In 1962 Carl Adam Petri introduced a family of graphs, the so-called Petri nets (PNs) [291]. PNs are

bipartite graphs populated with tokens that flow through the graph that are used to model the dynamic

rather than static behavior of systems (e.g., detecting synchronization anomalies).

A bipartite graph is one with two classes of nodes; arcs always connect a node in one class with

one or more nodes in the other class. In the case of Petri nets the two classes of nodes are places and

transitions; thus, the name place-transition (P/T) nets is often used for this class of bipartite graphs.

Arcs connect one place with one or more transitions or a transition with one or more places.

To model the dynamic behavior of systems, the places of a Petri net contain tokens. Firing of

transitions removes tokens from the input places of the transition and adds them to its output places

(see Figure 2.17).

Petri nets can model different activities in a distributed system. A transition may model the occurrence

of an event, the execution of a computational task, the transmission of a packet, a logic statement,

and so on. The input places of a transition model the pre-conditions of an event, the input data for the computational task, the presence of data in an input buffer, or the pre-conditions of a logic

statement. The output places of a transition model the post-conditions associated with an event, the

results of the computational task, the presence of data in an output buffer, or the conclusions of a logic

statement.

The distribution of a token in place of a PN at a given time is called the marking of the net and

reflects the state of the system being modeled. PNs are very powerful abstractions and can express both

concurrency and choice, as we can see in Figure 2.18.

Petri nets can model concurrent activities. For example, the net in Figure 2.18(a) models conflict

or choice; only one of the transitions t1 and t2 may fire, but not both. Two transitions are said to be

concurrent if they are causally independent. Concurrent transitions may fire before, after, or in parallel

with each other; examples of concurrent transitions are t1 and t3 in Figure 2.18(b) and (c).

When choice and concurrency are mixed, we end up with a situation called confusion. Symmetric

confusion means that two or more transitions are concurrent and, at the same time, they are in conflict

with another one. For example, transitions t1 and t3 in Figure 2.18(b) are concurrent and, at the same

time, they are in conflict with t2. If t2 fires, either one or both of them will be disabled. Asymmetric

confusion occurs when a transition t1 is concurrent with another transition t3 and will be in conflict with

t2 if t3 fires before t1, as shown in Figure 2.18(c).

The concurrent transitions t2 and t3 in Figure 2.19(a) model concurrent execution of two processes.

A marked graph can model concurrency but not choice; transitions t2 and t3 in Figure 2.19(b) are concurrent,

so there is no causal relationship between them. Transition t4 and its input places p3 and p4 in

Figure 2.19(b) model synchronization; t4 can only fire if the conditions associated with p3 and p4 are

satisfied.

Petri nets can be used to model priorities. The net in Figure 2.19(c) models a system with two

processes modeled by transitions t1 and t2; the process modeled by t2 has a higher priority than the

one modeled by t1. If both processes are ready to run, places p1 and p2 hold tokens. When the two

processes are ready, transition t2 will fire first, modeling the activation of the second process. Only after

t2 is activated will transition t1 fire, modeling the activation of the first process.

Petri nets are able to model exclusion. For example, the net in Figure 2.19(d), models a group of n

concurrent processes in a shared-memory environment.At any given time only one processmay write,

but any subset of the n processes may read at the same time, provided that no process writes. Place p3

models the process allowed to write, p4 the ones allowed to read, p2 the ones ready to access the

shared memory, and p1 the running tasks. Transition t2 models the initialization/selection of the process

allowed to write and t1 of the processes allowed to read, whereas t3 models the completion of a

write and t4 the completion of a read. Indeed, p3 may have at most one token, whereas p4 may have

at most n. If all n processes are ready to access the shared memory, all n tokens in p2 are consumed when

transition t1 fires.However, place p4 may contain n tokens obtained by successive firings of transition t2.

After this informal discussion of Petri nets we switch to a more formal presentation and give several

definitions.

Labeled Petri Net. A tuple N = (p, t, f , l) such that:

• p ⊆ U is a finite set of places,

• t ⊆ U is a finite set of transitions,

• f ⊆ (p   t) ∪ (t   p) is a set of directed arcs, called flow relations, and

• l : t → L is a labeling or a weight function,

withU a universe of identifiers and L a set of labels. The weight function describes the number of tokens

necessary to enable a transition. Labeled PNs describe a static structure; places may contain tokens,

and the distribution of tokens over places defines the state, or the markings of the PN. The dynamic

behavior of a PN is described by the structure together with the markings of the net.

Marked Petri Net. A pair (N, s) where N = (p, t, f , l) is a labeled PN and s is a bag7 over p

denoting the markings of the net.

Preset and Postset of Transitions and Places. The preset of transition ti denoted as •ti is the set of

input places of ti , and the postset denoted by ti• is the set of the output places of ti . The preset of place

p j denoted as •p j is the set of input transitions of p j , and the postset denoted by p j• is the set of the

output transitions of p j .

Figure 2.17(a) shows a PN with three places, p1, p2, and p3, and one transition, t1. The weights of

the arcs from p1 and p2 to t1 are two and one, respectively; the weight of the arc from t1 to p3 is three.

The preset of transition t1 in Figure 2.17(a) consists of two places, •t1 = {p1, p2}, and its postset

consist of only one place, t1• = {p3}. The preset of place p4 in Figure 2.19(a) consists of transitions t3

and t4, •p4 = {t3, t4}, and the postset of p1 is p1• = {t1, t2}.

Ordinary Net. A PN is ordinary if the weights of all arcs are 1.

The nets in Figure 2.19 are ordinary nets since, the weights of all arcs are 1.

Enabled Transition. A transition ti ∈ t of the ordinary PN (N, s), with s the initial marking of N,

is enabled if and only if each of its input places contain a token, (N, s)[ti >⇔ •ti ∈ s. The notation

(N, s)[ti > means that ti is enabled.

Themarking of a PNchanges as a result of transition firing; a transitionmust be enabled in order to fire.

Firing Rule. The firing of the transition ti of the ordinary net (N, s) means that a token is removed

from each of its input places and one token is added to each of its output places, so its marking changes

s  → (s − •ti + ti • ). Thus, firing of transition ti changes a marked net (N, s) into another marked

net (N, s − •ti + ti • ).

Firing Sequence. A nonempty sequence of transitions σ ∈ t∗ of the marked net (N, s0) with N =

(p, t, f , l) is called a firing sequence if and only if there exist markings s1, s2, . . . , sn ∈ B(p) and

transitions t1, t2, . . . , tn ∈ t such that σ = t1, t2, . . . , tn and for i ∈ (0, n), (N, si )ti+1 > and si+1 =

si − •ti + ti•. All firing sequences that can be initiated from marking s0 are denoted as σ(s0).

Reachability. The problem of finding whether marking sn is reachable from the initialmarking s0, sn ∈

σ(s0).Reachability is a fundamental concern for dynamic systems; the reachability problem is decidable,

but reachability algorithms require exponential time and space.

Liveness. Amarked Petri net (N, s0) is said to be live if it is possible to fire any transition starting from

the initialmarking, s0. The absence of deadlock in a system is guaranteed by the liveness of its net model.

Incidence Matrix. Given a Petri net with n transitions andm places, the incidence matrix F = [ fi , j ] is

an integer matrix with fi , j = w(i , j )−w( j , i ). Here w(i , j ) is theweight of the flowrelation (arc) from

transition ti to its output place p j , and w( j , i ) is theweight of the arc from the input place p j to transition

ti . In this expression w(i , j ) represents the number of tokens added to the output place p j and w( j , i ) the

ones removed from the input place p j when transition ti fires. FT is the transpose of the incidence matrix.

A marking sk can be written as an m   1 column vector, and its j -th entry denotes the number of

tokens in place j after some transition firing. The necessary and sufficient condition for transition ti to

be enabled at a marking s is that w( j , i )   s( j ) ∀s j ∈ •ti , the weight of the arc from every input place

of the transition, be smaller or equal to the number of tokens in the corresponding input place.

Extended Nets. PNs with inhibitor arcs; an inhibitor arc prevents the enabling of a transition. For

example, the arc from p2 to t1 in the net in Figure 2.19(a) is an inhibitor arc; the process modeled by

transition t1 can be activated only after the process modeled by transition t2 is activated.

Modified Transition Enabling Rule for Extended Nets. A transition is not enabled if one of the

places in its preset is connected with the transition with an inhibitor arc and if the place holds a token.

For example, transition t1 in the net in Figure 2.19 (c) is not enabled while place p2 holds a token.

Based on their structural properties, Petri nets can be partitioned in several classes:

• State machines are used to model finite state machines and cannot model concurrency and synchronization.

• Marked graphs cannot model choice and conflict.

• Free-choice nets cannot model confusion.

• Extended free-choice nets cannot model confusion but they do allow inhibitor arcs.

• Asymmetric choice nets can model asymmetric confusion but not symmetric ones.

This partitioning is based on the number of input and output flow relations from/to a transition or

a place and by the manner in which transitions share input places. The relationships between different

classes of Petri nets are illustrated in Figure 2.20.

State Machine. A Petri net is a state machine if and only if ∀ti ∈ t then (| • ti| = 1 ∧ |ti • | = 1).

All transitions of a state machine have exactly one incoming and one outgoing arc. This topological

constraint limits the expressiveness of a state machine, so no concurrency is possible. For example, the

transitions t1, t2, t3, and t4 of the state machine in Figure 2.19(a) have only one input and one output arc,

so the cardinality of their presets and postsets is one. No concurrency is possible; once a choicewasmade

by firing either t1, or t2, the evolution of the system is entirely determined. This state machine has four

places p1, p2, p3, and p4 and themarking is a 4-tuple (p1, p2, p3, p4); the possiblemarkings of this net

are (1, 0, 0, 0), (0, 1, 0, 0), (0, 0, 1, 0), (0, 0, 0, 1),with a token in places p1, p2, p3, or p4, respectively.

Marked Graph. A Petri net is a marked graph if and only if ∀pi ∈ p then (|• pi| = 1∧|pi •| = 1). In

a marked graph each place has only one incoming and one outgoing flow relation; thus, marked graphs

do no not allow modeling of choice.

Free Choice, Extended Free Choice, and Asymmetric Choice Petri Nets. The marked net, (N, s0)

with N = (p, t, f , l) is a free-choice net if and only if

( • ti ) ∩ ( • t j ) = ∅ ⇒ |• ti| = | • t j| ∀ti , j ∈ t. (2.38)

N is an extended free-choice net if ∀ti , t j ∈ t then ( • ti ) ∩ ( • t j ) = ∅ ⇒ •ti = •t j .

N is an asymmetric choice net if and only if (•ti )∩(•t j )  = ∅⇒(•ti ⊆ •t j ) or (•ti ⊇ •t j ), ∀ti , t j ∈ t.

In an extended free-choice net, if two transitions share an input place they must share all places in

their presets. In an asymmetric choice net, two transitions may share only a subset of their input places.

Several extensions of Petri nets have been proposed. For example, colored Petri nets (CPSs) allow

tokens of different colors, thus increasing the expressivity of the PNs but not simplifying their analysis.

Several extensions of Petri nets to support performance analysis by associating a random time with each

transition have been proposed. In case of stochastic Petri nets (SPNs), a random time elapses between

the time a transition is enabled and the moment it fires. This random time allows the model to capture

the service time associated with the activity modeled by the transition.

Applications of stochastic Petri nets to performance analysis of complex systems is generally limited

by the explosion of the state space of the models. Stochastic high-level Petri nets (SHLPNs) were

introduced in 1988 [219]; they allow easy identification of classes of equivalent markings even when the

corresponding aggregation of states in the Markov domain is not obvious. This aggregation could reduce

the size of the state space by one or more orders of magnitude, depending on the system being modeled.