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 1973, 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 Britain; 1987, Lectures given in this course were published in its proceedings [15], which is currently out of print.
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, M0. The underlying graph G 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. Arrows are labeled with 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 pth 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 |
|-----------------|-----------------|-----------------|
| Preconditions   | Transition       | Output Places   |
| Input Places    | Transition       | Output Places   |
| 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:

\[ P = \{ p_1, p_2, \ldots, p_n \} \] is a finite set of places,
\[ T = \{ t_1, t_2, \ldots, t_m \} \] is a finite set of transitions,
\[ F \subseteq (P \times T) \cup (T \times P) \] is a set of arcs (flow relation),
\[ W: F \rightarrow \{ 1, 2, 3, \ldots \} \] is a weight function,
\[ M_0: P \rightarrow \{ 0, 1, 2, 3, \ldots \} \] is the initial marking,
\[ P \cap T = \varnothing \] and \( P \cup T \not= \varnothing \).

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 an 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: \( 2\text{H}_2 + \text{O}_2 \rightarrow 2\text{H}_2\text{O} \). Two tokens in each input place in Fig. 1(a) show that two units of \( \text{H}_2 \) and \( \text{O}_2 \) 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.](image)

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 \( (p, t) \) or \( (t', p) \) 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 \( t_1 \) and \( t_2 \) 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 \( p \) and \( p' \) the same and equal to \( K(p_1) \), \( i = 1, 2, \) before and after firing \( t \). For example, since \( w(t_1, p_1) = 1 \), we have \( w(p_1, t_1) = 1 \). Similarly, \( w(t_2, p_2) = w(p_2, t_2) = 2 \), and \( w(p_1, t_2) = w(t_1, p_2) = 1 \), since firing \( t_1 \) removes two tokens from \( p_1 \) and adds one token in \( p_1 \) (we draw the two-weight arc from \( t_1 \) to \( p_1 \) and the unit-weight arc from \( p_1 \) to \( t_1 \)). Likewise, two additional arcs \( (t_2, p_2) \) and \( (t_2, p_2) \) 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.
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. 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 [77]. 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.

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 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
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.

It has been pointed out [172] that concurrency can be regarded as a binary relation (denoted by co on the set of events $A = \{e_1, e_2, \ldots \}$ which is 1) reflexive ($e_i \text{ co } e_i$) and 2) symmetric ($e_i \text{ co } e_j$ implies $e_j \text{ co } e_i$), 3) but not transitive ($e_1 \text{ co } e_2$ and $e_2 \text{ co } e_3$ do not necessarily imply $e_1 \text{ co } e_3$). For example, one may drive a car (event $e_1$) or walk (event $e_2$) while singing (event $e_3$), 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 $e_i$ and $e_j$ are in conflict if either $e_i$ or $e_j$ 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 $t_3$ 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 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 $t_2$ 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 $t_3$, $t_4$, or $t_{wait}$ fires if response 1, response 2, or no response is received before a specified time ($t_{wait}$), 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
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 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 \( p_3 \) (reading) if no token is in place \( p_5 \) 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 as buffer A has items (tokens), but B can consume only if buffer B 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 \( p_5 \) to \( t_2 \) 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
firing sequences 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. It is easy to see that λ (null symbol), abc, aabbcc, aabbbccc, · · · 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 \( L(M_0) = \{ a^n b^m c^n | n \geq 0 \} \) (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 representing processors executing in their private memory, and \( p_2 \) contains tokens representing free buses. Transition \( t_1 \) represents the issuing of access requests, and \( p_2 \) contains requests that have not yet been served. Tokens in \( p_3 \) represent processors having access to common memories. Tokens in \( p_4 \) represent processors requesting the same common memory that has been accessed by a token (processor) in \( p_2 \). Firing \( t_3 \) represents the end of the access to the memory for which \( p_4 \) is enabled.

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 \( M_0 \) is said to be reachable from a marking \( M_0 \) if there exists a sequence of firings that transforms \( M_0 \) to \( M_0 \). A firing or occurrence sequence is denoted by \( o = M_0 t_1 M_1 t_2 M_2 \cdots t_n M_n \) or simply \( o = t_1 t_2 \cdots t_n \). In this case, \( M_n \) is reachable from \( M_0 \) by \( o \) and we write \( M_0 \circ M_n \). 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 \( M_0 \) 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 \( M_0 \) is any marking whose restriction to a given subset of places agrees with that of a given marking \( M_0 \). 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., $M_0(p) \leq k$ 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 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 it 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 it is L1-live for every marking $M$ in $R(M_0)$.

A Petri net $(N, M_0)$ is said to be L$k$-live if every transition in the net is L$k$-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 L$k$-live if it is L$k$-live but not L$(k + 1)$-live, $k = 1, 2, 3$.

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

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 $t_0$, $t_1$, $t_2$, and $t_3$. The transitions $t_0$, $t_4$, $t_5$, and $t_6$ in Fig. 16 are L0-live (dead), L1-live, L2-live, and L3-live, respectively, all strictly.

D. Reversibility and Home State

A Petri net $(N, M_0)$ is said to be reversible if, for each marking $M$ in $R(M_0)$, $M$ 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 $\overline{B}$, $\overline{L}$, and $\overline{R}$ 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 $M'(p) \leq M(p)$ 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 schemes [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.
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

$$d_{t_1 t_2} = \max_s |\tilde{\sigma}(t_1) - \tilde{\sigma}(t_2)|$$

(1)

where $\sigma$ is a firing sequence starting at any marking $M$ in $R(M_0)$ and $\tilde{\sigma}(t_i)$ is the number of times that transition $t_i$, $i = 1, 2$ fires in $\sigma$. For example, in the net shown in Fig. 17(d) $d_{t_1 t_2}$
= 1, \( d_{10} = 1, d_{13} = \infty \), 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 \( t_2 \) there is a firing sequence \( s = t_2 t_1 t_1 \) in which \( \delta(s) = 2 \) and \( \delta(t_1) = 0 \).

The synchronous 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 synchronous 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 \( s \) is said to be unconditionally (globally) fair if it is finite or every transition in the net appears infinitely often in \( s \). A Petri net \( (N, M_0) \) is said to be an unconditionally fair net if every firing sequence \( s \) 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(b) 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 \( t_2 \) and \( t_4 \) will not appear in an infinite firing sequence \( s = t_2 t_4 t_2 t_4 \cdot \cdot \cdot \). 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 \( t_4 \) 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 specific 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 \(\omega\), which can be thought of as “infinity.” It has the properties that for each integer \(n\), \(\omega > n\), \(\omega = n\), 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 \(M_0\), set its parent 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** If 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 \(\omega\) for each place \(p\) 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 \(t_2\) 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 \(t_2\) at \(M_1\) results in \(M_2 = (1 \ 1 \ 0)\), which covers \(M_0 = (1 \ 0 \ 0)\). Therefore, the new marking is \(M_0 = (1 \ 0 \ 0)\), where two transitions \(t_1\) and \(t_2\) are again enabled. Firing \(t_1\) transforms \(M_2\) to \(M_3 = (0 \ 0 \ 1)\), from which \(t_2\) can be fired, resulting in an “old” node \(M_3 = M_0\). Firing \(t_2\) at \(M_3\) results in an “old” node \(M_4 = M_0\). Thus, we have the coverability tree shown in Fig. 18(a).

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 if (and only if) \(\omega\) does not appear in any node labels in \(T\).
2. A net \((N, M_0)\) is safe if only 0's and 1's appear in node labels in \(T\).
3. A transition \(t\) is dead if 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 it is an exhaustive method. However, in general, because of the information lost by the use of the symbol \(\omega\) (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.

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

tinct labeled nodes in the coverability tree, and the arc set $E$ is the set of arcs labeled with single transition $t_j$ representing all possible single transition firings such that $M[t_j] > M_j$, where $M_j$ and $M_i$ 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 nondeterministic 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

$$a_{ij} = a_{ij}^- - a_{ij}^+$$  \hspace{1cm} (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
arc to transition \( j \) 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 \( a_{ii}^+, a_{ii}^-, a_{ii}^0 \), 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

\[
a_{ii}^0 \leq M(j), \quad j = 1, 2, \ldots, m.
\]  

**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 \( j \)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 \( j \)th row of the incidence matrix \( A \) denotes the change of the marking as the result of firing transition \( j \), we can write the following state equation for a Petri net [198]:

\[
M_k = M_{k-1} + A^Tu_k \quad k = 1, 2, \ldots.
\]  

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

\[
M_d = M_0 + A^T \sum_{k=1}^{d} u_k
\]

which can be rewritten as

\[
A^T x = \Delta M
\]

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 [195] that a set of linear algebraic equations (6) has a solution \( x \) iff \( \Delta M \) is orthogonal to every solution \( y \) of its homogeneous system,

\[
Ay = 0.
\]

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

\[
A = \begin{bmatrix}
A_{11} & A_{12} \\
A_{21} & A_{22}
\end{bmatrix}
\]

where \( A_{22} \) is a nonsingular square matrix of order \( n - r \). 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 \( B_l \):

\[
B_l = [I_{(m - r)} A_{22}^{-1}]
\]

where \( I_{(m - r)} \) is the identity matrix of order \( m - r \). Note that \( AB_l^T = 0 \). That is, the vector space spanned by the row vectors of \( A \) is orthogonal to the vector space spanned by the row vectors of \( B_l \). The matrix \( B_l \) corresponds to the fundamental circuit matrix [13] in the case of a marked graph.

Now, the condition that \( \Delta M \) is orthogonal to every solution for \( Ay = 0 \) is equivalent to the following condition:

\[
B_l \Delta M = 0.
\]

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 \( B_l \Delta M = 0 \), where \( \Delta M = M_d - M_0 \) and \( B_l \) is given by (9).

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 the difference is a linear combination of the row vectors of \( B_l \), that is,

\[
\Delta M = B_l^T z
\]

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

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

**Example 5:** For the Petri net shown in Fig. 21, the state equation (4) is illustrated below, where the transition \( t_i \) fires to result in the marking \( M_1 = (3 \ 0 \ 0 \ 2)^T \) from \( M_0 = (2 \ 0 \ 1 \ 0)^T \):

\[
\begin{bmatrix}
3 & 2 & -2 & 1 & 1 \\
0 & 0 & 1 & -1 & 0 \\
0 & 1 & 1 & 0 & -1 \\
2 & 0 & 0 & -2 & 2
\end{bmatrix}
\begin{bmatrix}
0 \\
0 \\
0 \\
1
\end{bmatrix}
\]

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

\[
A_{11} = \begin{bmatrix}
-2 & 1 \\
1 & -1
\end{bmatrix}
\quad \text{and} \quad A_{22} = \begin{bmatrix}
1 & 0 \\
0 & -2
\end{bmatrix}.
\]

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

\[
B_l = \begin{bmatrix}
1 & 0 & 2 & 1/2 \\
0 & 1 & -1 & -1/2
\end{bmatrix}
\]

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

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

\[
A^T x = 0
\]

is called a \( T \)-invariant, and an integer solution \( y \) of the transposed homogeneous equation \( Ay = 0 \) is called an \( S \)-in-
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 VIII-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).

Example 7: 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 \( t \)

\[ \bullet \tau = \{ p | (t, p) \in F \} \] = the set of output places of \( t \)

\[ \bullet p = \{ t | (t, p) \in F \} \] = the set of input transitions of \( p \)

\[ p^* = \{ t | (p, t) \in F \} \] = 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 \( S_t \subseteq P \), then \( \bullet S_t \) is the union of all \( \bullet p \) such that \( p \in S_t \). 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 \( \bullet p = p^* = \emptyset \) or \( t = \bullet t = \emptyset \).

\[ \begin{array}{c}
\text{Fig. 22. Six transformations preserving liveness, safety, and boundedness.}
\end{array} \]
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 has exactly one input place and exactly one output place, i.e.,

$$|\bullet| = |\bullet^*| = 1$$

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.,

$$|\bullet| = |\bullet^*| = 1$$

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.,

for all $p \in P$, $|\bullet| \leq 1$ or $\bullet^* = \{p\}$; equivalently,

for all $p_1, p_2 \in P$, $p_1 \cap p_2 = \emptyset$ => $|\bullet| = |\bullet^*| = 1$.

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

$$p_1 \cap p_2 = \emptyset$$

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

$$p_1 \cap p_2 = \emptyset$$

for all $p_1, p_2 \in P$.

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 $p_1 = \{t_1, t_2\}$ and $p_2 = \{t_3, t_4\}$. Extended free-choice nets (EFC) allow the structure shown in Fig. 25(c) but not the one shown in Fig. 25(d), where $p_1 = \{t_1\}$ and $p_2 = \{t_4, t_5\}$. 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 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 $p_1 = \{t_1, t_2\}$ and $p_2 = \{t_3, t_4\}$. 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 conflict (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. 25. Key structures characterizing subclasses of Petri nets and their Venn diagram, where MG, SM, etc., denote nonMG, nonSM, etc.

Fig. 26. Transformation of EFC structure to FC structure.
in Fig. 17(a) is not an AC because \( p_1 \cap p_2 \neq \emptyset \), \( p_1 = \{ t_1, t_2 \}, p_2 = \{ t_1, t_4 \} \), \( p_1 \cap p_2 \neq \emptyset \), 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 \( p_1 = \{ t_1 \} \subseteq p_2 = \{ t_1, t_2 \} \) in Fig. 17(d) \( p_1 = \{ t_1 \} \cap p_2 = \{ t_1, t_2, t_4 \} \) and \( p_1 = \{ t_1 \} \subseteq p_2 = \{ t_1, t_2 \} \) in Fig. 17(g). The net in Fig. 17(h) is not an AC because \( p_1 = \{ t_1, t_2 \} \) and \( p_1 = \{ t_1, t_4 \} \). The net in Fig. 17(i) is both an MG and an SM. The net in Fig. 17(e) is an MG.

8. 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 the 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 \( p^* = \emptyset \) (\( t^* = \emptyset \)). It is not difficult to see the following theorem [208] from Table 3.

<table>
<thead>
<tr>
<th>Case</th>
<th>If such as then</th>
<th>t is not live.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \bullet p = \emptyset ) (source place)</td>
<td>( p ) ( \rightarrow ) ( t )</td>
</tr>
<tr>
<td>2</td>
<td>( \bullet p = \emptyset ) (sink place)</td>
<td>( t ) ( \rightarrow ) ( p )</td>
</tr>
<tr>
<td>3</td>
<td>( \bullet t = \emptyset ) (source transition)</td>
<td>( p ) ( \rightarrow ) ( t )</td>
</tr>
<tr>
<td>4</td>
<td>( \bullet t = \emptyset ) (sink transition)</td>
<td>( t ) ( \rightarrow ) ( p )</td>
</tr>
</tbody>
</table>

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 \( x \in P \cup T \), \( x^* \neq \emptyset \). 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 Figs. 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 nonzero safe markings.

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

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

Theorem 6: 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 Table 3 can be redrawn as the marked 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 7: For a marked graph, the token count in a directed circuit is invariant under any firing, i.e., \( M(C) = M(D) \) for each directed circuit \( C \) and for any \( M \) in \( R(M_0) \), where \( M(C) \) denotes the total number of tokens on \( C \).

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

The following consideration is helpful in understanding the above mini-max theorem. Consider all directed circuits \(C_1, C_2, \ldots, C_n\) 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 \(y\) as many times as possible without firing the node \(y\). It can be seen that \(\text{Min}(M_0(C_1), M_0(C_2), \ldots, M_0(C_n))\) is the maximum possible tokens that can be brought on the arc \(e\). In particular, if \(\text{Min}(M_0(C_1), M_0(C_2), \ldots, M_0(C_n)) = 1\), 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 if 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 \(G\) if \(G\) 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 \(e\) and marking \(M_0\) (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 \(M_0\).

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 if 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 if 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 to a directed circuit with token count one. In fact, the firing sequence \(e = (1 \rightarrow 3 \rightarrow 4 \rightarrow 1)\) 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 \(S \subseteq \bullet S\), 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 \(\bullet Q \subseteq Q\), 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 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 (or trap) is called a basic siphon (basic 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 (or trap) is said to be minimal if it does not contain any other siphon (trap). A minimal siphon (or trap) is a basis siphon (or trap), but not all basis siphons (traps) are minimal.

**Example 8:** In the Petri net shown in Fig. 31, let \(S_1 = \{p_1, p_2, p_3\}, S_2 = \{p_1, p_2, p_3\}, S_3 = \{p_2, p_4\}, S_4 = \{p_2, p_3\}\), and \(S_5 = \{p_2, p_3, p_4\}\). Then, we have \(S_5 \subseteq S_4 \subseteq S_3 \subseteq S_2 \subseteq S_1\). Thus, \(S_1\) is a siphon. Since \(S_2 \subseteq S_1\), \(S_2\) is a trap. Similarly, it is easy to verify that \(S_3\) is a siphon, \(S_4\) is 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, and \(S_1\) and \(S_2\) 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

\[ p_1 \Rightarrow p_2 \quad \text{if } p_1 \text{ is in a siphon } S, \text{ then } p_2 \text{ is in } S \]

\[ p_2 \Rightarrow p_1 \lor p_4 \quad \text{if } p_2 \text{ is in } S, \text{ then } p_1 \text{ or } p_4 \text{ is in } S \]

\[ p_3 \Rightarrow p_1 \land p_2 \quad \text{if } p_3 \text{ is in } S, \text{ then } p_1 \text{ and } p_2 \text{ are in } S \]

\[ p_4 \Rightarrow p_1 \quad \text{if } p_4 \text{ is in } S, \text{ then } p_1 \text{ is in } S. \]

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

\[ \neg p_1 \lor p_2, \neg p_2 \lor p_1 \lor p_4, \neg p_3 \lor p_1, \]

\[ \neg p_1 \lor p_3, \neg p_3 \lor p_4, \neg p_4 \lor p_1. \]

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 \( p_i \notin S \):

\[
\begin{align*}
-p_1 + p_2 & \geq 0 \\
-p_2 + p_1 + p_4 & \geq 0 \\
-p_3 + p_1 & \geq 0 \\
-p_1 + p_3 & \geq 0 \\
-p_3 + p_4 & \geq 0 \\
-p_4 + p_1 & \geq 0 
\end{align*}
\]

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.

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 if every siphon in \( N \) contains a marked trap.

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

**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 \( M \in R(M) \) such that each component \((N_i, M_i)\) is a live and safe MG, where \( M_i \) is \( M \) restricted to \( N_i \).

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

In Theorem 13 (Theorem 14), an SM-component (MG-component) \( N_i \) of a net \( N \) is defined as a subnet generated by places (transitions) in \( N_i \) having the following two properties: i) each transition (place) in \( N_i \) has at most one incoming 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 \( \{ p_2, p_3, p_4 \} \) contains no traps (thus no marked traps). The AC net shown in Fig. 32 is live since the minimal siphon \( \{ p_1, p_2, p_4 \} \) contains a marked trap \( \{ p_1, p_4, p_3 \} \) and the siphon \( \{ p_1, p_2, p_3, p_4 \} \) contains marked traps \( \{ p_1, p_3 \} \) and \( \{ p_1, p_2, p_4 \} \). 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

![Diagram](image-url)

**Fig. 32.** A live AC net.

![Diagram](image-url)

**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).
It is known that an asymmetric choice net \((N, M_0)\) is live iff it is place-live, i.e., for each \(M_a \in R(M_0)\) and for each place \(p\) in \(N\), there exists a marking \(M\) in \(R(M_0)\) 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], and [211], [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

\[
M_f = M_0 + A^T x
\]

is a necessary condition for \(M_f\) 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 \(N_r\) 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. \(M_{n0}\) denotes the subvector of \(M_0\) for places in \(N_r\).

**Theorem 16:** In an acyclic Petri net, \(M_f\) 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 \((N_r, M_{n0})\), which is acyclic. There is at least one transition \(t\) that is firable at \(M_{n0}\). 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 \(M_f \geq 0\). Now, fire \(t\).
Let the resulting marking be $M' = M_0 + A' u_i, x' = x - u_i$. Then, $M_0 = M' + A' x', x' \geq 0$, and the subnet $(N_{x'}, M_{x'})$ is acyclic. Repeat the above process until $x'$ reduces to a zero vector.

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 MGSs (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, $M_0$ is reachable from $M_0$ if there exists a nonnegative integer solution $x$ such that $i \Rightarrow (13)$ holds and $ii) (N_{x'}, M_{x'})$ has no token-free siphons.

Proof: $(\Rightarrow i)$ is obvious. $ii)$ If $(N_{x'}, M_{x'})$ has a token-free siphon $S$, then it is not possible to fire any transition $t \in S^+$. This contradicts the fact that every transition in $N_x$ fires in a firing sequence transforming $M_0$ into $M_0$. $(\Leftarrow)$ Since there are no token-free siphons in $(N_{x'}, M_{x'})$, there is at least one transition $t$ fireable at $M_0$. Fire $t$ and let $M' = M_0 + A' u_i, x' = x - u_i$. Then $M_0 = M' + A' x', x' \geq 0$. 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 $M_0 \geq 0$. 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 $t \not\in S$ 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 tokens 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, $t \not\in S^+, t \not\in S$. 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 $S^+$ are fireable. By back-tracing token-free input places of nonfirable transitions in $S^+$, we can find a token-free siphon $S' \subseteq S$ such that for each $p \in S \cap S'$, $p \not\in S'$. This means that $S$ was a token-free siphon in $(N_{x'}, M_{x'})$ 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. 

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 subnet of places and a trap (siphon) in $N^{-1}$ is a siphon (traps) 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, $M_0$ is reachable from $M_0$ if there exists a nonnegative integer solution $x$ such that $i) \Rightarrow (13)$ holds and $ii) (N_{x'}, M_{x'})$ has no token-free siphons where $M_0$ is the subvector of $M_0$ restricted to places in $N_x$. 

**Example 10:** Consider the (non-asymmetric choice) net shown in Fig. 38(a). There are two directed circuits $p_1, p_2, p_3, p_1, p_1, p_1, p_1, p_1,$ and $p_1, p_1, p_2, p_3, p_1$. The set of places in each of these directed circuits is a trap $\{p_1, p_2, p_3\}$. Thus, this is a TC net. Suppose $M_0 = (1, 0, 0)^T$. Then it is easy to verify that (13) holds. A solution $x = (1, 0, 1)$.

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

$p_1, p_1, p_1, p_1, p_1, p_1, p_1, p_1, p_1, p_1$. The set of places in each of these directed circuits is a trap $\{p_1, p_2, p_3\}$. Thus, this is a TC net. Suppose $M_0 = (1, 0, 0)^T$. Then it is easy to verify that (13) holds. A solution $x = (1, 0, 1)$.

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

$t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}$. The set of places in each of these directed circuits is a trap $\{p_1, p_2, p_3\}$. Thus, this is a TC net. Suppose $M_0 = (1, 0, 0)^T$. Then it is easy to verify that (13) holds. A solution $x = (1, 0, 1)$.

**Fig. 38. Illustration of Theorem 17.** (a) A given TC net $(N_x, M_0)$ and (b) the subnet $(N_{x'}, M_{x'})$.
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 $B$ 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.,

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

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

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

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

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

or

$$B_f M_0 = B_f M_d.$$  (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 $M_d$ to reach another marking $M_d$. In other words, we have the following theorem.

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

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 $(G, M_0, M_d)$, $M_d$ 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 $(G, M_d)$.

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

![Example 11 Diagram](image)

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

$$B_f = \begin{bmatrix} 1 & 0 & 0 & -1 & 1 & 1 \\ 0 & 1 & 0 & 1 & -1 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

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

PROCEEDINGS OF THE IEEE, VOL. 77, NO. 4, APRIL 1989
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 \( \{d, e, f\} \)):

\[
\begin{align*}
A'x &= \begin{bmatrix} -1 & 1 & 0 & 0 \end{bmatrix} \Rightarrow x_1 = 0, x_2 = 2, x_3 = 1.
\end{align*}
\]

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 \( A'x = \Delta M \). Note that \( x = (k, k + 2, \ldots, k + m)^T \) 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.

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

\[
A'x = 0.
\]

For a connected marked graph with \( n \) nodes, the rank of \( A \) is \( n - 1 \) and (18) has only one independent solution \( x = (k, k + 2, \ldots, k + m)^T \). 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 \( M_0 \) if it fires every node an equal number of times.

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 \( M_a \) and \( M_b \) 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 if the underlying graph of \( G \) is a tree.

(The two vectors \( x \) and \( \Delta M \) 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 \( \Delta M \) can be chosen arbitrarily ifv the network is a tree.)

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

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

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 \( M_a \sim M_b \) if \( M_b \) is reachable from \( M_a \). 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

\[
\rho(N_k) = k - 1
\]

and

\[
\rho(K_n) = (n - 1)!.
\]

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

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

**Theorem 24:** For a strongly connected graph \( G \), \( \rho(G) = 1 \) if 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 \((x, z, y)\).

**Theorem 25:** For a strongly connected graph \( G \), \( \rho(G) = 1 \) if there is a marking of \( G \) which places exactly one token on every directed circuit in \( G \).
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 \( (G, M_6) \) shown in Fig. 44, it can be verified that \( \rho(G) = 1 \) since \( M_6 \) places exactly one token on each of the following possible directed circuits in \( G \): \{a, b, c\}, \{b, d, e, f\}, \{f, g, h\}, \{a, b, d, i\}, \{b, d, i, h, f\}, \{c, h, f, b\}, and \{h, f, g, i\} (Theorem 25).

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

Fig. 44. A marked graph \( (G, M_6) \) with \( \rho(G) = 1 \).

2) Expansion Rules for LSGM Synthesis:
It has been shown [202], [203] that \( \rho(G) \) is invariant under the following operations on a digraph \( G \).

a) Series Expansion (SE)—addition of an arc \( e \) and node \( x \) in series with an existing arc \( e' \) (see Fig. 45(a)).
b) Parallel Expansion (PE)—addition of an arc \( e \) in parallel with an existing arc \( e' \) (see Fig. 45(b)).
c) 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)).
d) 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 \( (\alpha, \beta) \) is replaced by \( (\beta, \alpha) \) everywhere) (see Fig. 45(d)).
e) 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 if \( \rho(G_1) = \rho(G_2) = 1 \).

The above stepwise expansion operations can be used for synthesizing LSGMs \( (G, M_6) \). 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
we know that $\rho(G_0) = 1$ for the marked graph $(G_0, M_0)$ 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 $(G_0, M_0)$ 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, $\rho(G_0) = \rho(N_2) = 1$.

C. Weighted Sum of Tokens

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

$$AI = 0$$

(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)

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

(22)

or

$$M^T I = M_0^T I.$$  

(23)

Equation (23) states that the weighted sum of tokens $\sum_{e \in S} (M(e)) \rho(e)$ 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

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

(24)

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

(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:

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

(26)
where
\[ C = \begin{bmatrix} W \\ 0 \end{bmatrix}, z = \begin{bmatrix} M \\ x \end{bmatrix}, D = \left[ I_m - A \right]. \]

and \( I_m \) is the identity matrix of order \( m \). The dual problem of (26) can be stated as
\[ \min y = M^T x \text{ subject to } D^T y \geq C \] (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 \( |A| = 0 \) since the sum of the \( n \) rows of \( A \) is always zero, i.e., \( \begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} A = 0 \). 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 (23) can be proved similarly. \( \square \)

Example 15: Consider the marked graph shown in Fig. 48, where \( M_0 = \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} \) and \( W = \begin{bmatrix} 2 & 1 & 2 & 1 \end{bmatrix} \).

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.,
\[ t_{ij} = \begin{cases} \min M_0(P_{ij}), \\ \infty, \text{if no directed path } P_{ij} \text{ exists.} \end{cases} \] (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 \( i = 1, 2, \ldots, n \).
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 \( \infty \).

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 \( i = 1, 2, \ldots, n \) (\( i \neq j \)), 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 \( t_{ij} > 0 \) for \( k = 1, 2, \ldots, n \) (\( k \neq i \)), 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 \( k \)-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
\[
T = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 \\ 2 & 1 & 0 & 0 \end{bmatrix}
\]
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 synchronous 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 fireable 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,

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

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 whose corresponding columns from the distance matrix have only zero entries for their off-diagonal entries. 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 fireable nodes.

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

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:

$$\begin{align*}
\text{max} & \quad \sum_{i=1}^{n} x(i) \\
\text{subject to:} & \\
A^T x & \leq M \quad \text{(Firability)} \\
B_i M & = B_i M_0 \quad \text{(Reachability)} \\
\text{where} & \\
x(i) & = \begin{cases} 
1 & \text{if node } i \text{ is enabled} \\
0 & \text{otherwise.}
\end{cases}
\end{align*}
$$

**E. MG Synthesis of Synchronous Distance Matrix**

The synchronous distance matrix of a marked graph is an $n \times n$ symmetric matrix $D = [d_{ij}]$, where $d_{ij}$ is the synchronous 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 synchronous distance matrix of a marked graph $(G, M_0)$. Then

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

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

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

iv) $d_{ij} = d_{ji}$ for all $i, j$.

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

$$D + D^T = D^*$$

(29)

where $\ast$ 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 synchronous distance matrix, then (29) holds.

Given a matrix $D$, we are interested in the problem of finding a marked graph whose synchronous 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 $d_{\text{max}}$ in $D$. List all rows $i_0$ in which $d_{\text{max}}$ 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 $d_{\text{min}}$. List the column $j_0$ in which $d_{\text{min}}$ is located. Draw an arc between nodes $i_0$ and $j_0$ with arc weight $d_{\text{min}}$. (If $d_{\text{min}}$ is not unique, $D$ is not realizable as an $n$-node tree.)

Step 2.3. Delete all the rows and columns in which $d_{\text{max}}$ 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 \( e = (v_i, v_j) \) in the tree by a pair of oppositely directed arcs, \( e_1 = (v_i, v_j) \) and \( e_2 = (v_j, v_i) \). 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 \( e \).

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

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

The maximum entry \( d_{\text{max}} = 6 \) is found for the following rows

\[
\begin{array}{ccc}
1 & 1 & 1 \\
4 & 4 & 4 \\
7 & 7 & 7
\end{array}
\]

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 \( d_{\text{min}} \) is located at column \( j_0 = 2, 3, 6, 6 \), with \( d_{\text{min}} = 1, 3, 1, 1 \), 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

\[
\begin{bmatrix}
2 & 3 & 6 \\
2 & 0 & 2 \\
D_1 = 3 & 2 & 0 \\
6 & 4 & 2 \end{bmatrix}
\]

For \( D_1 \), \( d_{\text{max}} = 4 \) is found in rows \( i_0 = 2 \) and 6. \( d_{\text{min}} = 2 \) 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).

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], [238], [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-

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 \( y \), \( x \geq y \) means that \( x(i) \geq y(i) \) for each \( i \), and \( x \geq y \) means that \( x \geq y \) and \( x(i) = y(i) \) 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

\[
\text{Rank } A = m.
\]

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

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

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

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:** (\( \Rightarrow \)) Suppose

\[
y > 0, \quad Ay \leq 0. \quad (32)
\]

Let \( M \in \mathbb{R}(M_0) \). Then from (5), we have

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

Consider the inner product of \( M \) and \( y \)

\[
M^T y = M_0^T y + x^T Ay. \quad (34)
\]

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 \( y \).

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

<table>
<thead>
<tr>
<th>Case</th>
<th>System ( \alpha )</th>
<th>System ( \beta )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 (M-F)</td>
<td>( A^T x \geq b, x ) unrestricted</td>
<td>( Ay = 0, y \geq 0, y' b &gt; 0 )</td>
</tr>
<tr>
<td>2 (M-F)</td>
<td>( A^T x \geq b, x \geq 0 )</td>
<td>( Ay = 0, y \geq 0, y' b &gt; 0 )</td>
</tr>
<tr>
<td>3 (Stiemke)</td>
<td>( A^T x \geq 0, x \leq a ) unrestricted</td>
<td>( Ay = 0, y &gt; 0 ) (not conservative)</td>
</tr>
<tr>
<td>4 (Farkas)</td>
<td>( A^T x \geq 0, x \geq 0 )</td>
<td>( Ay = 0, y &gt; 0 ) (structurally bounded)</td>
</tr>
</tbody>
</table>

_DGRAM

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

A place \( p \) in a Petri net is said to be structurally unbounded if there exists a marking \( M_0 \) and a firing sequence \( \sigma \) 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 \( A^T x = \Delta M \geq 0 \), 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, \( M^T y = M_0^T y = \text{a constant} \), for every \( M \in \mathbb{R}(M_0) \) 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 \( \sigma \) from \( M_0 \) such that every (some) transition occurs infinitely often in \( \sigma \).

**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 so that a firing sequence \( \sigma \), such that \( \sigma = x \), can be repeated indefinitely. Then every transition will occur infinitely often in this firing sequence. The converse is also true.

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

**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 \( \sigma \), such that \( \sigma = x \), can be repeated.

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 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

<table>
<thead>
<tr>
<th>Case</th>
<th>If</th>
<th>Then</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N is structurally bounded and structurally live</td>
<td>N is both conservative and consistent.</td>
</tr>
<tr>
<td>2</td>
<td>≥ 0, AY ≥ 0</td>
<td>≥ 0, no live M₀ for N. N is not consistent.</td>
</tr>
<tr>
<td>3</td>
<td>≥ 0, AY ≥ 0</td>
<td>(N, M₀) is not bounded for a live M₀.</td>
</tr>
<tr>
<td>4</td>
<td>≥ 0, AᵀX ≥ 0</td>
<td>≥ 0, no live M₀ for structurally bounded N.</td>
</tr>
<tr>
<td>5</td>
<td>≥ 0, AᵀX ≥ 0</td>
<td>N is not structurally bounded.</td>
</tr>
</tbody>
</table>

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

![Diagram](image)

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

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

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>d</th>
<th>e</th>
<th>f</th>
<th>g</th>
<th>h</th>
<th>i</th>
<th>j</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structurally bounded</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>Conservative</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Partially conservative</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Repetitive</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Partially repetitive</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Consistent</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Partially consistent</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Completely controllable</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
<td>−</td>
</tr>
<tr>
<td>Structurally B-fair</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>

+ 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

\[
y = (1 1 0 0 1 1 1) ≥ 0\
\]

such that \( AY = (0 0 -1 0 0) \).

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 \( AY = 0 \) (\( AᵀX = 0 \)). The following two theorems are obvious from the preceding discussion.

**Theorem 33:** An m-vector \( y \) is an S-invariant iff \( M₀^ty = M₀^t \) for any fixed initial marking \( M₀ \) and any \( M \) of \( M₀ \).

**Theorem 34:** An n-vector \( x \) is a T-invariant iff there exists a marking \( M₀ \) and a firing sequence \( s \) from \( M₀ \) back to \( M₀ \) with its firing count vector \( s \) equal to \( x \).

The set of places (transitions) corresponding to nonzero entries in an S-invariant \( y \) (T-invariant \( x \)) 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 \( y₁ \) such that \( y₁(p) ≤ y(p) \) 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, \( x₁ = (1 0 1) \) and \( x₂ = (0 1 1) \) are all possible minimal-sup-

![Diagram](image)

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

port T-invariants, where \( \{ x₁, x₂ \} \) and \( \{ t₁, t₂ \} \) are corresponding minimal supports. All other T-invariants such as \( x₃ = (1 1 1) \) and \( x₄ = (1 0 1) \) can be expressed as linear combinations of \( x₁ \) and \( x₂ \). That is, \( x₃ = x₁ \) and \( x₄ = 2x₁ + x₂ \). Note that there are many (non-unique) T-invariants such as \( x₅, x₆, \) etc., corresponding to a nonminimal support \{t₁, t₂, t₃\}. (One easy way to find T-invariants in an example like this is to sumivate 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), \( x₁ = (1 1 1) \), \( x₂ = (2 0 1) \) and \( x₃ = (0 2 1) \) are
three minimal T-invariants. However, only \( x_1 \) and \( x_3 \) are minimal-support T-invariants. The support of \( x_1 \) is \( \{ t_1, t_2, t_3, t_4 \} \) is not minimal since its proper subset \( \{ t_1, t_2, t_4 \} \) is the support of another T-invariant \( x_3 \). 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 \( p_i \), \( i = 1, 2, 3, 4 \).

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,

\[
M(p) \leq \text{Min} (M'_y/\gamma_y(p)) \tag{37}
\]

where the minimum is taken over all nonnegative minimal-support S-invariant \( y \) such that \( y(p) \neq 0 \). It is shown in [179] that this upper bound cannot 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

\[
\begin{array}{cccccc}
  & p_1 & p_2 & p_3 & p_4 \\
 t_1 & -1 & 1 & -1 & 0 \\
t_2 & -1 & 0 & -k & 1 \\
t_3 & 1 & -1 & 1 & 0 \\
t_4 & 1 & 0 & k & -1 \\
\end{array}
\]

It is easy to verify the following.

1) \( A_y = 0 \) and \( A_z = 0 \) 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 \( M_6 = (k \ 0 \ k \ 0)^t \).

\[
M(p_4) \leq \text{Min} (M'_y/\gamma_y(p_4), M'_z/\gamma_z(p_4))
\]

\[
= \text{Min} (k/1, k/k) = 1.
\]

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.

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. An 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 \( M_i \) such that \( (N, M_i) \) is not a B-fair net as is shown in Fig. 54(b), where

![Diagram](image)

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 \( t_1, t_3, t_0 \) can be repeated infinitely often without firing \( t_2, t_4 \) or \( t_0 \).

3) A structurally bounded net is structurally B-fair iff either a) it is consistent and there is only one reproduction vector (minimum 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 \( r \) is defined as the time to complete a firing sequence leading back to the starting marking after firing each transition at least once. \( r \) is called a cycle time. Thus, it is assumed that the net is consistent, i.e.,

\[ \exists x > 0, A^T x = 0. \]  

Suppose there is a delay of at least \( d_i \) sec associated with transition \( \tau_i, i = 1, 2, \cdots, n \). This means that when \( \tau_i \) is enabled, \( a_{i \tau} \) tokens will be reserved in place \( p_i \), for at least \( d_i \) sec before their removal by firing \( \tau_i \), where \( a_{i \tau} \) is the weight of the arc from \( p_i \) to \( \tau_i \). 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 \( a_{i \tau} d_i x \), which can be written in matrix form

\[ (A^-)^T D x \]  

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

\[ \bar{M} \geq (A^-)^T D x. \]  

Taking the inner product of (40) with a nonnegative \( S \)-invariant \( y \) and using the invariance, \( y_i \bar{M} = y_i \bar{M}_0 \), we have

\[ y_i^T \bar{M} \geq y_i (A^-)^T D x \]

and

\[ r \geq y_i (A^-)^T D x / y_i^T \bar{M}_0. \]

Therefore, a lower bound of the cycle \( r \) or the minimum cycle time is given by

\[ \tau_{min} = \max_k \{ y_i^T (A^-)^T D x / y_i^T \bar{M}_0 \} \]

where the maximum is taken over all independent minimal-support \( S \)-invariants, \( \sum_i y_i \geq 0 \).

If we model a timed Petri net by assigning delay \( d_0 \) to each place \( p_i \), instead of the transitions, then it can be shown that \( \tau_{min} \) is given by

\[ \tau_{min} = \max_k \{ y_i^T D(A^-)^T x / y_i^T \bar{M}_0 \} \]

where \( D \) is the diagonal matrix of \( d_i, i = 1, 2, \cdots, m \) and \( A^- = [a_{i \tau}]_{n \times m} \) with \( a_{i \tau} \) being the weight of the arc from \( p_i \) to \( \tau_i \).

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

\[ \tau_{min} = \max_k \{ \text{the total delay in } C_j / M_0(C_j) \} \]

where \( M_0(C_j) \) denotes the number of tokens in \( C_j \) 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 \( t_i, t = 1, 2, 3, 4 \). From Example 21, we know that \( y_1 = (1 \ 1 \ 0 \ 1)^T \) and \( y_2 = (0 \ 1 \ 1 \ 1)^T \) 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

\[ \tau_{min} = \max \{ (d_1 + d_2 + d_3 + d_4)/d_1, d_3 + d_4 \}
\]

\[ + \ (d_1 + d_2)/k \]

\[ = d_2 + d_4 + (d_1 + d_3)/k. \]

\[ \square \]

B. Stochastic Nets and Performance Modeling

Suppose the delay \( d \), associated with transition \( \tau \), is a nonnegative continuous random variable \( X \) with the exponential distribution function

\[ F_X(x) = \Pr [X \leq x] = 1 - e^{-\lambda x}. \]

Then, the average delay is given by

\[ d = \int_0^\infty [1 - F_X(x)] dx = \int_0^\infty e^{-\lambda x} dx = \frac{1}{\lambda} \]

where \( \lambda \) is the firing rate of transition \( \tau_i \).

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 \( (N, M_0) \) under the SPN as follows. The MC state space is the reachability set \( R(M_0) \), and the transition rate from state \( M \) to state \( M' \) is given by \( q_{i j} = \lambda_{i j} \), the (possibly marking-dependent) firing rate of transition \( \tau_i \), transforming \( M \) into \( M' \). Then, \( q_{i j} = 0 \) if there are no transitions transforming \( M \) into \( M' \), \( i \neq j \) and \( q_{i j} \) is determined so as to satisfy \( \sum_j q_{i j} = 0 \). The square matrix \( Q = [q_{i j}] \) of order \( \varepsilon = |R(M_0)| \) is known as the transition rate matrix [30].

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

\[ \Pi Q = 0, \sum_i \pi_i = 1 \]
where \( \pi_i \) is the probability of being in state \( M_i \), and \( \Pi = (\pi_1, \pi_2, \ldots, \pi_s) \). From the steady-state distribution \( \Pi \), 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

\[
P(B) = \sum_{i \in B} \pi_i.
\]

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

\[
E[m_i] = \sum_{n=1}^{\infty} [nP(B(i, n))].
\]

3) The mean number of firings in unit time: Let \( B_j \) 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

\[
f_j = \sum_{M \in B_j} \frac{x_j}{-q_{ij}}
\]

where \( x_j \) is the firing rate of \( t_j \) and \( -q_{ij} \) is the sum of firing rates of transitions enabled at \( M_j \), i.e., the transition rate leaving state \( M_j \).

Example 23: Consider the SPN shown in Fig. 55. Transition \( t_5 \) fires at a marking-dependent rate given by \( \lambda_5 \).

\[\begin{align*}
\lambda_1 & \quad \lambda_2 & \quad \lambda_3 & \quad \lambda_4 & \quad \lambda_5 \\
\lambda_2 & \quad \lambda_3 & \quad \lambda_4 & \quad \lambda_5 & \quad \lambda_6 \\
\lambda_3 & \quad \lambda_4 & \quad \lambda_5 & \quad \lambda_6 & \quad \lambda_7 \\
\lambda_4 & \quad \lambda_5 & \quad \lambda_6 & \quad \lambda_7 & \quad \lambda_8 \\
\lambda_5 & \quad \lambda_6 & \quad \lambda_7 & \quad \lambda_8 & \quad \lambda_9
\end{align*}\]

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

where \( m_2 \) is the number of tokens in \( p_2 \). Transitions \( t_6, t_7, t_8, t_9 \) have (marking-independent) firing rates \( \lambda_6, \lambda_7, \lambda_8, \lambda_9 \) 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

\[
\begin{bmatrix}
-\lambda_{15} & \lambda_3 & 0 \\
\lambda_2 & -\lambda_{15} & \lambda_3 & \lambda_{15} \\
0 & \lambda_4 & 0 & 0 \\
\lambda_3 & 0 & 0 & 0 \\
\lambda_4 & 0 & 0 & 0 \\
\lambda_5 & 0 & 0 & 0
\end{bmatrix}
\]

where \( \lambda_{15} = \lambda_1 + \lambda_5 \).

Let \( \lambda_1 = \lambda_3 = 1/2 \) and \( \lambda_2 = \lambda_4 = \lambda_5 = 1 \). Then, we can solve (46) numerically for \( \Pi \) and find \( \pi_2 = 1/11, \pi_3 = \pi_5 = \pi_9 = 2/11 \). Thus, for example, we can find the average number of tokens in place \( p_2 \) as follows: since \( p_2 \) has one token at \( M_0, M_1, M_2, \) and 2 tokens at \( M_3, \) we have

\[
E[m_2] = \pi_0 + \pi_2 + 2\pi_3 = 6/11.
\]

Also, the mean number of firings of \( t_3 \) is given by

\[
f_3 = \frac{\lambda_3}{\lambda_{15} + \lambda_3} + \frac{\lambda_3}{\lambda_{15} + \lambda_2 + \lambda_3} + \frac{\lambda_3}{\lambda_{15} + \lambda_4} = \frac{1}{2} \pi_0 + \frac{1}{3} \pi_1 + \frac{1}{3} \pi_3 = \frac{7}{33}
\]

since \( t_3 \) is enabled at \( M_0, M_1, M_2 \), and no other states.

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 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.

![Diagram of a high-level net with transition firing rules](image)

The net consists of one transition $t$ and four places (two input places $p_1$ and $p_3$, and two output places $p_4$ and $p_5$). Note that the two arcs are labeled with $2x$, $(x,y)+(y,z)$, $(x,z)$, 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, $x$;
- $p_1$ loses two tokens of different colors, $(x,y)$ and $(y,z)$;
- $p_1$ gets one token of the color, $(x,z)$; and
- $p_1$ 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), $(a, b)$, $(b, c)$, and $(d, a)$;
- $p_1$ and $p_4$ have no tokens initially.

In the above, variables are denoted by $x$, $y$, $z$, · · · and constants are by $a$, $b$, $c$, $d$, · · · . 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 $\{a \mid x, b \mid y, c \mid z\}$ and $\{d \mid x, a \mid y, b \mid z\}$. 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 $\{a \mid x, b \mid y, c \mid z\}$ and $\{d \mid x, a \mid y, b \mid z\}$, 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 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.

![Unfolded net of the high-level net shown in Fig. 57](image)

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

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

All the $A_i$'s and $B$ are atomic formulae having the form $P(t_1, t_2, \cdots, t_k)$, where $P$ is a predicate symbol with $k$-ary arguments, and the $t_i$'s are terms. A term can be a variable or a constant. Clause (50) states that $B$ holds if $A_1, A_2, \cdots, A_n$ are true. It can be represented as a transition having $n$ input places, $A_1, A_2, \cdots, A_n$, 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 $\neg A_1, A_2, \cdots, A_n, n \geq 1$, 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) ←
2) Parent (Mary, Tom) ←
3) Ancestor (x, y) ← Parent (x, y)
4) Ancestor (x, z) ← Parent (x, y), Ancestor (y, z)
5) ← 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 × 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 ith clause and in the jth predicate symbol, where an argument to the right of the ← is prefixed with a negative sign. If the jth predicate symbol appears more than once in the ith clause, then a_{ij} will be the formal sum of all those arguments in the ith row and jth column.

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

\[
\begin{align*}
A &= \begin{pmatrix}
(D, M) & 0 \\
(M, T) & 0 \\
-\langle x, y \rangle & (\langle x, y \rangle \\
-\langle x, y \rangle & -\langle y, z \rangle + \langle x, z \rangle \\
0 & 0
\end{pmatrix}
\end{align*}
\]

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 \( s_1 \) and \( s_2 \) which start from the empty marking, fire the goal transition \( t_5 \), and end at the empty marking. The first one \( s_1 \) is as follows: fire \( t_1 \) 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 \( \{M \mid x\} \), i.e., \( x = \text{Mary} \) is an ancestor of Tom. The second firing sequence \( s_2 \) is as follows: fire \( t_1 \) and \( t_3 \) to produce the two tokens \( \langle D, M \rangle, \langle M, T \rangle \) in \( p_1 \); fire \( t_5 \) with substitution \( \{M \mid x \} \{T \mid y\} \) to move \( \langle M, T \rangle \) from \( p_1 \) to \( p_2 \); then fire \( t_5 \) with substitution \( \{D \mid x\}, \{M \mid y\}, \{T \mid z\} \) resulting a token \( \langle D, T \rangle \) in \( p_2 \); finally fire \( t_5 \) with \( \{D \mid x\} \), i.e., \( x = \text{David} \) is another ancestor of Tom.

It should be noted that the above two firing sequences

\[
\sigma_1 \text{ and } \sigma_2 \text{ have the following substitution vectors } X_1 \text{ and } X_2:
\]

\[
\begin{align*}
X_1 &= \begin{pmatrix}
\{M \mid x\} \\
\{T \mid y\}
\end{pmatrix} \\
X_2 &= \begin{pmatrix}
\{D \mid x\}, \{M \mid y\}, \{T \mid z\}
\end{pmatrix}
\end{align*}
\]

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 \( \mathcal{A}^+ X_1 = 0 \) and \( \mathcal{A}^+ X_2 = 0 \), where \( \mathcal{A} \) 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 \( t_5 \) be a goal transition. There exists a firing sequence which reproduces the empty marking and fires the goal transition \( t_5 \) in \( N \) if and only if \( N \) has a nonnegative T-invariant \( X \) such that \( \mathcal{A} X = X \).

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. Coto, D. J. Lee, H. Silver, V. Silva, 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

[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).


Additional Bibliography


[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,
2-12, June 1977.

[281] ---, "Petri nets," in Systems & Control Encyclopedia: The-
ory, Technology, Applications, M. G. Singh (Editor-in-Chief),

in Handbook of Software Engineering, C. R. Vick and C. V.


dations on Computer Science: in Proc. Symp. and Summer
School, High Tatras, Sept. 3–8, 1973, Math. Inst. of the Slovak

[285] ---, "Modelling as a communication discipline," in Com-
puting System Design: Proc. Joint IBM–University of Newcastle-upon-Tyne

[286] ---, "Modelling and Evaluating Computer Systems," H. Be-

[287] ---, "Introduction to general net theory," in Lecture Notes
in Computer Science, vol. 84, Net Theory and Applica-

[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 compu-
tation," Int. J. Theoret. Physics, vol. 21, no. 12, pp. 591–602,
1982.

[290] L. Pomello, "Some equivalence notions for concurrent sys-
tems," in Lecture Notes in Computer Science, vol. 222,
Advances in Petri Nets 1985, G. Rozenberg, Ed. New York:

[291] W. Reisig, "Deterministic buffer synchronization of sequen-

[292] G. S. Sacerdotto and R. L. Tenney, "The decidability of the
Symp. Theory Computing, Boulder, Colorado, May 2–

[293] H. A. Schmid and E. Best, "A step towards a solution of the
liveness problem in Petri nets," University of Newcastle-
1978.

[294] E. Schnieder, Proze Informatik. Braunschweig, Wiesba-

[295] C. L. Seitz, "Graph representations for logical machines," Cam-
bridge, MA: MIT, Department of Electrical Engineering,

[296] J. Sefakis, "A unified approach for studying the properties
of transition systems," Theoret. Comput. Sci., vol. 18,

Wissenschaften, 1980.

method for a Petri net with prescribed firing sequence," Trans.

[299] K. Tsui, S. Kumagai, S. Kodama, and T. Yamada, "Modelling
and verification of sequential control systems by Petri nets," in Proc.

models for automated manufacturing systems," Proc. 1987

[301] G. Winskel, "Petri nets, morphisms and compositionality,
" in Lecture Notes in Computer Science, vol. 222. New York:

[302] Tadao Murata (Fellow) received the M.S. and Ph.D. degrees in electrical en-
ingineering 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 Illi-
nois, he taught at the University of Cali-
ifornia at Berkeley and Tokai University,
Tokyo, Japan, and was invited to visit Peri'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 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 Sci-
ences/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 Interna-
tional 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