INTERNATIONAL STANDARD

ISO/IEC 15909-1

First edition
2004-12-01

AMENDMENT 1
2010-05-15

Software and system engineering — High-level Petri nets —

Part 1: Concepts, definitions and graphical notation

AMENDMENT 1: Symmetric Nets

Ingénierie du logiciel et du système — Réseaux de Petri de haut niveau

Partie 1: Concepts, définitions et notation graphique

AMENDEMENT 1: Réseaux symétriques

Reference number
ISO/IEC 15909-1:2004/Amd.1:2010(E)

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

Amendment 1 to ISO/IEC 15909-1:2004 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*.

This amendment to ISO/IEC 15909-1 concerns the addition of a class of high-level nets, known as Symmetric Nets, to Annex B and the corresponding changes required to the Conformance Clause. Additional references related to Symmetric Nets are to be included in the Bibliography. Revised Annex B is included in full, due to some minor notational corrections in clause B.1.

Software and system engineering — High-level Petri nets —

Part 1:
Concepts, definitions and graphical notation

AMENDMENT 1: Symmetric Nets

Cover page and page 1

In the document title, replace "Software and system engineering" with "Systems and software engineering".

Page 19, Conformance

Insert the following subclause after subclause 9.1 (PN Conformance):

9.2 Conformance to Symmetric Nets

This subclause expresses the requirements for a tool implementing high-level Petri nets to conform to the Symmetric Net class.

9.2.1 Level 1

To claim Level 1 conformance to the Symmetric Net class of this International Standard, an implementation shall demonstrate that it has the semantics defined in clause 4, with the types (domains) and pre and post functions that can be derived from the Symmetric Net Graph defined in Annex B.2, by providing a mapping from the implementation's syntax to the semantic model in a similar way to that defined in clause 8.

9.2.2 Level 2

To claim Level 2 conformance to the Symmetric Net class of this International Standard, an implementation must satisfy the requirements for Level 1 conformance to the Symmetric Net class and additionally shall include the syntax of the Symmetric Net Graph defined in Annex B.2 and the notational conventions of clause 7.

Change the numbering of HLPN Conformance to subclause 9.3.

Page 25, Annex B

Replace normative Annex B with the text starting on the next page, which adds new clause B.2 to define the Symmetric Net class of High-level Petri Net Graphs (HLPNGs).

Annex B
(normative)
Net Classes

The purpose of this Annex is to define various classes of nets as subclasses of the HLPNG. It currently comprises two clauses: B.1 for Place/Transition nets (without capacities), which is a common form of Petri nets where tokens are simply 'black dots'; and B.2 for Symmetric Nets, which describes a basic form of coloured Petri nets with simple types that are amenable to efficient analysis. 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

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

where

- $NG = (P, T, F)$ is a net graph
- $Sig = (S, O)$ with $S = \{Dot, Bool, Mdot\}$, $O = \{\bullet_{Dot}, true_{Bool}, 1_{Mdot}, 2_{Mdot}, \ldots\}$
- $V = \emptyset$
- $H = (\{dot, Boolean, \mu dot\}, \{\bullet, true, 1`\bullet, 2`\bullet, \ldots\})$ a many-sorted algebra for the signature Sig, with $dot = \{\bullet\}$, $\mu dot = \{\{(\bullet, n)\} | n \in N\}$ and $H_{Dot} = dot$, $H_{Bool} = Boolean$, $H_{Mdot} = \mu dot$, $(\bullet_{Dot})_H = \bullet$, $(true_{Bool})_H = true$, $(1_{Mdot})_H = 1`\bullet$, $(2_{Mdot})_H = 2`\bullet$ etc.
- $Type : P \rightarrow \{dot, Boolean, \mu dot\}$ is a function that assigns the type dot to all places ($\forall p \in P$, $Type(p) = dot$).
- $AN = (A, TC)$ is a pair of net annotations.
  - $A : F \rightarrow \{1_{Mdot}, 2_{Mdot}, \ldots\}$ is a function that annotates each arc with a syntactic 'positive integer' constant, that when evaluated becomes the corresponding multiset over dot.
  - $TC : T \rightarrow \{true_{Bool}\}$ is a function that annotates every transition with the syntactic constant true (which by convention is omitted) that on evaluation is the Boolean value $true$.
- $M_0 : P \rightarrow \mu dot$.

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.

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

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.

B.2 Symmetric Nets

B.2.1 Introduction

Symmetric Net Graphs place restrictions on the many-sorted algebra of HLPNGs. Firstly, the carriers of the algebra (Types) are finite. Secondly, basic types are defined and then further types (products and multisets) are built from them. Basic types are classified as unordered, linearly ordered or circular. This classification depends on the functions that are associated with the type as defined below (see subclause B.2.5).

A symmetric net graph, **SNG**, is a special HLPNG

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

with the following restrictions on the signature $Sig = (S, O)$, algebra, $H = (S_H, O_H)$, typing function, $Type$, and arc annotations, AN. The set of sorts, S, is partitioned to reflect the structure of the types of Symmetric Net Graphs. The allowed operators are defined for the sorts (including explicit operators for multiset constants). The restrictions on S_H are determined by defining all the allowable types. Then the allowable set of functions are defined, which enables the set of functions comprising O_H to be determined.

B.2.2 Sorts

Symmetric Net Graphs allow the use of three kinds of basic sorts: $Usorts$, $LOsorts$ and $Csorts$ which are disjoint and where $Bool \in Usorts \cup LOsorts$. Basic sorts are defined as the union:

$BasicSorts = Usorts \cup LOsorts \cup Csorts$.

A product sort may be created for each combination of basic sorts:

$Psorts \subseteq \{PROD_\sigma \mid \sigma \in BasicSorts^* \text{ and } Length(\sigma) \geq 2\}$

where $Length$ is a function that takes a string and returns its length.

A multiset sort may be created for each basic sort and product sort:

$Msorts \subseteq \{s_{ms} \mid s \in BasicSorts \cup Psorts\}$.

Finally, there is the special dedicated sort, nat, (that is always interpreted as the Natural numbers, N, in the algebra) which is required for various operations involving $Msorts$.

Hence the set of sorts, S, is given by

$S = Usorts \cup LOsorts \cup Csorts \cup Psorts \cup Msorts \cup \{nat\}$.

B.2.3 Operators

Operators for $LOsorts$

Comparison operators are defined for each sort in $LOsorts$:
$CompOps = \{<_{(s.s,Bool)}, \leq_{(s.s,Bool)}, >_{(s.s,Bool)}, \geq_{(s.s,Bool)} \mid s \in LOsorts\}$

NOTE 1: Infix notation is used for these comparison operators.

Operators for $Csorts$:

A set of unary operators is defined for each sort in $Csorts$:
$CircularOps = \{Succ_{(s,s)}, Pred_{(s,s)} \mid s \in Csorts\}$.

Specific Operators on Basic Sorts:

The following specific operators are defined for basic sorts:

- Operators for Bool:
  $BoolOps = \{not_{(Bool,Bool)}, and_{(Bool.Bool,Bool)}, or_{(Bool.Bool,Bool)}, implies_{(Bool.Bool,Bool)}\};$

  NOTE 2: Infix notation is used for binary operators.

- A set of unary operators with output sorts in $LOsorts$:
  $PartitioningOps = \{PartitionOp_{(b,lo)} \mid b \in BasicSorts, lo \in LOsorts\};$

  NOTE 3: Symmetric nets are based on "well formed nets" (see Bibliography items 25 and 26). The intent of this operator is to allow the notion of "static subclasses" introduced for "well formed nets" to be used in Symmetric nets.

- A set of tupling operators with output sorts in $Psorts$:
  $TuplingOps = \{()_{(\sigma,PROD_\sigma)} \mid \sigma \in BasicSorts^*, PROD_\sigma \in Psorts\}.$

  NOTE 4: Tupling operators are required to allow us to write a tuple on an arc. The convention is adopted to use outfix notation, where the additional set of parenthesis is dropped, e.g. $((x, y, z))$ is written (x, y, z).

Projection Operators on Product Sorts:

A set of projection operators that select the ith component of a tuple:
$ProjectionOps = \{Proj^i_{(PROD_{(b_1...b_n)},b_i)} \mid i \in \{1, \ldots, n\}, n > 1, b_1 \ldots b_n \in BasicSorts$, and $PROD_{(b_1...b_n)} \in Psorts\}$

Operators for both Basic Sorts and Product Sorts:

- Equality Predicate:
  $EqualityOps = \{=_{(s.s,Bool)} \mid s \in BasicSorts \cup Psorts\};$

- A set of conversion operators with output sorts in $Msorts$:
  $Convert2MOps = \{`_{(nat.s,s_{ms})} \mid s \in BasicSorts \cup Psorts, s_{ms} \in Msorts\}.$

NOTE 5: As usual, the convention is adopted to use infix notation for these operators.

Operators on Multiset Sorts:

The following operators are defined for multiset sorts:

- Addition and subtraction operations:
  $MbinaryOps = \{+_{(s.s,s)}, -_{(s.s,s)} \mid s \in Msorts\};$

- Scaling Operation:
  $MscalingOps = \{*_{(nat.s,s)} \mid s \in Msorts\};$

- Predicates for equality and comparison:
  $Mpredicates = \{=_{(s.s,Bool)}, \leq_{(s.s,Bool)} \mid s \in Msorts\};$

- Cardinality operation:
  $Mcardinality = \{||_{(s,nat)} \mid s \in Msorts\}.$

NOTE 6: Infix Notation is used for multiset operations, except for cardinality which uses Outfix notation.

Constants for all sorts

A set called $Constants$ is defined which contains constants of any sort. In particular, it includes dedicated constants:

- $BoolConstants \subset Constants$ where $BoolConstants = \{true_{Bool}, false_{Bool}\};$

- $NatConstants \subset Constants$ where $NatConstants = \{0_{nat}, 1_{nat}, 2_{nat}, \ldots\}$;

  NOTE 7: Natural constants are required for multiset scaling and conversion.

- for $s \in Msorts, all_s, empty_s \in Constants$.

  NOTE 8: all_s denotes (in the algebra) a multiset with exactly one occurrence of each element of its basis set.

Set of allowed Operators:
All the operators (and constants) defined above are gathered into the set Ops_{SN}. Then $O \subseteq Ops_{SN}$, where

- the input and output sorts of each operator must be in S; and

- only one unary operator with output sort in $LOsorts$ is allowed for each basic sort: for $b1, b2 \in BasicSorts$ and $lo1, lo2 \in LOsorts$, $o_{(b1,lo1)}, o_{(b2,lo2)} \in O$ and $o_{(b1,lo1)} \neq o_{(b2,lo2)}$ implies $b1 \neq b2$.

  NOTE 9: The rationale for this restriction is to enable the use of symbolic reachability graph techniques.

B.2.4 Types

In the algebra, $H = (S_H, O_H)$, a type is associated with each sort, i.e. $S_H = \{H_s |\, s \in S\}$. The corresponding sets of types in the algebra are:

- $UnorderedTypes = \{H_s |\, s \in Usorts\}$;

- $LinearOTypes = \{H_s |\, s \in LOsorts\}$;

- $CircularTypes = \{H_s |\, s \in Csorts\}$;

- $ProductTypes = \{H_s |\, s \in Psorts\}$;

- $MultisetTypes = \{H_s |\, s \in Msorts\}$;

- $H_{nat} = N$;

- $H_{Bool} = Boolean$.

Basic types:

$BasicTypes = UnorderedTypes \cup LinearOTypes \cup CircularTypes$.

$BasicTypes$ is the set of finite types, including $Boolean$ (i.e. $Boolean \in BasicTypes$).

NOTE: Finite types include any finite range of the *Integers* (e.g. $\{0, 1, 2, 3, 4\}$) and usual enumerated types such as $rainbow = \{red, orange, yellow, green, blue, indigo, violet\}$. Compound sets, such as products, are not included.

Product types:

$ProductTypes$ is the set of Cartesian Products formed from $BasicTypes$.

$ProductTypes = \{bt_1 \times \ldots \times bt_n \,|\, bt_1, \ldots, bt_n \in BasicTypes$ and $n \in N^+ \setminus \{1\}\}$

Multiset types:

Let $MultisetTypes$ be the set of the set of multisets over each basic type or product type.

$MultisetTypes = \{\mu TYPE \,|\, TYPE \in BasicTypes \cup ProductTypes\}$

Set of allowed types:

Define $D_{SN} = BasicTypes \cup ProductTypes \cup MultisetTypes \cup \{N\}$. Then $S_H \subseteq D_{SN}$, where $Boolean \in S_H$.