



ISO/IEC JTC1/SC7
Software Engineering
Secretariat: CANADA (SCC)

ISO/IEC JTC1/SC7 N1947

1998-06-21

Document Type	FCD Ballot
Title	FCD15909 Information Technology - High Level Petri Net Standard.
Source	SC7 Secretariat
Project	07.19.03
Status	Final CD Ballot
References	N1851, N1946
Action ID	ACT
Due Date	1998-10-26
Mailing Date	1998-06-26
Distribution	SC7_AG, JTC1 Sec.
Medium	Encoded Acrobat
No. of Pages	42
Disk	
Note	



ISO/IEC JTC1/SC7 FCD 15909

Date 1998-06-21	Reference number ISO/JTC 1/SC 7 N1947
Supersedes document N1793	

THIS DOCUMENT IS STILL UNDER STUDY AND SUBJECT TO CHANGE. IT SHOULD NOT BE USED FOR REFERENCE PURPOSES.

ISO/JTC 1/SC 7 Committee Title Software Engineering Secretariat: Standards Council of Canada (SCC)	Circulated to P- and O-members, and to technical committees and organizations in liaison for: X voting by (P-members only) 1998-10-21 Please return all votes and comments in electronic form directly to the SC 7 Secretariat by the due date indicated.
---	---

ISO/IEC JTC1/SC7

Title: FCD15909 Information Technology - High Level Petri Net Standard.

Project: 07.19.03

Introductory note: See page ii of the document

Medium: Encoded Acrobat

No. of pages: 46



Vote on FCD 15909

Date of circulation 1998-06-21	Reference number ISO/JTC 1/SC 7 N1947
Closing date 1998-10-21	

ISO/JTC 1/SC 7 Committee Title Software Engineering Secretariat: Standards Council of Canada (SCC)	Circulated to P-members of the committee for voting Please return all votes and comments in electronic form directly to the SC 7 Secretariat by the due date indicated.
---	--

ISO/IEC JTC1/SC7

Title: FCD15909 Information Technology - High Level Petri Net Standard.

Project: 07.19.03

Vote:

- APPROVAL OF THE DRAFT AS PRESENTED
- APPROVAL OF THE DRAFT WITH COMMENTS AS GIVEN ON THE ATTACHED
 - general:
 - technical:
 - editorial:
- DISAPPROVAL OF THE DRAFT FOR REASONS ON THE ATTACHED
 - Acceptance of these reasons and appropriate changes in the text will change our vote to approval
- ABSTENTION (FOR REASONS BELOW):

P-member voting:

National Body (Acronym)

Date:

YYYY-MM-DD

Submitted by:

Your Name

Note: do NOT send this form when voting by e-mail. Simply cast your vote (with comments where appropriate) into a normal email message and send to sc7@qc.bell.ca.

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

Final Committee Draft ISO/IEC 15909
June 19, 1998
Version 4.0

Editor's Foreword

Previous drafts of this standard have been discussed in WG11 meetings during 1995 and 1996. A Working Draft was circulated to SC7 national bodies in February 1997, and the first Committee Draft was balloted in October 1997. The letter ballot summary is recorded in SC7N1851. This document incorporates the comments agreed at an editing meeting at the WG11 meeting in Johannesburg, 25-29 May 1998. The disposition of comments on CD 15909 has been sent to the secretary of SC7 for circulation to National Bodies.

National Bodies should send comments to the SC7 secretariat by email to sc7@QC.Bell.ca. WG11 experts may send their comments to the WG11 secretary, Mr Tony Williamson, at WILLIAMSONJA%AM5@mr.nawcad.navy.mil. It would be appreciated if all comments could also be copied to the editor, Jonathan Billington, at j.billington@unisa.edu.au.

The major differences from CD15909 (2 October 1997) are:

1. The technical contents of Clause 4 has been moved to normative Annex A, on the suggestion of the UK.
2. The subclass of Petri nets (without capacities) has been defined in normative Annex B.
3. The symbols used in the definitions have been changed according to the editor's foreword in CD15909, except that F has been retained for the flow relation (this is very well accepted, and also A is already in use for arc annotation, and hence would also need changing - not worth it.)
4. The introductory paragraph to clause 7 has been moved to clause 6, and replaced by a more appropriate introduction.
5. Several technical improvements have been made to clause 4 (now Annex A), stimulated by German comments.

6. The Conformance clause has been updated to reflect the discussions at the last editing meeting at Johannesburg.
7. I hope I have now removed all usage of 'we' in the document.

Jonathan Billington
Editor Project 1.7.19.3 Petri nets
Email: j.billington@unisa.edu.au

Contents

0	Introduction	6
1	Scope	7
1.1	Purpose	7
1.2	Field of Application	7
1.3	Audience	7
2	Normative References	8
3	Terms and Definitions	8
3.1	Glossary	8
3.2	Abbreviations	11
4	Conventions and Notation	11
5	Semantic Model for High-level Petri Nets	11
5.1	Definition	11
5.2	Marking of HLPN	12
5.3	Enabling of Transition Modes	12
5.4	Transition Rule	12
6	Concepts Required for the High-level Petri Net Graph	13
6.1	Introduction	13
6.2	High-level Petri Net Graph components	13
6.3	Net execution	14
6.3.1	Enabling	14
6.3.2	Transition Rule	14
6.4	Graphical Concepts and Notation	15
6.5	Conditionals in Arc Expressions, and Parameters	16
7	Definition of the High-level Petri Net Graph	18
7.1	Introduction	18
7.2	Definition	18
7.3	Marking	19
7.4	Enabling	19

7.5	Transition Rule	20
8	Notation for High-level Petri Net Graphs	20
8.1	General	20
8.2	Places	20
8.3	Transitions	20
8.4	Arcs	21
8.5	Markings and Tokens	21
9	Semantics of HLPN Graph	21
10	Conformance	22
10.1	PN Conformance	23
10.1.1	Level 1	23
10.1.2	Level 2	23
10.2	HLPN Conformance	23
10.2.1	Level 1	23
10.2.2	Level 2	23
Annex A:	Mathematical Conventions (normative)	24
A.1	Sets	24
A.2	Multisets	24
A.2.1	Sum representation	24
A.2.2	Membership	24
A.2.3	Empty multiset	25
A.2.4	Cardinality and Finite Multiset	25
A.2.5	Multiset Equality and Comparison	25
A.2.6	Multiset Operations	25
A.3	Concepts from Algebraic Specification	25
A.3.1	Signatures with Variables	25
A.3.2	Natural and Boolean Signatures	26
A.3.3	Terms of a Signature with Variables	26
A.3.4	Multiset Terms	27
A.3.5	Many-sorted Algebras	27
A.3.6	Assignment and Evaluation	28

Annex B: Net Classes (normative)	30
B.1 Petri nets	30
Annex C: Tutorial (informative)	32
C.1 Introduction	32
C.2 Net Graphs	32
C.2.1 Places and tokens	33
C.2.2 Transitions	33
C.2.3 Arcs	33
C.2.4 The net graph	34
C.3 Transition conditions	35
C.4 Net Dynamics	36
C.5 A larger example: flow control	38
Annex D: Analysis Techniques (informative)	40
Bibliography	41

0 Introduction

This International Standard provides a well-defined graphical technique for the specification 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 net that this standard builds on are Predicate-Transition nets and Coloured Petri nets, that were first introduced in 1979 and have been developed during the 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 to define the semantics of data flow diagrams or directly to specify systems. The technique is particularly suited to parallel and distributed systems development as it supports concurrency.

This standard may be cited in contracts for the supply of software services, 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 a 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. 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 Petri nets (without capacities) as a restriction of the definition of Clause 7. Two informative annexes are provided: Annex C is a tutorial on the High-level Petri Net Graph; and Annex D provides pointers to analysis techniques for High-level Petri Nets. A bibliography concludes the 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. The 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 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

The standard may be applied to a broad range of systems, including information systems, operating systems, databases, communication protocols, computer hardware architectures, security systems, control systems, fault-tolerant systems, manufacturing systems, defence command and control, business processes, banking systems, chemical processes, nuclear waste systems and telecommunications.

1.3 Audience

The standard is written as a reference for systems analysts, designers, developers, maintainers and procurers, and for Petri net tool developers and standards developers.

2 Normative References

None.

3 Terms and Definitions

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 (on variable substitution) to be 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: see Step

3.1.8 Declarations: A set of statements which define the sets, constants, parameter values, typed variables and functions required for defining the inscriptions on a high-level 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 placed 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 placed 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 types to places,

and modes (types) to transitions; *Pre* function determining 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 (ie non-negative integer) which describes the number of repetitions of a set element in a corresponding multiset.

3.1.15.2 Multiset cardinality: The cardinality of a multiset, is the sum of the multiplicities of each of the members of the multiset.

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

3.1.16.1 Node (of a net): A vertex of the net graph.

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: An 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 Boolean, and one of the constants is $true_{Bool}$.

3.1.22.2 Natural signature: A signature where one of the sorts corresponds to the Natural numbers, and at least one natural constant is included.

3.1.22.3 Signature with Variables: A signature that includes a set of variable names, as well as the set of sorts and the set of operators.

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 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 built from a signature and comprising constants, variables and operators.

3.1.24.1 Closed term: A term comprising constants and operators but no variables.

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 their reduction to simplest form.

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

3.1.26 Transition: A node of a net, taken from the transition kind, 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: 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.

3.2 Abbreviations

3.2.1 HLPN: High-level Petri Net

3.2.2 HLPNG: High-level Petri Net Graph

3.2.3 PN: Petri Net

3.2.4 PNG: Petri Net Graph

4 Conventions and Notation

This International Standard uses the notation for sets, multisets and universal algebra defined in Annex A. The notion of multisets is required for clauses 5, 6, 7, 8 and 9. An understanding of many-sorted signatures with variables and many-sorted algebras provided in Annex A is required for clauses 7 and 9.

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

The graphical notation used in clause 6.3 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 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.

5.2 Marking of HLPN

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

5.3 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

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

Thus a multiset of transition modes is enabled if there are enough tokens on the input places to satisfy the linear combination of the pre maps for each transition mode in T_μ .

5.4 Transition Rule

Given that a 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 $M[T_\mu]M'$ 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 (up to the maximum).

6 Concepts Required for the High-level Petri Net Graph

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 a mathematical syntax for inscribing the graphical elements. The HLPNG is an abstract syntactical representation for HLPNs.

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 at this syntactical level 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 C.

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 and function images (eg $f(x)$). The variables are typed. The expressions are evaluated by substituting values for 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 (eg $x < y$) 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 or substitution of values for the *transition's variables*, that satisfies the transition condition (ie 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

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.

Declarations

$A = \{1, 2, 3, 4\}$
 $B = \{3, 4, 5, 7\}$
 $<: Z \times Z \rightarrow \text{Boolean}$ arithmetic 'less than'
 $x : A, y : B$

Graph

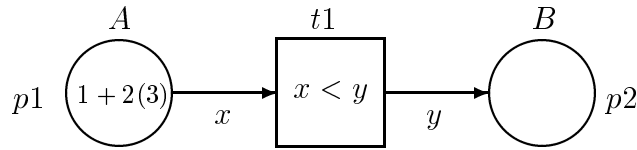


Figure 1: HLPN-Graph with a Transition Condition

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
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.

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 the next subclause.

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 square (more generally by a rectangle), 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 $M_0(p1) = 1 + 2(3)$, 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, $M_0(p2) = \emptyset$.

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 substitution for x , and the second, a substitution for y which satisfies $x < y$.

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

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

$$\begin{aligned} M(p1) &= 1 + 3 \\ M(p2) &= 5. \end{aligned}$$

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

$$\begin{aligned} M(p1) &= \emptyset \\ M(p2) &= 3 + 2(5). \end{aligned}$$

6.5 Conditionals in Arc Expressions, and Parameters

The HLPN graph of figure 2 uses a variant of the readers/writers problem to illustrate many of the features of a HLPN graph 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. An HLPN graph 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 ordinary tokens (represented by a dot \bullet) and *Access* is empty. The marking of place *Wait* is given by the set A , which is interpreted to be a multiset over A ,

Declarations

Set of Agents: $A = \{a_1, \dots, a_N\}$
 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 $[false] = 0$
 $M_0(\text{Wait}) = A$
 $M_0(\text{Control}) = L \bullet$
 $M_0(\text{Access}) = \emptyset$

Graph

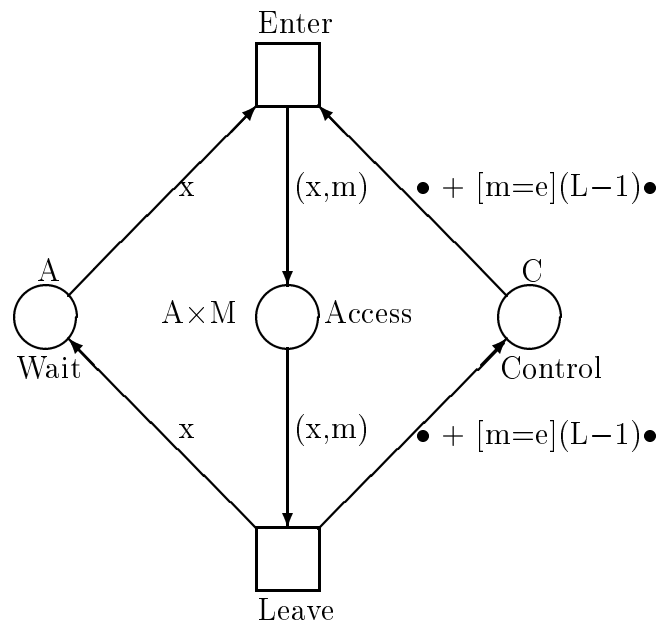


Figure 2: HLPN Graph of Resource Management

where each of the multiplicities of the agents is one. 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 $[m=e]=0$, and thus the arc expression evaluates to \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 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 integer addition and subtraction, the equality predicate and a tupling operator are primitive and do not need to be defined in the Declaration.

7 Definition of the High-level Petri Net Graph

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 at this level. The enabling and transition rules are syntactical representations of the corresponding notions in the semantic model.

7.2 Definition

A HLPNG is a structure

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

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;
- $Sig = (S, O, V)$ is a Natural-Boolean signature with variables, defined in Annex A.
- $H = (S_H, O_H)$ 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.

- $A : F \rightarrow BTERM(O \cup V)$ such that for $Type(p) = H_s$ and for all $(p, t), (t', p) \in F$, $A(p, t), A(t', p) \in BTERM(O \cup V)_s$. $BTERM(O \cup V)$ is defined in Annex A. A is a function that annotates arcs with a multiset of terms of the same sort as the carrier associated with the arc's place.
- $TC : T \rightarrow TERM(O \cup V)_{Bool}$ is a function that annotates transitions with Boolean expressions. $TERM(O \cup V)$ is defined in Annex A.
- $M_0 : P \rightarrow \bigcup_{p \in P} \mu Type(p)$ 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 correct type which each place.

A HLPNG has a net graph where the arcs are annotated by multiset terms. The coefficients of the terms are natural terms. Transitions are annotated by Boolean terms. The terms are built from a Natural-Boolean signature which has an associated many-sorted algebra. A typing function associates a carrier of the many-sorted algebra with each place. 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 : When generating multiset terms for the arc inscriptions, the co-efficients of terms are natural number terms, so that the value can depend on the values of variables and operators of other types. This allows there to be conditionals in arc expressions.

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

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

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

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, \emptyset , at the level of the signature, so that $Val_{\alpha}(\Phi) = \emptyset$.

7.5 Transition Rule

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

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


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 a matter of taste, and 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,

$$\boxed{x < y} \ t1$$

represents a transition with a name $t1$, and a transition condition, $x < y$, where both 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 a matter of taste, and 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 multiset terms. Multiset terms use the symbolic sum representation defined in the Conventions (clause A.2.1). In order to distinguish co-efficients from terms, the convention is adopted that terms are enclosed in parentheses.

8.5 Markings and Tokens

A token is a member of $\cup_{p \in P} Type(p)$. 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. Parentheses should be used to distinguish token multiplicities (Natural numbers) from token values (e.g. Integers) when required.

9 Semantics of HLPN Graph

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 there be n_t free variables associated with the arcs surrounding a transition $t \in T$. Let these have names $v_{s_1}(t), \dots, v_{s_{n_t}}(t) \in V$. In the *Sig*-Algebra, H , for all $i \in \{1, 2, \dots, n_t\}$, let the carrier corresponding to s_i , H_{s_i} , be denoted by G_i with typed variables $v_i(t) : G_i$. For all i , let $g_i \in G_i$, then

$$Type(t) = \{(g_1, \dots, g_{n_t}) \mid TC'(t)\} \text{ where}$$

$$TC'(t) = \lambda(v_1(t), \dots, v_{n_t}(t)).TC(t)(g_1, \dots, g_{n_t}).$$

Tuples which satisfy $TC(t)$ are included in $Type(t)$. (The λ -expression provides a means for formally substituting values for the variables in the Transition Condition.)

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

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 from $Type(t)$ into $\mu Type(p)$

$$Pre_{(p,t)} = \lambda(v_1(t), \dots, v_{n_t}(t)).A(p, t)$$

$$Post_{(p,t)} = \lambda(v_1(t), \dots, v_{n_t}(t)).A(t, p)$$

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 $m \in Type(t)$

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

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

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 Petri 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 $\{\bullet\}$ (ie 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, ie 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 section 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.

A.1 Sets

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

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

$Z = \{\dots, -1, 0, 1, \dots\}$ 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).

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

When A is the set of real (or complex) numbers, the parenthesis around a is required to separate the multiplicity $B(a)$ from the basis element a to avoid incorrect interpretation. Where there is no confusion the parenthesis may be dropped. If $B(a) = 1$, then it may be omitted and just a used in the sum.

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:

$$\begin{aligned} B = B1 + B2 & \text{ iff } \forall a \in A B(a) = B1(a) + B2(a) \\ B = B1 - B2 & \text{ iff } \forall a \in A (B1(a) \geq B2(a)) \wedge (B(a) = B1(a) - B2(a)) \end{aligned}$$

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

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

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 sums of scaled terms involving variables, and transitions with 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, terms and many-sorted algebras that are required for the definition of the HLPNG.

A.3.1 Signatures with Variables

A *many-sorted* (or *S-sorted*) signature with variables, Sig , is a triple:

$$Sig = (S, O, V)$$

where

- S is a set of sorts (the **names** of sets, e.g. Int for the integers);
- 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; and
- V is a set of sorted variables, called an S -sorted set of variables.

It is assumed that S , O and V 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 . (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 -cf 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} .

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 or 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.3.2 Natural and Boolean Signatures

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

Similarly, the term *Natural Signature* is used when one of the sorts is Nat , corresponding to the Naturals (N) in any associated algebra. A Natural signature also includes all the natural constants.

Editor's note Perhaps we only need the natural constant one here. We need at least one natural term to allow us to build $BTERM(O \cup V)_s$, (because of its new definition) - otherwise we could have no natural terms, if there are no natural operators, or variables.

The term *Natural-Boolean Signature* refers to a signature that is both a Natural signature and a Boolean signature, as defined above.

A.3.3 Terms of a Signature with Variables

Terms of sort $s \in S$ may be built from a signature $Sig = (S, O, V)$ 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 $s, s_1, \dots, s_n \in S$ ($n > 0$)

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. If $e_1 \in TERM(O \cup V)_{s_1}, \dots, e_n \in TERM(O \cup V)_{s_n}$ are sorted terms and $o_{(s_1 \dots s_n, s)} \in O$, is an operator, then $o_{(s_1 \dots s_n, s)}(e_1, \dots, e_n) \in TERM(O \cup V)_s$

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 of a signature with variables 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.3.4 Multiset Terms

Multiset terms are defined to allow terms to have multiplicities which are terms of sort Nat , (rather than just the natural constants). This allows, for example, the introduction of conditions into arc expressions. Multiset terms can be built inductively from a Natural signature.

Let $BTERM(O \cup V)$ denote the set of multiset terms, defined inductively as follows. For $i \in TERM(O \cup V)_{Nat}$ and $a \in TERM(O \cup V)$,

- $i(a) \in BTERM(O \cup V)$;
- for $b \in BTERM(O \cup V)$, $i \times b \in BTERM(O \cup V)$ where ‘ \times ’ represents scalar multiplication; and
- if $b_1, b_2 \in BTERM(O \cup V)$, then $(b_1 + b_2) \in BTERM(O \cup V)$, where $+$ is the multiset addition operator.

Where there is no confusion, the parenthesis and the ‘ \times ’ symbol will be dropped and juxtaposition will be used for scalar multiplication (e.g. ‘ $3(x)$ ’ will be replaced by $3x$ and $4 \times [m = e]_{Nat}(x)$ by $4[m = e]_{Nat}x$. However $4 \times 3(x)$ will not be replaced by $43x$. When the multiplicity of a term is one (eg $1x$) the convention is adopted to drop the one, so that $1x$ is represented by x . Conversely, when evaluating multiset terms (cf clause A.3.6) the parentheses and ‘ \times ’ symbols must be re-instated.

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 $o_{(s_1 \dots s_n, s)} \in O$, there is a corresponding function

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

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: $\{\dots, -1, 0, 1, \dots\}$

$Boolean = \{true, false\}$

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: $\{0, 1, \dots\}$

$Boolean = \{true, false\}$

and $lessthan : N \times N \rightarrow Boolean$.

For signatures with variables, variables are S -sorted. In the algebra, the variable is typed by the carrier corresponding to the sort.

A.3.6 Assignment and Evaluation

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

$$\alpha_s : V_s \rightarrow H_s.$$

This function may be extended to terms by considering the family of functions *assign* comprising

$$assign_s : TERM(O \cup V)_s \rightarrow H_s$$

for each sort $s \in S$. The values are determined inductively as follows. For $\sigma \in S^* \setminus \{\varepsilon\}$, $\sigma = s_1 s_2 \dots s_n$, with $s, s_1, \dots, s_n \in S$, $e \in TERM(O \cup V)$, and $e_1 \in TERM(O \cup V)_{s_1}, \dots, e_n \in TERM(O \cup V)_{s_n}$

- If $e \in V_s$ is a variable, then $assign_s(e) = \alpha_s(e)$
- For a constant, $o_s \in O$, $assign_s(o_s) = o_H \in H_s$.
- If $e = o_{(\sigma,s)}(e_1, \dots, e_n)$, then $assign_s(e) = o_H(assign_{s_1}(e_1), \dots, assign_{s_n}(e_n)) \in H_s$.

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

A multiset term is evaluated by determining the values of each of the terms (and their co-efficients) comprising a multiset term for a particular assignment to variables. This is defined inductively as follows for $a \in TERM(O \cup V)$, $i \in TERM(O \cup V)_{Nat}$ and $b, b1, b2 \in BTERM(O \cup V)$

- $Val_\alpha(i(a)) = assign_{Nat}(i)(assign(a))$
- $Val_\alpha(i \times b) = assign_{Nat}(i) \times Val_\alpha(b)$
- $Val_\alpha(b1 + b2) = Val_\alpha(b1) + Val_\alpha(b2)$

The evaluation of a multiset term results in a multiset, represented by a symbolic sum (cf clause A.2.1).

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 Petri nets (without capacities). Other subclasses may include Elementary Net systems and other high-level nets.

B.1 Petri nets

A Petri net graph **PNG** is an HLPNG

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

where

- $NG = (P, T; F)$ is a net graph
- $Sig = (S, O, \emptyset)$ with $S = \{Dot, Bool, Nat\}$, $O = \{\bullet_{Dot}, true_{Bool}, 1_{Nat}, 2_{Nat}, \dots\}$
- $H = (\{dot, Boolean, N\}, \{\bullet, true, 1, 2 \dots\})$ a many-sorted algebra for the signature Sig , where $dot = \{\bullet\}$ and the obvious correspondences are made ($Dot_H = dot$, $Bool_H = Boolean$, $Nat_H = N$, $(\bullet_{Dot})_H = \bullet$, $(true_{Bool})_H = true$, $(1_{Nat})_H = 1$ etc.).
- $Type : P \rightarrow \{dot\}$ is a function which assigns the type dot to all places.
- $AN = (A, TC)$ is a pair of net annotations.
 - $A : F \rightarrow B$ where $B = \{1_{Nat}\bullet_{Dot}, 2_{Nat}\bullet_{Dot}, \dots\}$
 - $TC : T \rightarrow \{true_{Bool}\}$ is a function that annotates every transition with the Boolean constant $true$.
- $M_0 : P \rightarrow \mu dot$.

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

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

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 (\bullet).

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, and given by a single value \bullet , and hence there is no need for typing places,
- the number of dots (\bullet) associated with each place (marking), and with each arc (Weight function), are in one to one correspondence with the Naturals.

Annex C

(informative)

Tutorial

C.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.

C.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 representation of a net graph is shown in figure 3.

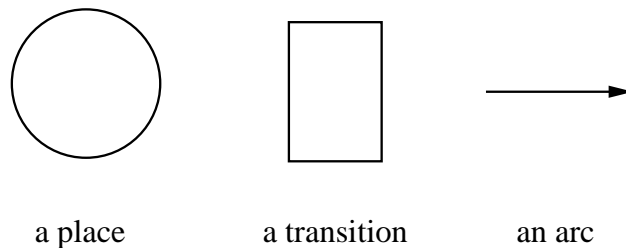


Figure 3: Graphic conventions.

C.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 4 consists of a single place, `Alice_s_purse`, 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, `Alice_s_purse`, 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.

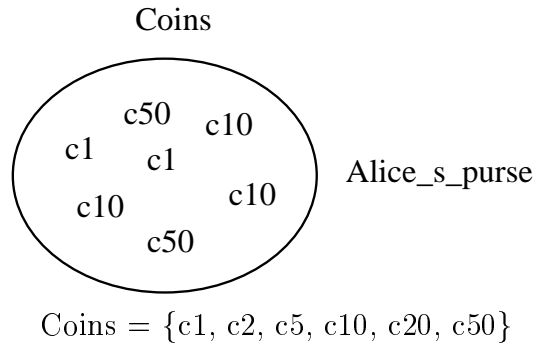


Figure 4: *Example_A*.

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,

$$M_0(\text{Alice_s_purse}) = 2c1 + 3c10 + 2c50$$

and the net graph is then drawn (admittedly in a less illustrative way) without tokens.

C.2.2 Transitions

Example_B in figure 5 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.

C.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

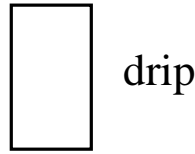
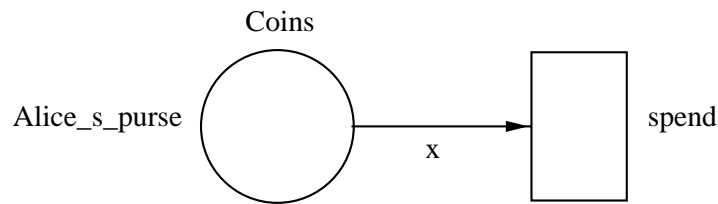


Figure 5: *Example_B*.

on the place. In figure 6, *Example_C*, Alice has a smaller coin collection. She may spend any number of coins at a time.



$$\text{Coins} = \{c1, c2, c5, c10, c20, c50\}$$

$$x : \text{Coins}$$

$$M_0(\text{Alice_s_purse}) = c10 + 2c50$$

Figure 6: *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.

C.2.4 The net graph

The size and position of the nodes, as well as the size and shape of the arcs, though 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 Alice_s_purse, one transition, spend, and one arc from Alice_s_purse to spend. Formally this can be expressed as:

$$P = \{\text{Alice_s_purse}\}$$

$$T = \{\text{spend}\}$$

$$F = \{(\text{Alice_s_purse}, \text{spend})\}$$

Traditionally, S denotes the set of places, but in this standard we use P (since it is written 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.

C.3 Transition conditions

Example_D in figure 7 models that Bob starts with an empty purse and collects 10-cent coins.

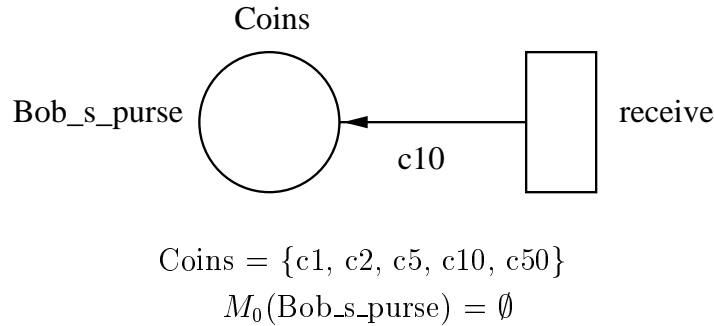


Figure 7: *Example_D*.

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*.

In the next example, depicted in figure 8, Alice is ready to give Bob any of her coins. Bob, however, accepts only 10c coins from Alice.

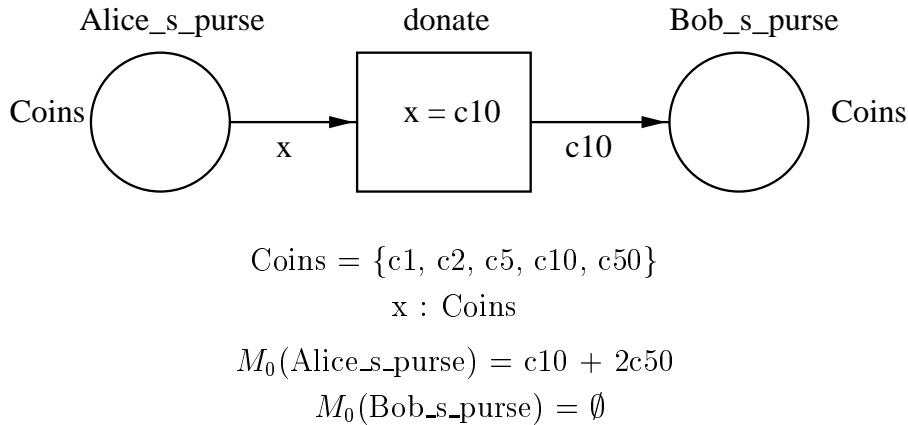
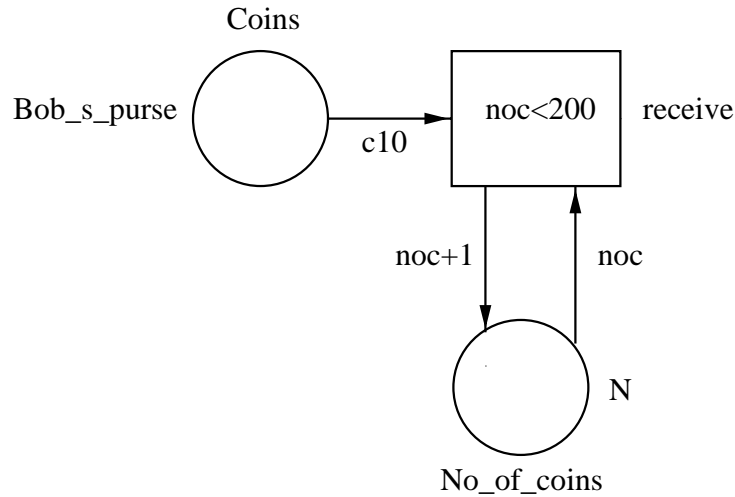


Figure 8: *Example_E*.

Now we have added a *transition condition*, requiring that $x=c10$. The transition **donate** can occur only with such variable values as fulfill the condition. If there are no appropriate, i.e. $c10$, tokens in *Alice_s_purse*, 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 9 limits the number of 10 cent coins that Bob can receive to 200.



Declarations

\mathbb{N} set of natural numbers

Coins = {c1, c2, c5, c10, c50}

noc : \mathbb{N}

$<$: $\mathbb{N} \times \mathbb{N} \rightarrow \text{Bool}$ usual “less than” predicate

$+$: $\mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ arithmetic addition

$M_0(\text{Bob_s_purse}) = \emptyset$

$M_0(\text{No_of_coins}) = 0$

Figure 9: *Example_F*.

C.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 $c10$. This *occurrence* of spend is denoted by $(\text{spend}, \{(x, c10)\})$. 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 $c10$.

By the occurrence $(\text{spend}, \{(x, c10)\})$ a new marking of the net is created: $M_1(\text{Alice_s_purse}) = 2c50$.

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 *Alice_s_purse* 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 *Alice_s_purse*. 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 10.

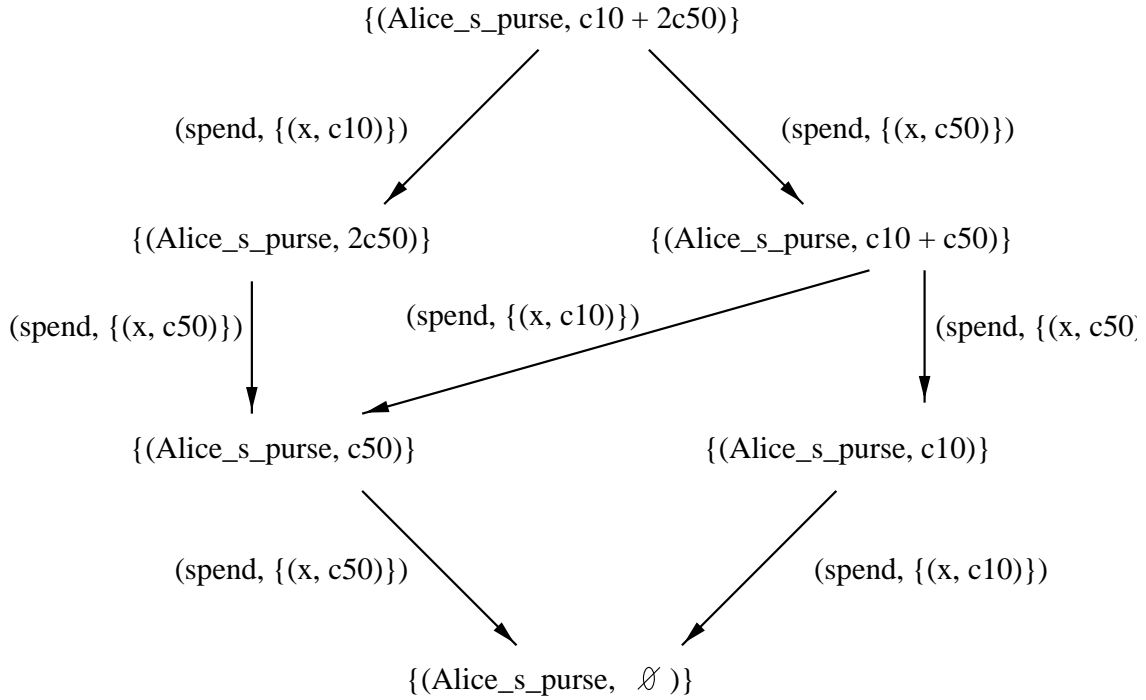


Figure 10: Reachability graph of *Example_C*.

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 $\{\dots\}$ are written around the pair parentheses (\dots) 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 $\{(\dots)\}$ looks unnecessarily complicated.

The reachability graph for *Example_D* consists of a single infinite chain, as indicated in figure 11. The occurrence $(\text{receive}, \emptyset)$ does not mean that Bob receives no coins, but that no variable is assigned a value.

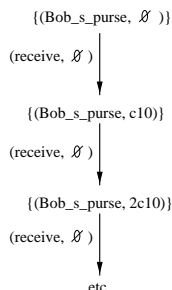


Figure 11: Reachability graph of *Example_D*.

C.5 A larger example: flow control

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 (for example, distributed to retailers, or opened and the contents consumed - it does not matter here). This procedure is modelled in figure 12.

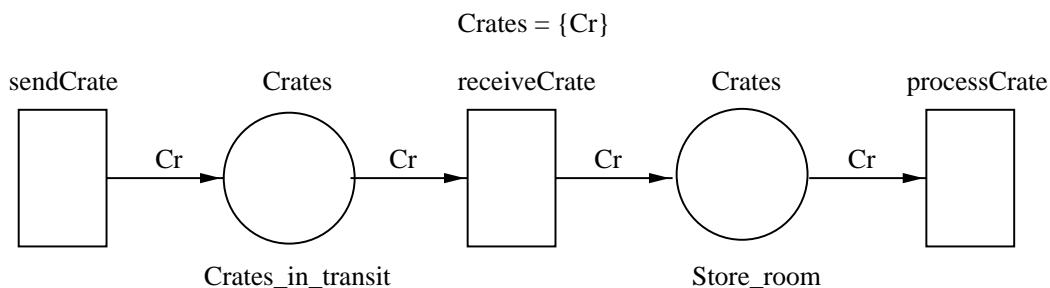


Figure 12: Crate procedure.

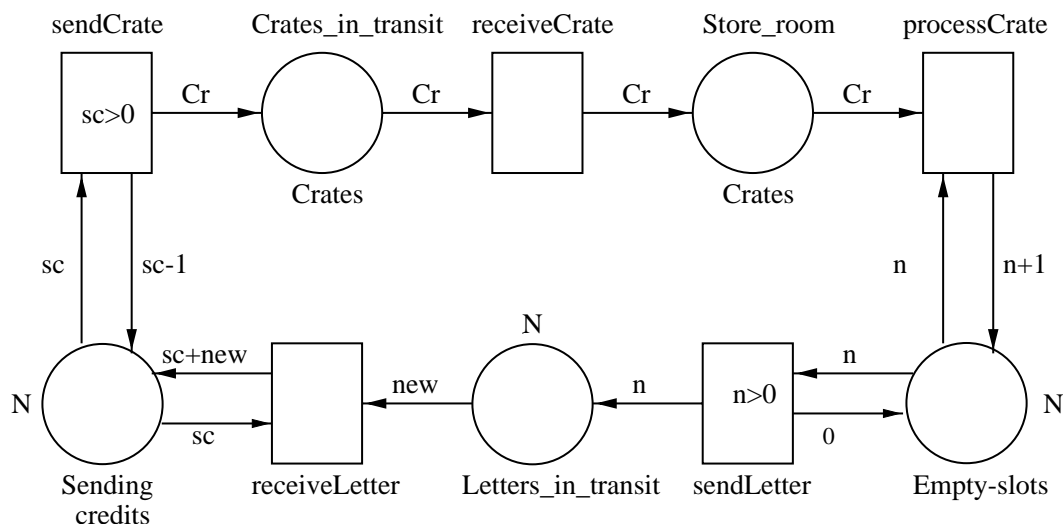
The people from B_Co have a problem. The store-room of B_Co can only hold a certain number, say, MAX, of these crates. In order to avoid being forced either to leave crates in the street or to rent another store-room, B_Co agrees with A_Co on a “flow control protocol”.

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 give the number of empty slots as sending credits for crates to A_Co. B_Co does this by sending a letter with this number and setting the number of empty slots to 0. Whenever A_Co receives such a letter, it increases *SendingCredits* 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 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.

This distributed system is modelled by figure 13.



$$\begin{aligned}
 M_0(\text{Crates_in_transit}) &= M_0(\text{Letters_in_transit}) = M_0(\text{Store_room}) = \emptyset \\
 M_0(\text{SendingCredits}) &= 0 \\
 M_0(\text{Empty-slots}) &= \text{MAX}
 \end{aligned}$$

Declarations

Crates = {Cr}
 N = {0, 1, 2, ...}
 Z is the set of integers
 n, new, sc : N
 MAX : N
 + : Z × Z → Z is arithmetic addition
 - : Z × Z → Z is arithmetic subtraction

Figure 13: Example_G.

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)*.

Annex D

(informative)

Analysis Techniques

There are a large number of analysis techniques for Petri nets, including reachability analysis (in many forms), 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 standard to these possibilities. The Petri net community plan to publish a 3 volume set in the Lecture Notes in Computer Science series as a handbook on Petri nets in 1998. This will be based on lectures given at a two week advanced course in Petri nets held in Germany 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. J. L. Peterson, "Petri Net Theory and the Modeling of Systems", Prentice-Hall, N.J., 1981.
2. W. Reisig, "Petri Nets, An Introduction", EATCS, Monographs on Theoretical Computer Science, W.Brauer, G. Rozenberg, A. Salomaa (Eds.), Springer Verlag, Berlin, 1985.
3. T. Murata, "Petri nets: properties, analysis and applications", Proc IEEE, Volume 77, No. 4, pp. 541-580, 1989.
4. B. Baumgarten, "Petri-Netze, Grundlagen und Anwendungen", Wissenschaftsverlag, Mannheim, 1990.
5. K. Jensen, "Coloured Petri Nets", Volume 1: Basic Concepts, Springer-Verlag 1992.
6. K. Jensen, "Coloured Petri Nets", Volume 2: Analysis Methods, Springer-Verlag 1994.
7. K. Jensen, "Coloured Petri Nets", Volume 3: Practical Use, Springer-Verlag 1997.
8. J. Desel and J. Esparza, "Free Choice Petri Nets", Cambridge Tracts in Theoretical Computer Science 40, Cambridge University Press, 1995
9. R. David and H. Alla, "Petri nets and Grafcet", Prentice Hall, 1992.
10. A. A. Desrochers and R.Y. Al'Jaar, "Applications of Petri nets in Manufacturing Systems: Modelling, Control and Performance Analysis", IEEE Press 1995.
11. Advanced Course on Petri Nets, Bad Honnef, West Germany, September 1986. Published in 'Advances' series, LNCS Vols 254, 255, 1987.
12. E. Best and C. Fernandez, "Notations and Terminology on Petri Net Theory", Arbeitspapiere der GMD 195, March 1987.
13. 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.
14. 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.
15. J. Billington, "Extensions to Coloured Petri Nets and their Application to Protocols", University of Cambridge Computer Laboratory Technical Report No. 222, May 1991.
16. W. Reisig, "Petri nets and algebraic specifications", TCS, Vol. 80, pp. 1-34, May, 1991.

17. J.K. Truss, “Discrete Mathematics for Computer Scientists”, Addison-Wesley, 1991.