc from t_3 to p'_1 and the unit-weight arc from p'_2 to t_3). Likewise, two additional arcs (t_2, p'_1) and (t_4, p'_2) are drawn to obtain the net (N', M'_0) shown in Fig. 2(b). In a similar manner, as illustrated for (N, M_0) , it is easy to draw the reachability graph for the net (N', M'_0) . It is also easy to verify that the two reachability graphs are isomorphic, and that the two nets (N, M_0) and (N', M'_0) are equivalent with respect to the behavior of all possible firing sequences. □

The above discussions may be summarized in the following theorem.

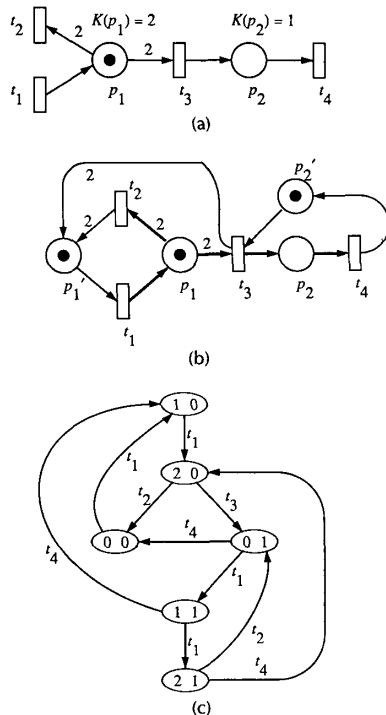


Fig. 2. Example 2: An illustration of the complementary-place transformation: (a) A finite-capacity net (N, M_0) . (b) The net (N', M'_0) after the transformation. (c) The reachability graph for the net (N, M_0) shown in (a).

Theorem 1: Let (N, M_0) be a pure finite-capacity net, where the strict transition rule is to be applied. Let (N', M'_0) be the net obtained from (N, M_0) by the complementary-place transformation, where the weak transition rule is applicable to (N', M'_0) . Then the two nets (N, M_0) and (N', M'_0) are equivalent in the sense that both have the same set of all possible firing sequences. \square

In view of Theorem 1, every pure finite-capacity net (N, M_0) can be transformed into an equivalent net (N', M'_0) , where the weak transition rule is applicable, and thus we only need consider the weak-transition rule. Therefore, unless otherwise stated, we consider only infinite-capacity nets with the weak-transition rule in the rest of this paper. The reason is that all properties associated with a finite-capacity net can be discussed in terms of those with an infinite-capacity net using the complementary-place transformation.

In Theorem 1, it is assumed that a Petri net be pure to avoid confusion since there are many different interpretations of the enabling condition for a self-loop in a finite capacity net [171]. But this is not a real restriction, because a self-loop can be "refined" or transformed into a loop by introducing a dummy pair of a transition and a place, as is illustrated in Fig. 3.

III. INTRODUCTORY MODELING EXAMPLES

In this section, several simple examples are given to introduce the reader to some basic concepts of Petri nets that are useful in modeling.

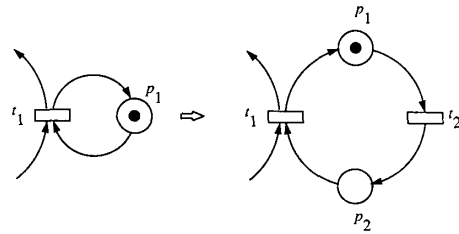


Fig. 3. Transformation of a self-loop to a loop.

A. Finite-State Machines

Finite-state machines or their state diagrams can be equivalently represented by a subclass of Petri nets. As an example of a finite-state machine, consider a vending machine which accepts either nickels or dimes and sells 15¢ or 20¢ candy bars. For simplicity, suppose the vending machine can hold up to 20¢. Then, the state diagram of the machine can be represented by the Petri net shown in Fig. 4, where the five states are represented by the five places

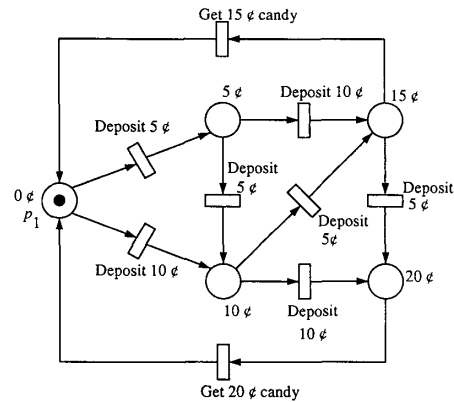


Fig. 4. A Petri net (a state machine) representing the state diagram of a vending machine, where coin return transitions are omitted.

labeled with 0¢, 5¢, 10¢, 15¢, and 20¢, and transformations from one state to another state are shown by transitions labeled with input conditions, such as "deposit 5¢." The initial state is indicated by initially putting a token in the place p_1 , with a 0¢ label in this example. Note that each transition in this net has exactly one incoming arc and exactly one outgoing arc. The subclass of Petri nets with this property is known as *state machines*. Any finite-state machine (or its state diagram) can be modeled with a state machine. The structure of the place p_1 having two (or more) output transitions t_1 and t_2 , as shown in Fig. 5, is referred to as a *conflict*, *decision*, or *choice*, depending on applications. State machines allow the representation of decisions, but not the synchronization of parallel activities.

B. Parallel Activities

Parallel activities or concurrency can be easily expressed in terms of Petri nets. For example, in the Petri net shown in Fig. 6, the parallel or concurrent activities represented

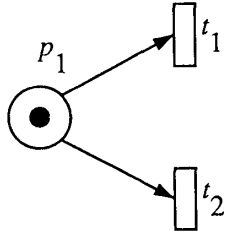


Fig. 5. A Petri-net structure called a conflict, choice, or decision. It is a structure exhibiting nondeterminism.

by transitions t_2 and t_3 begin at the firing of transition t_1 and end with the firing of transition t_4 . In general, two transitions are said to be *concurrent* if they are causally independent, i.e., one transition may fire before or after or in parallel with the other, as in the case of t_2 and t_3 in Fig. 6.

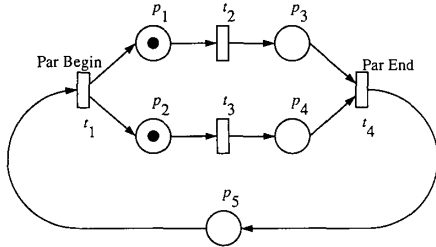


Fig. 6. A Petri net (a marked graph) representing deterministic parallel activities.

It has been pointed out [172] that concurrency can be regarded as a binary relation (denoted by co on the set of events $A = \{e_1, e_2, \dots\}$) which is 1) reflexive ($e_i \cdot co \cdot e_i$) and 2) symmetric ($e_1 \cdot co \cdot e_2$ implies $e_2 \cdot co \cdot e_1$), 3) but not transitive ($e_1 \cdot co \cdot e_2$ and $e_2 \cdot co \cdot e_3$ do not necessarily imply $e_1 \cdot co \cdot e_3$). For example, one may drive a car (event e_1) or walk (event e_3) while singing (event e_2), but one cannot drive and walk concurrently.

Note that each place in the net shown in Fig. 6 has exactly one incoming arc and exactly one outgoing arc. The subclass of Petri nets with this property is known as *marked graphs*. Marked graphs allow representation of concurrency but not decisions (conflicts).

Two events e_1 and e_2 are in conflict if either e_1 or e_2 can occur but not both, and they are concurrent if both events can occur in any order without conflicts. A situation where conflict and concurrency are mixed is called a *confusion*. Two types of confusion are shown in Fig. 7. Fig. 7(a) shows a symmetric confusion, since two events t_1 and t_2 are concurrent while each of t_1 and t_2 is in conflict with event t_3 . Fig. 7(b) shows an asymmetric confusion, where t_1 is concurrent with t_2 but will be in conflict with t_3 if t_2 fires first.

C. Dataflow Computation

Petri nets can be used to represent not only the flow of control but also the flow of data. The net shown in Fig. 8 is a Petri-net representation of a dataflow computation. A *dataflow computer* is one in which instructions are enabled for execution by the arrival of their operands, and may be

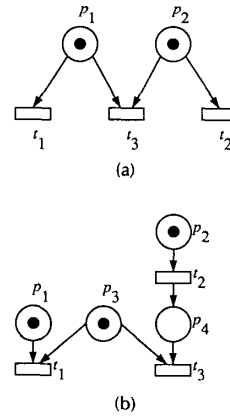


Fig. 7. Two types of a confusion. (a) Symmetric confusion: t_1 and t_2 are concurrent as well as in conflict with t_3 . (b) Asymmetric confusion: t_1 is concurrent with t_2 but will be in conflict with t_3 , if t_2 fires before t_1 .

executed concurrently. In the Petri-net representation of a dataflow computation, tokens denote the values of current data as well as the availability of data. In the net shown in Fig. 8, the instructions represented by transitions t_1 and

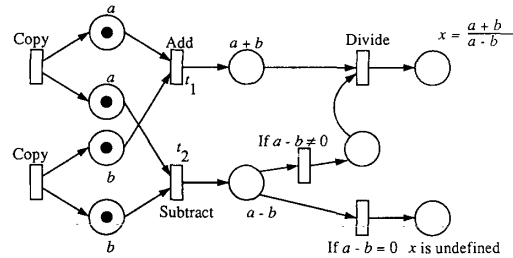


Fig. 8. A Petri net showing a dataflow computation for $x = (a + b)/(a - b)$.

t_2 can be executed concurrently and deposit the resulting data $(a + b)$ or $(a - b)$ in the respective output places.

D. Communication Protocols

Communication protocols are another area where Petri nets can be used to represent and specify essential features of a system. The liveness and safeness properties (see Section V) of a Petri net are often used as correctness criteria in communication protocols. The Petri net shown in Fig. 9 is a very simple model of a communication protocol between two processes. Figure 10 shows the Petri-net representation of a nondeterministic wait process where t_{r1} , t_{r2} , or t_{out} fires if response 1, response 2, or no response is received before a specified time (t_{out}), respectively.

E. Synchronization Control

In a multiprocessor or distributed-processing system, resources and information are shared among several processors. This sharing must be controlled or synchronized to insure the correct operation of the overall system. Petri nets have been used to model a variety of synchronization

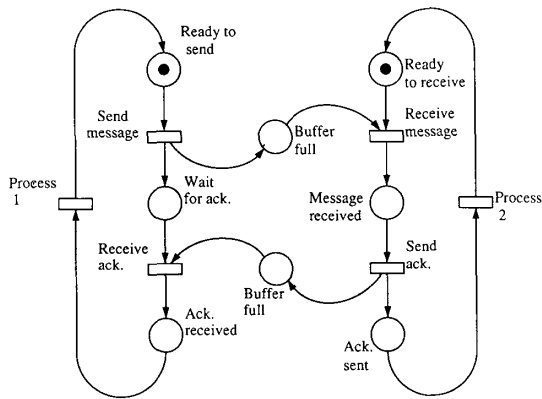


Fig. 9. A simplified model of a communication protocol.

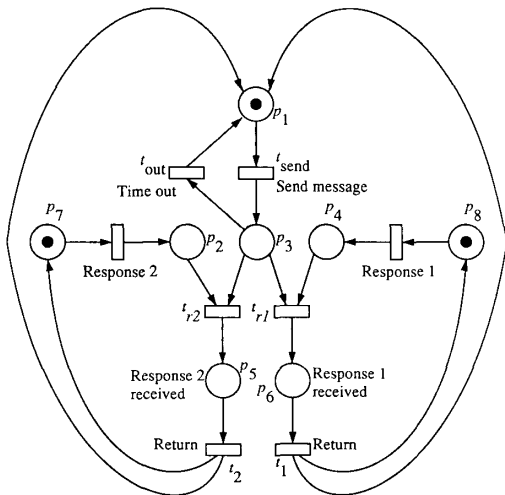


Fig. 10. A Petri-net representation of a nondeterministic wait process.

mechanisms, including the mutual exclusion, readers-writers, and producers-consumers problems. The Petri net shown in Fig. 11 represents a readers-writers synchronization, where the k tokens in place p_1 represent k processes (programs) which may read and write in a shared memory represented by place p_3 . Up to k processes may be reading

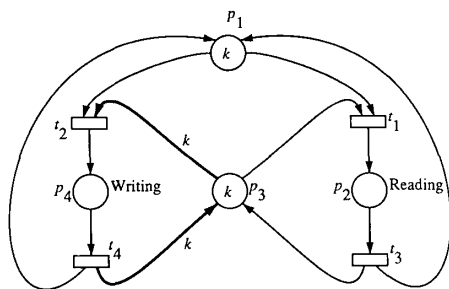


Fig. 11. A Petri-net representation of a readers-writers system.

concurrently, but when one process is writing, no other process can be reading or writing. It is easily verified that up to k tokens (processes) may be in place p_2 (reading) if no token is in place p_4 , and that only one token (process) can be in place p_4 (writing) since all k tokens in place p_3 will be removed through the k -weight arc when t_2 fires once. This Petri net will be analyzed in Example 21 in Section VIII.

F. Producers-Consumers System with Priority

The net shown in Fig. 12 represents a producers-consumers system with priority, i.e., consumer A has priority over consumer B in the sense that A can consume as long

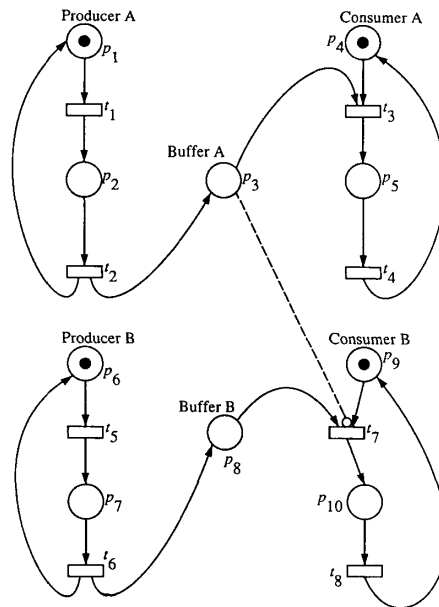


Fig. 12. An extended Petri-net representation of a producers-consumers system with priority.

as buffer A has items (tokens), but B can consume only if buffer A is empty and buffer B has items (tokens). It has been shown [173] that this system cannot be modeled without introducing a new kind of arc called an *inhibitor arc*. An inhibitor arc connects a place to a transition and is represented by a dashed line terminating with a small circle instead of an arrowhead at the transition, like the arc from p_3 to t_7 in Fig. 12. The inhibitor arc disables the transition when the input place has a token and enables the transition when the input place has no token and other (normal) input places have at least one token per arc weight. No tokens are moved through an inhibitor arc when the transition fires. A class of Petri nets with inhibitor arcs is referred to as *extended Petri nets*. The introduction of inhibitor arcs adds the ability to test "zero" (i.e., absence of tokens in a place) and increases the modeling power of Petri nets to the level of Turing machines [10].

G. Formal Languages

When the transitions in a Petri net are labeled with a set of not necessarily distinct symbols, a sequence of transition

firings generates a string of symbols. The set of strings generated by all possible firing sequences defines a formal language called a *Petri-net language*. For example, consider all possible sequences of transition firings in the labeled Petri net shown in Fig. 13 [10]. It is easy to see that λ (null symbol),

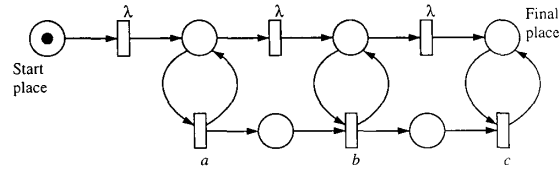


Fig. 13. A context-sensitive language $L(M_0) = \{a^n b^n c^n \mid n \geq 0\}$ is generated by this labeled Petri net.

$abc, aabbcc, aaabbbccc, \dots$ are strings of symbols generated by all of the possible firing sequences starting from the initial marking with one token in the "start place" and terminating when all the transitions are disabled. From this, it can be seen that the language generated by this net is given by $L(M_0) = \{a^n b^n c^n \mid n \geq 0\}$ (a context-sensitive Petri-net language). Since every finite-state machine can be modeled by a Petri net, every regular language is a Petri-net language. It has been shown that all Petri-net languages are context-sensitive languages [10].

H. Multiprocessor Systems

The Petri net shown in Fig. 14 is a model for a multiprocessor system with five processors, three common memories and two buses [30], [31]. Place p_1 contains tokens rep-

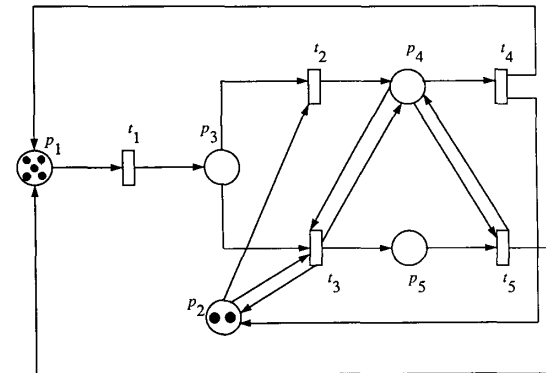


Fig. 14. A Petri-net model of a multiprocessor system, where tokens in p_1 represent active processors, p_2 available buses, p_3 , p_4 , and p_5 processors waiting for, having access to, queued for common memories, respectively.

resenting processors executing in their private memory, and p_2 contains tokens representing free buses. Transition t_1 represents the issuing of access requests, and p_3 contains requests that have not yet been served. Tokens in p_4 represent processors having access to common memories. Tokens in p_5 represent processors requesting the same common memory that has been accessed by a token (processor) in p_4 . Firing t_5 represents the end of the access to the memory for which processors in p_5 are queued. Firing

t_4 represents the end of the access to a memory for which there is no outstanding request (i.e., t_4 is enabled when $M(p_4) - U[M(p_5)] > 0$, where $U[x] = 1$ for $x > 0$ and $U[x] = 0$ otherwise.) The two transitions t_2 and t_3 model the memory choice: firing t_3 corresponds to choosing the memory that is being accessed by the processor in p_4 . The choice of any other memory corresponds to the firing of t_2 .

Actually, the net model shown in Fig. 14 can represent a two-bus multiprocessor system with any number of processors and memories. A generalized stochastic net version of this and more detailed models has been used for performance study of multiprocessor architectures [30], [31].

IV. BEHAVIORAL PROPERTIES

After modeling systems with Petri nets, an obvious question is "What can we do with the models?" A major strength of Petri nets is their support for analysis of many properties and problems associated with concurrent systems. Two types of properties can be studied with a Petri-net model: those which depend on the initial marking, and those which are independent of the initial marking. The former type of properties is referred to as *marking-dependent* or *behavioral* properties, whereas the latter type of properties is called *structural* properties. In this section, we discuss only basic behavioral properties and their analysis problems. Structural properties and their analysis will be considered in Section VIII.

A. Reachability

Reachability is a fundamental basis for studying the dynamic properties of any system. The firing of an enabled transition will change the token distribution (marking) in a net according to the transition rule described in Section II. A sequence of firings will result in a sequence of markings. A marking M_n is said to be *reachable* from a marking M_0 if there exists a sequence of firings that transforms M_0 to M_n . A *firing* or *occurrence sequence* is denoted by $\sigma = M_0 t_1 M_1 t_2 M_2 \dots t_n M_n$ or simply $\sigma = t_1 t_2 \dots t_n$. In this case, M_n is reachable from M_0 by σ and we write $M_0[\sigma > M_n]$. The set of all possible markings reachable from M_0 in a net (N, M_0) is denoted by $R(N, M_0)$ or simply $R(M_0)$. The set of all possible firing sequences from M_0 in a net (N, M_0) is denoted by $L(N, M_0)$ or simply $L(M_0)$.

Now, the *reachability problem* for Petri nets is the problem of finding if $M_n \in R(M_0)$ for a given marking M_n in a net (N, M_0) . In some applications, one may be interested in the markings of a subset of places and not care about the rest of places in a net. This leads to a *submarking reachability problem* which is the problem of finding if $M'_n \in R(M_0)$, where M'_n is any marking whose restriction to a given subset of places agrees with that of a given marking M_n . It has been shown that the reachability problem is decidable [174], [175] although it takes at least exponential space (and time) to verify in the general case [275]. However, the equality problem [138], [176], [177] is undecidable, i.e., there is no algorithm for determining if $L(N, M_0) = L(N', M'_0)$ for any two Petri nets N and N' .

B. Boundedness

A Petri net (N, M_0) is said to be *k-bounded* or simply *bounded* if the number of tokens in each place does not exceed a finite number k for any marking reachable from

M_0 , i.e., $M(p) \leq k$ for every place p and every marking $M \in R(M_0)$. A Petri net (N, M_0) is said to be *safe* if it is 1-bounded. For example, the nets shown in Figs. 2(b), 4, 6, and 9 are all bounded; in particular, the net in Fig. 2(b) is 2-bounded, and the rest of the nets are safe. Places in a Petri net are often used to represent buffers and registers for storing intermediate data. By verifying that the net is bounded or safe, it is guaranteed that there will be no overflows in the buffers or registers, no matter what firing sequence is taken.

C. Liveness

The concept of liveness is closely related to the complete absence of deadlocks in operating systems. A Petri net (N, M_0) is said to be *live* (or equivalently M_0 is said to be a *live marking* for N) if, no matter what marking has been reached from M_0 , it is possible to ultimately fire any transition of the net by progressing through some further firing sequence. This means that a live Petri net guarantees deadlock-free operation, no matter what firing sequence is chosen. Examples of live Petri nets are shown in Figs. 4, 6, and 9. On the other hand, the Petri nets shown in Figs. 15 and 16 are not

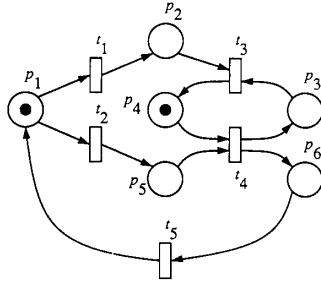


Fig. 15. A safe, nonlive Petri net. But it is strictly L1-live.

live. These nets are not live since no transitions can fire if t_1 fires first in both cases.

Liveness is an ideal property for many systems. However, it is impractical and too costly to verify this strong property for some systems such as the operating system of a large computer. Thus, we relax the liveness condition and define different levels of liveness as follows [8], [178]. A transition t in a Petri net (N, M_0) is said to be:

- 0) *dead (L0-live)* if t can never be fired in any firing sequence in $L(M_0)$.
- 1) *L1-live (potentially firable)* if t can be fired at least once in some firing sequence in $L(M_0)$.
- 2) *L2-live* if, given any positive integer k , t can be fired at least k times in some firing sequence in $L(M_0)$.
- 3) *L3-live* if t appears infinitely, often in some firing sequence in $L(M_0)$.
- 4) *L4-live or live* if t is L1-live for every marking M in $R(M_0)$.

A Petri net (N, M_0) is said to be *Lk-live* if every transition in the net is Lk-live, $k = 0, 1, 2, 3, 4$. L4-liveness is the strongest and corresponds to the liveness defined earlier. It is easy to see the following implications: L4-liveness \Rightarrow L3-liveness \Rightarrow L2-liveness \Rightarrow L1-liveness, where \Rightarrow means "implies." We say that a transition is *strictly Lk-live* if it is Lk-live but not $L(k+1)$ -live, $k = 1, 2, 3$.

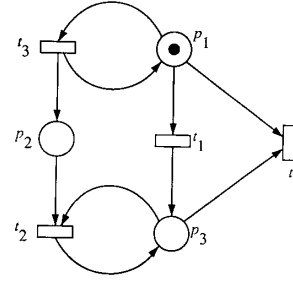


Fig. 16. Transitions t_0 , t_1 , t_2 , and t_3 are dead (L0-live), L1-live, L2-live, and L3-live, respectively.

Example 3: The Petri net shown in Fig. 15 is strictly L1-live since each transition can be fired exactly once in the order of t_2 , t_4 , t_5 , t_1 , and t_3 . The transitions t_0 , t_1 , t_2 , and t_3 in Fig. 16 are L0-live (dead), L1-live, L2-live, and L3-live, respectively, all strictly. \square

D. Reversibility and Home State

A Petri net (N, M_0) is said to be *reversible* if, for each marking M in $R(M_0)$, M_0 is reachable from M . Thus, in a reversible net one can always get back to the initial marking or state. In many applications, it is not necessary to get back to the initial state as long as one can get back to some (home) state. Therefore, we relax the reversibility condition and define a home state. A marking M' is said to be a *home state* if, for each marking M in $R(M_0)$, M' is reachable from M .

Example 4: Note that the above three properties (boundedness, liveness, and reversibility) are independent of each other. For example, a reversible net can be live or not live and bounded or not bounded. Fig. 17 [179] shows examples of eight Petri nets for all possible combination of these three properties, where \bar{B} , \bar{L} , and \bar{R} denote the negations of boundedness (B), liveness (L), and reversibility (R).

E. Coverability

A marking M in a Petri net (N, M_0) is said to be *coverable* if there exists a marking M' in $R(M_0)$ such that $M'(p) \geq M(p)$ for each p in the net. Coverability is closely related to L1-liveness (potential firability). Let M be the minimum marking needed to enable a transition t . Then t is dead (not L1-live) if and only if M is not coverable. That is, t is L1-live if and only if M is coverable.

F. Persistence

A Petri net (N, M_0) is said to be *persistent* if, for any two enabled transitions, the firing of one transition will not disable the other. A transition in a persistent net, once it is enabled, will stay enabled until it fires. The notion of persistence is useful in the context of parallel program schemata [82] and speed-independent asynchronous circuits [122], [126]. Persistency is closely related to conflict-free nets [180], and a safe persistent net can be transformed into a marked graph by duplicating some transitions and places [45]. Note that all marked graphs are persistent, but not all persistent nets are marked graphs. For example, the net shown in Fig. 17(c) is persistent, but it is not a marked graph.

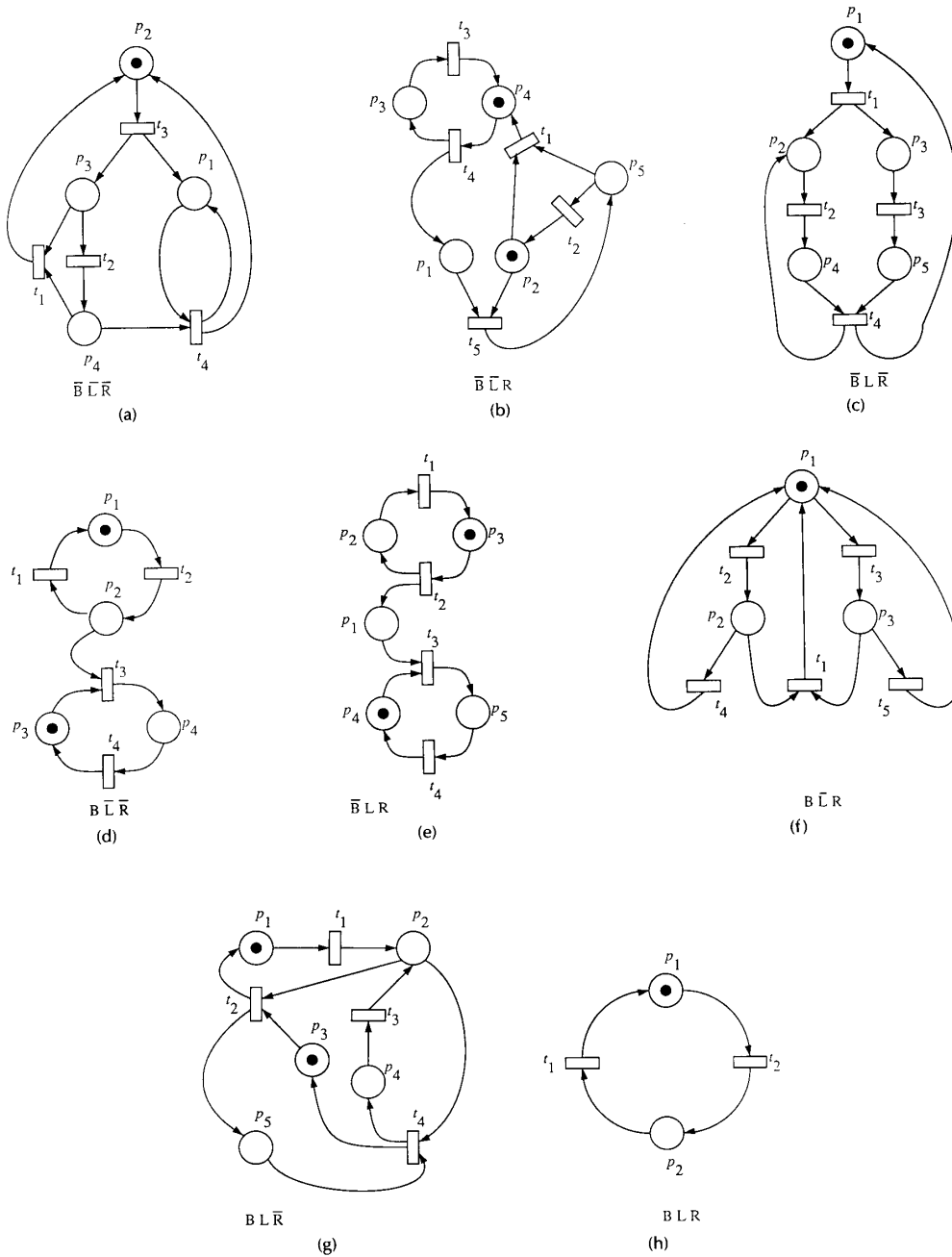


Fig. 17. Examples of Petri nets having all possible combinations of B (bounded), \bar{B} (unbounded), L (live), \bar{L} (nonlive), R (reversible), and \bar{R} (nonreversible) properties. (a) $\bar{B} \bar{L} \bar{R}$ (t_1 dead, p_1 unbounded). (b) $\bar{B} \bar{L} R$ (t_1 dead, p_1 unbounded). (c) $\bar{B} L \bar{R}$ (p_2 unbounded). (d) $\bar{B} \bar{L} \bar{R}$ (t_1, t_2, t_3, t_4 not L4-live). (e) $\bar{B} L R$ (p_1 unbounded). (f) $\bar{B} \bar{L} R$ (t_1 dead). (g) $B L \bar{R}$. (h) $B L R$.

G. Synchronic Distance

The notion of synchronic distances is a fundamental concept introduced by C. A. Petri [181]. It is a metric closely related to a degree of mutual dependence between two events in a condition/event system. We define the synchronic distance between two transitions t_1 and t_2 in a Petri

net (N, M_0) by

$$d_{12} = \max_{\sigma} |\bar{\sigma}(t_1) - \bar{\sigma}(t_2)| \quad (1)$$

where σ is a firing sequence starting at any marking M in $R(M_0)$ and $\bar{\sigma}(t_i)$ is the number of times that transition t_i , $i = 1, 2$ fires in σ . For example, in the net shown in Fig. 17(d) d_{12}

$= 1$, $d_{34} = 1$, $d_{13} = \infty$, etc. In the net shown in Fig. 6 transitions t_2 and t_3 represent two parallel events, and $d_{23} = 2$ because after firing t_3 there is a firing sequence $\sigma = t_2 \ t_4 \ t_1 \ t_2$ in which $\bar{\sigma}(t_2) = 2$ and $\bar{\sigma}(t_3) = 0$.

The synchronic distance given by (1) represents a well-defined metric for condition/event nets [184] and marked graphs. However, there are some difficulties when it is applied to more general class of Petri nets [182]. For further information on synchronic distances, the reader is referred to [105], [181]–[186].

H. Fairness

Many different notions of fairness have been proposed in the literature on Petri nets. We present here two basic fairness concepts: bounded-fairness and unconditional (global) fairness. Two transitions t_1 and t_2 are said to be in a *bounded-fair* (or *B-fair*) *relation* if the maximum number of times that either one can fire while the other is not firing is bounded. A Petri net (N, M_0) is said to be a *B-fair net* if every pair of transitions in the net are in a B-fair relation. A firing sequence σ is said to be *unconditionally* (globally) *fair* if it is finite or every transition in the net appears infinitely often in σ . A Petri net (N, M_0) is said to be an *unconditionally fair net* if every firing sequence σ from M in $R(M_0)$ is unconditionally fair.

There are some relationships between these two types of fairness. For example, every B-fair net is an unconditionally-fair net and every bounded unconditionally-fair net is a B-fair net [187]. The net shown in Fig. 17(h) is a B-fair net as well as an unconditionally fair net. The net shown in Fig. 17(d) is neither a B-fair net nor an unconditionally fair net since t_3 and t_4 will not appear in an infinite firing sequence $\sigma = t_2 \ t_1 \ t_2 \ t_1 \ \dots$. The unbounded net shown in Fig. 17(c) is an unconditionally fair net but not a B-fair net since there is no bound on the number of times that t_2 can fire without firing the others when the number of tokens in p_2 is unbounded. For further information on fairness, the reader is referred to [187]–[197], [211].

V. ANALYSIS METHODS

Methods of analysis for Petri nets may be classified into the following three groups: 1) the coverability (reachability) tree method, 2) the matrix-equation approach, and 3) reduction or decomposition techniques. The first method involves essentially the enumeration of all reachable markings or their coverable markings. It should be able to apply to all classes of nets, but is limited to "small" nets due to the complexity of the state-space explosion. On the other hand, matrix equations and reduction techniques are powerful but in many cases they are applicable only to special subclasses of Petri nets or special situations.

A. The Coverability Tree

Given a Petri net (N, M_0) , from the initial marking M_0 , we can obtain as many "new" markings as the number of the enabled transitions. From each new marking, we can again reach more markings. This process results in a tree representation of the markings. Nodes represent markings generated from M_0 (the root) and its successors, and each arc represents a transition firing, which transforms one marking to another.

The above tree representation, however, will grow infinitely large if the net is unbounded. To keep the tree finite, we introduce a special symbol ω , which can be thought of as "infinity." It has the properties that for each integer n , $\omega > n$, $\omega \pm n = \omega$ and $\omega \geq \omega$.

The coverability tree for a Petri net (N, M_0) is constructed by the following algorithm.

- Step 1) Label the initial marking M_0 as the root and tag it "new."
- Step 2) While "new" markings exist, do the following:
 - Step 2.1) Select a new marking M .
 - Step 2.2) If M is identical to a marking on the path from the root to M , then tag M "old" and go to another new marking.
 - Step 2.3) If no transitions are enabled at M , tag M "dead-end."
 - Step 2.4) While there exist enabled transitions at M , do the following for each enabled transition t at M :
 - Step 2.4.1) Obtain the marking M' that results from firing t at M .
 - Step 2.4.2) On the path from the root to M if there exists a marking M'' such that $M'(p) \geq M''(p)$ for each place p and $M' \neq M''$, i.e., M'' is coverable, then replace $M'(p)$ by ω for each p such that $M'(p) > M''(p)$.
 - Step 2.4.3) Introduce M' as a node, draw an arc with label t from M to M' , and tag M' "new."

Example: Consider the net shown in Fig. 16. For the initial marking $M_0 = (1 \ 0 \ 0)$, the two transitions t_1 and t_3 are enabled. Firing t_1 transforms M_0 to $M_1 = (0 \ 0 \ 1)$, which is a "dead-end" node, since no transitions are enabled at M_1 . Now, firing t_3 at M_0 results in $M_3 = (1 \ 1 \ 0)$, which covers $M_0 = (1 \ 0 \ 0)$. Therefore, the new marking is $M_3 = (1 \ \omega \ 0)$, where two transitions t_1 and t_3 are again enabled. Firing t_1 transforms M_3 to $M_4 = (0 \ \omega \ 1)$, from which t_2 can be fired, resulting in an "old" node $M_5 = M_4$. Firing t_3 at M_3 results in an "old" node $M_6 = M_3$. Thus, we have the coverability tree shown in Fig. 18(a). \square

Some of the properties that can be studied by using the coverability tree T for a Petri Net (N, M_0) are the following:

- 1) A net (N, M_0) is *bounded* and thus $R(M_0)$ is finite iff (if and only if) ω does not appear in any node labels in T .
- 2) A net (N, M_0) is *safe* iff only 0's and 1's appear in node labels in T .
- 3) A transition t is *dead* iff it does not appear as an arc label in T .
- 4) If M is reachable from M_0 , then there exists a node labeled M' such that $M \leq M'$.

For a bounded Petri net, the coverability tree is called the *reachability tree* since it contains all possible reachable markings. In this case, all the analysis problems discussed in Section IV can be solved by the reachability tree. The disadvantage is that this is an exhaustive method. However, in general, because of the information lost by the use of the symbol ω (which may represent only even or odd numbers, increasing or decreasing numbers, etc.), the reachability and liveness problems cannot be solved by the coverability-tree method alone. For example, the two different Petri nets shown in Fig. 19(a) and (b) [10] have the same coverability

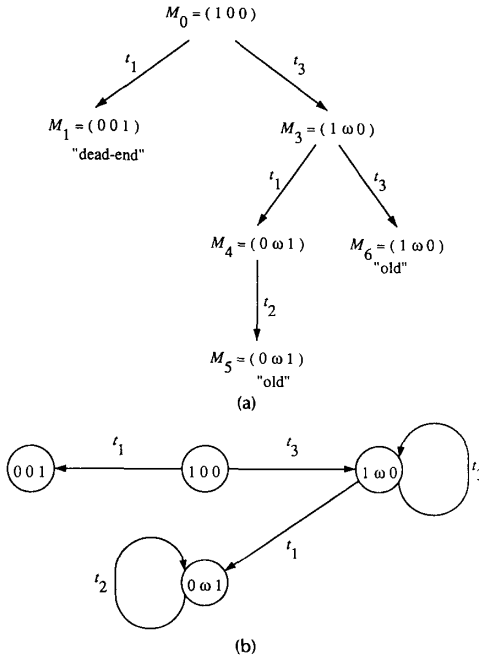


Fig. 18. (a) The coverability tree of the net shown in Fig. 16. (b) The coverability graph of the net shown in Fig. 16.

tree shown in Fig. 20(a). Yet, the net shown in Fig. 19(a) is a live Petri net, while the net shown in Fig. 19(b) is not live, since no transitions are enabled after firing t_1 , t_2 , and t_3 .

The *coverability graph* of a Petri net (N, M_0) is a labeled directed graph $G = (V, E)$. Its node set V is the set of all dis-

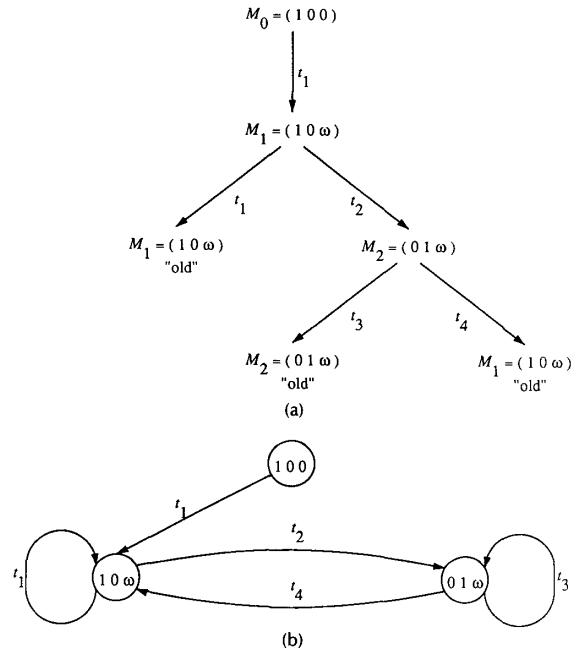


Fig. 20. (a) The coverability tree for both Petri nets shown in Fig. 19(a) and 19(b). (b) The coverability graph for the two nets shown in Fig. 19(a) and 19(b).

tinct labeled nodes in the coverability tree, and the arc set E is the set of arcs labeled with single transition t_k representing all possible single transition firings such that $M_i[t_k > M_j$, where M_i and M_j are in V . For example, the coverability graph for the nets shown in Fig. 19 is shown in Fig. 20(b). For a bounded Petri net, the coverability graph is referred to as the *reachability graph*, because the vertex set V becomes the same as the reachability set $R(M_0)$. An application of reachability graphs will be discussed in Section IX-B.

B. Incidence Matrix and State Equation

The dynamic behavior of many systems studied in engineering can be described by differential equations or algebraic equations. It would be nice if we could describe and analyze completely the dynamic behavior of Petri nets by some equations. In this spirit, we present matrix equations that govern the dynamic behavior of concurrent systems modeled by Petri nets. However, the solvability of these equations is somewhat limited, partly because of the non-deterministic nature inherent in Petri-net models and because of the constraint that solutions must be found as non-negative integers. Whenever matrix equations are discussed in this paper, it is assumed that a Petri net is pure or is made pure by adding a dummy pair of a transition and a place as is discussed in Section II (Fig. 3).

Incidence Matrix: For a Petri net N with n transitions and m places, the incidence matrix $A = [a_{ij}]$ is an $n \times m$ matrix of integers and its typical entry is given by

$$a_{ij} = a_{ij}^+ - a_{ij}^- \quad (2)$$

where $a_{ij}^+ = w(i, j)$ is the weight of the arc from transition i to its output place j and $a_{ij}^- = w(j, i)$ is the weight of the

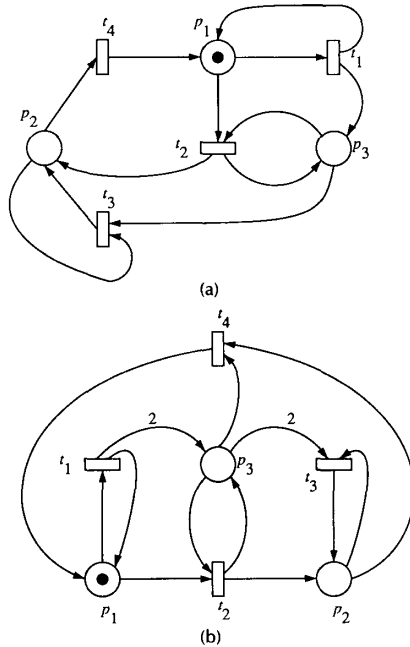


Fig. 19. Two Petri nets having the same coverability tree. (a) A live Petri net. (b) A nonlive Petri net.

arc to transition i from its input place j . We use A as the incidence matrix instead of its transpose A^T because A reduces to the well-known incidence matrix of a directed graph for marked graphs, a subclass of Petri nets.

It is easy to see from the transition rule described in Section II that a_{ij}^- , a_{ij}^+ , and a_{ij} , respectively, represent the number of tokens removed, added, and changed in place j when transition i fires once. Transition i is enabled at a marking M iff

$$a_{ij}^- \leq M(j), \quad j = 1, 2, \dots, m. \quad (3)$$

State Equation: In writing matrix equations, we write a marking M_k as an $m \times 1$ column vector. The j th entry of M_k denotes the number of tokens in place j immediately after the k th firing in some firing sequence. The k th firing or control vector u_k is an $n \times 1$ column vector of $n - 1$ 0's and one nonzero entry, a 1 in the i th position indicating that transition i fires at the k th firing. Since the i th row of the incidence matrix A denotes the change of the marking as the result of firing transition i , we can write the following state equation for a Petri net [198]:

$$M_k = M_{k-1} + A^T u_k, \quad k = 1, 2, \dots \quad (4)$$

Necessary Reachability Condition: Suppose that a destination marking M_d is reachable from M_0 through a firing sequence $\{u_1, u_2, \dots, u_d\}$. Writing the state equation (4) for $i = 1, 2, \dots, d$ and summing them, we obtain

$$M_d = M_0 + A^T \sum_{k=1}^d u_k \quad (5)$$

which can be rewritten as

$$A^T x = \Delta M \quad (6)$$

where $\Delta M = M_d - M_0$ and $x = \sum_{k=1}^d u_k$. Here x is an $n \times 1$ column vector of nonnegative integers and is called the *firing count vector*. The i th entry of x denotes the number of times that transition i must fire to transform M_0 to M_d . It is well known [199] that a set of linear algebraic equations (6) has a solution x iff ΔM is orthogonal to every solution y of its homogeneous system,

$$Ay = 0. \quad (7)$$

Let r be the rank of A , and partition A in the following form:

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{matrix} \uparrow r \\ \downarrow n-r \end{matrix} \quad (8)$$

where A_{12} is a nonsingular square matrix of order r . A set of $(m - r)$ linearly independent solutions y for (7) can be given as the $(m - r)$ rows of the following $(m - r) \times m$ matrix B_i :

$$B_i = [I_{\mu}; -A_{11}^T(A_{12}^T)^{-1}] \quad (9)$$

where I_{μ} is the identity matrix of order $\mu = m - r$. Note that $AB_i^T = 0$. That is, the vector space spanned by the row vectors of A is orthogonal to the vector space spanned by the row vectors of B_i . The matrix B_i corresponds to the fundamental circuit matrix [13] in the case of a marked graph. Now, the condition that ΔM is orthogonal to every solution for $Ay = 0$ is equivalent to the following condition:

$$B_i \Delta M = 0. \quad (10)$$

Thus, if M_d is reachable from M_0 , then the corresponding firing count vector x must exist and (10) must hold. Therefore, we have the following necessary condition for reachability in an unrestricted Petri net [198].

Theorem 2: If M_d is reachable from M_0 in a Petri net (N, M_0) , then $B_i \Delta M = 0$, where $\Delta M = M_d - M_0$ and B_i is given by (9). \square

The contrapositive of Theorem 2 provides the following sufficient condition for nonreachability.

Corollary 1: In a Petri net (N, M_0) , a marking M_d is not reachable from M_0 ($\neq M_d$) if their difference is a linear combination of the row vectors of B_i , that is,

$$\Delta M = B_i^T z \quad (11)$$

where z is a nonzero $\mu \times 1$ column vector.

Proof: If (11) holds, then $B_i \Delta M = B_i B_i^T z \neq 0$, since $z \neq 0$ and $B_i B_i^T$ is a $\mu \times \mu$ nonsingular matrix (because the rank of B_i is $\mu = m - r$). Therefore, by Theorem 2, M_d is not reachable from M_0 . \square

Example 5: For the Petri net shown in Fig. 21, the state equation (4) is illustrated below, where the transition t_3 fires

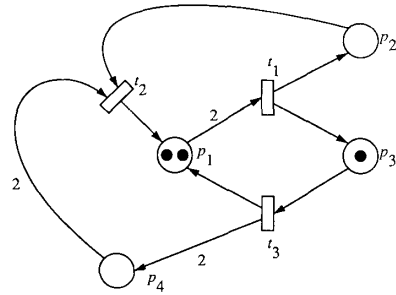


Fig. 21. Example 5: A Petri net.

to result in the marking $M_1 = (3 \ 0 \ 0 \ 2)^T$ from $M_0 = (2 \ 0 \ 1 \ 0)^T$:

$$\begin{bmatrix} 3 \\ 0 \\ 0 \\ 2 \end{bmatrix} = \begin{bmatrix} 2 \\ 0 \\ 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -2 & 1 & 1 \\ 1 & -1 & 0 \\ 1 & 0 & -1 \\ 0 & -2 & 2 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}.$$

The incidence matrix A is of rank 2 and can be partitioned in the form of (8), where

$$A_{11} = \begin{bmatrix} -2 & 1 \\ 1 & -1 \end{bmatrix} \text{ and } A_{12} = \begin{bmatrix} 1 & 0 \\ 0 & -2 \end{bmatrix}.$$

Thus, the matrix B_i can be found by (9):

$$B_i = \begin{bmatrix} 1 & 0 & 2 & 1/2 \\ 0 & 1 & -1 & -1/2 \end{bmatrix}.$$

It is easy to verify that $B_i \Delta M = 0$ holds for $\Delta M = M_1 - M_0 = (1 \ 0 \ -1 \ 2)^T$. \square

An integer solution x of the homogeneous equation ($\Delta M = 0$ in (6))

$$A^T x = 0 \quad (12)$$

is called a *T-invariant*, and an integer solution y of the transposed homogeneous equation $Ay = 0$ is called an *S-invariant*.

variant. These invariants which we will discuss in Section VIII provide powerful tools for studying structural properties of Petri nets.

C. Simple Reduction Rules for Analysis

To facilitate the analysis of a large system, we often reduce the system model to a simpler one, while preserving the system properties to be analyzed. Conversely, techniques to transform an abstracted model into a more refined model in a hierarchical manner can be used for synthesis. There exist many transformation techniques for Petri nets. In this section, we present only the simplest transformations, which can be used for analyzing liveness, safeness, and boundedness. Several transformation rules for marked graphs will be discussed in Section VII-B2.

It is not difficult to see that the following six operations [179], [203] preserve the properties of liveness, safeness, and boundedness. That is, let (N, M_0) and (N', M'_0) be the Petri nets before and after one of the following transformations. Then (N', M'_0) is live, safe, or bounded *iff* (N, M_0) is live, safe, or bounded, respectively.

- 1) *Fusion of Series Places* (FSP) as depicted in Fig. 22(a).
- 2) *Fusion of Series Transitions* (FST) as depicted in Fig. 22(b).
- 3) *Fusion of Parallel Places* (FPP) as depicted in Fig. 22(c).
- 4) *Fusion of Parallel Transitions* (FPT) as depicted in Fig. 22(d).
- 5) *Elimination of Self-loop Places* (ESP) as depicted in Fig. 22(e).
- 6) *Elimination of Self-loop Transitions* (EST) as depicted in Fig. 22(f).

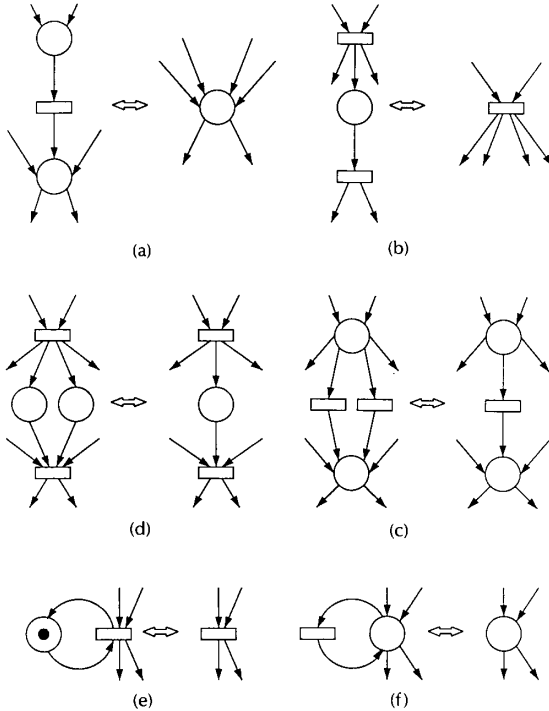


Fig. 22. Six transformations preserving liveness, safeness, and boundedness.

Example 6: The net shown in Fig. 17(d) can be reduced to the one shown in Fig. 23(a) after firing t_2 to remove the token in p_1 and then fusing t_1 and t_2 into t_{12} , and t_3 and t_4 into t_{34} . The net in Fig. 23(a) can then be reduced to the one shown in Fig. 23(b) after eliminating self-loop transition t_{12} and place p_3 . It is easy to see that both nets shown in Fig. 17(d) and Fig. 23(b) are bounded and non-live (and nonreversible). \square

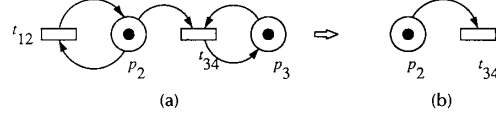


Fig. 23. Example 6: Illustration of reduction rules. The net shown in Fig. 17(d) is reduced to the two nets shown, where all the three nets are bounded, nonlive, and nonreversible.

As pointed out in the introduction, a major weakness of Petri nets is the complexity problem. Thus, it is very important to develop methods of transformations which allow hierarchical or stepwise reductions and preserve the system properties to be analyzed. Such an approach is discussed in [204], where subnets are reduced to single transitions or places while keeping liveness and/or boundedness properties. However, much work remains to be done in this area of research. For example, given a property or a set of properties, it is desired to develop a complete set of transformations which allows transformation between any two nets having the given properties. For further information on this subject, the reader is referred to [200]–[205], [245], [246], and [256].

VI. CHARACTERIZATIONS OF LIVENESS, SAFENESS, AND REACHABILITY

In this section, we first discuss some subclasses and then liveness, safeness, and reachability criteria within each subclass of Petri nets.

A. Subclass of Petri Nets

Recall that a Petri net is called *ordinary* when all of its arc weights are 1's. All Petri nets considered in this section are ordinary. Note that both ordinary and nonordinary Petri nets have the same modeling power. The only difference is modeling efficiency or convenience.

We use the following symbols for a pre-set and a post-set (where F is the set of all arcs defined in Table 2):

- $\bullet t = \{p | (p, t) \in F\}$ = the set of input places of t
- $t \bullet = \{p | (t, p) \in F\}$ = the set of output places of t
- $\bullet p = \{t | (t, p) \in F\}$ = the set of input transitions of p
- $p \bullet = \{t | (p, t) \in F\}$ = the set of output transitions of p .

The above symbols are illustrated in Fig. 24. This notation can be extended to a subset. For example, let $S_1 \subseteq P$, then $\bullet S_1$ is the union of all $\bullet p$ such that $p \in S_1$. With the above notation, we can now define subclasses of Petri nets by imposing some restrictions on their underlying structures [8], [206], [207]. Unless otherwise stated, it is assumed throughout this paper that a net N has no isolated places and transitions, i.e., no p or t such that $\bullet p = p \bullet = \emptyset$ or $\bullet t = t \bullet = \emptyset$.

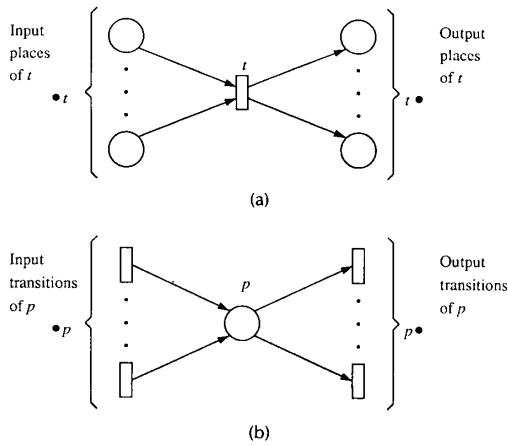


Fig. 24. The symbols for (a) the sets of input and output places of t , and (b) the sets of input and output transitions of p .

1) A *state machine* (SM) is an ordinary Petri net such that each transition t has exactly one input place and exactly one output place, i.e.,

$$|\bullet t| = |t\bullet| = 1 \quad \text{for all } t \in T.$$

2) A *marked graph* (MG) is an ordinary Petri net such that each place p has exactly one input transition and exactly one output transition, i.e.,

$$|\bullet p| = |p\bullet| = 1 \quad \text{for all } p \in P.$$

3) A *free-choice net* (FC) is an ordinary Petri net such that every arc from a place is either a unique outgoing arc or a unique incoming arc to a transition, i.e.,

$$\begin{aligned} &\text{for all } p \in P, |\bullet p| \leq 1 \text{ or } p\bullet = \{p\}; \text{ equivalently,} \\ &\text{for all } p_1, p_2 \in P, p_1\bullet \cap p_2\bullet \neq \emptyset \Rightarrow |p_1\bullet| = |p_2\bullet| \\ &= 1. \end{aligned}$$

4) An *extended free-choice net* (EFC) is an ordinary Petri net such that

$$p_1\bullet \cap p_2\bullet \neq \emptyset \Rightarrow p_1\bullet = p_2\bullet \text{ for all } p_1, p_2 \in P.$$

5) An *asymmetric choice net* (AC) (also known as a *simple net*) is an ordinary Petri net such that

$$\begin{aligned} &p_1\bullet \cap p_2\bullet \neq \emptyset \Rightarrow p_1\bullet \subseteq p_2\bullet \text{ or } p_1\bullet \supseteq p_2\bullet \\ &\text{for all } p_1, p_2 \in P. \end{aligned}$$

The Petri net structures shown in Fig. 25 are the key structures that characterize these subclasses. It is easy to recognize the key structures of SMs and MGs shown in Fig. 25(a) and (b), respectively. FCs are a generalization of the structures common to both SMs and MGs. They allow the conflict structure of SM shown in Fig. 25(a) and the synchronization structure of MG shown in Fig. 25(b), but exclude the structure shown in Fig. 25(c), where $p_1\bullet = p_2\bullet = \{t_1, t_2\}$. Extended free choice nets (EFC) allow the structure shown in Fig. 25(c) but not the one shown in Fig. 25(d), where $p_1\bullet = \{t_1\} \subseteq p_2\bullet = \{t_1, t_2\}$. Both FCs and EFCs have the behavioral property that if t_1 and t_2 share a common input place, then there are no markings for which one is

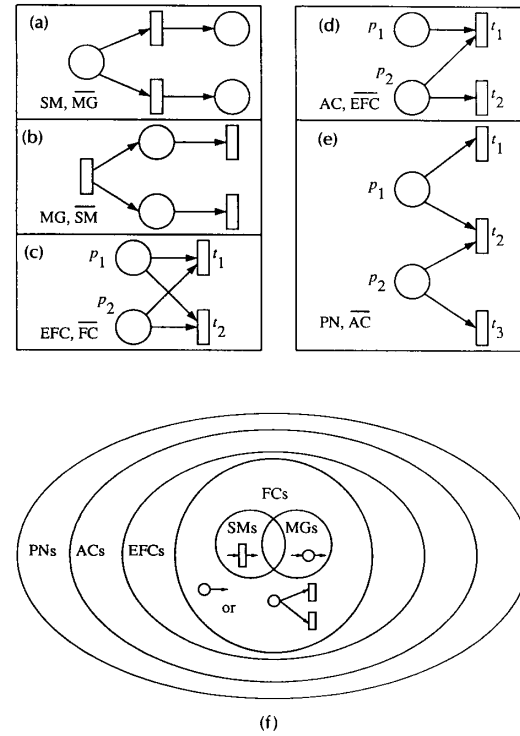


Fig. 25. Key structures characterizing subclasses of Petri nets and their Venn diagram, where \overline{MG} , \overline{SM} , etc., denote nonMG, nonSM, etc.

enabled and the other is disabled. Thus, we have “free-choice” about which transition to fire. In this sense, the EFC structure shown in Fig. 25(c) can be transformed to its equivalent FC structure as is illustrated in Fig. 26 [207]. Asymmetric choice nets (AC) allow the structure shown in Fig. 25(d) but not the structure of a confusion shown in Fig. 25(e), where $p_1\bullet = \{t_1, t_2\}$ and $p_2\bullet = \{t_2, t_3\}$. Unlike the behavioral property of FCs and EFCs, ACs can have a marking at which t_1 is enabled but t_2 is disabled.

In summary, SMs admit no synchronization, MGs admit no conflicts, FCs admit no confusion, and ACs allow asymmetric confusion (Fig. 7(b) but disallow symmetric confusion (Fig. 7(a)). Their Venn diagram relation is shown in Fig. 25(f).

Example 7: We apply the above classification of subclasses to classify the nets shown in Fig. 17. The net shown

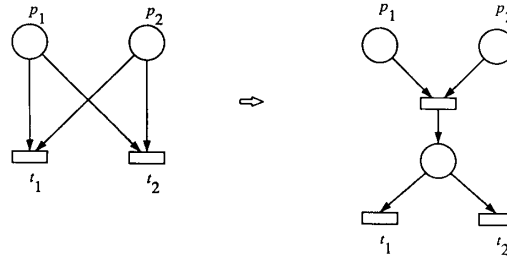


Fig. 26. Transformation of EFC structure to FC structure.

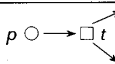
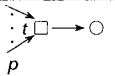
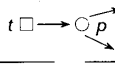
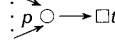
in Fig. 17(a) is not an AC because $p_3 \bullet = \{t_1, t_2\}$, $p_4 \bullet = \{t_1, t_4\}$, $p_3 \bullet \cap p_4 \bullet \neq \emptyset$, but one is not a subset of the other. The net in Fig. 17(c) is FC since each place has a unique outgoing arc. The nets in Fig. 17(d) and (g) are ACs since $p_3 \bullet = \{t_3\} \subset p_2 \bullet = \{t_1, t_3\}$ in Fig. 17(d) $p_3 \bullet = \{t_2\} \subset p_2 \bullet = \{t_2, t_4\}$ and $p_5 \bullet = \{t_4\} \subset p_2 \bullet = \{t_2, t_4\}$ in Fig. 17(g). The net in Fig. 17(f) is not an AC because $p_2 \bullet = \{t_1, t_4\}$ and $p_3 \bullet = \{t_1, t_5\}$. The net in Fig. 17(h) is both an MG and an SM. The net in Fig. 17(e) is an MG.

B. Liveness and Safeness Criteria

1) *Existence of Live-Safe Markings:* Live and safe Petri nets (LSPNs) are fundamental to both the applications and theoretical developments of Petri nets. In this section, we present liveness and safeness conditions for subclasses of Petri nets.

First, we discuss necessary conditions for existence of an LS marking M_0 for a Petri net structure PN . A place p (transition t) is said to be a *source place* (*source transition*) if $\bullet p = \emptyset$ ($\bullet t = \emptyset$). A place p (transition t) is said to be a *sink place* (*sink transition*) if $p \bullet = \emptyset$ ($t \bullet = \emptyset$). It is not difficult to see the following theorem [208] from Table 3.

Table 3 Explanation as to Why a Live and Safe Petri Net Cannot Have Source or Sink Places and Transitions

Case	If	such as	then
1	$\bullet p = \emptyset$ (source place)		t is not live.
2	$p \bullet = \emptyset$ (sink place)		p is not safe for live t .
3	$\bullet t = \emptyset$ (source transition)		p is not safe.
4	$t \bullet = \emptyset$ (sink transition)		t is not live for safe p .

Theorem 3: If a Petri net (N, M_0) is live and safe, then there are no source or sink places and source or sink transitions, i.e., for all $x \in P \cup T$, $\bullet x \neq \emptyset \neq x \bullet$ \square

This theorem can be generalized and we can state that if a connected Petri net (N, M_0) is live and safe, then N is strongly connected, i.e., there exists a directed path from every node to every other node in $P \cup T$. However, not all strongly connected nets have a live and safe marking. For example, the nets shown in Fig. 27(a) and (b) are strongly connected, but the net in Fig. 27(a) has no live markings and the net in Fig. 27(b) has no nonzero safe markings [208]. In the case of marked graphs and state machines, strongly-connectedness becomes a necessary and sufficient condition for existence of a live and safe marking (see Theorem 10).

We now provide conditions for liveness and/or safeness for subclasses of Petri nets. Since a dead net (a net in which every transition is dead) is trivially safe, we are normally interested in safeness for live nets.

2) *Liveness and Safeness in SM and MG:* Since a transition firing in a state machine moves only one token from a place to another place, it is easy to verify the following theorem.

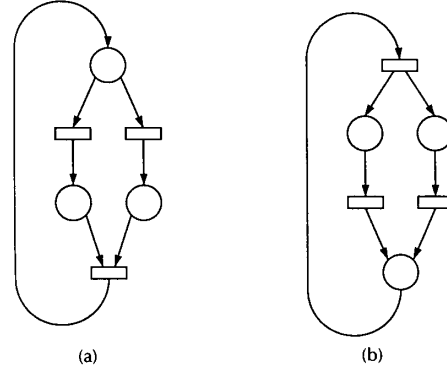


Fig. 27. (a) Strongly connected net that has no live markings. (b) Strongly connected net that has no nonzero safe markings.

Theorem 4: A state machine (N, M_0) is live iff N is strongly connected and M_0 has at least one token. \square

Theorem 5: A state machine (N, M_0) is safe iff M_0 has at most one token. A live state machine (N, M_0) is safe iff M_0 has exactly one token. \square

For marked graphs, each place has exactly one incoming arc and exactly one outgoing arc with unit weight. Thus, a marked graph (N, M_0) can be drawn as a marked directed graph (G, M_0) , where arcs correspond to places, nodes to transitions, and tokens are placed on arcs. For example, the Petri net of a communication protocol shown in Fig. 9 can be redrawn as the marked directed graph shown in Fig. 28.

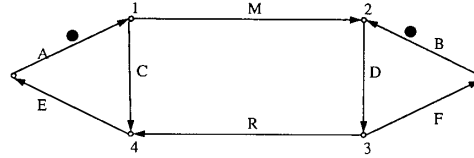


Fig. 28. The marked graph representation of a communication protocol shown in Fig. 9 and used for Example 14.

The firing of a node (transition) in a marked graph consists of removing one token from each incoming arc (input place) and adding one token to each outgoing arc (output place). If a node is on a directed circuit (or loop), then exactly one of its incoming arcs and one of its outgoing arcs belong to the directed circuit. If a node does not lie on the directed circuit in question, none of the arcs incident to that node will belong to the directed circuit. Thus, we have the following token invariance property [7].

Theorem 6: For a marked graph, the token count in a directed circuit is invariant under any firing, i.e., $M(C) = M_0(C)$ for each directed circuit C and for any M in $R(M_0)$, where $M(C)$ denotes the total number of tokens on C . \square

By Theorem 6, if there are no tokens on a directed circuit at the initial marking, then this directed circuit remains token-free. Thus, the nodes on this directed circuit will never be enabled. On the other hand, if a node is never enabled by any firing sequence, then by back-tracking token-free arcs, one can find a token-free directed circuit. Therefore, we have the following theorem.

Theorem 7: A marked graph (G, M_0) is live iff M_0 places at least one token on each directed circuit in G . \square

The following theorem is a special case of more general theorems which will be proved in Section VII (weighted sum of tokens) and Section VIII (S-invariants).

Theorem 8: The maximum number of tokens that an arc can have in a marked graph (G, M_0) is equal to the minimum number of tokens placed by M_0 on a directed circuit containing this arc. \square

The following consideration is helpful in understanding the above mini-max theorem. Consider all directed circuits C_1, C_2, \dots, C_m passing through the arc e . Bring as many tokens as possible on the incoming arcs of the initial node x of $e = (x, y)$, and fire the node x as many times as possible without firing the node y . It can be seen that $\text{Min} \{M_0(C_1), M_0(C_2), \dots, M_0(C_m)\}$ is the maximum possible tokens that can be brought on the arc e . In particular, if $\text{Min} \{M_0(C_1), M_0(C_2), \dots, M_0(C_m)\} = 1$, then $M(e) \leq 1$ for all M in $R(M_0)$. Thus, we have the following theorem.

Theorem 9: A live marked graph (G, M_0) is safe iff every arc (place) belongs to a directed circuit C with $M_0(C) = 1$.

Theorem 10: There exists a live and safe marking in a directed graph G iff G is strongly connected.

Proof: The necessity is due to Theorem 3. The sufficiency can be proved as follows. Suppose G is strongly connected. Choose a marking M_0 which places at least one token in each directed circuit in G . Then this marked directed graph, (G, M_0) is live. If (G, M_0) is not safe, then there is an arc e and a marking M in $R(M_0)$ such that $M(e) \geq 2$. Reduce the number of tokens on e to one by removing tokens from e ; call the new marking M' , i.e., $M'(e) = 1$. Repeat the above token removal which will not destroy the liveness property, until (G, M'_0) is safe for a new marking M'_0 . \square

A subset of arcs E' in a directed graph $G = (V, E)$ is said to be a *feedback arc set* (FAS) if $G' = (V, E - E')$ is acyclic, i.e., has no directed circuits. A FAS is said to be *minimal* if no proper subset of the FAS is a FAS, and *minimum* if no other FAS contains a smaller number of arcs. It is easy to see that the subset of marked arcs in a live marked graph is a FAS. Conversely, if each arc in a FAS of a directed graph is marked, we have a live marked graph. Furthermore, the following theorem [209] holds.

Theorem 11: A strongly-connected live marked graph G is safe iff for every marking M in $R(M_0)$, the set of marked arcs is a minimal FAS.

A minimum FAS is more important than a minimal FAS in applications. It is obvious that a subset E' in a directed graph G is a minimum FAS iff the marking M such that $M(e) = 1$ for all e in E' is a live marking for G with the minimum number of tokens. However, a minimum FAS does not necessarily yield a safe marking. For example, the marking M_0 shown in Fig. 29 is a live marking with the minimum number of tokens and corresponds to a minimum FAS (G becomes acyclic if the two marked arcs a and b are removed). However, this marking is not safe since arcs d and f do not belong

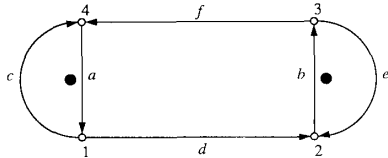


Fig. 29. The set of marked arcs a and b is a minimum feedback arc set, and this marking is minimally live but not safe.

to a directed circuit with token count one. In fact, the firing sequence $\sigma = (1 \ 3 \ 4 \ 1)$ brings two tokens on arc d .

3) *Liveness and Safeness in FC and AC Nets:*

Siphon and trap: A nonempty subset of places S in an ordinary net N is called a *siphon* (also known as a *deadlock*) if $\bullet S \subseteq S^\bullet$, i.e., every transition having an output place in S has an input place in S . (We use a siphon instead of a deadlock since the latter is used for a circular waiting condition or behavior in computer science). A nonempty subset of places Q in an ordinary net N is called a *trap* if $Q^\bullet \subseteq \bullet Q$, i.e., every transition having an input place in Q has an output place in Q . A siphon is illustrated in Fig. 30(a), where

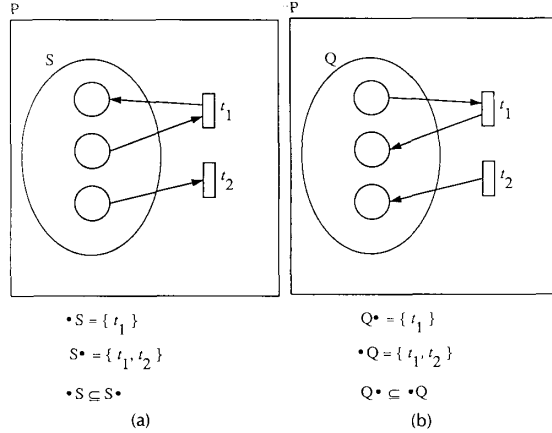


Fig. 30. Illustration of (a) a siphon and (b) a trap.

the token count in the siphon remains the same by firing t_1 but decreases by firing t_2 . Thus, a siphon has a behavioral property that if it is token-free under some marking, then it remains token-free under each successor marking. A trap is illustrated in Fig. 30(b), where the token count in the trap remains the same by firing t_1 but increases by firing t_2 . Thus, a trap has a behavioral property that if it is marked (i.e., it has at least one token) under some marking, then it remains marked under each successor marking. It is easy to verify that the union of two siphons (traps) is again a siphon (trap). A siphon (trap) is called a *basis siphon* (*basis trap*) if it cannot be represented as a union of other siphons (traps). All siphons (traps) in a Petri net can be generated by the union of some basis siphons (traps) [210]. A siphon (trap) is said to be minimal if it does not contain any other siphon (trap). A minimal siphons (traps) are basis siphons (traps), but not all basis siphons (traps) are minimal.

Example 8: In the Petri net shown in Fig. 31, let $S_1 = \{p_1, p_2, p_3\}$, $S_2 = \{p_1, p_2, p_4\}$, $S_3 = \{p_1, p_2, p_3, p_4\}$, $S_4 = \{p_2, p_3\}$ and $S_5 = \{p_2, p_3, p_4\}$. Then, we have $\bullet S_1 = \{t_1, t_2, t_4\} \subseteq S_1^\bullet = \{t_1, t_2, t_3, t_4\}$. Thus, S_1 is a siphon. Since $S_4^\bullet = \{t_1, t_4\} \subseteq \bullet S_4 = \{t_1, t_2, t_3\}$, S_4 is a trap. Similarly, it is easy to verify that S_2 is a siphon, S_3 is both a siphon and a trap, and S_5 is a trap. In fact, both S_1 and S_2 are minimal and basis siphons. S_3, S_4 , and S_5 are basis traps, S_3 and S_5 are not minimal traps.

Siphons and traps can be found from a set of logic equations or linear inequalities describing their behavioral properties [179]. For example, in the Petri net shown in Fig.

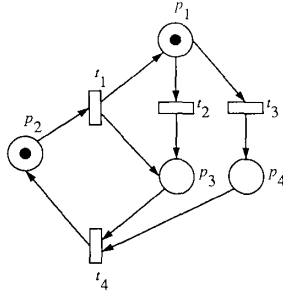


Fig. 31. The net used in Examples 8 and 9.

31, we see that

$$p_1 \Rightarrow p_2 \quad (\text{if } p_1 \text{ is in a siphon } S, \text{ then } p_2 \text{ is in } S)$$

$$p_2 \Rightarrow p_3 \vee p_4 \quad (\text{if } p_2 \text{ is in } S, \text{ then } p_3 \text{ or } p_4 \text{ is in } S)$$

$$p_3 \Rightarrow p_1 \wedge p_2 \quad (\text{if } p_3 \text{ is in } S, \text{ then } p_1 \text{ and } p_2 \text{ are in } S)$$

$$p_4 \Rightarrow p_1 \quad (\text{if } p_4 \text{ is in } S, \text{ then } p_1 \text{ is in } S).$$

The above set of “if-then” rules is equivalent to the following set of clauses:

$$\begin{aligned} &\{\neg p_1 \vee p_2, \neg p_2 \vee p_3 \vee p_4, \neg p_3 \vee p_1, \\ &\quad \neg p_3 \vee p_2, \neg p_4 \vee p_1\}. \end{aligned}$$

Thus, siphons can be found as (0,1)-solutions of the following set of inequalities, where $p_i = 1$ if $p_i \in S$, and $p_i = 0$ if $p_i \notin S$:

$$\begin{cases} -p_1 + p_2 \geq 0 \\ -p_2 + p_3 + p_4 \geq 0 \\ -p_3 + p_1 \geq 0 \\ -p_3 + p_2 \geq 0 \\ -p_4 + p_1 \geq 0 \end{cases}$$

For example, $p_1 = p_2 = p_3 = 1, p_4 = 0$ satisfy the above inequalities. Therefore, $\{p_1, p_2, p_3\}$ is a siphon. \square

The following theorems are well known in the literature [207], [211]. However, since their proofs are beyond the scope of this paper, they are omitted. Examples are given to illustrate the theorems.

Theorem 12: A free-choice net (N, M_0) is live iff every siphon in N contains a marked trap. \square

Theorem 13: A live free-choice net (N, M_0) is safe iff N is covered by strongly-connected SM components each of which has exactly one token at M_0 . \square

Theorem 14: Let (N, M_0) be a live and safe free-choice net. Then, N is covered by strongly-connected MG components. Moreover, there is a marking $M \in R(M_0)$ such that each component (N_1, M_1) is a live and safe MG, where M_1 is M restricted to N_1 . \square

Theorem 15: An asymmetric choice net (N, M_0) is live if (but not only if) every siphon in N contains a marked trap. \square

In Theorem 13 (Theorem 14), an *SM-component* (MG-component) N_1 of a net N is defined as a subnet generated by places (transitions) in N_1 having the following two properties: i) each transition (place) in N_1 has at most one incom-

ing arc and at most one outgoing arc; and ii) a subnet generated by places (transitions) is the net consisting of these places (transitions), all of their input and output transitions (places), and their connecting arcs. Theorem 13 (Theorem 14) leads to the observation [211] that a live and safe free-choice net can be viewed as an interconnection of live and safe state machines (marked graphs). This observation is useful for many applications including decompositions and abstraction of Petri nets [205].

Example 9: The FC net shown in Fig. 31 is not live since the siphon $\{p_1, p_2, p_4\}$ contains no traps (thus no marked traps). The AC net shown in Fig. 32 is live since the minimal

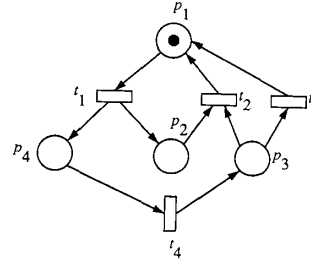


Fig. 32. A live AC net.

siphon $\{p_1, p_3, p_4\}$ contains a marked trap $\{p_1, p_3, p_4\}$ and the siphon $\{p_1, p_2, p_3, p_4\}$ contains marked traps $\{p_1, p_2\}$ and $\{p_1, p_3, p_4\}$. The live FC net shown in Fig. 33(a) is not safe and the safe FC net shown in Fig. 33(b) is not live since

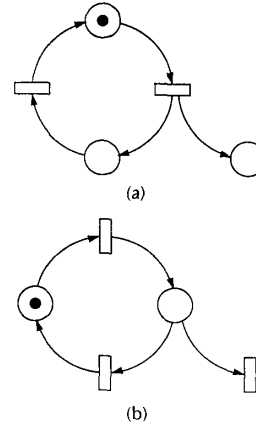


Fig. 33. (a) A live, nonsafe FC net. (b) A safe, nonlive FC net.

these nets are not covered by strongly connected MG components (nor by strongly connected SM components). The net shown in Fig. 34 is live and safe but is not covered by strongly-connected MG components since it is not FC. The AC net shown in Fig. 34 is live since every siphon contains a marked trap. However, the AC net shown in Fig. 35 is live even though the siphon $\{p_1, p_2, p_3, p_4\}$ contains no marked traps (see Theorem 15). The live and safe FC net shown in Fig. 36(a) is covered by the two strongly-connected MG components shown in Fig. 36(b). It is also covered by the two strongly-connected SM components shown in Fig. 36(c). \square

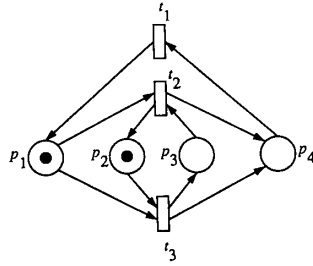


Fig. 34. A live and safe AC net.

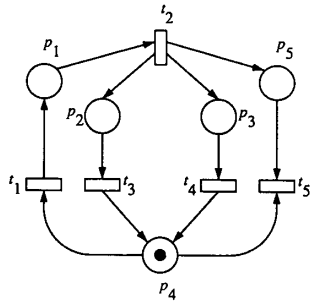


Fig. 35. A live AC net.

It is known that an asymmetric choice net (N, M_0) is live iff it is *place-live*, i.e., for each M_1 in $R(M_0)$ and for each place p in N , there exists a marking M in $R(M_1)$ such that $M(p) > 0$. The Petri net shown in Fig. 17(a) is place-live, but it is not live since t_1 is dead. Another useful property of asymmetric choice nets is that the conflict relation is transitive. For example, in the asymmetric choice net shown in Fig. 37(a), any pair of transitions among t_1 , t_2 , and t_3 are in a conflict relation. However, in the net shown in Fig. 37(b), which is not an asymmetric choice net, the pair (t_1, t_2) is in conflict and the pair (t_2, t_3) is in conflict but (t_1, t_3) is not in conflict.

References [11], [206]–[208], [211], and [214] are suggested for further reading on free-choice nets and other topics discussed in this section.

C. Reachability Criteria

In Section V-B, it has been shown that the existence of a nonnegative integer solution x satisfying (6) or

$$M_d = M_0 + A^T x \quad (13)$$

is a necessary condition for M_d to be reachable from M_0 . A Petri net having no directed circuits is called an *acyclic Petri net*. For this subclass, it can be shown [212] that this condition is necessary and sufficient. Given a nonnegative integer solution x satisfying (13), let N_x denote the (firing count) subnet of N consisting of transitions t such that $x(t) > 0$, together with their input and output places and their connecting arcs. M_{0x} denotes the subvector of M_0 for places in N_x .

Theorem 16: In an acyclic Petri net, M_d is reachable from M_0 iff there exists a nonnegative integer solution x satisfying (13).

Proof: Only sufficiency remains to be shown. Suppose there exists such a solution x . Consider the subnet (N_x, M_{0x}) ,

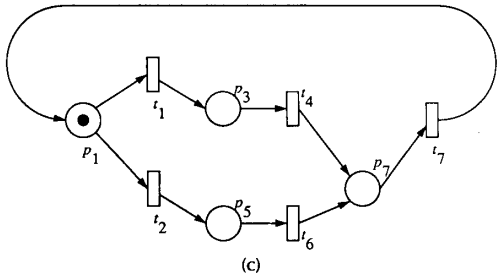
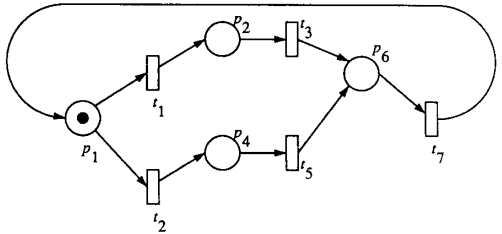
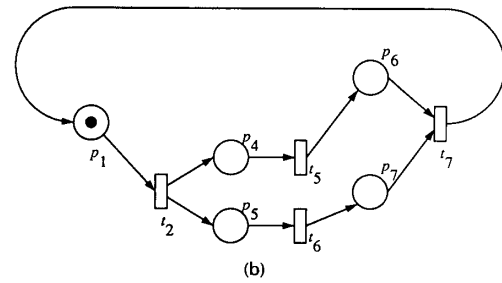
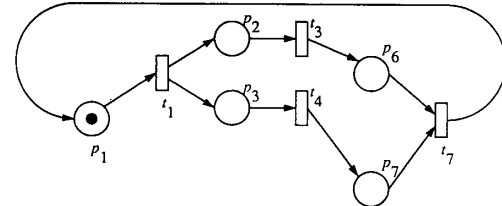
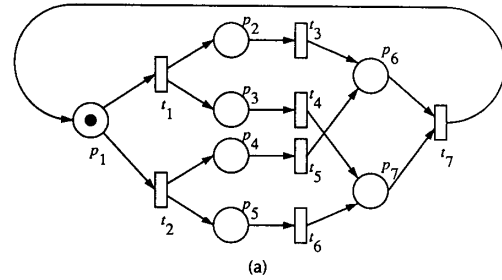


Fig. 36. (a) A live and safe FC net. (b) Its two strongly-connected MG-components generated by the two sets of transitions $\{t_1, t_3, t_4, t_7\}$ and $\{t_2, t_5, t_6, t_7\}$, respectively. (c) Its two strongly-connected SM-components generated by the two sets of places $\{p_1, p_2, p_4, p_6\}$ and $\{p_1, p_3, p_5, p_7\}$, respectively.

which is acyclic. There is at least one transition t that is firable at M_{0x} . (If not, back-tracing token-free input places of nonfirable transitions would end at a token-free source place p . This contradicts the fact that $M_d \geq 0$.) Now, fire t .

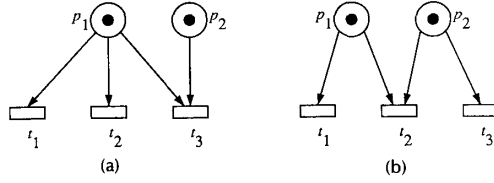


Fig. 37. The conflict relation is transitive in the AC net shown in (a), but not in the nonAC net shown in (b).

Let the resulting marking be $M' = M_0 + A^T u_t$, $x' = x - u_t$. Then, $M_d = M' + A^T x'$, $x' \geq 0$, and the subnet (N_x, M'_x) is acyclic. Repeat the above process until x' reduces to a zero vector. \square

A Petri net in which the set of places in every directed circuit is a trap (siphon) is called a *trap-circuit net* or *TC net* (a *siphon-circuit net* or *SC net*). Note that TC nets and SC nets are not necessarily free-choice or asymmetric-choice. The following theorem and corollary are recent results [101], [212] and generalize the reachability criteria for MGs (Theorem 20). The proof given below is based on [213]. (The casual readers may wish to skip the proof for the first reading since it is quite technical.)

Theorem 17: In a trap-circuit net, M_d is reachable from M_0 iff there exists nonnegative integer solution x such that i) e.g., (13) holds and ii) (N_x, M_{0x}) has no token-free siphons.

Proof: (\Rightarrow) i) is obvious. ii) If (N_x, M_{0x}) has a token-free siphon S , then it is not possible to fire any transition $t \in S^\bullet$. This contradicts the fact that every transition in N_x fires in a firing sequence transforming M_0 into M_d .

(\Leftarrow) Since there are no token-free siphons in (N_x, M_{0x}) , there is at least one transition t fireable at M_{0x} . Fire t and let $M' = M_0 + A^T u_t$, $x' = x - u_t$. Then $M_d = M' + A^T x'$, $x' \geq 0$. We claim that (N_x, M'_x) has no token-free siphons. First, we know that (N_x, M'_x) has no token-free source places (this would contradict $M_d \geq 0$). Next, consider an arbitrary siphon S in N_x . There are two cases to consider: 1) S was not a siphon in N_x ; 2) S was a siphon in N_x . In Case 1), S becomes a siphon in N_x after firing t . This is possible only if $t \in \bullet S$ in N_x , and t is removed in N_x . In this case, S is not token-free in (N_x, M'_x) since a firing of t brings some token(s) into S . Next, consider Case 2) when S was a siphon in N_x . Suppose S becomes token-free in (N_x, M'_x) . This means that a firing of t has removed all tokens from S and has brought no tokens into S . That is, $t \in S^\bullet$, $t \notin \bullet S$. Also, if $p \in S$ and p is an input place of t , then p cannot belong to any directed circuit consisting of places in S , since every directed circuit in N_x is a trap which will not become token-free, when it has a token. Now, S is token-free in (N_x, M'_x) and no transitions in S^\bullet are fireable. By back-tracing token-free input places of nonfireable transitions in S^\bullet , we can find a token-free siphon $S' \subset S$ such that for each $p \in \bullet t \cap S$, $p \notin S'$. This means that S' was a token-free siphon in (N_x, M_{0x}) as well. But this contradicts the condition ii). Thus, S cannot become token-free after firing t in Case 2). Therefore, (N_x, M'_x) has no token-free siphons. Furthermore, N_x is a TC net. Repeat the above process of firings until x' becomes a zero vector. \square

The *reversed net* N^{-1} of a Petri net N is the net obtained by reversing the direction of each arc in N . Note that a subset of places is a trap (siphon) in N^{-1} iff it is a siphon (trap) in N , and that the incidence matrix of N^{-1} is the negative

of the incidence matrix of N . Applying Theorem 17 to reversed nets yields the following corollary [101], [212].

Corollary 2: In a siphon-circuit net, M_d is reachable from M_0 iff there exists a nonnegative integer solution x such that i) eq. (13) holds and ii) (N_x, M_{dx}) has no token-free traps where M_{dx} is the subvector of M_d restricted to places in N_x . \square

Example 10: Consider the (non-asymmetric choice) net shown in Fig. 38(a). There are two directed circuits $p_1 t_2 p_2$

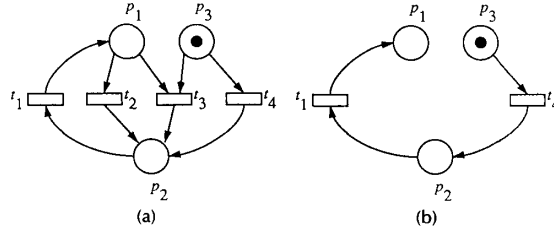


Fig. 38. Illustration of Theorem 17. (a) A given TC net (N, M_0) and (b) the subnet (N_x, M_{0x}) .

$t_1 p_1$ and $p_1 t_3 p_2 t_1 p_1$. The set of places in each of these directed circuits is a trap $\{p_1, p_2\}$. Thus, this is a TC net. Suppose $M_d = (1 \ 0 \ 0)^T$. Then it is easy to verify that (13) has a solution $x = (1 \ 0 \ 0 \ 1)^T$. Fig. 38(b) shows the subnet (N_x, M_{0x}) , where all three siphons $\{p_3\}$, $\{p_2, p_3\}$ and $\{p_1, p_2, p_3\}$ have a token. Thus, the condition of Theorem 17 holds. It is easy to see that M_d is reachable from M_0 by firing t_4 and then t_1 . \square

A Petri net in which the set of places in every directed circuit contains a trap (siphon) is called a *TCC net* (SCC net). For this generalized TC net (generalized SC net), a sufficient condition for reachability is known [101] and is stated as follows.

Theorem 18: In a TCC net, M_d is reachable from M_0 if there exists a nonnegative integer solution x such that i) eq. (13) holds and ii) every siphon in (N_x, M_{0x}) has a marked trap.

Corollary 3: In an SCC net, M_d is reachable from M_0 if there exists a nonnegative integer solution x such that i) eq. (13) holds and ii) (N_x, M_{dx}) has no token-free traps. \square

A *forward- (backward-) conflict-free net* [FCF (BCF) net] is a subclass of Petri nets such that each place has at most one outgoing (incoming) arc. A *nondecreasing- (non-increasing-) circuit net* [NDC (NIC) net] is a subclass of Petri nets such that the token content in any directed circuit is never decreased (increased) by any transition firing. It should be noted that TC nets (SC nets) contain these nets as their subclasses. Thus, Theorem 17 (Corollary 2) is applicable to these subclasses.

Furthermore, for these subclasses only a *minimal* non-negative integer solution x of (13) needs to be tested [212]. (A solution x is said to be *minimal* if $x \leq y$ for any other solution y .) This is not the case for TC (SC) nets because there may exist a firing sequence for a non-minimum solution even though there may not for a minimal solution. For example, consider the TC net shown in Fig. 39. Let $M_0 = (0 \ 0 \ 0)^T$ and $M_d = (0 \ 0 \ 1)^T$. Then, a non-minimal solution $y = (2 \ 1 \ 1 \ 1 \ 1)^T$ has the corresponding firing sequence $t_5 t_3 t_1 t_2 t_1 t_4$, but a minimal solution $x = (1 \ 1 \ 1 \ 0 \ 0)^T$ does not, since (N_x, M_{0x}) has a token-free siphon.

Fig. 40 depicts the relationship among the subclasses of Petri nets discussed in this subsection. Note that marked

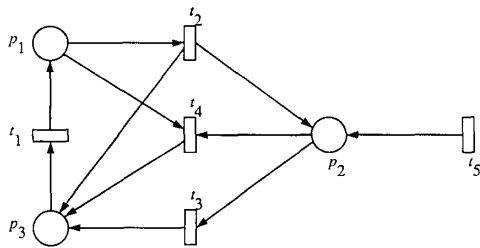


Fig. 39. A TC net which is not a FCF net.

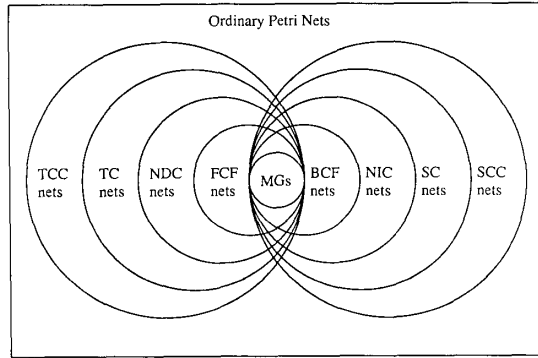


Fig. 40. Relationship among subclasses of Petri nets for which reachability criteria are known.

graphs are contained in the intersection of all these subclasses.

VII. ANALYSIS AND SYNTHESIS OF MARKED GRAPHS

Among models that can represent concurrent activities, marked graphs are the most amenable to analysis. Marked graphs basically model decision-free (or deterministic) concurrent systems. A marked graph representation of a system with decisions is possible only if each decision can be embedded in a single-entry-and-single-exit subsystem because this subsystem can then be represented by a place having exactly one incoming and one outgoing arc. This section presents detailed discussions on analysis and synthesis techniques for marked graphs.

A. Reachability in MGS

As stated in Section V-B, the incidence matrix A of a marked graph (G, M_0) corresponds to the node-to-arc incidence matrix of its underlying directed graph or digraph G . The matrix B_f of a marked graph (G, M_0) , which is defined by (9), corresponds to the fundamental circuit matrix B_f of G [13]. It is well known that the two matrices are orthogonal to each other, i.e.,

$$B_f A^T = 0. \quad (14)$$

If a marking M_d is reachable from M_0 through a firing sequence σ in a marked graph (G, M_0) , we can write (6) or

$$A^T x = \Delta M \quad (15)$$

where $\Delta M = M_d - M_0$ and $x = \bar{\sigma}$ = the firing count vector of the firing sequence σ . From (14) and (15), we have

$$B_f \Delta M = 0 \quad (16)$$

or

$$B_f M_0 = B_f M_d. \quad (17)$$

Equation (17) states that the algebraic sum of tokens placed by M_0 on a fundamental circuit is equal to that placed by M_d . Equation (17) is a generalization of the token invariance on a directed circuit (Theorem 6). Equation (16) has an interpretation like Kirchhoff's voltage law (KVL) in circuit theory. It has been shown [215] that this generalized token invariance expressed by (17) is not only necessary but also sufficient for a live marking M_0 to reach another marking M_d . In other words, we have the following theorem.

Theorem 19: In a live marked graph (G, M_0) , M_d is reachable from M_0 iff (17) holds. \square

Note that if G is strongly connected, then condition (17) is equivalent to saying that the token count on each directed circuit under M_0 is the same as that under M_d , i.e., $M_0(C) = M_d(C)$ for each directed circuit C in G .

The above theorem can be extended to nonlive marked graphs if one imposes the additional condition that the nodes (transitions) that are to fire should not lie on a token-free directed circuit. In other words, we have the following theorem [215].

Theorem 20: In a marked graph (G, M_0) , M_d is reachable from M_0 iff (17) holds and for the minimal nonnegative solution x for $A^T x = \Delta M$, no nodes t such that $x(t) > 0$ are on any token-free directed circuit in (G, M_0) . \square

The above theorem has been further generalized to submarking reachability [216], [217]. Since every marked graph is a TC net as well as an SC net, both Theorem 17 and Corollary 2 reduce to Theorem 20.

Example 11: Consider the marked graph (G, M_0) shown in Fig. 41, where G is not strongly connected and M_0 is not a

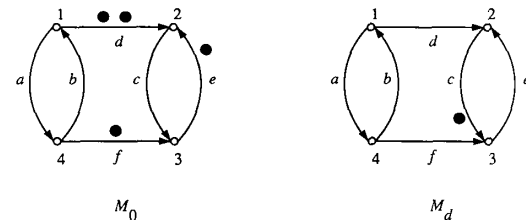


Fig. 41. The marked graph for Example 11 to illustrate reachability conditions.

safe marking. The fundamental circuit matrix B_f with respect to a spanning tree $\{d, e, f\}$ for G is given by

$$B_f = \begin{bmatrix} a & b & c & d & e & f \\ 1 & 0 & 0 & -1 & 1 & 1 \\ 0 & 1 & 0 & 1 & -1 & -1 \\ 0 & 0 & 1 & 0 & 1 & 0 \end{bmatrix}.$$

For the two markings $M_0 = (0 \ 0 \ 0 \ 2 \ 1 \ 1)^T$ and $M_d = (0 \ 0 \ 1 \ 0 \ 0 \ 0)^T$, it is easy to verify that (17) holds. Now, (15) for this marked graph can be written as

$$\begin{array}{c}
\begin{array}{cccc}
& 1 & 2 & 3 & 4 \\
\begin{array}{c} a \\ b \\ c \\ d \\ e \\ f \end{array} & \begin{bmatrix} 1 & 0 & 0 & -1 \\ -1 & 0 & 0 & 1 \\ 0 & 1 & -1 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & -1 & 1 & 0 \\ 0 & 0 & -1 & 1 \end{bmatrix} & \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} & = & \begin{bmatrix} 0 \\ 0 \\ 1 \\ -2 \\ -1 \\ -1 \end{bmatrix}
\end{array}
\end{array}$$

Since the rank of the above coefficient matrix is three (the number of nodes minus one), we only need to solve a set of three independent equations. Thus, by setting $x_4 = 0$ and solving the following set of three equations (corresponding to the spanning tree $\{d, e, f\}$):

$$\begin{array}{c}
d \\ e \\ f
\end{array}
\begin{bmatrix} 1 & -1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \end{bmatrix}
\begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} -2 \\ -1 \\ -1 \end{bmatrix}$$

we have $x_1 = 0$, $x_2 = 2$, and $x_3 = 1$. Therefore, $x = (0 \ 2 \ 1 \ 0)^T$ is the minimal nonnegative solution for $A^T x = \Delta M$. Note that $x' = (k \ k+2 \ k+1 \ k)^T$ for any positive integer k is also a nonnegative solution, but it is not minimal. Since nodes 2 and 3 corresponding to nonzero entries in x are not on the token-free directed circuit $\{a, b\}$, all the conditions of Theorem 20 are satisfied. Therefore, M_d is reachable from M_0 (by firing nodes 2, 3 and again 2). \square

Consider a firing sequence σ which starts and ends at M_0 . In this case, $\Delta M = 0$ and the firing count vector $\bar{\sigma}$ is a solution x of the homogeneous equation

$$A^T x = 0. \quad (18)$$

For a connected marked graph with n nodes, the rank of A is $n - 1$ and (18) has only one independent solution $x = (k \ k \ \dots \ k)^T$. This corresponds to a firing sequence which fires every node k times. The following theorem is easily shown [215].

Theorem 21: For a connected marked graph (G, M_0) , a firing sequence leads back to the initial marking M_0 iff it fires every node an equal number of times. \square

Now, suppose the underlying graph G_T of a marked graph is a tree. Any marking on G_T is live since a tree has no directed circuits. For any two markings M_0 and M_d for G_T , (17) holds since G_T has no circuits. Therefore, we have the following theorem [215].

Theorem 22: Any two markings on a directed graph G are mutually reachable iff the underlying graph of G is a tree. \square

(The two vectors x and ΔM in (15) can be interpreted as the node voltage vector and branch voltage vector, respectively, in an electrical network. Then, Theorem 21 can be seen from the fact that all the branch voltages are zero iff all the node voltages are the same. Theorem 22 is equivalent to saying that the branch voltage vector ΔM can be chosen arbitrarily iff the network is a tree.)

From (14) and (16), it is easy to see that (16) holds if ΔM is a row of the incidence matrix A . Also, from the well-known relationship $B_i C_i^T = 0$, where C_i is the fundamental cutset matrix, it can be seen that (16) holds if ΔM is a row of the cutset matrix C_i . In fact, the following theorem can be shown [215].

Theorem 23: Two markings M_0 and M_d in a live marked graph (G, M_0) are mutually reachable iff their difference $\Delta M = M_d - M_0$ is a linear combination of a set of fundamental cutsets of G . \square

In system design, we are often given a set of states that are mutually reachable. If a state is coded with an m -tuple of 0's and 1's, then the given set of states can be regarded as a set of (possibly safe) markings in a marked graph with m arcs. A synthesis problem is to find a marked graph from given sets of mutually reachable markings. Theorem 23 has been used as a basis for converting this synthesis problem to that of realizing cutset matrices of directed graphs [215]. This synthesis method produces live (but not necessarily safe) marked graphs.

B. Synthesis of Live-Safe MG

1) Live-Safe Equivalence Classes: Define a relation \sim on the set of live markings of a digraph G to be $M_0 \sim M_d$ if M_d is reachable from M_0 . Then it is easy to see that \sim is reflexive, symmetric, transitive, and thus an equivalence relation. The relation \sim partitions the set of live markings into equivalence classes. Let $\rho(G)$ be the number of equivalence classes of live-safe (LS) markings for a strongly connected graph G . Finding $\rho(G)$ for a general case is a major unsolved problem on marked graphs. However, for some specific types of digraphs, simple formulas for $\rho(G)$ are available. For example, it is known [218] that

$$\rho(N_k) = k - 1 \quad (19)$$

$$\rho(K_n) = (n - 1)! \quad (20)$$

where N_k is the necklace of k nodes shown in Fig. 42, and K_n is the complete digraph of n nodes.

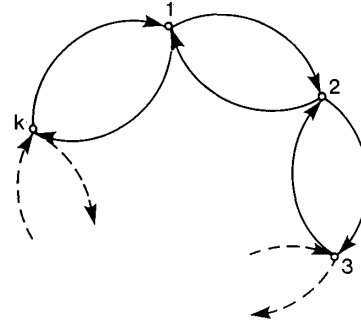


Fig. 42. N_k , the necklace of k nodes.

Example 12: N_2 , N_3 , and N_4 are shown in Fig. 43(a), (b), and (c), respectively, where a representative marking for each equivalence class in each of N_k , $k = 2, 3, 4$, are also shown. \square

The following two theorems [203], [218] provide necessary and sufficient conditions for $\rho(G) = 1$.

Theorem 24: For a strongly connected graph G , $\rho(G) = 1$ iff there do not exist three distinct nodes x, y, z which appear in two distinct directed circuits in the orders of (x, y, z) and (x, z, y) . \square

Theorem 25: For a strongly connected graph G , $\rho(G) = 1$ iff there is a marking of G which places exactly one token on every directed circuit in G . \square

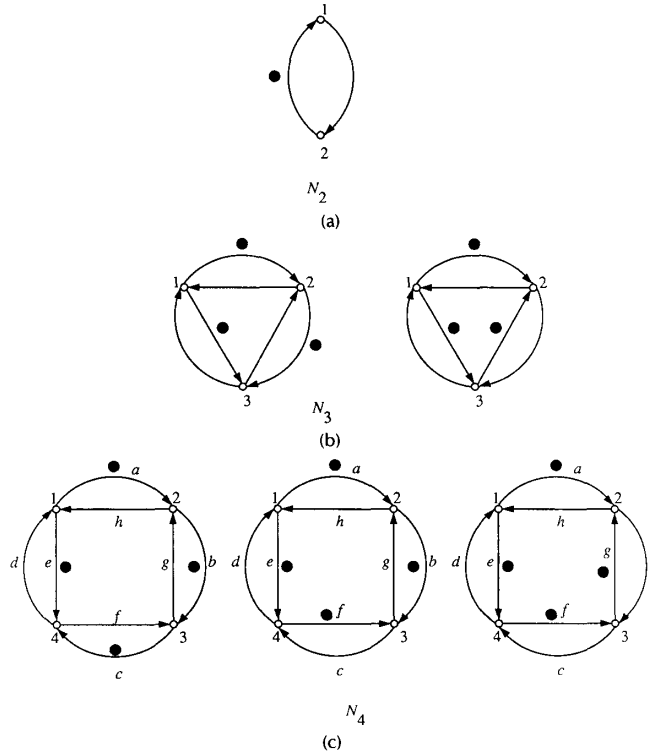


Fig. 43. (a) An example of $\rho(G) = 1$, $G = N_2$. (b) An example of $\rho(G) = 2$, $G = N_3 = K_3$. (c) An example of $\rho(G) = 3$, $G = N_4$.

Example 13: In the marked graphs shown in Fig. 43(c), $\rho(N_4) \neq 1$ since there are three distinct nodes 1, 2, 3 such that the sequence (1 2 3) is in the outer directed circuit $\{a, b, c, d\}$, and the sequence (1 3 2) is in the inner directed circuit $\{e, f, g, h\}$ (Theorem 24). Also, it can be seen that there is no marking M such that $M(C) = 1$ for every directed circuit in N_4 . In the marked graph (G, M_0) shown in Fig. 44, it can be verified that $\rho(G) = 1$ since M_0 places

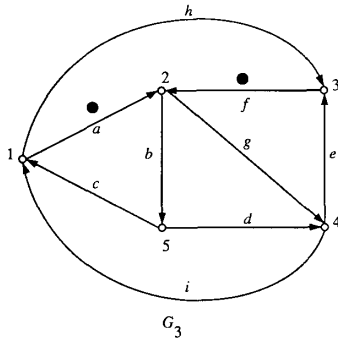


Fig. 44. A marked graph (G, M_0) with $\rho(G) = 1$.

exactly one token on each of the following possible directed circuits in G : $\{a, b, c\}$, $\{b, d, e, f\}$, $\{f, g, e\}$, $\{a, g, i\}$, $\{a, b, d, i\}$, $\{b, d, i, h, f\}$, $\{c, h, f, b\}$, and $\{h, f, g, i\}$ (Theorem 25). \square

2) Expansion Rules for LSMG Synthesis:

It has been shown [202], [203] that $\rho(G)$ is invariant under the following operations on a digraph G .

- Series Expansion (SE)**—addition of an arc e and node x in series with an existing arc e' (see Fig. 45(a)).
- Parallel Expansion (PE)**—addition of an arc e in parallel with an existing arc e' (see Fig. 45(b)).
- Unique-Circuit Expansion (UE)**—addition of an arc $e = (v_1, v_2)$ to a unique directed path P_{21} from v_2 to v_1 (see Fig. 45(c)).
- V-Y Expansion (VYE)**—addition of a node x and an arc $e = (x, y)$ to a pair of existing arcs $e_1 = (y, z)$ and $e_2 = (y, w)$ (also applicable if (α, β) is replaced by (β, α) everywhere) (see Fig. 45(d)).
- Separable Graph Expansion (SGE)**—joining two graphs G_1 and G_2 at exactly one node x to produce a separable graph G (see Fig. 45(e)). In this expansion, we have $\rho(G) = \rho(G_1) \rho(G_2)$. Thus, $\rho(G) = 1$ is invariant iff $\rho(G_1) = \rho(G_2) = 1$.

The above stepwise expansion operations can be used for synthesizing LSMGs (G, M_0) . In this synthesis, the following properties of decision-free concurrent systems can be prescribed: liveness (absence of deadlocks), safeness (absence of overflows), $\rho(G) = 1$ (all LS markings or states are mutually reachable), minimum cycle time, and resource requirements [219]. Note that, in order to maintain the liveness and safeness properties, the newly added arc e must be token-free in the series and V-Y expansions, $M(e) = M(e')$ in the

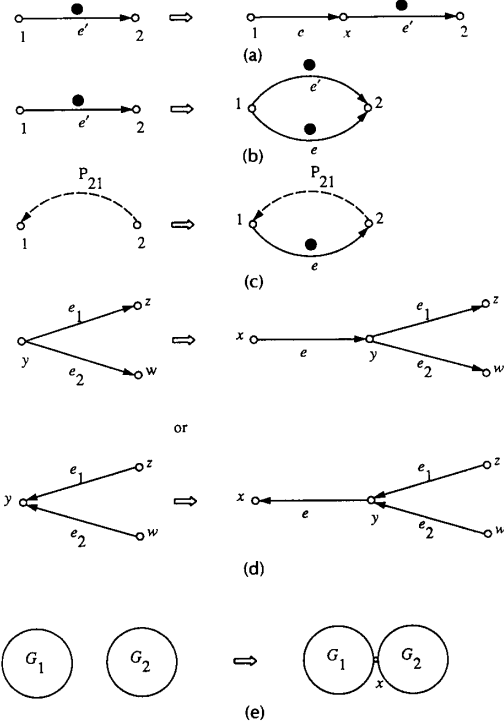


Fig. 45. Illustration of the five expansion rules (a) series expansion, (b) parallel expansion, (c) unique circuit expansion, (d) V-Y expansion, and (e) separate graph expansion

parallel expansion, and $M(e) = 1$ after making the unique path P_{21} token-free by appropriate firings.

Example 14: The LSMG model (G_1, M_0) of a communication protocol shown in Fig. 28 can be synthesized from N_2 by applying the above expansion rules six times. This is illustrated in Fig. 46, where $PE(\cdot)$, $SE(\cdot)$, and $UE(\cdot)$ denote

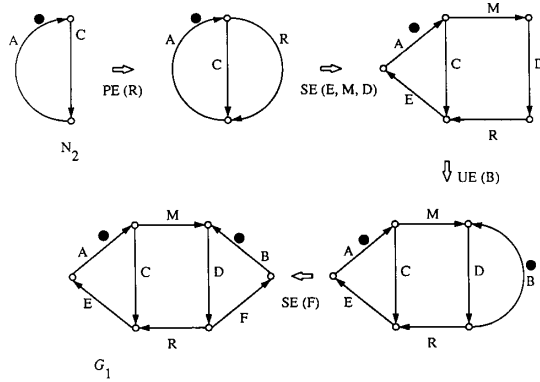


Fig. 46. Illustration of the use of expansion rules for LSMG synthesis.

parallel, series, and unique circuit expansions, respectively, and the labels inside the parentheses indicate the arcs added by the expansions. Thus, we have $\rho(G_1) = \rho(N_2) = 1$. By applying the SCE rule to n necklaces of 2 nodes (N_2),

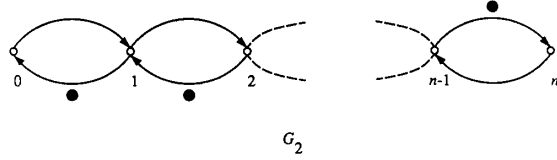


Fig. 47. A LS marked graph model of an n -stage pipeline operation.

we know that $\rho(G_2) = 1$ for the marked graph (G_2, M_0) of an n -stage pipeline operation shown in Fig. 47. The reverse operations of the above expansions are the reduction rules that can be used to find $\rho(G)$ for a given digraph G . For example, it is easy to verify that the marked graph (G_3, M_0) shown in Fig. 44 can be reduced to N_2 after applying a V-Y reduction first, and then series, parallel, and unique-circuit reductions. Thus, $\rho(G_3) = \rho(N_2) = 1$. \square

C. Weighted Sum of Tokens

Given a marked graph (G, M_0) , let I be an $m \times 1$ column vector (S-invariant) satisfying

$$AI = 0 \quad (21)$$

where A is the $n \times m$ incidence matrix of G . For any marking M reachable from M_0 , we have from (15) and (21)

$$(\Delta M)^T I = (x^T A) I = x^T (AI) = 0 \quad (22)$$

or

$$M^T I = M_0^T I. \quad (23)$$

Equation (23) states that the weighted sum of tokens $\sum_{i=1}^m M(e_i) I(e_i)$ is invariant for all markings M reachable from M_0 . In electrical network terminology, I corresponds to the branch current vector, (21) to Kirchhoff's current law, and (22) to Tellegen's theorem [92].

Let W be an $m \times 1$ column vector whose i th entry is $W(e_i)$, a nonnegative integer weight of arc e_i . $W(e_i)$ may represent the storage space or cost to accommodate a token on arc e_i . Then, $M^T W$ denotes the weighted sum of tokens for a marking M . Given a bounded live marked graph (G, M_0) , we are often interested in finding the maximum or minimum value of $M^T W$ for all markings reachable from M_0 . This value can be found from M_0 and a certain S-invariant I by the following theorem [92].

Theorem 26: For a strongly connected live marked graph (G, M_0) , we have

$$\max \{M^T W | M \in R(M_0)\} = \min \{M_0^T I | I \geq W, AI = 0\} \quad (24)$$

$$\min \{M^T W | M \in R(M_0)\} = \max \{M_0^T I | I \leq W, AI = 0\}. \quad (25)$$

Proof: Since M_0 is live, $M \in R(M_0)$ iff $M = M_0 + A^T x$, where x is an $n \times 1$ vector of nonnegative integers. Thus, the left-hand side of (24) can be written as the following linear programming problem:

$$\max U = C^T z \text{ subject to } Dz = M_0 \text{ and } z \geq 0 \quad (26)$$

where

$$C = \begin{bmatrix} W \\ 0 \end{bmatrix}, z = \begin{bmatrix} M \\ x \end{bmatrix}, D = [I_m] - A^T,$$

and I_m is the identity matrix of order m . The dual problem of (26) can be stated as

$$\min V = M_0^T y \text{ subject to } D^T y \geq C \quad (27)$$

where $y = I$ is unrestricted. $D^T y \geq C$ is equivalent to $I \geq W$ and $AI \leq 0$. However, $AI \leq 0$ is equivalent to $AI = 0$ since the sum of the n rows in A is always zero, i.e., $[1 \ 1 \ \dots \ 1]AI = 0I = 0$. Thus, (24) is equivalent to the problem on the right-hand side of (27). It is well known in linear programming that the optimal solution of (26) or (27) is an extreme point of the corresponding constraint set. Note that all the extreme points of the constraint set have only integral coordinates since D is totally unimodular, i.e., every square submatrix of D has determinant 0, 1, or -1 . Therefore, the optimal values of (26) and (27) are attained at integral values, and (24) follows from the theory of duality. Equation (25) can be proved similarly. \square

Example 15: Consider the marked graph shown in Fig. 48, where $M_0 = (1 \ 0 \ 1 \ 0 \ 0)^T$ and $W = (1 \ 2 \ 1 \ 2 \ 1)^T$.

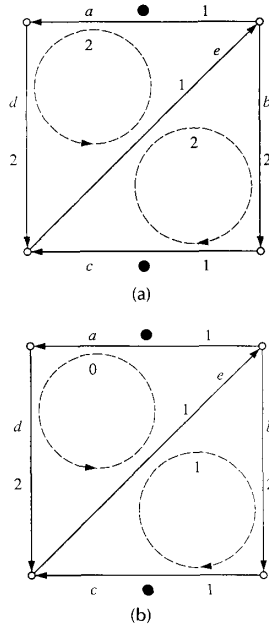


Fig. 48. Illustration of finding the maximum and minimum weighted sums of tokens in Example 15.

The "loop-current" distribution shown by the dotted lines in Fig. 48(a) yields an S-invariant $I_1 = [2 \ 2 \ 2 \ 2 \ 4]^T$ such that $I_1 \geq W$. Thus, by (24), $\max M^T W = M_0^T I_1 = 4$, which is attained at the marking $M = (0 \ 1 \ 0 \ 1 \ 0)^T$ reachable from M_0 . The "loop-current" distribution shown by the dotted lines in Fig. 48(b) gives an S-invariant $I_2 = (0 \ 1 \ 1 \ 0 \ 1)^T$ such that $I_2 \leq W$. By (25), we have $\min M^T W = M_0^T I_2 = 1$, which is attained at $M = (0 \ 0 \ 0 \ 0 \ 1)^T$ reachable from M_0 .

D. Token Distance and Maximum Concurrency

1) **Token Distance Matrix for MG:** The token distance t_{ij} between two nodes i and j in a marked graph (G, M_0) is defined as the minimum token content among all possible directed paths P_{ij} from node i to node j in G , i.e.,

$$t_{ij} = \begin{cases} \min M_0(P_{ij}), \\ \infty, \text{ if no directed path } P_{ij} \text{ exists.} \end{cases} \quad (28)$$

Given a marked graph with n nodes, the token distance matrix defined by $T = [t_{ij}]$ is an $n \times n$ matrix having the following properties:

- 1) $t_{ii} = 0$ for $i = 1, 2, \dots, n$.
- 2) $t_{ij} \leq t_{ik} + t_{kj}$ for all $1 \leq i, j, k \leq n$.
- 3) t_{ij} is a nonnegative integer or ∞ .

It has been shown [220], [221] that the token distance matrix $T = [t_{ij}]$ has the following useful applications.

Firability: A node j is firable (enabled) at a marking M iff all the off-diagonal entries of the j th column in T are positive.

Necessity of Firing: A node i must fire in order to enable another node j iff $t_{ij} = 0$, i.e., there exists a token-free directed path from i to j .

Synchronic Distance: The synchronic distance d_{ij} , defined by (1), between two nodes i and j in a marked graph is given by $d_{ij} = t_{ij} + t_{ji}$.

Maximum Firing Deviation: Let $t_{ij} + t_{ji} = k$ in a marked graph (G, M_0) . Then in any marking reachable from M_0 , k is the maximum number of times that one node i or j can fire without firing the other node.

Liveness: A marked graph is live iff $t_{ij} + t_{ji} = d_{ij} \neq 0$ for all $i \neq j$.

Shortest Firing Sequence: The following algorithm yields a shortest firing sequence to enable a node j in a live marked graph (G, M_0) . (The length of a firing sequence is defined as the number of transitions (nodes) fired in the sequence.)

- Step 1) If $t_{ij} > 0$ for $i = 1, 2, \dots, n$ ($i \neq j$), then node j is enabled. If not, go to Step 2.
- Step 2) Find a node i such that $t_{ij} = 0$ ($i \neq j$) and $t_{ki} > 0$ for $k = 1, 2, \dots, n$ ($k \neq i$), i.e., node i is enabled.
- Step 3) Fire the node i found in Step 2 and update the token distance matrix $T = [t_{ij}]$ (by subtracting 1 from each entry of the i th column, and adding 1 to each entry of the i th row in T). Go to Step 1.

The firing sequence obtained in the above algorithm is shortest since every node firing in Step 3 is necessary in order to enable node j .

Example 16: For the marked graph (G, M_0) shown in Fig. 49(a), the token distance matrix is given by

$$T = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

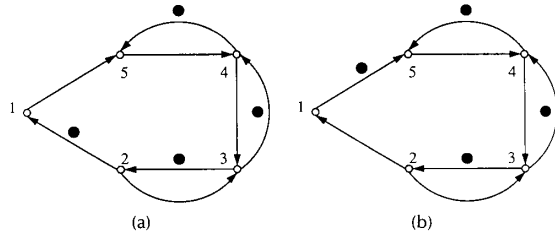


Fig. 49. MG in Example 16.

The nodes 1 and 2 are enabled since all the entries in the first and second columns in T are positive, except for the diagonal entries. There are token-free directed paths from node 1 to nodes 3, 4, and 5, since $t_{13} = t_{14} = t_{15} = 0$. The synchronic distance between nodes 1 and 2 is given by $d_{12} = t_{12} + t_{21} = 1 + 1 = 2$. Fire node 2 once. Then node 1 can be fired twice without firing node 2. The shortest firing sequence to enable node 5 is found from the zero entries in the fifth column in T . Since $t_{15} = 0$ is the only off-diagonal zero entry, it is necessary to fire only node 1 to enable node 5. Node 1 is firable and when it is fired, the updated token distance matrix T' is obtained from T by subtracting 1 from each entry of the first column and adding 1 to each entry of the first row. That is,

$$T' = \begin{bmatrix} 0 & 2 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 2 \\ 1 & 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

which is the token distance matrix of the MG shown in Fig. 49(b).

2) *Maximum Concurrency in MG*: A set of nodes which are enabled at the initial marking M_0 in a live marked graph (G, M_0) can be found from the token distance matrix $T = [t_{ij}]$. They are nodes corresponding to columns whose off-diagonal entries are all positive. For example, the first and second columns in T in Example 16 are such columns, and thus nodes 1 and 2 are concurrently enabled at M_0 . A more important problem is to find a maximum set of nodes that can be fired at a marking M reachable from M_0 . The following theorem proved in [221], [222] can be used to find such a set of concurrently firable nodes.

Theorem 27: A k -node set V_k in a live marked graph (G, M_0) is a k -node concurrent set *iff* i) every $(k - 1)$ -node subset of V_k is a $(k - 1)$ -node concurrent set, and ii) V_k is not contained in any directed circuit with token count less than k . \square

The necessity of conditions i) and ii) is obvious because: i) if all k nodes are concurrently enabled, then nodes in each proper subset are also concurrently enabled; and ii) at most k nodes in a directed circuit with k tokens can be concurrently enabled. Theorem 27 is used in [221] to state an algorithm for finding a maximum set of nodes which can be fired concurrently at some marking in $R(M_0)$. Alternatively, the problem of finding a maximum set of concurrently enabled nodes in a live marked graph (G, M_0) can be transformed

into the following $(0, 1)$ -integer programming problem:

$$\max \sum_{i=1}^n x(i)$$

subject to:

$$A^T x \leq M \quad (\text{Firability})$$

$$B_f M = B_f M_0 \quad (\text{Reachability})$$

where

$$x(i) = \begin{cases} 1 & \text{if node } i \text{ is enabled} \\ 0 & \text{otherwise.} \end{cases}$$

E. MG Synthesis of Synchronic Distance Matrix

The synchronic distance matrix of a marked graph is an $n \times n$ symmetric matrix $D = [d_{ij}]$, where d_{ij} is the synchronic distance between nodes i and j . It is easy to verify the following necessary conditions for D .

Property 1: Let $D = [d_{ij}]$ be the synchronic distance matrix of a marked graph (G, M_0) . Then

$$\text{i) } d_{ii} = 0 \text{ for all } i.$$

$$\text{ii) } d_{ij} \leq d_{ik} + d_{kj} \text{ for all } i, j, k.$$

$$\text{iii) } d_{ij} \text{ is a nonnegative integer or } \infty.$$

$$\text{iv) } d_{ij} = d_{ji} \text{ for all } i, j. \quad \square$$

A matrix satisfying i) and ii) is called a distance matrix. It is well known that a matrix D is a distance matrix *iff*

$$D * D = D \quad (29)$$

where $*$ denotes the matrix multiplication in Carre's algebra, that is, addition $x + y$ is replaced by $\min\{x, y\}$ and multiplication $x \cdot y$ is replaced by addition $x + y$. If D is a synchronic distance matrix, then (29) holds.

Given a matrix D , we are interested in the problem of finding a marked graph whose synchronic distance matrix is D . The following procedure [220] gives a method for finding a marked graph when D is realizable as the distance matrix of a tree (undirected-graph) with positive integer arc weights.

Procedure for Finding a Marked Graph from a Given Matrix D :

Step 1. Test the necessary condition (29).

Step 2. Find a tree (a weighted undirected graph) by the following procedure:

Step 2.1. Find a maximum entry d_{\max} in D . List all rows i_0 in which d_{\max} is located. (Then node i_0 is a pendant node of a tree.)

Step 2.2. For each row i_0 , find a unique minimum off-diagonal entry d_{\min} . List the column j_r in which d_{\min} is located. Draw an arc between nodes i_0 and j_r with arc weight d_{\min} . (If d_{\min} is not unique, D is not realizable as an n -node tree.)

Step 2.3. Delete all the rows and columns in which d_{\max} is located.

Step 2.4. Repeat Steps 2.1 through 2.3 until $(n - 1)$ arcs of a tree are drawn.

Step 3. Replace each arc $e = (v_1, v_2)$ in the tree by a pair of oppositely directed arcs, $e_1 = (v_1, v_2)$ and $e_2 = (v_2, v_1)$. Assign initial tokens on e_1 and e_2 such that the sum of tokens on e_1 and e_2 equals the weight of the arc e . \square

Example 17: Find a marked graph whose synchronic distance matrix is given by

$$D = \begin{matrix} & \begin{matrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \end{matrix} \\ \begin{matrix} 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ 7 \end{matrix} & \begin{bmatrix} 0 & 1 & 3 & 6 & 6 & 5 & 6 \\ 1 & 0 & 2 & 5 & 5 & 4 & 5 \\ 3 & 2 & 0 & 3 & 3 & 2 & 3 \\ 6 & 5 & 3 & 0 & 6 & 5 & 6 \\ 6 & 5 & 3 & 6 & 0 & 1 & 2 \\ 5 & 4 & 2 & 5 & 1 & 0 & 1 \\ 6 & 5 & 3 & 6 & 2 & 1 & 0 \end{bmatrix} \end{matrix}$$

The maximum entry $d_{\max} = 6$ is found for the following rows $i_0 = 1, 4, 5, 7$. Thus, nodes 1, 4, 5, 7 are pendant nodes, as shown in Fig. 50(a). For each row $i_0 = 1, 4, 5, 7$, the unique minimum entry d_{\min} is located at column $j_i = 2, 3, 6, 6$, with $d_{\min} = 1, 3, 1, 1$, respectively. Thus, we know that arcs (1, 2), (4, 3), (5, 6), (7, 6) have weights 1, 3, 1, 1, respectively, as is shown in Fig. 50(b). Now, deleting the four rows and four columns for $i_0 = j_0 = 1, 4, 5, 7$, we have

$$D_1 = \begin{matrix} & \begin{matrix} 2 & 3 & 6 \end{matrix} \\ \begin{matrix} 2 \\ 3 \\ 6 \end{matrix} & \begin{bmatrix} 0 & 2 & 4 \\ 2 & 0 & 2 \\ 4 & 2 & 0 \end{bmatrix} \end{matrix}$$

For D_1 , $d_{\max} = 4$ is found in rows $i_0 = 2$ and 6. $d_{\min} = 2$ is found at (2, 3) and (6, 3), respectively. Thus, D_1 can be realized as the tree shown in Fig. 50(c). Therefore, from Fig. 50(b) and (c), we have the tree realization of the matrix D shown in Fig. 50(d), from which we find the marked graph shown in Fig. 50(e). It is easy to verify that the given matrix D is indeed the synchronic distance matrix of the MG shown in Fig. 50(e). \square

Note that given a matrix D satisfying Property 1, we can always find an undirected graph G whose distance matrix is D . For example, we can draw a complete graph G where the arc between nodes i and j has weight d_{ij} . However, application of Step 3 to G does not always result in a marked graph whose synchronic distance is D if G is not a tree. A necessary and sufficient condition for D to be the synchronic distance matrix of a marked graph (or a Petri net) is an open problem.

References [7], [218], [223] are suggested for further reading on the subject of marked graphs and their applications.

VIII. STRUCTURAL PROPERTIES

Structural properties are those that depend on the topological structures of Petri nets. They are independent of the initial marking M_0 in the sense that these properties hold for any initial marking or are concerned with the existence of certain firing sequences from some initial marking. Thus, these properties can often be characterized in terms of the incidence matrix A and its associated homogeneous equa-

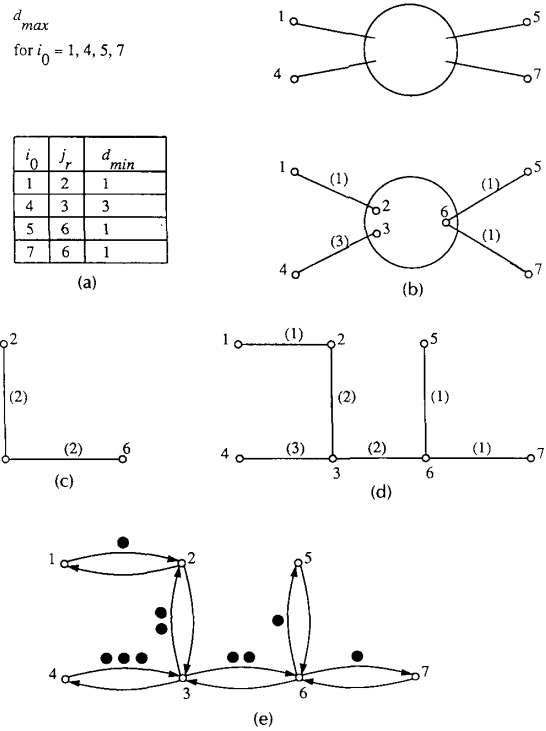


Fig. 50. Example 17: Synthesis of a synchronic distance matrix.

tions or inequalities. It is assumed that all nets considered in this section are pure. The i th entry of a vector x is denoted by $x(i)$. For two vectors x and y , $x > y$ means that $x(i) > y(i)$ for each i , $x \geq y$ means that $x(i) \geq y(i)$ for each i , and $x \neq y$ means that $x \geq y$ and $x(i) \neq y(i)$ for some i .

Structural Liveness: A Petri net N is said to be *structurally live* if there exists a live initial marking for N . It is easy to see from Theorem 7 that every marked graph is structurally live. Also from Theorem 12 we can see that a free-choice net is structurally live *iff* every siphon has a trap. A complete characterization of structural liveness for a general Petri net is unknown.

Controllability: A Petri net N is said to be *completely controllable* if any marking is reachable from any other marking.

Theorem 28: If a Petri net N with m places is completely controllable, then we have

$$\text{Rank } A = m. \quad (30)$$

Proof: If a Petri net is completely controllable, then (6) must have a solution x for any ΔM . Thus, from solvability of linear equations [199] we must have for any ΔM

$$\text{Rank } A^T = \text{Rank } [A^T : \Delta M]$$

which implies that the rank of an $m \times n$ matrix A^T must be of its full rank equal to m . \square

Note that (30) is only a necessary condition for complete controllability of Petri nets. However, the same condition (30) is sufficient as well as necessary for marked graphs. That is, a connected marked graph G is completely controllable *iff* (30) holds or G is a tree [198]. Because for a connected

marked graph G with n nodes, it is known that

$$\text{Rank } A = n - 1. \quad (31)$$

From (30) and (31), we have $m = n - 1$. That is, G has only $(n - 1)$ arcs to connect n nodes. This means that G is a circuitless connected graph, i.e., a tree. The controllability follows from Theorem 22.

Structural Boundedness: A Petri net N is said to be *structurally bounded* if it is bounded for any finite initial marking M_0 .

Theorem 29: A Petri net N is structurally bounded iff there exists an m -vector y of positive integers such that $Ay \leq 0$.

Proof: (\Leftarrow) Suppose

$$\exists y > 0, Ay \leq 0. \quad (32)$$

Let $M \in R(M_0)$. Then from (5), we have

$$M = M_0 + A^T x, x \geq 0. \quad (33)$$

Consider the inner product of M and y

$$M^T y = M_0^T y + x^T A y. \quad (34)$$

Since $Ay \leq 0$ and $x \geq 0$, we have

$$M^T y \leq M_0^T y. \quad (35)$$

Thus, $M(p)$, the number of tokens in each place p , is bounded by

$$M(p) \leq (M_0^T y) / y(p) \quad (36)$$

where $y(p)$ is the p th entry of y .

(\Rightarrow) Suppose (32) does not hold. Then by Minkowski-Farkas' lemma [247] or Case 4 in Table 4, there exists an

Table 4 One of Two Alternatives: Either System α or System β Has a Solution ("exclusive-or"). Case 1 and Case 2 (Minkowski-Farkas Lemma) are Well Known in the Theory of Linear Inequalities [247], [249]. Case 3 (Stiemke Theorem) and Case 4 are Special Cases (for $b \geq 0$ and $y > 0$) of Cases 1 and 2, Respectively

Case	System α	System β
1 (M-F)	$A^T x \geq b, x$ unrestricted	$Ay = 0, y \geq 0, y^T b > 0$
2 (M-F)	$A^T x \geq b, x \geq 0$	$Ay \leq 0, y \geq 0, y^T b > 0$
3 (Stiemke)	$A^T x \geq 0, x$ unrestricted (not conservative)	$Ay = 0, y > 0$ (conservative)
4 (Farkas)	$A^T x \geq 0, x \geq 0$ (not structurally bounded)	$Ay \leq 0, y > 0$ (structurally bounded)

n -vector $x \geq 0$ such that $A^T x \not\geq 0$. Then there exist two markings M and M_0 , such that $M - M_0 = A^T x \not\geq 0$ or $M \not\geq M_0$. Choose M_0 (and thus M) large enough so that a firing sequence σ , such that $\bar{\sigma} = x$, can be repeated indefinitely. The net will be unbounded. \square

A place p in a Petri net is said to be *structurally unbounded* if there exists a marking M_0 and a firing sequence σ from M_0 such that p is unbounded. It is easy to see that the following corollary holds.

Corollary: A place p in a Petri net N is structurally unbounded iff there exists an n -vector x of nonnegative integers such that $A^T x = \Delta M \not\geq 0$, where the p th entry of $\Delta M > 0$ (i.e., $\Delta M(p) > 0$).

Conservativeness: A Petri net N is said to be (partially) *conservative* if there exists a positive integer $y(p)$ for every (some) place p such that the weighted sum of tokens, $M^T y = M_0^T y = a$ constant, for every $M \in R(M_0)$ and for any fixed initial marking M_0 . It is easy to see from (34) that:

Theorem 30: A Petri net N is (partially) conservative iff there exists an m -vector y of positive (nonnegative) integers such that $Ay = 0, y \neq 0$.

Repetitiveness: A Petri net N is said to be (partially) *repetitive* if there exists a marking M_0 and a firing sequence σ from M_0 such that every (some) transition occurs infinitely often in σ .

Theorem 31: A Petri net N is (partially) repetitive iff there exists an n -vector x of positive (nonnegative) integers such that $A^T x \geq 0, x \neq 0$.

Proof: Suppose that there exists $x > 0$ such that $A^T x \geq 0$. Then there exist two markings M_0 and M such that $M - M_0 = A^T x \geq 0$ or $M \geq M_0$. Choose M_0 (and thus M) large enough that a firing sequence σ , such that $\bar{\sigma} = x$, can be repeated indefinitely. Then every transition will occur infinitely often in this firing sequence. The converse is also true. \square

Consistency: A Petri net N is said to be (partially) *consistent* if there exists a marking M_0 and a firing sequence σ from M_0 back to M_0 such that every (some) transition occurs at least once in σ .

Theorem 32: A Petri net N is (partially) consistent iff there exists an n -vector x of positive (nonnegative) integers such that $A^T x = 0, x \neq 0$.

Proof: Suppose a Petri net is consistent. Then from (5) there exists an $x > 0$ such that $M_0 = M_0 + A^T x$ or $A^T x = 0$. Conversely, suppose $x > 0, A^T x = 0$. Choose M_0 and M large enough that $M - M_0 = A^T x = 0$, so that a firing sequence σ , such that $\bar{\sigma} = x$, can be repeated. \square

It is obvious that conservativeness is a special case of structural boundedness and that consistency is a special case of repetitiveness. Partial conservativeness, consistency, and repetitiveness are the relaxation of positive vectors x or y to nonnegative vectors x or y . The complete characterizations (necessary and sufficient conditions) of these structural properties are summarized in Table 5. Table 6

Table 5 Necessary and Sufficient Conditions for Some Structural Properties

Symbols	Properties	Necessary and Sufficient Conditions
SB	Structurally Bounded	$\exists y > 0, Ay \leq 0$ (or $\nexists x > 0, A^T x \geq 0$)
CN	CoNservative	$\exists y > 0, Ay = 0$ (or $\nexists x, A^T x \geq 0$)
PCN	Partially CoNservative	$\exists y \geq 0, Ay = 0$
RP	RePetitive	$\exists x > 0, A^T x \geq 0$
PRP	Partially RePetitive	$\exists x \geq 0, A^T x \geq 0$
CS	ConSistent	$\exists x > 0, A^T x = 0$ (or $\nexists y, Ay \leq 0$)
PCS	Partially ConSistent	$\exists x \geq 0, A^T x = 0$

presents a list of corollaries that can be derived from the properties in Tables 4 and 5 using equations (33) and (34). It is helpful to understand the inequality conditions in Tables 4 and 5 if we interpret the expression $A^T x$ as the difference of markings in each place, and the expression Ay

Table 6 Additional Structural Properties

Case	If	Then
1	N is structurally bounded and structurally live	N is both conservative and consistent.
2	$\exists y \geq 0, Ay \leq 0$	\exists no live M_0 for N . N is not consistent.
3	$\exists y \geq 0, Ay \neq 0$	(N, M_0) is not bounded for a live M_0 . N is not consistent.
4	$\exists x \geq 0, A^T x \leq 0$	\exists no live M_0 for structurally bounded N . N is not conservative.
5	$\exists x \geq 0, A^T x \neq 0$	N is not structurally bounded. N is not conservative.

as the change in a weighted sum of tokens for each transition firing.

Example 18: From the definitions of structural properties, it is easy to analyze structural properties for each net shown in Fig. 51. The results are shown in Table 7, where each entry

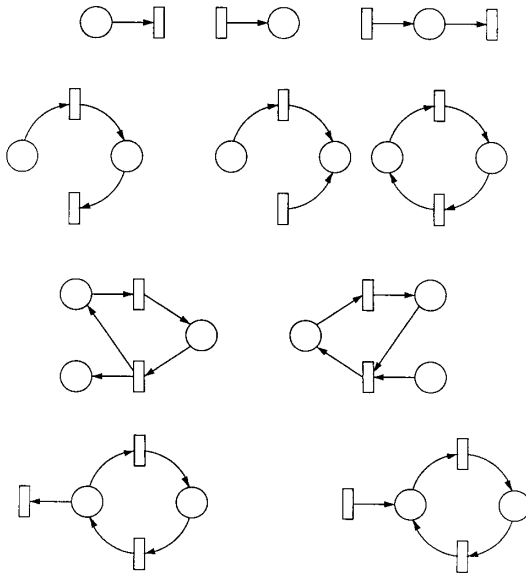


Fig. 51. Illustration of structural properties in Example 18.

Table 7 Structural Properties of the 10 Nets Shown in Fig. 51, Where + and - In Each Row Indicate Holding or Not Holding the Structural Property in the Row for Each Net Indicated in the Column

	a	b	c	d	e	f	g	h	i	j
Structurally bounded	+	-	-	+	-	+	-	+	+	-
Conservative	-	-	-	-	-	+	-	-	-	-
Partially conservative	-	-	-	-	-	+	+	+	-	-
Repetitive	-	+	+	-	-	+	+	-	-	+
Partially repetitive	-	+	+	-	+	+	+	-	+	+
Consistent	-	-	+	-	-	+	-	-	-	-
Partially consistent	-	-	+	-	-	+	-	-	+	+
Completely controllable	-	-	+	-	-	-	-	-	-	-
Structurally live	-	+	+	-	-	+	+	-	-	+
Structurally B-fair	+	+	-	+	-	+	+	+	+	-

+ or - in the table indicates whether or not the property in the corresponding row holds for the net indicated in the corresponding column, respectively. The matrix inequality conditions in Table 5 can be used to verify the results shown in Table 7 (except for structural B-fairness, which is discussed later in this section). □

Example: For the Petri net shown in Fig. 15, there exists $y = (1 \ 1 \ 0 \ 0 \ 1 \ 1)^T \geq 0$ such that $Ay = (0 \ 0 \ -1 \ 0 \ 0)^T \leq 0$. Therefore, by Case 2 in Table 6, this net is not live for any initial marking. □

S- and T-invariants: An m -vector y (n -vector x) of integers is called an *S-invariant* (*T-invariant*) if $Ay = 0$ ($A^T x = 0$). The following two theorems are obvious from the preceding discussion.

Theorem 33: An m -vector y is an S-invariant iff $M^T y = M_0^T y$ for any fixed initial marking M_0 and any M in $R(M_0)$. □

Theorem 34: An n -vector $x \geq 0$ is a T-invariant iff there exists a marking M_0 and a firing sequence σ from M_0 back to M_0 with its firing count vector $\bar{\sigma}$ equal to x . □

The set of places (transitions) corresponding to nonzero entries in an S-invariant $y \geq 0$ (T-invariant $x \geq 0$) is called the *support of an invariant* and is denoted by $\|y\|$ ($\|x\|$). A support is said to be *minimal* if no proper nonempty subset of the support is also a support. An invariant (vector) y is said to be *minimal* if there is no other invariant y_1 such that $y_1(p) \leq y(p)$ for all p . Given a minimal support of an invariant, there is a unique minimal invariant corresponding to the minimal support. We call such an invariant a *minimal-support invariant*. The set of all possible minimal-support invariants can serve as a generator of invariants. That is, any invariant can be written as a linear combination of minimal-support invariants [224].

Example 19: For the Petri net shown in Fig. 52, $x_1 = (1 \ 0 \ 1)^T$ and $x_2 = (0 \ 1 \ 1)^T$ are all possible minimal-sup-

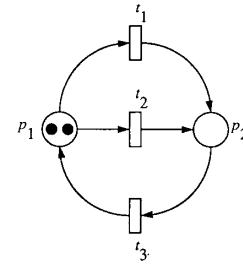


Fig. 52. Illustration of minimal-support T-invariants $(1 \ 0 \ 1)^T$ and $(0 \ 1 \ 1)^T$ in Example 19.

port T-invariants, where $\|x_1\| = \{t_1, t_3\}$ and $\|x_2\| = \{t_2, t_3\}$ are corresponding minimal supports. All other T-invariants such as $x_3 = (1 \ 1 \ 2)^T$ and $x_4 = (2 \ 1 \ 3)^T$ can be expressed as linear combinations of x_1 and x_2 . That is, $x_3 = x_1 + x_2$ and $x_4 = 2x_1 + x_2$. Note that there are many (non-unique) T-invariants such as x_3, x_4 , etc., corresponding to a nonminimal support $\{t_1, t_2, t_3\}$. (One easy way to find T-invariants in an example like this is to simulate all "firing sequences" which would reproduce a marking, using the concept of "negative or borrowed" tokens, if necessary.)

Example 20: For the Petri net shown in Fig. 53(a), $x_1 = (1 \ 1 \ 1 \ 1)^T$, $x_2 = (2 \ 0 \ 1 \ 1)^T$ and $x_3 = (0 \ 2 \ 1 \ 1)^T$ are

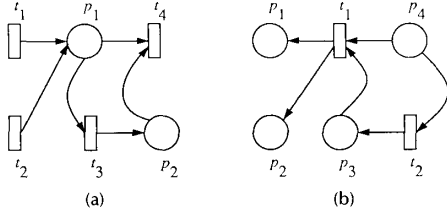


Fig. 53. Example 20: the net shown in (a) is the reverse-dual of the net shown in (b), and vice-versa. One is obtained from the other by transposing the incidence matrix.

three minimal T-invariants. However, only x_2 and x_3 are minimal-support T-invariants. The support of x_1 , $\{t_1, t_2, t_3, t_4\}$ is not minimal since its proper subset $\{t_1, t_3, t_4\}$ is the support of another T-invariant x_2 . In other words, the support of a minimal T-invariant is not necessarily a minimal support, although there is a unique minimal T-invariant corresponding to each minimal support [224]. x_1 can be expressed as a linear combination of x_2 and x_3 , namely $x_1 = (x_2 + x_3)/2$. The Petri net shown in Fig. 53(b) is the “reverse-dual” of the net shown in Fig. 53(a), i.e., the net obtained by transposing the incidence matrix. Therefore, all the above statements can apply to the net shown in Fig. 53(b) if T-invariants are replaced by S-invariants and t_i by p_i , $i = 1, 2, 3, 4$. \square

Equation (36) gives an upper bound on the number of tokens that place p can ever have. This upper bound can be improved if we apply (36) for all minimal-support S-invariants. That is,

$$M(p) \leq \text{Min} [M_0^T y_i / y_i(p)] \quad (37)$$

where the minimum is taken over all nonnegative minimal-support S-invariant y_i such that $y_i(p) \neq 0$. (It is shown in [179] that this upper bound can not be improved by using any other invariants.) For a marked graph, the set of arcs in a directed circuit is a minimal support S-invariant, and (37) reduces to Theorem 8.

Example 21: Consider the Petri net model of a readers-writers system shown in Fig. 11. Its incidence matrix A is given by

$$\begin{array}{c} p_1 \quad p_2 \quad p_3 \quad p_4 \\ \begin{array}{l} t_1 \\ t_2 \\ t_3 \\ t_4 \end{array} \begin{bmatrix} -1 & 1 & -1 & 0 \\ -1 & 0 & -k & 1 \\ 1 & -1 & 1 & 0 \\ 1 & 0 & k & -1 \end{bmatrix} \end{array}$$

It is easy to verify the following.

- 1) $Ay_1 = 0$ and $Ay_2 = 0$ for $y_1 = (1 \ 1 \ 0 \ 1)^T$ and $y_2 = (0 \ 1 \ 1 \ k)^T$. y_1 and y_2 are minimal-support S-invariants. Consider (37) for place p_4 and $M_0 = (k \ 0 \ k \ 0)^T$.

$$\begin{aligned} M(p_4) &\leq \text{Min} [M_0^T y_1 / y_1(p_4), M_0^T y_2 / y_2(p_4)] \\ &= \text{Min} [k/1, k/k] = 1. \end{aligned}$$

Thus, at most, one process can be in the state of writing as required in the readers-writers system.

- 2) The net is not completely controllable since $\text{Rank } A = 2 \neq m = 4$.
- 3) The net is SB, CN, PCN, RP, PRP, CS and PCS. \square

Structural B-Fairness: The concept of B-fairness discussed in Section IV-H can be extended to the following structural properties. Two transitions are said to be in a *structural B-fair relation* if they are in a B-fair relation for any initial marking. A Petri net is said to be *structurally B-fair* if it is a B-fair net for any initial marking. It is known [193] that:

- 1) A structural B-fair relation (as well as a B-fair relation) on the set of transitions T is an equivalence relation, and thus partitions T into equivalence classes.
- 2) Structural B-fairness implies B-fairness but the converse is not true. For example, the net (N, M_0) shown in Fig. 54(a) is a B-fair net. But N is not structurally B-fair since there is an initial marking M_1 such that (N, M_1) is not a B-fair net as is shown in Fig. 54(b), where

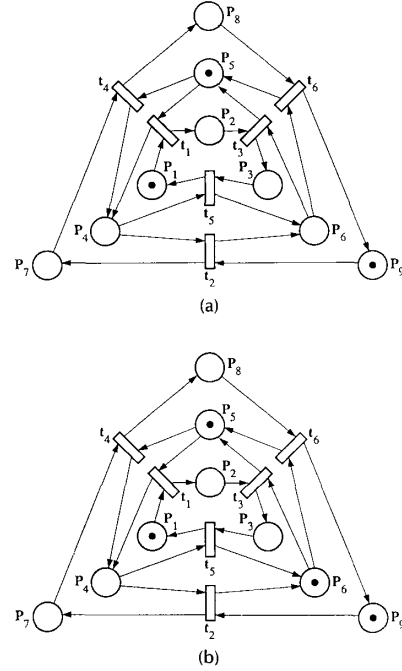


Fig. 54. An example of a live asymmetric choice net which is not structurally B-fair: (a) (N, M_0) is live and B-fair; (b) (N, M_1) is live but not B-fair, i.e., N is not structurally B-fair.

the firing sequence $t_1 \ t_3 \ t_5$ can be repeated infinitely often without firing t_2 , t_4 , or t_6 .

- 3) A structurally bounded net is structurally B-fair iff either a) it is consistent and there is only one reproduction vector (minimal nonnegative T-invariant $x \neq 0$), or b) it is not consistent and there is no reproduction vector.
- 4) Every strongly-connected marked graph is structurally B-fair.

References [185], [193], [224]–[226], [250]–[252] are suggested for further reading on structural properties.

IX. MODIFIED PETRI NETS AND THEIR APPLICATIONS

In this section, we discuss some modifications and extensions made on Petri nets that are useful for applications.

A. Timed Nets and Minimum Cycle Time

The concept of time is not explicitly given in the original definition of Petri nets. (See C. A. Petri's views [19] regarding the concepts of time and probability for Petri nets.) However, for performance evaluation and scheduling problems of dynamic systems, it is (at present) necessary and useful to introduce time delays associated with transitions and/or places in their net models. Such a Petri net model is known as a (deterministic) *timed net* if the delays are deterministically given, or as a *stochastic net* if the delays are probabilistically specified. The former is discussed in this subsection and the latter in the next subsection.

We are interested in finding how fast each transition can initiate firing in a periodically operated timed Petri net, where a period τ is defined as the time to complete a firing sequence leading back to the starting marking after firing each transition at least once. τ is called a *cycle time*. Thus, it is assumed that the net is consistent, i.e.,

$$\exists x > 0, A^T x = 0. \quad (38)$$

Suppose there is a delay of at least d_i sec associated with transition t_i , $i = 1, 2, \dots, n$. This means that when t_i is enabled, a_{ij} tokens will be reserved in place p_j for at least d_i sec before their removal by firing t_i , where a_{ij} is the weight of the arc from p_j to t_i . We define the *resource-time product (RTP)* as the product of the number of tokens (resources) and the length of time that these tokens reside in a place. Thus, the RTP is given by $a_{ij}d_i x_i$, which can be written in matrix form

$$(A^-)^T D x \quad (39)$$

where $A^- = [a_{ij}]_{n \times m}$ and D is the diagonal matrix of d_i , $i = 1, 2, \dots, n$. $(A^-)^T D x$ represents the vector of m RTP's for m places, and each RTP considers only reserved tokens. Now, suppose there are on the average $\bar{M}(p_i)$ tokens in place p_i during one cycle τ . Then, the RTP in the vector is given by $\bar{M}\tau$. Since the RTP obtained by this way of measuring includes both reserved and nonreserved tokens, we have the following inequality:

$$\bar{M}\tau \geq (A^-)^T D x. \quad (40)$$

Taking the inner product of (40) with a nonnegative S-invariant y and using the invariance, $y_k^T \bar{M} = y_k^T M_0$, we have

$$y_k^T M_0 \tau \geq y_k^T (A^-)^T D x$$

and

$$\tau \geq y_k^T (A^-)^T D x / y_k^T M_0. \quad (41)$$

Therefore, a lower bound of the cycle τ or the minimum cycle time is given by

$$\tau_{\min} = \max_k \{ y_k^T (A^-)^T D x / y_k^T M_0 \} \quad (42)$$

where the maximum is taken over all independent minimal-support S-invariants, $y_k \geq 0$.

If we model a timed Petri net by assigning delay d_i to each place p_i instead of the transitions, then it can be shown that τ_{\min} is given by

$$\tau_{\min} = \max_k \{ y_k^T D (A^+)^T x / y_k^T M_0 \} \quad (43)$$

where D is the diagonal matrix of d_i , $i = 1, 2, \dots, m$ and $A^+ = [a_{ij}^+]_{n \times m}$ with a_{ij}^+ being the weight of the arc from t_i to p_j .

For timed marked graphs, each directed circuit C_k yields a minimal-support S-invariant y_k . Thus, both (42) and (43) reduce to

$$\tau_{\min} = \max_k \{ \text{the total delay in } C_k / M_0(C_k) \}$$

where $M_0(C_k)$ denotes the number of tokens in C_k at M_0 . References [38], [44], [45], [46], [48], [49] are suggested for further reading on deterministic timed nets.

Example 22: Consider the Petri net shown in Fig. 11, and let the delay of transition t_i be d_i , $i = 1, 2, 3, 4$. From Example 21, we know that $y_1 = (1 \ 1 \ 0 \ 1)^T$ and $y_2 = (0 \ 1 \ 1 \ k)^T$ are two minimal-support S-invariants and that $x = (1 \ 1 \ 1 \ 1)^T > 0$ is a minimal positive T-invariant. Application of (42) yields

$$\begin{aligned} \tau_{\min} &= \max \{ (d_1 + d_2 + d_3 + d_4)/k, d_2 + d_4 \\ &\quad + (d_1 + d_3)/k \} \\ &= d_2 + d_4 + (d_1 + d_3)/k. \end{aligned} \quad \square$$

B. Stochastic Nets and Performance Modeling

Suppose the delay d_i associated with transition t_i is a non-negative continuous random variable X with the exponential distribution function

$$F_X(x) = \Pr[X \leq x] = 1 - e^{-\lambda_i x} \quad (44)$$

(or the probability density function, $f_X(x) = \lambda_i e^{-\lambda_i x}$).

Then, the average delay is given by

$$\bar{d}_i = \int_0^\infty [1 - F_X(x)] dx = \int_0^\infty e^{-\lambda_i x} dx = \frac{1}{\lambda_i} \quad (45)$$

where λ_i is the firing rate of transition t_i .

A *stochastic Petri net (SPN)* is a Petri net where each transition is associated with an exponentially distributed random variable that expresses the delay from the enabling to the firing of the transition. In a case where several transitions are simultaneously enabled, the transition that has the shortest delay will fire first. Due to the memoryless property of the exponential distribution of firing delays, it has been shown [40] that the reachability graph of a bounded SPN is isomorphic to a finite Markov Chain. The Markov Chain (MC) of a SPN can be obtained from the reachability graph of the Petri net (N, M_0) underlying the SPN as follows. The MC state space is the reachability set $R(M_0)$, and the transition rate from state M_i to state M_j is given by $q_{ij} = \lambda_i$, the (possibly marking-dependent) firing rate of transition t_i transforming M_i into M_j ($q_{ij} = \lambda_{i1} + \lambda_{i2} + \dots$, if there are two or more transitions t_{i1}, t_{i2}, \dots transforming M_i into M_j); $q_{ij} = 0$ if no transitions transforming M_i into M_j , $i \neq j$; and q_{ii} is determined so as to satisfy $\sum_j q_{ij} = 0$. The square matrix $Q = [q_{ij}]$ of order $s = |R(M_0)|$ is known as the *transition rate matrix* [30].

Let SPN (N, M_0) be reversible, i.e., $M_0 \in R(M_i)$ for every $M_i \in R(M_0)$. Then, the SPN generates an ergodic continuous-time MC and it is possible to compute the steady-state probability distribution Π by solving the linear system

$$\Pi Q = 0, \sum_{i=1}^s \pi_i = 1 \quad (46)$$

where π_i is the probability of being in state M_i and $\Pi = (\pi_1, \pi_2, \dots, \pi_s)$. From the steady-state distribution Π , it is possible to find various performance estimates of a system modeled by the SPN. For example,

1) *The probability of a particular condition:* Let B be the subset of $R(M_0)$ satisfying a particular condition. Then, the required probability is given by

$$P\{B\} = \sum_{i \in B} \pi_i. \quad (47)$$

2) *The expected value of the number of tokens:* Let $B(i, n)$ be the subset of $R(M_0)$ for which the number of tokens in a k -bounded place p_i is n . Then, the expected value of the number of tokens in place p_i is given by

$$E[m_i] = \sum_{n=1}^k [nP\{B(i, n)\}]. \quad (48)$$

3) *The mean number of firings in unit time:* Let B_i be the subset of $R(M_0)$ in which a given transition t_j is enabled. Then, the mean number of firings of t_j in unit time is given by

$$f_j = \sum_{M_i \in B_i} \pi_i \left(\frac{\lambda_j}{-q_{ii}} \right) \quad (49)$$

where λ_j is the firing rate of t_j and $-q_{ii}$ is the sum of firing rates of transitions enabled at M_i , i.e., the transition rate leaving state M_i .

Example 23: Consider the SPN shown in Fig. 55. Transition t_2 fires at a marking-dependent rate given by $m_2\lambda_2$,

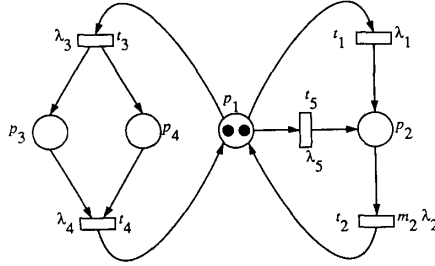


Fig. 55. The stochastic Petri net used in Example 23.

where m_2 is the number of tokens in p_2 . Transitions t_1, t_3, t_4 have (marking-independent) firing rates $\lambda_1, \lambda_3, \lambda_4$, respectively. The reachability graph and the MC of the SPN are shown in Fig. 56(a) and (b), respectively. The transition rate matrix Q is given by

$$Q = \begin{matrix} & \begin{matrix} M_0 \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \end{matrix} \end{matrix} \begin{bmatrix} -\lambda_{15} - \lambda_3 & \lambda_{15} & 0 & 0 & 0 & 0 \\ \lambda_2 & -\lambda_{15} - \lambda_2 - \lambda_3 & \lambda_{15} & 0 & 0 & 0 \\ 0 & 2\lambda_2 & -2\lambda_2 & 0 & 0 & 0 \\ \lambda_4 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_4 & 0 & 0 & 0 \end{bmatrix}$$

where $\lambda_{15} = \lambda_1 + \lambda_5$.

Let $\lambda_1 = \lambda_5 = 1/2$ and $\lambda_2 = \lambda_3 = \lambda_4 = 1$. Then, we can solve (46) numerically for Π and find $\pi_2 = 1/11$, $\pi_0 = \pi_1 = \pi_3 = \pi_4 = \pi_5 = 2/11$. Thus, for example, we can find the average number of tokens in place p_2 as follows: since p_2 has one

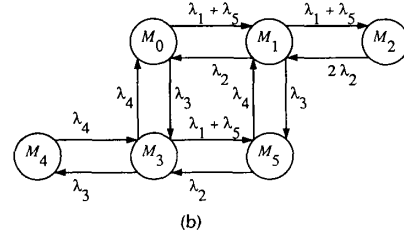
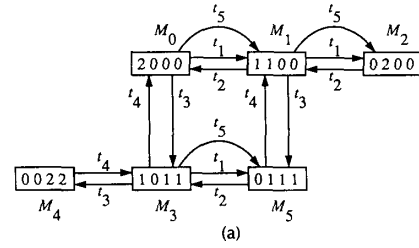


Fig. 56. (a) The reachability graph and (b) Markov chain of the SPN shown in Fig. 55.

token at M_1 and M_5 , and 2 tokens at M_2 , we have

$$E[m_2] = \pi_1 + \pi_5 + 2\pi_2 = 6/11.$$

Also, the mean number of firings of t_3 is given by

$$f_3 = \frac{\lambda_3}{\lambda_{15} + \lambda_3} \pi_0 + \frac{\lambda_3}{\lambda_{15} + \lambda_2 + \lambda_3} \pi_1 + \frac{\lambda_3}{\lambda_{15} + \lambda_3 + \lambda_4} \pi_3$$

$$= \frac{1}{2} \pi_0 + \frac{1}{3} \pi_1 + \frac{1}{3} \pi_3 = \frac{7}{33}$$

since t_3 is enabled at M_0, M_1, M_3 , and no other states. \square

SPNs have been extended to a class of generalized stochastic Petri nets (GSPN) [31] in order to cope with the state-space explosion problem. A GSPN has two types of transitions (timed and immediate). A timed transition has an exponentially distributed firing rate, and an immediate transition has no firing delay and is used to represent a logical control or an activity whose delay is negligible compared with those associated with timed transitions. Reduction of the state space is achieved by discarding vanishing markings that correspond to some intermediate states in which the system spends zero or negligible amount of time. When two or more immediate transitions are in conflict and enabled at the same time, the conflict must be resolved by specifying marking-dependent or independent branching

$$Q = \begin{matrix} & \begin{matrix} M_0 \\ M_1 \\ M_2 \\ M_3 \\ M_4 \\ M_5 \end{matrix} \end{matrix} \begin{bmatrix} \lambda_3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \lambda_3 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ -\lambda_{15} - \lambda_3 - \lambda_4 & \lambda_3 & \lambda_{15} & 0 & 0 & 0 \\ \lambda_4 & -\lambda_4 & 0 & 0 & 0 & 0 \\ \lambda_2 & 0 & -\lambda_2 - \lambda_4 & 0 & 0 & 0 \end{bmatrix}$$

probabilities. Useful results concerning stochastic Petri nets with generally distributed transition delays have been given in [34] and their model is called an extended stochastic Petri net (ESPN). They partition transitions into three classes: exclusive, competitive, and concurrent. ESPN can be

mapped onto semi-Markov processes, under the conditions that the firing delay of all concurrent transitions is exponentially distributed, and that competitive transitions resample a new firing delay whenever they are enabled. Using a similar approach, an embedded Markov chain technique has been presented for the analysis of DSPN (deterministic and stochastic Petri nets) containing both deterministic and stochastic transition firing delays [32]. Papers in two Proceedings [28], [29] and their references are suggested for further reading on these and other types of stochastic nets and their applications.

C. High-Level Nets and Logic Programs

High-level nets, in a broad sense, include predicate/transition nets [227], colored Petri nets [229], and nets with individual tokens [248]. A detailed discussion of these nets is beyond the scope of this paper. Here, we informally discuss only elementary aspects of high-level nets and their applications to modeling and analysis of logic programs.

We illustrate the transition firing rule of high-level nets using the simple predicate/transition net shown in Fig. 57.

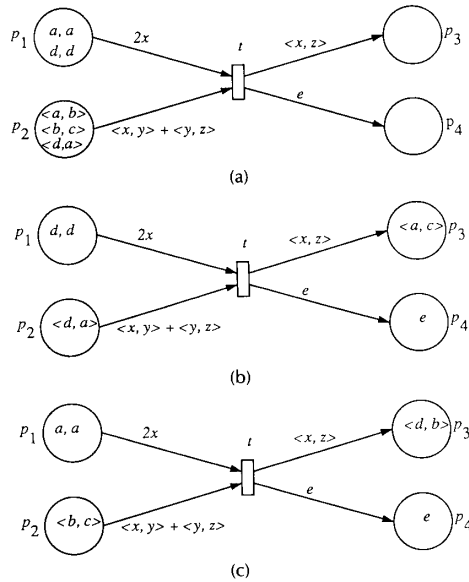


Fig. 57. Illustration of transition firing rule in a high-level net: (a) before firing, (b) after firing with substitution $\{a|x, b|y, c|z\}$, and (c) after firing with substitution $\{d|x, a|y, b|z\}$.

The net consists of one transition t and four places (two input places p_1 and p_2 and two output places p_3 and p_4). Note that the four arcs are labeled with $2x$, $\langle x, y \rangle + \langle y, z \rangle$, $\langle x, z \rangle$ and e . The arc label dictates how many and which kinds of "colored" tokens will be removed from or added to the places. For example, when the transition t in Fig. 57 fires, the following will occur:

- p_1 loses two tokens of the same color, x ;
- p_2 loses two tokens of different colors, $\langle x, y \rangle$ and $\langle y, z \rangle$;
- p_3 gets one token of the color, $\langle x, z \rangle$; and
- p_4 gets one token of the color, e (a constant).

The initial marking of the net consists of the following:

- p_1 has four colored tokens, two a 's and two d 's;
- p_2 has three colored tokens (ordered pairs), $\langle a, b \rangle$, $\langle b, c \rangle$ and $\langle d, a \rangle$;
- p_3 and p_4 have no tokens initially.

In the above, variables are denoted by x, y, z, \dots and constants are by a, b, c, d, \dots . For each transition, a variable of the same symbol appearing on incoming and outgoing arcs denotes the same variable. A constant of the same symbol is the same throughout the entire net. A transition t is said to be *enabled* if there are enough tokens of the "right" colors in each input place of t . Here, the "right" colors mean the existence of consistent substitutions of constants into variables, which are consistent with the arc labelings and possibly additional constraints. For example, the transition t in Fig. 57(a) is enabled since there are enough tokens in its input places and there are two consistent substitutions $\{a|x, b|y, c|z\}$ and $\{d|x, a|y, b|z\}$. Thus, there are two different (colored) ways of firing t with these two different substitutions. The nets shown in Fig. 57(b) and (c) show the markings after firing t with the substitutions $\{a|x, b|y, c|z\}$ and $\{d|x, a|y, b|z\}$, respectively.

A high-level net can be considered as a structurally folded version of a regular Petri net if the number of colors is finite. Thus, a high-level net can be unfolded into a regular Petri net by unfolding each place p into a set of places, one for each color of tokens which p may hold, and by unfolding each transition t into a set of transitions, one for each way that t may fire. For example, the high-level net shown in Fig. 57(a) can be unfolded into the regular Petri net shown in Fig. 58.

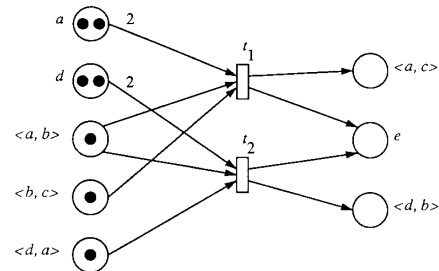


Fig. 58. The unfolded net of the high-level net shown in Fig. 57.

We now consider a high-level net representation of logic programs. A logic program consists of a set of Horn clauses written as

$$B \leftarrow A_1, A_2, \dots, A_n, \quad n \geq 0. \quad (50)$$

All the A_i 's and B are atomic formulae having the form $P(t_1, t_2, \dots, t_k)$, where P is a predicate symbol with k -ary argument, and the t_i 's are terms. A term can be a variable or a constant. Clause (50) states that B holds if A_1, A_2, \dots and A_n are true. It can be represented as a transition having n input places, A_1, A_2, \dots, A_n , and one output place B . When $n = 0$, eq. (50) represents the assertion of a fact, $B \leftarrow$, which corresponds to a source transition without input places. Another special form of (50) is $\leftarrow A_1, A_2, \dots, A_n, n \geq 1$, which

is a goal statement and corresponds to a sink transition without output places.

Consider a simple logic program consisting of the following five clauses:

- 1) Parent (David, Mary) \leftarrow
- 2) Parent (Mary, Tom) \leftarrow
- 3) Ancestor (x, y) \leftarrow Parent (x, y)
- 4) Ancestor (x, z) \leftarrow Parent (x, y), Ancestor (y, z)
- 5) \leftarrow Ancestor (x, Tom)

Clauses 1) and 2) state "David is a parent of Mary" and "Mary is a parent of Tom," respectively, and are assertions of facts. Clause 3) states, "x is an ancestor of y if x is a parent of y," and 4) states "x is an ancestor of z if x is a parent of y and y is an ancestor of z." Clause 5) is a goal statement saying "Who is an ancestor of Tom?"

A formal procedure for transforming a given logic program into a high-level net is described in [148]. Presented below is an informal method for converting a logic program into the incidence matrix of its high-level net.

Given a logic program consisting of n clauses and m distinct predicate symbols, the $n \times m$ incidence matrix $A = [a_{ij}]$ of a high-level net corresponding to the logic program can be found by the following procedure.

- Step 1) Each clause in the program will be one row of the matrix (one transition in the net).
- Step 2) Each distinct predicate symbol in the program will be one column of the matrix (one place of the net).
- Step 3) The (i, j) entry a_{ij} is the argument in the i th clause and in the j th predicate symbol, where an argument to the right of the \leftarrow is prefixed with a negative sign. If the j th predicate symbol appears more than once in the i th clause, then a_{ij} will be the formal sum of all those arguments in the i th row and j th column. \square

This procedure converts the above logic program example into the following incidence matrix:

$$A = \begin{array}{c} \begin{array}{cc} \text{Parent } (p_1) & \text{Ancestor } (p_2) \end{array} \\ \begin{array}{l} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{bmatrix} \langle D, M \rangle & 0 \\ \langle M, T \rangle & 0 \\ -\langle x, y \rangle & \langle x, y \rangle \\ -\langle x, y \rangle & -\langle y, z \rangle + \langle x, z \rangle \\ 0 & -\langle x, T \rangle \end{bmatrix} \end{array}$$

where D, M, T denotes David, Mary, and Tom, respectively.

From this incidence matrix, it is easy to draw the high-level net of the logic program shown in Fig. 59. There are two firing sequences σ_1 and σ_2 which start from the empty marking, fire the goal transition t_5 , and end at the empty marking. The first one σ_1 is as follows: fire t_2 to produce a token $\langle M, T \rangle$ in p_1 ; then fire t_3 to move the token $\langle M, T \rangle$ from p_1 to p_2 ; finally fire t_5 with substitution $\{M|x\}$, i.e., $x = \text{Mary}$ is an ancestor of Tom. The second firing sequence σ_2 is as follows: fire t_1 and t_2 to produce the two tokens $\langle D, M \rangle$, $\langle M, T \rangle$ in p_1 ; fire t_3 with substitution $\{M|x, T|y\}$ to move $\langle M, T \rangle$ from p_1 to p_2 ; then fire t_4 with substitution $\{D|x, M|y, T|z\}$ resulting a token $\langle D, T \rangle$ in p_2 ; finally fire t_5 with $\{D|x\}$, i.e., $x = \text{David}$ is another ancestor of Tom.

It should be noted that the above two firing sequences

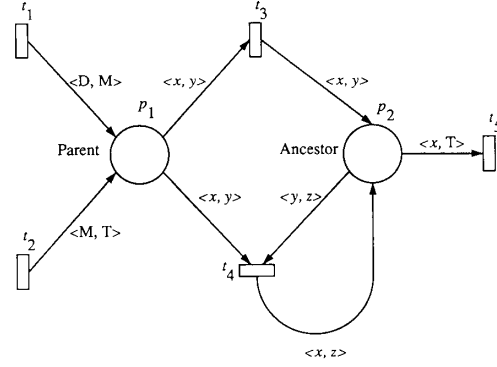


Fig. 59. A high-level net representation of a logic program.

σ_1 and σ_2 have the following substitution vectors X_1 and X_2 :

$$X_1 = \begin{array}{c} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{bmatrix} \emptyset \\ \{ \} \\ \{M|x, T|y\} \\ \emptyset \\ \{M|x\} \end{bmatrix}$$

and

$$X_2 = \begin{array}{c} t_1 \\ t_2 \\ t_3 \\ t_4 \\ t_5 \end{array} \begin{bmatrix} \{ \} \\ \{ \} \\ \{M|x, T|y\} \\ \{D|x, M|y, T|z\} \\ \{D|x\} \end{bmatrix}$$

where \emptyset denotes no firings and $\{ \}$ denotes a firing with no substitutions. The above vectors can be interpreted as "T-invariants" of the high-level net since they satisfy $A^T \circ X_1 = 0$ and $A^T \circ X_2 = 0$, where \circ denotes "matrix-product with substitutions" [148].

In general, the following theorem has been proved in [149].

Theorem 35: Let N be a high-level net representation of a Horn clause logic program (i.e., every transition in N has at most one output place). Let N be finitely colored and t_g be a goal transition. There exists a firing sequence which reproduces the empty marking and fires the goal transition t_g in N if and only if N has a nonnegative T-invariant X such that $X(t_g) \neq \emptyset$. \square

References [227]–[238], [253], [254], and [315] are suggested for further reading on high-level nets and their applications.

Several other modifications and extensions of Petri nets have been proposed. Examples of such nets are continuous Petri nets [239], FIFO nets [240], [255], place/transition (pta) nets [241], self-modifying nets [242], [243], and a hierarchy of nets [244]. Due to the space limitation, we have to refer the interested readers to the above references.

X. CONCLUDING REMARKS

What has been presented in this tutorial paper is a brief review of a rich body of knowledge in the field of Petri nets. It is not possible to discuss all aspects of the field in a single

paper. Thus, emphasis is placed on the area known as place/transition systems, as well as on applied Petri-net theory. Timed, stochastic, and high-level nets and their application examples deserve more space, since there is growing interest in these areas. However, a separate paper is necessary for a more comprehensive presentation of these subjects. The field is still young and much work remains to be done. We hope that this paper will help stimulate further research and developments in the emerging field of Petri nets.

ACKNOWLEDGMENT

Many individuals read part or all of an earlier version of this paper and made contributions for improving the presentation. The author wishes to thank M. Silva of U. Zaragoza, Spain, for his helpful suggestions on the entire manuscript; S. Kodama of Osaka Univ.; A. Ichikawa and K. Hiraishi of Tokyo Institute of Technology, Japan, for their inputs to Section VI-C; and the following colleagues and graduate students at the University of Illinois at Chicago: T. G. Moher, S. M. Shatz, J. P. Tsai, M. Aoyama, R. Bhatia, J. Jeffrey, M. Goto, D. J. Leu, H. Silver, V. Sliva, I. Suzuki (now with the University of Wisconsin at Milwaukee), T. Suzuki, S. Tu, and J. Yim for their useful comments. J. Yim typed the entire manuscript and drew all the figures using a graphics tool.

REFERENCES

- [1] C. A. Petri, "Kommunikation mit Automaten." Bonn: Institut für Instrumentelle Mathematik, Schriften des IM Nr. 3, 1962. Also, English translation, "Communication with Automata." New York: Griffiss Air Force Base. Tech. Rep. RADC-TR-65-377, vol. 1, Suppl. 1, 1966.
- [2] —, "Fundamentals of a theory of asynchronous information flow," in *Proc. IFIP Congress 62*, pp. 386-390, 1963.
- [3] A. W. Holt, H. Saint, R. Shapiro, and S. Warshall, *Final Report of the Information Systems Theory Project*, Tech. Rep. RADC-TR-68-305. New York: Griffiss Air Force Base, Sept. 1968.
- [4] A. W. Holt and F. Commoner, "Events and conditions," Princeton, N.J., Applied Data Research Inc., Information System Theory Project, 1970. Also, *Record Project MAC Conf. Concurrent Systems Parallel Computation*, pp. 3-52, 1970.
- [5] A. W. Holt, "Introduction to occurrence systems," in *Associative Information Techniques*, L. Jacks, Ed. New York: American Elsevier, pp. 175-203, 1971.
- [6] R. M. Shapiro and H. Saint, "A new approach to optimization of sequencing decisions," *Ann. Rev. Automatic Programming*, vol. 6, no. 5, pp. 257-288, 1970.
- [7] F. Commoner, A. W. Holt, S. Even, and A. Pnueli, "Marked directed graphs," *J. Comput. Syst. Sci.*, vol. 5, pp. 511-523, 1971.
- [8] F. Commoner, "Deadlocks in Petri nets," Wakefield, Applied Data Research, Inc., Report #CA-7206-2311, 1972.
- [9] J. B. Dennis, Ed., *Record Project MAC Conf. Concurrent Systems and Parallel Computation, June 1970*, 199 pages.
- [10] J. L. Peterson, *Petri Net Theory and the Modeling of Systems*. Englewood Cliffs, NJ: Prentice-Hall, Inc., 1981.
- [11] W. Reisig, *Petri Nets*, EATCS Monographs on Theoretical Computer Science, vol. 4. New York: Springer-Verlag, 1985.
- [12] T. Agerwala, "Putting Petri nets to work," *Computer*, vol. 12, no. 12, pp. 85-94, Dec. 1979.
- [13] R. Johnsonbaugh and T. Murata, "Petri nets and marked graphs—mathematical models of concurrent computation," *The American Math. Monthly*, vol. 89, no. 8, pp. 552-566, Oct. 1982.
- [14] J. L. Peterson, "Petri nets," *ACM Computing Surveys*, vol. 9, no. 3, pp. 223-252, Sept. 1977.
- [15] W. Brauer, Ed., *Net Theory and Applications*, Lecture Notes in Computer Science (LNCS), vol. 84. New York: Springer-Verlag, 1980, out of print.
- [16] W. Brauer, W. Reisig, and G. Rozenberg, Eds., *Petri Nets: Central Models and Their Properties*, Lecture Notes in Computer Science (LNCS), vol. 254, New York: Springer-Verlag, 1987.
- [17] W. Brauer, W. Reisig, and G. Rozenberg, (Eds.), *Petri Nets: Applications and Relationships to Other Models of Concurrency*, Lecture Notes in Computer Science (LNCS), vol. 255. New York: Springer-Verlag, 1987.
- [18] C. A. Petri, "Concurrency theory," in LNCS, vol. 254 [16].
- [19] —, "Forgotten topics" of net theory," in LNCS, vol. 255 [17].
- [20] C. Girault and W. Reisig, Eds., *Application and Theory of Petri Nets*, Selected Papers from the First and Second European Workshop on Applications and Theory of Petri Nets, Informatik-Fachberichte, Band 52. New York: Springer-Verlag, 1982.
- [21] A. Pagnoni and G. Rozenberg, Ed. *Applications and Theory of Petri Nets*, Selected Papers from the 3rd European Workshop on Applications and Theory of Petri Nets, Varenna, Italy, Sept. 1982. Informatik-Fachberichte, Band 66. New York: Springer-Verlag, 1983, out of print.
- [22] G. Rozenberg, Ed. *Advances in Petri Nets 1984*, Lecture Notes in Computer Science, vol. 188. New York: Springer-Verlag, 1985.
- [23] —, Ed. *Advances in Petri Nets 1985*, Lecture Notes in Computer Science, vol. 222. New York: Springer-Verlag, 1986.
- [24] —, Ed. *Advances in Petri Nets 1987*, Lecture Notes in Computer Science, vol. 266. New York: Springer-Verlag, 1987.
- [25] K. Voss, H. J. Genrich, and G. Rozenberg, Eds. *Concurrency and Nets*, Special volume in the series "Advances in Petri Nets." New York: Springer-Verlag, 1987.
- [26] S. Dress, D. Gomm, H. Plünneke, W. Reisig, and R. Walter, "Bibliography of Petri Nets," in LNCS, vol. 266 [24], pp. 319-451.
- [27] Petri Net Newsletters (published three times a year by Gesellschaft für Informatik, Postfach 1669, D-5300 Bonn 1, W. Germany, from which subscription applications forms can be obtained).
- [28] *Proc. Int. Workshop Timed Petri Nets*, Torino, Italy, July 1-3, 1985, 1985, 307 pages.
- [29] *Proc. Int. Workshop Petri Nets and Performance Models*, Madison, WI, August 24-26, 1987, 184 pages.
- [30] M. Ajmone Marsan, G. Balbo, and G. Conte, *Performance Models of Multiprocessor Systems*. Cambridge MA: The MIT Press, 1987.
- [31] M. Ajmone Marsan, G. Conte, and G. Balbo, "A class of generalized Petri nets for the performance evaluation of multiprocessor systems," *ACM Trans. Comput. Sys.*, vol. 2, no. 2, pp. 93-122, May 1984.
- [32] M. Ajmone Marsan and G. Chiola, "On Petri nets with deterministic and exponential transition firing times," in LNCS, vol. 266 [24], pp. 132-145, 1987.
- [33] H. H. Ammar, Y. F. Huang, and R. W. Liu, "Hierarchical models for systems reliability, maintainability, and availability," *IEEE Trans. Circuits Syst.*, vol. 34, no. 6, pp. 629-638, June 1987.
- [34] J. B. Dugan, K. S. Trivedi, R. M. Geist, and V. F. Nicola, "Extended stochastic Petri nets: Applications and analysis," *PERFORMANCE '84, Models of Comput. System Performance*, in *Proc. 10th Int. Symp.*, Paris, E. Gelenbe, Ed. Amsterdam: Elsevier, pp. 507-519, 1984.
- [35] K. Garg, "An approach to performance specification of communication protocols using timed Petri nets," *IEEE Trans. Software Eng.*, vol. SE-11, no. 10, pp. 1216-1225, Oct. 1985.
- [36] M. A. Holiday and Mary K. Venon, "A generalized timed Petri net model for performance analysis," *IEEE Trans. Software Eng.*, vol. SE-13, no. 12, pp. 1297-1310, Dec. 1987.
- [37] P. J. Hass and G. S. Shedler, "Stochastic Petri net representation of discrete event simulation," in the special section in Petri net performance models in *IEEE Trans. Software Eng.*, vol. 15, no. 4, pp. 381-393, Apr. 1989; an earlier version in [29].
- [38] J. Magott, "Performance evaluation of concurrent systems using Petri nets," *Inform. Processing Lett.*, vol. 18, no. 1, pp. 7-13, 1984.

- [39] J. F. Meyer, A. Movaghar, and W. H. Sanders, "Stochastic activity networks: structure behaviour and application," in *Proc. Int. Workshop Timed Petri Nets*, Torino, Italy, July 1-3, 1985, pp. 106-115, 1985.
- [40] M. K. Molloy, "Performance analysis using stochastic Petri nets," *IEEE Trans. Computers*, vol. C-31, no. 9, pp. 913-917, Sep. 1982.
- [41] —, "Discrete time stochastic Petri nets," *IEEE Trans. Software Eng.*, vol. SE-11, no. 4, pp. 417-423, April 1985.
- [42] T. Murata, "Use of resource-time product concept to derive a performance measure of timed Petri nets," in *Proc. 1985 Midwest Symp. Circuits Systems*, Aug. 19-20, 1985.
- [43] J. D. Noe and G. J. Nutt, "Macro E-nets for representation of parallel systems," *IEEE Trans. Comp.*, vol. TC-22, no. 8, pp. 718-727, Aug. 1973.
- [44] K. Onaga, K. Tani, and S. P. Chan, "Modeling and scheduling of resource-sharing concurrent processes in networks of recurrent multiprograms and multi-PERTs," in *Proc. 14th Asilomar Conf. Circuits, Systems, Computers*, D. E. Kirk, Ed., pp. 168-172, 1981.
- [45] C. V. Ramamoorthy and G. S. Ho, "Performance evaluation of asynchronous concurrent systems using Petri nets," *IEEE Trans. Software Eng.*, vol. SE-6, no. 5, pp. 440-449, Sept. 1980.
- [46] C. Ramchandani, "Analysis of asynchronous concurrent systems by timed Petri nets," Cambridge, MA: MIT, Project MAC, Tech. Rep. 120, Feb. 1974.
- [47] S. D. Shapiro, "A stochastic Petri net with applications to modelling occupancy times for concurrent task systems," *Networks*, vol. 9, no. 4, pp. 375-379, 1979.
- [48] J. Sifakis, "Use of Petri nets for performance evaluation," *Acta Cybernet.*, vol. 4, no. 2, pp. 185-202, 1978.
- [49] K. Tani and T. Murata, "Scheduling parallel computations with storage constraints," in *Proc. 12th Annual Asilomar Conf. Circuits, Systems, Computers*, Nov. 1978, pp. 736-743.
- [50] W. M. Zuberek, "M-timed Petri nets, priorities, preemptions, and performance evaluation of systems," in *Lecture Notes in Computer Science*, Vol. 222. New York: Springer-Verlag, pp. 478-498, 1986.
- [51] G. Berthelot and R. Terrat, "Petri nets theory for the correctness of protocols," *IEEE Trans. Commun.*, vol. COM-30, no. 12, pp. 2497-2505, Dec. 1982.
- [52] G. Bochmann and J. Gecsel, "A unified method for the specification and verification of protocols," *Information Processing 77*, IFIP, B. Gilchrist, Ed. Amsterdam: North-Holland, pp. 229-234, 1977.
- [53] M. Diaz and P. Azema, "Petri net based models for the specification and validation of protocols," in *Lecture Notes in Computer Science*, vol. 188. New York: Springer-Verlag, pp. 101-121, 1985.
- [54] M. Diaz, "Modeling and analysis of communication and cooperation protocols using Petri net based models," *Comput. Networks*, vol. 6, pp. 419-441, Dec. 1982.
- [55] G. Florin and S. Natkin, "Evaluation based upon stochastic Petri nets of the maximum throughput of a full duplex protocol," in *Informatik-Fachberichte 52* [20]. New York: Springer-Verlag, pp. 208-288, 1982.
- [56] G. Juanole, B. Algayres, and J. Dufau, "On communication protocol modeling and design," in *LNCS*, vol. 188, [22], pp. 267-287.
- [57] I. Lopez, "The use of GALILEO to represent and analyze telecommunications protocols," in *Proc. 2nd Europ. Workshop Appl. Theory Petri Nets*, Sept. 1981, pp. 377-410.
- [58] P. M. Merlin, "A methodology for the design and implementation of communication protocols," *IEEE Trans. Commun.* vol. COM-24, no. 6, pp. 614-621, June 1976.
- [59] P. M. Merlin and D. J. Farber, "Recoverability of communication protocols: Implications of a theoretical study," *IEEE Trans. Commun.*, vol. COM-24, no. 9, pp. 1036-1043, Sept. 1976.
- [60] R. Razouk and G. Estrin, "Modeling and verification of communication protocols in SARA: The X.21 interface," *IEEE Trans. Comput.*, vol. C-29, no. 12, Dec. 1980, pp. 1038-1052.
- [61] F. J. W. Symons, "Development and application of Petri net based techniques in Australia," in *Concurrency and Nets* [25], pp. 497-510, 1987.
- [62] G. R. Wheeler, M. C. Wilbur-Ham, J. Billington, and J. A. Gil-mour, "Protocol analysis using numerical Petri nets," in *Lecture Notes in Computer Science*, vol. 222 [23], pp. 435-452, 1986.
- [63] P. Azema and B. Berthomieu, "The design and validation by Petri nets of a mechanism for the invocation of remote servers," *Information Processing 80*, S. H. Lavington, Ed., pp. 599-604, 1980.
- [64] P. Azema, G. Juanole, E. Sanchis, and M. Montbernard, "Specification and verification of distributed systems using PROLOG interpreted Petri nets," in *Proc. 7th Int. Conf. Software Eng.*, Orlando, USA, 1984, pp. 510-518, 1984.
- [65] G. Estrin, R. S. Fenchel, R. R. Razouk, and M. K. Vernon, "SARA (System ARchitects Apprentice): Modeling, analysis, and simulation support for design of concurrent systems," *IEEE Trans. Software Eng.*, vol. SE-12, pp. 293-311, Feb. 1986.
- [66] B. Krämer, "Stepwise construction of non-sequential software system using a net-based specification language," in *LNCS*, vol. 188 [22], pp. 307-330.
- [67] D. Mandrioli, R. Zicari, C. Ghezzi, and F. Tisato, "Modeling the Ada task system by Petri nets," *Comput. Lang.*, vol. 10, no. 1, pp. 43-61, 1985.
- [68] L. J. Mekly and S. S. Yau, "Software design representation using abstract process networks," *IEEE Trans. Software Eng.*, vol. SE-6, no. 5, pp. 420-435, Sept. 1980.
- [69] T. Murata, B. Shenker, and S. M. Shatz, "Detection of Ada static deadlocks using Petri net invariants," *IEEE Trans. Software Eng.*, vol. 15, no. 3, pp. 314-326, Mar. 1989.
- [70] S. M. Shatz and W. K. Cheng, "Static analysis of Ada programs using the Petri net model," *Proc. ISCAS 85*, pp. 719-746, 1985.
- [71] S. S. Yau and M. U. Caglayan, "Distributed software system design representation using modified Petri nets," *IEEE Trans. Software Eng.*, vol. SE-9, no. 6, pp. 733-745, Nov. 1983.
- [72] J. B. Dugan and G. Ciardo, "Stochastic Petri net analysis of a replicated file system," in [29], also in the special section of Petri net performance models in *IEEE Trans. Software Eng.*, vol. 15, no. 4, pp. 394-401, Apr. 1989.
- [73] H. J. Genrich and K. Lautenbach, "The analysis of distributed systems by means of predicate/transition-nets," *Lecture Notes in Computer Science*, vol. 70, Semantics of Concurrent Computation. New York: Springer-Verlag, pp. 123-146, 1979.
- [74] M. T. Ozsu, "Modeling and analysis of distributed database concurrency control algorithms using an extended Petri net formalism," *IEEE Trans. Software Eng.*, vol. SE-11, no. 10, pp. 1225-1240, Oct. 1985.
- [75] K. Voss, "Using predicate/transition-nets to model and analyze distributed database systems," *IEEE Trans. Software Eng.*, vol. SE-6, no. 6, pp. 539-544, Nov. 1980.
- [76] E. Best, "COSY: Its relation to nets and to CSP," in *LNCS*, vol. 255 [17].
- [77] S. I. Baranov, L. N. Zhuravina, and V. A. Peshanskii, "Method of representing parallel flow charts of algorithms by sets of sequential flowcharts," (in Russian), English translation in *Autom. Control Comput. Sci.*, vol. 18, no. 5, pp. 71-78, 1984.
- [78] H. J. Genrich and P. S. Thiagarajan, "A theory of bipolar synchronization schemes," *Theoret. Comput. Sci.*, vol. 30, pp. 241-318, 1984.
- [79] U. Goltz and A. Mycroft, "On the relationship of CCS and Petri nets," in *Lecture Notes in Computer Science*, vol. 172, Automata, Languages and Programming, J. Paredaens, Ed. New York: Springer-Verlag, pp. 196-208, 1984.
- [80] U. Goltz and W. Reisig, "CSP-programs as nets with individual tokens," in *Lecture Notes in Computer Science*, vol. 188 [22], pp. 169-196, 1985.
- [81] R. M. Karp and R. E. Miller, "Properties of a model for parallel computations: Determinacy, termination, queuing," *SIAM J. Applied Math.*, vol. 14, no. 6, pp. 1390-1411, Nov. 1966.
- [82] —, "Parallel program schemata," *J. Comput. Syst. Sci.*, vol. 3, no. 2, pp. 147-195, May 1969.
- [83] T. Kasai and R. E. Miller, "Homomorphisms between models of parallel computation," *J. Comput. Sys. Sci.*, vol. 25, pp. 285-331, Dec. 1982.
- [84] R. M. Keller, "Parallel program schemata and maximal parallelism, I. Fundamental results," *J. ACM*, vol. 20, no. 3, pp. 514-537, July 1973.

- [85] —, "Parallel program schemata and maximal parallelism, II. Construction of closures," *J. ACM*, vol. 20, no. 4, pp. 696-710, Oct. 1973.
- [86] —, "Formal verification of parallel programs," *Comm. ACM*, vol. 19, no. 7, pp. 371-384, 1976.
- [87] P. E. Lauer and R. H. Campbell, "Formal semantics of a class of high-level primitives for coordinating concurrent processes," *Acta Informatica*, vol. 5, no. 4, pp. 297-332, 1975.
- [88] K. Lautenbach and H. A. Schmid, "Use of Petri nets for proving correctness of concurrent process systems," *Proc. IFIP Congress 74*, pp. 187-191, 1974.
- [89] K. Lautenbach and P. S. Thiagarajan, "Analysis of a resource allocation problem using Petri nets," in *Proc. 1st European Conf. Parallel Distributed Processing*, J. C. Syre, Ed., pp. 260-266, 1979.
- [90] R. E. Miller, "A comparison of some theoretical models of parallel computation," *IEEE Trans. Comp.*, Vol. TC-22, no. 8, pp. 710-717, 1973.
- [91] —, "Some relationships between various models of parallelism and synchronization," Yorktown Heights, New York, IBM T. J. Watson Research Center, Rep. RC-5074, Oct. 1974.
- [92] T. Murata, "Relevance of network theory to models of distributed/parallel processing," *J. Franklin Institute*, vol. 310, no. 1, pp. 41-50, 1980.
- [93] H. Alla, P. Ladet, J. Martinez, and M. Silva-Suarez, "Modelling and validation of complex systems by coloured Petri nets; application to a flexible manufacturing system," in *Lecture Notes in Computer Science*, vol. 188. New York: Springer-Verlag, pp. 15-31, 1985.
- [94] G. Bruno and G. Marchetto, "Process-translatable Petri nets for the rapid prototyping of process control systems," *IEEE Trans. Software Eng.*, vol. SE-12, no. 2, pp. 346-357, Feb. 1986.
- [95] D. Crockett, A. Desrochers, F. Dicesare, and T. Ward, "Implementation of a Petri net controller for a machine workstation," in *Proc. 1987 IEEE Int. Conf. Robotics Automat.*, pp. 1861-1867, 1987.
- [96] H. Demmou, M. Courvoisier, E. Thuriot, and R. Valette, "A new synchronization scheme for microprocessor based real-time control systems," in *Proc. 1983 Conf. Ind. Electron.*, pp. 237-242, 1983.
- [97] H. P. Lipp, "Application of a fuzzy Petri net for controlling complex industrial processes," in *IFAC Proc.*, Series 6, Pergamon Press, pp. 471-477, July 1983.
- [98] J. Martinez, H. Alla, and M. Silva, "Petri nets for the specification of flexible manufacturing systems," in *Modeling and Design of Flexible Manufacturing Systems*. New York: Elsevier Science Publ., pp. 389-406, 1986.
- [99] T. Murata, N. Komoda, K. Matsumoto, and K. Haruna, "A Petri net-based controller for flexible and maintainable sequence control and its application in factory automation," *IEEE Trans. Ind. Electron.* vol. 33, no. 1, pp. 1-8, 1986.
- [100] T. Murata and M. Silva (organizers), special sessions on Petri Nets and Flexible Manufacturing in *Proc. 1987 IEEE Int. Conf. Robotics Automat.*, pp. 999-1018 and pp. 1160-1185, 1987.
- [101] A. Ichikawa and K. Hiraishi, "Analysis and control of discrete event systems represented by Petri nets," in *Lecture Notes in Control and Information Sciences*, Vol. 103, Discrete Event Systems: Models and Applications, pp. 115-134. New York: Springer-Verlag, 1987.
- [102] S. Kodama (organizer), special sessions on Modeling, Analysis, and Control of Discrete Event Systems: Net Theoretical Approach, in *Proc. 1985 IEEE Int. Symp. Circuits Syst.*, Kyoto, Japan, pp. 471-498, pp. 719-746, pp. 917-940, June 1985.
- [103] B. H. Krogh, "Controlled Petri nets and maximally permissive feedback logic," *Proc. 25 Allerton Conf. Communication, Control and Computing*, pp. 317-326, Oct. 1987.
- [104] M. A. Holiday and M. K. Vernon, "Performance estimates for multiprocessor memory and bus interference," *IEEE Trans. Comput.*, vol. C-36, pp. 76-85, Jan. 1987.
- [105] W. E. Kluge and K. Lautenbach, "The orderly resolution of memory access conflicts among competing channel processes," *IEEE Trans. Comput.*, vol. C-31, no. 3, pp. 194-207, March 1982.
- [106] W. Kluge, "Reduction, data flow and control flow models of computation," in *LNCS*, vol. 255 [17], pp. 466-498, 1987.
- [107] S. Y. Kung, S. C. Lo, and P. S. Lewis, "Timing analysis and design optimization of VLSI dataflow arrays," in *Proc. 1986 Int. Conf. Parallel Processing*, also, a revised version, "Performance analysis and optimization of VLSI dataflow arrays," *J. Parallel and Distributed Computing*, vol. 4, no. 6, pp. 592-618, Dec. 1987.
- [108] T. Smigelski, T. Murata, and M. Sowa, "A timed Petri net model and simulation of dataflow computer," in *Proc. Int. Workshop Timed Petri Nets*, Torino, Italy, July 1-3, 1985, pp. 56-63, 1985.
- [109] N. G. Leveson and J. L. Stolzy, "Safety analysis using Petri nets," *IEEE Trans. Software Eng.*, vol. SE-13, no. 3, pp. 386-397, March 1987.
- [110] M. Lu, D. Zhang, and T. Murata, "Stochastic net model for self-stability measures of fault-tolerant clock synchronization," *Proc. Int. Workshop Petri Nets and Performance Models*, pp. 104-110, 1987.
- [111] S. Kumagai, H. Shiizuka, and S. Kodama, "Optimal realization of fault-tolerant decision-free concurrent systems," in *Proc. 28th Midwest Symposium Circuits and Systems*, pp. 253-256, 1985.
- [112] S. K. Paranjpe, A. B. Ektare, and D. P. Mital, "Fault diagnosis of alignment networks using Petri nets," *Int. J. Electron. (GB)*, vol. 56, no. 3, pp. 365-370, March 1984.
- [113] J. Sifakis, "Realization of fault-tolerant systems by coding Petri nets," *J. Design Automation and Fault Tolerant Computing*, vol. III, no. 1, pp. 93-107, 1979.
- [114] M. Silva and S. Velilla, "Error detection and correction on Petri net models of discrete events control systems," in *Proc. 1985 IEEE Int. Symposium Circuits and Systems*, pp. 921-924, 1985.
- [115] M. Courvoisier, R. Valette, J. M. Bigou, and P. Esteban, "A programmable logic controller based on a high level specification tool," in *Proc. 1983 Conf. Ind. Electron.*, pp. 174-179, 1983.
- [116] F. Distant, "A Petri net matrix approach in VLSI functional testing," *Microprocessing and Microprogramming*, vol. 16, no. 2-3, p. 194, 1985.
- [117] F. J. Rammig, "Hierarchical modulator description of VLSI systems," in Workshop Report, VLSI and Software Eng. Workshop, Silver Spring, MD, USA, IEEE Computer Society Press, pp. 112-116, 1983.
- [118] P. B. Sheridan, L. M. Haibt, and R. A. Nelson, "Casting Boolean logic arrays into Petri nets," IBM Research Rep., RC 10130, (45036), IBM T. J. Watson Research Center, New York, Aug. 1983.
- [119] M. Silva and S. Velilla, "Programmable logic controllers and Petri nets: A comparative study," *IFAC Software for Computer Control, Madrid, Spain*, G. Ferrate, E. A. Puente, Eds. Oxford, England, Pergamon, pp. 83-88, 1982.
- [120] R. Valette, M. Courvoisier, J. M. Begou, and J. Albuquerque, "Petri net based programmable logic controllers," in *Proc. 1st Int. IFIP Conf.: Comp. Appl. in Production and Engineering*, pp. 103-116, 1983.
- [121] J. B. Dennis, "Modular, asynchronous control structures for a high performance processor," in *Proc. Project MAC Conference*, pp. 55-80, 1970.
- [122] J. B. Dennis and S. S. Patil, "Speed independent asynchronous circuits," in *Proc. 4th Hawaii Int. Conf. Syst. Sci.*, pp. 55-58, 1971.
- [123] J. R. Jump and P. S. Thiagarajan, "On the equivalence of asynchronous control structures," in *Proc. 13th Annual Switching and Automata Theory Symp.*, pp. 212-223, Oct. 1972.
- [124] J. R. Jump, "Asynchronous Control Arrays," *IEEE Trans. Comput.*, vol. TC-23, no. 10, pp. 1020-1029, Oct. 1974.
- [125] J. R. Jump and P. S. Thiagarajan, "On the interconnection of asynchronous control structures," *J. ACM*, vol. 22, no. 4, pp. 596-612, Oct. 1975.
- [126] D. P. Misunas, "Petri nets and speed independent design," *Comm. ACM* 16, No. 8, pp. 474-481, 1973.
- [127] S. S. Patil, "Coordination of asynchronous events," Mass. Inst. of Tech., Project MAC, Tech. Rep. 72, June 1970.
- [128] —, "An asynchronous logic array," Mass. Inst. of Tech., Project MAC, Comp. Struct. Group Memo 111-1, Feb. 1975.
- [129] M. Yoeli, "Specification and verification of asynchronous

- circuits using marked graphs," in *Concurrency and Nets* [25], pp. 605-622, 1987.
- [130] J. L. Baer and C. A. Ellis, "Model design and evaluation of a compiler for a parallel processing environment," *IEEE Trans. Software Eng.*, vol. SE-3, no. 6, pp. 394-405, Nov. 1977.
 - [131] J. D. Noe, "A Petri net model of the CDC 6400," *Proc. ACM/SIGOPS Workshop on Systems Performance Evaluation*, pp. 362-378, 1971.
 - [132] F. De Cindio, G. De Michelis, and C. Simone, "GAMERU: A language for the analysis and design of human communication pragmatics within organizational systems," in *LNCS*, vol. 266 [24], pp. 21-44, 1987.
 - [133] C. A. Ellis and G. J. Nutt, "Office information systems and computer science," *Computing Surveys*, vol. 12, no. 1, pp. 27-60, March 1980.
 - [134] A. W. Holt, "Coordination technology and Petri nets," in *Lecture Notes in Computer Science*, vol. 222 [23], pp. 278-296, 1986.
 - [135] M. D. Zisman, "Representation, specification and automation of office procedures," Philadelphia, PA.: University of Pennsylvania, Wharton School, Department of Decision Sciences, Ph.D. Thesis, Sept. 1977.
 - [136] S. Crespi-Reghezzi and D. Mandrioli, "Petri nets and szilard languages," *Information and Control*, vol. 33, no. 2, pp. 177-192, Feb. 1977.
 - [137] —, "Some algebraic properties of Petri nets," *Alta Frequenza*, vol. 45, no. 2, pp. 130-137, Feb. 1976.
 - [138] M. Hack, "Petri net languages," *Comp. Struct. Group Memo* 124, Project MAC, MIT, 1975.
 - [139] D. Mandrioli, "A note on Petri net languages," *Information and Control*, vol. 34, no. 2, pp. 169-171, 1977.
 - [140] J. L. Peterson, "Computation sequence sets," *J. Comput. Syst. Sci.*, vol. 13, no. 1, pp. 1-24, Aug. 1976.
 - [141] A. Mazurkiewicz, "Trace theory," in *LNCS*, vol. 255 [17], pp. 279-324, 1987.
 - [142] G. Rozenberg and R. Verraedt, "Subset languages of Petri nets part II: the relationship to string languages and normal forms," *Theoret. Comput. Sci.*, vol. 26, pp. 301-326, 1983.
 - [143] J. L. Darlington, "A net based theorem prover for program verification and synthesis," *Gesellschaft für Math. und Datenverarbeitung mbH Bonn, Interner Bericht des IST 3/79 Dez.* 1979.
 - [144] H. J. Genrich and G. Thieler-Mevissen, "The calculus of facts," in *Mathematical Foundations of Computer Science 1976*, A. Mazurkiewicz, Ed. New York: Springer-Verlag, pp. 588-595, 1976.
 - [145] A. Giordana and L. Saitta, "Modeling production rules by means of predicate transition networks," *Inform. Sci.*, vol. 35, pp. 1-41, 1985.
 - [146] K. Lautenbach, "On logical and linear dependencies," *Gesellschaft für Math. und Datenverarbeitung mbH Bonn, Arbeitspapiere der GMD Nr. 147*, March 1985.
 - [147] T. Murata and K. Matsuyama, "Inconsistency check of a set of clauses using Petri net reductions," *J. Franklin Institute*, vol. 325, no. 1, pp. 73-93, 1988.
 - [148] T. Murata and D. Zhang, "A predicate-transition net model for parallel interpretation of logic programs," *IEEE Trans. Software Eng.*, vol. 14, no. 4, pp. 481-497, April 1988.
 - [149] G. Peterka and T. Murata, "Proof procedure and answer extraction in Petri net model of logic programs," *IEEE Trans. Software Eng.*, vol. 15, no. 2, pp. 209-217, Feb. 1989.
 - [150] G. Thieler-Mevissen, "The Petri net calculus of predicate logic," *St. Augustin: Gesellschaft für Mathematik und Datenverarbeitung Bonn, Interner Bericht ISF-76-09*, 1976.
 - [151] M. Ajmone Marsan, G. Chiola, and A. Fumagalli, "An accurate performance model of CSMA/CD bus LAN," in *LNCS*, vol. 266 [24], pp. 146-161, 1987.
 - [152] E. Gressier, "A stochastic Petri net model for Ethernet," in *Proc. Int. Workshop on Timed Petri Nets*, Torino, Italy, July 1-3, 1985, pp. 296-303, 1985.
 - [153] K. Voss, "A net model of a local area network protocol," in *Lecture Notes in Computer Science*, vol. 188, *Advances in Petri Nets 1984*. New York: Springer-Verlag, pp. 413-437, 1985.
 - [154] J. A. Meldman, "A Petri-net representation of civil procedure," *IDEA: J. Law Technol.*, vol. 19, no. 2, pp. 123-148, 1978.
 - [155] W. R. van Biljon, "Extending Petri nets for specifying man-machine dialogues," *Int. J. Man-Machine Studies*, vol. 28, no. 4, pp. 437-455, 1988.
 - [156] H. Oberquelle, I. Kupka, and H. Maass, "A view of human-machine communication and cooperation," *Int. J. Man-Machine Studies*, vol. 19, pp. 309-333, 1983.
 - [157] N. Stoica and G. Roth, "Neuronal networks modelled by Petri type nets with controllers," in *Proc. 12th IFIP Conf., Budapest*, in *Lecture Notes in Control Inf. Sci.*, pp. 923-932, 1986.
 - [158] M. R. Zargham and M. Tyman, "Neural Petri nets," in *Proc. Int. Workshop Timed Petri Nets*, Torino, Italy, pp. 72-77, 1985.
 - [159] S. C. Brofferio, "A Petri net control unit for high-speed modular signal processors," *IEEE Trans. Commun.*, vol. C35, no. 6, pp. 577-583, June 1987.
 - [160] R. Nouta and O. Simula, "On the use of modified Petri nets for multiprocessor realization of digital filters," in *Circuit Theory and Design*, G. S. Moschytz, J. Niernynch, Eds., St. Saphorin, Georgi, pp. 315-319, 1978.
 - [161] W. M. Zuberek, "Application of timed Petri nets to analysis of multiprocessor realizations of digital filters," in *Proc. 25th Midwest Symp. Circuits and Systems*, pp. 134-139, 1982.
 - [162] D. Tabak and A. H. Levis, "Petri net representation of decision models," *IEEE Trans. Syst. Man, Cybern.*, vol. SMC-15, no. 6, pp. 812-818, Nov-Dec. 1985.
 - [163] F. Feldbrugge and K. Jensen, "Petri net tool overview 1986," in *LNCS*, vol. 255 [17], pp. 20-61, 1987.
 - [164] J. Billington, G. R. Wheeler, and M. C. Wilbur-Ham, "PROTEAN: A High-level Petri net tool for the specification and verification of communication protocols," *IEEE Trans. Software Eng.*, vol. 14, no. 3, pp. 301-316, March 1988.
 - [165] G. Chiola, "A software package for the analysis of generalized stochastic Petri net models," in *Proc. Int. Workshop Timed Petri Nets*, Torino, Italy, July 1-3, 1985, pp. 136-143, 1985.
 - [166] J. B. Dugan, A. Bobbio, G. Ciardo, and K. S. Trivedi, "The design of a unified package for the solution of stochastic Petri net models," in *Proc. Workshop Timed Petri Nets*, Torino, Italy, July 1-3, 1985, pp. 6-13, 1985.
 - [167] E. Le Mer, "OVIDE: A software package for verifying and validating Petri nets," in *Software for Computer Control* (Proc. Third IFAC/IFIP Symposium, Madrid), G. Ferrate, E. A. Puente, Eds. Oxford: Pergamon Press, pp. 255-260, 1983.
 - [168] J. Martinez and M. Silva, "A package for computer design of concurrent logic control systems," in *Software for Computer Control* (Proc. Third IFAC/IFIP Symposium, Madrid, Oct. 5-8, 1982), G. Ferrate, E. A. Puente, Eds. Oxford: Pergamon Press, pp. 243-248, 1983.
 - [169] M. K. Molloy, "A CAD tool for stochastic Petri nets," in *Proc. 1986 Fall Joint Comp. Conf.*, pp. 1082-1091, 1986.
 - [170] M. A. Holliday and M. K. Vernon, "The GTPN analyzer: numerical methods and user interface," in *Proc. 1986 Fall Joint Computer Conf.*, pp. 1099-1105, 1986.
 - [171] R. Devillers, "The semantics of capacities in P/T nets: A first look," in *proc. 6th European Workshop Appl. Theory Petri Nets*, Espoo, Finland, pp. 171-190, 1985.
 - [172] C. A. Petri, "Concurrency as a basis of systems thinking," *St. Augustin: Gesellschaft für Mathematik und Datenverarbeitung Bonn, Interner Bericht ISF-78-06*, Sept. 1978.
 - [173] T. Agerwala, "A complete model for representing the coordination of asynchronous processes," *Baltimore: John Hopkins University, Hopkins Computer Science Program, Res. Rep. No. 32*, July 1974.
 - [174] S. R. Kosaraju, "Decidability of reachability in vector addition systems," in *Proc. 14th Annual ACM Symp. Theory Computing*, San Francisco, May 5-7, 1982, pp. 267-281, 1982.
 - [175] E. W. Mayr, "An algorithm for the general Petri net reachability problem," *SIAM, J. Comput.* vol. 13, no. 3, pp. 441-460, Aug. 1984.
 - [176] M. Hack, "Decidability questions for Petri nets," *Cambridge, Mass., MIT, Dept. Electrical Engineering, Ph.D. Thesis*, Dec. 1975.
 - [177] —, "The equality problem for vector addition systems is undecidable," *Theoret. Comput. Sci.*, vol. 2, pp. 77-95, 1976.
 - [178] K. Lautenbach, "Liveness in Petri nets," *St. Augustin, Gesellschaft für Mathematik und Datenverarbeitung Bonn, Interner Bericht ISF-75-02.1*, 1975.

- [179] M. Silva, *Petri Nets in Automation and Computer Engineering*. Madrid, Spain, Editorial AC, (in Spanish) 1985; English translation to be published by Kluwer Academic Press, Hingham, MA, 1990 (planned).
- [180] L. H. Landweber and E. L. Robertson, "Properties of conflict free and persistent Petri nets," *J. ACM*, vol. 25, no. 3, pp. 352-364, July 1978.
- [181] C. A. Petri, "Interpretations of net theory," St. Augustin: Gesellschaft für Mathematik und Datenverarbeitung Bonn, Interner Bericht ISF-75-07, Second Edition Dez. 1976.
- [182] T. Murata and Z. Wu, "Fair relation and modified synchronic distances in a Petri net," *J. Franklin Inst.*, vol. 320, no. 2, pp. 63-82, Aug. 1985.
- [183] H. J. Genrich, K. Lautenbach, and P. S. Thiagarajan, "Elements of general net theory," *Lecture Notes in Computer Science*, vol. 84 [15], pp. 21-163, 1980.
- [184] U. Goltz and Y. Chong-Yi, "Synchronic structure—A tutorial," *Lecture Notes in Computer Science*, vol. 222 [23], pp. 233-252, 1986.
- [185] M. Silva, "Toward a synchrony theory for P/T nets," in *Concurrency and Nets* [25], pp. 435-460, 1987.
- [186] I. Suzuki and T. Kasami, "Three measures for synchronic dependence in Petri nets," *Acta Informatica*, vol. 19, no. 4, pp. 325-338, 1983.
- [187] D. Leu, M. Silva, J. M. Colom, and T. Murata, "Interrelationships among various concepts of fairness for Petri nets," in *Proc. 31th Midwest Symposium Circuits and Systems*, 1988.
- [188] E. Best, "Fairness and conspiracies," in *Inform. Process. Lett.* 18. New York: North-Holland, pp. 215-220, 1984.
- [189] H. Carstensen and R. Valk, "Infinite behaviour and fairness in Petri nets," *Lecture Notes in Computer Science*, vol. 188 [22], pp. 83-100, 1985.
- [190] N. Francez, *Fairness*. New York: Springer-Verlag, 1986.
- [191] A. Pagnoni, "A fair competition between two or more partners," in *Informatik-Fachberichte* 52 [20], pp. 327-337, 1982.
- [192] J. P. Queille and J. Sifakis, "Fairness and related properties in transition systems—A temporal logic to deal with fairness," *Acta Informatica*, 19, pp. 195-220, 1983.
- [193] M. Silva and T. Murata, "B-fairness and structural B-fairness in Petri net models of concurrent systems," Tech. Rep. No. UIC-EECS-86-10, Univ. of Illinois at Chicago, June 1986.
- [194] Z. Wu and T. Murata, "Use of Petri nets for distributed control of fairness in concurrent systems," in *Proc. First Conf. Computers Applications*, Peking, 1984, pp. 84-91, 1984.
- [195] —, "A Petri net model of a starvation-free solution to the dining philosopher's problem," IEEE Workshop on Languages for Automation, Chicago, November 7-9, 1983, pp. 192-195, 1983.
- [196] R. Valk, "Infinite behaviour and fairness," in *LNCS*, vol. 254 [16], pp. 377-396, 1987.
- [197] D. Leu, T. Murata, and M. Silva, "Maximum firing deviation and fair relations in Petri nets," *Proc. 1986 IEEE Int. Symp. Circuits Syst.*, pp. 1008-1010, May 1986.
- [198] T. Murata, "State equation, controllability, and maximal matchings of Petri nets," *IEEE Trans. Automat. Contr.*, vol. AC-22, no. 3, pp. 412-416, Jun. 1977.
- [199] F. E. Hohn, *Elementary Matrix Algebra*. Macmillan: New York, 1958.
- [200] G. Berthelot, G. Roucairol, and R. Valk, "Reduction of nets and parallel programs," in *Lecture Notes in Computer Science*, vol. 84 [15], pp. 277-290, 1980.
- [201] G. Berthelot, "Checking properties of nets using transformations," in *Lecture Notes in Computer Science*, vol. 222 [23], pp. 19-40, 1986.
- [202] R. Johnsonbaugh and T. Murata, "Additional methods for reduction and expansion of marked graphs," *IEEE Trans. Circuit Syst.*, vol. CAS-28, no. 10, pp. 1009-1014, Oct. 1981.
- [203] T. Murata and J. Y. Koh, "Reduction and expansion of live and safe marked graphs," *IEEE Trans. Circuits Syst.*, vol. CAS-27, no. 1, pp. 68-70, Jan. 1980.
- [204] I. Suzuki and T. Murata, "A method for stepwise refinements and abstractions of Petri nets," *J. Comput. Syst. Sci.*, vol. 27, no. 1, pp. 51-76, Aug. 1983.
- [205] T. A. Chu, "A method of abstraction for Petri nets," pp. 164-173, in [29].
- [206] M. Hack, "Analysis of production schemata by Petri nets," Cambridge, Mass., MIT, Dept. Electrical Engineering, MS Thesis, 1972.
- [207] E. Best, "Structural theory of Petri nets: The free choice hiatus," in *Lecture Notes in Computer Science*. New York: Springer-Verlag, vol. 254, pp. 168-206, 1987.
- [208] E. Best and P. S. Thiagarajan, "Some classes of live and safe Petri nets," in *Concurrency and Nets* [25], pp. 71-94, 1987.
- [209] A. T. Amin and T. Murata, "A characterization of live and safe markings of a directed graph," in *Proc. 10th Conf. Inform. Sci. Syst.*, pp. 295-299, 1976.
- [210] M. Kinuyama and T. Murata, "Generating siphons and traps by Petri-net representation of logic equations," in *Proc. 2nd IECE (Japan) Conference on Net Theory*, pp. 93-100, Dec. 1986.
- [211] P. S. Thiagarajan and K. Voss, "A fresh look at free choice nets," *Inform. Contr.*, vol. 61, no. 2, pp. 85-113, May 1984.
- [212] A. Ichikawa and K. Hiraishi, "A class of Petri nets that a necessary and sufficient condition for reachability is obtainable," *Trans. Society of Instrument and Control Engineers (SICE)* (in Japanese), vol. 24, no. 6, 1988.
- [213] S. Kodama and T. Murata, "On necessary and sufficient reachability conditions for some subclasses of Petri nets," Tech. Rep. #UIC-EECS 88-8, Univ. of Illinois at Chicago, June 1988.
- [214] M. Jantzen and R. Valk, "Formal properties of place/transition nets," *Lecture Notes in Computer Science*, vol. 84 [15], 1980.
- [215] T. Murata, "Circuit theoretic analysis and synthesis of marked graphs," *IEEE Trans. Circuits Syst.*, vol. CAS-24, no. 7, pp. 400-405, July 1977.
- [216] S. Kumagai, S. Kodama, and M. Kitagawa, "Submarking reachability of marked graphs," *IEEE Trans. Circuits Syst.*, vol. CAS-31, no. 2, pp. 159-164, 1984.
- [217] M. A. Comeau and K. Thulasiraman, "Structure of the submarking-reachability problem and network programming," *IEEE Trans. Circuits Syst.*, vol. 35, no. 1, pp. 89-100, Jan. 1988.
- [218] F. Commoner et al., "Final report for the project-development of theoretical foundations for description and analysis of discrete information systems," vol. II-Mathematics, CADD-7405-2011, Mass. Comp. Assoc., Inc., Wakefield, MA 01880, May 1974.
- [219] T. Murata, "Synthesis of decision-free concurrent systems for prescribed resources and performance," *IEEE Trans. Software Eng.*, vol. SE-6, no. 6, pp. 525-530, Nov. 1980.
- [220] T. Murata, V. B. Le, and D. J. Leu, "Method for realizing the synchronic distance matrix as a marked graph," in *Proc. IEEE Int. Symp. Circuits Syst.*, Rome, Italy, vol. 2, pp. 609-612, May 1982.
- [221] D. J. Leu and T. Murata, "Properties and applications of the token distance matrix of a marked graph," in *Proc. 1984 IEEE Int. Symp. Circuits Syst.*, vol. 3, pp. 1381-1385, 1984.
- [222] —, "On maximum concurrency in a decision-free concurrent system," in *Proc. Int. Comput. Symp.*, 1984, Tamkang Univ., Taipei, Rep. of China, Dec. 12-14, 1984, vol. 2, pp. 1065-1071, 1985.
- [223] H. J. Genrich and K. Lautenbach, "Synchronisationsgraphen," *Acta Informatica* 2, pp. 143-161, 1973.
- [224] G. Memmi and G. Roucairol, "Linear algebra in net theory," in *Lecture Notes in Computer Science*, vol. 84, [15], pp. 213-223, 1980.
- [225] M. Silva and J. M. Colom, "On the computation of structural synchronic invariants in P/T nets," to appear in *Advances in Petri Nets 1988*, Lecture Notes in Computer Science, New York: Springer-Verlag, 1989.
- [226] J. Sifakis, "Structural properties of Petri nets," *Mathematical Foundations of Computer Science*, 1978, J. Winkowski, Ed. New York: Springer-Verlag, pp. 474-483, 1978.
- [227] H. J. Genrich and K. Lautenbach, "System modelling with high-level Petri nets," *Theoret. Comp. Sci.*, vol. 13, pp. 109-136, 1981.
- [228] P. Huber, A. M. Jensen, L. O. Jepsen, and K. Jensen, "Towards reachability trees for high-level Petri nets," *Theoret. Comput. Sci.*, vol. 45, pp. 261-292, 1986.
- [229] K. Jensen, "Coloured Petri nets and the invariant-method," *Theoret. Comput. Sci.*, vol. 14, pp. 317-336, 1981.

- [230] —, "How to find invariants for coloured Petri nets," in *Lecture Notes in Computer Science*, vol. 118, Math. Found. of Computer Science, 1981, 10th Symp. New York: Springer-Verlag, pp. 327–338, 1981.
- [231] K. Lautenbach and A. Pagnoni, "Invariance and duality in predicate/transition nets and in coloured nets," *Gesellschaft für Math. und Datenverarbeitung mbH Bonn, Arbeitspapiere der GMD Nr. 132*, Feb. 1985.
- [232] H. Mizuba, J. Herath, K. Ueda, and N. Saito, "Predicate/transition net simulation based on concurrent PROLOG," in *Software Science and Eng.*, (Proc. Symp. Kyoto 1984, RIMS Kokyuroku 547), pp. 23–34, 1985.
- [233] Y. Narahari and N. Viswanadham, "On the invariants of coloured Petri nets," in *Lecture Notes in Computer Science*, vol. 222 [23], pp. 330–345, 1986.
- [234] J. L. Peterson, "A note on colored Petri nets," *Inform. Process. Lett.*, vol. 11, no. 1, pp. 40–43, Aug. 1980.
- [235] M. Silva, J. Martinez, P. Ladet, and H. Alla, "Generalized inverses and the calculation of symbolic invariants for colored Petri nets," *TSI-Technique et Science Informatiques*, vol. 4, no. 1, G. Memmi, Ed., pp. 113–126, 1985.
- [236] J. Vautherin, "Non-linear invariants for coloured Petri nets with interdependent token; Application to the proof of parallel programs," in *Lecture Notes in Computer Science*, vol. 222 [23], pp. 418–434, 1986.
- [237] C. R. Zervos and K. B. Irani, "Colored Petri nets: Their properties and applications," Ann Arbor, Michigan, Michigan University, Systems Eng. Lab., Rep. RADC-TR-77-246, Aug. 1977.
- [238] A. Zenie, "Colored stochastic Petri nets," in *Proc. Int. Workshop Timed Petri Nets*, Torino, Italy, July 1–3, 1985, pp. 262–271, 1985.
- [239] R. David and H. Alla, "Continuous Petri nets," in *Proc. 8th European Workshop Application and Theory of Petri Nets*, pp. 275–294, Zaragoza, Spain, 1987.
- [240] A. Finkel and G. Memmi, "An introduction to FIFO nets—Monogeneous nets: A subclass of FIFO nets," *Theoret. Comput. Sci.*, vol. 35, pp. 191–214, 1985.
- [241] H. Fuss, "Numerical simulations with place/transition nets," in *Concurrency and Nets* [25], pp. 187–199, 1987.
- [242] R. Valk, "Self-modifying nets, a natural extension of Petri nets," in *Lecture Notes in Computer Science*, Vol. 62, Automata, Languages and Programming, G. Ausiello, C. Böhm, Eds., Berlin, New York: Springer-Verlag, pp. 464–476, 1978.
- [243] —, "Generalizations of Petri nets," in *Lecture Notes in Computer Science*, Vol. 118, Math. Found. of Computer Science, 1981, 10th Symp. New York: Springer-Verlag, pp. 140–155, 1981.
- [244] S. Porat and M. Yoeli, "Towards a hierarchy of nets," *J. Comput. Syst. Sci.*, vol. 29, no. 2, pp. 198–206, Oct. 1984.
- [245] Y. S. Kwong, "On reduction of asynchronous systems," *Theoret. Comput. Sci.*, vol. 5, pp. 25–50, 1977.
- [246] R. Valette, "Analysis of Petri nets by stepwise refinements," *J. Comput. Syst. Sci.*, vol. 18, pp. 35–46, 1979.
- [247] H. W. Kuhn and A. W. Tucker, Eds. *Linear Inequalities and Related Systems*, Annals of Mathematics Study, No. 38, Princeton, NJ: Princeton Univ. Press, 1956.
- [248] W. Reisig, "Petri nets with individual tokens," *Informatik-Fachberichte 66* [21], pp. 229–249, 1983.
- [249] T. C. Hu, *Integer Programming and Network Flows*. Reading, MA: Addison Wesley Pub. Co. 1970.
- [250] Y. E. Lien, "Termination properties of generalized Petri nets," *SIAM J. Computing*, vol. 5, no. 2, pp. 251–265, June 1976.
- [251] —, "A note on transition systems," *J. Inform. Sci.*, vol. 10, no. 4, pp. 347–362, 1976.
- [252] J. Martinez and M. Silva, "A simple and fast algorithm to obtain all invariants of a generalized Petri net," *Informatik-Fachberichte 52*, Application and Theory of Petri Nets. New York: Springer-Verlag, pp. 301–310, 1982.
- [253] H. J. Genrich, "Predicate/transition nets," in *LNCS*, vol. 254 [16], pp. 207–247.
- [254] K. Jensen, "Coloured Petri nets," in *LNCS*, vol. 254 [16], pp. 248–299.
- [255] G. Roucairol, "FIFO-Nets," in *LNCS*, vol. 254 [16], pp. 436–459.
- [256] C. Andre, "Use of the behaviour equivalence in place-transition net analysis," *Informatik-Fachberichte 52* [20], pp. 241–250, 1982.

Additional Bibliography

- [257] T. Araki and T. Kasami, "Some undecidable problems for Petri nets," *Syst. Comput. Contr.*, vol. 7, no. 1, pp. 20–28, Jan.–Feb., 1976.
- [258] —, "Decidable problems on the strong connectivity of Petri net reachability sets," *Theoret. Comput. Sci.*, vol. 4, no. 1, pp. 99–119, Feb. 1977.
- [259] M. Auguin, F. Boeri, and C. Andre, "Systematic method of realization of interpreted Petri nets," *Digital Processes*, vol. 6, pp. 55–68, 1980.
- [260] H. H. Ammar and R. W. Liu, "Analysis of the generalized stochastic Petri nets by state aggregation," in [28], pp. 88–95, 1985.
- [261] J. L. Baer, "A survey of some theoretical aspects of multi-processing," *ACM Computing Surveys*, vol. 5, no. 1, pp. 31–80, 1973.
- [262] E. Best and H. A. Schmid, "Systems of open paths in Petri nets," in *Lecture Notes in Computer Science*, vol. 32. New York: Springer-Verlag, pp. 186–193, 1975.
- [263] A. Datta and S. Ghosh, "Synthesis of a class of deadlock-free Petri nets," *J. Assoc. Comput. Machinery*, vol. 31, no. 3, pp. 486–506, July 1984.
- [264] F. De Cindio, G. De Michelis, L. Pomello and C. Simone, "Superposed automata nets," *Informatik-Fachberichte 52* [20]. New York: Springer-Verlag, pp. 269–279, 1982.
- [265] J. Grabowski, "The decidability of persistence for vector addition systems," *Inform. Process. Lett.*, vol. 11, no. 1, pp. 20–23, Aug. 1980.
- [266] C. Girault, C. Chatelain, and S. Haddad, "Specification and properties of a cache coherence protocol model," in *LNCS*, vol. 266 [24] pp. 1–20, 1987.
- [267] M. Hack, "The recursive equivalence of the reachability problem and the liveness problem for Petri nets and vector addition systems," in *Proc. 15th Ann. Symp. Switching Automata Theory*, pp. 156–164, 1974.
- [268] G. S. Hura and J. W. Atwood, "Program verification for microprocessors through Petri net modeling," *Microelectron. Reliab.*, vol. 25, no. 5, pp. 1001–1010, 1985.
- [269] J. L. Johnson and T. Murata, "Structure matrices for Petri nets and their applications," *J. Franklin Inst.*, vol. 319, no. 3, pp. 299–309, March 1985.
- [270] A. A. Khan, G. S. Hura, H. Singh, and N. K. Nanda, "On the determination of the solution of a class of Murata's state equation of Petri nets," *Proc. IEEE*, vol. 69, no. 4, pp. 466–467, April 1981.
- [271] F. Krückeberg and M. Jaxy, "Mathematical methods for calculating invariants in Petri nets," in *LNCS*, vol. 266 [24], pp. 104–131, 1987.
- [272] C. L. Lagen and T. Murata, "Use of prime numbers for computer-aided analysis of colored Petri nets," in *Proc. 26th Midwest Symp. Circuits Syst.*, Instituto Nacional de Astrofisica, Optica y Electronica (INAOE), Puebla, Mexico, August 15–16, 1983, E. S. Sinencio, Ed., pp. 124–128, 1983.
- [273] J. Van Leeuwen, "A partial solution to the reachability-problem for vector addition systems," in *Proc. 6th Annual ACM Symp. Theory Computing*, pp. 303–309, 1974.
- [274] T. Y. Lin, "Petri net models for computer security," *Bull. Inst. Math. Acad. Sinica*, vol. 11, no. 3, pp. 439–457, 1983.
- [275] R. J. Lipton, "The reachability problem requires exponential space," New Haven, CT, Yale University, Department of Computer Science, Res. Rep. 62, Jan. 1976.
- [276] R. Liu, Q. Huang, C. Lin, H. Ammar, and Y. F. Huang, "Petri net application to functional fault diagnosis," in *Proc. 1986 IEEE Int. Symp. Circuits Sys.*, San Jose, Calif. May, 1986, pp. 1323–1327.
- [277] A. C. Liu and H. C. Lin, "Modeling nuclear power plant by Petri nets," in *Computers in Engrg. 1986* (AMSE Pressure Vessel and Piping Conf., Chicago), pp. 151–156, 1986.
- [278] T. Murata and T. Shah, "On liveness, deadlock, and reachability of E-nets," in *Proc. 14th Allerton Conf. Circuit Syst. Theory*, Oct. 1976, pp. 597–605, 1976.
- [279] T. Murata, "State equation for E-net interpreted marked

- graphs," in *Proc. 19th Midwest Symp. Circuits Syst.*, pp. 152-157, 1976.
- [280] —, "Petri nets, marked graphs, and circuit-system theory: A recent CAS application," *Circuits and Systems* 11, no. 3, pp. 2-12, June 1977.
- [281] —, "Petri nets," in *Systems & Control Encyclopedia: Theory, Technology, Applications*, M. G. Singh (Editor-in-Chief), pp. 3665-3670. New York: Pergamon Press, 1987.
- [282] —, "Modeling and analysis of concurrent systems," Ch. 3 in *Handbook of Software Engineering*, C. R. Vick and C. V. Ramamoorthy Eds. New York: Van Nostrand, Reinhold, pp. 39-63, 1984.
- [283] R. A. Nelson, L. M. Haibt, and P. B. Sheridan, "Casting Petri nets into programs," *IEEE Trans. Software Eng.*, vol. SE-9, no. 5, pp. 590-602, Sept. 1983.
- [284] C. A. Petri, "Concepts of net theory," *Mathematical Foundations on Computer Science: in Proc. Symp. and Summer School*, High Tatras, Sept. 3-8, 1973, Math. Inst. of the Slovak Acad. of Sciences, pp. 137-146, 1973.
- [285] —, "Communication disciplines," in *Computing System Design: Proc. Joint IBM-University of Newcastle-upon-Tyne Seminar*, Sept. 1976, B. Shaw, Ed., pp. 171-183, 1977.
- [286] —, "Modelling as a communication discipline," in *Measuring, Modelling and Evaluating Computer Systems*, H. Beilner, E. Gelenbe, Eds. Amsterdam: North-Holland, pp. 435-449, 1977.
- [287] —, "Introduction to general net theory," in *Lecture Notes in Computer Science*, vol. 84, Net Theory and Applications. New York: Springer-Verlag, pp. 1-19, 1980.
- [288] —, "Concurrency," in *Lecture Notes in Computer Science*, vol. 84, Net Theory and Applications. New York: Springer-Verlag, pp. 251-260, 1980.
- [289] —, "State-transition structures in physics and in computation," *Int. J. Theoret. Physics*, vol. 21, no. 12, pp. 979-992, 1982.
- [290] L. Pomello, "Some equivalence notions for concurrent systems," in *Lecture Notes in Computer Science*, vol. 222, *Advances in Petri Nets 1985*, G. Rozenberg, Ed. New York: Springer-Verlag, pp. 381-400, 1986.
- [291] W. Reisig, "Deterministic buffer synchronization of sequential processes," *Acta Informatica* 18, pp. 117-134, 1982.
- [292] G. S. Sacerdote and R. L. Tenney, "The decidability of the reachability problem for vector addition systems," *Proc. 9th Ann. Symp. Theory Computing*, Boulder, Colorado, May 2-4, 1977, pp. 61-76, 1977.
- [293] H. A. Schmid and E. Best, "A step towards a solution of the liveness problem in Petri nets," University of Newcastle-upon-Tyne, Computing Laboratory, Tech. Rep. 114, Feb. 1978.
- [294] E. Schnieder, *Proze Informatik*. Braunschweig, Wiesbaden: Vieweg Verlag, 1986.
- [295] C. L. Seitz, "Graph representations for logical machines," Cambridge, MA: MIT, Department of Electrical Engineering, Ph.D. Thesis, Jan. 1971.
- [296] J. Sifakis, "A unified approach for studying the properties of transition systems," *Theoret. Comput. Sci.*, vol. 18, pp. 227-258, 1982.
- [297] P. H. Starke, *Petrinetze*. Berlin, DDR: Deutscher Verlag der Wissenschaften, 1980.
- [298] K. Tagahashi, K. Hasegawa, and Z. Banaszak, "Synthesis method for a Petri net with prescribed firing sequence," *Trans. Soc. Instrum. Control Eng. (Japan)*, vol. 21, no. 3, pp. 277-283, March 1985.
- [299] K. Tsuji, S. Kumagai, S. Kodama, and T. Yamada, "Modelling and verification of sequential control systems by Petri nets," in *Proc. IEEE Int. Symp. Circuits Syst.*, San Jose, vol. 3, pp. 988-991, 1986.
- [300] N. Viswanadham and Y. Narahari, "Coloured Petri net models for automated manufacturing systems," *Proc. 1987 IEEE Int. Conf. Robot. Automat.*, pp. 1985-1990, 1987.
- [301] G. Winskel, "Petri nets, morphisms and compositionality," in *Lecture Notes in Computer Science*, vol. 222. New York: Springer-Verlag, pp. 453-477, 1986.
- [302] H. Yamasaki, "Normal Petri nets," *Theoret. Comput. Sci.*, vol. 31, no. 3, pp. 307-315, June 1984.
- [303] S. H. Yu and T. Murata, "PT-marked graphs: a reduced model of Petri nets," in *Proc. 16th Ann. Allerton Conf. Commun., Cont. Comput.*, Oct. 1978, pp. 175-184, 1978.
- [304] Y. W. Han, "Performance evaluation of a digital system using a Petri net-like approach," in *Proc. Nat. Electron. Conf.*, vol. 32, Oct. 16-18, 1978, W. H. Tranter, Ed., pp. 166-172, 1978.
- [305] R. M. Shapiro and R. E. Millstein, "Failure recovery in a distributed database system," in *Proc. IEEE COMPCON Spring 1978*, pp. 66-70, 1978.
- [306] O. L. Bandman, "The correctness of asynchronous parallel-flow systems of data reduction," (in Russian), English translation in *Program. Comput. Software*, vol. 12, no. 1, pp. 9-17, 1986.
- [307] V. E. Kotov and L. A. Cherkasova, "From nets to logic and back in the specification of processes," *Concurrency and Nets*, Special volume in the series "Advances in Petri Nets." New York: Springer-Verlag, pp. 253-268, 1987.
- [308] H. Lee-Kwang, J. Favrel, and P. Baptiste, "Generalized Petri net reduction method," *IEEE Trans. Syst. Man, Cybernet.*, vol. SMC-17, no. 2, pp. 297-303, March/April 1987.
- [309] M. Silva, "Sur le concept de macroplace et son utilisation pour l'analyse des reseaux de Petri," *RAIRO-Automatique*, vol. 15, no. 4, pp. 57-67, 1981.
- [310] J. Martinez and M. Silva, "A language of description of concurrent systems modeled by colored Petri nets: Application to the control of flexible manufacturing systems," in *Languages for Automation*, S. K. Chang, Ed. New York: Plenum Press, pp. 369-388, 1985.
- [311] J. M. Colom, J. Martinez, and M. Silva, "Packages for validating discrete production systems modelled with Petri nets," in *Applied Modelling and Simulation of Technological Systems* (P. Borne and S. Tzafestas, Eds.), Amsterdam and New York: North-Holland, pp. 529-536, 1987.
- [312] J. A. Meldman and A. W. Holt, "Petri nets and legal systems," *Jurimetrics J.*, vol. 12, no. 2, pp. 65-75, 1971.
- [313] C. G. Looney, "Fuzzy Petri nets for rule-based decision-making," *IEEE Trans. Syst. Man, Cybernet.*, vol. 18, no. 1, pp. 178-183, Feb./Mar., 1988.
- [314] T. Ushio and R. Matsumoto, "State feedback and modular control synthesis in controlled Petri nets," *Proc. 27th IEEE Conf. Decision Contr.*, Austin, TX, pp. 1502-1507, Dec. 1988.
- [315] C. Lin and D. C. Marinescu, "Stochastic high-level Petri nets and applications," *IEEE Trans. Comput.*, vol. 37, no. 7, pp. 815-825, July 1988.



Tadao Murata (Fellow, IEEE) received the M.S. and Ph.D. degrees in electrical engineering from the University of Illinois at Urbana, in 1964 and 1966, respectively.

He is presently a Professor of Electrical Engineering and Computer Science at the University of Illinois at Chicago, where he initially joined in 1966. During occasional leaves of absence from the University of Illinois, he taught at the University of California at Berkeley and Tokai University, Tokyo, Japan, and was invited to visit Petri's Institute at GMD mbH in Germany and several other research institutes and universities in Europe. His current research interests include Petri net theory and its applications, especially to problems related to design, modeling and analysis of concurrent, parallel and/or distributed processing systems, VLSI systems, and logic programs applications. In these areas, he has published extensively and been awarded several National Science Foundation research grants since 1976. Prior to that, he worked in the area of circuits, systems and applied graph theory. He has served on the U.S. National Academy of Sciences/Computer Science and Technology Board panels.

Dr. Murata is an Editor for the IEEE TRANSACTIONS ON SOFTWARE ENGINEERING and served as the General Chairman of the 1987 International Workshop on Petri Nets and Performance Models. He is a member of the Association for Computing Machinery, EATCS, the Institute of Electronics and Communications Engineers of Japan, and the Information Processing Society of Japan. He is listed in *Who's Who in Engineering* and *Who's Who in America*.