INTERNATIONAL STANDARD

ISO/IEC 15909-1

Second edition
2019-08

Systems and software engineering — High-level Petri nets —

Part 1: Concepts, definitions and graphical notation

Ingénierie du logiciel et des systèmes — Réseaux de Petri de haut niveau —

Partie 1: Concepts, définitions et notation graphique

Reference number
ISO/IEC 15909-1:2019(E)

Contents

Page

Foreword ..... **v**

Introduction ..... **vi**

1 Scope **1**

2 Normative references **1**

3 Terms and definitions **1**

4 Conformance **5**
- 4.1 General 5
- 4.2 Mandatory conformance: common concepts for Petri nets 5
- 4.3 Place/Transition nets 5
- 4.4 Symmetric nets 5
- 4.5 High-level Petri nets 5
  - 4.5.1 Level 1 5
  - 4.5.2 Level 2 5
- 4.6 Petri nets with priorities 5
- 4.7 Petri nets with time — Level 1 6

5 Common concepts for Petri nets **6**
- 5.1 General 6
- 5.2 Definition 6
  - 5.2.1 Concept 1 (net) 6
  - 5.2.2 Concept 2 (marking of a net) 6
  - 5.2.3 Concept 3 (marked net) 6
  - 5.2.4 Concept 4 (precondition of a transition) 6
  - 5.2.5 Concept 5 (postcondition of a transition) 6
  - 5.2.6 Notation 1 (precondition and postcondition of a transition) 7
- 5.3 Behavioral semantics 7
  - 5.3.1 General 7
  - 5.3.2 Concept 6 (net enabling rule) 7
  - 5.3.3 Concept 7 (net firing rule) 7

6 Place/Transition nets **8**
- 6.1 General 8
- 6.2 Definition — Concept 8 (place/transition net) 8
- 6.3 Behavioral semantics 8
  - 6.3.1 General 8
  - 6.3.2 Concept 9 (marking of a P/T net) 8
  - 6.3.3 Concept 10 (P/T net enabling rule) 9
  - 6.3.4 Concept 11 (enabling function of enabled transitions) 9
  - 6.3.5 Concept 12 (P/T net firing rule) 9

7 Symmetric nets **9**
- 7.1 Definition 9
  - 7.1.1 Concept 13 (color class) 9
  - 7.1.2 Concept 14 (color domain) 9
  - 7.1.3 Concept 15 (symmetric net) 9
  - 7.1.4 Concept 16 (basic color functions) 10
  - 7.1.5 Concept 17 (class color functions) 10
- 7.2 Behavioral semantics 11
  - 7.2.1 Concept 18 (marking of a symmetric net) 11
  - 7.2.2 Concept 19 (symmetric net enabling rule) 11
  - 7.2.3 Concept 20 (symmetric net firing rule) 11

8 High-level Petri nets **11**
- 8.1 General 11
- 8.2 Definition — Concept 21 (high-level Petri net) 11

8.3 Behavioral semantics 12
8.3.1 Concept 22 (marking of a high-level Petri net) 12
8.3.2 Concept 23 (high-level Petri net enabling rule) 12
8.3.3 Concept 24 (high-level Petri net firing rule) 12

9 Petri nets with priorities 12
9.1 General 12
9.2 Definition 12
9.2.1 Concept 25 (dynamic prioritized Petri net) 12
9.2.2 Concept 26 (statically prioritized Petri net) 13
9.3 Behavioral semantics 13
9.3.1 Concept 27 (marking of a prioritized Petri net) 13
9.3.2 Concept 28 (prioritized Petri net enabling rule) 13

10 Time Petri nets 14
10.1 General 14
10.2 Definition 14
10.2.1 Concept 29 (generic time Petri net) 14
10.2.2 Concept 30 (time Petri net) 14
10.3 Behavioral semantics 14
10.3.1 General 14
10.3.2 Concept 31 (time Petri net enabling rule) 15
10.3.3 Concept 32 (time Petri net firing rule) 15

Annex A (normative) **Mathematical conventions 16**

Annex B (informative) **Guidelines for graphical notations 20**

Bibliography 29

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*.

This second edition cancels and replaces the first edition (ISO/IEC 15909-1:2004), which has been technically revised.

The main change compared to the previous edition is as follows:

— a complete redrafting of the concepts and definitions of Petri nets and Petri net types in a simpler, modular and incremental way.

A list of all parts in the ISO/IEC 15909 series can be found on the ISO website.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The ISO/IEC 15909 series is concerned with defining a modeling language and its transfer format, known as high-level Petri nets. This document is Part 1 of the ISO/IEC 15909 series. This document provides the mathematical definition of high-level Petri nets, called the semantic model, its execution semantics, the graphical form of the technique and its mapping to the semantic model. This document also introduces some common notational conventions for the graphical form of high-level Petri nets.

Petri nets have been used to describe a wide range of systems since their invention in 1962. The technique is mathematically defined and can thus be used to provide unambiguous specifications and descriptions of applications. It is also an executable technique, allowing specification prototypes to be developed to test ideas at the earliest and cheapest opportunity. Specifications written in the technique can be subjected to analysis methods to prove properties about the specifications, before implementation commences, thus saving on testing and maintenance time and providing a high level of quality assurance.

A problem with Petri nets is the explosion of the number of elements in their graphical form when they are used to describe complex systems. High-level Petri nets were developed to overcome this problem by introducing higher-level concepts, such as the use of complex structured data as tokens, and using algebraic expressions to annotate net elements. The use of "high-level" to describe these Petri nets is analogous to the use of "high-level" in high-level programming languages (as opposed to assembly languages), and is the usual term used in the Petri net community. Two of the early forms of high-level nets that this document builds on are predicate-transition nets and coloured Petri nets, first introduced in 1979 and developed during the 1980s. It also uses some of the notions developed for algebraic Petri nets, first introduced in the mid-1980s. It is believed that this document captures the spirit of these earlier developments (see Bibliography).

The technique has multiple uses. For example, it can be used directly to specify systems or to define the semantics of other less formal languages. It can also serve to integrate techniques currently used independently such as state-transition diagrams and data flow diagrams. The technique is particularly suited to parallel and distributed systems development as it supports concurrency. The technique is able to specify systems at a level that is independent of the choice of implementation (i.e. by software, hardware (electronic and/or mechanical) or humans or a combination). This document may be cited in contracts for the development of systems (particularly distributed systems) or used by application developers or Petri net tool vendors or users.

This document provides an abstract mathematical syntax and a formal semantics for the technique. Conformance to the document is possible at several levels. The level of conformance depends on the class of high-level net chosen.

This document is Part 1 of the ISO/IEC 15909 series. It describes definitions, semantics, execution and graphical notations for high-level Petri nets. A transfer format for the high-level Petri nets is the subject of Part 2, while Part 3 addresses techniques for enrichments, extensions and structuring mechanisms.

Reliable software development requires powerful mathematical models and tools. The usability of Petri nets has been proven for non-trivial industrial applications.

This document is written as a reference for systems analysts, designers, developers, maintainers and procurers, and for Petri net tool designers and developers.

This document defines high-level Petri nets showing common concepts for Petri nets first, and then describing several typical types of Petri nets, such as place/transition nets, symmetric nets, and Petri nets with time. Each of the Petri net types is described with its definition, semantics, and execution. Their graphical notations are provided in Annex B.

Systems and software engineering — High-level Petri nets —

Part 1: Concepts, definitions and graphical notation

1 Scope

This document defines a Petri net modeling language or technique, called high-level Petri nets, including its syntax and semantics. It provides a reference definition that can be used both within and between organizations, to ensure a common understanding of the technique and of the specifications written using the technique. This document also facilitates the development and interoperability of Petri net computer support tools.

This document is applicable to a wide variety of concurrent discrete event systems and in particular distributed systems. Generic fields of application include:

— requirements analysis;

— development of specifications, designs and test suites;

— descriptions of existing systems prior to re-engineering;

— modeling business and software processes;

— providing the semantics for concurrent languages;

— simulation of systems to increase confidence;

— formal analysis of the behavior of systems;

— and development of Petri net support tools.

This document can be applied to the design of a broad range of systems and processes, including aerospace, air traffic control, avionics, banking, biological and chemical processes, business processes, communication protocols, computer hardware architectures, control systems, databases, defense command and control systems, distributed computing, electronic commerce, fault-tolerant systems, games, hospital procedures, information systems, Internet protocols and applications, legal processes, logistics, manufacturing systems, metabolic processes, music, nuclear power systems, operating systems, transport systems (including railway control), security systems, telecommunications and workflows.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 15909-2, *Systems and software engineering — High-level Petri nets — Part 2: Transfer format*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 15909-2 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at http://www.electropedia.org/

3.1
arc
directed edge of a net which may connect a *place* (3.28) to a *transition* (3.37) or a transition to a place, normally represented by an arrow

3.2
arc annotation
expression that may involve constants, variables and *operators* (3.21) used to annotate an *arc* (3.1) of a net

Note 1 to entry: The expression shall evaluate to a *multiset* (3.17) over the type of the arc's associated *place* (3.28).

3.3
basis set
set of objects used to create a *multiset* (3.17)

3.4
carrier
set of a *many-sorted algebra* (3.14)

3.5
concurrent enabling
<transition modes> state, for a *multiset* (3.17) of *transition* (3.37) modes, when all the involved *input places* (3.13) contain enough tokens to satisfy the sum of all of the demands imposed on them by each *input arc* (3.12) *annotation* (3.2) evaluated for each transition mode in the multiset

3.6
concurrency
property of a system in which events may occur independently of each other, and hence are not ordered

Note 1 to entry: See also step and *concurrent enabling* (3.5).

3.7
declaration
set of statements which define the sets, constants, *parameter* (3.24) values, typed variables and functions required for defining the annotations on a *high-level Petri net* (3.9)

3.8
enabling
<a transition> state of a *transition* (3.37) in a particular mode and net *marking* (3.15) when the following conditions are met: the marking of each *input place* (3.13) of the transition satisfies the demand imposed on it by its *arc annotation* (3.2) evaluated for the particular transition mode; the demand is satisfied when the place's marking contains (at least) the *multiset* (3.17) of tokens indicated by the evaluated arc annotation; the determination of transition modes guarantees that the Transition Condition is satisfied

Note 1 to entry: See *concurrent enabling* (3.5).

3.9
high-level Petri net
high-level net
HLPN
algebraic structure comprising: a set of *places* (3.28); a set of *transitions* (3.37); a set of types; a function associating a type to each place, and a set of modes (a type) to each transition; a pre-function imposing token demands (*multisets* (3.17) of tokens) on places for each transition mode; a post-function determining output tokens (multisets of tokens) for places for each transition mode; and an *initial marking* (3.10)

3.10
initial marking
<a net> set of initial making of places (3.11) given with the *high-level net* (3.9) definition

3.11
initial marking of place
special *marking* (3.15) of a *place* (3.28), defined with the net

3.12
input arc
<a transition>arc directed from a *place* (3.28) to the *transition* (3.37)

3.13
input place
<a transition> *place* (3.28) connected to the *transition* (3.37) by an *input arc* (3.12)

3.14
many-sorted algebra
mathematical structure comprising a set of sets and a set of functions taking these sets as domains and co-domains

3.15
marking
<a net> set of the *place* (3.28) markings for all places of the net.

3.16
marking of a place
multiset (3.17) of tokens associated with ('residing in') the *place* (3.28)

3.17
multiset
bag
collection of items where repetition of items is allowed

3.18
multiplicity
natural number (i.e. non-negative integer) which describes the number of repetitions of an item in a *multiset* (3.17)

3.19
multiset cardinality
cardinality of a multiset
sum of the multiplicities of each of the members of the *multiset* (3.17)

3.20
node
vertex of a net graph, i.e. either a *place* (3.28) or a *transition* (3.37)

3.21
operator
symbol representing the name of a function

3.22
output arc
<a transition> arc directed from the *transition* (3.37) to a *place* (3.28)

3.23
output place
<a transition> *place* (3.28) connected to the *transition* (3.37) by an *output arc* (3.22)

3.24
parameter
symbol that can take a range of values defined by a set

Note 1 to entry: It is defined as a constant in the *signature* (3.32).

3.25
Petri net
net
algebraic structure with two sets, one called *places* (3.28) and the other called *transitions* (3.37), together with their associated relations and functions, and named after their inventor, Carl Adam Petri

3.26
Petri net with priorities
Petri net (3.25) having priorities which can be used for selecting the enabled *transitions* (3.37) according to the given priority scheme

Note 1 to entry: The firing rule is the same as a Petri net without priorities.

3.27
Petri net with time
Petri net (3.25) having timing constraints associated with the *nodes* (3.20) or arcs

Note 1 to entry: These constraints affect the *enabling* (3.8) and firing rules.

3.28
place
node (3.20) of a net, usually represented by an ellipse

3.29
place/transition net
P/T net
Petri net (3.25) comprising a net with positive integers associated with arcs and an *initial marking* (3.10) function which associates a natural number of simple tokens ('black dots') with *places* (3.28)

3.30
reachability graph
directed graph where the *nodes* (3.20) correspond to *reachable markings* (3.31) and the edges to *transition* (3.37) occurrences

3.31
reachable marking
marking (3.15) of the net that can be reached from the *initial marking* (3.10) by the occurrence of *transitions* (3.37)

3.32
signature
mathematical structure comprising a set of *sorts* (3.33) and a set of *operators* (3.21)

3.33
sort
symbol representing the name of a set

3.34
symmetric net
high-level Petri net (3.9) where types are finite and only built-in operations are allowed

3.35
term
expression comprising constants, variables and *operators* (3.21) built from a *signature* (3.32) and a set of sorted variables

3.36
time Petri net
Petri net (3.25) with time where timing constraints are associated with *transitions* (3.37)

3.37
transition
node (3.20) of a net, usually represented by a rectangle

4 Conformance

4.1 General

There are different levels of conformance to this document. The level of conformance is defined according to the net type.

4.2 Mandatory conformance: common concepts for Petri nets

Prior to claim conformance to any type of Petri net, defined in this document, an implementation shall demonstrate that it has the semantics and the execution mechanisms defined in Clause 5.

4.3 Place/Transition nets

To claim P/T net Level conformance to this document an implementation shall demonstrate that it has the semantics and the execution defined in Clause 6, by providing a mapping from the implementation to the semantic model and the execution.

4.4 Symmetric nets

To claim Symmetric net Level conformance to this document an implementation shall demonstrate that it has the semantics and the execution defined in Clause 7, by providing a mapping from the implementation to the semantic model and the execution.

4.5 High-level Petri nets

4.5.1 Level 1

To claim high-level Petri net Level 1 conformance to this document an implementation shall demonstrate that it has the semantics and the execution defined in Clause 7 (Symmetric nets), by providing a mapping from the implementation to the semantic model and the execution.

4.5.2 Level 2

To claim high-level Petri net Level 2 conformance to this document an implementation shall have satisfied the requirements of high-level Petri net Level 1 conformance, and in addition shall comply with the definition and semantics defined in Clause 8.

4.6 Petri nets with priorities

To claim Petri nets with priorities conformance to this document an implementation shall demonstrate that it has the semantics, the execution, and the prioritized Petri net model defined in Clause 9, by providing a mapping from the implementation to the semantic model and the execution.

4.7 Petri nets with time — Level 1

To claim Petri nets with time Level 1 conformance to this document an implementation shall demonstrate that it has the semantics, the execution, and the time Petri net model defined in Clause 10, by providing a mapping from the implementation to the semantic model and the execution.

IMPORTANT — This document only defines a single type of Petri nets with time among the many variants available. It corresponds to the semantics defined by Merlin in 1974, which is the most commonly used. Other types of Petri nets with time can still be defined using tool-specific information.

5 Common concepts for Petri nets

5.1 General

All common concepts for Petri nets in this Clause, and Petri net types in subsequent Clauses 6 to 10, shall be defined using the mathematical conventions in Annex A.

5.2 Definition

5.2.1 Concept 1 (net)

A net is defined by a triple $N = \langle P, T, F\rangle$, where:

— P is a set of elements called places,

— T is a set of elements called transitions, disjoint from P,

— F is a flow relation $F \subseteq (P \times T) \cup (T \times P)$.

If P and T are finite, then N is said to be finite.

5.2.2 Concept 2 (marking of a net)

A marking of a net N is a mapping $M : P \rightarrow MS$, where MS is a multiset (see A.5). A place p is said to be marked by a marking M if $M(p)$ is not an empty multiset.

5.2.3 Concept 3 (marked net)

A marked net is a tuple $N = \langle P, T, F, M\rangle$, where $\langle P, T, F\rangle$ is a net and M a marking of this net.

By convention, the initial marking of a marked net N is noted M_0.

By convention, the set of possible marked nets derived from N is noted N.

5.2.4 Concept 4 (precondition of a transition)

A place $p \in P$ is a precondition of the transition $t \in T$ iff $(p,t) \in F$.

5.2.5 Concept 5 (postcondition of a transition)

A place $p \in P$ is a postcondition of the transition $t \in T$ iff $(t,p) \in F$.

5.2.6 Notation 1 (precondition and postcondition of a transition)

Let •*t* and *t*• be respectively:

— the subset of places which constitute the precondition of a transition $t \in T$ (also called preset),

— the subset of places which constitute the postcondition of a transition $t \in T$ (also called postset).

Let us assume:

— A net consists of two different kinds of sets *P* and *T* corresponding respectively to the set of places and the set of transitions.

— The flow relation *F* defines arcs connecting places to transitions, and transitions to places. There is an arc from a place $p \in P$ to a transition $t \in T$ iff *p* is a precondition of *t* and there is an arc from a transition $t \in T$ to a place $p \in P$ iff *p* is a postcondition of *t*.

— The marking of a place *M(p)* is a collection of items associated with the place *p*. The items associated with places are called tokens.

— A net marking is the vector of all place markings (assuming an arbitrary order on the places).

5.3 Behavioral semantics

5.3.1 General

A Petri net has a strong behavioral semantics allowing to "execute" unambiguously the specification, thus producing a so-called reachability graph, or marking graph, that represents all the possible configurations the modeled system can have. This reachability graph can be infinite or finite, depending on the characteristics of the model (boundedness). Such an execution is useful to analyze the behavior of a system; this technique is called Model Checking[6].

The behavioral semantics of a Petri net relies on two notions: (i) the enabling rule, and (ii) the firing rule. This subclause defines them in a generic way. Clauses 6 to 10 will instantiate these notions for the types of Petri net defined by this document.

The common enabling rule in the context of interleaving semantics for all net types is defined as follows.

5.3.2 Concept 6 (net enabling rule)

Let $E: N \times T \rightarrow B$ be the enabling function of a net, which has two parameters: a net within a marked net type $N = \langle P, T, F, M \rangle \in N$ and the transition $t \in T$ to which it applies. *M* of $N \in N$ denotes the current marking of *N*. *E* is defined as follows:

$$E(N \in N, t \in T): \begin{cases} True & when\, t\, is\ firable \\ False & otherwise \end{cases}$$

5.3.3 Concept 7 (net firing rule)

The firing rule is defined as computed by the following steps:

1. Computation of *T'*, the set of enabled transitions. *T'* is the subset of *T* for which $E(N \in N, t_i \in T) = True$.

2. Selection of an action, which can be either *t* in *T'*, or some action *A*-$_{pre}$ (such as the elapse of time as in the case of time Petri nets).

3. Update of the state of net N, and, possibly some action $A\text{-}_{post}$ (such as a reset arc emptying a place from its tokens).

By default, $A\text{-}_{pre}$ and $A\text{-}_{post}$ are the identity function. They are placeholders to host additional actions required to capture the semantics of subsequently defined Petri net types (e.g. in time Petri nets in Clause 10}, $A\text{-}_{pre}$ is the elapse of time).

When not otherwise stated, in the subsequent Petri net types definitions, these functions are empty.

For each particular type of net, the appropriate action A- will be defined, as well as the corresponding state update, in the appropriate section.

6 Place/Transition nets

6.1 General

Place/Transition nets are the most widely-used type of Petri nets. This clause provides their formal definition, semantics and execution.

6.2 Definition — Concept 8 (place/transition net)

A place/transition net (P/T net) is defined by a tuple $PT = <N, W, M_0>$, where:

— $N = <P, T, F>$ is a net,

— W is the arc annotation function $W : F \rightarrow \mathbb{N}$,

— M_0 is a vector in $\mathbb{N}^{|P|}$ defining the initial number of tokens in places (initial marking).

From these definitions, the enabling and firing rules for P/T nets are defined as follows.

6.3 Behavioral semantics

6.3.1 General

A transition is enabled with respect to a net marking. A net marking comprises the set of all place markings of the net.

Enabling a transition involves the marking of its input places. An input place of a transition is a place which is connected to the transition by an arc leading from that place to the transition. An arc that leads from an input place to a transition is called an input arc of the transition. If each input place's marking contains at least its input arc's weight of tokens, then the transition is enabled.

Two transitions can be concurrently enabled for a particular marking, if for the associated transitions, each input place's marking contains at least the sum of the enabling tokens of each input arc associated with that input place.

Enabled transitions can be fired (or occurs). When a transition is fired, tokens are removed from its input places, and tokens are added to its output places. An output place of a transition is a place which is connected to the transition by an arc directed from the transition to the place. An arc that leads from a transition to a place (an output place of the transition) is called an output arc of the transition.

A place may be both an input place and an output place of the same transition.

6.3.2 Concept 9 (marking of a P/T net)

A marking M of a P/T net is a vector $M \in \mathbb{N}^{|P|}$ such that $M(p)$ is the marking of place p. A marked P/T net is a tuple $PT = <N, W, M>$, where M denotes a marking of the net.

6.3.3 Concept 10 (P/T net enabling rule)

A transition $t \in T$ is enabled in marking *M*, denoted by , $M[t>$ iff $\forall p \in \bullet t, M(p) \geq W(p,t)$.

From the definition above, the enabling function for P/T nets is defined as follows.

6.3.4 Concept 11 (enabling function of enabled transitions)

$$E_{pt}(N \in N, t \in T): \begin{cases} True & iff \forall p \in \bullet t, M(p) \geq W(p,t) \\ False & otherwise \end{cases}$$

6.3.5 Concept 12 (P/T net firing rule)

If a transition $t \in T$ is enabled in marking *M*, it can fire leading to a successor marking *M'*, denoted by $M[t > M'$, where: $\forall p \in P, M'(p) = M(p) - W(p,t) + W(t,p)$. This corresponds to the update of the state of the net.

On the occurrence of a transition, the following actions occur atomically:

a) for each input place of the transition: the enabling tokens of the input arc are subtracted from the input place's marking, and

b) for each output place of the transition: tokens are added to the marking of the output place according to the weight of the output arc.

7 Symmetric nets

7.1 Definition

7.1.1 Concept 13 (color class)

Let *C* be a non-empty finite set. A non-empty finite color class *Cl* over *C* defines a type over *C*. *C* is the carrier set of *Cl*. The elements of a color class *Cl* may be linearly ordered, circular or unordered.

Let us remark:

— The order on the carrier set *C* of *Cl* is a total order.

— *Cl* being circular subsumes *Cl* being linearly ordered, with the additional successor function defined below.

7.1.2 Concept 14 (color domain)

A color domain *D* is a finite cartesian product of *n* color classes: $D = \prod_{i=1}^{n} Cl_i$.

7.1.3 Concept 15 (symmetric net)

A symmetric net (SN) is a tuple $SN = \langle N, W, Cl, C, \Phi \rangle$ where:

— *N* is a net,

— $Cl = \{C_1, \ldots, C_k\}$ is a finite set of finite classes, each being partitioned into s_i static sub-classes ($C_i = \uplus_{q=1..s_i} C_{i,q}$),

— C is a mapping which defines for each place and each transition its color domain,

— W is the weight function which associates with each pair $(p,t) \in P \times T$ and each pair $(t,p) \in T \times P$ a general color function defined from *C(t)* to $Bag(C(p))$,

— Φ is a mapping that associates a guard with each transition.

In symmetric nets, arcs are labelled by color functions which select tokens in adjacent places depending on the instantiation performed for the firing.

The simplest color functions are the projections, denoted $X_{C_i}^j, i \in 1..k, j \in 1..e_i$, that select one component of a color, the successor functions, denoted $X_{C_i}^j ++, i \in 1..k, j \in 1..e_i$, that select the successor of a component of a color and the "global" selections $C_i.all = \sum_{c \in C_i} c$ that map any color to the "sum" of colors in class C_i.

7.1.4 Concept 16 (basic color functions)

Let C_i be a color class and $D = C_1^{e_1} \times \ldots \times C_k^{e_k}$ a color domain. The basic color functions are defined from D to $Bag(C_i)$ by for all $c = \left\langle c_1^1, \ldots, c_1^{e_1}, \ldots, c_k^1, \ldots, c_k^{e_k} \right\rangle$:

— $X_{C_i}^j(c) = c_i^j$ (for all *j* such that $1 \le j \le e_i$);

— $X_{C_i}^j(c)++ =$ the successor of c_i^j in C_i (C_i is supposed to be ordered and *j* is such that $1 \le j \le e_i$);

— $C_i.all(c) = \sum x$ and $C_{i,q}.all(c) = \sum x$

IMPORTANT — The *all* function is a constant function. It can thus be viewed as a constant bag and used to define (initial) markings.

The color functions ranging over a class are obtained by linear combinations of basic color functions (some constraints are required to ensure that the color functions select a *positive* number of tokens).

7.1.5 Concept 17 (class color functions)

Let C_i be a class color and $D = C_1^{e_1} \times \ldots \times C_k^{e_k}$ be a color domain. A color function from D to $Bag(C_i)$ is a linear combination

$$f_{C_i} = \sum_{k=1..e_i} \alpha_{i,k} . \left\langle X_{C_i}^k \right\rangle + \sum_{q=1..s_i} \beta_{i,q} . \left\langle C_{i,q}.all \right\rangle + \sum_{k=1..e_i} \gamma_{i,k} . \left\langle X_{C_i}^k ++ \right\rangle$$

such that $\forall d \in D, \forall c \in C_i, f_{C_i}(d)(c) \ge 0$.

Some simple constraints on $\alpha_{i,k}, \beta_{i,q}$ and $\gamma_{i,k}$ can be defined to ensure that $f_{C_i}(d)(c)$ is positive for all $c \in C_i$.

For instance, the constraint: $\forall q \in 1..s_i, \forall K \subseteq \{1..e_i\}, \left[\beta_{i,q} + \sum Min\left(\alpha_{i,k}, \gamma_{i,k}\right)\right] \ge 0$ achieves this goal.

There is an arc annotated with an expression f_{C_i} from a place $p \in P$ to a transition $t \in T$ iff $W(p,t) = f_{C_i}$ with $f_{C_i} \neq undefined$ and there is an arc from a transition $t \in T$ to a place $p \in P$ iff $W(t,p) = f_{C_i}$ $with f_{C_i} \neq undefined$.

7.2 Behavioral semantics

7.2.1 Concept 18 (marking of a symmetric net)

A marking *M* of a symmetric net is a vector M in $Bag(C(p))$ such that *M(p)* is the marking of place *p*.

A marked symmetric net is a tuple $SN = \langle N, W, Cl, C, \Phi, M_0 \rangle$ where M_0 denotes the initial marking of the net.

7.2.2 Concept 19 (symmetric net enabling rule)

A transition $t \in T$ is enabled in marking *M*, denoted by $M[t>$, iff: $\forall p \in \bullet t, M(p) \geq W(p,t) \wedge \Phi(t) = True$.

From these definitions, we can define the enabling function for symmetric nets as follows:

$$E_{sn}(N \in N, t \in T): \begin{cases} True & iff \forall p \in \bullet t, M(p) \geq W(p,t) \wedge \Phi(t) = True \\ False & otherwise \end{cases}$$

7.2.3 Concept 20 (symmetric net firing rule)

If a transition $t \in T$ is enabled in marking *M*, it can fire leading to marking *M'*, denoted by $M[t > M'$, where: $\forall p \in P, M'(p) = M(p) - W(p,t) + W(t,p)$. This corresponds to the update of the state of the net.

For symmetric nets, *A*-$_{pre}$ and *A*-$_{post}$ are the identity function.

8 High-level Petri nets

8.1 General

Symmetric nets are a subclass of high-level Petri nets (HLPN). Therefore, all definitions in Clause 7 also apply to high-level Petri nets. In high-level Petri nets, the carrier sets of color classes may be infinite, and there are no restrictions on functions, that can be user-defined.

8.2 Definition — Concept 21 (high-level Petri net)

A high-level Petri net is a tuple $HL = \langle N, W, D, C, V, \Sigma, \Phi \rangle$ where:

— $D = \{C_1, \ldots, C_k\}$ is a finite set of finite classes, each being possibly partitioned into s_i static sub-classes ($C_i = \uplus_{q=1..s_i} C_{i,q}$).

— C is a mapping which defines for each place and each transition its color domain.

— $\Sigma = (S, O)$, is a signature, where *S* is a set of color domains, and *O* the set of associated operators.

— *V* is a S-indexed set of variables, disjoint of *O*.

— $W = TERM(O \cup V)$ defines the arcs annotations, which are terms over the operators and variables. W associates with each pair $(p,t) \in P \times T$ and each pair $(t,p) \in T \times P$ a general color function defined from $C(t)$ to $Bag(C(p))$;

— Φ is a mapping $T \rightarrow TERM_B(O \cup V)$ that associates a guard with each transition.

In high-level Petri nets, arcs are labelled by color functions that select tokens in adjacent places depending on the instantiation performed for the firing.

The color functions are defined over the set of operators O and the set of variables V, i.e. over $TERM(O \cup V)$.

8.3 Behavioral semantics

8.3.1 Concept 22 (marking of a high-level Petri net)

A marking M of a HLPN is a vector $M \in Bag(C(p))$ such that $M(p)$ is the marking of place p.

A marked HLPN is a tuple $HL = \langle N, W, D, C, V, \Sigma, \Phi, M_0 \rangle$ where M_0 denotes the initial marking of the net.

8.3.2 Concept 23 (high-level Petri net enabling rule)

A transition t in T is enabled in marking M, denoted by $M[t>$, iff: $\forall p \in \bullet t, M(p) \geq W(p,t) \wedge \Phi(t) = True$.

From these definitions, we can define the enabling function for HLPN as follows:

$$E_{hpln}(N \in N, t \in T) : \begin{cases} True & iff \forall p \in \bullet t, M(p) \geq W(p,t) \wedge \Phi(t) = True \\ False & otherwise \end{cases}$$

8.3.3 Concept 24 (high-level Petri net firing rule)

If a transition $t \in T$ is enabled in marking M, it can fire leading to marking M', denoted by $M[t > M'$, where: $\forall p \in P, M'(p) = M(p) - W(p,t) + W(t,p)$. This corresponds to the update of the state of the net.

For high-level Petri nets, $A\text{-}_{pre}$ and $A\text{-}_{post}$ are the identity function.

9 Petri nets with priorities

9.1 General

This clause defines Petri nets with priorities. There are two types of Petri nets with priorities: the static ones and the dynamic ones.

9.2 Definition

9.2.1 Concept 25 (dynamic prioritized Petri net)

A dynamic prioritized Petri net (DPPN) is a marked Petri net $DP = \langle N, \rho \rangle$, where:

— N is either a P/T, symmetric or high-level net,

— ρ is the priority function mapping a marking and a transition into $\mathbb{R}^+$.

9.2.2 Concept 26 (statically prioritized Petri net)

A statically prioritized Petri net (SPPN) is a DPPN for which the ρ priority function is restricted to a constant value over $\mathbb{R}^+$.

The behavior of a prioritized Petri net is now detailed, markings being those of the component Petri net.

IMPORTANT — The firing rule is the same as for non-prioritized Petri nets, the priority scheme influencing only the enabling condition. Here the highest value corresponds to the highest priority.

The semantics is the same as the semantics of its Petri net component.

9.3 Behavioral semantics

9.3.1 Concept 27 (marking of a prioritized Petri net)

The marking of a prioritized Petri net is the same as the marking of its Petri net component.

9.3.2 Concept 28 (prioritized Petri net enabling rule)

The prioritized enabling rule is added to the one of the Petri net where priorities are added.

A transition $t \in T$ is priority enabled in marking *M*, denoted by $M[t>^{\rho}$, iff:

— it is enabled, i.e. $M[t>$, and

— no transition of higher priority is enabled, i.e. $\forall t' : M[t'> \Rightarrow \rho(M,t) \leq \rho(M,t')$.

The definition of the priority function ρ is extended to sets and sequences of transitions (and even markings *M*):

— $\forall X \subseteq T : \rho(M,X) = max\{\rho(M,t) | t \in X \wedge M[t>\}$

— $\forall \sigma \in T^* : \rho(M,\sigma) = min\{\rho(M',t') | M'[t'>^{\rho}$ occurs in $M[\sigma>^{\rho}\}$.

The filtering function $F_p(N,t)$ that returns true when *prio(t)* has the lowest value over the net is:

$$F_p(N \in N, t \in T) : \begin{cases} True & iff \forall t', \rho(M,t) \leq \rho(M,t') \\ False & otherwise \end{cases}$$

The enabling function for P/T nets with priorities is thus:

$$E_{ptp}(N,t) = E_{pt}(N,t) \wedge F_p(N,t)$$

The enabling function for symmetric nets with priorities is thus:

$$E_{snp}(N,t) = E_{sn}(N,t) \wedge F_p(N,t)$$

The enabling function for high-level Petri nets with priorities is thus:

$$E_{hlpnp}(N,t) = E_{hlpn}(N,t) \wedge F_p(N,t)$$

10 Time Petri nets

10.1 General

Petri nets with time are Petri nets where timing constraints are associated with the nodes or arcs. Timing constraints are given as time intervals. This document focusses on the definition of time Petri nets, where the timing constraints are associated with transitions.

10.2 Definition

10.2.1 Concept 29 (generic time Petri net)

A generic time Petri net is a tuple $GTP = \langle N, W, S_0, \alpha, \beta \rangle$ such that:

— N is either a P/T, a symmetric or a high-level Petri net,

— S_0 denotes its initial state, i.e. marking and clocks,

— $\alpha : X \to \mathbb{R}_+$ is a mapping from $X \in \{P, T, P \times T \cup T \times P\}$ to the set $\mathbb{R}_+$,

— $\beta : X \to \mathbb{R}_+ \cup \{\infty\}$ is a mapping from $X \in \{P, T, P \times T \cup T \times P\}$ to the set $\mathbb{R}_+ \cup \{\infty\}$.

The lower and upper bounds associated with an element *x* are denoted by $\alpha(x)$ and $\beta(x)$.

10.2.2 Concept 30 (time Petri net)

A time Petri net is a generic time Petri net, where $\alpha : T \to \mathbb{Q}_+$ and $\beta : T \to \mathbb{Q}_+ \cup \{\infty\}$ are functions satisfying $\forall t \in T, \alpha(t) \leq \beta(t)$ called respectively earliest (α) and latest (β) transition firing times.

Given a marking *M*, let $En(M) = \{t \in T \mid M[t>\}$ for the set of transitions enabled in *M*. A clock is implicitly associated with each transition and a state of the system is a pair *(M,ν)*, where *M* is a marking and $\nu \in \mathbb{R}_+^{En(M)}$ is a mapping associating a clock value with each transition enabled in *M*.

10.3 Behavioral semantics

10.3.1 General

The semantics of TPNs is based on the notion of clocks. One or more clocks can be associated with a time interval and the value of all clocks progress synchronously as time elapses. The firability of enabled transitions depends on having the value of the related clocks in their associated intervals. A clock may be reset upon meeting a condition on the marking of the net, usually after the firing of a transition.

The semantics of TPNs is defined in terms of:

— Reset policy: the value of a clock is reset upon firing some transition. It is the only way to decrease its value. But it is also meaningful not to reset a clock.

— Strong firing policy: in a strong semantics, when the upper bound of the interval associated with a clock is reached, transitions shall fire instantaneously, until the clock is reset. The clock can go beyond the upper bound of the interval, if there is no possible instantaneous sequence of firings, in which case dead tokens are usually generated. This generally models a bad behavior, since tokens become too old to satisfy the timing constraints.

— Weak firing policy: in this case, the clock leaving the interval prevents the associated firings from taking place. Dead tokens may also be generated, but this time they are considered part of the normal behavior of the net.

— Monoserver setting: each interval only has one associated clock, which usually denotes a single task processing.

— Multiserver setting: each interval has more than one associated clock, which usually denotes the handling of several similar tasks. In this setting, each clock evolves independently of the others.

Time Petri nets use a strong and monoserver semantics.

10.3.2 Concept 31 (time Petri net enabling rule)

A transition $t \in T$ is time enabled in state *(M,v)*, denoted by $(M,v)[t>$, if:

— it is enabled, i.e. $M[t>$, and

— $v(t) \in [\alpha(t), \beta(t)]$.

The filtering function $E_{tpn}(N,t)$ returns true when the local *v(t)* associated with *t* is in the range $[\alpha, \beta]$ (constants associated with *t*).

$$F_{tpn}(N \in N, t \in T): \begin{cases} True & iff v(t) \geq \alpha(t) \wedge v(t) \leq \beta(t) \\ False & otherwise \end{cases}$$

The enabling condition for P/T nets with timing constraints is thus:

$$E_{pttpn}(N,t) = E_{pt}(N,t) \wedge F_{tpn}(N,t)$$

The same applies to symmetric nets and high-level Petri nets when they are the net component of a time Petri net.

10.3.3 Concept 32 (time Petri net firing rule)

From a state *(M,v)*, two types of transitions are possible:

— if transition $t \in T$ is time enabled in state *(M,v)*, firing *t* leads to state *(M′,v′)*, denoted by $(M,v)[t>(M',v')$, where:

  — $\forall p \in P, M'(p) = M(p) - W(p,t) + W(t,p)$,

  — $\forall t' \in En(M'), v'(t') = 0$ if *t′* is newly enabled (explained below), and *v(t′)* otherwise.

— if $\forall t \in En(M), v(t) + d \leq \beta(t)$, time elapsing by delay $d \in \mathbb{R}_+$ leads to state *(M,v′)*, where $v'(t) = v(t) + d$ for all $t \in En(M)$.

Transition *t′* is newly enabled by the firing of *t* if:

— *t′* belongs to *En(M′)* and

— either $t' = t$ or *t′* is not enabled in $M - W(.,t)$ (where *W(.,t)* denotes the vector $(W(p,t))_{p \in P}$).

For time Petri nets, *A-*$_{pre}$ corresponds to time elapse, and *A-*$_{post}$ is the identity function.

Annex A
(normative)

Mathematical conventions

A.1 General

This annex defines the mathematical conventions used for the definition of Petri nets in this document.

A.2 Sets — Notation 2 (basic sets)

— $\mathbb{N}$ denotes the natural numbers.

— $\mathbb{N}^+$ denotes the positive integers.

— $\mathbb{Z}$ denotes the integers.

— $\mathbb{Q}$ denotes the rational numbers.

— $\mathbb{Q}^+$ denotes the rational positive numbers.

— $\mathbb{R}$ denotes the real numbers.

— $\mathbb{R}^+$ denotes the real positive numbers.

— B denotes the Booleans.

— S denotes any set. It is used in generic definitions.

A.3 Boolean functions

The following classical Boolean functions in two-valued Boolean algebra are used in this document:

— $AND{:}\ B \times B \rightarrow B$

— $OR{:}\ B \times B \rightarrow B$

— $AND{:}\ B \rightarrow B$

— $IMPLIES{:}\ B \times B \rightarrow B$

They may be represented by their associated notation:

— $\wedge$ for *AND*

— $\vee$ for *OR*

— ! for *NOT*

— $\Rightarrow$ for *IMPLIES*

A.4 Relation functions

The following classical inequality relations in the order theory are used in document:

— the less than relation, denoted by the symbol <

— the less than or equal to relation, denoted by the symbol ≤

— the greater than relation, denoted by the symbol >

— the greater than or equal to relation, denoted by the symbol ≥

— the not equal to relation, denoted by the symbol ≠

A.5 Multisets

A.5.1 Definitions

A.5.1.1 Concept 33 (multiset)

Let S be a non-empty set. A multiset B over S (also known as a bag, see A.5.3), is a pair $B = S, f$, where S is the basis set and f is a function $f : S \rightarrow \mathbb{N}$, which associates a multiplicity, possibly zero, with each of the basis elements. The multiplicity of $\alpha \in S$ in B, is given by $f(\alpha)$. A set is a special case of a multiset, where the multiplicity of each of the basis elements is either zero or one.

A.5.1.2 Concept 34 (support of a multiset)

Let $A = \langle A, f \rangle$ be a multiset. A subset B of A is called the support of A if for every α such that $f(\alpha) > 0$ this implies that $\alpha \in B$, and for every α such that $f(\alpha) = 0$ this implies that $\alpha \notin B$. Note that the support of a multiset B may be equal to the basis set of B.

EXAMPLE Given the multiset $B = [a,a,b,c,c]$, then its support is the set $\{a,b,c\}$.

A.5.1.3 Notation 3 (set of multisets)

The set of multisets over S is denoted by μS.

A.5.1.4 Concept 35 (membership)

Given a multiset, $B \in \mu S, \alpha \in S$ is a member of B, denoted $\alpha \in B$, if $f(\alpha) > 0$, and conversely if $f(\alpha) = 0$, then $\alpha \notin B$.

EXAMPLE Given the set $A = \{a,b,c\}$, $B = [a,a,b,c,c]$ is a multiset over A, where $f(a) = 2$.

A.5.1.5 Notation 4 (occurrences of an element)

Given a multiset $B = \langle S, f \rangle$, the occurrences of an element (i.e. an element scaled by its multiplicity) $\alpha \in S$ in B is denoted by $n`\alpha, n \in \mathbb{N}$, such that $f(\alpha) = n$.

A.5.1.6 Concept 36 (empty multiset)

The empty multiset, $\langle \varnothing, f \rangle$, has no members: $\forall \alpha \in \varnothing, f(\alpha)=0$.

A.5.1.7 Concept 37 (sum representation)

A multiset may be represented as a symbolic sum of basis elements scaled by their multiplicities (sometimes known as coefficients).

$$B=\sum_{\alpha \in S} f(\alpha)`\alpha$$

EXAMPLE The multiset $B=[a,a,b,c,c]$ can also be represented by $B=\{2`a,b,2`c\}$.

A.5.1.8 Concept 38 (cardinality and finite multiset)

Multiset cardinality is defined in the following way. The cardinality $|B|$ of a multiset B, is the sum of the multiplicities of each of the members of the multiset. $|B|=\Sigma f(\alpha)$ When $|B|$ is finite, the multiset is called a finite multiset.

A.5.1.9 Concept 39 (sub-multiset)

Let $B_1=\langle S, f_1 \rangle, B_2=\langle S, f_2 \rangle \in \mu S$. B_1 is a sub-multiset of B_2 (also: is less than or equal to, or: is contained in), $B_1 \subseteq B_2$ *iff* $\forall \alpha \in S, f_1(\alpha) \leq f_2(\alpha)$.

A.5.1.10 Concept 40 (multiset equality)

Two multisets, $B_1=\langle S, f_1 \rangle, B_2=\langle S, f_2 \rangle \in \mu S$, are equal, $B_1=B_2$, *iff* $\forall \alpha \in S, f_1(\alpha)=f_2(\alpha)$.

A.5.2 Multiset operations

A.5.2.1 General

The addition and subtraction operations on multisets, $B_1, B_2 \in \mu S$, are defined as follows. In the definitions below, let $B_1=\langle S, f_1 \rangle, B_2=\langle S, f_2 \rangle \in \mu S$ and $B=\langle S, f \rangle$.

A.5.2.2 Concept 41 (addition of multisets)

$B=B_1+B_2$ *iff* $\forall \alpha \in S, f(\alpha)=f_1(\alpha)+f_2(\alpha)$.

A.5.2.3 Concept 42 (subtraction of multisets)

$B=B_1-B_2$ *iff* $\forall \alpha \in S, (f_1(\alpha) \geq f_2(\alpha)) \wedge (f(\alpha)=f_1(\alpha)-f_2(\alpha))$.

A.5.2.4 Concept 43 (scalar multiplication of a multiset)

The scalar multiplication of a multiset, $B_1 \in \mu S$, by a natural number, $n \in \mathbb{N}$, is defined as:

$B=nB_1$ *iff* $\forall \alpha \in S, f(\alpha)=n \times f_1(\alpha)$.

A.5.3 Bags

A.5.3.1 General

Multisets are also known as bags. In this subclause, a shorthand notation is introduced for multisets, through the definition of the equivalent notion of bag.

A.5.3.2 Concept 44 (bag, support and sum representation)

Let C be a set, then a bag (or multiset) over C, is a mapping $a : C \to \mathbb{N}$.

The support of a is the set $\{x \in C \mid a(x) \neq 0\}$.

A bag a may be represented by the symbolic expression $\sum a(x).x$. In this sum, scalars $a(x) = 1$ and terms when $a(x) = 0$ are not considered.

A.5.3.3 Notation 5 (bag notation, bag equivalence with the supporting set)

Let C be a set, then a *bag* over C is denoted by *Bag(C)*. Considering the sum representation of a bag, x denotes the bag reduced to the single item x and $\sum x$ denotes the bag equivalent to the whole set C.

A.5.3.4 Concept 45 (cardinality)

The cardinality of $a \in Bag(C)$, noted *size(a)*, is defined by $\text{size}(a) = \sum a(x)$.

A.5.3.5 Concept 46 (sub-bag)

Let $a, b \in Bag(C)$, then b is a sub-bag of a (also: less than or equal to, or: is contained in), noted $a \geq b$, iff $\forall x \in C, a(x) \geq b(x)$.

A.5.3.6 Concept 47 (addition of bags)

Let $a, b \in Bag(C)$. $a+b$ is defined by $(a+b)(x)=a(x)+b(x)$.

A.5.3.7 Concept 48 (subtraction of bags)

Let $a, b \in Bag(C)$. When $a \geq b$, then $a\text{-}b$ is defined by $(a\text{-}b)(x)=a(x)\text{-}b(x)$.

A.5.3.8 Concept 49 (embedding of cartesian products)

Let C and C' be two sets, $b \in Bag(C)$ and $b' \in Bag(C')$. Then $Bag(C) \times Bag(C')$ may be viewed as a subset of $Bag(C \times C')$, by mapping $\langle b, b' \rangle$ onto $\sum b(c) b'(c') \langle c, c' \rangle$. This embedding can be generalized to any cartesian product of sets of bags.

EXAMPLE For instance, $\langle 2x + 3y, 4z \rangle \equiv 8\langle x, z \rangle + 12\langle y, z \rangle$.

Annex B
(informative)

Guidelines for graphical notations

B.1 General

This annex describes the state of the graphical notations and illustrates them with small significant examples.

B.2 Common concepts for Petri nets

B.2.1 General

Petri nets are usually represented using a graphical form which is very useful to visualize the dynamics representation of a system such as data and control flows.

Places and transitions and their relationships are visually displayed, as well as needed annotations to the graphical elements, such as marking, guards and color functions. The concepts of enabling and firing transitions rules are also introduced to the graphical form to visualize systems dynamics.

The graphical form comprises two parts: the graph itself which represents the Petri net elements graphically and carries textual annotations; and a declaration, defining all the types, variables, constants and functions that are used to annotate the graph. The declaration may also include the initial marking and the typing function, if these cannot be written on the graph part due to a lack of space. There needs to be a visual association between an annotation and the net element to which it belongs. The width, color and patterns of the lines used to draw the graph are not mandated by this document.

B.2.2 Overview of all annotations

First, consider the following color classes and variables declarations:

$C1 = [a, b, c]$, is an unordered enumeration;

$C2 = [1, 2, 3]$, is a linearly ordered enumeration;

$x, y{:}C1$, are variables of domain *C1*;

$z, t{:}C2$, are variables of domain *C2*.

Table B.1 shows the notation that can be used in marking annotations and arc expressions. These annotations are illustrated for P/T nets on the one hand, and high-level Petri nets and time Petri nets on the other hand. Note that high-level Petri nets include symmetric nets.

Table B.1 — Notation in possible situations for marking annotations and arc expressions

	PN Type	No domain, 1 token	No domain, multiple tokens	*C1*	*C1* x *C1*	*C1* x *C2*	*C1*
Marking	PT	• , or 1	•• , or 2	—	—	—	—
	HLPN & TPN	$\langle \bullet \rangle$	$2'\langle \bullet \rangle$	$\langle a \rangle$	$\langle a,b \rangle$	$\langle a,1 \rangle +$ $2'\langle b,2 \rangle$	<*C1*.all>
Arc expression	P/T	—	2	—	—	—	—
	HLPN & TPN	$1'\langle \bullet \rangle$	$2'\langle \bullet \rangle$	$\langle a \rangle + \langle x \rangle$	$\langle a,b \rangle, \langle a\ y, \rangle\ \langle a\ x \rangle + \langle x,b \rangle$	$\langle x,z \rangle +$ $2'\langle y,t \rangle$	<*C1*.all>

B.2.3 Places

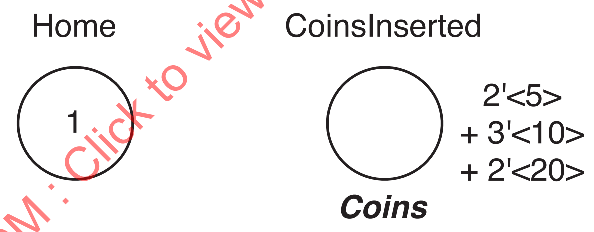
Places are represented by rounded graphical symbols (circles, ellipses). Three annotations are associated with a place *p*:

— the place name, mandatory;

— the name of the type (*Type*(*p*)) associated with the place, optional in the case of P/T nets, mandatory in the case of symmetric and high-level Petri nets;

— the marking (*M*(*p*)), optional;

— the capacity of the place, optional.



NOTE The marking of Home is *M*(Home) = 1. The type of CoinsInserted is *Type*(CoinsInserted) = Coins; its marking is *M*(CoinsInserted) = 2'<5> + 3'<10> + 2'<20>.

Figure B.1 — A P/T or time net named Home, and a high-level Petri net place, named CoinsInserted

A mechanism shall be provided to remove any ambiguity regarding the association of these annotations with the correct place. The position of the annotations with respect to places is not mandated. For example, the marking could be shown inside the graphical element, and its name and type outside, or the name of the place could be included inside the graphical element, and the type and marking outside. If the marking is empty, then it may be omitted.

A token can be considered as an element of a set. The marking *M(p)* of a place may be shown graphically by annotating the place with a multiset of tokens.

Figure B.1 shows examples of markings for a place named Home, in the case of P/T or time Petri nets, and another place named InsertedCoins, in the case of high-level Petri nets.

B.2.4 Arcs

An arc is represented by an arrow: $\rightarrow$.

It is generally used to connect a place *p* to a transition *t* and vice versa, a transition to a place. An arc from a place to a transition indicates that this transition consumes tokens from the place. An arc in the opposite direction indicates that a transition produces tokens on a place.

Arcs are annotated with terms. The notation for terms is not mandated, but usual mathematical conventions should be adopted as necessary.

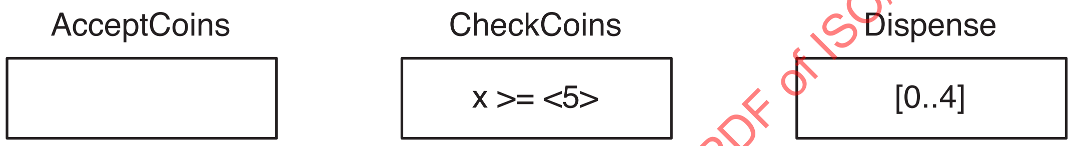
B.2.5 Transitions

A transition is represented by an edged graphical element (rectangle, square, bar). Three annotations are associated with a transition *t*:

— the transition name, mandatory;

— a boolean expression ($\Phi(t)$), the transition condition or the guard, optional. If the guard is true ($\Phi(t)$ = *true*), it may be omitted. For example, ($x<y$)($t1$) represents a transition with a name *t1*, and a guard, $x<y$, where both the types of the variables, *x* and *y*, and the operator less than, <, are either defined in the declarations or built-in by the virtue of their respective defining relation function;

— the earliest ($\alpha(t)$) and latest ($\beta(t)$) firing times of the clock associated with the transition, optional.



NOTE The guard of the transition CheckCoins is Φ(CheckCoins) = $x \geq$ <5>. The earliest firing time of transition Dispense is α(Dispense) = 0, and its latest firing time is β(Dispense) = 4.

Figure B.2 — A P/T net transition named AcceptCoins, a high-level Petri net transition named CheckCoins, and a time Petri net transiton named Dispense

A mechanism shall be provided to remove any ambiguity regarding the association of these annotations with the correct transition. The position of the annotations with respect to transitions is not mandated. For example, the guard could be shown inside the rectangle, as in the example in Figure B.2, and its name outside, or the name of the transition could be included inside the rectangle, and the guard outside.

B.3 Place/Transition Petri nets

B.3.1 General

For place/transition Petri nets, the following representation applies.

B.3.2 Places

Places are represented in the same way as in B.2.3, with only two annotations, the name and the marking, as typing is not necessary.

A token is represented by a black dot (•). The marking of a place is a natural number of tokens. Therefore, the annotation as a natural number is the most common way to represent the marking of a place.

B.3.3 Arcs

Arcs are represented in the same way as in B.2.4. Every arc is annotated by a natural positive integer, called the weight. In the case the weight is equal to 1, no annotation is required.