A New Approach to Petri Net Synthesis

The Knitting Technique

D.Y. Chao

Fig. 12(d) Add \([t_8 \ p_9 \ t_7]\), \([t_6 \ p_{10} \ t_5]\), \([t_5 \ p_{11} \ t_4]\), and \([t_3 \ p_{12} \ t_2]\); Add \([t_4 \ p_{13} \ t_3]\) and \([p_{13} \ t_8]\) via TT.2 & TT.4.
Preface

The advantages of distributed systems are many: resource sharing, cost reduction, improved accessibility and availability, and improved performance. The distributed nature allows concurrency or parallelism, and separate components of a system can work asynchronously. Concurrency increases the computation power, but different components have to cooperate with each other to allow resource sharing.

Due to the distributed nature, it is generally a very intricate problem to design a distributed system correctly and efficiently. The first step to a correct design lies in choosing the most appropriate model. The choice of which method to use for modeling concurrent systems is influenced by (1) ease of use and understanding, (2) modeling and analysis power, (3) generality, and (4) support of automation.

There are various models at hand: shared variables, exchange functions, communicating sequential processes, actors, data flow, and Petri nets. Petri nets is preferred to others due to the following reasons:

(1) The ability to show a precise and graphical representation,
(2) The availability of machine readable descriptions,
(3) The existence of analysis techniques for control aspects.
(4) The capability of employing top-down design methodology, and
(5) The possibility of design and analysis automation.

Thus, PNs have been used for modeling and analyzing concurrent systems. The net behavior depends not only on the graphical structure, but also on the initial marking of the net. Therefore they cannot be determined by static analysis such as dependency analysis; rather, they can be obtained with reachability analysis. The size of reachability graph depends not only by the structure of the net, but also by the initial marking. In general, the larger the initial marking (i.e., more tokens are involved), the larger the reachability graph. It has been shown that the complexity of the reachability analysis of the PNs is exponential. Though such analysis methods as reachability graph, reduction and linear algebra based methods are available, they are of limited use due to their limited capacity. Another disadvantageous fact is that modification and re-analysis may have to be conducted if the analysis methods have detected some undesired properties. Therefore, it is desired to have a synthesis approach which can build up a PN systematically such that the net has following desired logical properties: boundedness, liveness, and reversibility. These properties are critical for a system to operate in a stable, deadlock-free and cyclic way.
Since the PN synthesis research started in late 70's, significant results have been developed for special classes of PNs such as marked Graphs, free-choice PNs, and safe and live PNs. Two dominant synthesis approaches for general classes of PNs are bottom-up and top-down. Bottom-up approaches start with decomposition of systems into subsystems, construct sub-PNs for subsystems, and merge these subnets to reach a final PN by sharing places, transitions, and/or elementary paths or by linking subnets. Top-down approaches begin with a first-level PN and refine the net to satisfy system specification until a certain level is reached such as refinement of transitions and places by general modules with boundedness and liveness or well-defined modules. This approach has the advantage of reducing the scope of analysis from global to local.

However, the existing bottom-up approaches suffer from the difficulty to analyze the global system properties although such useful information as invariants of the global PN can be derived from those of its subnets. In the top-down approaches, there is no easy way to validate all general modules and their design is still a problem given the system specification. the other hand, many reduction rules are very much powerful in reducing PNs. Nevertheless, they are often difficult to be applied to synthesis directly. All these difficulties and problems motivate us to devise some simple but effective rules which can guide the synthesis of PNs with desired properties.

The Knitting Technique (KT) is a rule-based interactive approach. It tackles the synthesis problem from a different perspective. It aims to find the fundamental constructions to build any PN. There are two advantages of KT: (1) reduction of the complexity of synthesis as an interactive tool, and (2) providing knowledge of which construction building which class of nets. It therefore opens a novel avenue to the PN analysis.

Rather than refining transitions, KT generates new paths upon a PN, to produce a larger PN'. The new generations are performed in such a fashion that all reachable markings in PN remain unaffected in PN'; hence all transitions and places in PN stay live and bounded respectively. PN' is live and bounded by making the subnet of the new paths live and bounded. This notion is novel compared with other approaches and could synthesize more general PN than others.

Using KT, designers start with a basic process which is modeled by a set of closed-loop sequentially-connected places and transitions with a so-called home-place marked with a certain number of tokens. The tokens may represent the number of raw materials which can be present in a system each time. Then parallel, alternative, and exclusive processes or operations are added according to the system specification. Closed loops for the operations are added according to the resources required by the operations involved. Since expansions are conducted among nodes (either transitions or places) in a global way, the knitting technique is so called. This approach is easy to use due to the simple rules, and leads to a final net which is bounded, live, conservative, consistent, and reversible. The other advantage is that the approach is easily adapted to computer implementation to perform the synthesis of PNs in a user-friendly fashion.

The knitting technique is unconventional but powerful, specially practical in the field of manufacturing and robotics because it can be used to synthesize large and complex Petri nets. It may very well be the best and/or the unique method for synthesizing large and complex Petri nets.

Chapter 2 presents the basic knitting rules: TT and PP rules. Based on which, new paths are from transitions to transitions or from places to places maintaining good properties. The correctness of the rules are proved by finding the invariants of the synthesized net. It is simple, yet it creates more classes of nets than most other synthesis techniques using rules. Paths generations from transitions to places or vice versa are prohibited because the straightforward application of which renders the resulting net unbounded or
deadlocked. This restriction is removed in Chapter 3 a second order generation where any generation from a transition to a place must be followed by a generation from a place to a transition and vice versa. This allows more complicated classes of nets to be created, further proving the superiority of our approach.

Chapter 4 presents the reduction algorithm based on the same set of synthesis rules. The algorithm takes polynomial time complexity and is very powerful in the sense that it can reduce nets where traditional methods, usually with no algorithms, could not do it. Chapter 5 further enhances the knitting technique to the synthesis of structures of sequential mutual exclusion which is essential for maintaining the fairness in resource sharing such as in flexible manufacturing systems and other distributed systems. It removes the previous restriction of disallowing IT generations among transitions exclusive to each other. Chapter 6 extends the knitting technique to general Petri nets where any arc can carry multiple weights. Chapter 7 presents the algorithm for finding the structural matrix which records the temporal relationship of processes in a Petri net. It further applies the structure matrix to derive deadlock-free conditions. Chapter 8 applies our knitting technique to a automated manufacturing system which frequently can suffer from an ill-design.
# Table of Contents

1. The fundamental knowledge for Petri net theory 6

1.1 Introduction 6

1.2 Petri net structure 6

1.2.1 Petri net definition 6

1.2.2 Marked Petri nets 6

1.2.3 Node and element path 6

1.2.4 Other related definitions 6

1.2.5 Graphical representation 7

1.3 Petri net firing rule 7

1.3.1 Enabled transition 7

1.3.2 Firing rules 7

1.3.3 Conflict 7

1.3.4 Mutually exclusive 7

1.3.5 Home place 8

1.4 Petri nets modelling applications 8

1.4.1 General concept 8

1.4.2 Typical examples 8

1.5. Subclass of Petri nets 10

1.5.1 State machine 10

1.5.2 Marked graph (MG) 10

1.5.3 Free choice net (FC) 10

1.5.4 Extended Free choice net (EFC) 10

1.5.5 Asymmetric choice choice net (AC) 10

1.6 Petri net behavior properties 12

1.6.1 Safeness 12

1.6.2 Reachability 12

1.6.3 Boundedness 12

1.6.4 Liveness 12

1.6.5 Reversibility and home state 12

1.6.6 Conservation 13

1.6.7 Coverability 14

1.6.8 Persistence 14

1.6.9 Synchronic distance 14

1.7 Analysis method of Petri nets 14

1.7.1 Reachability tree 15

1.7.2 Incidence matrix and state equation 16

1.8 Structural properties of Petri nets 17

1.8.1 Structure liveness 17

1.8.2 Controllability 18

1.8.3 Structural boundedness 18

1.8.4 Conservativeness 18
8.3 The synthesis rules and deadlock-free condition 110
8.4 Examples of Petri net synthesis 113
8.5 Application illustration 119
8.5.1 Synthesis of automated manufacturing systems 119
8.5.2 Application to flexible manufacturing system preserving well-behaved properties 124

9. References 128
Chapter 1 The fundamental knowledge for petri net theory

1.1 Introduction

Petri nets are a graphical and mathematical tool for modeling numerous systems owing to their generality and permissiveness [1]. They have been quite useful as a formal tool to model and analyze systems with such characteristics as being concurrent, asynchronous, distributed, parallel, non deterministic, and/or stochastic. As a graphical tool, Petri nets, similar to flow charts and block diagrams, can be applied as a visual-communication aid. Besides, the dynamic and concurrent activities of systems can be simulated by moving tokens in these nets. It can also serves as a mathematical tool by building up state equations, algebraic equations, and other mathematics models governing the behavior of systems. Petri nets provide a powerful communication medium between practitioners and theoreticians. Practitioners can learn from theoreticians how nets have been proposed for a very wide variety of applications including modeling and analysis of distributed operating systems, distributed database systems, concurrent and parallel processes, flexible manufacturing/industrial control systems, discrete event systems, multiprocessor systems, data flow computing systems, fault-tolerant systems, programmable logic and VLSI arrays. Other interesting applications considered are communication protocol, neural networks, digital filters, and decision models.

1.2 Petri net structure

1.2.1 Petri net definition

A Petri net is kind of directed, weighted, bipartite graph consisting two kinds of nodes: places and transitions where arcs are either from a transition to a place or from a place to a transition.

Definition of Petri nets: A Petri net can be represented as four-tuple vector PN=(P,T,I,O) which consists of a set of places P, a set of transitions T, and input function I, and output function O.

1. P=(p1, p2, ..., pn)
2. T={t1,t2,...,ts}, s>0 with P∪T ≠ φ and P∩T ≠ φ
3. I:P×T→{0,1,...,k);
4. O:T×P→{0,1,...,h);

where p_i, t_i is an arbitrary element of sets P and T, respectively and the cardinality of the sets P and T is n and s, respectively. The net is called an Ordinary Petri net (OPN) if k=h=1. Otherwise, it is called a General Petri net (GPN).

Definition: Define for a Petri net PN the following set functions as

I(p, t)= •t = {p | (p, t) ∈I} = the set of input places of transition t.
O(p, t)= t•= {p | (t, p) ∈O} = the set of output places of transition t.
I(t, p)= •p = {t | (t, p) ∈O} = the set of input transitions of place p.
O(t, p)= p • = {t | (p, t) ∈I} = the set of output transitions of place p.

1.2.2 Marked Petri nets

A marking is to assign a nonnegative integer number k in a place p. Place p is then called a place with k tokens.

Definition: The marking M = {M1, M2, ..., Mn} can be defined as an n-vector where n is the total number of
places. The marking for place $p_i$, denoted by $M_i$ or $M(p_i)$, represents the number of tokens in place $p_i$.

**Definition:** A marked Petri net $(N, M_o)$ or $PN(M_o)$ is a Petri net $PN$ with an initial Marking $M_o$.

### 1.2.3 Node and elementary path

**Definition:** Given a marked Petri net $PN(M_o) = \{T, P, I, O, M_o\}$, a node is either a place in $P$ or a transition in $T$.

**Definition:** An elementary path is a sequence of nodes: $x_1, x_2, ... x_n$, $n \geq 1$, such that there exists an arc $(x_i, x_{i+1})$, $1 \leq i < n$ if $n > 1$, and $x_i = x_j$ implies that $i = j \forall 1 \leq i, j \leq n$.

### 1.2.4 Other related definitions

**Definition:** A transition without any input place is called a source transition.

**Definition:** A transition without any output place is called a sink transition.

**Definition:** If a place can accommodate an infinite amount of tokens, the corresponding Petri net is called a net with an infinite capacity. Otherwise, it is called a finite capacity net.

### 1.2.5 Graphical representation

In Petri net graphical representation, places are represented by circles, transitions by bars. Arcs are labeled with a positive integer to represent their weights. An $m$-weighted arc can be replaced by a set of $m$ parallel arcs with unity weight (weight equals to 1). Usually, the label for unity weight can be omitted. Marking can be represented by placing tokens in a place. Pictorially, this is drawn as black dots in places. For instance, a place $p$ which is marked with $k$ tokens can be graphed as a circle with $k$ dots in the place.

### 1.3 Petri net firing rule

#### 1.3.1 Enabled transition

A transition $t$ is said to be enabled if each of its input place $p_i$ consists of at least one token. The formal definition is described as follows:

**Definition:** A transition $t_j \in T$ in a marked Petri net $PN$ with marking $M$ is said to be enabled if for all $p_i \in \bullet t$, $M(p_i) \geq I(p_i, t_j)$.

#### 1.3.2 Firing rules

In a marked Petri net with marking $M$, an enabled transition $t$ may or may not fire depending on where the input tokens actually be consumed. The firing of an enabled transition $t$ removes one token from each of its input places and adds one token into each of its output places. The consequence of firing a transition will result a change from original marking $M$ to a new marking $M'$.

**Definition:** The firing rules are:
A transition \( t \in T \) is enabled if and only if \( M(p) > 0 \) when \( I(p, t) = 1 \) \( \forall p \in P \).

Firing an enabled transition \( t \) results in a new marking \( M' \) defined by
\[
M'(p) = M(p) + O(p, t) - I(p, t), \quad \forall p \in P.
\]

For a transition in simple Petri nets, the time from enabling to firing is indeterminate. Firing of a transition is assumed to be an instantaneous event.

### 1.3.3 Conflict

**Definition:** A set of transitions \( T_c \subseteq T \) is called in conflict if the firing of any subset of transitions \( \{t_i\} \) of \( T_c \subseteq T \) results in a marking in which some other transition \( t_j \in T_c \) and \( t_j \notin \{t_i\} \) is disabled.

In Figure 1-1, several examples of Petri nets being conflict are given. Note that transitions which share input places need not be in conflict. If the marking in the shared input places has enough tokens to enable each competing transition individually, then those transitions are not in conflict.

![Figure 1-1: Examples of Petri nets being conflict](image)

**No Conflict**     **In Conflict**     **No Conflict**     **In Conflict**

Figure 1-1. The figure is extended from Michael K. MOLLOY

### 1.3.4 Mutually exclusive

**Definition:** A set of transitions \( T_e \subseteq T \) is termed mutually exclusive if the firing of any transition \( t_i \in T_e \) results in a marking in which all other transitions \( t_j \in T_e \) \( \forall i \neq j \) are disabled.

### 1.3.5 Home place

The home place(s) is defined as the place(s) with tokens which can enable a transition at the initial marking. A formal definition is described as follows.

**Definition:** Suppose transition is enabled in the initial marking of a PN and a place \( p_h \) is an input place of \( t_i \), \( p_h \) is defined as a home place.

### 1.4 Petri nets modelling applications

#### 1.4.1 General concept

In modeling a particular discrete event system, places represent conditions, and transitions represent events. A set of input places to a transition can be viewed as pre-conditions of event, whereas the output places from a transition represent post-conditions of the event. The presence of a token in a place is explicated as holding the truth of the
condition associated with the place. Some typical examples of interpretation of a transition and their input places and output places are shown in Table 1-1.

<table>
<thead>
<tr>
<th>Transition</th>
<th>Input places</th>
<th>Output places</th>
</tr>
</thead>
<tbody>
<tr>
<td>Computation step</td>
<td>Input data</td>
<td>Output data</td>
</tr>
<tr>
<td>Task or job</td>
<td>Resources requested</td>
<td>Resources released</td>
</tr>
<tr>
<td>Clause in logic</td>
<td>Conditions</td>
<td>Conclusions</td>
</tr>
<tr>
<td>Event</td>
<td>Precondition</td>
<td>Postcondition</td>
</tr>
</tbody>
</table>

Table 1-1 Some interpretation of transitions and places

1.4.2 Typical examples

A. Communication protocols

Petri nets can be employed to represent and model essential features of a communication system. The properties of a Petri net are usually applied to validate a communication protocol. The Petri net shown in Figure 1-2 is a typical example of modeling a communication protocol between two processes.

![Sender Receiver Diagram](image)

Figure 1-2 A example of modeling a communication protocol between two processes.

B. Synchronization control

In a multiprocessor concurrent or distributed system, resources and data are usually shared among sites. Processes must be controlled or synchronized to insure correct overall operation. Petri nets have been used to model various synchronization mechanism, including readers/writers processes, producers/consumers processes, and dining philosopher problems. Figure 1-3 illustrates a synchronization problem of readers/writers, where the k tokens in p₁ represent k processes which may invoke read and write actions in a shared storage modeled by place p₃. Writer processes must mutually exclusive all other reader and writer process, but up to k processes may be resided in place.
p2(reading) when there is no token in p4(writing). p4 is safe because only zero' or one token is allowed in that place.

Figure 1-3 A reader/writers synchronization problem.

C. Multiprocessor systems

A multiprocessor system can be easily and explicitly represented by Petri nets. For example, Figure 1-4 shows a multiprocessor system with four processors, three common memories, and three buses. Tokens in place p1 represent the processors executing in their private memory, and tokens in p2 represent free buses. Transition t1 indicates the issuing of access requests. p3 contains outstanding requests and p4 represents processors having access to common memories. t2 and t3 model the memory choice: firing t3 choose the memory accessed by p4, whereas firing t2 refers to any other memory. Tokens in place p5 represent processors are queued to the same common memory that has been accessed by another processor (token) in p4. Firing t5 represents the completion of the access to the memory for which processors in p5 are queued. On the other hand, firing t4 represents the end of access to a memory for which there is no outstanding request.
1.5 Subclasses of Petri nets

This part briefly introduces some Petri net subclasses. These subclasses which have been defined are all syntactic or structural subclasses. The major advantage of this classification is to increase the modeling power for Petri nets. The characteristics reside in these subclasses can be used to determine if a given Petri net is a member of the specified subclass. The major properties, advantages, and drawbacks of each subclass is described later in this section. Before we introduce some subclasses of Petri nets, the ordinary Petri net is defined as follows:

**Definition:** A Petri net is called ordinary when all of its arc weights are 1's.

1.5.1 State Machine (SM)

**Definition:** An SM is an ordinary Petri net such that each transition has exactly one input place and one output place; i.e.,

\[ |\cdot t| = |t\cdot| = 1 \text{ for all } t \in T. \]

Several properties of SMs are apparent. First, since the number of tokens in SMs remains constant, they are strictly conservative.

1.5.2 Marked Graph (MG)

**Definition:** An MG is an ordinary Petri net such that each place has exactly one input transition and one output transition. i.e.,

\[ |\cdot p| = |p\cdot| = 1 \text{ for all } p \in P. \]

Since in an SM, each transition has one input and one output. In an MG, each place has one input and one output, MGs are dual of SMs. They are also dual from modeling viewpoint. An SM can be used to model a conflict situation such that one place has multiple outgoing arcs but can't represent a concurrent activity or the waiting of synchronization such that one transition has multiple outgoing arcs. On the other hand, an MG can represent concurrency and synchronization but cannot model conflict or data dependent decisions.
The properties for an MG includes liveness, safeness, and reachability. In the analysis of these properties, the major structural parts of an MG of interest are its cycles.

**Definition:** A cycle in an MG is a sequence of transitions \( t_{j1}, t_{j2}, t_{j3}, ..., t_{jk} \) such that for each \( t_{jr} \) and \( t_{jr+1} \) in the sequence there is a place \( p_{ir} \) with \( p_{ir} = t_{jr} \cdot, p_{ir} = \cdot t_{jr+1} \) and \( t_{j1} = t_{jk} \). A cycle is a closed path from a transition back to that same transition.

**1.5.3 Free Choice Net (FC)**

**Definition:** A FC is an ordinary Petri net such that each arc is either an unique output of a place or a unique input to a transition. i.e.,

\[ \lvert p \cdot \rvert = 1 \text{ or } (\lvert p \cdot \rvert = \{p\} \text{ for all } p \in P. \]

And this is equivalent to

\[ p_1 \cdot \cap p_2 \cdot \neq \emptyset \rightarrow \lvert p_1 \cdot \rvert = \lvert p_2 \cdot \rvert = 1 \text{ for all } p_1, p_2 \in P. \]

By the definition of FC, if a place is an input to multiple transitions (i.e., potentially conflict), then it is the only input for all of these transitions. Therefore, either none of these conflicting transitions are enabled or all are enabled simultaneously. This makes the firing choices of these conflict transitions freely.

**1.5.4 Extended Free-Choice Net (EFC)**

**Definition:** An EFC is an ordinary Petri net such that

\[ p_1 \cdot \cap p_2 \cdot \neq \emptyset \rightarrow p_1 \cdot = p_2 \cdot \text{ for all } p_1, p_2 \in P. \]

In EFC, each transition has at most one shared input place with another place and so also restrain the situation in which conflicts may occur.

**1.5.5 Asymmetric Choice Net (AC)**

**Definition:** An AC is an ordinary Petri net such that

\[ p_1 \cdot \cap p_2 \cdot \neq \emptyset \rightarrow p_1 \cdot \subseteq p_2 \cdot \text{ or } p_1 \cdot \supseteq p_2 \cdot \text{ for all } p_1, p_2 \in P. \]

Figure 1-5 displays the typical structures that represents these subclasses and an overview of these subclass classification in the Petri net structures is shown in Figure 1-6. It can be easily shown that FCs are a generalization of the structures of both SMs and MGs.
1.6 Petri net behavior properties

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 one referred to as marking-dependent or behavioral properties, whereas the latter type of properties is called structural properties. In this section, we only discuss behavioral properties and we will discuss the structural properties in section 1.8.
1.6.1 Safeness

**Definition:** A place \( p_i \in P \) of a Petri net \( C = \{P, T, I, 0, M_o\} \) is safe if for all \( M_i \in R(C, M_o) \), \( M(p_i) \leq 1 \). A Petri net is safe if each place in that net is safe.

A place in a Petri net is safe if the number of tokens in that place never exceeds one. A Petri net is safe if all places in the net are safe. Safeness is a very important property for hardware devices. If a place is safe, then it can be easily implemented by a single flip-flop.

1.6.2 Reachability

Reachability is perhaps a fundamental basis for analyzing the dynamic properties of any Petri net system. The firing of an enabled transition will change the markings. And a sequence of firings will result in a sequence of markings.

**Definition:** Given a Petri net \( PN \), a marking \( M_k \) is reachable if there exists a sequence of firings that lead \( M_o \) to \( M_k \). This expression can be denoted by \( M_k \in R(PN, M_o) \).

1.6.3 Boundedness

**Definition:** A place in a PN is said to be \( k \)-bounded if there exists a finite number \( k \in N \) such that \( M(p_i) < k \forall M \in S \). A PN is said to be bounded if each place \( p_i \) is bounded \( \forall p_i \in P \). For example, in some particular case as figure 1-2 shown, a Petri net is both bounded and safe. This result is specially useful for modeling an operating system where places are used to represent buffers or registers for storing intermediate data. By examining whether the net is bounded and safe, it can be predicted that if there will be overflows in the buffers or registers.

1.6.4 Liveness

Liveness usually means the complete absence of deadlocks in operating systems. A Petri net is said to be live if, no matter what marking has been reached from, 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.

**Definition:** A transition \( t_i \) is said to be live if for all \( M_k \in S \) there exists a marking sequence \( M_j \) which enables \( t_i \).

**Definition:** A PN is said to be "live" if each transition \( t_i \in T \) is live.

There are other level concepts related to liveness which have been considered in studies of deadlock. A transition \( t_i \) can be categorized as follow:

- **Level 0**: \( t_i \) can never be fired.
- **Level 1**: \( t_i \) is potentially fireable.
- **Level 2**: For every integer \( n \), there exists a firing sequence in which \( t_i \) occurs at least \( n \) times.
- **Level 3**: There exists an infinite firing sequence in which \( t_i \) occurs infinitely often.
1.6.5 Reversibility and Home state

**Definition:** A Petri net PN is said to be reversible if, for each marking \( M \) in \( R(M_0) \), \( M_0 \) is reachable from \( M \).

In other words, a reversible Petri net, one can always get back to the initial marking. In many applications one may want to go back to some (home) state instead of get back to the initial state. In this case, reversibility condition can be relaxed to a home state.

**Definition:** A marking \( M' \) is defined as a home state if, for each marking \( M \) in \( R(M_0) \), \( M' \) is reachable from \( M \).

Readers should notice that the above three properties, boundedness, liveness, and reversibility, are independent of each other. For example, a bounded Petri net can be live or nonlive and reversible or nonreversible. Figure 1-7 shows examples of Petri nets for all possible combinations of three properties.

![Diagram](image_url)

- (a) Bounded only
- (b) Reversible only
- (c) Live only
- (d) Live and reversible only
- (e) Bounded and Reversible
- (f) Bounded live reversible

Figure 1-7 Examples of Petri net for all possible combination of three properties.
1.6.6 Conservation

Conservation is an important property when Petri nets are used to model a resource sharing system where the number of resources in the system always remains constant. For example, a set of two line printers is represented by an initial marking of two tokens. A request of the printing job can be represented by a transition which has this place as an input. A release of the line printer is a transition with an output to the line printer place. We would like to show that tokens represented by resources are neither created nor destroyed.

Definition: A marked Petri net $PN(M) = \{P,T,I,O,M\}$ is strictly conservative if for $M' \in R(PN,M)$, $\sum_{p \in P} p \cdot M'(p) = \sum_{p \in P} p \cdot M(p)$.

Strict conservation is a very strong relationship. For instance, one can easily show that the number of inputs to each transition equals the number of outputs from that transition, $|I(ti)| = |O(ti)|$. However, a Petri net may not necessarily preserve the one-to-one mapping between resources and tokens. For instance, one token may be later used to produce multiple tokens through a transition. In general, we would like to define a weighting of tokens (an integer is assigned to each place). For a broader perspective, the weighted sum for all reachable marking should be constant while considering the conservation with respect to a weighting factor.

Definition: A marked Petri net $PN(M) = \{P,T,I,O,M\}$ is conservative with respect to a weighting vector $w = (w_1, w_2, ... , w_n)$, $n = |P|$, $w_i \geq 0$, if for all $M' \in R(PN,M)$, $\sum_{p \in P} p \cdot w \cdot M'(p) = \sum_{p \in P} p \cdot w \cdot M(p)$.

Readers can easily find that a strictly conservative Petri nets is just a special case with the weighting vector equals $(1, 1, ... , 1)$. On the other hand, all Petri nets are conservative with respect to weighting vector $(0, 0, ... , 0)$.

1.6.7 Coverability

Suppose we would like to know if any state is reachable with $M(p_4) \leq 1$ and $M(p_9) \leq 1$. This problem is similar but slightly different to reachability problem. It is called coverability problem.

Definition: A marking $M$ in a Petri net $(PN, M_0)$ is said to be coverable if for each $p$ in the net, there exits a marking $M'$ in $R(M_0)$ such that $M'(p) \geq M(p)$. Coverability is closely related to level 1 liveness (potentially firable). Let $M$ be the minimum marking needed to enable a transition $t$. Then $t$ is dead (non level 1 live) if and only if $M$ is not coverable.

Definition: $t$ is level 1 live if and only if $M$ is coverable.

1.6.8 Persistence

Remind that conflict means that for any two enabled transitions, the firing of any one will disable the other one, whereas persistence can be thought as a conflict-free property. A Petri net is said to be persistent if, for any two enabled transitions, the firing of any one will not disable the other.

In the other words, the persistence property guarantees that once a transition is enabled, it will stay enabled until it fires. Persistence property is useful in modeling the context of parallel programming schemata and
speed-independent asynchronous circuits. Besides, a safe persistent Petri net can be transformed to a marked graph
where for any transition, the input to a transition and the output from a transition is always 1. Note that all marked
graphs are persistent, but not vice versa. For example, the unbounded(not safe) net shown in Figure 1-7 (c) is persistent
but not a marked graph.

1.6.9 Synchronic Distance

Synchronic distance concept is firstly introduced by C.A.Petri. It is a metric used to determine the degree of
mutual dependence between two events in a condition/event system. The synchronic distance between any two
transitions can be obtained as follows:
For each firing sequence of the Petri net in one iteration, calculate the absolute value of the difference between the
numbers of the firing of these two transitions.

Definition: Given any two transitions \( t_1 \) and \( t_2 \) in a initial marked Petri net \( PN=\{P, T, I, O, M_0\} \), the
synchronic distance is defined by \( d_{12}=\max |D_1(t_1)-D_2(t_2)| \) where \( D \) is any firing sequence starting at any marking \( M \) in
\( R(M_0) \) and \( D(t_i) \) is the number of times that transition \( t_i \), i=1 or 2 fires in \( D \). For example, in the net shown in Figure
1-7 (d), \( d_{12} = 1 \), \( d_{34} = 1 \), and \( d_{14} = \infty \).

1.7 Analysis method for Petri nets

Usually, there are three categories of approaches used for analyzing Petri. nets: A( Reachability, tree approach, B)
Incidence matrix and state equation approach. The reachability tree approach is usually limited to a small net because it
has been shown that the: complexity of reachability analysis for Petri nets is exponential. On the other hand, matrix
approach is powerful in analysis but also applicable to some particular subclasses of Petri nets.

1.7.1 Reachability Tree

Given a initial marking \( M_0 \) for a Petri net \( PN \), it can result numerous new marking by a sequence of firing of
enabled transitions. And from these new markings, we can obtain even more new markings. We can then generate a
tree structure called reachability tree to represent these markings starting from \( M_0 \) (the root). In reachability tree,
markings generated from initial marking are represented by nodes. Each arc represents the firing of an enabled
transition which will transform from one marking to another.

In order to simplify the representation, the symbol \( \omega \) is introduced to represent an infinite set of values. For any
integer \( n>\omega \), \( n+\omega=\omega \) and \( \omega \geq \omega \). The physical meaning for that is that if a component of a covering marking is \( \omega \), then
there exists a "loop" in the path from root to particular covering marking.

If one covering marking contains more than one it may indicate a "loop" interaction among the marking
exchanges. Consider the example in Figure 1-8 and its corresponding reachability tree in Figure 1-9.
Again, the reachability tree can be used to detect the safeness, boundness, conservation, and coverability problems. However, the use of $w$ instead of an exact value also degrades its analytical power due to a loss of information. It can't use to solve the reachability or liveness problems or determine the possible firing sequences. For instance, two different Petri nets may result in the same readability tree. Figure 1-10(a) and Figure 1-10(b) describe the problems, whose reachability tree is given in Figure 1-11. Another problem exists in the liveness analysis where a covering marking in a reachability tree may be a deadlock. Figure 1-12 presents this problem.
1.7.2 Incidence Matrix and State Equation

This incidence matrix and state equation approach basically uses matrix equation to govern and present the dynamic behaviors of a Petri net system. However, because of non-deterministic nature in Petri net and some pre-constraint in the equations, the solvability of these equations is somewhat limited.

A. Incidence Matrix

**Definition:** The incidence matrix $A = [a_{ij}]$ is an $n \times m$ matrix of integers for a Petri net PN with $n$ places and $m$ transitions. Its entry $a_{ij}$ is defined by

$$a_{ij} = a_{+ij} - a_{-ij}$$

where $a_{+ij} = w(i,j)$ is the weight for the outgoing arc from transition $i$ to place $j$ and $a_{-ij} = w(j,i)$ is the weight for the incoming arc from place $j$ to transition $i$.

The physical meaning for the number of tokens changed for place $j$ when each time transition $i$ fires. Accordingly, $a_{+ij}$ represents the number of tokens removed and $a_{-ij}$ represents the number of tokens added when transition $i$ fires. Therefore, each row of $A$ corresponds to a transition and each column of $A$ corresponds to a place.

For example, suppose the weight for each arc in Figure 1-13(a) is 1. The incidence matrix is shown in Figure 1-13(b).

![Figure 1-13](image.png)

B. State Equation

In writing matrix equation, we write a marking $M_i$ as a $M \times 1$ column vector. The $j$th entry of $M_i$ denotes the number of tokens in place $j$ right after the $i$th firing in certain firing sequence. The $i$th firing vector $u_i$ is an $n \times 1$ column vector of $n-1$ 0's and one nonzero entry. We can write the following state equation for a Petri net

$$M_i = M_{i-1} + A^T u_i$$

C. Necessary Reachability Condition

A coverable marking $M_k$ is reachable from initial $M_0$ through a firing sequence $\{u_1, u_2, ..., u_k\}$, then $M_k$ can be written as follows:

$$M_k = M_0 + A^T \sum_{i=1}^{k} u_i$$

or
\[ A^T x = \Delta M \] where \( x = \sum_{i=1}^{k} u_i \) is a firing count vector and \( \Delta M = M_k - M_o \).

For a homogeneous system, \( \Delta M = 0 \).

## 1.8 Structural Properties of Petri Nets

In the following sections, we will introduce several important structural properties of Petri nets. The basic difference between behavior properties and structural properties is the dependency on initial marking. Structural properties are those properties that are independent on initial marking but dependent on topological structures of Petri nets. That is, those properties remain true no matter what initial marking is or are concerned with certain firing sequence from some initial marking. These properties can often be identified and categorized by means of the incidence matrix \( A \) and its associated homogeneous equations or inequalities.

### 1.8.1 Structural Liveness

**Definition:** A Petri net \( PN \) is considered to be structurally live if there exists a live initial marking for \( PN \).

It can be easily shown that a MG is structurally live.

### 1.8.2 Controllability

**Definition:** A Petri net is controllable if any marking is reachable from any other marking. That is, for \( \forall M, \exists M' \) such that \( M \in R(M') \).

**Theorem:** If a Petri net \( PN \) with \( m \) places is completely controllable, then we have

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

### 1.8.3 Structural Boundedness

**Definition:** If a Petri net \( PN \) is bounded for any finite initial marking \( M_o \), then it is said to be structurally bounded.

**Theorem:** A Petri net \( PN \) is structurally bounded iff there exists an integer column vector \( y^T > 0 \) (i.e., each component of the vector \( y \) is greater than zero), such that

\[ A^T y \leq 0, \] where \( A \) is an incidence matrix.

**Corollary:** A Petri net \( PN \) is structurally unbounded iff there exists an integer column vector \( x \geq 0 \) (i.e., each component of the vector is nonnegative), such that

\[ A^T x = \Delta M > 0, \] where \( \Delta M(p) > 0 \) (i.e., the \( p \)th entry of \( \Delta M > 0 \)), \( p \) \( \in \) \( P \) \( A^T \) is the transpose of \( A \).

### 1.8.4 Conservativeness

**Theorem:** A Petri net \( PN \) is said to be (strictly) conservative if there exists a (positive) nonnegative integer \( y(p) \) for every place \( p \) such that the weighted sum of tokens, \( M^T y = M_o^T y \) is a constant, for all \( M \in R(M) \).
**Theorem:** A Petri net is (strictly) conservative iff there exists an (positive) nonnegative in eger column vector y such that Ay = 0.

### 1.8.5 Repetitiveness

**Definition:** A Petri net PN is (partially) repetitive if there exists an initial marking and a firing sequence such that any transition in can appear infinite times.

**Theorem:** A Petri net PN is (partially) repetitive iff there exists an positive (nonnegative) integer such that $A^Tx \geq 0$.

### 1.8.6 Consistency

**Definition:** A Petri net is consistent if there exists a firing sequence a such that the initial marking $M_0$ can be recovered (starts from $M_0$ and back to $M_0$), and this firing sequence contains every transition in the net at least once.

**Theorem:** A Petri net PN is consistent iff there exists a positive integer vector $x > 0$ such that $A^Tx = 0$.

It is apparent that conservativeness is a stronger property than structural boundedness, whereas consistency is a special case of repetitiveness. Also, if a Petri net is structurally bounded and structurally live, then it is both conservative and consistent. Table 1-2 summarizes these structural properties.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Necessary and Sufficient conditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structurally Bounded</td>
<td>$\exists \ y &gt; 0, Ay \leq 0 \ (or \ Not \ \exists \ x &gt; 0, A^Tx &gt; 0)$</td>
</tr>
<tr>
<td>Conservative</td>
<td>$\exists \ y &gt; 0, Ay = 0 \ (or \ Not \ \exists \ x &gt; 0, A^Tx &gt; 0)$</td>
</tr>
<tr>
<td>Repetitive</td>
<td>$\exists \ x &gt; 0, A^Tx \geq 0$</td>
</tr>
<tr>
<td>Consistent'</td>
<td>$\exists \ x &gt; 0, A^Tx = 0 \ (or \ Not \ \exists y, Ay &gt; 0)$</td>
</tr>
</tbody>
</table>

Table 1-2

### 1.8.7 P-invariant, T-invariant, support and T-condition

**Definition of P-invariant:** An integer y vector is called a P-invariant if $Ay = 0$.

**Theorem:** An m-vector y is an P-invariant iff $M_0^T = M_0$ for any $M_0$ and any $M \in R(M_0)$

**Definition of T-invariant:** An integer x vector is called a T-invariant if $A^T x = 0$. Let $Z$ be a set of places, $y_i$ the i-th component of the y vector and $Y(Z) = \sum_{p \in Z} Y_i$.

**Theorem:** An n-vector $x \geq 0$ is T-invariant iff there exists an initial marking $M_0$ and a firing sequence $\sigma$ starting from $M_0$ back to $M_0$ where its firing count vector equals to $x$.

**Definition:** The set of places p(transition t such that $y(p) > 0$, $x(t) > 0$) for P-invariant (T-invariant) is called the support of the P-invariant(T-invariant) and is defined as $||y|| \ (|| x ||)$.

**Definition:** A minimum support is a support and none of its proper subset is a support.

**Definition:** An invariant y is said to be minimal if there is no other invariant $y'$ such that $y'(p) < y(p)$ for all...
p. Minimal support invariant is a unique minimal invariant corresponding to the minimal support.

**Theorem:** The condition is defined as a T-condition if for any ordinary Petri net (OP), $Ay \leq 0$ iff $\forall t \in T$, $Y(\cdot t) \geq Y(t \cdot)$.

1.9 Summary

In this chapter, we provide a brief review of the fundamental knowledge for the Petri net theory. The structure of Petri nets are introduced and some examples of application are given for illustration. We also explain some important properties for Petri nets. These properties especially the structural properties, are very important concepts when we discuss the synthesis rules for Petri nets in later chapters. We also introduce the subclass of Petri nets. These subclasses enhance the analytical power but sacrifice some modeling capability. Best of all, readers will see that the knitting technique that will be introduced starting from chapter 3 through this book covers even more than asymmetric nets does and automatically reserves all those well behavior properties as well.
9. REFERENCES


[CHA 93f] "A CAD tool for constructing large Petri nets", revised for IEEE Trans.SMC.


[CHA 93h] and Wang, D.T., "XPN-FMS: A Modeling and simulation software for FMS Using Petri nets and X window", accepted by Int'l. J. FMS.


