Discrete Event System Modeling and Simulation

Xiaoou Li
Sección de Computación
Departamento de Ingeniería Eléctrica
Av. IPN 2508, A.P.14-740, México D.F., 07360,
México
email: lixo@cs.cinvestav.mx

Mayo 8, 2000
1

Modeling Concepts and Methodology—Petri Net Approach

1. Petri net basics (6hrs)
   (a) History
   (b) Petri net notation and definitions
   (c) Petri net marking and state space
   (d) Petri net dynamics
   (e) Petri net language
   (f) Examples, exercises

2. Analysis of Petri nets (4hrs)
   (a) Problem classification (properties) (2hrs)
   (b) The coverability tree (2hrs)
   (c) Incidence matrix and state equation (1hrs)
   (d) Simple reduction rules for analysis (1hrs)

3. Abbreviations, extensions and particular structures (6hrs)
   (a) Classification
   (b) Modeling by autonomous PNs
   (c) Modeling by non-autonomous PNs
1. Modeling Concepts and Methodology—Petri Net Approach

4. Colored petri net and its applications (4hrs)

5. Fuzzy Petri net and its applications (2hrs)

6. Generalized stochastic Petri net and its applications (2hrs)

7. Hybrid Petri net and its applications (2hrs)

1.1 Petri Net Basics

1.1.1 History


1. 1962, Petri’s doctoral thesis, west Germany

2. —1970, early developments and applications of Petri nets are found in

   (a) reports associated with the project "Information System Theory Project of Applied Data Research" (in USA)

   (b) record of the 1970 Project MAC Conference on Concurrent Systems and Parallel Computation

3. 1970—1975, the Computation structure Group at MIT was the most active. In July, 1975, a conference on Petri Nets and Related Methods at MIT. (no proceedings)

1. Modeling Concepts and Methodology—Petri Net Approach


6. 1985—another series of international workshops, placed emphasis on timed and stochastic nets and their applications to performance evaluation. (every two years)

7. 1980–1989, European Workshops on Applications and Theory of Petri Nets had been held every year at different location in Europe. Selected papers have been published by Springer-Verlag as *Advances in Petri Nets*.

8. 1989—*Petri Net Newsletter* lists short abstracts of recent publications

1.1.2 Where can a Petri net apply to?

They can be applied informally to any area or system that can be described graphically like flow charts and that needs some means of representing parallel or concurrent activities.

Artificial Systems (or DEDS): manufacturing systems, software systems, traffic systems, information systems, etc.

- Can not be described by traditional difference equation
- Characterized by states and events
- Can be modeled using models which are specially used in DEDS such as automata, Petri nets, finite-state machines
- can be studied under those models mentioned above or using simulation method.

Two successful application areas: *performance evaluation and communication protocols*.

Promising areas:
• modeling and analysis of distributed-software systems

• distributed-database systems

• concurrent and parallel programs

• flexible manufacturing/industrial control systems

• discrete-event systems

• multiprocessor memory systems

• data-flow computing systems

• fault-tolerant systems

• programmable logic and VLSI arrays

• asynchronous circuits and structures

• compiler and operating systems

• office-information systems

Other interesting applications: local-area networks, legal systems, human factors, neural networks, digital filters, and decision models.

Comments on applications: Modeling, Seeking properties, Logic controller, Performance evaluation

1.1.3 Petri net notation and definitions

Definition 1 (Petri net graph) A Petri net graph (or Petri net structure) is a weighted bipartite graph
1. Modeling Concepts and Methodology—Petri Net Approach

\[(P, T, A, w)\]

where

\[P = \{p_1, p_2, \ldots, p_n\}\] is the finite set of places (one type of node in the graph)

\[T = \{t_1, t_2, \ldots, t_m\}\] is the finite set of transitions (the other type of node in the graph)

\[F \subseteq (P \times T) \cup (T \times P)\] is the set of arcs from places to transitions and from transitions to places in the graph.

\[w : I \cup O \rightarrow \{1, 2, 3, \ldots\}\] is the weight function on the arcs.

We assume that \((P, T, F, w)\) has no isolated places or transitions.

the set of input (output) places of transition \(t_j\):

\[\cdot t = I(t_j) = \{p_i \in P : (p_i, t_j) \in F\}, t \cdot = O(t_j) = \{p_i \in P : (t_j, p_i) \in F\}\]

Similar notation can be used \(I(p_i), O(p_i)\).

**Example 1.1** a well-known chemical reaction: \(2H_2 + O_2 \rightarrow 2H_2O\) can be described by a Petri net.

**Example 1.2** Products on sale

**Example 1.3** Two computers use a common memory

1.1.4 Petri net marking and state space

We need a mechanism indicating whether these conditions are in fact met or not. This mechanism is provided by assigning tokens to places.

A *token* is something we "put in a place" essentially to indicate the fact that the condition described by that place is satisfied. The way in which tokens are assigned to a Petri net graph defines a marking.
Formally, a marking $x$ of a Petri net graph $(P,T,F,w)$ is a function $x : P \rightarrow N = \{0, 1, 2, \ldots \}$. Thus marking defines row vector $x = [x(p_1), x(p_2), \ldots, x(p_n)]$, where $n$ is the number of places in the Petri net. The $i$th entry of this vector indicates the number of tokens in place $p_i$, $x(p_i) \in N$.

**Definition 2 (Marked Petri net)** A marked Petri net is a five-tuple $(P,T,F,w,x)$ where $(P,T,F,w)$ is a Petri net graph and $x$ is a marking of the set of places $P$; $x = [x(p_1), x(p_2), \ldots, x(p_n)] \in N^n$ is the row vector associated with $x$.

For simplicity, we shall henceforth refer to a marked Petri net as just a Petri net.

The state of a Petri net is defined to be its marking row vector $x = [x(p_1), x(p_2), \ldots, x(p_n)]$.

The state space $X$ of a Petri net with $n$ places is defined by all $n$-dimensional vectors whose entries are non-negative integers, that is $X \in N^n$.

**Definition 3 (enabled transition)** A transition $t_j \in T$ in a Petri net is said to be enabled if

$$x(p_i) \geq w(p_i, t_j) \text{ for all } p_i \in I(t_j)$$
1. Modeling Concepts and Methodology—Petri Net Approach

In words, transition \( t_j \) in the Petri net is enabled when the number of tokens in \( p_i \) is at least as large as the weight of the arc connecting \( p_i \) to \( t_j \), for all places \( p_i \) that are input to transition \( t_j \).

Since places are associated with conditions regarding the occurrence of a transition, then a transition is enabled when all the conditions required for its occurrence are satisfied; tokens are the mechanism used to determine satisfaction of conditions.

1.1.5 Petri Net Dynamics (Execution rules)

The state transition mechanism is provided by moving tokens through the net and hence changing the state of the Petri net. When a transition is enabled, we say that it can fire. The state transition function of a Petri net is defined through the change in the state of the Petri net due to the firing of an enabled transition.

**Definition 4** (Petri net dynamic) The state transition function, \( f : \mathbb{N}^n \times \rightarrow \mathbb{N}^n \), of Petri net \((P, T, F, w, x)\) is defined for transition \( t_j \in T \) if and only if

\[
x(p_i) \geq w(p_i, t_j) \quad \text{for all } p_i \in I(t_j)
\]
If \( f(x, t_j) \) is defined, then we set

\[
x'(p_i) = x(p_i) - w(p_i, t_j) + w(t_j, p_i), \quad i = 1, \ldots, n
\]  

(1.1)

**Remark 1.1** The number of tokens need not be conserved upon the firing of a transition in a Petri net. Since it is entirely possible that

\[
\sum_{p_i \in P} w(t_j, p_i) > \sum_{p_i \in P} w(p_i, t_j) \quad \text{or} \quad \sum_{p_i \in P} w(t_j, p_i) < \sum_{p_i \in P} w(p_i, t_j)
\]

In general, it is entirely possible that after several transition firings, the resulting state is \( x = [0, 0, \ldots, 0] \), or that the number of tokens in one or more places grows arbitrarily large after an arbitrarily large number of transition firings.

**Example 1.4** (Firing of transitions)

**Definition 5** (reachable states) The set of reachable states of Petri net \((P, T, F, w, x)\) is

\[
R[(P, T, F, w, x)] := \{ y \in \mathbb{N}^n : \exists s \in T^*(f(m, s) = y) \}.
\]

(example fig4_4)

**State Equations — a convenient algebraic tool**

Let us define the firing vector \( u \), an \( m \)-dimensional row vector of the form

\[
u = [0, \ldots, 0, 1, 0, \ldots, 0]
\]

where the only 1 appears in the \( j \)th position, \( j \in \{1, \ldots, m\} \), to indicate that fact that the \( j \)th transition is currently firing.

The incidence matrix of a Petri net, \( A \), an \( m \times n \) matrix whose \((j, i)\) entry is of the form

\[
a_{ji} = w(t_j, p_i) - w(p_i, t_j).
\]
Using the incidence matrix $A$, we can now write a vector state equation

$$x' = x + u \cdot A$$

(1.2)

which describes that state transition process as a result of an "input" $u$, that is, a particular transition firing. The $i$th equation in 1.2 is precisely equation 1.1. Therefore, $f(x, t_j) = x + u \cdot A$, where $f(x, t_j)$ is the transition function.

The state equation provides a convenient algebraic tool and an alternative to purely graphical means for describing the process of firing transitions and changing the state of a Petri net.

**Example 1.5** (State equation) consider the Petri net of figure 1.5(a), with the initial state $x_0 = [2, 0, 0, 1]$. We can first write down the incidence matrix by inspection of the Petri net graph, which in this case is

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

The $(1,2)$ entry, for example is given by $w(t_1, p_2) - w(p_2, t_1) = 1 - 0$. Using equation 1.2, the state equation when transition $t_1$ fires at state $x_0$ is

$$x_1 = \begin{bmatrix} 2 & 0 & 0 & 1 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & -1 & -1 \end{bmatrix}$$

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

which is precisely what we obtained in example (??). Similarly, we can determine $x_2$ as a result of firing $t_2$ next, as in fig. 1.5(c):

$$x_2 = \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} -1 & 1 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & -1 & -1 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 0 & 2 \end{bmatrix}$$
1. Modeling Concepts and Methodology—Petri Net Approach

1.1.6 Petri net Language

Definition 6 (Labeled Petri net) A labeled Petri net $N$ is an eight-tuple

$$N = (P, T, F, w, E, \ell, x_0, X_m)$$

where

$(P, T, F, w)$ is a petri net graph

$E$ is the event set for transition labeling

$\ell : T \rightarrow E$ is the transition labeling function

$x_0 \in N^n$ is the initial state of the net (i.e., the initial number of tokens in each place)

$X_m \subseteq N^n$ is the set of marked states of the net.

In Petri net graphs, the label of a transition is indicated next to the transition.

Definition 7 (Language generated and marked) The language generated by labeled Petri net $N = (P, T, F, w, E, \ell, x_0, X_m)$ is

$$\mathcal{L}(N) := \{ \ell(s) : s \in T^* \text{ and } f(x_0, s) \text{ is defined} \}.$$  

The language marked by $N$ is

$$\mathcal{L}_m(N) := \{ \ell(s) \in \mathcal{L}(N) : s \in T^* \text{ and } f(x_0, s) \in X_m \}.$$  

(This definition uses the extended form of the transition labeling function $\ell : T^* \rightarrow E^*$; this extension is done in the usual manner.)

These definitions are completely consistent with the corresponding definitions for automata. The language $\mathcal{L}(N)$ represents all strings of transition labels that are obtained by all possible (finite) sequences of transition firings in $N$, starting in the initial state $x_0$ of $N$; The marked language $\mathcal{L}_m(N)$ is the subset of these strings that leave the Petri net in a state that is a member of the set of marked states given in the definition of $N$. 
The class of languages that can be represented by labeled Petri nets is

$$\mathcal{PNL} := \{K \subseteq E^* : \exists N = (P, T, w, E, \ell, x_0, X_m) | \mathcal{L}_m(N) = K \}.$$ 

This is a general definition and the properties of $\mathcal{PNL}$ depend heavily on the specific assumptions that are made about $\ell$ (e.g., injective or not) and $X_m$ (e.g., finite or infinite).

1.1.7 Modeling with Petri net (examples)

Events and Conditions

The simple Petri net view of a system concentrates on two primitive concepts: events and conditions.

<table>
<thead>
<tr>
<th>System</th>
<th>Petri net</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditions</td>
<td>Places</td>
</tr>
<tr>
<td>Events</td>
<td>Transitions</td>
</tr>
<tr>
<td>Preconditions of an event</td>
<td>Inputs of the corresponding transition</td>
</tr>
<tr>
<td>Postconditions of an event</td>
<td>Outputs of the corresponding transition</td>
</tr>
</tbody>
</table>

Example 1.6 (a simple machine shop modeling problem) The machine shop waits until an order appears and then machines the ordered parts and sends it for delivery.

The conditions for the system are:

1. (a) The machine shop is waiting  
   (b) An order has arrived and is waiting  
   (c) The machine shop is working on the order  
   (d) The order is complete

The events would be
1. An order arrives

2. The machine shop starts on the order

3. The machine shop finishes the order

4. The order is sent for delivery

<table>
<thead>
<tr>
<th>Event</th>
<th>Preconditions</th>
<th>Postconditions</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>None</td>
<td>b</td>
</tr>
<tr>
<td>2</td>
<td>a,b</td>
<td>c</td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>d,a</td>
</tr>
<tr>
<td>4</td>
<td>d</td>
<td>None</td>
</tr>
</tbody>
</table>

Example 1.7 (More complicated machine shop) The machine shop has three different machines, $M_1$, $M_2$ and $M_3$ and two operators, $F_1$ and $F_2$. Operator $F_1$ can operate machines $M_1$ and $M_2$ while operator $F_2$ can operate machines $M_1$ and $M_3$. Orders require two stages of machining. First they must be machined by machine $M_1$ and then by either machine $M_2$ or $M_3$.

This more complex system would have the following conditions

1. (a) An order has arrived and is waiting for machining by $M_1$

   (b) An order has been processed by $M_1$ and is waiting to be processed by $M_2$ or $M_3$.

   (c) The order is complete

   (d) Machine $M_1$ is idle

   (e) Machine $M_2$ is idle

   (f) Machine $M_3$ is idle
(g) Operator $F_1$ is idle
(h) Operator $F_2$ is idle
(i) Machine $M_1$ is being operated by $F_1$
(j) Machine $M_1$ is being operated by $F_2$
(k) Machine $M_2$ is being operated by $F_1$
(l) Machine $M_3$ is being operated by $F_2$

The following events can occur:

1. An order arrives
2. Operator $F_1$ starts the order on machine $M_1$
3. Operator $F_1$ finishes the order on machine $M_1$
4. Operator $F_2$ starts the order on machine $M_1$
5. Operator $F_2$ finishes the order on machine $M_1$
6. Operator $F_1$ starts the order on $M_2$
7. Operator $F_1$ finishes the order on $M_2$
8. Operator $F_2$ starts the order on $M_3$
9. Operator $F_2$ finishes the order on $M_3$
10. The order is sent for delivery

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

Petri Net Model for Queueing Systems

Insert figure 4.5

a: customer arrival
s: service starts
c: service completes and customer departs

(a) Simple queueing system
(b) Petri net model of simple queueing system in initial state [0,1,0]
(c) Petri net model of simple queueing system with initial state [0,1,0] after firing sequence \{a,s,a,a,c,s,a\}.

\textit{Finite-State Machines}

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 a Petri net.

Note that each transition in this net has exactly one incoming arc and exactly one outgoing arc. The subclass of Petri net with this property is known as state machines. State machines allow the representation of decisions (or conflicts, choice), but not the synchronization of parallel activities.

**Example 1.8 (Dataflow Computation)**

Insert figure 8 (murata)

Concurrency and Conflict

Petri net features:

Concurrency (or parallelism): In Petri net model, two events which are both enabled and do not interact may occur independently. There is no need to synchronize events unless it is required by the underlying system which is being modeled. When synchronization is needed, it is easy to model this also. Thus, Petri nets would seem ideal for modeling systems of distributed control with multiple processes executing concurrently in time.

Asynchronous: There is no inherent measure of time or the flow of time in a Petri net. This reflects a philosophy of time which states that the only important property of time, from a logical point of view, is in defining a partial ordering of the occurrence of events. Events take variable amounts of time in real life, and this variability is reflected in the Petri net model by not depending on a notion of time to control the sequence of events. The Petri net structure itself contains all necessary information to define the possible sequence of events. (example)

Nondeterminism: The order of occurrence of the events is one of possibly many allowed by the basic structure.

Instantaneous: taking zero time, and the occurrences of two events cannot happen simultaneously.
1.2 Analysis of Petri Net

1.2.1 Problem classification (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:

1. those which depend on the initial marking (referred as marking-dependent or behavioral properties)

2. those which depend on the initial marking (structural properties)

The following are some of the key issues related to the logical behavior of Petri nets. These issues relate primarily to desirable properties that often have their direct analogs in CVDS. Many of these properties are motivated by the fact that Petri net models are often used in resource sharing environments where we would like to ensure efficient and fair usage of the resources.

Reachability

A marking $M_n$ is said to be reachable from a marking $M_0$ if there exists a sequence of firings that transforms $M_0$ to $M_n$.

- $\sigma = M_0 \ t_1 \ M_1 \ t_2 \ M_2 \ \cdots \ t_n \ M_n$: a firing or occurrence sequence. or simply $\sigma = t_1 \ t_2 \ \cdots \ t_n$.

- $M_0 | \sigma > M_n$: $M_n$ is reachable from $M_0$ by $\sigma$.

- $R(N, M_0)$: the set of all possible markings reachable from $M_0$ in a net $(N, M_0)$.

- or simply $R(M_0)$.

- $L(N, M_0)$: the set of all possible firing sequences from $M_0$ in a net $(N, M_0)$.

- or simply $L(M_0)$.

Boundedness
In many instances, tokens represent customers in a resource sharing system. For example, queue......

Clearly, allowing queues to grow to infinity is undesirable, since it means that customers wait forever to access a server. In classical system theory, a state variable that is allowed to grow to infinity is generally an indicator of instability in the system. Similarly here, unbounded growth in state components (markings) leads to some form of instability.

Boundedness refers to the property of a place to maintain a number of tokens that never exceeds a given positive integer.

**Definition 8** (boundedness, safe) A Petri net \((N, M_0)\) is said to be \(k\)-bounded or simply bounded if the number of tokens in each place does not exceed a finite number \(k\) for any marking reachable from \(M_0\), i.e., \(M(p_i) \leq k\) for every place \(p_i\) and every marking \(M \in R(M_0)\). A Petri net is said to be safe if it is 1-bounded.

Given a DEDS modeled as a Petri net, a boundedness problem consists of checking if the net is bounded and determining a bound. If boundedness is not satisfied, then our task may be to alter the model so as to ensure boundedness. If the Petri net is bounded, then it can be transformed into an equivalent finite-state automaton, allowing the application of the analysis techniques for finite-state automata, if so desired.

**Liveness**

The concept liveness is closely related to the complete absence of deadlocks in operating systems.

**Definition 9** (liveness) A Petri net \((N, M_0)\) is said to be live 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.

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.

A transition \( t \) in a Petri net \((N, M_0)\) is said to be:

0) **dead (L0-live)** if \( t \) can never be fired in any firing sequence in \( L(M_0) \)

1) **L1-live (potentially firable)** if \( t \) can be fired at least once in some firing sequence in \( L(M_0) \)

2) **L2-live** if given any positive integer \( k \), \( t \) can be fired at least \( k \) times in some firing sequence in \( L(M_0) \)

3) **L3-live** if \( t \) appears infinitely, often in some firing sequence in \( L(M_0) \)

4) **L4-live or live** if \( t \) is L1-live for every marking \( M \) in \( R(M_0) \)

It is easy to see the following implications:

\[
L_4\text{-liveness} \implies L_3\text{-liveness} \implies L_2\text{-liveness} \implies L_1\text{-liveness}.
\]

Insert figure 4.10

**Reversibility and Home State**

**Definition 10** (reversibility) A Petri net \((N, M_0)\) is said to be reversible if, for each marking \( M \) in \( R(M_0) \), \( M_0 \) is reachable from \( M \).

Thus, in a reversible net one can always get back to the initial marking or state. In many applications, it is not necessary to get back to the initial state as long as one can get back to some (home) state. Therefore, we relax the reversibility condition and define a home state.

A marking \( M' \) is said to be a home state if, for each marking \( M \) in \( R(M_0) \), \( M' \) is reachable from \( M \).

Note that boundedness, liveness, and reversibility are independent of each other. For example
Coverability

Definition 11 (coverability) A marking $M$ in a Petri net $(N, M_0)$ is said to be coverable if there exists a marking $M'$ in $R(M_0)$ such that $M'(p) \geq M(p)$ for each $p$ in the net.

Coverability is closely related to L1-liveness (potential firable). 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.

Persistence

Definition 12 (persistent) 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 persisteince is useful in the context of parallel program schemata and speed-independent asynchronous circuits.

Persistency is closely related to conflict-free nets, and a safe persistent net can be transformed into a marked graph by duplicating some transitions and places. Note that all marked graphs are persistent, but not all persistent nets are marked graphs. For example, ....fig.17(c)

Synchronic Distance

It is a metric closely related to a degree of mutual dependence between two events in a condition/event system.

Definition 13 (synchronic distance) The synchronic distance between two transitions $t_1$ and $t_2$ in a Petri net $(N, M_0)$ is defined by

$$d_{12} = \max_{\sigma} |\tilde{\sigma}(t_1) - \tilde{\sigma}(t_2)|$$
where \( \sigma \) is a firing sequence starting at any marking \( M \) in \( R(M_0) \) and \( \bar{\sigma}(t_i) \) is the number of times that transition \( t_i, i = 1, 2 \) fires in \( \sigma \).

For example, in Fig. 17(d) \( d_{12} = 1, d_{34} = 1, d_{13} = \infty \).

The synchronic distance represents a well-defined metric for condition/event nets and marked graphs. However, there are some difficulties when it is applied to more general class of Petri nets.

**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 \( \sigma \) is said to be unconditionally (globally) fair if it is finite or every transition in the net appears infinitely often in \( \sigma \). A Petri net is said to be an unconditionally fair net if every firing sequence \( \sigma \) from \( M \) in \( R(M_0) \) is unconditionally fair.

Relationships: every B-fair net is an unconditionally-fair net and every bounded unconditionally-fair net is a B-fair net. For example...

- Fig. 17(h) is a B-fair net as well as an unconditionally-fair net.

- Fig. 17(d) is neither a B-fair net nor an unconditionally-fair net, since \( t_3 \) and \( t_4 \) will not appear in an infinite firing sequence \( \sigma = t_2t_1t_2t_1 \cdots \).

- The unbounded net (fig. 17(c)) is an unconditionally fair net but not a B-fair net.
Analysis Methods:

1. The reachability tree
   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.

2. Incidence matrix and state equation

3. Simple reduction rules for analysis
   These two techniques are powerful but in many cases they are applicable only to special subclasses of Petri nets or special situations.

1.2.2 The coverability tree

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. Examples: simple reachability tree, or infinite reachability tree.

Insert Fig 4.11, fig4.12 (DES)

Notations:

1. Root node This is the first node of the tree, corresponding to the initial state of the given Petri net. For example, fig4.12, \([1,0,0,0]\)

2. Terminal node. This is any node from which no transition can fire. For example, fig 4.12, \([0,0,1,1]\)

3. Duplicate node. This is a node identical to a node already in the tree. For example, Fig 4.11, \([1,1,0]\)

4. Node dominance. Let \( M = [m(p_1), \ldots, m(p_n)] \) and \( M' = [m'(p_1), \ldots, m'(p_n)] \) be two markings, i.e., nodes in the coverability tree. We will say that "\( M \) dominates \( M' \)" denoted by \( M >_d M' \), if the following two conditions hold:
1. Modeling Concepts and Methodology—Petri Net Approach

(a) \( m(p_i) \geq m'(p_i) \), for all \( i = 1, \ldots, n \)

(b) \( m(p_i) > m'(p_i) \), for at least some \( i = 1, \ldots, n \) for example, Fig. 4.12

5. The symbol \( \omega \). This may be thought of as "infinity" in representing the marking of an unbounded place. We use \( \omega \) when we identify a node dominance relationship in the coverability tree. \( \omega \) has the properties that for each integer \( n, \omega > n, \omega \pm n = \omega \) and \( \omega \geq \omega \). As an example, in Fig. 4.12

Coverability Tree Construction Algorithm

**Step 1:** Initialize \( M = M_0 \) (initial marking)

**Step 2:** For each new node, \( M \), evaluate the transition function \( f(M, t_j) \) for all \( t_j \in T \):

**Step 2.1:** If \( f(M, t_j) \) is undefined for all \( t_j \in T \) (i.e., no transition is enabled at marking \( M \)), then \( M \) is a terminal node.

**Step 2.2:** If \( f(M, t_j) \) is defined for some \( t_j \in T \), create a new node \( M' = f(M, t_j) \).

**Step 2.2.1:** If \( m(p_i) = \omega \) for some \( p_i \), set \( m'(p_i) = \omega \)

**Step 2.2.2:** If there exists a node \( M'' \) in the path from the root node \( M_0 \) (included) to \( M \) such that \( M' >_d M'' \), set \( m'(p_i) = \omega \) for all \( p_i \) such that \( m'(p_i) > m''(p_i) \)

**Step 2.2.3:** Otherwise, set \( m'(p_i) = f(M, t_j) \)

**Step 3:** If all new nodes are either terminal or duplicate nodes, stop.

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

Step 1) Label the initial marking \( M_0 \) as the root and tag it "new".

Step 2) While "new" markings exist, do the following:

Step 2.1) Select a new marking \( M \).

Step 2.2) If \( M \) is identical to a marking on the path from the root to \( M \), then tag \( M \) "old" and go to another new marking.
1. Modeling Concepts and Methodology—Petri Net Approach

Step 2.3) If no transitions are enabled at $M$, tag $M$ ”dead-end”.

Step 2.4) While there exist enabled transitions at $M$, do the following for each enabled transition $t$ at $M$:

Step 2.4.1) Obtain the marking $M'$ that results from firing $t$ at $M$.

Step 2.4.2) On the path from the root to $M$ if there exists a marking $M'$ such that $M'(p) \geq M'(p)$ for each place $p$ and $M' \neq M'$, i.e., $M'$ is coverable, then replace $M'(p)$ by $\omega$ for each $p$ such that $M'(p) > M(p)$.

Step 2.4.3) Introduce $M'$ as a node, draw an arc with label $t$ from $M$ to $M'$, and tag $M'$ ”new”.

$\omega$ may be thought as ”infinity”.

Example

Insert Fig. 16, fig. 18 (murata)

For a bounded Petri net, the coverability tree is also called the reachability tree since it contains all possible markings.

Application of the Coverability Tree

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

**Boundedness, safety, and blocking problems**

1) A net $(N, M_0)$ is bounded and thus $R(M_0)$ is finite iff (if and only if) $\omega$ does not appear in any node labels in $T$.

2) A net $(N, M_0)$ is safe iff 0’s and 1’s appear in node labels in $T$.

3) A transition is dead iff it does not appear as an arc label in $T$.

**Coverability problems**

4) If $M$ is reachable from $M_0$, then there exists a node labeled $M'$ such that $M \leq M'$.

**Conservation problems**
Disadvantage (Limitation): an exhaustive method. 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, Fig 19(a) and (b) have the same coverability tree, yet, Fig 19(a) is a live net, while Fig 19(b) is not live.

Insert fig. 19 (murata)

1.2.3 Incidence matrix and state equation

State equation can be used to address problems such as state reachability and conservation. It provides an algebraic alternative to the graphical methodology based on the coverability tree, and it can be quite powerful in identifying structural properties that are mostly dependent on the topology of the Petri net graph captured in the incidence matrix \( A \).

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 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 written as

\[
A x = \Delta M
\]

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

\[
A y = 0
\]
Let $r$ be the rank of $A$, and partition $A$ in the following form:

$$A = \begin{bmatrix} m - r & r \\ A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

(1.6)

where $A_{12}$ is a nonsingular square matrix of order $r$. A set of $(m - r)$ linearly independent solution $y$ for (1.5) can be given as the $(m - r)$ rows of the following $(m - r) \times m$ matrix $B_f$:

$$B_f = [I_\mu : -A_{11}^T(A_{12}^T)^{-1}]$$

(1.7)

where $I_\mu$ is the identity matrix of order $\mu = m - r$. Note that $AB_f^T = 0$. That is, the vector space spanned by the row vectors of $B_f$. Now, the condition that $\Delta M$ is orthogonal to every solution for $Ay = 0$ is equivalent to the following condition:

$$B_f \Delta M = 0$$

(1.8)

Thus, if $M_d$ is reachable from $M_0$, then the firing counter vector $x$ must exist and (1.8) must hold. Therefore, we have the following necessary condition for reachability in an unrestricted Petri net.

**Theorem 1.1** If $M_d$ is reachable from $M_0$ in a Petri net $(N, M_0)$, then $B_f \Delta M = 0$, where $\Delta M = M_d - M_0$ and $B_f$ is given by 1.7.

**Corollary 1.1** In a Petri net $(N, M_0)$, a marking $M_d$ is not reachable from $M_0(\neq M_d)$ if their difference is a linear combination of the row vectors of $B_f$, that is

$$\Delta M = B_f^Tz$$

where $z$ is a nonzero $\mu \times 1$ column vector.
Example 1.9  For the Petri net shown in Fig 1.8, the state equation is illustrated below, where the transition $t_3$ fires to result in the marking $M_1 = (3 \ 0 \ 0 \ 2)^T$ from $M_0 = (2 \ 0 \ 1 \ 0)^T$:

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

The Incidence matrix $A$ is of rank 2 and can be partitioned in the form of (1.6), where

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

Thus, the matrix $B_f$ can be found by (1.7):

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

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

T-invariant and P-invariant

An integer solution $x$ of the homogeneous equation ($\Delta M = 0$ in (1.4))

$$A^T x = 0$$

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

Study structural properties with T-invariant and P-invariant:

Conservativeness: A Petri net $N$ is said to be (partially) conservative if there exists a positive integer $y(p)$ for every (some) place $p$ such that the weighted sum of tokens, $M^T y = M_0^T y = a$ constant, for every $M \in R(M_0)$ and for any fixed initial marking $M_0$. It is easy to see that
Theorem 1.2 A Petri net $N$ is (partially) conservative iff there exists an $m$-vector of positive (nonnegative) integers such that $Ay = 0, y \neq 0$. (a positive $P$-invariant.)

Proof.

$$M = M_0 + A^T x, \quad x \geq 0$$

Consider the inner product of $M$ and $y$

$$M^T y = M_0^T y + x^T Ay$$

$$M^T y = M_0^T y = \text{a constant } \iff Ay = 0, y \neq 0$$

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

Theorem 1.3 A Petri net $N$ is said to be (partially) consistent if there exists an $n$-vector of positive (nonnegative) integers such that $A^T x = 0, x \neq 0$. (a positive $T$-invariant.)

Proof. Suppose a Petri net is consistent. Then from (1.3) 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 $\bar{\sigma} = x$, can be repeated.

Theorem 1.4 An $m$-vector $y$ is an $P$-invariant iff $M^T y = M_0^T y$ for any fixed initial marking $M_0$ and any $M$ in $R(M_0)$.

Theorem 1.5 An $n$-vector $x \geq 0$ is a $T$-invariant iff there exists a marking $M_0$ and a firing sequence $\sigma$ from $M_0$ back to $M_0$ with its firing count vector $\bar{\sigma}$ equal to $x$. 
The set of places (transitions) corresponding to nonzero entries in an P-invariant \( y \geq 0 \) (T-invariant \( x \geq 0 \)) is called the support of an invariant and is denoted by \( \|y\| = \|x\| \). A support is said to be minimal if no proper nonempty subset of the support is also a support. An invariant (vector) \( y \) is said to be minimal if there is no other invariant \( y_1 \) such that \( y_1(p) \leq y(p) \) for all \( p \). Given a minimal support of an invariant, there is a unique minimal invariant corresponding to the minimal support. We call such an invariant a minimal support invariant. The set of all possible minimal-support invariants can serve as a generator of invariants. That is, any invariant can be written as a linear combination of minimal support invariants.

**Example 1.10** For the Petri net in fig. 1.9, \( x_1 = (1 \ 0 \ 1)^T \) \( x_2 = (0 \ 1 \ 1)^T \) are possible minimal-support T-invariants, where \( \|x_1\| = \{t_1, t_3\} \) and \( \|x_2\| = \{t_2, t_3\} \) are corresponding minimal supports. All other T-invariants such as \( x_3 = (1 \ 1 \ 2)^T \) \( x_4 = (2 \ 1 \ 3)^T \) can be expressed as linear combinations of \( x_1 \) and \( x_2 \). That is, \( x_3 = x_1 + x_2 \) and \( x_4 = x_2 + x_2 \). Note that there are many (non-unique) T-invariants such as \( x_3, x_4 \), etc., corresponding to a nonminimal support \( \{t_1, t_2, t_3\} \).

One easy way to find T-invariants in an example like this is to simulate all “firing sequence” which would reproduce a marking, using the concept of “negative or borrowed” tokens, if necessary.

### 1.2.4 Simple reduction rules or decomposition techniques

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.

The following 6 operations 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 Fig. (a)
2. Fusion of Series Transitions (FST) as Fig. (b)
3. Fusion of Parallel Places (FPP) as Fig. (c)
4. Fusion of Parallel Transitions (FPT) as Fig. (d)
5. Elimination of Self-loop Places (ESP) as Fig. (e)
6. Elimination of Self-loop Transitions (EST) as Fig. (f)

**Example 1.11** The net shown in Fig. 1.10 can be reduced to the one shown in Fig. 1.11 after firing \(t_2\) to remove the token in \(p_1\) and then fusing \(t_1\) and \(t_2\) into \(t_{12}\), and \(t_3, t_4\) into \(t_{34}\).

1.3 Abbreviations, extensions and particular structures

1.3.1 classifications

1.3.2 Modeling by Abbreviations

1.3.3 Modeling by Extensions

1.3.4 particular structures (subclasses)
1.4 Summary

- 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 characterized as being concurrent, asynchronous, distributed, parallel, non-deterministic, 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.

- 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.

1. Colored Petri nets:
   Very Brief Introduction to CP-nets
   History of Petri Nets
   CP-nets at University of Aarhus
   Why use CP-nets?
   Analysis of CP-nets
1. Modeling Concepts and Methodology—Petri Net Approach

Where to Read More about CP-nets
http://www.daimi.aau.dk/CPnets/intro/index.html

2. International Journal on Software Tools for Technology Transfer: Special section on Coloured Petri Nets
http://sttt.cs.uni-dortmund.de/contents.html

3. Examples of Industrial Use of CP-nets and Design/CPN
http://www.daimi.aau.dk/CPnets/intro/example_indu.html
FIGURE 1.3. Third example: two computers use a common memory
1. Modeling Concepts and Methodology—Petri Net Approach

\[ m_1 = [1, 0] \]
\[ m_2 = [2, 1] \]

FIGURE 1.4. two markings

FIGURE 1.5. Sequence of transition firings in a petri net
FIGURE 1.6. A Petri net (a state machine) representing the state diagram of a vending machine, where coin return transitions are omitted.
1. Modeling Concepts and Methodology—Petri Net Approach

FIGURE 1.7. Simple queue system and its Petri net model

FIGURE 1.8.
1. Modeling Concepts and Methodology—Petri Net Approach

FIGURE 1.9.

FIGURE 1.10.

FIGURE 1.11.
FIGURE 1.12. classification of Petri net models