INTERNATIONAL STANDARD

ISO 18629-12

First edition
2005-10-01

Industrial automation systems and integration — Process specification language —

Part 12: **Outer core**

Systèmes d'automatisation industrielle et intégration — Langage de spécification de procédé —

Partie 12: Noyau externe

Reference number
ISO 18629-12:2005(E)

Published in Switzerland

Content

Page

Figures

Foreword

The International Organisation for Standardisation (ISO) is a worldwide 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 standardization.

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

Draft International Standards (DIS) 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 elements of this part of ISO 18629 may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 18629-12 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration,* Sub-committee SC4, *Industrial data*

A complete list of parts of ISO 18629 is available from the Internet.

http://www.tc184-sc4.org/titles

Introduction

ISO 18629 is an International Standard for the computer-interpretable exchange of information related to manufacturing processes. Taken together, all the parts contained in the ISO 18629 Standard provide a generic language for describing a manufacturing process throughout the entire production process within the same industrial company or across several industrial sectors or companies, independently from any particular representation model. The nature of this language makes it suitable for sharing process information related to manufacturing during all the stages of a production process.

This part provides a description of the core elements of the language defined within the International Standard.

This part of ISO 18629 and all other parts in ISO 18629 are independent of any specific process representation or model proposed in a software application in the domain of manufacturing management. Collectively, they provide a structural framework for improving the interoperability of these applications.

Industrial automation systems and integration — Process specification language – Part 12: PSL Outer core

1 Scope

This part of ISO 18629 provides a representation of the concepts that belong to the Outer core of the language, through a set of axioms and definitions. These axioms provide an axiomatization of the semantics for terminology in the ISO 18629 Outer core.

The following are within the scope of this part of ISO 18629:

— the aggregation of activities and sub-activities;

— discrete activities and state;

— constraints on the occurrence of activities;

— concurrency for primitive activities;

— complex activities;

— occurrences of complex activities.

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 8824-1, *Information technology — Abstract Syntax Notation One (ASN.1): Specification of basic notation*

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

ISO 15531-1: *Industrial automation systems and integration – Industrial manufacturing management data – Part 1: General overview.*

ISO 18629-1: *Industrial automation systems and integration – Process specification language – Part 1: Overview and basic principles*

ISO 18629-11: *Industrial automation systems and integration – Process specification language – Part 11: PSL core*

3 Terms, definitions, and abbreviations

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply:

3.1.1
axiom
well-formed formula in a formal language that provides constraints on the interpretation of symbols in the lexicon of a language

[ISO 18629-1]

3.1.2
conservative definition
definition that specifies necessary and sufficient conditions that a term shall satisfy and that does not allow new inferences to be drawn from the theory

[ISO 18629-1]

3.1.3
core theory
set of predicates, function symbols and individual constants, associated with some axioms, the primitive concepts of the ontology

[ISO 18629-1]

3.1.4
data
a representation of information in a formal manner suitable for communication, interpretation, or processing by human beings or computers

[ISO 10303-1]

3.1.5
defined lexicon
set of symbols in the non-logical lexicon which denote defined concepts

NOTE Defined lexicon is divided into constant, function and relation symbols.

EXAMPLE terms with conservative definitions.

[ISO 18629-1]

3.1.6
definitional extension
extension of PSL-Core that introduces new linguistic items which can be completely defined in terms of the PSL-Core

NOTE: Definitional extensions add no new expressive power to PSL-Core but are used to specify the semantics and terminology in the domain application.

[ISO 18629-1]

3.1.7
extension
augmentation of PSL-Core containing additional axioms

NOTE 1 The PSL-Core is a relatively simple set of axioms that is adequate for expressing a wide range of basic processes. However, more complex processes require expressive resources that exceed those of the PSL-Core. Rather than clutter the PSL-Core itself with every conceivable concept that might prove useful in describing one process or another, a variety of separate, modular extensions need to be developed and added to the PSL-Core as necessary. In this way a user can tailor the language precisely to suit his or her expressive needs.

NOTE 2 All extensions are core theories or definitional extensions.

[ISO 18629-1]

3.1.8
grammar
specification of how logical symbols and lexical terms can be combined to make well-formed formulae

[ISO 18629-1]

3.1.9
idempotent
element (a) of a set under an internal binary operation (OP) with respect to which : a OP a = a

NOTE a function is idempotent when every element of the definition domain is idempotent with respect to it

EXAMPLE set-theoretic intersection and union are idempotent, since $S \cap S = S$ and $S \cup S = S$.

3.1.10
information
facts, concepts, or instructions

[ISO 10303-1]

3.1.11
interpretation
universe of discourse and assignment of truth values (TRUE or FALSE) to all sentences in a theory

NOTE See annex C for an example of an interpretation.

3.1.12
language
combination of a lexicon and a grammar

[ISO 18629-1]

3.1.13
lexicon
set of symbols and terms

NOTE The lexicon consists of logical symbols (such as Boolean connectives and quantifiers) and non-logical symbols. For ISO 18629, the non logical part of the lexicon consists of expressions (constants, function symbols, and relation symbols) chosen to represent the basic concepts of the ontology.

[ISO 18629-1]

3.1.14
manufacturing
function or act of converting or transforming material from raw material or semi-finished state to a state of further completion

[ISO 15531-1]

3.1.15
manufacturing process
structured set of activities or operations performed upon material to convert it from the raw material or a semifinished state to a state of further completion

NOTE Manufacturing processes may be arranged in process layout, product layout, cellular layout or fixed position layout. Manufacturing processes may be planned to support make-to-stock, make-to-order, assemble-to-order, etc., based on strategic use and placements of inventories.

[ISO 15531-1]

3.1.16
model
combination of a set of elements and a truth assignment that satisfies all well-formed formulae in a theory

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: 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 A model is typically represented as a set with some additional structure (partial ordering, lattice, or vector space). The model then defines meanings for the terminology and a notion of truth for sentences of the language in terms of this model. Given a model, the underlying set of axioms of the mathematical structures used in the set of axioms then becomes available as a basis for reasoning about the concepts intended by the terms of the language and their logical relationships, so that the set of models constitutes the formal semantics of the ontology.

[ISO 18629-1]

3.1.17
ontology
a lexicon of specialised terminology along with some specification of the meaning of terms in the lexicon

NOTE 1: structured set of related terms given with a specification of the meaning of the terms in a formal language. The specification of meaning explains why and how the terms are related and conditions how the set is partitioned and structured.

NOTE 2: The primary component of a process specification language such as ISO 18629 is an ontology The primitive concepts is the ontology according to ISO 18629 are adequate for describing basic manufacturing, engineering, and business processes.

NOTE 3: The focus of an ontology is not only on terms, but also on their meaning. An arbitrary set of terms is included in the ontology, but these terms can only be shared if there is an agreement about their meaning. It is the intended semantics of the terms that is being shared, not simply the terms.

NOTE 4: Any term used without an explicit definition is a possible source of ambiguity and confusion. The challenge for an ontology is that a framework is needed for making explicit the meaning of the terms within it. For the ISO 18629 ontology, it is necessary to provide a rigorous mathematical characterisation of process information as well as a precise expression of the basic logical properties of that information in the ISO 18629 language.

[ISO 18629-1]

3.1.18
Outer core
set of core theories that are extensions of PSL-Core and that are so generic and pervasive in their applicability that they have been put apart

NOTE In practice, extensions incorporate the axioms of the Outer core.

[ISO 18629-1]

3.1.19
primitive concept
lexical term that has no conservative definition

[ISO 18629-1]

3.1.20
primitive lexicon
set of symbols in the non-logical lexicon which denote primitive concepts

NOTE Primitive lexicon is divided into constant, function and relation symbols.

[ISO 18629-1]

3.1.21
process
structured set of activities involving various enterprise entities, that is designed and organised for a given purpose

NOTE The definition provided here is very close to that given in ISO 10303-49. Nevertheless ISO 15531 needs the notion of structured set of activities, without any predefined reference to the time or steps. In addition, from the point of view of flow management, some empty processes may be needed for a synchronisation purpose although they are not actually doing anything (ghost task).

[ISO 15531-1]

3.1.22
product
a thing or substance produced by a natural or artificial process

[ISO 10303-1]

3.1.23
proof theory
set of theories and lexical elements necessary for the interpretation of the semantics of the language

NOTE It consists of three components: the PSL-Core, the Outer core and the extensions.

[ISO 18629-1]

3.1.24
PSL-Core
set of axioms for the concepts of activity, activity-occurrence, time-point, and object

NOTE The motivation for PSL-Core is any two process-related applications shall share these axioms in order to exchange process information, and hence is adequate for describing the fundamental concepts of manufacturing processes. Consequently, this characterisation of basic processes makes few assumptions about their nature beyond what is needed for describing those processes, and the PSL-Core is therefore rather weak in terms of logical expressiveness. In particular, PSL-Core is not strong enough to provide definitions of the many auxiliary notions that become necessary to describe all intuitions about manufacturing processes.

[ISO 18629-1]

3.1.25
semilattice
partial ordering in which every two elements have a unique least upper bound

3.1.26
theory
set of axioms and definitions that pertain to a given concept or set of concepts

NOTE this definition reflects the approach of artificial intelligence in which a theory is the set of assumptions on which the meaning of the related concept is based.

[ISO 18629-1]

3.1.27
universe of discourse
the collection of concrete or abstract things that belong to an area of the real world, selected according to its interest for the system to be modelled and for its corresponding environment

[ISO 15531-1]

3.2 Abbreviations

— **FOL** First-Order Logic:

— **BNF** Backus-Naur form;

— **KIF** Knowledge Interchange Format;

— **PSL** Process Specification Language.

4 ISO 18629 general

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 dif and only iferent 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 Core theory and Situation Calculus (see annex B). As such it follows a significantly dif and only iferent 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 1x series of parts of ISO 18629 specify core theories needed to give precise definitions and the axioms of the primitive concepts of ISO 18629, thus enabling precise semantic translations between different schemes.

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

— the representation of the basic elements of the language;

— the provision of standardized sets of axioms that correspond to intuitive semantic primitive concepts adequate to describe basic processes;

— the set of rules to develop other core theories or extensions in compliance with PSL-Core.

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

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

5 Organization of ISO 18629-12

This clause specifies the fundamental theories from which the ISO 18629-12 Outer core is composed.

5.1 Extensions in Outer core

The core theories that are part of the Outer core are:

— Subactivity core theory (subactivity.th);

— Occurrence tree core theory (occtree.th);

— Discrete states core theory (disc_state.th);

— Atomic activity core theory (atomic.th);

— Complex activity core theory (complex.th);

— Activity occurrence core theory (act_occ.th).

Figure 1 shows the relationships between these theories within the Outer core, in which the arrows represent the dependencies among the theories. All theories in the Outer core are extensions of PSL-Core. The Atomic activity core theory is an extension of both the Subactivity and the Occurrence Trees theories, while the Discrete states core theory is an extension of the Occurrence Trees core theory alone. The Activity occurrence core theory is an extension of the Complex Acivities core theory, which in turn is an extension of Atomic Activities.

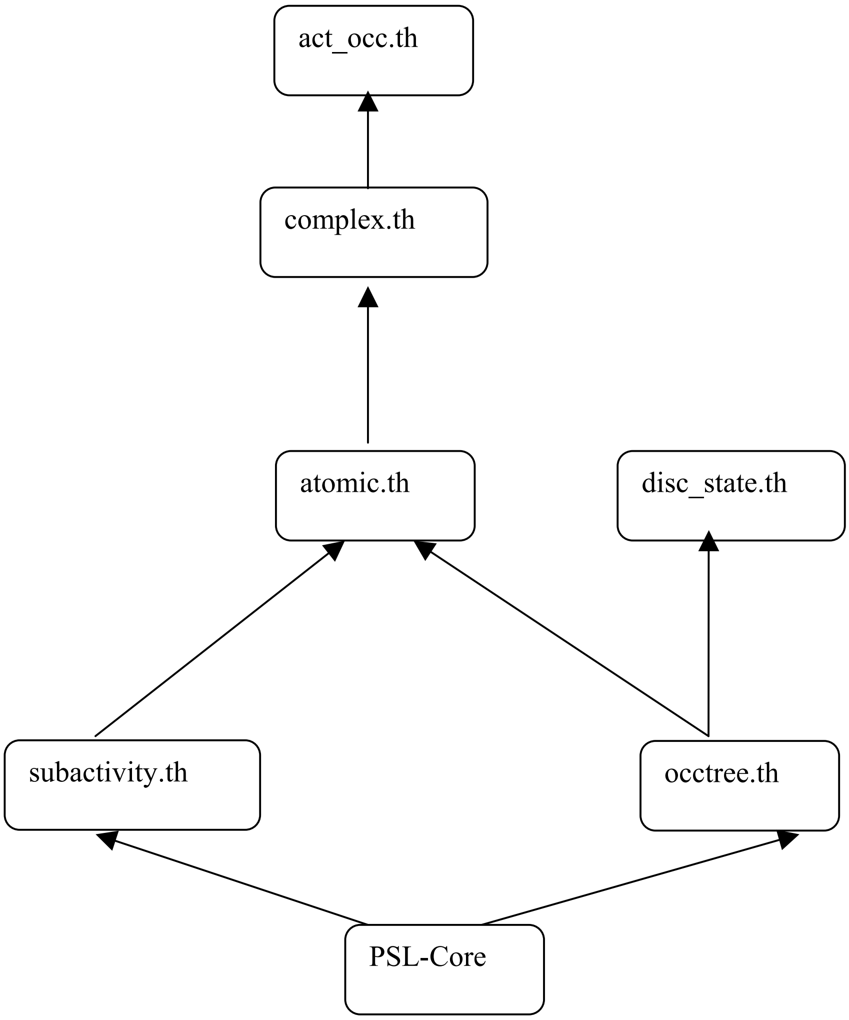

Figure 1: Relationships among sets of axioms within PSL Outer core.

6 Subactivity core theory

This core theory provides axioms for concepts for process composition. The only constraint imposed within this core theory is that the subactivity relation is isomorphic to a discrete partial ordering. Other core theories impose additional constraints.

6.1 Primitive Relations of the Subactivity core theory

The nonlogical lexicon of the Subactivity core theory contains one primitive relation symbol:

— subactivity.

6.2 Defined Relations of the Subactivity core theory

The nonlogical lexicon of the Subactivity core theory contains one defined relation symbol:

— primitive.

6.3 Relationship to other sets of axioms

The Subactivity core theory requires psl_core.th as specified in ISO 18629-11 (PSL-Core).

No definitional extensions are required by the Subactivity core theory.

6.4 Informal Semantics of the Subactivity core theory

6.4.1 subactivity

KIF notation for subactivity:

(subactivity ?a1 ?a2)

Informal semantics for subactivity:

(subactivity ?a1 ?a2) is TRUE in an interpretation of Subactivity core theory if and only if activity ?a1 is a subactivity of activity ?a2.

The subactivity relation forms a discrete partial ordering over the set of activities.

6.4.2 primitive

KIF notation for primitive:

(primitive ?a)

Informal semantics for primitive:

(primitive ?a) is TRUE in an interpretation of Subactivity core theory if and only if the activity ?a has no subactivities except for itself.

6.5 Definitions in the Subactivity core theory

An activity is primitive if and only if it has no subactivities except for itself.

6.5.1 Definition 1:

```
(forall (?a) (iff (primitive ?a)
    (forall (?a1)
        (implies    (subactivity ?a1 ?a)
                    (= ?a1 ?a))))))
```

6.6 Axioms of the Subactivity core theory

6.6.1 Axiom 1

subactivity is a relation over activities

```
(forall (?a1 ?a2)
    (implies        (subactivity ?a1 ?a2)
                    (and    (activity ?a1)
                            (activity ?a2))))
```

6.6.2 Axiom 2

subactivity is reflexive.

```
(forall (?a)
    (implies  (activity ?a)
              (subactivity ?a ?a)))
```

6.6.3 Axiom 3

subactivity is antisymmetric.

```
(forall (?a1 ?a2)
    (implies        (and    (subactivity ?a1 ?a2)
                            (subactivity ?a2 ?a1))
                    (= ?a1 ?a2)))
```

6.6.4 Axiom 4

subactivity is transitive.

```
(forall (?a1 ?a2 ?a3)
    (implies        (and    (subactivity ?a1 ?a2)
                            (subactivity ?a2 ?a3))
                    (subactivity ?a1 ?a3)))
```

6.6.5 Axiom 5

The subactivity relation is a discrete ordering, so every activity has an downwards successor in the ordering.

```
(forall (?a1 ?a2)
    (implies (subactivity ?a1 ?a2)
        (exists (?a3)
        (and (subactivity ?a1 ?a3)
            (subactivity ?a3 ?a2)
            (forall (?a4)
            (implies (and (subactivity ?a1 ?a4)
                          (subactivity ?a4 ?a3)
                     (or  (= ?a4 ?a1)
                          (= ?a4 ?a3))))))))
```

6.6.6 Axiom 6

The subactivity relation is a discrete ordering, so every activity has an upwards successor in the ordering.

```
(forall (?a1 ?a2)
    (implies (subactivity ?a1 ?a2)
        (exists (?a3)
        (and (subactivity ?a1 ?a3)
            (subactivity ?a3 ?a2)
                (forall (?a4)
            (implies (and (subactivity ?a3 ?a4)
                          (subactivity ?a4 ?a2)
                     (or  (= ?a4 ?a2)
                          (= ?a4 ?a3))))))))
```

7 Occurrence tree core theory

An occurrence tree is the set of all discrete sequences of activity occurrences. They are isomorphic to substructures of the situation tree from situation calculus (see Annex C), the primary difference being that rather than a unique initial situation, each occurrence tree has a unique initial activity occurrence. As in the situation calculus, the *poss* relation is introduced to allow the statement of constraints on activity occurrences within the occurrence tree. Since the occurrence trees include sequences that modellers of a domain will consider impossible, the *poss* relation "prunes" away branches from the occurrences tree that correspond to such impossible activity occurrences.

It should be noted that the occurrence tree is not a structure that represents the occurrences of subactivities of an activity. The occurrence tree is not representing a particular occurrence of an activity, but rather all occurrences of all activities in the domain.

7.1 Primitive Relations

The nonlogical lexicon of Occurrence Trees contains three primitive relation symbols:

— earlier;

— initial;

— legal.

7.2 Primitive Functions

The nonlogical lexicon of Occurrence Trees contains one primitive function symbol:

— successor.

7.3 Relationship to other sets of axioms

The Occurrence tree core theory requires psl_core.th as specified in ISO 18629-11 (PSL-Core).

No definitional extensions are required by the Occurrence tree core theory.

7.4 Informal Semantics of the Occurrence tree core theory

7.4.1 earlier

KIF notation for earlier :

(earlier ?occ1 ?occ2)

Informal semantics for earlier :

(earlier ?occ1 ?occ2) is TRUE in an interpretation of the Occurrence tree core theory if and only if the two activity occurrences ?occ1 and ?occ2 are on the same branch of the tree and ?occ1 is closer to the root of the tree than ?occ2.

In interpretations of Occurrence Trees, the set of all sequences of activity occurrences forms a tree; the earlier relation specifies the partial ordering over the activity occurrences in this tree.

7.4.2 initial

KIF notation for initial :

(initial ?occ)

Informal semantics for initial :

(initial ?occ) is TRUE in an interpretation of the Occurrence tree core theory if and only if the activity occurrence ?occ is the root of an occurrence tree.

7.4.3 legal

KIF notation for legal:

(legal ?occ)

Informal semantics for poss:

(legal ?occ) is TRUE in an interpretation of the Occurrence tree core theory if and only if the activity occurrence ?occ is an element of the legal occurrence tree.

7.4.4 poss

KIF notation for poss :

(poss ?occ1 ?occ2)

Informal semantics for poss:

(poss ?a ?occ2) is TRUE in an interpretation of the Occurrence tree core theory if and only if the activity ?a has a legal occurrence that is a successor of the activity occurrence ?occ in the occurrence tree.

7.4.5 precedes

KIF notation for precedes :

(precedes ?occ1 ?occ2)

Informal semantics for precedes:

(precedes ?occ1 ?occ2) is TRUE in an interpretation of the Occurrence tree core theory if and only if the activity occurrence ?occ1 is earlier than the activity occurrence ?occ2 in the occurrence tree and such that all activity occurrences between them correspond to activities that are possible. This relation specifies the sub-tree of the occurrence tree in which every activity occurrence is the occurrence of an activity that is possible.

7.4.6 successor

KIF notation for successor :

(successor ?a ?occ)

Informal semantics for successor:

(= (successor ?a ?occ) ?occ2) is TRUE in an interpretation of the Occurrence tree core theory if and only if ?occ2 denotes the occurrence of ?a that follows consecutively after the activity occurrence ?occ in the occurrence tree.

7.5 Axioms for the Occurrence tree core theory

The set of axioms in the Occurrence tree core theory is as follows:

7.5.1 Axiom 1

The earlier relation is restricted to activity occurrences.

```
(forall (?occ1 ?occ2)
    (implies        (earlier ?occ1 ?occ2)
                    (and   (activity_occurrence ?occ1)
                                  (activity_occurrence ?occ2))))
```

7.5.2 Axiom 2

The earlier relation over occurrences is irreflexive.

```
(forall (?occ1 ?occ2)
    (implies        (earlier ?occ1 ?occ2)
                    (not   (earlier ?occ2 ?occ1))))
```

7.5.3 Axiom 3

The earlier relation over occurrences is transitive.

```
(forall (?occ1 ?occ2 ?occ3)
    (implies        (and   (earlier ?occ1 ?occ2)
                           (earlier ?occ2 ?occ3))
                    (earlier ?occ1 ?occ3)))
```

7.5.4 Axiom 4

A branch in the occurrence tree is a totally ordered set of activity occurrences.

```
(forall (?occ1 ?occ2 ?occ3)
    (implies  (and   (earlier ?occ1 ?occ2)
                     (earlier ?occ3 ?occ2)
              (or    (earlier ?occ1 ?occ3)
                     (earlier ?occ3 ?occ1)
                     (= ?occ3 ?occ1))))
```

7.5.5 Axiom 5

No occurrence is earlier than an initial occurrence.

```
(forall (?occ1 ?occ2)
    (implies  (initial ?occ1)
              (not (exists (?occ2)
                     (earlier ?occ2 ?occ1)))))
```

7.5.6 Axiom 6

Every branch of the occurrence tree has an initial occurrence.

```
(forall (?occ1 ?occ2)
    (implies         (earlier ?occ1 ?occ2)
                     (exists (?occp)
                             (and   (initial ?occp)
                             (or    (earlier ?occp ?occ1)
                                    (= ?occp ?occ1))))))
```

7.5.7 Axiom 7

There is an initial occurrence for each activity.

```
(forall (?a)
    (implies  (activity ?a)
        (exists (?s)
            (and    (occurrence_of ?s ?a)
                (initial ?s)))))
```

7.5.8 Axiom 8

No two initial activity occurrences in the occurrence tree are occurrences of the same activity.

```
(forall (?occ1 ?occ2 ?a)
	(implies	(and		(initial ?occ1)
					(initial ?occ2)
					(occurrence_of ?occ1 ?a)
					(occurrence_of ?occ2 ?a))
			(= ?occ1 ?occ2)))
```

7.5.9 Axiom 9

The successor of an activity occurrence is an occurrence of the activity.

```
(forall (?a ?occ)
	(implies		(and	(activity_occurrence ?occ)
					(activity ?a))
			(occurrence_of (successor ?a ?occ) ?a))))
```

7.5.10 Axiom 10

Every non-initial occurrence is the successor of another occurrence.

```
(forall (?occ)
	(implies		(not (initial ?occ))
			(exists (?a ?occp)
					(= ?occ (successor ?a ?occp)))))
```

7.5.11 Axiom 11

An occurrence ?occ1 is earlier than the successor occurrence of ?occ2 if and only if the occurrence ?occ2 is later than ?occ1.

```
(forall (?a ?occ1 ?occ2)
	(iff	(earlier ?occ1 (successor ?a ?occ2))
		(or	(earlier ?occ1 ?occ2)
			(= ?occ1 ?occ2)))))
```

7.5.12 Axiom 12

The legal relation restricts activity occurrences.

(forall (?occ)

(implies (legal ?occ)

(activity_occurrence ?occ)))

7.5.13 Axiom 13

If an activity occurrence is legal, then all earlier activity occurrences in the occurrence tree are legal.

(forall (?occ1 ?occ2)

(implies (and (legal ?occ1)

(earlier ?occ2 ?occ1))

(legal ?occ2)))

7.5.14 Axiom 14

The endof an activity occurrence is before the beginof the successor of the activity occurrence.

(forall (?occ1 ?occ2)

(implies (earlier ?occ1 ?occ2)

(before (endof ?occ1) (beginof ?occ2))))

7.5.15 Definition 1

The poss relation is between activities and activity occurrences.

(forall (?a ?occ) (iff (poss ?a ?occ)

(legal (successor ?a ?occ))))

7.5.16 Definition 2

An activity occurrence ?occ1 precedes another activity occurrence ?occ2 if and only if ?occ1 is earlier than ?occ2 in the ordering relation and every activity occurrence between them is possible.

(forall (?occ1 ?occ2) (iff (precedes ?occ1 ?occ2)

(and (earlier ?occ1 ?occ2)

(legal ?occ2))))

8 Discrete state core theory

The Discrete states core theory introduces the notion of state (fluents). Fluents are changed only by the occurrence of activities, and fluents do not change during the occurrence of primitive activities. In addition, activities have preconditions (fluents that must hold before an occurrence) and effects (fluents that always hold after an occurrence).

8.1 Primitive categories in the Discrete state core theory

The nonlogical lexicon of the Discrete state core theory contains one primitive category:

— state.

8.2 Primitive relations in the Discrete state core theory

The nonlogical lexicon of Discrete states core theory contains two primitive relation symbols:

— holds;

— prior.

8.3 Relationship to other sets of axioms

The core theories required are :

— psl_core.th as specified in ISO 18629-11 (PSL-Core),

— occtree.th (Core theory of occurrence trees).

No definitional extensions are required by the Discrete state core theory.

8.4 Informal semantics of the Discrete state core theory

8.4.1 state

KIF notation for state :

(state ?f)

Informal semantics for state:

(state ?f) is TRUE in an interpretation of the Discrete state core theory if and only if ?f is a member of the set of states in the universe of discourse of the interpretation. States are a subcategory of object.

NOTE Object are defined in ISO 18629-11

They intuitively represent properties and relationships in the domain that can change as the result of the occurrence of activities.

8.4.2 holds

KIF notation for holds :

(holds ?f ?occ)

Informal semantics for holds :

(holds ?f ?occ) is TRUE in an interpretation of the Discrete State Core theory if and only if the state ?f is true after the activity occurrence ?occ.

8.4.3 prior

KIF notation for prior :

(prior ?f ?occ)

Informal semantics for prior :

(prior ?f ?occ) is TRUE in an interpretation of the Discrete State Core theory if and only if the state ?f is true prior to the activity occurrence ?occ.

8.5 Axioms for the Discrete state core theory

The set of axioms in the Discrete state core theory is as follows:

8.5.1 Axiom 1

States are objects.

```
(forall (?f)
    (implies        (state ?f)
                    (object ?f)))
```

8.5.2 Axiom 2

The holds relation is only between states and activity occurrences. Intuitively, it means that the state is true after the activity occurrence ?o.

```
(forall (?f ?occ)
    (implies (holds ?f ?occ)
            (and    (state ?f)
            (activity_occurrence ?occ))))
```

8.5.3 Axiom 3

The prior relation is only between states and activity occurrences. Intuitively, it means that the state is true before the activity occurrence ?o.

```
(forall (?f ?occ)
    (implies (prior ?f ?occ)
        (and        (state ?f)
                    (activity_occurrence ?occ))))
```

8.5.4 Axiom 4

All initial occurrences agree on the states that hold prior to them.

```
(forall (?occ1 ?occ2 ?f)
    (implies        (and    (initial ?occ1)
                            (initial ?occ2))
                    (iff    (prior ?f ?occ1)
                            (prior ?f ?occ2))))
```

8.5.5 Axiom 5

A state holds after an occurrence if and only if it holds prior to the successor occurrence.

```
(forall (?a ?occ)
    (iff  (holds ?f ?occ)
          (prior ?f (successor ?a ?occ))))
```

8.5.6 Axiom 6

If a fluent holds after some activity occurrence, then there exists an earliest activity occurrence along the branch where the fluent holds.

```
(forall (?f ?occ1)
    (implies  (holds ?f ?occ1)
              (exists (?occ2)
                    (and    (precedes ?occ2 ?occ1)
```

```
(holds ?f ?occ2)
(or     (initial ?occ2)
        (not (prior ?f ?occ2)))
(forall (?occ3)
        (implies    (and   (precedes ?occ2 ?occ3)
                           (precedes ?occ3 ?occ1))
                    (holds ?f ?occ3)))))))
```

8.5.7 Axiom 7

If a fluent does not hold after some activity occurrence, then there exists an earliest activity occurrence along the branch where the fluent does not hold.

```
(forall (?f ?occ1)
    (implies   (not (holds ?f ?occ1))
               (exists (?occ2)
                       (and   (precedes ?occ2 ?occ1)
                              (not (holds ?f ?occ2))
                              (or     (initial ?occ2)
                                      (prior ?f ?occ2))
                              (not (exists (?occ3)
                                      (and   (precedes ?occ2 ?occ3)
                                             (precedes ?occ3 ?occ1)
                                             (holds ?f ?occ3))))))))
```

9 Atomic activity core theory

The core theory of Atomic Activities provides axioms for intuitions about the concurrent aggregation of primitive activities. Such concurrent aggregation is represented by the occurrence of concurrent activities rather than concurrent activity occurrences.

9.1 Primitive Relations in the Atomic activity core theory

The nonlogical lexicon of the Atomic activity core theory contains one primitive relation symbol:

— atomic.

9.2 Primitive Functions in the Atomic activity core theory

The nonlogical lexicon of the Atomic activity core theory contains one primitive function symbol:

— conc.

9.3 Relationship to other sets of axioms

The core theories required by the Atomic activity core theory are :

— psl_core.th as specified in ISO 18629-11 (PSL-Core);

— occtree.th (Core theory of Occurrence trees);

— subactivity.th (Subactivity core theory).

No definitional extensions are required by the Atomic activity core theory.

9.4 Informal semantics of the Atomic activity core theory

9.4.1 atomic

KIF notation for atomic :

(atomic ?a)

Informal semantics for atomic :

(atomic ?a) is TRUE in an interpretation of the Atomic activity core theory if and only if either ?a is primitive or it is the concurrent superposition of a set of primitive activities.

9.4.2 conc

KIF notation for conc :

(conc ?a1 ?a2)

Informal semantics for conc :

(= ?a3 (conc ?a1 ?a2)) is TRUE in an interpretation of the Atomic activity core theory if and only if

?a3 is the atomic activity that is the concurrent superposition of the two atomic activities ?a1 and ?a2.

9.5 Definitions in Atomic activity core theory

No definitions are introduced by this core theory.

9.6 Axioms of the Atomic activity core theory

The set of axioms in the Atomic activity core theory is as follows:

9.6.1 Axiom 1

Primitive activities are atomic.

```
(forall (?a)
    (implies        (primitive ?a)
                    (atomic ?a)))
```

9.6.2 Axiom 2

The function conc is idempotent.

```
(forall (?a)
    (= ?a (conc ?a ?a)))
```

9.6.3 Axiom 3

The function conc is commutative.

```
(forall (?a1 ?a2)
    (= (conc ?a1 ?a2) (conc ?a2 ?a1)))
```

9.6.4 Axiom 4

The function conc is associative.

```
(forall (?a1 ?a2 ?a3)
    (= (conc ?a1 (conc ?a2 ?a3)) (conc (conc ?a1 ?a2) ?a3)))
```

9.6.5 Axiom 5

The concurrent aggregation of atomic activity is an atomic activity.

```
(forall (?a1 ?a2)
    (iff      (atomic (conc ?a1 ?a2))
              (and    (atomic ?a1)
                      (atomic ?a2))))
```

9.6.6 Axiom 6

An atomic activity ?a1 is a subactivity of an atomic activity ?a2 if and only if ?a2 is idempotent for ?a1.

```
(forall (?a1 ?a2)
    (implies    (and    (atomic ?a1)
                        (atomic ?a2))
                (iff    (subactivity ?a1 ?a2)
                        (= ?a2 (conc ?a1 ?a2)))))
```

9.6.7 Axiom 7

An atomic activity has a subactivity if and only if there exists another atomic activity which can be concurrently aggregated.

```
(forall (?a1 ?a2)
    (implies    (atomic ?a2)
                (iff    (subactivity ?a1 ?a2)
                        (exists (?a3)
                                (= ?a2 (conc ?a1 ?a3))))))
```

9.6.8 Axiom 8

The semilattice of atomic activities is distributive.

```
(forall (?a ?b0 ?b1)
    (implies    (and    (subactivity ?a (conc ?b0 ?b1))
                        (not (primitive ?a)))
                (exists (?a0 ?a1)
                        (and    (subactivity ?a0 ?a)
                                (subactivity ?a1 ?a)
                                (= ?a (conc ?a0 ?a1))))))
```

9.6.9 Axiom 9

Only atomic activity occurrences can be elements of the legal occurrence tree.


```
(forall (?a ?occ)
    (implies    (and    (occurrence_of ?occ ?a)
                        (legal ?occ))
                (atomic ?a)))
```

10 Complex activity core theory

This core theory provides the foundation for representing and reasoning about complex activities and the relationship between occurrences of an activity and occurrences of its subactivities. Within models of the Complex Activities core theory, occurrences of complex activities correspond to subtrees of the occurrence tree. An activity may have subactivities that do not occur; the only constraint is that any subactivity occurrence must correspond to a subtree of the activity tree that characterizes the occurrence of the activity. Not every occurrence of a subactivity is a subactivity occurrence. There may be other external activities that occur during an occurrence of an activity. Different subactivities may occur on different branches of the activity tree, so that different occurrences of an activity may have different subactivity occurrences.

10.1 Primitive Relations in the Complex activity core theory

The nonlogical lexicon of the Complex activity core theory contains two primitive relation symbols:

— min_precedes;

— root.

10.2 Defined Relations in the Complex activity core theory

The nonlogical lexicon of the Complex activity core theory contains six defined relation symbols:

— subtree;

— do;

— leaf;

— next_subocc;

— sibling.

10.3 Relationship to other sets of axioms

The core theories required by the Complex activity core theory are :

— psl_core.th as specified in ISO 18629-11 (PSL-Core);

— occtree.th (Core theory of Occurrence trees);

— subactivity.th (Subactivity core theory);

— atomic.th (Core theory of Atomic activities).

No definitional extensions are required by the Complex activity core theory.

10.4 Informal Semantics of the Complex activity core theory

10.4.1 min_precedes

KIF notation for min_precedes :

(min_precedes ?occ1 ?occ2 ?a)

Informal semantics for min_precedes :

(min_precedes ?occ1 ?occ2 ?a) is TRUE in an interpretation of the Complex activity core theory if and only if ?occ1 and ?occ2 are subactivity occurrences in the activity tree for ?a, and that ?occ1 precedes ?occ2 in the subtree. Any occurrence of an activity ?a corresponds to an activity tree (which is a subtree of the occurrence tree). The activity occurrences within this subtree are the subactivity occurrences of the occurrence of ?a.

10.4.2 root

KIF notation for root :

(root ?occ ?a)

Informal semantics for root :

(root ?occ ?a) is TRUE in an interpretation of the Complex activity core theory if and only if the activity occurrence ?occ is the root of an activity tree for ?a.

10.4.3 subtree

KIF notation for subtree:

(subtree ?occ ?a1 ?a2)

Informal semantics for subtree:

(subtree ?occ ?a1 ?a2) is TRUE in an interpretation of the Complex activity core theory if and only if

every atomic subactivity occurrence in the activity tree for ?a1 with root ?occ is an element of the activity tree for ?a2.

10.4.4 leaf

KIF notation for leaf:

(leaf ?occ ?a)

Informal semantics for leaf:

(leaf ?occ ?a) is TRUE in an interpretation of the Complex activity core theory if and only if the activity occurrence ?occ is the leaf of an activity tree for ?a.

10.4.5 do

KIF notation for do :

(do ?a ?occ1 ?occ2)

Informal semantics for do :

(do ?a ?occ1 ?occ2) is TRUE in an interpretation of the Complex activity core theory if and only if ?occ1 is the root of an activity tree and ?occ2 is a leaf of the same activity tree such that both activity occurrences are elements of the same branch of the activity tree.

10.4.6 next_subocc

KIF notation for next_subocc:

(next_subocc ?occ1 ?occ2 ?a)

Informal semantics for next_subocc:

(next_subocc ?occ1 ?occ2 ?a) is TRUE in an interpretation of the Complex activity core theory if and only if?occ1 precedes ?occ2 in the tree and there does not exist a subactivity occurrence that is between them in the tree.

10.4.7 sibling

KIF notation for sibling:

(sibling ?occ1 ?occ2 ?a)

Informal semantics sibling:

(sibling ?occ1 ?occ2 ?a) is TRUE in an interpretation of the Complex activity core theory if and only.

10.5 Definitions in Complex activity core theory

The definitions introduced by the Complex activity core theory core theory are:

10.5.1 Definition 1

The activity tree for ?a1 with root occurrence ?occ1 is a subtree of an activity tree for ?a2 if and only if every atomic subactivity occurrence in the activity tree for ?a1 is an element of the activity tree for ?a2.

```
(forall (?occ1 ?a1 ?a) (iff (subtree ?occ1 ?a1 ?a2)
(and    (root ?occ1 ?a1)
        (exists (?occ2)
                (and    (root ?occ2 ?a2)
                        (min_precedes ?occ1 ?occ2 ?a1)))
        (exists (?occ3)
                (and    (min_precedes ?occ1 ?occ3 ?a1)
                        (not (min_precedes ?occ2 ?occ3 ?a2)))))))
```

10.5.2 Definition 2

An occurrence is the leaf of an activity tree if and only if there exists an earlier atomic subactivity occurrence but there does not exist a later atomic subactivity occurrence.

```
(forall (?occ ?a) (iff (leaf ?occ ?a)
(exists (?occ1)
    (and        (min_precedes ?occ1 ?occ ?a)
                (not (exists (?occ2)
                        (min_precedes ?occ ?occ2 ?a))))))
```

10.5.3 Definition 3

The do relation specifies the initial and final atomic subactivity occurrences of an activity.

```
 (forall (?a ?occ1 ?occ2) (iff (do ?a ?occ1 ?occ2)
(and    (min_precedes ?occ1 ?occ2 ?a)
        (not (exists (?occ3)
                        (min_precedes ?occ3 ?occ1 ?a)))
        (not (exists (?occ4)
                        (min_precedes ?occ2 ?occ4 ?a)))))))
```

10.5.4 Definition 4

An activity occurrence ?occ2 is the next subactivity occurrence after ?occ1 in an activity tree for ?a if and only of ?occ1 precedes ?occ2 in the tree and there does not exist a subactivity occurrence that is between them in the tree.

```
(forall (?occ1 ?occ2 ?a) (iff (next_subocc ?occ1 ?occ2 ?a)

(and    (min_precedes ?occ1 ?occ2 ?a)

        (not (exists (?occ3)

                (and    (min_precedes ?occ1 ?occ3 ?a)

                        (min_precedes ?occ3 ?occ2 ?a))))))
```

10.5.5 Definition 5

The atomic subactivity occurrences ?occ1 and ?occ2 are siblings with respect to an activity ?a off they either have a common predecessor in an activity tree for ?a or they are both roots of activity trees for ?a that have a common predecessor in the occurrence tree.

```
(forall (?occ1 ?occ2 ?a) (iff (sibling ?occ1 ?occ2 ?a)

(or   (exists (?occ3)

        (and    (next_subocc ?occ3 ?occ1 ?a)

                (next_subocc ?occ3 ?occ2 ?a)))

      (and      (root ?occ1 ?a)

                (root ?occ2 ?a)

                (or     (and    (initial ?occ1)

                                (initial ?occ2))

                        (exists (?occ4 ?a1 ?a2)

                                (and    (= ?occ1 (successor ?a1 ?occ4))

                                        (= ?occ2 (successor ?a2 ?occ4)))))))))
```

10.6 Axioms of the Complex activity core theory

The set of axioms needed by this core theory is:

10.6.1 Axiom 1

Occurrences in the activity tree for an activity correspond to atomic subactivity occurrences of the activity.

```
(forall (?a ?occ1 ?occ2)
    (implies    (min_precedes ?occ1 ?occ2 ?a)
        (exists (?a1 ?ap)
                (and    (subactivity ?a1 ?a)
                        (atomic ?ap)
                        (subactivity ?a1 ?ap)
                        (occurrence_of ?occ2 ?ap)))))
```

10.6.2 Axiom 2

Occurrences in the activity tree for an activity correspond to atomic subactivity occurrences of the activity.

```
(forall (?a ?occ1 ?occ2)
    (implies    (min_precedes ?occ1 ?occ2 ?a)
        (exists (?a2 ?ap)
                (and    (subactivity ?a2 ?a)
                        (atomic ?ap)
                        (subactivity ?a2 ?ap)
                        (occurrence_of ?occ1 ?ap)))))
```

10.6.3 Axiom 3

Root occurrences in the activity tree correspond to atomic subactivity occurrences of the activity.

```
(forall (?a ?occ1)
    (implies    (root ?occ1 ?a)
                (exists (?a2 ?ap)
                        (and    (subactivity ?a2 ?a)
```

(atomic ?ap)

(subactivity ?a2 ?ap)

(occurrence_of ?occ1 ?ap)))))

10.6.4 Axiom 4

All activity trees have a root subactivity occurrence.

```
(forall (?occ1 ?occ2 ?a)
    (implies (min_precedes ?occ1 ?occ2 ?a)
             (exists (?occ3)
                 (and (root ?occ3 ?a)
                      (or (min_precedes ?occ3 ?occ1 ?a)
                          (= ?occ3 ?occ1))))))
```

10.6.5 Axiom 5

No subactivity occurrences in an activity tree occur earlier than the root subactivity occurrence.

```
(forall (?occ ?a)
    (implies (root ?occ ?a)
             (not (exists (?occ2)
                      (min_precedes ?occ2 ?occ ?a)))))
```

10.6.6 Axiom 6

An activity tree is a subtree of the occurrence tree.

```
(forall (?occ1 ?occ2 ?a)
    (implies (min_precedes ?occ1 ?occ2 ?a)
             (exists (?occ0)
                 (and (initial ?occ0)
                      (or (precedes ?occ0 ?occ1)
```

```
                                        (= ?occ0 ?occ1))
                                (precedes ?occ1 ?occ2)))))
```

10.6.7 Axiom 7

Root occurrences are elements of the occurrence tree.

```
(forall (?occ ?a)
    (implies    (root ?occ ?a)
                (exists (?occ0)
                        (and    (initial ?occ0)
                                        (or     (precedes ?occ0 ?occ)
                                                (= ?occ0 ?occ))))))
```

10.6.8 Axiom 8

Every atomic activity occurrence is an activity tree containing only one occurrence.

```
(forall (?a1 ?a2 ?occ)
    (implies    (and    (atomic ?a1)
                        (occurrence_of ?occ ?a1)
                                (subactivity ?a2 ?a1))
                (root ?occ ?a2)))
```

10.6.9 Axiom 9

Activity trees are discrete.

```
(forall (?occ1 ?occ2)
    (implies    (min_precedes ?occ1 ?occ2 ?a)
                (exists (?occ3)
                        (and    (next_subocc ?occ1 ?occ3 ?a)
                                (or     (min_precedes ?occ3 ?occ2 ?a)
                                        (= ?occ3 ?occ2))))))
```

10.6.10 Axiom 10

Subactivity occurrences on the same branch of the occurrence tree are on the same branch of the activity tree.

```
(forall (?a ?occ1 ?occ2 ?occ3)
    (implies    (and        (min_precedes ?occ1 ?occ2 ?a)
                            (min_precedes ?occ1 ?occ3 ?a)
                            (precedes ?occ2 ?occ3))
                (min_precedes ?occ2 ?occ3 ?a)))
```

10.6.11 Axiom 11

The activity tree for a complex subactivity occurrence is a subtree of the activity tree for the activity occurrence.

```
(forall (?a1 ?a2)
    (implies    (subactivity ?a1 ?a2)
                (not (exists (?occ)
                        (subtree ?occ ?a2 ?a1)))))
```

11 Activity occurrence core theory

The Complex Activities only provides axioms for constraints on atomic subactivity occurrences. The Activity Occurrences core theory generalizes these intuitions to arbitrary complex subactivities.

11.1 Primitive Relations in the Activity occurrence core theory

The nonlogical lexicon of the Activity occurrence core theory contains two primitive relation symbols:

— subactivity_occurrence;

— mono.

11.2 Defined Relations in the Activity occurrence core theory

The nonlogical lexicon of the Activity occurrence core theory contains six defined relation symbols:

— root_occ;
— leaf _occ;
— same_grove;
— iso_occ;
— equiv_occ;
— hom.

11.3 Relationship to other sets of axioms

The core theories required are:

psl_core.th as specified in ISO 18629-11 (PSL-Core);

— occtree.th (Core theory of Occurrence trees);

— subactivity.th (Subactivity core theory);

— atomic.th (Core theory of Atomic activities);

— complex.th (Activity occurrence core theory).

No definitional extensions are required by the Activity occurrence core theory.

11.4 Informal Semantics of the Activity occurrence core theory

11.4.1 subactivity_occurrence

KIF notation for subactivity_occurrence :

(subactivity_occurrence ?occ1 ?occ2)

Informal semantics for subactivity_occurrence :

There is a one-to-one correspondence between activity occurrences and branches in the activity trees for the activity. The informal description for subactivity_occurrence is:

(subactivity_occurrence ?occ1 ?occ2) is TRUE in an interpretation of the Activity occurrence core theory if and only if the branch corresponding to the activity occurrence ?occ1 is a subset of the branch corresponding to activity occurrence ?occ2.

11.4.2 mono

KIF notation for mono :

(mono ?occ1 ?occ2 ?a)

Informal semantics for mono :

(mono ?occ1 ?occ2 ?a) is TRUE in an interpretation of the Activity occurrence core theory if and only if there is a one-to-one mapping between branches of an activity tree for ?a that maps the atomic subactivity occurrence ?occ1 to the atomic activity occurrence ?occ2.

11.4.3 root_occ

KIF notation for root_occ :

(root_occ ?occ1 ?occ2 ?a)

Informal semantics for root_occ :

(root_occ ?occ1 ?occ2 ?a) is TRUE in an interpretation of the Activity occurrence core theory if and only if activity occurrence ?occ1 is the root occurrence in the branch of the activity tree for ?a corresponding to the activity occurrence ?occ2.

11.4.4 leaf_occ

KIF notation for leaf_occ :

(leaf_occ ?occ1 ?occ2 ?a)

Informal semantics for leaf_occ :

(leaf_occ ?occ1 ?occ2 ?a) is TRUE in an interpretation of the Activity occurrence core theory if and only if activity occurrence ?occ1 is the leaf occurrence in the branch of the activity tree for ?a corresponding to the activity occurrence ?occ2.

11.4.5 iso_occ

KIF notation for iso_occ :

(iso_occ ?occ1 ?occ2)

Informal semantics for iso_occ :

(iso_occ ?occ1 ?occ2) is TRUE in an interpretation of the Activity occurrence core theory if and only if both ?occ1 and ?occ2 are occurrences of an atomic activity that contain a common subactivity.

11.4.6 hom

KIF notation for hom :

(hom ?occ1 ?occ2 ?a)

Informal semantics for hom :

(hom ?occ1 ?occ2 ?a) is TRUE in an interpretation of the Activity occurrence core theory if and only if there is a mapping between branches of an activity tree for ?a that maps the atomic subactivity occurrence ?occ1 to the atomic subactivity occurrence ?occ2.

11.4.7 same_grove

KIF notation for same_grove :

(same_grove ?occ1 ?occ2 ?a)

Informal semantics for same_tree :

(same_grove ?occ1 ?occ2 ?a) is TRUE in an interpretation of the Activity occurrence core theory if and only if activity occurrences ?occ1 and ?occ2 of ?a correspond to branches in the same activity tree for ?a.

11.5 Definitions for Activity occurrence core theory

The definitions introduced by the Activity occurrence core theory are:

11.5.1 Definition 1

An occurrence ?occ1 is the root occurrence of an occurrence of ?a if and only if it is a subactivity occurrence and it is the root of an activity tree for ?a.

(forall (?occ1 ?occ2) (iff (= ?occ2 (root_occ ?occ1)

(exists (?a)

(and (occurrence ?occ1 ?a)

(subactivity_occurrence ?occ2 ?occ1)

(root ?occ2 ?a)))))

11.5.2 Definition 2

An occurrence ?occ1 is the leaf occurrence of an occurrence of ?a if and only if it is a subactivity occurrence and it is the leaf of an activity tree for ?a.

(forall (?s ?occ) (iff (leaf_occ ?s ?occ)

(exists (?a)
(and (occurrence_of ?occ ?a)
(subactivity_occurrence ?s ?occ)
(leaf ?s ?a)))))

11.5.3 Definition 3

An activity occurrence ?occ1 is isomorphic to an activity occurrence ?occ2 if both are occurrences of an atomic activity that contain a common subactivity.

(forall (?s1 ?s2) (iff (iso_occ ?s1 ?s2)
(exists (?a1 ?a2)
(and (occurrence_of ?s1 ?a1)
(occurrence_of ?s2 ?a2)
(subactivity ?a1 ?a2)))))

11.5.4 Definition 3

Two activity occurrences are occurrence equivalent if and only if they are occurrences of atomic activities with a common subactivity.

```
(forall (?occ1 ?occ2) (iff (equiv_occ ?occ1 ?occ2)

(exists (?a1 ?a2 ?a3)

    (and    (occurrence_of ?occ1 (conc ?a1 ?a2))

            (occurrence_of ?occ2 (conc ?a1 ?a3)))))))
```

11.5.5 Definition 4

```
(forall (?s1 ?s2 ?a) (iff (hom ?s1 ?s2 ?a)

(exists (?occ1 ?occ2)

    (and    (iso_occ ?s1 ?s2)

            (subactivity_occurrence ?s1 ?occ1)

            (subactivity_occurrence ?s2 ?occ2)

            (occurrence_of ?occ1 ?a)

            (occurrence_of ?occ2 ?a)

            (not (= ?occ1 ?occ2))))))
```

11.5.6 Definition 5

Two occurrences of an activity are in the grove of activity trees if and only if they have the same root occurrence.

```
(forall (?occ1 ?occ2) (iff (same_grove ?occ1 ?occ2)

(exists (?a)
    (and  (occurrence_of ?occ1 ?a)
          (occurrence_of ?occ2 ?a)
          (or   (and   (initial (root_occ ?occ1))
                       (initial (root_occ ?occ2)))
                (exists (?s4 ?a1 ?a2)
                       (and   (= (root_occ ?occ1) (successor ?a1 ?s4))
                              (= (root_occ ?occ2) (successor ?a2 ?s4)))))))))
```

11.6 Axioms for Activity occurrence core theory

The set of axioms in the Activity occurrence core theory is as follows:

11.6.1 Axiom 1

There exists an occurrence of an activity ?a for every branch of an activity tree for ?a. All atomic subactivity occurrences on the branch are subactivity occurrences of the occurrence of ?a.

```
(forall (?a ?occ1 ?occ2)
    (implies        (min_precedes ?occ1 ?occ2 ?a)
                    (exists (?occ)
                        (and    (occurrence ?occ ?a)
                                (subactivity_occurrence ?occ1 ?occ)
                                (subactivity_occurrence ?occ2 ?occ)))))
```

11.6.2 Axiom 2

There exists an occurrence of an activity ?a for each branch of an activity tree for ?a. The root subactivity occurrence on the branch is a subactivity occurrence of the occurrence of ?a.

```
(forall (?a ?s)
    (implies    (root ?s ?a)
            (exists (?occ)
                    (occurrence_of ?occ ?a)
                    (subactivity_occurrence ?s ?occ))))
```

11.6.3 Axiom 3

Every occurrence of an activity a contains atomic subactivity occurrences that are elements of activity trees for a.

```
(forall (?occ ?a)
    (implies  (exists ?s1 ?s2)
            (and   (subactivity_occurrence ?s1 ?occ)
                (subactivity_occurrence ?s2 ?occ)
                (root ?s2 ?a)
                (or    (min_precedes ?s2 ?s1 ?a)
                    (= ?s1 ?s2))))))
```

11.6.4 Axiom 4

Distinct occurrences of an activity correspond to distinct branches of an activity tree.

```
(forall (?a ?s1 ?occ1 ?occ2)
	(implies  (and  (occurrence_of ?occ1 ?a)
			(occurrence_of ?occ2 ?a)
			(not (= ?occ1 ?occ2))
			(subactivity_occurrence ?s1 ?occ1)
			(subactivity_occurrence ?s1 ?occ2))
		(exists (?s2)
			(and    (min_precedes ?s1 ?s2 ?a)
				(subactivity_occurrence ?s1 ?occ1)
				(not (subactivity_occurrence ?s1 ?occ2))))))
```

11.6.5 Axiom 5

All atomic subactivity occurrences of a complex activity occurrence are elements of the same branch of the activity tree.

```
(forall (?a ?a1 ?a2 ?occ ?s1 ?s2)
    (implies  (and  (occurrence_of ?occ ?a)
                    (occurrence_of ?s1 ?a1)
                    (atomic ?a1)
                    (occurrence_of ?s2 ?a2)
                    (atomic ?a2)
                    (subactivity_occurrence ?s1 ?occ)
                    (subactivity_occurrence ?s2 ?occ))
              (or   (min_precedes ?s1 ?s2 ?a)
                    (min_precedes ?s2 ?s1 ?a)
                    (= ?s1 ?s2))))
```

11.6.6 Axiom 6

All elements of the same branch of an activity tree are atomic subactivity occurrences of the same activity occurrences.

```
(forall (?a ?s1 ?s2 ?occ)
    (implies  (and  (min_precedes ?s1 ?s2 ?a)
                    (subactivity_ooccurrence ?s2 ?occ))
              (subactivity_occurrence ?s1 ?occ))
```

11.6.7 Axiom 7

The subactivity_occurrence relation preserves the subactivity relation.

```
(forall (?a1 ?a2 ?occ1 ?occ2)
    (implies  (and  (occurrence_of ?occ1 ?a1)
```

(occurrence_of ?occ2 ?a2)

(subactivity_occurrence ?occ1 ?occ2))

(subactivity ?a1 ?a2)))

11.6.8 Axiom 8

The subactivity_occurrence relation is transitive.

```
(forall (?occ1 ?occ2 ?occ3)
    (implies  (and  (subactivity_occurrence ?occ1 ?occ2)
                    (subactivity_occurrence ?occ2 ?occ3))
              (subactivity_occurrence ?occ1 ?occ3)))
```

11.6.9 Axiom 9

Occurrences of subactivities are subactivity occurrences if the occurrences satisfy branch containment.

```
(forall (?a1 ?a2 ?occ1 ?occ2)
    (implies  (and  (occurrence_of ?occ1 ?a1)
                    (occurrence_of ?occ2 ?a2)
                    (subactivity ?a1 ?a2)
                    (not (subactivity_occurrence ?occ1 ?occ2)))
              (exists (?s)
                    (and  (subactivity_occurrence ?s ?occ2)
                          (not (subactivity_occurrence ?s ?occ1))))))
```

11.6.10 Axiom 10

The beginof timepoint for a complex activity occurrence is equal to the beginof timepoint of its root occurrence.

```
(forall (?occ)
    (implies  (activity_occurrence ?occ)
              (= (beginof ?occ) (beginof (root_occ ?occ)))))
```

11.6.11 Axiom 11

The endof timepoint for a complex activity occurrence is equal to the endof timepoint of its leaf occurrence.

```
(forall (?s ?occ)
    (implies    (leaf_occ ?s ?occ)
                (= (endof ?occ) (endof ?s))))
```

11.6.12 Axiom 12

The mono relation is a branch homomorphism.

```
(forall (?s1 ?s2 ?a)
    (implies    (mono ?s1 ?s2 ?a)
                (hom ?s1 ?s2 ?a)))
```

11.6.13 Axiom 13

If an atomic subactivity occurrence is mapped in a branch homomorphism, then there exists another atomic subactivity occurrence that is mono with it.

```
(forall (?s1 ?s2 ?a)
    (implies    (hom ?s1 ?s2 ?a)
                (exists (?s3)
                        (and    (mono ?s3 ?s2 ?a)
                                (or     (min_precedes ?s1 ?s3 ?a)
                                        (min_precedes ?s3 ?s1 ?a)
                                        (= ?s1 ?s3))))))
```

11.6.14 Axiom 14

The mono relation is restricted to one-to-one homomorphisms between different branches of the activity tree.

```
(forall (?s1 ?s2 ?s3 ?a)
    (implies    (and    (mono ?s1 ?s2 ?a)
```

```
                (mono ?s3 ?s2 ?a))
          (not (or (min_precedes ?s1 ?s3 ?a)
                (min_precedes ?s3 ?s1 ?a)))))
```

11.6.15 Axiom 15

```
(forall (?s1 ?s2 ?s3 ?s4 ?occ ?a)
    (implies   (and   (min_precedes ?s1 ?s2 ?a)
                      (mono ?s1 ?s3 ?a)
                      (mono ?s2 ?s4 ?a)
                      (subactivity_occurrence ?s3 ?occ)
                      (subactivity_occurrence ?s4 ?occ)
                      (iso_occ ?s1 ?s2))
               (min_precedes ?s3 ?s4 ?a)))
```

Annex A
(normative)
Use of ASN.1 Identifiers in SC4 standards

To provide for unambiguous identification of an information object in an open system, the object identifier

```
iso standard 18629 part 12 version 1
```

is assigned to this part of ISO 18629. The meaning of this value is defined in ISO/IEC 8824-1 and is described in ISO 18629-1.

Annex B
(informative)
Relationship to Situation Calculus

The theories of Occurrence trees, Discrete states, and Atomic activities extend earlier work arising from knowledge representation research in the artificial intelligence community. In particular, the notion of occurrence trees is a generalization of the situation calculus. Situation calculus is a formalism for reasoning about actions that first appeared in [2]. The original ontology that was proposed in this paper consisted of situations, fluents, and actions. Although no axiomatization of the situation calculus was provided in this early work, there were informal intuitions for the interpretation of these basic categories. Situations were thought of as snapshots of the world, fluents represented properties that could vary across situations, and actions changed the values of fluents.

The language of situation calculus in [2] included a relation symbol holds(f,s) (to denote that fluent f is true in situation s) and a function symbol do(a,s) (to denote the situation that is the result of performing an action a in situation s. The preconditions and effects of an action could then be represented by axioms that specify what fluents must hold when an action is performed.

Later work in [4] provided an explicit set of axioms for situation calculus. In particular, Reiter proposed a second-order axiomatization of situation calculus that also included a new relation symbol, poss(a,s), that denoted the fact that action a is possible in situation s, and a new constant symbol S0 (denoting the initial situation). Models of this core theory are trees with root S0 and in which branches represent hypothetical futures showing all possible ways in which events in the world could unfold. Any arbitrary sequence of actions identifies a branch in the tree of situations; thus two different sequences of actions lead to different situations. In this approach, situations were intuitively sequences of action occurrences. It is important to note that situation trees do not represent state transitions; rather, they are representing future hypothetical states of the world as being the result of a particular sequence of actions.

The axiomatization situation calculus provided in [4] did not include a core theory of time or provide any means of representing concurrency. Later work in [3] extended the axiomatization to include both of these concepts.

The core theory of Occurrence Trees in the Outer core of ISO 18629-12 modifies the axioms from [4] and [3]. Rather than considering trees in which situations are sequences of actions, we consider trees in which situations are roughly analogous to activity occurrences. Of course, the initial situation S0 does not correspond to the occurrence of any activity, which leads to the introduction of initial activity occurrences. Also, the axiomatization within ISO 18629-12 is first-order.

Annex C
(informative)
Example of process description using PSL-Outercore

The purpose of this annex is to provide a detailed scenario in which the ISO 18629 PSL is used in a knowledge-sharing effort which involves multiple manufacturing functions.

This scenario is an "interoperability" manufacturing scenario. This means that its goal is to show how PSL can be used to facilitate the communication of process knowledge in a manufacturing environment. Specifically, this scenario is centred around the exchange of knowledge from a process planner to a job shop scheduler.

This annex extends the test case introduced in ISO 18629-11 annex C to illustrate the application of outercore concepts in the specification of the manufacturing process of a product named GT-350.

C.1 GT-350 Manufacturing Processes

This section unites the various departmental processes into a high-level collection of activities which are enacted to create a GT-350 product. As described in the GT-350 product structure (see ISO 18629-11 Annex C table C1), subcomponents of this product are either purchased, sub-contracted, or made internally. These process descriptions address the activities performed to manufacture the internal subcomponents. This top-down view of the manufacturing process provides an overall picture from an abstract, “make GT350” activity which is expanded down to the detailed departmental levels.

As the Figure C.1 below shows, the GT-350 manufacturing process is divided into 6 main areas of work. The first five: make interior, make drive, make trim, make engine and make chassis are all unordered with respect to each other but they must all be completed before final assembly takes place.

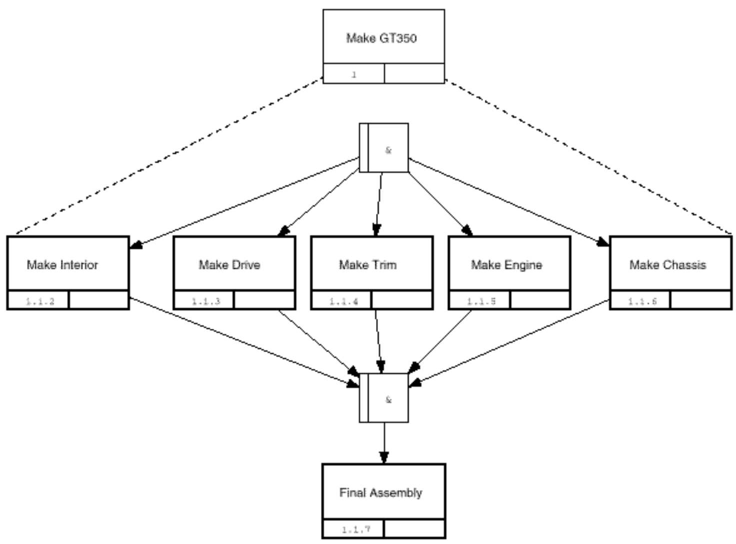

Figure C1: TOP level process for manufacturing a GT350 [1]

The PSL-Outercore-based representation of the top level process is :

```
(subactivity make-chassis make_gt350)

(subactivity make-interior make_gt350)

(subactivity make-drive make_gt350)

(subactivity make-trim make_gt350)

(subactivity make-engine make_gt350)

(subactivity final-assembly make_gt350)


(forall (?occ)
    (⇔      (occurrence_of ?occ make_gt350)
            (exists (?occ1 ?occ2 ?occ3 ?occ4 ?occ5 ?occ6)
                    (and    (occurrence_of ?occ1 make_chassis)
                            (occurrence_of ?occ2 make_interior)
                            (occurrence_of ?occ3 make_drive)
                            (occurrence_of ?occ4 make_trim)
                            (occurrence_of ?occ5 make_engine)
                            (occurrence_of ?occ6 final_assembly)
                            (subactivity_occurrence ?occ1 ?occ)
                            (subactivity_occurrence ?occ2 ?occ)
                            (subactivity_occurrence ?occ3 ?occ)
                            (subactivity_occurrence ?occ4 ?occ)
                            (subactivity_occurrence ?occ5 ?occ)
                            (subactivity_occurrence ?occ6 ?occ)
                                (forall (?s1 ?s2 ?s3 ?s4 ?s5 ?s6)
                                (implies        (and    (leaf_occ ?s1 ?occ1)
```

```
                                        (leaf_occ ?s2 ?occ2)
                                        (leaf_occ ?s3 ?occ3)
                                        (leaf_occ ?s4 ?occ4)
                                        (leaf_occ ?s5 ?occ5)
                                        (root_occ ?s6 ?occ6))
                              (min_precedes ?s1 ?s6 make_gt350)
                              (min_precedes ?s2 ?s6 make_gt350)
                              (min_precedes ?s3 ?s6 make_gt350)
                              (min_precedes ?s4 ?s6 make_gt350)
                         (min_precedes ?s5 ?s6 make_gt350)))))))
```

Each of these abstract activities can be further detailed, however for the example proposed in this annex, we will not develop all of them.

On the basis of the IDEF3 [1] representation (in terms of process representation) of the abstract activities met during the different stages of the manufacturing process, we will extract some examples of process descriptions using the PSL-Outer core presented in this part 12 of the ISO 18629 standard.

C.2 The "make-engine" abstract activity

The 350-Engine is assembled from work performed in several CMW departments. The manufacturing process is shown in Figure C.2. The part is made up of an engine block, a harness, and wiring. The sub-processes are detailed in the sub-sections below. The 350-Engine is assembled at the A004 assembly bench and takes 5 minutes per piece.

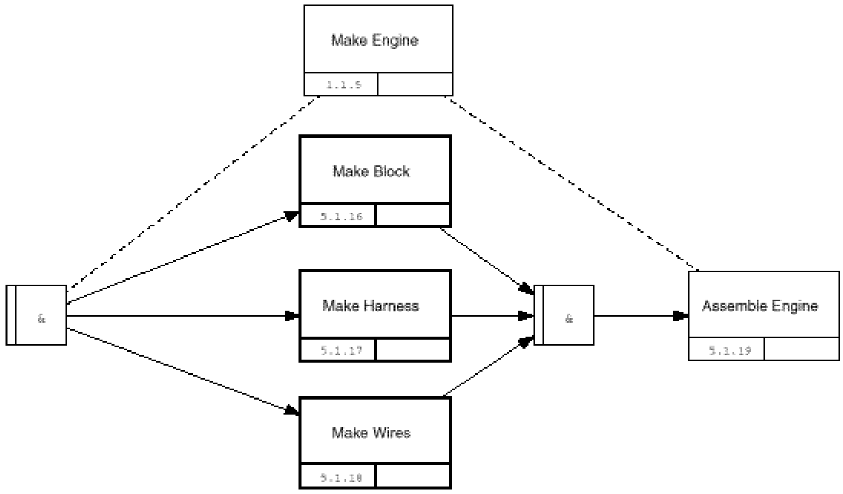

Figure C.2: PROCESS for manufacturing the 350–Engine [1]

The PSL-Outercore-based representation of some activities and of the process related information at the make-engine stage is :

(subactivity make_block make_engine)

(subactivity make-harness make_engine)

(subactivity make-wires make_engine)

(subactivity assemble_engine make_engine)

(forall (?occ)

(⇔(occurrence_of ?occ make_engine)

(exists (?occ1 ?occ2 ?occ3 ?occ4)

(and (occurrence_of ?occ1 make_block)

(occurrence_of ?occ2 make_harness)

(occurrence_of ?occ3 make_wires)

(occurrence_of ?occ4 assemble_engine)

(subactivity_occurrence ?occ1 ?occ)

(subactivity_occurrence ?occ2 ?occ)

(subactivity_occurrence ?occ3 ?occ)

(subactivity_occurrence ?occ4 ?occ)

(forall (?s1 ?s2 ?s3 ?s4)

(implies (and (leaf_occ ?s1 ?occ1)

(leaf_occ ?s2 ?occ2)

(leaf_occ ?s3 ?occ3)

(root_occ ?s4 ?occ4))

(min_precedes ?s1 ?s4 make_engine)

(min_precedes ?s2 ?s4 make_engine)

(min_precedes ?s3 ?s4 make_engine)))))

C.2.1 Make Block

The 350-Block is manufactured as part of the 350–Engine sub-assembly. This involves an integration of work from the foundry and machine shop, as shown in the Figure C.3.

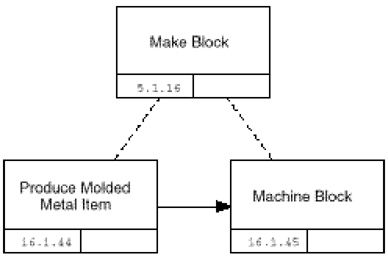

Figure C.3: PROCESS for manufacturing the 350–Block [1]

The PSL-Outercore-based representation of some activities and of the process related information is :

(subactivity produce_molded_metal make_block)

(subactivity machine_block make_block)

(primitive machine_block)

(primitive produce_molded_metal)

(forall (?occ)

(⇔ (occurrence_of ?occ make_block)

(exists (?occ1 ?occ2)

(and (occurrence_of ?occ1 produce_molded_metal)

(occurrence_of ?occ2 machine_block)

(min_precedes ?occ1 ?occ2 make_block)))))

C.2.2 Make Harness

The 350-Harness (Figure C.4) is manufactured as part of the 350–Engine sub-assembly. This involves work performed at the wire and cable department. Figure C.5 expands the harness wire production process. The 350-Harness is assembled by a bench worker at a wire and cable bench. It takes 10 minutes per set.

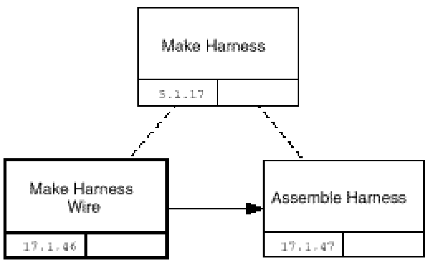

Figure C.4: PROCESS for manufacturing the 350–Harness [1]

The PSL-Outercore-based representation of some activities and of the process related information is :

(subactivity make_harness_wire make_harness)

(subactivity assemble_harness make_harness)

(primitive assemble_harness)

(forall (?occ)

(⇔ (occurrence_of ?occ make_harness)

(exists (?occ1 ?occ2 ?occ3)

(and (occurrence_of ?occ1 make_harness_wire)

(occurrence_of ?occ2 assemble_harness)

(leaf_occ ?occ3 ?occ1)

(min_precedes ?occ3 ?occ2 make_harness)))))

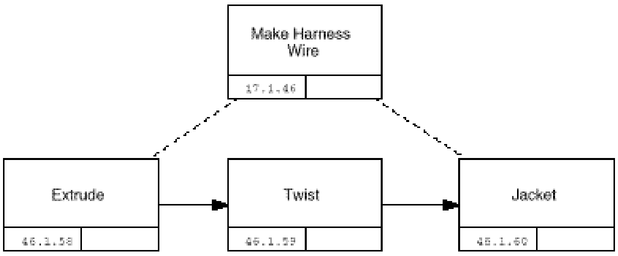

Figure C.5: PROCESS for manufacturing the harness wire [1]

C.2.3 Make Harness Wires

The 350-Wire-Set is manufactured as part of the 350–Engine sub-assembly. This involves work performed at the wire and cable department.

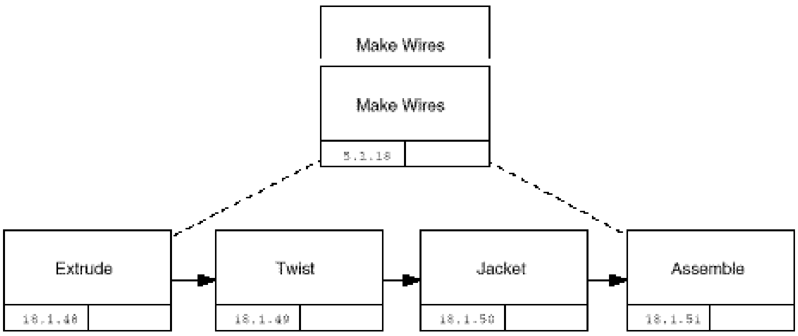

Figure C.6 : Process for manufacturing the 350-Wire [1]

The PSL-Outercore-based representation of some activities and of the process related information is :

(subactivity extrude make_harness_wire)

(subactivity twist make_harness_wire)


```
(subactivity jacket make_harness_wire)

(primitive extrude)

(primitive twist)

(primitive jacket)


(forall (?occ)

    (⇔      (occurrence_of ?occ make_harness_wire)

            (exists (?occ1 ?occ2 ?occ3)

                    (and    (occurrence_of ?occ1 extrude)

                                    (occurrence_of ?occ2 twist)

                                    (occurrence_of ?occ3 jacket)

                                    (min_precedes ?occ1 ?occ2 make_harness_wire)

                                    (min_precedes ?occ2 ?occ3 make_harness_wire))))
```

Bibliography

[1] *Federal Information Processing Standards Publication 184, Integration Definition for Information Modeling (IDEF3)*, FIPS PUB 184, National Institute of Standards and Technology, December 1993. IDEF3. Available from the Internet: <http://www.idef.com >.

[2] McCarthy, J., and Hayes, P. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*, B. Meltzer and D. Michie, eds. Edinburgh University Press, Edinburgh, 1969, pp 463-502.

[3] Pinto, J. *Temporal Reasoning in the Situation Calculus*, Technical Report KRR-TR-94-1, Department of Computer Science, University of Toronto, 1994.

[4] Reiter, R. The frame problem in the situation calculus: a simple solution (sometimes) and a completeness result for goal regression. In Vladimir Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation: Papers in Honor of John McCarthy*, 1991, pp 418-440. Academic Press, San Diego.

Index

ICS 25.040.40

Price based on 57 pages