ISO TC184/SC4/JWG8 N236 **Date:** 2001-05-19

ISO/WD 18629-11
Industrial automation system and integration -- Process specification language: Part 11: PSL-Core

ABSTRACT:

This document provides a description of the core of the ISO 18629 standard, which defines fundamental elements of the process specification language (PSL) aimed at structuring the semantic concepts intrinsic to the capture and exchange of process information related to discrete manufacturing.

KEYWORDS:

Manufacturing, process information, first order language, ISO 15531, manufacturing process, process language, ontology

COMMENTS TO READER:

Project Leader: M Gruninger

Address: National Institute of Science and Technology
Gaithersburg MD 20879
Telephone: +1 301-975-6536
Telefacsimile: +1 301-258-9749
Electronic mail: gruning@cme.nist.gov

Project Editor: AF Cutting-Decelle

Address : Université d'Evry / IUT
91025 Evry Cedex, France
Telephone: + 33 1 69 47 73 11
Telefacsimile: +33 1 69 47 73 06
Electronic mail: afcd@univ-savoie.fr

Contents

Foreword

ISO (the International Organisation for Standardisation) is a world-wide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organisations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardisation.

Draft International Standards adopted by technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75% of the member bodies casting a vote.

Attention is drawn to the possibility that some of the element of this part of ISO 18629 may be the subject of patents rights. ISO shall not held the responsible for identifying any or all such patent rights. ISO 18629 consists of the following parts under the general title *Industrial automation systems and integration - Process specification language*:

ISO 18629-1x series of parts consists of the following:

— Part 11: PSL-Core;

— Part 12: Outer Core;

— Part 13: Time and ordering theories;

— Part 14: Resource theories;

— Part 15: Activity performance theories.

Annex A (PSL-Core) is normative and Annex B (Intuitions for PSL-Core) is informative.

Introduction

This part of ISO 18629 and all other parts in ISO 18629 are independent of any specific process representation or model used in a given application. Collectively, they provide a structural framework for interoperability.

The International Standard ISO 18629 describes what elements inter-operable systems should encompass, but not how a specific application implements these elements. It is not the purpose of the International Standard ISO 18629 to enforce uniformity in manufacturing process representations. Objectives and design of software applications vary. Therefore the implementation of an inter-operable application must necessarily be influenced by the particular objectives and processes of each specific application. This part provides a description of the core elements of the language defined within the International Standard.

Industrial automation systems and integration -- Process specification language -- Part 11: PSL-Core

1. Scope

1.1. Scope of ISO 18629-1x series

ISO 18629 specifies a language for the representation of process information, which is a process specification language. It is composed of a lexicon, an ontology, and a grammar for process descriptions.

NOTE 1 PSL is a language for specifying manufacturing processes based on a mathematically well defined vocabulary and grammar. As such, it is different from the other languages used in the standards ISO 10303, ISO 13584, ISO 15531 and ISO 15926. In the context of an exchange of information between two processes, PSL specifies each process independently of its behaviour. For example, an object viewed as a resource within one process can be recognised as the same object even though it is viewed as a product within a second process.

NOTE 2 PSL is based on Mathematical Set Theory and Situation Calculus (see annex B). As such it follows a significantly different method of description from the method used by existing languages defined in the standard ISO 10303. The meaning of the concepts within PSL follows from a set of axioms and supporting definitions rather than from a formal set of defined terms. A set of supporting notes and examples are provided to aid the understanding of the primitive lexicon of the language.

The parts 1x of ISO 18629 specify foundational theories needed to give precise definitions and the axiomatics of the primitive concepts of ISO 18629, thus enabling precise semantic translations between dif and only iferent schemes.

The following are within the scope of ISO 18629-1x:

— the representation of the basic elements of the language;

— the provision of standardised sets of axioms that correspond to intuitive semantic primitive concepts adequate for describing basic processes;

— a set of rules to for developing other foundational theories or extension in compliance with PSL-Core.

The following are outside the scope of ISO 18629-1x:

— the representation of information involving concepts that are not part of foundational theories.

1.2. Scope of ISO 18629-11

Part 11 of ISO 18629 provides a representation of the core concepts of the language, by means of a set of axioms written in KIF and using the basic lexicon of ISO 18629. These axioms provide a syntactic representation of the ISO 18629 model theory, in that they are sound and complete with regard to the model theory.

The following is within the scope of Part 11 of ISO 18629:

— the representation of the core concepts of the language.

2. Normative references

The following standards contain provisions that, through reference in this text, constitute provisions for this part of ISO 18629. At the time of publication the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this part of ISO 18629 are encouraged to investigate the possibility of applying the most recent editions of the standards indicated below. Members of IEC and ISO maintain registers of currently valid International Standards.

— ISO 10303-1: 1994, *Industrial automation systems and integration - Product data representation and exchange - Part 1: Overview and fundamental principles.*

— ISO 10303-11: 1994, *Industrial automation systems and integration - Product data representation and exchange - Part 11: Description methods: The EXPRESS language reference manual*.

— ISO/IEC 8824-1:1995, *Information technology - Open systems interconnection - Abstract syntax notation one (ASN.1) - Part 1: Specification of basic notation*.

— ISO 13584-1:[1], *Industrial automation systems and integration - Parts library - Part 1: Overview and fundamental principles*.

— ISO 15531-1:[2], *Industrial automation systems and integration - Industrial manufacturing management data - Part 1: General overview*.

— ISO 14258: 1998, *Industrial automation systems and integration - Concepts and rules for enterprise models*.

— ISO 8879: 1986, *Information processing - Text and office systems - Standard Generalised Markup Language (SGML).*

3. Terms, definitions, and abbreviations

3.1. Terms defined in ISO 10303-1

This part of ISO 18629 makes use of the following terms defined in ISO 10303-1:

— data;

— exchange structure;

— generic resource;

— information;

— product;

[1] To be published
[2] To be published

— product data;

— product information.

3.2. Terms defined in ISO 15531-1

This part of ISO 18629 makes use of the following terms defined in ISO 15531-1:

— continuous process;

— discrete manufacturing;

— industrial process;

— manufacturing;

— manufacturing process;

— process;

— process planning;

— manufacturing facilities;

— resource;

— scheduling.

3.3. Terms defined in ISO 18629-1

— axiom;

— conservative definition;

— defined lexicon;

— extension;

— foundational theory

— grammar;

— language;

— lexicon;

— model;

— non-definitional extension;

— ontology;

— outer-core;

— primitive concept;

— primitive lexicon;

— proof theory;

— PSL-Core;

— theory;

— translation definition.

3.4. Other terms and definitions

For the purpose of this part of ISO 18629 the following definitions apply:

3.4.1
antisymmetric

3.4.1
irreflexive

3.4.1
linear ordering
A linear ordering over a set of elements is a binary relation that is transitive, irreflexive, and antisymmetric, and such that any two elements in the set are related.

3.5. Abbreviations

For the purpose of this part of ISO 18629, the following abbreviations apply:

FOL	First-Order Logic

PSL	Process Specification Language
PSL-Core	Process Specification Language, core concepts

4. Organization of ISO 18629-11

4.1. Extensions in PSL-Core

ISO 18629-11 consists of one extension:
- psl_core.th (PSL-Core)

The basic elements of the language are :

- the primitive lexicon;
- the defined lexicon with supporting definitions;
- the axioms.

Since all these concepts provide the core elements of the language, no additional theory is needed.

The meaning of the following terms follows either from the axioms or from the supporting definitions.

4.2. Primitive Nonlogical Lexicon of PSL-Core

4.2.1. Primitive Sort Relation Symbols

The nonlogical lexicon of PSL-Core contains four primitive sort relation symbols:

— object;

— activity;

— activity_occurrence;

— timepoint.

4.2.2. Primitive Function Symbols

The nonlogical lexicon of PSL-Core contains two primitive function symbols:

— beginof;

— endof.

4.2.3. Primitive Relation Symbols

The nonlogical lexicon of PSL-Core contains three primitive relation symbols:

— before;

— occurrence;

— participates-in.

4.2.4. Constant Symbols

The nonlogical lexicon of PSL-Core contains two constant symbols:

— inf-;

— inf+.

4.3. Defined Nonlogical Lexicon of PSL-Core

The nonlogical lexicon of PSL-Core contains two defined relation symbols:

— between;

— before-eq;

— between-eq;

— is-occurring-at;

— exists-at.

4.4. Theories Required by PSL-Core

No theories are required by PSL-Core.

4.5. Definitional Extensions Required by PSL-Core

No definitional extensions are required by PSL-Core.

5. Informal Semantics of ISO 18629-11

This section provides both formal and informal documentation for the PSL-Core.

5.1. Primitive Sort Relations in PSL-Core

5.1.1. (activity ?a)

Arguments: ?a ∈ ***Activity***
Informal Semantics: An activity is an entity that is associated with a set of occurrences or events.

> EXAMPLE 1 "paint-part" is an example of activity. It is the class of actions in which parts are being painted.

5.1.2. (activity_occurrence ?occ)

Arguments: ?occ ∈ ***Activity_Occurrence***
Informal Semantics: An activity occurrence is an entity that is associated with a unique activity, and that begins and ends at specific timepoints.

> EXAMPLE 2 an instance or occurrence of an activity, such as "paint-part" is an activity, painting in Maryland at 2 pm on May 25, 1998 is an activity-occurrence.

5.1.3. (timepoint ?t)

Arguments: ?t ∈ ***Timepoint***
Informal Semantics: A point in time, where all timepoints form a linear ordering with endpoints.

5.1.4. (object ?x)

Arguments: ?a ∈ ***Object***
Informal Semantics: Anything that is not a timepoint, activity, or activity occurrence.

NOTE 1 Intuitively, an OBJECT is a concrete or abstract thing that can participate in an ACTIVITY. The most typical examples of OBJECTs are ordinary, tangible things, such as people, chairs, car bodies, NC-machines, although abstract objects, such as numbers, are not excluded. OBJECTs can come into existence (e.g., be created) and go out of existence (e.g., be "used up" as a resource) at certain points in time. In such cases, an OBJECT has a begin and/or end point. Some OBJECTs, e.g., numbers, do not have finite begin and end points. In some contexts it may be useful to consider some ordinary OBJECTs as having no such points either.

NOTE 2 An ACTIVITY-OCCURRENCE is a limited, temporally extended piece of the world. Any ACTIVITY-OCCURRENCE is simply taken to be characterized chiefly by two things: its temporal extent, as determined by its begin and end POINTs (possibly at infinity), and the set of OBJECTs that participate in that ACTIVITY at some point between its begin and end POINTs.

EXAMPLE 3 The first mountain stage of the 1997 Tour de France or the eruption of Mt. St. Helen are examples of ACTIVITY-OCCURRENCE.

5.2. Primitive Functions in PSL-Core

5.2.1. (beginof ?x)

Domain Arguments: ?x ∈ ***Activity_Occurrence*** ∪ ***Object***
Range: ***Timepoint***
Informal Semantics: If ?x is an activity occurrence, then the value of this function is the timepoint at which the activity occurrence ?x begins. If ?x is an object, then the value of this function is the timepoint at which ?x becomes possible to participate in an activity. This timepoint must always be before the timepoint associated with the endof ?x.

5.2.2. (endof ?x)

Domain Arguments: ?x ∈ ***Activity_Occurrence*** ∪ ***Object***
Range: ***Timepoint***

Informal Semantics: If ?x is an activity occurrence, then the value of this function is the timepoint at which the activity occurrence ?occ ends. If ?x is an object, then the value of this function is the timepoint at which ?x becomes no longer possible to participate in an activity. This timepoint must always be after the timepoint associated with the beginof ?x.

5.3. Primitive Relations in PSL-Core

5.3.1. (occurrence ?occ ?a)

Arguments: ?occ ∈ ***Activity_Occurrence***
?a ∈ ***Activity***

Informal Semantics: ?occ is a specific occurrence of the activity ?a. Every activity occurrence is associated with a unique activity. An activity may have no occurrences or multiple occurrences.

5.3.2. (before ?t1 ?t2)

Arguments: ?t1 ∈ ***Timepoint***
?t2 ∈ ***Timepoint***

Informal Semantics: Timepoint ?t1 is earlier than ?t2 in the linear ordering over timepoints.

NOTE TIMEPOINTs are ordered by the BEFORE relation. This relation is transitive, non-reflexive, and total ordering. In PSL-Core, the time is not dense (i.e., between any two distinct TIMEPOINTs there is a third TIMEPOINT), though it is assumed that time is infinite. POINTs at infinity (INF+ and INF-) are assumed for convenience. (Denseness, of course, could easily be added by a user as an additional postulate.) Time intervals are not included among the primitives of PSL-Core, as intervals can be defined with respect to TIMEPOINTs and ACTIVITIES.

5.3.3. (participates_in ?x ?a ?t)

Arguments: ?x ∈ ***Object***
?a ∈ ***Activity***
?t ∈ ***Timepoint***

Informal Semantics: The object ?x plays some (indeterminate) role in an occurrence of the activity ?a at the timepoint ?t. An object can participate in an activity only at those timepoints at which both the object exists and the activity is occurring.

5.4. Constants in PSL-Core

5.4.1. inf-

Arguments: inf- ∈ ***Timepoint***

Informal Semantics: The unique timepoint that is before all other timepoints in the linear ordering over timepoints.

5.4.2. inf+

Arguments: inf+ ∈ ***Timepoint***

Informal Semantics: The unique timepoint that is after all other timepoints in the linear ordering over timepoints.

5.5. Defined Relations for the PSL-Core

5.5.1. (between ?t1 ?t2 ?t3)

Arguments: ?t1 ∈ ***Timepoint***
?t2 ∈ ***Timepoint***
?t3 ∈ ***Timepoint***

Informal Semantics: Timepoint ?t2 is between timepoints ?t1 and ?t3 if and only if ?t1 is before ?t2 and ?t2 is before ?t3 in the linear ordering over timepoints.

5.5.2. (beforeEq ?t1 ?t2)

Arguments: ?t1 ∈ ***Timepoint***
?t2 ∈ ***Timepoint***

Informal Semantics: Timepoint ?t1 is beforeEq timepoint ?t2 if and only if ?t1 is before or equal to ?t2 in the linear ordering over timepoints.

5.5.3. (betweenEq ?t1 ?t2 ?t3)

Arguments: ?t1 ∈ ***Timepoint***
?t2 ∈ ***Timepoint***
?t3 ∈ ***Timepoint***

Informal Semantics: Timepoint ?t2 is betweenEq timepoints ?t1 and ?t3 if and only if ?t1 is before or equal to ?t2, and ?t2 is before or equal to ?t3 in the linear ordering over timepoints.

5.5.4. (exists-at ?x ?t)

Arguments: ?x ∈ ***Object***
?t ∈ ***Timepoint***

Informal Semantics: An object ?x exists-at a timepoint ?t if and only if ?t is betweenEq its begin and end points.

5.5.5. (is-occurring-at ?a ?t)

Arguments: ?a ∈ ***Activity***
?t ∈ ***Timepoint***

Informal Semantics: An activity ?a is-occurring-at a timepoint ?t if and only if ?t is betweenEq the begin and end point of an occurrence of the activity ?a.

6. Formal Definitions and Axioms for the PSL-Core

NOTE The lexicon 'defrelation', 'exists', 'forall', 'and', 'or', 'not', '=', '⇔', and '=>' are defined in the KIF Reference Manual [4]

Definition 1. Timepoint q is between timepoints p and r if and only if p is before q and q is before r.

```
(defrelation between (?p ?q ?r) :=
  (and (before ?p ?q) (before ?q ?r)))
```

Definition 2. Timepoint p is beforeEq timepoint q if and only if p is before or equal to q.

```
(defrelation beforeEq (?p ?q) :=
  (and (timepoint ?p) (timepoint ?q)
       (or (before ?p ?q) (= ?p ?q))))
```

Definition 3. Timepoint q is betweenEq timepoints p and r if and only if p is before or equal to q, and q is before or equal to r.

```
(defrelation betweenEq (?p ?q ?r) :=
  (and (beforeEq ?p ?q)
       (beforeEq ?q ?r)))
```

Definition 4. An object exists-at a timepoint p if and only if p is betweenEq its begin and end points.

```
(defrelation exists-at (?x ?p) :=
  (and (object ?x)
       (betweenEq (beginof ?x) ?p (endof ?x))))
```

Definition 5. An activity is-occurring-at a timepoint p if and only if p is betweenEq the activity's begin and end points.

```
(defrelation is-occurring-at (?a ?p) :=
(exists (?occ)
 (and  (occurrence-of ?occ ?a)
       (betweenEq (beginof ?occ) ?p (endof ?occ)))))
```
Axiom 1. The before relation only holds between timepoints.
```
(forall (?p ?q)
(=>    (before ?p ?q)
          (and   (timepoint ?p)
       (timepoint ?q))))
```
Axiom 2. The before relation is a total ordering.
```
(forall (?p ?q)
(=>    (and   (timepoint ?p)
              (timepoint ?q))
       (or    (= ?p ?q)
              (before ?p ?q)
              (before ?q ?p))))
```
Axiom 3. The before relation is irreflexive.
```
(forall (?p) (not (before ?p ?p)))
```
Axiom 4. The before relation is transitive.
```
(forall (?p ?q ?r)
(=>    (and   (before ?p ?q)
              (before ?q ?r))
       (before ?p ?r)))
```
Axiom 5. The timepoint inf- is before all other timepoints.
```
(forall (?t)
(=> (and (timepoint ?t) (not (= ?t inf-)))
    (before inf- ?t))
```
Axiom 6. Every other timepoint is before inf+.
```
(forall (?t)
(=> (and (timepoint ?t) (not (= ?t inf+)))
    (before ?t inf+))
```
Axiom 7 . Given any timepoint t other than inf-, there is a timepoint between inf- and t.
```
(forall (?t)
(=>    (and   (timepoint ?t) (not (= ?t inf-)))
       (exists (?u)
          (between inf- ?u ?t))))
```
Axiom 8. Given any timepoint t other than inf+, there is a timepoint between t and inf+.
```
(forall (?t)
(=>    (and   (timepoint ?t)  (not (= ?t inf+)))
       (exists (?u)
          (between ?t ?u inf+))))
```
Axiom 9. Everything is either an activity, object, or timepoint.
```
(forall (?x)
(or    (activity ?x)
       (activity-occurrence ?x)
       (object ?x)
       (timepoint ?x)))
```
Axiom 10. Objects, activities, activity occurrences, and timepoints are all distinct kinds of things.
```
(forall (?x)
(and (=> (activity ?x)
         (not (or (activity-occurrence ?x) (object ?x) (timepoint ?x))))
     (=> (activity-occurrence ?x)
         (not (or (object ?x) (timepoint ?x))))
     (=> (object ?x)
         (not (timepoint ?x))))
```

Axiom 11. The occurrence relation only holds between activities and activity occurrences.

```
(forall (?a ?occ)
(=>(occurrence-of ?occ ?a)
   (and   (activity ?a)
          (activity-occurrence ?occ))))
```

Axiom 12. An activity occurrence is associated with a unique activity.

```
(forall (?occ ?a1 ?a2)
        (=>     (and    (occurrence-of ?occ ?a1)
                        (occurrence-of ?occ ?a2))
                (= ?a1 ?a2))))
```

Axiom 13. The begin and end of an activity occurrence or object are timepoints.

```
(forall (?x)
(=>    (or (activity-occurrence ?x) (object ?x))
       (and   (timepoint (beginof ?x))
              (timepoint (endof ?x)))))
```

Axiom 14. The begin point of every activity occurrence or object is before or equal to its end point.

```
(forall (?x)
(=>    (or (activity-occurrence ?x) (object ?x))
       (beforeEq (beginof ?x) (endof ?x))))
```

Axiom 15. The participates-in relation only holds between objects, activities, and timepoints, respectively.

```
(forall (?x ?a ?t)
(=>    (participates-in ?x ?a ?t)
       (and   (object ?x)
              (activity ?a)
              (timepoint ?t))))
```

Axiom 16. An object can participate in an activity only at those timepoints at which both the object exists and the activity is occurring.

```
(forall (?x ?a ?t)
(=>    (participates-in ?x ?a ?t)
       (and   (exists-at ?x ?t)
              (is-occurring-at ?a ?t))))
```

Axiom 17. Every activity-occurrence is an occurrence an activity.

```
(forall (?occ)
(=>    (activity-occurrence ?occ)
       (exists (?a)
          (and (activity ?a) (occurrence-of ?occ ?a)))))
```

7. Grammar for process descriptions

The underlying grammar used for ISO 18629 is that of KIF (Knowledge Interchange Format).

NOTE KIF is a formal language based on first-order logic developed for the exchange of knowledge among dif and only iferent computer programs with disparate representations. KIF provides the level of rigour necessary to unambiguously define concepts in the ontology.

7.1. Basic Tokens and Syntactic Categories

We first define a set of basic tokens and certain categories of expressions built up from them that will be used to define any first-order PSL language. They are defined in the following BNF.

```
<uc-letter> ::=  A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<lc-letter> ::=  a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z
<letter> ::=     <uc-letter> | <lc-letter>
<digit> ::=      0 |1|2|3|4|5|6|7|8|9
<oper> ::=       -|~| #|$|*|+|/
<punct>  ::=     _|-|~|!|@|#|$|%|^|&|*|(|)|+|=|`|:|;|'|<|>|,|.|?|/|||
                 [|]|{|}|<space>
```

An *expression* is any string of basic tokens. We define four basic categories of expression.

```
<b-con> ::=       {<lc-letter> | <digit>} {<letter> | <digit>}* {{ _ | - } {<letter> |
                  <digit>}+}*
<b-var> ::=       ?<b-con>[']
<b-func> ::=      {<oper> | <uc-letter>} {<letter> | <digit>}* {{ _ | - } {<letter> |
                  <digit>}+}*
<b-pred> ::=      {<uc-letter>} {<letter> | <digit>}* {{ _ | - } {<letter> | <digit>}+}*
<doc-string>  ::= " {<letter> | <digit> | <punct> | \" | \\ }* "
```

Thus, a <b-con> (i.e., an expression derived from the nonterminal <b-con>) is a string of alphanumeric characters, dashes, and underscores that begins with a lower case letter or digit and in which every dash and underscore is flanked on either side by a letter or digit. A <b-var> is the result of prefixing a <b-con> with a question mark and, optionally, appending a single quote (a "prime"). A <b-func> is just like a <b-con> except that it must begin with either an <oper>, a <punc>, or an upper case letter, and a <b-pred> is just like a <b-con> except that it must begin with an upper case letter. (Every <b-pred> is thus a <b-func>.) A <doc-string> is the result of quoting any string of tokens; double quotes and the backslash can be used as well as long as they are preceded by a backslash.

7.2. Lexicons

A first-order PSL language L is given in terms of a *lexicon* and a *grammar*. The lexicon provides the basic "words" of the language, and the grammar determines how the lexical elements may be used to build the complex, well-formed expressions of the language. An PSL *lexicon* λ consists of the following:

- The expressions <space>, `(`, `)`, `not`, `and`, `or`, =>, <=>, `forall`, and `exists`;
- A denumerable recursive set V_λ of <b-var>s (i.e., expressions derived from the nonterminal <b-var> in the above BNF), known as the *variables* of λ;

- A recursive set C_λ of <b-con>s, known as the *constants* of λ which includes at least the strings `inf-`, `inf+`, `max-`, and `max+`.
- A recursive set F_λ of <b-func>s, known as the *function symbols* of λ, which includes at least the strings `beginof` and `endof`.
- A recursive set P_λ of <b-pred>s known as the *predicates* of λ, which includes at least the strings = `activity`, `activity-occurrence`, `object`, `timepoint`, `is-occurring-at`, `occurrence-of`, `exists-at`, `before`, and `participates-in`.

7.3. Grammars

Given an PSL lexicon λ, the *grammar based on* λ is given in the following BNF.

```
<con> ::=           a member of Cλ
<var>  ::=          a member of Vλ
<func> ::=          a member of Fλ
<pred> ::=          a member of Pλ
<term> ::=          <atomterm> | <compterm>
<atomterm>  ::=     <var> | <con>
<compterm> ::=      (<func> <term>)
<sentence> ::=      <atomsent> | <boolsent> | <quantsent>
<atomsent>  ::=     (<pred> <term>+) | (= <term> <term>)
<boolsent> ::=      (not <sentence>) | (and <sentence> <sentence>+) | (or <sentence>
                    <sentence>+) | (=> <sentence> <sentence>) | (<=> <sentence>
                    <sentence>)
<quantsent> ::=     ({forall|exists} {<var>|(<var>+[ : <sentence>])} <sentence>)
<psl-sentence> ::=     <sentence>
```

8. Conformance to PSL-Core

8.1. Conformance of Ontologies

An ontology is in conformance with PSL-Core if and only if it is consistent with the axioms and definitions of PSL-Core in Clause 6.

8.2. Conformance of Process Descriptions

A process description is in conformance with PSL-Core if and only if the ontology of the process description is in conformance with PSL-Core and the grammar of the process description satisfies Clause 7.

Annex A: Fundamental Principles for PSL-Core

(Informative)

The primary component of PSL is its terminology for classes of processes and relations for processes and resources, along with definitions of these classes and relations. Such a lexicon of terminology along with some specification of the meaning of terms in the lexicon constitutes what is known as an ontology.

NOTE for further information, see [3]

In the case of PSL, this will be the PSL ontology for processes. The focus of the ontology is not only on the terms, but also on their meaning. It is possible to include an arbitrary set of terms in the ontology, but they can only be shared if there is an agreement on their meaning. It is the semantics of the terms that is being shared, *not simply* the terms.
The challenge is that some framework is needed for making the meaning of the terminology for ontologies explicit. Any intuitions that are implicit are a possible source of ambiguity and confusion. For the PSL ontology, it is necessary to provide a rigorous mathematical characterization of process information as well as precise expression of the basic logical properties of that information in the PSL language. In providing the ontology, three notions are therefore specified :

— language
— model theory
— proof theory (axioms and definitions)

A language is a lexicon (a set of symbols) and a grammar (a specification of how these symbols can be combined to make well-formed formulae). The lexicon consists of logical symbols (such as connectives, variables, and quantifiers) and nonlogical symbols. For PSL, the nonlogical part of the lexicon consists of expressions (constants, function symbols, and predicates) that refer to everything needed to describe processes.

The underlying language used for PSL is KIF (Knowledge Interchange Format). Briefly stated, KIF is a formal language based on first-order logic developed for the exchange of knowledge among different computer programs with disparate representations. KIF (Annex A) provides the level of rigor necessary to define concepts in the ontology unambiguously, a necessary characteristic to exchange manufacturing process information using the PSL Ontology.

The model theory of PSL provides a rigorous mathematical characterization of the semantics of the terminology of PSL. The objective is to identify each concept in the ontology with an element of some mathematical structure, such as a set with additional structure (e.g. lattices, linear orderings, vector spaces). The underlying theory of the mathematical structure then becomes available as a basis for reasoning about the terms of the language and their relationships, so that the set of models constitutes the formal semantics of the ontology.

NOTE 1 The word "model" is used, in logic, in a way that differs from the way it is used in most scientific and everyday contexts [4] : if a sentence is true in a certain interpretation, it is possible to say that the interpretation is a model of the sentence. The kind of semantics presented here is often called model-theoretical semantics.

NOTE 2 Correspondence theory of truth : a statement in some language (whether a natural language learnt and spoken by human beings or an artificial language devised for specific purposes) is true if and only if it corresponds to some state-of-affairs.

NOTE 3 Generally, a model-theoretic interpretation has two parts :
- first, there is a model that represents precisely what events, properties and relations make up the situation being modelled. The model provides a description of the denotations of all the basic expressions in the object language. This reflects the fact that human beings make statements in order to convey information about some state-of-affairs. Although it is possible to know the meaning of a sentence without knowing what specific situation is being talked about (by knowing its truth-conditions), the sentence conveys no information unless it is associated with particular individuals and the relations that hold between them ;
- the second part of the theory provides the rules for interpreting expressions in the object language with respect to any arbitrary model. In other words, the model theory provides a specification of the truth-conditions of the sentences in the object language. The truth-conditions as specified by the model theory hold independently of particular models, but the interpretation of particular sentences may be carried out only with respect to some model or other. Not all possible models are interpretable according to a particular model theory and there is thus a two-way relation between the theory and the model.

EXAMPLE Several models used for PSL are based on set theory. The denotations of expressions are thus modelled as sets containing various sorts of concepts or as mathematical functions over these. The model theory uses fundamental notions from set theory, such as set membership, union and intersection, etc., to specify how the denotations of composite expressions can be constructed from these. Ultimately, the notion of truth is characterised according to the axioms of set theory, thus providing the rigorous and formal meta-language for their interpretation.

The proof theory of PSL provides a set of axioms (sentences in first-order logic) for the interpretation of concepts in the ontology. It is useful to distinguish two types of sentences in this set: core theories and definitions. A core theory is a set of predicates, function symbols, and individual constants representing the primitive concepts of the ontology, together with some axiomatization. Primitive concepts are those for which there are no definitions; the intended interpretations of these concepts are specified using the axioms in the core theories. For these terms, it is necessary to describe the set of models corresponding to the intuitions that exist for them. Then axioms are written that are sound and complete with respect to the set of models -- every interpretation that is consistent with the axioms is a model in the set, and any model in the set is an interpretation consistent with the axioms.

All other terms in the ontology outside the lexicon of the core theories are given definitions using the set of primitive concepts axiomatized in the core theories. These definitions are known as conservative definitions, since they do not add to the expressive power of the core theories ; that is, any sentence that can be deduced with the definitions, can also be deduced using the core theories alone. Since all of these terms are defined using the primitives, the set of models for defined concepts can be defined using the models of the core theories. It is therefore possible to characterize the semantics of the definitions using the classes of models that have already been specified for the core theories.

Bibliography

[1] - Booch G., Rumbaugh J., Jacobson I., The Unified Modelling Language User Guide, Addison-Wesley, 1999
[3] – Fox, M., et al, An Organisation Ontology for Enterprise Modelling: Preliminary Concepts", Computers in Industry, 1996, Vol. 19, pp. 123-134.
[4] – Genesereth, M., Fikes, R.: Knowledge Interchange Format (Version 3.0) - Reference Manual, 1992, Computer Science Dept., Stanford University, Stanford, CA.
[5] – Knutilla, A., et al., Process Specification Language: An Analysis of Existing Representations, NISTIR 6133, 1998, National Institute of Standards and Technology, Gaithersburg, MD.
[6] – Lee, J., et al, "The PIF Process Interchange Format and Framework", The Knowledge Engineering Review, Vol. 13(1), pp. 91-120, Special Issue on "Putting Ontologies to Use" (eds. Uschold, M. and Tate, A.), Cambridge University Press.
[7] – Schlenoff, C., Knutilla, A., Ray, S., Unified Process Specification Language: Requirements for Modelling Processes: NISTIR 5910, 1996, National Institute of Standards and Technology, Gaithersburg, MD.
[9] – Uschold, M. and Gruninger M., Ontologies: Principles, Methods, and Applications, Knowledge Engineering Review, 1996, Vol. 11, pp. 96-137.

Index