High-level Petri Nets - Concepts, Definitions and Graphical Notation

Final Draft International Standard ISO/IEC 15909
Version 4.7.3

May 10, 2002

Contents

Introduction

This International Standard provides a well-defined semi-graphical technique for the specification, design and analysis of systems. The technique, High-level Petri Nets, is mathematically defined, and may 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 may be subjected to analysis methods to prove properties about the specifications, before implementation commences, thus saving on testing and maintenance time.

Petri nets have been used to describe a wide range of systems since their invention in 1962. A problem with Petri nets is the explosion of the number of elements of 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 standard 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 standard captures the spirit of these earlier developments (see bibliography).

The technique promises to have multiple uses. For example, it may be used directly to specify systems or to define the semantics of other less formal languages. It may 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.

This International Standard may be cited in contracts for the development of software (particularly critical software), or used by application developers or Petri net tool vendors or users.

This International Standard provides an abstract mathematical syntax and a formal semantics for the technique. Conformance to the standard is possible at several levels. The level of conformance depends on the class of high-level net supported, and also the degree to which the syntax is supported. The basic level of conformance is to the semantic model.

Clause 1 describes the scope, areas of application and the intended audience of this International Standard. Clause 2 provides normative references (none at present), while clause 3 provides a glossary of terms and defines abbreviations. The main mathematical apparatus required for defining the standard is developed in normative Annex A, and referred to in clause 4. The basic semantic model for High-level Petri Nets is given in clause 5, while the main concepts behind the graphical form are informally introduced in clause 6. Clause 7 defines the High-level Petri Net Graph, the form of the standard intended for industrial use. The inscriptions are defined at a meta level allowing many different concrete syntaxes to be used. Clause 8 further describes syntactical conventions. Clause 9 relates the graphical form to the basic semantic model. The conformance clause is given in clause 10. Normative Annex B defines Place Transition nets (without capacities) as a restriction of the definition of Clause 7. Place Transition nets are what many readers would consider to be Petri nets, with 'black dots' as tokens, and positive integers

for arc weights. Three informative annexes are included: Annex C defines a High-level Petri Net Schema, which allows classes of systems to be described at a syntactic level; Annex D is a tutorial on the High-level Petri Net Graph; and Annex E provides pointers to analysis techniques for High-level Petri Nets. A bibliography concludes this International Standard.

1 Scope

1.1 Purpose

This International Standard defines a Petri net technique, called High-level Petri Nets, including its syntax and semantics. It provides a reference definition that can be used both within and between organisations, to ensure a common understanding of the technique and of the specifications written using the technique. This International Standard will also facilitate the development and interoperability of Petri net computer support tools.

This International Standard, defines a mathematical semantic model, an abstract mathematical syntax for inscriptions and a graphical notation for High-level Petri Nets.

This International Standard does not provide a concrete syntax nor a transfer syntax and it does not address techniques for modularity (such as hierarchies), augmentation of High-level Petri Nets with time, and methods for analysis which may become the subject of future standardisation efforts.

1.2 Field of Application

This International Standard 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;
- modelling business and software processes;
- providing the semantics for concurrent languages;
- simulation of systems to increase confidence;
- formal analysis of the behaviour of critical systems; and
- development of Petri net support tools.

This International Standard may be applied to the design of a broad range of systems and processes, including air traffic control, avionics, banking, biological and chemical processes, business processes, communication protocols, computer hardware architectures, control systems,

databases, defence command and control, distributed computing, electronic commerce, fault-tolerant systems, 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, space, telecommunications and workflow.

1.3 Audience

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

2 Normative References

None.

3 Terms, Definitions, Abbreviations and Symbols

For the purpose of this International Standard, the following definitions, abbreviations and symbols apply. Any ambiguity in the definitions is resolved by the mathematically precise definitions in the body of this International Standard.

3.1 Glossary

3.1.1 Arc: A directed edge of a net which may connect a place to a transition or a transition to a place. Normally represented by an arrow.

3.1.1.1 Input Arc (of a transition): An arc directed from a place to the transition.

3.1.1.2 Output Arc (of a transition): An arc directed from the transition to a place.

3.1.1.3 Arc annotation: An expression that may involve constants, variables and operators used to annotate an arc of a net. The expression must evaluate to a multiset over the type of the arc's associated place.

3.1.2 Arity: The input sorts and output sort for an operator.

3.1.3 Assignment: For a set of variables, the association of a value (of correct type) to each variable.

3.1.4 Basis set: The set of objects used to create a multiset.

3.1.5 Binding: See Assignment.

3.1.6 Carrier: A set of a many-sorted algebra.

3.1.7 Concurrency: The property of a system in which events may occur independently of each other, and hence are not ordered (see also Step and Concurrent Enabling).

3.1.8 Declaration: A set of statements which define the sets, constants, parameter values, typed variables and functions required for defining the inscriptions on a High-level Petri Net Graph.

3.1.9 Enabling (a transition): A transition is enabled in a particular mode and net marking, when the following conditions are met:

The marking of each input place of the transition satisfies the demand imposed on it by its arc annotation evaluated for the particular transition mode. The demand is satisfied when the place's marking contains (at least) the multiset of tokens indicated by the evaluated arc annotation.

NOTE: The determination of transition modes guarantees that the Transition Condition is satisfied (see Transition Mode).

3.1.10 Concurrent Enabling (of transition modes): A multiset of transition modes is concurrently enabled if all the involved input places contain enough tokens to satisfy the sum of all of the demands imposed on them by each input arc annotation evaluated for each transition mode in the multiset.

3.1.11 High-level Net (High-level Petri Net): An algebraic structure comprising: a set of places; a set of transitions; a set of types; a function associating a type to each place, and a set of modes (a type) to each transition; *Pre* function imposing token demands (multisets of tokens) on places for each transition mode; *Post* function determining output tokens (multisets of tokens) for places for each transition mode; and an initial marking.

3.1.12 High-level Petri Net Graph: A net graph and its associated annotations comprising Place Types, Arc Annotations, Transition Conditions, and their corresponding definitions in a set of Declarations, and an Initial Marking of the net.

3.1.13 Many-sorted Algebra: A mathematical structure comprising a set of sets and a set of functions taking these sets as domains and co-domains.

3.1.14 Marking (of a net): The set of the place markings for all places of the net.

3.1.14.1 Initial Marking (of the net): The set of initial place markings given with the high-level net definition.

3.1.14.2 Initial Marking of a place: A special marking of a place, defined with the high-level net.

3.1.14.3 Marking of a place: A multiset of tokens associated with ('residing in') the place.

3.1.14.4 Reachable Marking: Any marking of the net that can be reached from the initial marking by the occurrence of transitions.

3.1.14.5 Reachability Set: The set of reachable markings of the net, including the initial marking.

3.1.15 Multiset: A collection of items where repetition of items is allowed.

3.1.15.1 Multiplicity: A natural number (i.e., non-negative integer) which describes the number of repetitions of an item in a multiset.

3.1.15.2 Multiset cardinality (cardinality of a multiset): The sum of the multiplicities of each of the members of the multiset.

3.1.16 Net: A general term used to describe all classes of Petri nets.

3.1.16.1 Net Graph: A directed graph comprising a set of nodes of two different kinds, called places and transitions, and their interconnection by directed edges, called arcs, such that only places can be connected to transitions, and transitions to places, but never transitions to transitions, nor places to places.

3.1.16.2 Node (of a net): A vertex of a net graph (i.e., a place or a transition).

3.1.16.3 Petri Net: An algebraic structure with two sets, one called places and the other called transitions, together with their associated relations and functions, and named after their inventor, Carl Adam Petri.

3.1.16.4 Place/Transition Net: A Petri net comprising a net graph with positive integers associated with arcs and an initial marking function which associates a natural number of simple tokens ('black dots') with places.

3.1.17 Operator: A symbol representing the name of a function.

3.1.18 Parameter: A symbol that can take a range of values defined by a set. It is instantiated as a constant.

3.1.19 Parameterized High-level Net Graph: A high-level net graph that contains parameters in its definition.

3.1.20 Place: A node of a net, taken from the place kind, normally represented by an ellipse in the net graph. A place is typed.

3.1.20.1 Input Place (of a transition): A place connected to the transition by an input arc.

3.1.20.2 Output Place (of a transition): A place connected to the transition by an output arc.

3.1.20.3 Place Type: A non-empty set of data items associated with a place. (This set can describe an arbitrarily complex data structure.)

3.1.21 Reachability Graph: A directed graph of nodes and edges, where the nodes correspond to reachable markings, and the edges correspond to transition occurrences.

3.1.22 Signature/Many-sorted signature: A mathematical structure comprising a set of sorts and a set of operators.

3.1.22.1 Boolean signature: A signature where one of the sorts is $Bool$, corresponding to the carrier Boolean in any associated algebra, and one of the constants is $true_{Bool}$ corresponding to the value true in the algebra.

3.1.23 Sort: A symbol representing the name of a set.

3.1.23.1 Argument Sort: The sort of an argument of an operator.

3.1.23.2 Input Sort: The same as an argument sort.

3.1.23.3 Output Sort: The sort of an output of an operator.

3.1.23.4 Range Sort: The same as an output sort.

3.1.24 Term: An expression comprising constants, variables and operators built from a signature and a set of sorted variables.

3.1.24.1 Closed Term: A term comprising constants and operators but no variables. Also known as a **Ground Term**.

3.1.24.2 Term Evaluation: The result obtained after the binding of variables in the term, the computation of the results of the associated functions, and any simplifications performed (such as gathering like terms to obtain the symbolic sum representation of a multiset).

3.1.25 Token: A data item associated with a place and chosen from the place's type.

3.1.25.1 Enabling Tokens: The multiset of values obtained when an input arc annotation is evaluated for a particular binding to variables.

3.1.25.2 Simple Token: A valueless token, normally represented by a black dot, and used in Place/Transition nets (as opposed to high-level nets).

3.1.26 Transition: A node of a net, taken from the transition kind, and represented by a rectangle in the net graph.

3.1.26.1 Transition Condition: A boolean expression (one that evaluates to true or false) associated with a transition.

3.1.26.2 Transition Mode: A pair comprising the transition and a mode.

3.1.26.2.1 Mode: A value taken from the transition's type. When considering a High-level Petri Net Graph, a mode may be derived from an assignment of values to the transition's variables that satisfies the transition condition.

3.1.26.3 Transition Occurrence (Transition Rule): If a transition is enabled in a mode, it may occur in that mode. On the occurrence of the transition, the following actions occur indivisibly:

1. For each input place of the transition: the enabling tokens of the input arc with respect to that mode are subtracted from the input place's marking, and

2. For each output place of the transition: the multiset of tokens of the evaluated output arc expression is added to the marking of the output place.

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

3.1.26.4 Step: The simultaneous occurrence of a multiset of transition modes that are concurrently enabled in a marking.

3.1.26.5 Transition Variables: All the variables that occur in the expressions associated with the transition. These are the transition condition, and the annotations of arcs surrounding the transition.

3.1.27 Type: A set.

3.2 Abbreviations

3.2.1 HLPN: High-level Petri Net.

3.2.2 HLPNG: High-level Petri Net Graph.

3.2.3 HLPNS: High-level Petri Net Schema.

3.2.4 PN: Petri Net.

3.2.5 PTNG: Place/Transition Net Graph.

4 Conventions and Notation

This International Standard uses the notation for sets, multisets and universal algebra defined in Annex A. Annex A also defines the concept of multiset addition, (using the '$+$' symbol), which should not be confused with arithmetic addition. The notion of multisets is required for clauses 5, 6, 7, 8 and 9. An understanding of many-sorted signatures, sorted variables and many-sorted algebras provided in Annex A is required for clauses 7 and 9 and Annexes B and C.

Wherever possible standard mathematical notation has been used. An instance of notation specific to Petri nets is when a marking of the net is transformed to a new marking.

$M[T_\mu\rangle M'$ is used to denote that a new marking, M', is created on the occurrence of a multiset of transition modes, T_μ, when in the marking M.

This notation is defined in clause 5, and similar notation is used in clause 7.

In high-level nets, an arc may be annotated by a variable or constant (or combination) of the same type as the arc's place. The evaluation of the arc expression is thus an element of the place's type. By convention, if an arc expression evaluates to an element of the place's type, this is interpreted as the corresponding singleton multiset over the place's type. This is necessary as an arc annotation when evaluated must be a multiset over the associated place's type. A similar convention applies when an arc expression evaluates to a set, which may be a subset of the associated place's type. This is interpreted as a multiset over the place's type, with corresponding multiplicities of zero and one.

The graphical notation used in clause 6 is that defined in clause 8.

5 Semantic Model for High-level Petri Nets

This clause provides the basic semantic model for High-level Petri Nets (HLPN).

5.1 Definition

A **HLPN** is a structure $HLPN = (P, T, D; Type, Pre, Post, M_0)$ where

- P is a finite set of elements called Places.
- T is a finite set of elements called Transitions disjoint from P ($P \cap T = \emptyset$).
- D is a non-empty finite set of non-empty domains where each element of D is called a type.
- $Type : P \cup T \longrightarrow D$ is a function used to assign types to places and to determine transition modes.
- $Pre, Post : TRANS \longrightarrow \mu PLACE$ are the pre and post mappings with

$$TRANS = \{(t, m) \mid t \in T, m \in Type(t)\}$$

$$PLACE = \{(p, g) \mid p \in P, g \in Type(p)\}$$

- $M_0 \in \mu PLACE$ is a multiset called the initial marking of the net.

NOTE: $\mu PLACE$ is the set of multisets over the set, $PLACE$ (see Annex A, section A.2).

5.2 Marking of HLPN

A *Marking* of the HLPN is a multiset, $M \in \mu PLACE$.

5.3 Enabling of Transition Modes

5.3.1 Enabling of a Single Transition Mode

A transition mode, $tr \in TRANS$, is *enabled* at a marking M *iff*

$$Pre(tr) \leq M$$

5.3.2 Concurrent Enabling of Transition Modes

A finite multiset of transition modes, $T_\mu \in \mu TRANS$, is enabled at a marking M *iff*

$$Pre(T_\mu) \leq M$$

where the linear extension of Pre is given by

$$Pre(T_\mu) = \sum_{tr \in TRANS} T_\mu(tr) Pre(tr).$$

All transition modes in T_μ are said to be *concurrently enabled* if T_μ is enabled, i.e. there are enough tokens on the input places to satisfy the linear combination of the pre mappings for each transition mode in T_μ.

This definition allows a transition mode to be concurrently enabled with itself. Also if $|T_\mu| = 1$, then this reduces to the definition in the previous subclause.

5.4 Transition Rule

Given that a finite multiset of transition modes, T_μ, is enabled at a marking M, then a *step* may occur resulting in a new marking M' given by

$$M' = M - Pre(T_\mu) + Post(T_\mu)$$

where the linear extension of $Post$ is used.

A step is denoted by $M[T_\mu\rangle M'$ or $M \xrightarrow{T_\mu} M'$.

NOTE: A step may comprise the occurrence of: a single transition mode; or any number of concurrently enabled transition modes.

6 Concepts Required for High-level Petri Net Graphs

6.1 Introduction

High-level Petri Nets can be defined in a number of ways. Clause 5 provides the definition of the basic mathematical semantic model. The basic semantic model is not what is used by practitioners. High-level Petri Nets are normally represented using a graphical form which allows visualisation of system dynamics (flows of data and control). This approach is taken, as it is the graphical form of HLPNs that is most appropriate for industrial use. The graphical form is referred to as a High-level Petri Net Graph (HLPNG). It provides a graphical notation for places and transitions and their relationships, and determines the inscriptions of the graphical elements.

This clause introduces the concepts that are needed in the definition of the High-level Petri Net Graph. The concepts of enabling and transition rule are also introduced for the graphical form to demonstrate how the net may be executed to show system dynamics. Readers interested in a tutorial exposition on High-level Petri Net Graphs are referred to Annex D.

6.2 High-level Petri Net Graph Components

A High-level Petri Net Graph (HLPNG) comprises:

- *A Net Graph*, consisting of sets of nodes of two different kinds, called *places* and *transitions*, and *arcs* connecting places to transitions, and transitions to places.
- *Place Types*. These are non-empty sets. One type is associated with each place.
- *Place Marking*. A collection of elements (data items) chosen from the place's type and associated with the place. Repetition of items is allowed. The items associated with places are called *tokens*.
- *Arc Annotations*: Arcs are inscribed with expressions which may comprise constants, variables (e.g., x, y) and function images (e.g., $f(x)$). The variables are typed. The expressions are evaluated by assigning values to each of the variables. When an arc's expression is evaluated, it must result in a collection of items taken from the type of the arc's place. The collection may have repetitions.
- *Transition Condition*: A boolean expression (e.g., $x < y$) inscribing a transition.
- *Declarations*: comprising definitions of place types, typing of variables, and function definitions.

NOTE: A collection of items which allows repetitions is known in mathematics as a multiset.

6.3 Net Execution

HLPNGs are executable, allowing the flow of tokens around the net to be visualised. This can illustrate flow of control and flow of data within the same model. Key concepts governing this execution are *enabling* of transitions and the *occurrence* of transitions defined by the *Transition Rule*.

6.3.1 Enabling

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

A transition is also enabled in a particular *transition mode*. A transition mode is an assignment of values to the *transition's variables*, that satisfies the transition condition (i.e., the transition condition is true). The transition's variables are all those variables that occur in the expressions associated with the transition. These are the transition condition, and the annotations of arcs involving the transition.

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.

A transition is enabled in a specific mode, for a particular net marking. Each input arc expression is evaluated for the transition mode, resulting in a multiset of tokens of the same type as that of the input place. If each input place's marking contains at least its input arc's multiset of tokens (resulting from the evaluation of the input arc's expression in the specific mode), then the transition is enabled in that mode. An example is given in subclause 6.4.

The input arc's multiset of tokens resulting from the evaluation of the input arc's expression in a specific mode is called the input arc's *enabling tokens*, with respect to that mode.

Two transition modes are *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 (with respect to both modes) of each input arc associated with that input place.

6.3.2 Transition Rule for a Single Transition Mode

Enabled transitions can *occur*. When a transition occurs, 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.

If a transition is enabled in a mode, it may occur in that mode. On the occurrence of the transition in a specific mode, the following actions occur atomically:

1. For each input place of the transition: the enabling tokens of the input arc with respect to that mode are subtracted from the input place's marking, and

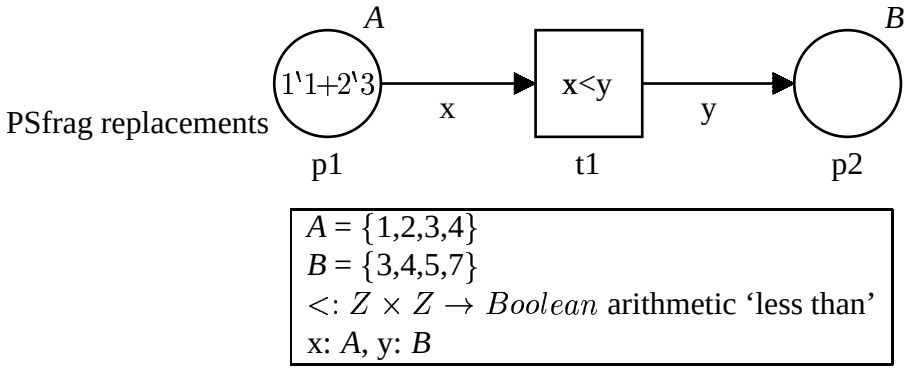


Figure 1: HLPNG with a Transition Condition.

2. For each output place of the transition: the multiset of tokens, resulting from the evaluation of the output arc expression for the mode, is added to the marking of the output place.

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

6.3.3 Step of Concurrently Enabled Transition Modes

Several concurrently enabled transition modes may occur in one *step*, that is in one atomic action. The change to the marking of the net when a step occurs is given by the sum of all the changes that occur for each transition mode, as described above. An example is given in subclause 6.4.

6.4 Graphical Concepts and Notation

The graphical representation of the net graph is introduced by the simple example of a HLPNG given in figure 1.

This example comprises two places, named p1 and p2, one transition, t1, and arcs from p1 to t1, and t1 to p2. The places are represented by ellipses (in this case, circles), the transition by a rectangle (in this example, a square), and the arcs by arrows.

The declarations define two types, A and B, that are different subsets of the positive integers. Variable x is of type A, and variable y is of type B. The transition is inscribed with the boolean expression x<y, where the *less than* operator is defined in the declarations. Arc (p1,t1) is annotated with the variable x, while arc (t1,p2) is annotated with y.

Place p1 is typed by A and has an initial marking M_0(p1)= $1`1 + 2`3$, using the sum representation of multisets of clause A.2.1. This states that associated with the place p1 are the value 1 and two instances of the value 3. Place p2 is typed by B, and is empty representing the empty multiset, M_0(p2)= $\emptyset$.

A convention adopted in high-level nets, is that arcs may be annotated by variables or constants of the same type as the arc's place. The evaluation of the arc expression is thus an element of the place's type. By convention, this is interpreted as the corresponding singleton multiset over the place's type. For example, for x bound to 1, and y bound to 3, the value associated with

arc (p1,t1) is 1, which is interpreted to mean the multiset $1`1$, and similarly the value associated with arc (t1,p2) is 3, which is interpreted to mean the multiset $1`3$. Where there is any possibility of ambiguity, the multiset conversion operator '`' should be used. For example, more formally, the annotation on the arc (p1,t1) would be 1`x, rather than x.

In the initial marking, t1 can be enabled in the following modes

$\{(1,3),(1,4),(1,5),(1,7),(3,4),(3,5),(3,7)\}$

where the first element of each pair represents a binding for x, and the second, a binding for y which satisfies x<y.

It can be seen that the multiset of modes, $1`(1,3) + 2`(3,5)$, is concurrently enabled. Another example of the concurrent enabling of modes is the multiset $1`(1,5) + 1`(3,4)$ and yet another is $1`(1,7) + 1`(3,5) + 1`(3,7)$.

If transition t1 occurred in mode $1`(3,5)$, then the resultant marking would be:

M(p1)= $1`1 + 1`3$
M(p2)= $1`5$.

Alternatively, if the multiset of modes $1`(1,3) + 2`(3,5)$ occurred concurrently, the resultant marking would be

M(p1)= $\emptyset$
M(p2)= $1`3 + 2`5$.

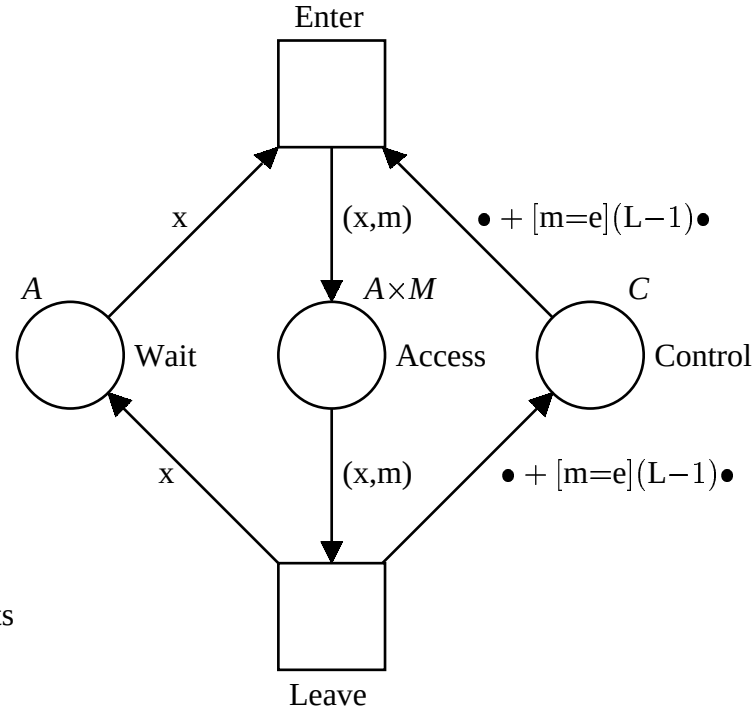
6.5 Conditionals in Arc Expressions, and Parameters

The HLPNG of figure 2 uses a variant of the readers/writers problem to illustrate many of the features of a HLPNG including the use of parameters and conditionals in arc expressions.

A number (N) of agents (processes) wish to access a shared resource (such as a file). Access can be in one of two modes: shared (s), where up to L agents may have access at the same time (e.g., reading); and exclusive (e), where only one agent may have access (e.g., writing). No assumptions are made regarding scheduling. A HLPNG model of figure 2 illustrates the use of two parameters, L and N, both of which are positive integers. This is therefore a parameterized HLPNG, which represents infinitely many readers/writers systems. Each instantiation of N and L would produce a HLPNG, which could then be executed.

It has been assumed that the initial state is when all the agents are waiting to gain access to the shared resource (with no queueing discipline assumed). In this example, the initial markings are given in the declaration. Place Wait is marked with all agents; the Control place contains L simple tokens (represented by a black dot •) and Access is empty. The marking of place Wait is given by the set A, which is interpreted to be a multiset over A, where each of the multiplicities of the agents is one. In a similar convention, the marking of Control is given by L•, which is interpreted as the multiset L`•. (That is the separator '`' is dropped where there is no ambiguity.)

An agent can obtain access in one of two modes: if shared (m=s), then a single token is removed from Control (as m=e is false, and $[\text{m=e}] = 0$, and thus the arc expression evaluates to •, interpreted as $1`\bullet$) when Enter occurs in a single mode; if exclusive (m=e), then all L tokens are removed preventing further access until the resource is released (transition Leave). Shared



PSfrag replacements

Set of Agents:$A = \{a_1, \ldots, a_N\}$
Set of Access Modes: $M = \{s,e\}$
Control: $C = \{\bullet\}$
Positive integer constants: N, L
Variables x: A; m: M
Function $[] : Bool \rightarrow \{0,1\}$ where
$[true] = 1$ and $[false] = 0$
$M_0(\text{Wait}) = A$
$M_0(\text{Control}) = L\bullet$
$M_0(\text{Access}) = \emptyset$

Figure 2: HLPNG of Resource Management.

access is limited to a maximum of L agents as transition Enter is disabled when Control is empty.

Outfix notation has been used for the function $[] : Bool \rightarrow \{0, 1\}$. This is the notation for conditionals in arc expressions. It is assumed that multiset addition, integer subtraction, the equality predicate and a tupling operator are primitive and do not need to be defined in the declaration.

7 Definition of High-level Petri Net Graphs

7.1 Introduction

This clause provides a formal definition for the graphical form of high-level nets. It provides an abstract mathematical syntax for inscribing the graphical elements. The concepts of marking, enabling and transition rule are also formally defined for the graphical form.

7.2 Definition

A **HLPNG** is a structure

$$\text{HLPNG} = (NG, Sig, V, H, Type, AN, M_0)$$

where

- $NG = (P, T; F)$ is called a net graph, with
  - P a finite set of nodes, called Places;
  - T a finite set of nodes, called Transitions, disjoint from P ($P \cap T = \emptyset$); and
  - $F \subseteq (P \times T) \cup (T \times P)$ a set of directed edges called arcs, known as the flow relation.
- $Sig = (S, O)$ is a Boolean signature defined in Annex A.
- V is an S-indexed set of variables, disjoint from O.
- $H = (S_H, O_H)$ is a many-sorted algebra for the signature Sig, defined in Annex A.
- $Type : P \rightarrow S_H$ is a function which assigns types to places.
- $AN = (A, TC)$ is a pair of net annotations.
  - $A : F \rightarrow TERM(O \cup V)$ such that for all $(p, t), (t', p) \in F$, for all bindings $\alpha, Val_\alpha(A(p, t)), Val_\alpha(A(t', p)) \in \mu Type(p)$. $TERM(O \cup V), \alpha$ and Val_α are defined in Annex A. A is a function that annotates each arc with a term that when evaluated (for any binding) results in a multiset over the associated place's type.

- $TC : T \rightarrow TERM(O \cup V)_{Bool}$ is a function that annotates transitions with Boolean expressions.

- $M_0 : P \rightarrow \bigcup_{p \in P} \mu Type(p)$ such that $\forall p \in P$, $M_0(p) \in \mu Type(p)$, is the initial marking function which associates a multiset of tokens (of the correct type) which each place.

A HLPNG comprises an annotated net graph. The annotations are derived from a many sorted signature, a set of sorted variables, and an associated many sorted algebra. A typing function associates a set of the many-sorted algebra with each place, known as its type. Terms are built from the signature and variables. Arcs are annotated by terms, which must evaluate to a multiset over the associated place's type. Transitions are annotated by Boolean terms. A place can only hold tokens of the same type as the place and hence the initial marking is a multiset over the place's type.

NOTE 1: Arbitrarily complex terms can be generated for the arc inscriptions, by defining the appropriate signature (operators and sorts) and sorted variables. Many applications will require the definition of an operator which will convert (in the algebra) an element of a set to a singleton multiset, and others will need a multiset addition operator (and its counterpart in the algebra). For example, this allows there to be multiterms and conditionals in arc expressions. The approach provides a great deal of flexibility in defining the syntax required for arc inscriptions.

NOTE 2: The evaluation of an arc term may result in an empty multiset.

NOTE 3: The carriers (sets) of the algebra may be arbitrarily complex, and may be related, for example, $H_{s2} = \mu H_{s1}$. Also, a variable, x_s, may be typed by a set of multisets, or a powerset, defined in the algebra (e.g., $H_s = \mu A$, where A is the basis set, so that $x : \mu A$).

NOTE 4: The restriction on the arc annotations (evaluating to a multiset over the place's type), allows terms which evaluate to multisets over subtypes of the place's type to be used in the annotations.

NOTE 5: In defining HLPNGs, a type has been intentionally associated with each place. This type is a carrier (set) of the many-sorted algebra, H. This allows us to specify concrete systems where the sets and functions have already been determined. There is also a need for a more abstract or syntactic form that allows classes of systems to be specified. In this case, the places become S-sorted, i.e., a sort (rather than a type) is associated with each place, markings become (appropriately sorted) ground terms of the signature, the terms for arc annotations are of the same sort as that associated with the arc's place, and the many-sorted algebra, H, is removed. This leads us to the notion of a HLPN schema which is included in informative Annex C and may be considered for future standardisation. The schemas are also useful for analysis (see bibliography).

7.3 Marking

A marking, M, of the HLPNG is defined in the same way as the initial marking.

$M : P \rightarrow \bigcup_{p \in P} \mu Type(p)$ such that for all $p \in P$, $M(p) \in \mu Type(p)$.

7.4 Enabling

7.4.1 Enabling of a Single Transition Mode

A transition $t \in T$ is enabled in a marking, M, for a particular assignment, α_t, to its variables, that satisfies the transition condition, $Val_{bool,\alpha_t}(TC(t)) = true$, known as a *mode* of t, *iff*

$$\forall p \in P \quad Val_{\alpha_t}(\overline{p,t}) \leq M(p)$$

where for $(u, v) \in (P \times T) \cup (T \times P)$,

- $\overline{u,v} = A(u, v)$, for $(u, v) \in F$,
- $\overline{u,v} = \Phi$, for $(u, v) \notin F$

where Φ is a symbol that represents the empty multiset at the level of the signature, so that $Val_\alpha(\Phi) = \emptyset$.

7.4.2 Concurrent Enabling of Transition Modes

Let α_t be an assignment for the variables of transition $t \in T$ that satisfies its transition condition and denote the set of all such assignments (that satisfy its transition condition) for transition t, by $Assign_t$.

Define the set of all transition modes to be

$$TM = \{(t, \alpha_t) \mid t \in T, \alpha_t \in Assign_t\}$$

and a *step* to be a finite non-empty multiset over TM.

A step, X, of transition modes is enabled in a marking, M, *iff*

$$\forall p \in P \quad \sum_{(t,\alpha_t) \in TM} X(t, \alpha_t) Val_{\alpha_t}(\overline{p,t}) \leq M(p)$$

Thus all of the transition modes in X are concurrently enabled, if X is enabled. If X comprises a single transition mode, then the sum is reduced to a single term so that the enabling condition is the same as that of the previous subclause.

7.5 Transition Rule

If $t \in T$ is enabled in *mode* α_t, for marking M, t may *occur in mode* α_t. When t occurs in mode α_t, the marking of the net is transformed to a new marking M', denoted $M[t, \alpha_t\rangle M'$, according to the following rule:

$$\forall p \in P \quad M'(p) = M(p) - Val_{\alpha_t}(\overline{p,t}) + Val_{\alpha_t}(\overline{t,p})$$

If a step X is enabled in marking M, then the step can occur resulting in a new marking, M', denoted $M[X\rangle M'$, where M' is given by

$$\forall p \in P \quad M'(p) = M(p) - \sum_{(t,\alpha_t)\in TM} X(t,\alpha_t)Val_{\alpha_t}(\overline{p,t}) + \sum_{(t,\alpha_t)\in TM} X(t,\alpha_t)Val_{\alpha_t}(\overline{t,p})$$

8 Notation for High-level Petri Net Graphs

8.1 General

The graphical form comprises two parts: a *Graph* which represents the net elements graphically and carries textual inscriptions; and a *Declaration*, defining all the types, variables, constants and functions that are used to annotate the Graph part. The declaration may also include the initial marking and the typing function if these cannot be inscribed on the graph part due to lack of space. There needs to be a visual association between an inscription and the net element to which it belongs.

The width, colour and patterns of the lines used to draw the graph are not mandated by this standard.

8.2 Places

Places are represented by ellipses (often circles). ◯

Three annotations are associated with a place p:

- the place name;
- the name of the type ($Type(p)$) associated with the place; and
- the initial marking, $M_0(p)$.

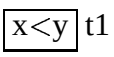
A mechanism must be provided to remove any ambiguity regarding the association of these annotations with the correct place. The position of the annotations with respect to places is not mandated. For example, the initial marking could be shown inside the ellipse, and its name and type outside, or the name of the place could be included inside the ellipse, and the type and initial marking outside.

If the initial marking is empty, then it may be omitted.

8.3 Transitions

A transition is represented by a rectangle and is annotated by a name and a boolean expression, the *Transition Condition*. If the Transition Condition is true ($TC(t) = true$), it may be omitted.

For example,



represents a transition with a name t1, and a transition condition, x<y, where both the types of the variables, x and y, and the operator less than, <, are defined in the declarations.

A mechanism must 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 transition condition could be shown inside the rectangle (as in the example), and its name outside, or the name of the transition could be included inside the rectangle, and the transition condition outside.

8.4 Arcs

An arc is represented by an arrow: $\longrightarrow$

For $(p,t) \in F$, an arrow is drawn from place p to transition t and vice versa for $(t,p) \in F$. If (p,t) and (t,p) have the same annotations (p is a side place of t), $A(p,t) = A(t,p)$, then this may be shown by a single arc with an arrowhead at both ends and annotated by a single inscription.

Arcs are annotated with terms. The notation for terms is not mandated by this International Standard, but usual mathematical conventions should be adopted where possible.

8.5 Markings and Tokens

A token is a member of $\bigcup_{p \in P} Type(p)$. A Marking of the net may be shown graphically by annotating a place with its multiset of tokens $M(p)$ using the symbolic sum representation (see clause A.2.1).

9 Semantics of High-level Petri Net Graphs

The HLPNG may be given an interpretation as a HLPN (see clause 5) in the following way.

1. Places: P is the set of places in the HLPN.
2. Transitions: T is the set of transitions in the HLPN.
3. Set of Types: The set of modes for a transition is determined by the types of the variables occurring in the surrounding arc annotations restricted by its transition condition.

   Let $V_t \subseteq V$ be the set of variables associated with transition t. Define α_t as an assignment for H and V_t (cf A.3.6), then

   $$Type(t) = \{\alpha_t | Val_{bool,\alpha_t}(TC(t)) = true\}$$

   The types of places are obtained directly from the HLPN graph definition. Thus the set of domains is given by $D = \{Type(x) | x \in P \cup T\}$.

4. The Typing Function: The typing function restricted to places is defined in the HLPNG and $Type(t)$ is given above.

5. Pre and Post Maps.

   The pre and post maps are given, for all $(p,t),(t,p) \in F$, by the following family of mappings:

   $$Pre_{(p,t)} : Type(t) \longrightarrow \mu Type(p)$$
   $$Post_{(p,t)} : Type(t) \longrightarrow \mu Type(p)$$

   For $(p,t) \notin F$ and $\forall m \in Type(t)$, $Pre_{(p,t)}(m) = \emptyset$ and for $(t,p) \notin F$ and $\forall m \in Type(t)$, $Post_{(p,t)}(m) = \emptyset$.

   Thus for all $t \in T$ and for all $m \in Type(t)$

   $$Pre(t,m) = \{(p,b) \mid p \in P, b \in Pre_{(p,t)}(m)\}$$
   $$Post(t,m) = \{(p,b) \mid p \in P, b \in Post_{(p,t)}(m)\}$$

   where $Pre_{(p,t)}(m) = Val_{\alpha_t=m}(A(p,t))$ and $Post_{(p,t)}(m) = Val_{\alpha_t=m}(A(t,p))$

6. Initial Marking.

   For all $p \in P$, $M_0(p)$ is as defined in the HLPNG.

10 Conformance

Conformance to this International Standard is according to net class. The lowest level is for the Place/Transition net class and the highest at the level of the HLPNG. Within a class, the lowest conformance level is to the semantics. If the semantics are adhered to, then the next level is with respect to syntax.

10.1 PN Conformance

10.1.1 Level 1

To claim PN Level 1 conformance to this International Standard an implementation shall demonstrate that it has the semantics defined in clause 5, by providing a mapping from the implementation's syntax to the semantic model in a similar way to that defined in clause 9.

The definition of the semantic model (clause 5), will be specialised by the restriction that D consists of one set with one element, such as $D = \{\{\bullet\}\}$. This has the following consequences.

- There is a one to one mapping between $PLACE$ and P.
- There is a one to one mapping between $TRANS$ and T.
- Each transition has a single mode.
- Each place is typed by $\{\bullet\}$ (i.e., there is only one token represented by a black dot).
- The initial marking determines the number of black dots allocated to each place initially.

10.1.2 Level 2

To claim PN Level 2 conformance to this International Standard an implementation shall have satisfied the requirements of PN Level 1 conformance and in addition shall include the syntax of the PNG defined in Annex B, section B.1 and the relevant notational conventions of clause 8, i.e., there is no need for place typing, nor a declaration, as these are trivial.

10.2 HLPN Conformance

10.2.1 Level 1

To claim HLPN Level 1 conformance to this International Standard an implementation shall demonstrate that it has the semantics defined in clause 5, by providing a mapping from the implementation's syntax to the semantic model in a similar way to that defined in clause 9.

10.2.2 Level 2

To claim HLPN Level 2 conformance to this International Standard an implementation shall have satisfied the requirements of HLPN Level 1 conformance and in addition shall include the syntax of the HLPNG defined in clause 7 and the notational conventions of clause 8.

Annex A
(normative)

Mathematical Conventions

This annex defines the mathematical conventions required for the definition of High-level Petri Nets.

NOTE: For notions of basic set theory including sets, functions and relations, see the book by Truss in the Bibliography.

A.1 Sets

$N = \{0, 1, \ldots\}$ the natural numbers.

$N^+ = N \setminus \{0\}$, the positive integers.

$Z = \{\ldots, -1, 0, 1, \ldots\}$ the integers.

$Boolean = \{true, false\}$

A.2 Multisets

A *multiset*, B, (also known as a bag) over a non-empty *basis* set, A, is a function

$$B : A \longrightarrow N$$

which associates a multiplicity, possibly zero, with each of the basis elements. The multiplicity of $a \in A$ in B, is given by $B(a)$. A set is a special case of a multiset, where the multiplicity of each of the basis elements is either zero or one.

The set of multisets over A is denoted by μA.

A.2.1 Sum Representation

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

$$B = \sum_{a \in A} B(a)`a$$

A.2.2 Membership

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

A.2.3 Empty Multiset

The empty multiset, $\emptyset$, has no members: $\forall a \in A, \emptyset(a) = 0$.

A.2.4 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.

$$|B| = \sum_{a \in A} B(a)$$

When $|B|$ is finite, the multiset is called a finite multiset.

A.2.5 Multiset Equality and Comparison

Two multisets, $B1, B2 \in \mu A$, are equal, $B1 = B2$, *iff* $\forall a \in A$, $B1(a) = B2(a)$.
$B1$ is less than or equal to (or contained in) $B2$, $B1 \leq B2$, *iff* $\forall a \in A$, $B1(a) \leq B2(a)$.

A.2.6 Multiset Operations

The addition operation and subtraction partial operation on multisets, $B1, B2 \in \mu A$, associate to the left and are defined as follows:

$B = B1 + B2$ *iff* $\forall a \in A\ B(a) = B1(a) + B2(a)$
$B = B1 - B2$ *iff* $\forall a \in A\ (B1(a) \geq B2(a)) \wedge (B(a) = B1(a) - B2(a))$

Scalar multiplication of a multiset, $B1 \in \mu A$, by a natural number, $n \in N$, is defined as

$B = nB1$ *iff* $\forall a \in A, B(a) = n \times B1(a)$

where $\times$ is arithmetic multiplication.

A.3 Concepts from Algebraic Specification

In order to define the HLPNG, concepts from algebraic specification are required. In the HLPNG, arcs are annotated by terms and transitions by Boolean expressions. Many-sorted signatures provide an appropriate mathematical framework for this representation. Signatures provide a convenient way to characterise many-sorted algebras at a syntactic level. This clause introduces the concepts of signatures, variables, terms and many-sorted algebras that are required for the definition of the HLPNG.

A.3.1 Signatures

A *many-sorted* (or *S-sorted*) signature, Sig, is a pair:

$$Sig = (S, O)$$

where

- S is a set of sorts (the **names** of sets, e.g., Int for the integers); and
- O is a set of operators (the **names** of functions) together with their *arity* in S which specifies the names of the domain and co-domain of each of the operators.

It is assumed that S and O are disjoint.

Arity is a function from the set of operator names to $S^* \times S$, where S^* is the set of finite sequences, including the empty string, ε, over S. Thus every operator in O is indexed by a pair (σ, s), $\sigma \in S^*$ and $s \in S$ denoted by $o_{(\sigma,s)}$. $\sigma \in S^*$ is called the *input* or *argument* sorts, and s the *output* or *range* sort of operator o. (In the algebra, the sequence of input sorts will define a cartesian product as the domain of the function corresponding to the operator and the output sort will define its co-domain – c.f. clause A.3.5.)

For example, if $S = \{Int, Bool\}$, then $o_{(Int.Int,Bool)}$ represents a binary predicate symbol, such as *equality* ($=$) or *less than* ($<$). Using a standard convention, the sort of a constant may be declared by letting $\sigma = \varepsilon$. For example an integer constant is denoted by $o_{(\varepsilon,Int)}$ or simply o_{Int}.

A.3.2 Boolean Signature

The term *Boolean Signature* is used to mean a many-sorted signature where one of the sorts is *Bool*, corresponding to the carrier Boolean ($Boolean = \{true, false\}$) in any associated algebra. A Boolean signature will also include the constant, $true_{Bool}$, which corresponds to the value, true, in any associated algebra.

A.3.3 Variables

Let V be a set of sorted variables, called an *S-sorted set of variables*. The sort of a variable may also be declared in the same way as that of constants, from the set of variable names to $\{\varepsilon\} \times S$. A variable in V of sort $s \in S$ would be denoted by $v_{(\varepsilon,s)}$ or more simply by v_s. For example, if $Int \in S$, then an integer variable would be $v_{(\varepsilon,Int)}$ or v_{Int}.

V may be partitioned according to sorts, where V_s denotes the set of variables of sort s (i.e., $v_a \in V_s$ *iff* $a = s$).

A.3.4 Terms built from a Signature and Variables

Terms of sort $s \in S$ may be built from a signature $Sig = (S, O)$ and an S-sorted set of variables, V, in the following way. Denote the set of terms of sort s by $TERM(O \cup V)_s$, and generate them inductively as follows. For $s \in S$

1. for all $o_{(\varepsilon,s)} \in O$, $o_{(\varepsilon,s)} \in TERM(O \cup V)_s$;
2. $V_s \subseteq TERM(O \cup V)_s$; and
3. for $s_1, \ldots, s_n \in S$ ($n > 0$), if $e_1 \in TERM(O \cup V)_{s_1}, \ldots, e_n \in TERM(O \cup V)_{s_n}$ are sorted terms and $o_{(s_1 \ldots s_n,s)} \in O$, is an operator, then $o_{(s_1 \ldots s_n,s)}(e_1, \ldots, e_n) \in TERM(O \cup V)_s$

Thus if Int is a sort, integer constants and variables, and operators (with appropriate arguments) of output sort Int are terms of sort Int.

Let $TERM(O \cup V)$ denote the set of all terms and $TERM(O)$ the set of all *closed* terms (those not containing variables, also called *ground* terms). Thus

$$TERM(O \cup V) = \bigcup_{s \in S} TERM(O \cup V)_s$$

A.3.5 Many-sorted Algebras

A many-sorted algebra, (or Sig-Algebra), H, provides an interpretation (meaning) for the signature Sig. For every sort, $s \in S$, there is a corresponding set, H_s, known as a *carrier* and for every operator $o_{(s_1 \ldots s_n, s)} \in O$, there is a corresponding function

$$o_H : H_{s_1} \times \ldots \times H_{s_n} \rightarrow H_s.$$

In case an operator is a constant, o_s, then there is a corresponding element $o_H \in H_s$, which may be considered as a function of arity zero.

Definition: A many-sorted Algebra, H, is a pair

$$H = (S_H, O_H)$$

where

- $S_H = \{H_s | s \in S\}$ is the set of carriers, with for all $s \in S$, $H_s \neq \emptyset$ and
- $O_H = \{o_H | o_{(\sigma, s)} \in O, \sigma \in S^* \text{and } s \in S\}$ the set of corresponding functions.

For example, if $Sig = (\{Int, Bool\}, \{<_{(Int, Int, Bool)}\})$ then a corresponding many-sorted algebra would be

$$H1 = (Z, Boolean; lessthan)$$

where Z is the set of integers: $\{\ldots, -1, 0, 1, \ldots\}$
$Boolean = \{true, false\}$
and $lessthan : Z \times Z \rightarrow Boolean$ is the usual integer comparison function.

It could also be

$$H2 = (N, Boolean; lessthan)$$

where N is the set of non-negative integers: $\{0, 1, \ldots\}$
$Boolean = \{true, false\}$
and $lessthan : N \times N \rightarrow Boolean$.

NOTE: In the algebra, S-sorted variables are typed by the carrier corresponding to the sort.

A.3.6 Assignment and Evaluation

Given an S-sorted algebra, H and S-sorted variables in V, an *assignment* [1] for H and V is a family of functions α, comprising an assignment function for each sort $s \in S$,

$$\alpha_s : V_s \to H_s.$$

This function may be extended to terms by defining the family of functions Val_α comprising, for each sort $s \in S$:

$$Val_{s,\alpha} : TERM(O \cup V)_s \to H_s$$

The values are determined inductively as follows. For $s \in S$

1. For a constant, $o_s \in O$, $Val_{s,\alpha}(o_s) = o_H \in H_s$;
2. For a variable, $v_s \in V_s$, $Val_{s,\alpha}(v_s) = \alpha_s(v_s)$; and
3. For $\sigma \in S^* \setminus \{\varepsilon\}$, $\sigma = s_1 s_2 \ldots s_n$, with $s_1, \ldots, s_n \in S$, $e \in TERM(O \cup V)$, and $e_1 \in TERM(O \cup V)_{s_1}, \ldots, e_n \in TERM(O \cup V)_{s_n}$, if $e = o_{(\sigma,s)}(e_1, \ldots, e_n)$, then $Val_{s,\alpha}(e) = o_H(Val_{s_1,\alpha}(e_1), \ldots, Val_{s_n,\alpha}(e_n)) \in H_s$.

[1] The terms *binding* and *valuation* are also used in this context.

Annex B
(normative)

Net Classes

The purpose of this Annex is to define various classes of nets as a subclass of the HLPNG. Currently it defines Place/Transition nets (without capacities) which is a common form of Petri nets where tokens are simply 'black dots'. Other subclasses may include Elementary Net systems and other high-level nets.

B.1 Place/Transition Nets

A Place/Transition net graph (without capacity), **PTNG**, is a special HLPNG

$$\mathbf{PTNG} = (NG, Sig, V, H, Type, AN, M_0)$$

where

- $NG = (P, T; F)$ is a net graph
- $Sig = (S, O)$ with $S = \{Dot, Bool, Mdot\}$, $O = \{\bullet_{Dot}, {}_{Bool}, 1_{Mdot}, 2_{Mdot}, \ldots\}$
- $V = \emptyset$
- $H = (\{dot, Boolean, \mu dot\}, \{\bullet, true, (\bullet, 1), (\bullet, 2) \ldots\})$ a many-sorted algebra for the signature Sig, where $dot = \{\bullet\}$ and the obvious correspondences are made ($Dot_H = dot$, $Bool_H = Boolean$, $Mdot_H = \mu dot$, $(\bullet_{Dot})_H = \bullet$, $({}_{Bool})_H = true$, $(1_{Mdot})_H = \{(\bullet, 1)\}, (2_{Mdot})_H = \{(\bullet, 2)\}$ etc.).
- $Type : P \to \{dot\}$ is a function which assigns the type dot to all places.
- $AN = (A, TC)$ is a pair of net annotations.
  - $A : F \to B$ where $B = \{1_{Mdot}, 2_{Mdot}, \ldots\}$ is a function that annotates each arc with a syntactic 'positive integer' constant, that when evaluated becomes the corresponding singleton multiset over dot.
  - $TC : T \to \{{}_{Bool}\}$ is a function that annotates every transition with nothing (a blank) that on evaluation is the Boolean constant $true$.
- $M_0 : P \to \mu dot$.

Although this is a rather baroque definition of Place/Transition nets, it can be seen to be in one to one correspondence with a more usual definition given below.

$$\mathbf{PTNG} = (NG, W, M_0)$$

where

- $NG = (P, T; F)$ is a net graph.
- $W : F \rightarrow N^+$ is the weight function, assigning a positive integer to each arc.
- $M_0 : P \rightarrow N$ is the initial marking assigning a natural number of tokens to each place. These are represented by dots (•).

This is because:

- the transition condition is true for each transition, and hence doesn't need to be considered,
- the type of each place is the same, comprising a single value •, and hence there is no need for typing places,
- the number of dots (•) associated with each arc (Weight function) are in one to one correspondence with the positive integers, and
- the number of dots (•) associated with each place (marking) are in one to one correspondence with the Naturals.

Annex C
(informative)

High-level Petri Net Schema

C.1 Introduction

This clause provides a formal definition for a high-level net schema. It provides an abstract mathematical syntax for inscribing the graphical elements, and allows classes of systems to be specified an an abstract level.

C.2 Definition

HLPNS is a structure given by

$$\text{HLPNS} = (NG, Sig, V, Sort, An, m_0)$$

where

- $NG = (P, T; F)$ is a net graph, with
  - P a finite set of nodes, called Places;
  - T a finite set of nodes, called Transitions, disjoint from P ($P \cap T = \emptyset$); and
  - $F \subseteq (P \times T) \cup (T \times P)$ a set of directed edges called arcs, known as the flow relation;
- $Sig = (S, O)$ is a Boolean signature defined in Annex A.
- V is an S-indexed set of variables, disjoint from O.
- $Sort : P \to S$ is a function which assigns sorts to places.
- $An = (a, TC)$ is a pair of net annotations.
  - $a : F \to TERM(O \cup V)$ such that for all $(p, t), (t', p) \in F$, $a(p, t), a(t', p) \in TERM(O \cup V)_{Sort(p)}$. $TERM(O \cup V)$ is defined in Annex A. a is a function that annotates each arc with a term of the same sort as that of the associated place.
  - $TC : T \to TERM(O \cup V)_{Bool}$ is a function that annotates transitions with Boolean terms.
- $m_0 : P \to TERM(O)$ such that $\forall p \in P$, $m_0(p) \in TERM(O)_{Sort(p)}$, is the initial marking function which associates a ground term which each place. The ground term has the same sort as the place.

Annex D
(informative)

Tutorial

D.1 Introduction

High level Petri net graphs (HLPNGs) are used to model discrete event systems. A discrete event system comprises

- collections of real or abstract objects and
- discrete actions which
  - modify or consume objects from some collections and
  - create objects in other collections.

The created objects may be related to objects that are consumed. It is assumed that the collections considered have some permanent identity, irrespective of varying contents. Take, for example, the collection of coins in someone's purse, or a data base. Generally, several instances of the same object can be contained in a collection.

D.2 Net Graphs

In HLPNGs, an action is modelled by a *transition*, which is graphically represented by a rectangle. A collection is modelled by a *place*, which is graphically represented by a circle or an ellipse. Places and transitions are called the *nodes* of a *net graph*. Arrows, called *arcs*, show which places a transition operates on. Each arc connects a place and a transition in one direction. Arcs never connect a place with a place nor a transition with a transition. The graphical representations of components of a net graph are shown in figure D.1.

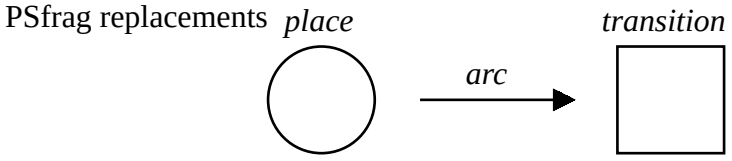


Figure D.1: Graphic conventions.

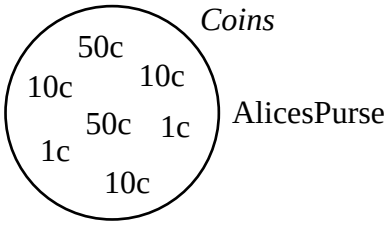
D.2.1 Places and Tokens

The objects of the system are modelled by (arbitrarily complex) data items called *tokens*. Tokens reside in places. The contents (i.e., the tokens) of a place is called the *marking* of the place. The

tokens form a collection (known in mathematics as a multiset), i.e., several instances of the same token can reside in the place. A *marking* of a net consists of the markings of each place.

Example_A in figure D.2 consists of a single place, AlicesPurse, which models that Alice's purse contains two 1 cent, three 10 cent and two 50 cent coins. The set of coins is defined in a textual part of the HLPNG called the Declarations.

The place, AlicesPurse, is typed by the set, *Coins*. This means that only coins (belonging to *Coins*) can reside in Alice's purse. In this example, the tokens correspond to coins.



Coins = {1c, 2c, 5c, 10c, 20c, 50c}

Figure D.2: *Example_A*.

Example_A is a net graph. It has neither transitions nor arcs. As no actions are modelled, nothing ever happens and nothing ever changes in this system.

When a particular instance of a HLPNG is defined, each place is defined with a special marking, called the *initial marking*, because other markings will usually evolve, once a net is executed. As a place can be marked with a large number of tokens, the initial marking may be declared textually instead of pictorially. Thus, Alice's present coin collection can be written as the initial marking,

M_0(AlicesPurse) = 2`1c + 3`10c + 2`50c

and the net graph is then drawn (admittedly in a less illustrative way) without tokens. The number of each of the coins in the purse, is separated from the value of the coin by a backprime (`), to avoid any confusion (for example when the values are themselves integers). For example, in some currencies there may be both a 5c and a 25c coin. We need to distinguish between two 5c coins and a 25c coin. If there was no separator, the notation 25c would be ambiguous.

D.2.2 Transitions

Example_B in figure D.3 models the dripping of a tap. Transition Drip can always happen, any number of times. *Example_B* is also a net, even though it has neither places nor arcs.

Figure D.3: *Example_B*.

D.2.3 Arcs

An arc from a place to a transition indicates that this transition consumes objects from the place. An arc in the opposite direction indicates that this transition produces tokens on the place. In figure D.4, *Example_C*, Alice has a smaller coin collection comprising one ten cent and two fifty cent coins. She may spend any number of coins at a time.

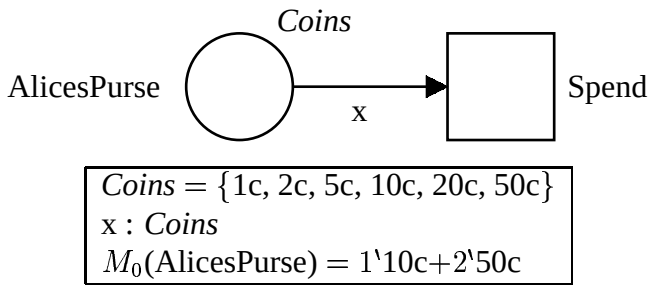


Figure D.4: *Example_C.*

Arc annotations determine the kinds and numbers of tokens that are produced or consumed. Here, the annotation x indicates that any coin (from Alice's purse) can be spent. However, it has to be declared in the textual part of *Example_C* that x denotes a variable for coins. Alice could spend: a ten cent coin; a fifty cent coin; a ten cent and a fifty cent coin; two fifty cent coins; and all her coins in one transaction, that is by the occurrence of transition spend.

D.2.4 The Net Graph

The size and position of the nodes, as well as the size and shape of the arcs, although often important for readability, are irrelevant to the mathematical description of a net, i.e., the places, transitions, and arcs of the net, the *net graph*. Informally, one might say, the net has one place, called AlicesPurse, one transition, Spend, and one arc from AlicesPurse to Spend. Formally this can be expressed as:

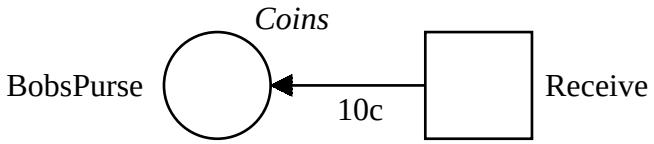
P = {AlicesPurse}
T = {Spend}
F = {(AlicesPurse, Spend)}

Traditionally, S denotes the set of places, but in this International Standard we use P (since it is written in English), T denotes the set of transitions, and F denotes the set of arcs. These letters are the initials of the German words *Stelle, Transition, Flussrelation*. Each arc is thus described as the pair consisting of its origin node and its target node.

D.3 Transition Conditions

Example_D in figure D.5 models that Bob starts with an empty purse and collects 10 cent coins.

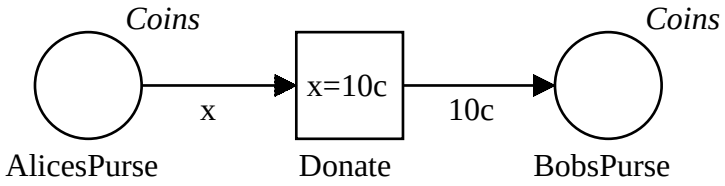
Example_D does not model where the coins may come from. It only shows what happens to Bob's purse as a consequence of an arbitrary number of occurrences of Receive.



Coins $= \{$1c, 2c, 5c, 10c, 50c$\}$
M_0(BobsPurse) $= \emptyset$

Figure D.5: *Example_D.*

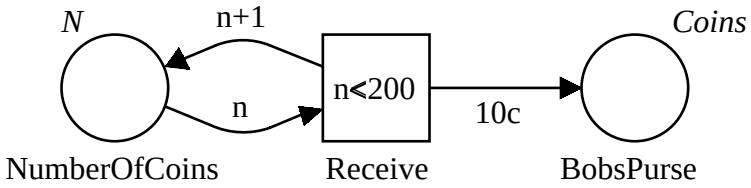
In the next example, depicted in figure D.6, Alice is ready to give Bob any of her coins. Bob, however, accepts only 10c coins from Alice.



Coins $= \{$1c, 2c, 5c, 10c, 50c$\}$
x : *Coins*
M_0(AlicesPurse) $= 1$`10c+2`50c
M_0(BobsPurse) $= \emptyset$

Figure D.6: *Example_E.*

A *transition condition* has been added, requiring that x=10c. The transition, Donate, can occur only with such variable values as fulfill the condition. If there are no appropriate (i.e., 10c) tokens in AlicesPurse, then Donate cannot occur. In a more realistic variant of *Example_D*, Bob cannot put arbitrarily many coins into his purse. *Example_F* in figure D.7 limits the number of 10 cent coins that Bob can receive to 200.



N set of natural numbers
Coins $= \{$1c, 2c, 5c, 10c, 50c$\}$
n : *N*
$<$: *N*$\times$*N*$\rightarrow$ *Bool* usual 'less than' predicate
$+$: *N*$\times$*N*$\rightarrow$ *N* arithmetic addition
M_0(BobsPurse) $= \emptyset$
M_0(NumberOfCoins) $= 1$`0

Figure D.7: *Example_F.*

D.4 Net Dynamics

Example_C can be used to illustrate the modelling of system dynamics in HLPNs. In her first action, Alice can spend any number of her coins. Let her spend a 10 cent coin. Then in the net, Spend occurs, with x having the value 10c. This *occurrence* of Spend is denoted by (Spend,{(x,10c)}). This indicates which transition occurs (Spend), and to which value each of the variables, appearing in the arc annotations around the transition, is bound. In this case only x appears, and is bound to 10c.

By the occurrence (Spend,{(x,10c)}) a new marking of the net is created:

M_1(AlicesPurse) $= 2$`50c.

The new marking is called a *reachable marking* of *Example_C*. A different marking would be reached by Alice spending a fifty cent coin. As long as Alice's purse contains coins, she can spend any of them. In the net, as long as AlicesPurse is marked with a non-empty multiset of tokens, Spend can occur with x bound to any one of the tokens in the marking of AlicesPurse. Markings reachable from reachable markings are also called reachable.

The dynamics of *Example_ C*, i.e.,

- the markings reachable in *Example_C*, as well as
- the transition occurrences performed to reach each one,

are depicted in the *reachability graph* in figure D.8.

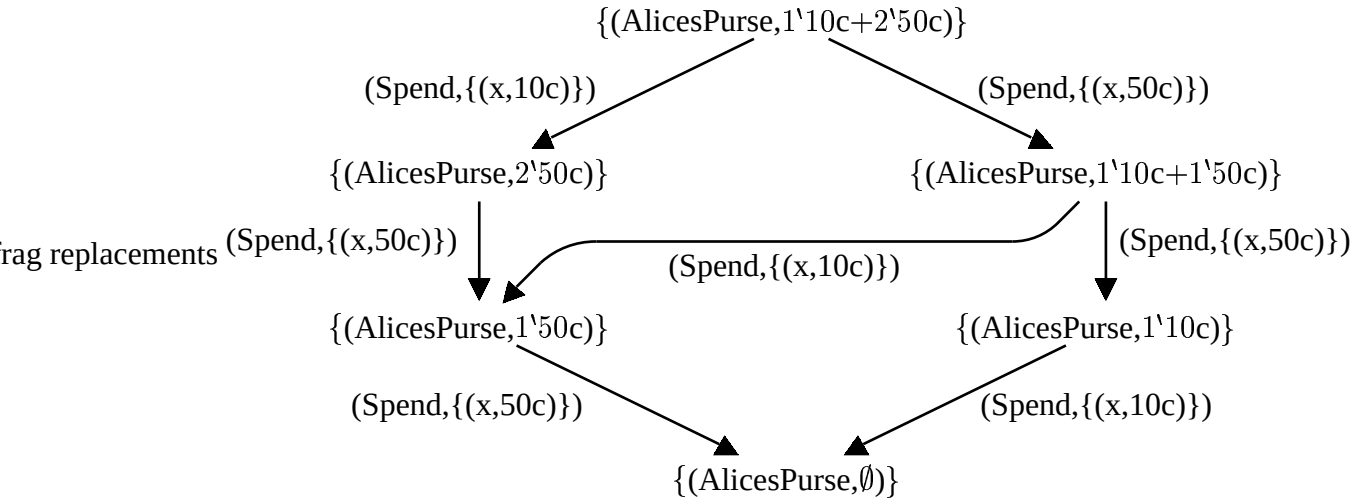


Figure D.8: Reachability graph of *Example_C.*

From this diagram, one can read, for example, the following facts about the dynamics of *Example_C*:

- If Alice spends first 10 cents and then 50 cents, or if she does it in the reverse order, then she will have 50 cents left.

- Alice can perform at most three actions.
- Every sequence of 3 actions ends with an empty purse.
- No sequence of actions (save, trivially, the empty sequence) will allow Alice to restore the contents of her purse.

All of this holds, of course, only within the range of actions considered in *Example_C*. In the reachability graph, set braces {...} are written around the pair parentheses (...) wherever usually an entire set appears in this position. Generally there are sets of place markings and sets of variable bindings to be described. It just happens that in our very simple example these are one-element sets and the {(...)} looks unnecessarily complicated.

The reachability graph for *Example_D* consists of a single infinite chain, as indicated in figure D.9. The occurrence (Receive,∅) does not mean that Bob receives no coins, but that no variable is assigned a value.

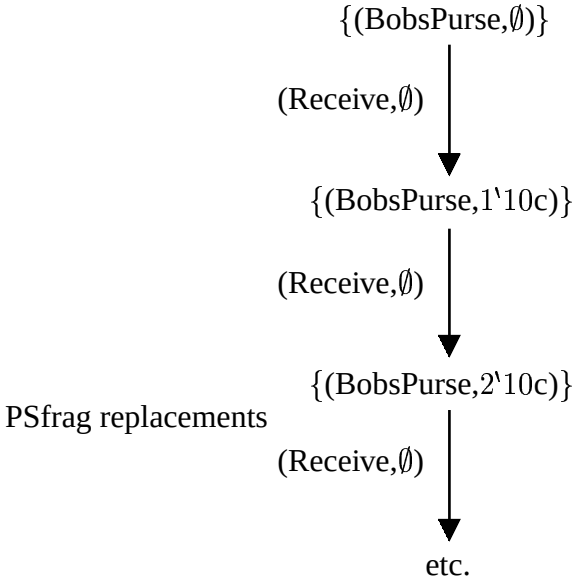


Figure D.9: Reachability graph of *Example_D*.

D.5 Flow Control Example

Two companies, A_Co and B_Co, reside in different cities. A_Co packs and sends big crates of equal size to B_Co, one by one. B_Co has a room where the crates are stored. Crates may be taken from the store room for processing (e.g., distributed to retailers, or opened and the contents consumed – it does not matter here). This procedure is modelled in figure D.10.

The people from B_Co have a problem. The store room of B_Co can only hold a certain number, say, MAX, of these crates. In order to avoid being forced either to leave crates in the street or to rent another store room, B_Co agrees with A_Co on a 'flow control protocol'.

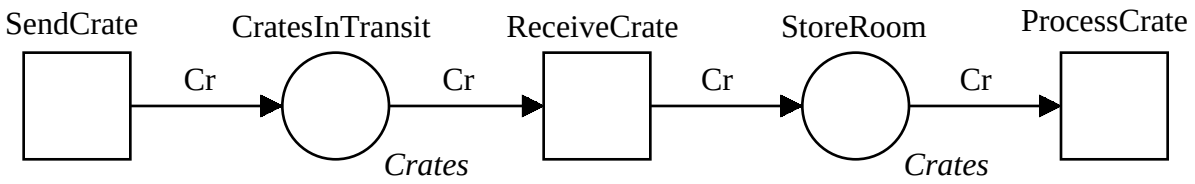


Figure D.10: Crate procedure.

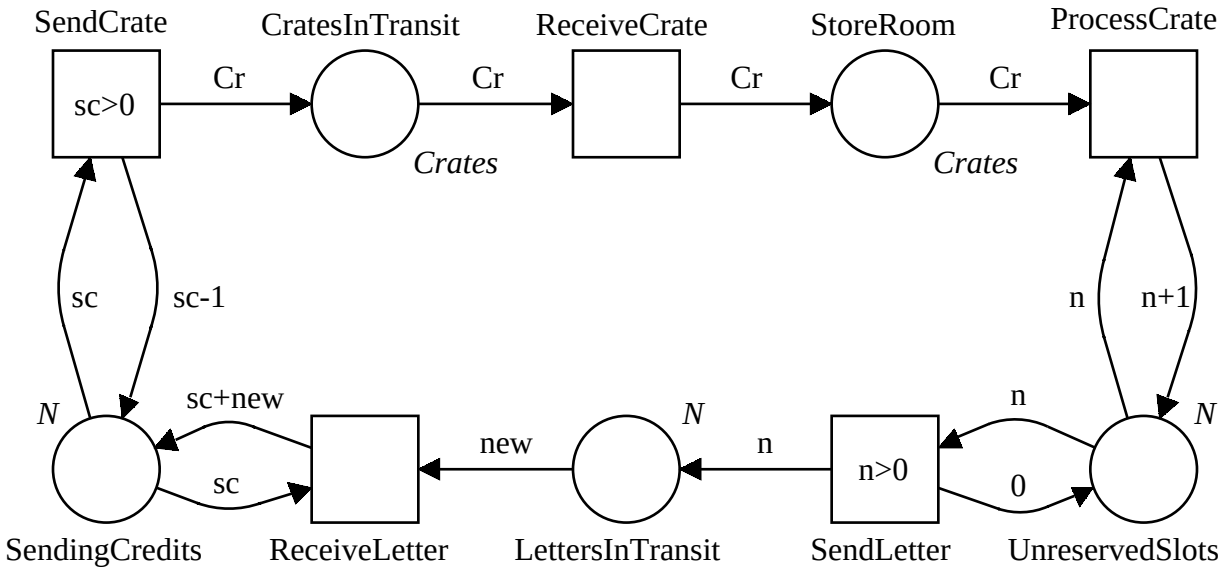
To implement the protocol, A_Co keeps a record of *SendingCredits*, while B_Co keeps a record of empty 'slots' available for placing crates in the store. Any time that there are empty slots, B_Co may reserve them and give the number of reserved (and empty) slots as sending credits for crates to A_Co. B_Co does this by sending a letter with this number to A_Co and setting the number of unreserved slots to 0. Whenever A_Co receives such a letter, it increases *Sending-Credits* by the number written in it.

Sending a crate, which is only possible if *SendingCredits* is 1 or more, lowers *SendingCredits* by 1, and processing a crate raises the number of empty and hence unreserved slots by one.

Initially, the situation is as follows: no crate or letter is in transit; the store room is empty; there is no sending credit; and all slots are empty and unreserved.

This distributed system is modelled by figure D.11.

Note that this net models infinitely many different systems. It is a *parameterized* HLPNG with a parameter, MAX, that may take any natural number as a value. Each such value *val*, substituted for MAX instantiates *Example_G* to an 'ordinary' HLPNG without parameters, *Example_G(val)*.



Crates $= \{\text{Cr}\}$
$N = \{0, 1, 2, \ldots\}$
Z is the set of integers
n, new, sc : N
MAX : N
$+ : Z \times Z \rightarrow Z$ is arithmetic addition
$- : Z \times Z \rightarrow Z$ is arithmetic substraction
$M_0(\text{CratesInTransit}) = M_0(\text{LettersInTransit}) = M_0(\text{StoreRoom}) = \emptyset$
$M_0(\text{SendingCredits}) = 1`0$
$M_0(\text{UnreservedSlots}) = 1`\text{MAX}$

Figure D.11: ***Example_G.***

Annex E
(informative)

Analysis Techniques

There are a large number of analysis techniques for Petri nets, including reachability analysis (with several state space reduction techniques and using model checking), structural analysis and invariants analysis that may be used to investigate properties of systems modelled by nets. This annex is just to alert readers of this International Standard to these possibilities. The Petri net community has published a 2 volume set in Lecture Notes in Computer Science (volumes 1491 and 1492) as a consolidation of the definitions and results in Petri nets, based on the Advanced Course held in 1996. Readers are also referred to the bibliography, for example, the second volume of Kurt Jensen's book on Coloured Petri Nets.

Bibliography

1. B. Baumgarten, *Petri-Netze, Grundlagen und Anwendungen*, Wissenschaftsverlag, Mannheim, 1990.

2. E. Best and C. Fernandez, *Notations and Terminology on Petri Net Theory*, Arbeitspapiere der GMD 195, March 1987.

3. J. Billington, "Extensions to Coloured Petri Nets", Proceedings of the Third International Workshop on Petri Nets and Performance Models, Kyoto, Japan, 11-13 December, 1989, pp. 61-70.

4. J. Billington, "Extensions to Coloured Petri Nets and their Application to Protocols", University of Cambridge Computer Laboratory Technical Report No. 222, May 1991.

5. J. Billington, "Many-sorted High-level Nets", invited paper in Proceedings of the Third International Workshop on Petri Nets and Performance Models, Kyoto, Japan, 11-13 December, 1989, pp. 166-179, also reprinted in K. Jensen, G. Rozenberg (Eds.) *High-Level Petri Nets: Theory and Application*, Springer-Verlag, 1991.

6. W. Brauer, W. Reisig, G. Rozenberg (eds), *Petri Nets: Central Models and Their Properties* Advances in Petri Nets, 1986, Part I, Proceedings of an Advanced Course, Bad Honnef, September1986. Lecture Notes in Computer Science, Volume 254 (and 255 for Part II), Springer-Verlag, 1987.

7. R. David and H. Alla, *Petri Nets and Grafcet*, Prentice Hall, 1992.

8. J. Desel and J. Esparza, *Free Choice Petri Nets*, Cambridge Tracts in Theoretical Computer Science 40, Cambridge University Press, 1995.

9. A. A. Desrochers and R.Y. Al'Jaar, *Applications of Petri Nets in Manufacturing Systems: Modelling, Control and Performance Analysis*, IEEE Press, 1995.

10. H.J. Genrich, "Predicate/Transition Nets", Lecture Notes in Computer Science, Vol. 254, pp 207-247, Springer-Verlag, 1987.

11. K. Jensen, *Coloured Petri Nets*, Volume 1: Basic Concepts, Second Edition, Springer-Verlag, 1997.

12. K. Jensen, *Coloured Petri Nets*, Volume 2: Analysis Methods, Second Edition, Springer-Verlag, 1997.

13. K. Jensen, *Coloured Petri Nets*, Volume 3: Practical Use, Springer-Verlag, 1997.

14. K. Jensen, G. Rozenberg (Eds.) *High-Level Petri Nets: Theory and Application*, Springer-Verlag, 1991.

15. E. Kindler and H. Völzer, "Flexibility in Algebraic Nets", Lecture Notes in Computer Science, Vol. 1420, pp. 345-364, Springer-Verlag, 1998.

16. C.A. Lakos and S. Christensen, “A General Systematic Approach to Arc Extensions for Coloured Petri Nets”, Lecture Notes in Computer Science, Vol. 815, pp. 338-357, Springer-Verlag, 1994.

17. T. Murata, “Petri Nets: Properties, Analysis and Applications”, Proc IEEE, Volume 77, No. 4, pp. 541-580, 1989.

18. J. L. Peterson, *Petri Net Theory and the Modeling of Systems*, Prentice-Hall, N.J., 1981.

19. W. Reisig, *Distributed Algorithms, Modeling and Analysis with Petri Nets*, Springer-Verlag, 1998.

20. W. Reisig, *Petri Nets and Algebraic Specifications*, TCS, Vol. 80, pp. 1-34, May, 1991.

21. W. Reisig, *Petri Nets, An Introduction*, EATCS, Monographs on Theoretical Computer Science, W.Brauer, G. Rozenberg, A. Salomaa (Eds.), Springer Verlag, Berlin, 1985.

22. W. Reisig and G. Rozenberg (eds), *Lectures in Petri Nets*, Lecture Notes in Computer Science, Volumes 1491 and 1492, Springer-Verlag, 1998.

23. K. Schmidt, “Parameterized Reachability Trees for Algebraic Petri Nets”, Lecture Notes in Computer Science, Volume 935, pp. 392-411, Springer-Verlag, 1995.

24. J.K. Truss, *Discrete Mathematics for Computer Scientists*, Addison-Wesley, 1991.