INTERNATIONAL STANDARD

ISO 18629-43

First edition
2006-08-15

Industrial automation systems and integration — Process specification language —

Part 43: Definitional extension: Activity ordering and duration extensions

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

Partie 43: Extension de définition: Extensions de la durée et de classement d'activité

Reference number
ISO 18629-43:2006(E)

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Published in Switzerland

Contents

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 the ISO/IEC Directives, Part 2.

The main task of technical committees is to prepare International Standards. 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 document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights.

ISO 18629-43 was prepared by Technical Committee ISO/TC 184, *Industrial automation systems and integration,* Subcommittee SC 4, *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 specifications and properties related to manufacturing during all the stages of a production process.

This part of ISO 18629 provides a description of the definitional extensions of the language related to activity extensions defined within ISO 18629.

All parts of ISO 18629 are independent of any specific process representation model used in a given application. Collectively, they provide a structural framework for improving the interoperability of these applications.

Industrial automation systems and integration — Process specification language —

Part 43:
Definitional extension: Activity ordering and duration extensions

1. Scope

This part of ISO 18629 provides a specification of non-primitive concepts of the language, using a set of definitions written in the language of ISO 18629. These definitions provide an axiomatization of the semantics for terminology in this part of ISO 18629.

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

— definitions of concepts using terminology specified in ISO 18629-13.

The following is outside the scope of this part of ISO 18629:

— definitions of state and time-related concepts using only terminology specified in ISO 18629-11 and ISO 18629-12.

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) — Part 1: Specification of basic notation*

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

ISO 15531-42, *Industrial automation systems and integration — Industrial manufacturing management data — Part 42: Time Model*

ISO 18629-1: 2004, *Industrial automation systems and integration — Process specification language — Part 1: Overview and basic principles*

ISO 18629-11 2005, *Industrial automation systems and integration – Process specification language – Part 11: PSL core*

ISO 18629-12, *Industrial automation systems and integration — Process specification language — Part 12: Outer core*

ISO 18629-13, *Industrial automation systems and integration — Process specification language — Part 13: Duration and ordering theories*

3. Terms, definitions, and abbreviations

3.1 Terms and definitions

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

3.1.1
automorphism
one-to-one mapping of elements on a set that preserves the relations and functions in some model

[ISO 18629-13]

3.1.2
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]

3.1.3
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]

3.1.4
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]

3.1.5
duration
interval of time
length of a period of time, measured using a given unit of time

[ISO 15531-42]

3.1.6
endomorphism
mapping from a set onto a subset that preserves the relations and functions in some model

[ISO 18629-13]

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
homomorphism
mapping between sets preserves some relations on the elements of the set

[ISO 18629-13]

3.1.10
language
combination of a lexicon and a grammar

[ISO 18629-1]

3.1.11
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]

3.1.12
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]

3.1.13
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]

3.1.14
monomorphism
one to one mapping between sets preserves some relation on the elements of the set

[ISO 18629-13]

3.1.15
primitive concept
lexical term that has no conservative definition

[ISO 18629-1]

3.1.16
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]

3.1.17
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]

3.1.18
product
Thing or substance produced by a natural or artificial process

[ISO 10303-1]

3.1.19
resource
any device, tool and means at the disposal of the enterprise to produce goods or services

NOTE 1 Adapted from ISO 15531-1 The concept of resource as defined in ISO 15531-1 includes an assumption seeing that resources except raw material, products and components that are considered from a system theory point of view as parts of the environment of the system and then do not belong to the system itself. That is not the case here. Furthermore ISO 15531-1 definition encompasses ISO 10303-49 definition but is included in the definition that applies for this part of ISO 18629 (In addition to ISO 15531 resources of this part of ISO 18629 resources include raw materials and consumables as well as in ISO 18629-14).

NOTE 2 Resources as they are defined here include human resources considered as specific means with a given capability and a given capacity. Those means are considered as being able to be involved in the manufacturing process through assigned tasks. That does not include any modelling of an individual or common behaviour of human resource excepted in their capability to perform a given task in the manufacturing process (e.g.: transformation of raw material or component, provision of logistic services). That means that human resources are only considered, as the other, from the point of view of their functions, their capabilities and their status (e.g.: idle, busy). That excludes any modelling or representation of any aspect of individual or common «social» behaviour.

[ISO 15531-1]

3.1.20
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]

3.2 Abbreviations

— **KIF** Knowledge Interchange Format.

4. General information on ISO 18629

The parts 41 to 49 of ISO 18629[1] specify definitional extensions needed to give precise definitions and the axiomatization of non-primitive concepts of ISO 18629. Definitional extensions are extensions of ISO 18629-11 and ISO 18629-12 that introduce new items for the lexicon. The items found in definitional extensions can be completely defined in terms using theories of ISO 18629-11 and ISO 18629-12. The definitional extensions provide precise semantic definitions for elements used in the specification of individual applications or types of applications for the purpose of interoperability. Definitional extensions exist in the following categories:

— Activity Extensions;

— Temporal and State Extensions;

— Activity Ordering and Duration Extensions;

— Resource Roles;

— Resource Sets;

— Processor Activity Extensions.

Individual users or groups of users of ISO 18629 may need to extend ISO 18629 for specifying concepts that are currently absent in parts 41 to 49 of ISO 18629. They shall use the elements presented in ISO 18629 for doing so. User-defined extensions and their definitions constitute definitional extensions but shall not become part of parts 41 to 49 of ISO 18629.

[1] Certain parts are under development

Note: User-defined extensions must conform to ISO 18629 as defined in ISO 18629-1:2004, 5.1 and 5.2.

Parts 41 to 49 of ISO 18629 provide:

— the semantic definitions, using concepts in ISO 18629-11 and ISO 18629-12, of elements that are specific to the six concepts outlined above;

— a set of axioms for constraining the use of elements in definitional extensions.

The parts 41 to 49 of ISO 18629 do not provide:

— definitions and axioms for concepts that are part of the ISO 18629-11 and ISO 18629-12;

— elements that are not defined using the elements in ISO 18629-11 and ISO 18629-12;

— user-defined extensions.

5. Organization of this part of ISO 18629

The fundamental theories that constitute this part of ISO 18629 are:

— Strong partially ordered activities;

— Duration constraints for activity occurrences;

— State-based duration;

— Time-based duration;

— Duration based on state and time;

— Ordering and duration constraints on activity occurrences;

— Ordering and duration constraints on embedded activity occurrences;

— Spoilage preconditions for activities;

— Scheduled embedding constraints;

— Duration-based effects;

— Effects of activities based on duration and state;

— Complex sequence ordering relations.

All definitional extensions in this part of ISO 18629 are extensions of the ISO 18629-13, itself an extension of ISO 18629-12 and ISO 18629-11.

6. Strong partially ordered activities

This clause characterizes all definitions pertaining to Strong partially ordered activities.

6.1 Primitive lexicon of the Strong partially ordered activities

No primitive relations are introduced by the lexicon of Strong partially ordered activities.

6.2 Defined lexicon for concepts of Strong partially ordered activities

The following relations are defined in this clause:

— (same_bag ?s1 ?s2 ?a);

— (snapshot ?s1 ?s2 ?a);

— (rotate ?s ?a);

— (reflect ?s ?a);

— (flip ?s ?a);

— (turn ?s ?a);

— (bag ?occ);

— (strong_poset ?occ);

— (choice_poset ?occ);

— (complex_poset ?occ).

Each concept is described by informal semantics and a KIF axiom.

6.3 Core theories required by Strong partially ordered activities

This extension requires:

— soo.th;

— act_occ.th;

— complex.th;

— atomic.th;

— subactivity.th;

— occtree.th;

— psl_core.th.

6.4 Definitional extensions required by Strong partially ordered activities

No definitional extensions are required by Strong partially ordered activities.

6.5 Definitions of concepts for Strong partially ordered activities

The following concepts are defined for Strong partially ordered activities.

6.5.1 same_bag

This relation is used to specify which subactivity occurrences are elements of the same AND-junction.

```
(forall (?s1 ?s2 ?a) (iff (same_bag ?s1 ?s2 ?a)
(exists (?s3 ?s4)
    (and(next_subocc ?s1 ?s3 ?a)
        (iso_occ ?s3 ?s2)
        (next_subocc ?s2 ?s4 ?a)
        (iso_occ ?s4 ?s1)))))
```

6.5.2 snapshot

The subactivity occurrence ?s1 in the activity tree for ?a is order-homomorphic to the element ?s2 of the subactivity occurrence ordering.

```
(forall (?s1 ?s2 ?a) (iff (snapshot ?s1 ?s2 ?a)
(and(iso_occ ?s1 ?s2)
    (soo ?s2 ?a)
    (implies  (root_soo ?s2 ?a)
        (root ?s1 ?a))
    (forall (?s3)
        (implies  (soo_precedes ?s3 ?s2 ?a)
            (exists (?s4)
                (and(min_precedes ?s4 ?s1 ?a)
                    (or   (mono ?s3 ?s4 ?a)
                        (= ?s3 ?s4)))))))))
```

6.5.3 rotate

Within the activity tree for ?a containing ?s, the set of next subactivity occurrences is a copy of the set of siblings for ?s.

```
(forall (?s ?a) (iff (rotate ?s ?a)
(and(forall (?s1)
```

```
(implies  (next_subocc ?s ?s1 ?a)
    (exists (?s2)
        (and(sibling ?s ?s2 ?a)
            (iso_occ ?s1 ?s2)))))
(forall (?s3)
    (implies  (sibling ?s ?s3 ?a)
        (same_bag ?s ?s3 ?a))))))
```

6.5.4 reflect

Within the activity tree for ?a containing ?s, the set of next subactivity occurrences is a copy of the set of successors of ?s in the subactivity occurrence ordering.

```
(forall (?s ?a) (iff (reflect ?s ?a)
(forall (?s1)
    (iff  (next_subocc ?s ?s1 ?a)
        (exists (?s2 ?s3)
            (and(next_soo ?s2 ?s3 ?a)
                (snapshot ?s ?s2 ?a)
                (iso_occ ?s1 ?s3)))))))
```

6.5.5 flip

Within the activity tree for ?a containing ?s, the set of next subactivity occurrences is a copy of the set of successors of ?s in the subactivity occurrence ordering, together with a copy of the siblings of ?s.

```
(forall (?s ?a) (iff (flip ?s ?a)
(forall (?s1)
    (iff  (next_subocc ?s ?s1 ?a)
        (or   (exists (?s2 ?s3)
                (and(snapshot ?s ?s2 ?a)
                    (next_soo ?s2 ?s3 ?a)
                    (iso_occ ?s1 ?s3)))
            (exists (?s2)
                (and(sibling ?s ?s2 ?a)
```

(iso_occ ?s1 ?s2)))))))

6.5.6 turn

Within the activity tree for ?a containing ?s, the set of next subactivity occurrences is a copy of the set of successors of ?s in the subactivity occurrence ordering, together with a copy of the siblings of ?s that are in the same bag.

(forall (?s ?a) (iff (turn ?s ?a)

(and(exists (?s5)

(and(sibling ?s5 ?a)

(same_bag ?s ?s5 ?a)))

(forall (?s1)

(implies (next_subocc ?s ?s1 ?a)

(or (exists (?s2 ?s3)

(and(snapshot ?s ?s2 ?a)

(next_soo ?s2 ?s3 ?a)

(iso_occ ?s1 ?s3)))

(exists (?s2)

(and(sibling ?s ?s2 ?a)

(iso_occ ?s1 ?s2)))))))

6.5.7 bag

A bag is an activity tree in which all elements are rotated. Intuitively, this corresponds to a process flow diagram which contains only AND junctions with no linear orderings.

(forall (?occ) (iff (bag ?occ)

(forall (?a ?s ?occ1)

(implies (and (same_grove ?occ ?occ1)

(occurrence_of ?occ ?a)

(subactivity_occurrence ?s ?occ1))

(rotate ?s ?a)))))

6.5.8 choice_poset

A choice poset is an activity tree in which all elements are reflected. Intuitively, this corresponds to a process flow diagram which contains only OR junctions.

```
(forall (?occ) (iff (choice_poset ?occ)
(forall (?a ?s ?occ1)
    (implies  (and      (same_grove ?occ ?occ1)
                (occurrence_of ?occ ?a)
                (subactivity_occurrence ?s ?occ1))
            (reflect ?s ?a)))))
```

6.5.9 strong_poset

A strong poset is an activity tree in which all elements are flipped. Intuitively, this corresponds to a process flow diagram which contains linear orderings within AND junctions.

```
(forall (?occ) (iff (strong_poset ?occ)
(forall (?a ?s ?occ1)
    (implies  (and      (same_grove ?occ ?occ1)
                (occurrence_of ?occ ?a)
                (subactivity_occurrence ?s ?occ1))
            (flip ?s ?a)))))
```

6.5.10 complex_poset

A complex poset is an activity tree in which all elements are turned. Intuitively, this corresponds to a process flow diagram which contains both AND and OR junctions.

```
(forall (?occ) (iff (complex_poset ?occ)
(forall (?a ?s ?occ1)
    (implies  (and      (same_grove ?occ ?occ1)
                (occurrence_of ?occ ?a)
                (subactivity_occurrence ?s ?occ1))
            (turn ?s ?a)))))
```

6.6 Grammar for process descriptions of Strong partially ordered activities

The following grammar sentences describe process descriptions and auxiliary rules specified in KIF for Strong partially ordered activities.

NOTE: The function and importance of grammar sentences in ISO 18629 is explained in ISO 18629-1: 2004, 3.1.9, 4.3.4, and 5.1.

```
< strongposet_spec > ::= (and   (strong_poset ?occ)
```

< soo_axiom >)

< tree_formula > ::= (exists (< variable >+)

(and (same_tree < variable > ?occ)

(subactivity_occurrence < variable > < variable >)))

< precedes_formula > ::= (soo_precedes < variable > < variable > < term >) |

(and < precedes_formula >+)

< parallel_formula > ::= (parallel < variable > < variable > < term >) |

(and < precedes_formula >+)

< soo_axiom > ::=

(forall (?occ < variable >*)

(implies (exists (< variable >+)

(and < precedes_formula >*

< parallel_formula >*

< tree_formula >))))

7. Duration constraints for activity occurrences

This clause characterizes all definitions pertaining to Duration constraints for activity occurrences.

7.1 Primitive lexicon of Duration constraints for activity occurrences

No primitive relations are introduced by the lexicon of Duration constraints for activity occurrences.

7.2 Defined lexicon of Duration constraints for activity occurrences

The following functions are defined in this clause:

— (dur ?occ);

— (delay ?occ1 ?occ2).

The following relations are defined in this clause:

— (dur_equiv ?occ1 ?occ2);

— (delay_equiv?occ1 ?occ2);

— (constant ?a);

— (interval_duration ?a);

— (variable ?a).

Each concept is described by informal semantics and a KIF axiom.

7.3 Core theories required by Duration constraints for activity occurrences

This definitional extension requires;

— duration.th;

— psl_core.th.

7.4 Definitional extensions required by Duration constraints for activity occurrences

No definitional extensions are required by Duration constraints for activity occurrences.

7.5 Definitions of Duration constraints for activity occurrences

The following concepts are defined for Duration constraints for activity occurrences.

7.5.1 dur

The dur function is the duration between the beginof an activity occurrence and the endof an activity occurrence.

(forall (?occ)

(= (dur ?occ) (duration (beginof ?occ) (endof ?occ))))

7.5.2 delay

The delay function is the duration between the beginof one activity occurrence and the beginof another activity occurrence.

(forall (?occ1 ?occ2)

(= (delay ?occ1 ?occ2) (duration (beginof ?occ1) (beginof ?occ2))))

7.5.3 dur_equiv

Two occurrences are duration-equivalent if and only if they have the same duration.

(forall (?occ1 ?occ2) (iff (dur_equiv ?occ1 ?occ2)

(= (dur ?occ1) (dur ?occ2)))))

7.5.4 delay_equiv

Two activity occurrences are delay-equivalent if and only if there exists another activity occurrence with equal delay to the two occurrences.

```
(forall (?occ1 ?occ2) (iff (delay_equiv ?occ1 ?occ2)
(exists (?occ)
    (= (delay ?occ ?occ1) (delay ?occ ?occ2)))))
```

7.5.5 constant

An activity is constant if and only if all occurences have the same duration.

```
(forall (?a) (iff (constant ?a)
(forall (?occ1 ?occ2)
    (implies  (and      (occurrence ?occ1 ?a)
            (occurrence ?occ2 ?a))
        (dur_equiv ?occ1 ?occ2)))))
```

7.5.6 interval_duration

An activity is interval duration if and only if there exist occurrences that have the same duration.

```
(forall (?a) (iff (interval_duration ?a)
(forall (?occ1)
    (implies  (occurrence ?occ1 ?a)
        (exists (?occ2)
            (and(occurrence ?occ2 ?a)
                (dur_equiv ?occ1 ?occ2)))))))
```

7.5.7 variable

An activity is variable if and only if all occurrences of the activity have unequal durations.

```
(forall (?a) (iff (variable ?a)
(not (exists (?occ1 ?occ2)
    (and(occurrence ?occ1 ?a)
        (occurrence ?occ2 ?a)
        (dur_equiv ?occ1 ?occ2))))))
```

7.6 Grammar for Duration constraints for activity occurrences

The following grammar sentences describe process descriptions and auxiliary rules specified in KIF for Duration constraints for activity occurrences.

< constant_spec > ::= (forall (?occ)

(implies (occurrence ?occ < term >)

< duration_literal >))

< interval_spec > ::= (forall (?occ)

(implies (occurrence ?occ < term >)

< interval_axiom >))

< duration_literal > ::= (= (dur ?occ) < term >)

< interval_literal > ::= (lesser (dur ?occ) < term >) |

(lesser < term > (dur ?occ))

< interval_axiom > ::= < interval_literal > |

(not < interval_axiom >) |

({and | or} < interval_axiom >*) |

({implies | iff} < interval_axiom >)

8. State-based duration

This clause characterizes all definitions pertaining to State-based duration.

8.1 Primitive lexicon of State-based duration

No primitive relations are introduced by the lexicon of State-based duration.

8.2 Defined relations of State-based duration

The following relations are defined in this clause:

— (conditional_duration ?a) ;

— (context_duration ?a) ;

— (unconditional_duration ?a).

Each concept is described by informal semantics and a KIF axiom.

8.3 Core theories required by State-based duration

This definitional extension requires:

— duration.th;

— disc_state.th;

— occtree.th;

— psl_core.th.

8.4 Definitional extensions required by State-based duration

The following definitional extensions are required by the State-based Duration:

— actdur.def;

— state_precond.def.

8.5 Definitions of State-based duration

The following concepts are defined in the State-based duration extension.

8.5.1 conditional_duration

An activity is a conditional duration activity iff whenever two occurrences agree on state, then they agree on duration.

(forall (?a) (iff (conditional_duration ?a)

(forall (?occ1 ?occ2)

(implies (and (occurrence ?occ1 ?a)

(occurrence ?occ2 ?a)

(state_equiv ?occ1 ?occ2))

(dur_equiv ?occ1 ?occ2))))

8.5.2 context_duration

An activity is a context duration activity if and only if there exist duration-preserving fluent automorphisms.

(forall (?a) (iff (context_duration ?a)

(and (exists (?occ1)

(forall (?occ2)

(implies (and (occurrence ?occ1 ?a)

(occurrence ?occ2 ?a)

(state_equiv ?occ1 ?occ2))

(dur_equiv ?occ1 ?occ2))))

(exists (?occ3 ?occ4)

(and(occurrence ?occ3 ?a)

(occurrence ?occ3 ?a)

(state_equiv ?occ3 ?occ4)

(not (dur_equiv ?occ3 ?occ4)))))))

8.5.3 unconditional_duration

An activity is an unconditional duration activity if and only if the only duration-preserving fluent automorphism is the trivial one.

(forall (?a) (iff (unconditional_duration ?a)

(forall (?occ1)

(implies (occurrence ?occ1 ?a)

(exists (?occ2)

(and(occurrence ?occ2 ?a)

(state_equiv ?occ1 ?occ2)

(not (dur_equiv ?occ1 ?occ2)))))))

8.6 Grammar for State-based duration

The following grammar sentences describe process descriptions and auxiliary rules specified in KIF for State-based duration.

< cond_dur_spec > ::= (forall (?occ)

(implies (and (occurrence ?occ < term >)

< simple_state_axiom >)

< duration_literal >))

< context_dur_spec > ::= (forall (?occ)

(implies (and (occurrence ?occ < term >)

< state_axiom >)

< duration_literal >)) |

(forall (?occ)

(implies (and (occurrence ?occ < term >)

< simple_state_axiom >)

< interval_literal >))

9. Time-based duration

This clause characterizes all definitions pertaining to Time-based duration.

9.1 Primitive lexicon of Time-based duration

No primitive relations are introduced by the lexicon of Time-based duration.

9.2 Defined relations of Time-based duration

The following relations are defined in this clause:

— (rushhour ?a);

— (weekend ?a);

— (gridlock ?a).

Each concept is described by informal semantics and a KIF axiom.

9.3 Core theories required by Time-based duration

This definitional extension requires:

— duration.th;

— occtree.th;

— psl_core.th.

9.4 Definitional extensions required by Time-based duration

This extension requires actdur.def and time_precond.def..

9.5 Definitions of Time-based duration

The following concepts are defined for Time-based duration.

9.5.1 rushhour

An activity is a rushhour activity iff whenever two occurrences agree on their beginof timepoints, then they agree on duration.

```
(forall (?a) (iff (rushhour ?a)
(forall (?occ1 ?occ2)
    (implies  (and      (occurrence ?occ1 ?a)
                (occurrence ?occ2 ?a)
                (begin_equiv ?occ1 ?occ2))
            (dur_equiv ?occ1 ?occ2))))
```

9.5.2 weekend

An activity is a weekend activity if and only if there exist duration-preserving beginof automorphisms.

```
(forall (?a) (iff (weekend ?a)
(and      (exists (?occ1)
            (forall (?occ2)
                (implies  (and      (occurrence ?occ1 ?a)
                            (occurrence ?occ2 ?a)
                            (begin_equiv ?occ1 ?occ2))
                        (dur_equiv ?occ1 ?occ2))))
    (exists (?occ3 ?occ4)
        (and(occurrence ?occ3 ?a)
            (occurrence ?occ3 ?a)
            (begin_equiv ?occ3 ?occ4)
            (not (dur_equiv ?occ3 ?occ4)))))))
```

9.5.3 gridlock

An activity is a gridlock activity if and only if the only duration-preserving beginof automorphism is the trivial one.

```
(forall (?a) (iff (gridlock ?a)
(forall (?occ1)
    (implies  (occurrence ?occ1 ?a)
        (exists (?occ2)
            (and(occurrence ?occ2 ?a)
                (begin_equiv ?occ1 ?occ2)
```

(not (dur_equiv ?occ1 ?occ2))))))))

9.6 Grammar for process descriptions of Time-based duration

The following grammar sentences describe process descriptions specified in KIF for Time-based duration.

< rushhour_spec > ::= (forall (?occ)

(implies (and (occurrence ?occ < term >)

< simple_time_axiom >)

< duration_literal >))

< weekend_spec > ::= (forall (?occ)

(implies (and (occurrence ?occ < term >)

< time_axiom >)

< duration_literal >)) |

(forall (?occ)

(implies (and (occurrence ?occ < term >)

< simple_time_axiom >)

< interval_literal >))

10. Duration based on state and time

This clause characterizes all definitions pertaining to Duration based on state and time.

10.1 Primitive lexicon of duration based on state and time

No primitive relations are introduced by the lexicon of Duration based on state and time.

10.2 Defined lexicon of duration based on state and time

The following relations are defined in this clause:

— (mixed_duration ?a);

— (nondet_mixed_duration ?a);

— (rigid_mixed_duration ?a).

Each concept is described by informal semantics and a KIF axiom.

10.3 Core theories required by duration based on state and time

This definitional extension requires:

— duration.th;

— disc_state;

— occtree.th;

— psl_core.th.

10.4 Definitional extensions required by Duration based on state and time

This extension requires the following definitional extensions:

— actdur.def;

— time_precond.def;

— state_precond.def.

10.5 Definitions of Duration based on state and time

The following concepts are defined for duration based on state and time.

10.5.1 mixed_duration

An activity is a mixed duration activity iff whenever two occurrences agree on fluents and their beginof timepoints, then they agree on duration.

(forall (?a) (iff (mixed_duration ?a)

(forall (?occ1 ?occ2)

(implies (and (occurrence ?occ1 ?a)

(occurrence ?occ2 ?a)

(state_equiv ?occ1 ?occ2)

(begin_equiv ?occ1 ?occ2))

(dur_equiv ?occ1 ?occ2))))

10.5.2 nondet_mixed_duration

An activity is a nondeterministically mixed duration activity if and only if there exist fluent and beginof automorphisms that are also duration-preserving.

(forall (?a) (iff (nondet_mixed_duration ?a)

(and (exists (?occ1)

(forall (?occ2)

(implies (and (occurrence ?occ1 ?a)

(occurrence ?occ2 ?a)

(state_equiv ?occ1 ?occ2)

(begin_equiv ?occ1 ?occ2))

(dur_equiv ?occ1 ?occ2))))

(exists (?occ3 ?occ4)

(and(occurrence ?occ3 ?a)

(occurrence ?occ3 ?a)

(begin_equiv ?occ3 ?occ4)

(state_equiv ?occ1 ?occ2)

(not (dur_equiv ?occ3 ?occ4)))))))

10.5.3 rigid_mixed_duration

An activity is a rigid mixed duration activity if and only if the only duration-preserving beginof and fluent automorphism is the trivial one.

(forall (?a) (iff (rigid_mixed_duration ?a)

(forall (?occ1)

(implies (occurrence ?occ1 ?a)

(exists (?occ2)

(and(occurrence ?occ2 ?a)

(begin_equiv ?occ1 ?occ2)

(state_equiv ?occ1 ?occ2)

(not (dur_equiv ?occ1 ?occ2))))))))

10.6 Grammar for of Duration based on state and time

The following grammar sentences describe process descriptions specified in KIF for Duration based on state and time.

< mix_dur_spec > ::= (forall (?occ)

(implies (and (occurrence ?occ < term >)

< simple_mix_axiom >)

< duration_literal >))

< nondet_mix_dur_spec > ::= (forall (?occ)

(implies (and (occurrence ?occ < term >)

< mix_formula >)

< duration_literal >)) |

(forall (?occ)

(implies (and (occurrence ?occ < term >)

< simple_mix_axiom >)

< interval_literal >))

11. Ordering and duration constraints on activity occurrences

This clause characterizes all definitions pertaining to Ordering and duration constraints on activity occurrences.

11.1 Primitive lexicon of Ordering and duration constraints on activity occurrences

No primitive relations are introduced in the lexicon of Ordering and duration constraints on activity occurrences.

11.2 Defined lexicon of Ordering and duration constraints on activity occurrences

The following relations are defined in this clause:

— (ordered_duration ?a);

— (partial_ordered_duration ?a);

— (unordered_duration ?a).

Each concept is described by informal semantics and a KIF axiom.

11.3 Core theories required by Ordering and duration constraints on activity occurrences

This definitional extension requires:

— duration.th;

— act_occ.th;

— complex.th;

— atomic.th;

— subactivity.th;

— occtree.th;

— psl_core.th.

11.4 Definitional extensions required by Ordering and duration constraints on activity occurrences

This Extension requires:

— permute.def.

11.5 Definitions of Ordering and duration constraints on activity occurrences

The following concepts are defined for ordering and duration constraints on activity occurrences.

11.5.1 ordered_duration

An activity is ordered duration if and only if all branch automorphic occurences have the same duration.

(forall (?a) (iff (ordered_duration ?a)

(forall (?occ1 ?occ2)

(implies (and (occurrence ?occ1 ?a)

(occurrence ?occ2 ?a)

(branch_automorphic ?a ?occ1 ?occ2))

(duration_equiv ?occ1 ?occ2)))))

11.5.2 partial_ordered_duration

An activity is a context duration activity if and only if there exist occurrences such that all branch automorphic occurrences have the same duration.

(forall (?a) (iff (partial_ordered_duration ?a)

(and (exists (?occ1)

(forall (?occ2)

(implies (and (occurrence ?occ1 ?a)

(occurrence ?occ2 ?a)

(branch_automorphic ?a ?occ1 ?occ2))

(dur_equiv ?occ1 ?occ2))))

(exists (?occ3 ?occ4)

(and (occurrence ?occ3 ?a)

(occurrence ?occ3 ?a)

(branch_automorphic ?a ?occ1 ?occ2)

(not (dur_equiv ?occ3 ?occ4)))))))

11.5.3 unordered_duration

An activity is unordered duration if and only if for any occurrence there exist branch automorphic occurrences of the activity that have unequal durations.

(forall (?a) (iff (unordered_duration ?a)

(exists (?occ1 ?occ2)

(and(occurrence ?occ1 ?a)

(occurrence ?occ2 ?a)

(branch_automorphic ?a ?occ1 ?occ2)

(not (duration_equiv ?occ1 ?occ2))))))

11.6 Grammar of process descriptions for Ordering and duration constraints on activity occurrences

The following grammar sentences describe process descriptions specified in KIF for Ordering and duration constraints on activity occurrences.

< order_dur_spec > ::= (forall (?occ)

(implies (and (occurrence ?occ < term >)

< ordered_sentence >)

< duration_literal >))

< partial_order_dur_spec > ::=

(forall (?occ)

(implies (and (occurrence ?occ < term >)

< ordered_formula >)

< duration_literal >)) |

(forall (?occ)

(implies (and (occurrence ?occ < term >)

< ordered_sentence >)

< interval_literal >))

12. Ordering and duration constraints on embedded activity occurrences

This clause characterizes all definitions pertaining to Ordering and duration constraints on embedded activity occurrences.

12.1 Primitive lexicon of Ordering and duration constraints on embedded activity occurrences

No primitive relations are defined by the lexicon of Ordering and duration constraints on embedded activity occurrences.

12.2 Defined lexicon of Ordering and duration constraints on embedded activity occurrences

The following relations are defined in this clause:

— (embed_duration ?a);

— (partial_embed_duration ?a);

— (nonembed_duration ?a).

Each concept is described by informal semantics and a KIF axiom.

12.3 Core theories required by Ordering and duration constraints on embedded activity occurrences

This theory requires:

— duration.th;

— act_occ.th;

— complex.th;

— atomic.th;

— subactivity.th;

— occtree.th;

— psl_core.th.

12.4 Definitional extensions required by Ordering and duration constraints on embedded activity occurrences

Ordering and duration constraints on embedded activity occurrences requires:

— permute.def.

12.5 Definitions of Ordering and duration constraints on embedded activity occurrences

The following concepts are defined for Ordering and duration constraints on embedded activity occurrences.

12.5.1 embed_duration

An activity is embed duration if and only if all branch automorphic occurences in the same tree have the same duration.

```
(forall (?a) (iff (embed_duration ?a)

(forall (?occ1 ?occ2)

    (implies  (and      (occurrence ?occ1 ?a)

                (occurrence ?occ2 ?a)

                (same_grove ?occ1 ?occ2)

                (branch_automorphic ?occ1 ?occ2))

            (duration_equiv ?occ1 ?occ2)))))
```

12.5.2 partial_embed_duration

An activity is a partial embed duration activity if and only if there exist occurrences such that all branch automorphic occurrences in the same tree have the same duration.

```
(forall (?a) (iff (partial_embed_duration ?a)

(and    (exists (?occ1)

            (forall (?occ2)

                (implies  (and  (occurrence ?occ1 ?a)

                                (occurrence ?occ2 ?a)

                                (same_grove ?occ1 ?occ2)

                                (branch_automorphic ?occ1 ?occ2))

                            (dur_equiv ?occ1 ?occ2))))

    (exists (?occ3 ?occ4)

            (and    (occurrence ?occ3 ?a)

                (occurrence ?occ3 ?a)

                (same_grove ?occ1 ?occ2)
```

(branch_automorphic ?occ1 ?occ2)

(not (dur_equiv ?occ3 ?occ4)))))))

12.5.3 nonembed_duration

An activity is nonembed duration if and only if for any occurrence there exist branch automorphic occurrences of the activity in the same tree that have unequal durations.

(forall (?a) (iff (nonembed_duration ?a)

(exists (?occ1 ?occ2)

(and(occurrence ?occ1 ?a)

(occurrence ?occ2 ?a)

(same_grove ?occ1 ?occ2)

(branch_automorphic ?occ1 ?occ2)

(not (duration_equiv ?occ1 ?occ2))))))

12.6 Grammar for Ordering and duration constraints on embedded activity occurrences

The grammar for process descriptions of Ordering and duration constraints on embedded activity occurrences:

< embed_dur__spec > ::= (forall (?occ)

(implies (and (occurrence ?occ < term >)

(same_tree < variable > ?occ))

< duration_literal >))

< embed_interval_spec > ::= (forall (?occ)

(implies (and (occurrence ?occ < term >)

(same_tree < variable > ?occ))

< interval_axiom >))

13.Spoilage preconditions for activities

This clause characterizes all definitions pertaining to Spoilage preconditions for activities.

13.1 Primitive lexicon of Spoilage preconditions for activities

No primitive relations are introduced by the lexicon of Spoilage preconditions for activities.

13.2 Defined lexicon of Spoilage precondition for activities

The following relations are defined in this clause:

— (spoilage ?a);

— (possible_spoilage ?a);

— (nonspoilage ?a).

Each concept is described by informal semantics and a KIF axiom.

13.3 Theories required by Spoilage preconditions for activities

This theory requires:

— duration.th;

— occtree.th;

— psl_core.th.

13.4 Definitional extensions required by Spoilage preconditions for activities

Spoilage preconditions for activities requires the following definitional extensions:

— occ_precond.def;

— time_precond.def;

— precond.def.

13.5 Definitions of Spoilage preconditions for activities

The following concepts are defined for Spoilage preconditions for activities.

13.5.1 spoilage

An activity is a spoilage if and only if legal occurrences depend on the delay between other activity occurrences. In particular, activity occurrences that are in the same orbit of occurrence tree endormorphisms are also in the same orbit of delay automorphisms.

```
(forall (?a) (iff (spoilage ?a)
(forall (?s1 ?s2)
    (implies  (and      (occurrence_of ?s1 ?a)
                (occurrence_of ?s2 ?a)
                (delay_equiv ?s1 ?s2)
                (tree_equiv ?s1 ?s2))
```

(legal_equiv ?s1 ?s2)))))

13.5.2 possible_spoilage

An activity is possible spoilage if and only if there exist activity occurrences that are in the same orbit of occurrence tree endormorphisms and that are also in the same orbit of delay automorphisms.

(forall (?a) (iff (possible_spoilage ?a)

(exists (?s1)

(and(occurrence_of ?s1 ?a)

(forall (?s2)

(implies (and (occurrence_of ?s2 ?a)

(delay_equiv ?s1 ?s2)

(tree_equiv ?s1 ?s2))

(legal_equiv ?s1 ?s2))))))

13.5.3 nonspoilage

An activity is nonspoilage if and only if there is no relationship between legal occurrences and the delay between other activity occurrences. In particular, the only occurrence tree endomorphism that preserves the delay between other activity occurrences is the trivial one.

(forall (?a) (iff (nonspoilage ?a)

(forall (?s1)

(implies (occurrence_of ?s1 ?a)

(exists (?s2)

(and(occurrence_of ?s2 ?a)

(delay_equiv ?s1 ?s2)

(tree_equiv ?s1 ?s2)

(not (legal_equiv ?s1 ?s2))))))))

13.6 Grammar for process descriptions of Spoilage preconditions for activities.

The grammar for process descriptions of Spoilage preconditions for activities:

< occ_constrained_precond > ::=

(forall (?s)

({implies | iff} (and (occurrence ?s < term >)

(legal ?s)

(ubiquitous < term > < term >))

{< leaf_delay_axiom > | < inner_delay_axiom}))

< leaf_delay_formula > ::= (exists (?occ ?s1)

(and (occurrence ?occ < term >)

(leaf_occ ?s1 ?occ)

< delay_literal >

(= ?s (successor < term > ?s1))))

< leaf_delay_axiom > ::= < leaf_delay_formula > |

(not < leaf_delay_formula >)

< inner_delay_formula > ::= (exists (?occ ?s1 ?s2)

(and (occurrence ?occ < term >)

< delay_literal >

(subactivity_occurrence ?s1 ?occ)

(subactivity_occurrence ?s2 ?occ)

(precedes ?s1 ?s)

(precedes ?s ?s2)))

< inner_delay_axiom > ::= < inner_delay_formula > |

(not < inner_delay_formula >)

14.Scheduled embedding constraints

This clause characterizes all definitions pertaining to Scheduled embedding constraints.

14.1 Primitive lexicon of Scheduled embedding constraints

No primitive relations are introduced by the lexicon of Scheduled embedding constraints.

14.2 Defined lexicon of Scheduled embedding constraints

The following relations are defined in this clause:

— (scheduled ?occ);

— (partial_scheduled ?occ);

— (unscheduled ?occ).

Each concept is described by informal semantics and a KIF axiom.

14.3 Core theories required by Scheduled embedding constraints

This theory requires:

— duration.th;

— act_occ.th;

— complex.th;

— atomic.th;

— subactivity.th;

— occtree.th;

— psl_core.th.

14.4 Definitional extensions required by Scheduled embedding constraints

This extension requires the following definitional extensions:

— embedding.def;

— time_precond.def.

14.5 Definitions of Scheduled embedding constraints

The following concepts are defined for Scheduled embedding constraints:.

14.5.1 scheduled

An activity occurrence ?occ is scheduled if and only if every subactivity occurrence of ?occ is delay constrained.

(forall (?occ) (iff (scheduled ?occ)

```
(forall (?a ?s1 ?s2 ?s3)
    (implies  (and      (occurrence_of ?occ ?a)
                (root_occ ?s3 ?occ)
                (subactivity_occurrence ?s1 ?occ)
                (iso_occ ?s1 ?s2)
                (embed_tree ?s1 ?s2 ?s3 ?a)
                (delay_equiv ?s1 ?s2))
            (subocc_equiv ?s1 ?s2 ?s3 ?a)))))
```

14.5.2 partial_scheduled

An activity occurrence ?occ is partially scheduled if and only if some subactivity occurrences of ?occ are delay constrained.

```
(forall (?occ) (iff (partial_scheduled ?occ)
(exists (?a ?s ?s1 ?s3 ?s4)
    (and      (occurrence_of ?occ ?a)
        (root_occ ?s ?occ)
        (subactivity_occurrence ?s1 ?occ)
        (forall (?s2)
            (implies  (and      (iso_occ ?s1 ?s2)
                        (embed_tree ?s1 ?s2 ?s ?a)
                        (delay_equiv ?s1 ?s2))
                    (subocc_equiv ?s1 ?s2 ?s ?a))))
        (subactivity_occurrence ?s3 ?occ)
        (iso_occ ?s3 ?s4)
        (delay_equiv ?s3 ?s4)
        (embed_tree ?s3 ?s4 ?s ?a)
        (not (subocc_equiv ?s3 ?s4 ?s ?a))))))
```

14.5.3 unscheduled

An activity occurrence ?occ is unscheduled if and only if none of the subactivity occurrences of ?occ are delay constrained.

```
(forall (?occ) (iff (unscheduled ?occ)
(forall (?s ?s1 ?a)
    (implies  (and      (occurrence_of ?occ ?a)
                (root_occ ?s ?occ)
                (subactivity_occurrence ?s1 ?occ))
            (exists (?s2)
                (and(iso_occ ?s1 ?s2)
                    (embed_tree ?s1 ?s2 ?s ?a)
                    (delay_equiv ?s1 ?s2)
                    (not (subocc_equiv ?s1 ?s2 ?s ?a))))))))
```

14.6 Grammar for Scheduled embedding constraints

The grammar for process descriptions of Scheduled embedding constraints :

```
< schedule_axiom > ::=  (forall (?s ?occ < variable >*)
                        < schedule_formula >)
< partial_schedule_axiom > ::=  (forall (?s ?occ < variable >+)
                        < partial_schedule_sentence >))
< schedule_formula > ::=  ({implies | and} < subocc_formula >
                            < delay_literal >)
< schedule_axiom > ::=  < schedule_formula> |
                (and < schedule_axiom > < schedule_axiom >+)
< partial_schedule_formula > ::=  ({implies | and} < subocc_formula >
                                < delay_interval_axiom >)
< partial_schedule_sentence > ::=
    < partial_schedule_formula> |
    (and < partial_schedule_sentence > < partial_schedule_sentence >+)
```

15.Duration-based effects

This clause characterizes all definitions pertaining to Duration-based effects.

15.1 Primitive lexicon of Duration-based effects

No primitive relations are introduced in the lexicon of Duration-based effects.

15.2 Defined lexicon of Duration-based effects

The following relations are defined in this clause:

— (duration_effects ?a);

— (partial_duration_effects ?a);

— (nonduration_effects ?a).

Each concept is described by informal semantics and a KIF axiom.

15.3 Core theories required by Duration-based effects

This theory requires:

— duration.th;

— disc_state.th;

— occtree.th;

— psl_core.th.

15.4 Definitional extensions required by Duration-based effects

Duration-based effects requires the following definitional extensions:

— actdur.def;

— effects.def.

15.5 Definitions of Duration-based effects

The following concepts are defined for Duration-based effects.

15.5.1 duration_effects

An activity is a duration effect activity iff whenever any two occurrences of the activity agree on their duration, then they agree on the effects (i.e. the states that hold after the occurrence).

```
(forall (?a) (iff (duration_effects ?a)

(forall (?s1 ?s2)

    (implies  (and      (occurrence ?s1 ?a)

                (occurrence ?s2 ?a)
```

(dur_equiv ?s1 ?s2))

(effects_equiv ?a ?s1 ?s2)))))

15.5.2 partial_duration_effects

An activity is a partially duration constrained activity if and only if there exist effect-preserving duration automorphisms.

(forall (?a) (iff (partial_duration_effects ?a)

(and (exists (?s1)

(forall (?s2)

(implies (and (occurrence ?s1 ?a)

(occurrence ?s2 ?a)

(dur_equiv ?s1 ?s2))

(effects_equiv ?a ?s1 ?s2))))

(exists (?s3 ?s4)

(and (occurrence ?s3 ?a)

(occurrence ?s4 ?a)

(dur_equiv ?s3 ?s4)

(not (effects_equiv ?a ?s3 ?s4)))))))

15.5.3 nonduration_constraints

An activity is a nonduration effects activity if and only if the only effects-preserving duration automorphism is the trivial one.

(forall (?a) (iff (nonduration_effects ?a)

(forall (?s1)

(implies (occurrence ?s1 ?a)

(exists (?s2)

(and (occurrence ?s2 ?a)

(dur_equiv ?s1 ?s2)

(not (effects_equiv ?a ?s1 ?s2))))))))

15.6 Grammar for Duration-based effects

The grammar for the process descriptions of Duration-based effects

< duration_effect_axiom > ::= (forall (?s)

(implies (and (occurrence ?s <term >)

< duration_literal >)

< simple_holds_axiom >)))

< partial_dur_effect > ::= (forall (< variable >*)

(implies (and (occurrence < variable > < term >)

< interval_axiom >)

< simple_holds_axiom >)))

16. Effects of activities based on duration and time

This clause characterizes all definitions pertaining to Effects of activities based on duration and time.

16.1 Primitive lexicon of Effects of activities based on duration and time

No primitive relations are introduced in the lexicon of Effects of activities based on duration and time.

16.2 Defined lexicon of Effects of activities based on duration and time

The following relations are defined in this clause:

— (maintain_effects ?a);

— (partial_maintain ?a);

— (nonmaintain ?a).

Each concept is described by informal semantics and a KIF axiom.

16.3 Core theories required by Effects of activities based on duration and time

This theory requires:

— duration.th;

— disc_state.th;

— occtree.th;

— psl_core.th.

16.4 Definitional extensions required by Effects of activities based on duration and time

This extension requires the following definitional extensions:

— actdur.def;

— effects.def;

— state_precond.def.

16.5 Definitions of Effects of activities based on duration and time

The following concepts are defined for Effects of activities based on duration and time.

16.5.1 maintain_effects

An activity is a maintain effect activity iff whenever any two occurrences of the activity agree on their duration and their state, then they agree on the effects (i.e. the states that hold after the occurrence).

```
(forall (?a) (iff (maintain_effects ?a)

(forall (?s1 ?s2)

    (implies  (and      (occurrence ?s1 ?a)

                (occurrence ?s2 ?a)

                (state_equiv ?s1 ?s2)

                (dur_equiv ?s1 ?s2))

            (effects_equiv ?a ?s1 ?s2)))))
```

16.5.2 partial_maintain

An activity is a partially maintained activity if and only if there exist effect-preserving automorphisms that preserve both duration and state.

```
(forall (?a) (iff (partial_maintain ?a)

(and    (exists (?s1)

            (forall (?s2)

                (implies  (and     (occurrence ?s1 ?a)

                            (occurrence ?s2 ?a)

                            (state_equiv ?s1 ?s2)

                            (dur_equiv ?s1 ?s2))

                        (effects_equiv ?a ?s1 ?s2))))

    (exists (?s3 ?s4)

            (and    (occurrence ?s3 ?a)
```

(occurrence ?s4 ?a)

(dur_equiv ?s3 ?s4)

(not (effects_equiv ?a ?s3 ?s4)))))))

16.5.3 nonmaintain

An activity is a nonmaintain effects activity if and only if the only effects-preserving duration and fluent automorphism is the trivial one.

(forall (?a) (iff (nonmaintain ?a)

(forall (?s1)

(implies (occurrence ?s1 ?a)

(exists (?s2)

(and (occurrence ?s2 ?a)

(dur_equiv ?s1 ?s2)

(not (effects_equiv ?a ?s1 ?s2)))))))

16.6 Grammar for Effects of activities based on duration and time

The grammar for the process descriptions of Effects of activities based on duration and time

< maintain_effect_axiom > ::= (forall (?s)

(implies (and (occurrence ?s <term >)

< duration_literal >

< simple_state_axiom >)

< simple_holds_axiom >)))

< partial_maintain_axiom > ::= (forall (< variable >)

(implies (and (occurrence < variable > < term >)

< duration_literal >

< state_axiom >)

< simple_holds_axiom >))) |

(forall (< variable >*)

(implies (and (occurrence < variable > < term >)

< interval_literal >

< simple_state_axiom >)

< simple_holds_axiom >))) |

17. Complex sequence ordering relations

This clause characterizes all definitions pertaining to Complex sequence ordering relations.

17.1 Primitive lexicon of Complex sequence ordering relations

No primitive relations are introduced in the lexicon of Complex sequence ordering relations.

17.2 Defined lexicon of Complex sequence ordering relations

The following relations are defined in this clause:

— (coo_precedes ?occ1 ?occ2 ?a);

— (strong_parallel ?occ1 ?occ2 ?a);

— (atomocc ?occ).

Each concept is described by informal semantics and a KIF axiom.

17.3 Theories required by Complex sequence ordering relations

This theory requires

— soo.th;

— actocc.th;

— complex.th;

— atomic.th;

— subactivity.th;

— occtree.th;

— psl_core.th.

17.4 Definitional extensions required by Complex sequence ordering relations

This extension requires the following definitional extension:

— strongposets.def.

17.5 Definitions of Complex sequence ordering relations

The following concepts are defined for Complex sequence ordering relations.

17.5.1 coo_precedes

A complex activity occurrence ?occ1 coo_precedes a complex activity occurrence ?occ2 iff every atomic subactivity occurrence of ?occ1 soo_precedes every atomic subactivity occurrence of ?occ2.

```
(forall (?occ1 ?occ2 ?a) (iff (coo_precedes ?occ1 ?occ2 ?a)
(forall (?s1 ?s2)
    (implies  (and      (subactivity_occurrence ?s1 ?occ1)
                        (subactivity_occurrence ?s2 ?occ2)
                        (atomocc ?s1)
                        (atomocc ?s2))
              (soo_precedes ?s1 ?s2 ?a)))))
```

17.5.2 strong_parallel

A complex activity occurrence ?occ1 is strong_parallel with a complex activity occurrence ?occ2 iff any atomic subactivity occurrence of ?occ1 is in the same bag as an atomic subactivity occurrence of ?occ2.

```
(forall (?occ1 ?occ2 ?a) (iff (strong_parallel ?occ1 ?occ2 ?a)
(forall (?s1 ?s2)
    (implies  (and      (subactivity_occurrence ?s1 ?occ1)
                        (subactivity_occurrence ?s2 ?occ2)
                        (atomocc ?s1)
                        (atomocc ?s2))
              (same_bag ?s1 ?s2 ?a)))))
```

17.5.3 atomocc

An activity occurrence is atomocc iff it is the occurrence of an atomic activity.

```
(forall (?s) (iff (atomocc ?s)
(exists (?a)
    (and(atomic ?a)
              (occurrence_of ?s ?a)))))
```

Annex A
(normative)
ASN.1 Identifier of ISO 18629-43

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

```
iso standard 18629 part 43 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)
Example of process description using ISO 18629-43

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: 2005, Annex C to illustrate the application of activity ordering and duration extension concepts in the specification of the manufacturing process of a product named GT-350.

B.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: 2005, Table C.1), 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 B.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.

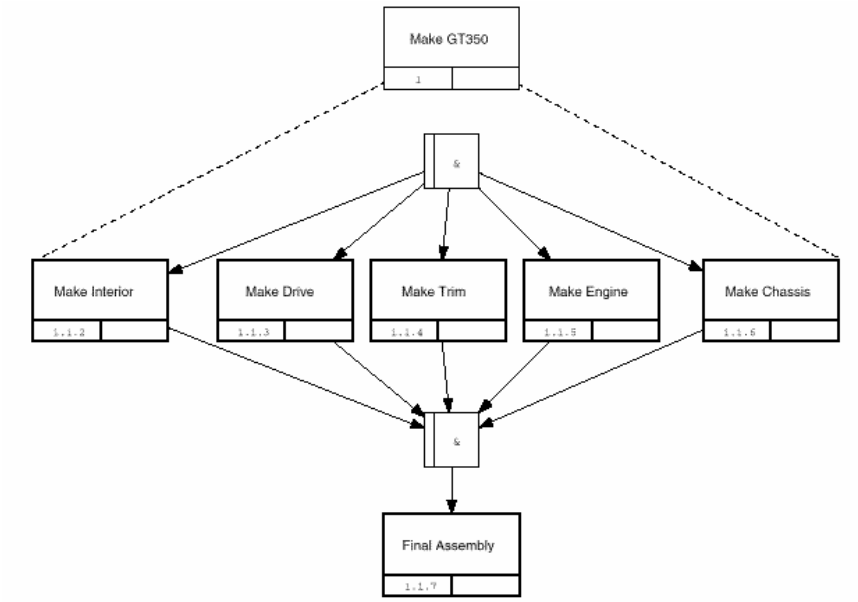

Figure B1: TOP level process for manufacturing a GT350 [5]

The PSL -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)
                (soo_precedes (soomap ?occ1) (soomap ?occ6)  make_gt350)
                (soo_precedes (soomap ?occ2) (soomap ?occ6)  make_gt350)
                (soo_precedes (soomap ?occ3) (soomap ?occ6)  make_gt350)
                (soo_precedes (soomap ?occ4) (soomap ?occ6)  make_gt350)
```

(soo_precedes (soomap ?occ5) (soomap ?occ6) make_gt350)

(strong_parallel ?occ1 ?occ2 make_gt350)

(strong_parallel ?occ2 ?occ3 make_gt350)

(strong_parallel ?occ3 ?occ4 make_gt350)

(strong_parallel ?occ4 ?occ5 make_gt350)))))))

(forall (?occ)

(implies (occurrence_of make_gt350)

(strong_poset ?occ)))

In this representation, the soo_precedes relation is used to specify the ordering constraints among the subactivity occurrences of make_chassis, make_interior, make_drive, make_engine, and final_assembly. Informally, each arrow in Figure B.1 corresponds to a soo_precedes formula. The soomap function is used to distinguish among any possible multiple occurrences of the subactivities.

The subactivities make_chassis, make_interior, make_drive, make_engine all occur in parallel, so their ordering constraints are specified by the strong_parallel relation. Occurrences of make_gt350 are strong posets, since all of the parallel subactivities must occur before final_assembly.

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 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-Outercore presented in this part 12 of the ISO 18629 standard.

B.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 B.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.

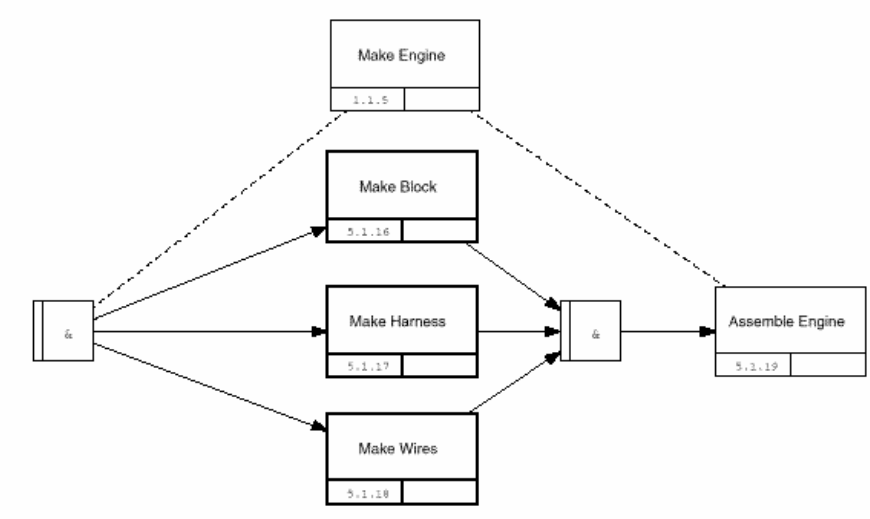

Figure B.2: PROCESS for manufacturing the 350–Engine [5]

The PSL-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)

(= (duration (beginof ?occ1) (endof ?occ1)) 10)

(= (duration (beginof ?occ2) (endof ?occ2)) 5)

(= (duration (beginof ?occ3) (endof ?occ3)) 12)

```
                    (= (beginof ?occ4) (time_add (endof ?occ3) 10))
                    (subactivity_occurrence ?occ1 ?occ)
                    (subactivity_occurrence ?occ2 ?occ)
                    (subactivity_occurrence ?occ3 ?occ)
                    (subactivity_occurrence ?occ4 ?occ)
                    (soo_precedes (soomap ?occ1) (soomap ?occ4)  make_gt350)
                    (soo_precedes (soomap ?occ2) (soomap ?occ4)  make_gt350)
                    (soo_precedes (soomap ?occ3) (soomap ?occ4)  make_gt350)))))
                    (strong_parallel ?occ1 ?occ2 make_gt350)
                    (strong_parallel ?occ2 ?occ3 make_gt350)))))))

(forall (?occ)
    (implies   (occurrence_of make_engine)
               (strong_poset ?occ)))
```

In this representation, the soo_precedes relation is used to specify the ordering constraints among the subactivity occurrences of make_block, make_harness, make_wires, and make_engine. Informally, each arrow in Figure B2 corresponds to a soo_precedes formula. The soomap function is used to distinguish among any possible multiple occurrences of the subactivities.

The subactivities make_block, make_harness, make_wire, all occur in parallel, so their ordering constraints are specified by the strong_parallel relation. Occurrences of make_engine are strong posets, since all of the parallel subactivities must occur before assemble_engine.

In addition, the representation specifies that the duration of the occurrence of make_block is 10 units, the duration of the occurrence of make_harness is 5 units, and the duration of make_wires is 12 units,

Finally, the representation uses the time_add function to specify that the occurrence of assemble_engine begins 10 time units after the occurrence of make_wires.

B.3 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 B.3.

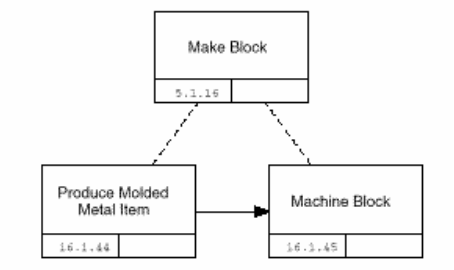

Figure B.3: PROCESS for manufacturing the 350–Block [5]

The PSL-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)

(= (beginof ?occ2) (time_add (endof ?occ1) 12))

(soo_precedes (soomap ?occ1) (soomap ?occ2) make_block)))))

This representation uses the time_add function to specify that the occurrence of the machine_block subactivity begins 12 time units after the end of the occurrence of the produce_molded_metal subactivity.

B.4 Make Harness

The 350-Harness (Figure B.4) is manufactured as part of the 350–Engine sub-assembly. This involves work performed at the wire and cable department. Figure B.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.

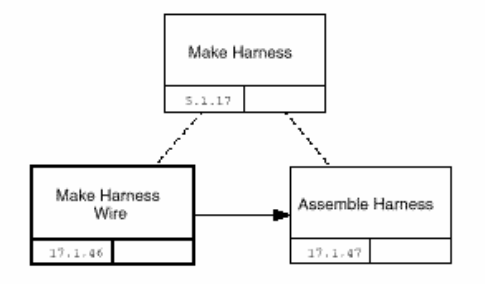

Figure B.4: PROCESS for manufacturing the 350–Harness [5]

The PSL-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)
                        (soo_precedes (soomap ?occ3) (soomap ?occ2) make_harness)))))
```

This representation formalizes the specification of the process in Figure B.5. In this representation, the soo_precedes relation is used to specify the ordering constraints among the subactivity occurrences of make_harness_wires and assemlbe_wires. Informally, each arrow in Figure C.5 corresponds to a soo_precedes formula. The soomap function is used to distinguish among any possible multiple occurrences of the subactivities.

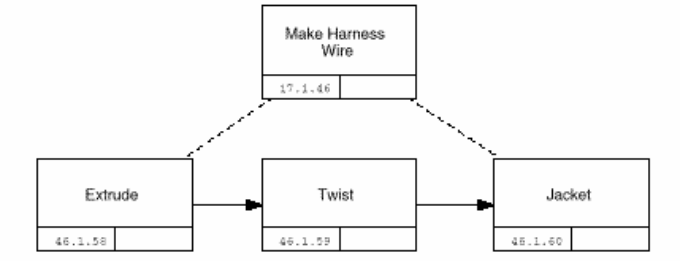

Figure B.5: PROCESS for manufacturing the harness wire [5]

B.5 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.

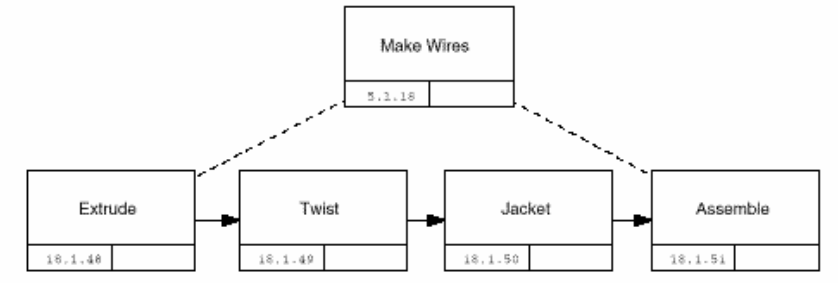

Figure B.6 : Process for manufacturing the 350-Wire [5]

The PSL-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 ?occ4)

(and(occurrence_of ?occ1 extrude)

(occurrence_of ?occ2 twist)

(occurrence_of ?occ3 jacket)

(occurrence_of ?occ4 assemble)

(soo_precedes (soomap ?occ1) (soomap ?occ2) make_harness_wire)

(soo_precedes (soomap ?occ3) (soomap ?occ4) make_harness_wire)

(soo_precedes (soomap ?occ2) (soomap ?occ3)
make_harness_wire))))

This representation formalizes the specification of the process in Figure B.6. In this representation, the soo_precedes relation is used to specify the ordering constraints among the subactivity occurrences of extrude, twist, jacket, and assemble. Informally, each arrow in Figure B.6 corresponds to a soo_precedes formula. The soomap function is used to distinguish among any possible multiple occurrences of the subactivities.

Bibliography

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

[2] ISO 10303-49, *Industrial automation systems and integration ─ Product data representation and exchange ─ Part 49 : Integrated generic resources: Process structure and properties*

[3] ISO 18629-14, *Industrial automation systems and integration ─ Process specification language ─ Part 14 : Resource theories*

[4] ISO 18629-44, *Industrial automation systems and integration ─ Process specification language ─ Part 44 : Definitional extension : Resource extensions*

[5] *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 >

Index

ICS 25.040.40

Price based on 53 pages