TECHNICAL SPECIFICATION

ISO/TS 15926-7

First edition
2011-10-15

Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities —

Part 7: Implementation methods for the integration of distributed systems: Template methodology

Systèmes d'automatisation industrielle et intégration — Intégration de données de cycle de vie pour les industries de «process», y compris les usines de production de pétrole et de gaz —

Partie 7: Méthodes de mise en œuvre pour l'intégration de systèmes distribués: Méthodologie de modèle

Reference number
ISO/TS 15926-7:2011(E)

COPYRIGHT PROTECTED DOCUMENT

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

Figures

Tables

Foreword

ISO (the International Organization for Standardization) 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 organizations, 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 adopted by the 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.

In other circumstances, particularly when there is an urgent market requirement for such documents, a technical committee may decide to publish other types of normative document:

— an ISO Publicly Available Specification (ISO/PAS) represents an agreement between technical experts in an ISO working group and is accepted for publication if it is approved by more than 50% of the members of the parent committee casting a vote;

— an ISO Technical Specification (ISO/TS) represents an agreement between the members of a technical committee and is accepted for publication if it is approved by 2/3 of the members of the committee casting a vote.

An ISO/PAS or ISO/TS is reviewed every three years with a view to deciding whether it can be transformed into an International Standard.

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/TS 15926-7 was prepared by Technical Committee ISO/TC 184, *Automation systems and integration*, Subcommittee SC 4, *Industrial data*.

ISO 15926 is organized as a series of parts, each published separately. The structure of ISO 15926 is described in ISO 15926-1.

Each part of ISO 15926 is a member of the following series: data model, reference data, implementation methods, conformance testing methodology and framework, characterization methods, abstract test suites. This part of ISO 15926 is a member of the implementation methods series.

A complete list of parts of ISO 15926 is available from the following URL:

`http://www.tc184-sc4.org/titles/OIL_GAS_Titles.htm`

Introduction

ISO 15926 is an International Standard for the representation of process plant life-cycle information. This representation is specified by a generic, conceptual data model that is suitable as the basis for implementation in a shared database or data warehouse. The data model is designed to be used in conjunction with reference data: standard instances that represent information common to a number of users, process plants, or both. The support for a specific life-cycle activity depends on the use of appropriate reference data in conjunction with the data model.

ISO 15926 is organized as a number of parts, each published separately. This part of ISO 15926 specifies the template methodology and is independent of implementation methodologies and computer languages.

This part of ISO 15926 deals with the template methodology, which defines strict models of ISO 15926-2 conceptual model elements which can be used in data modelling, integration and interoperability methods. This part of ISO 15926 is independent of implementation languages, implementation infrastructure and test methods.

This part of ISO 15926 serves as the basis for implementation languages, implementation infrastructure and test methods.

This part of ISO 15926 addresses:

— the method of first-order logic used;

— template syntax;

— the semantics of templates;

— the method of template expansion;

— the definition of proto templates;

— the initial set of templates.

Readers of this part of ISO 15926 require an understanding of conceptual data models and of ISO 15926-2.

The target audiences for this part of ISO 15926 are as follows:

— technical managers wishing to determine whether ISO 15926 is appropriate for their business needs;

— implementers.

In this part of ISO 15926, the same English language word might be used to refer to a real world thing, to an EXPRESS representation of the real world thing, or to an RDF/XML representation of the real-world thing. These uses are distinguished by the following typographic conventions:

— if a word or phrase occurs in the same typeface as the surrounding narrative text, the word or phrase refers to the real-world thing;

— if the word or phrase occurs in **bold** typeface, it refers to the EXPRESS representation from the ISO 15926-2 data model;

EXAMPLE 1 **class_of_inanimate_physical_object**

— if the word or phrase occurs in **bold camel case** typeface, it refers to a term in the ISO 15926-2 language, as specified in 4.1;

EXAMPLE 2 **ClassOfApprovalByStatus**

— if the word or phrase occurs in *italic camel case* typeface, it refers to a template name.

EXAMPLE 3 *RTriple*(z, x, y)

References to identifiers in examples are fictitious.

In this part of ISO 15926, diagrams are occasionally used to illustrate ISO 15926-2 modelling patterns. The symbols used in instance diagrams are a minor modification of the symbols used in ISO 15926-2.

Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities —

Part 7: Implementation methods for the integration of distributed systems: Template methodology

1 Scope

This part of ISO 15926 is a specification for data exchange and life-cycle information integration using templates based on the data model of ISO 15926-2. This part of ISO 15926 provides a methodology for data integration of ontologies using mathematical first-order logic, which makes it independent of computer languages.

The following are within the scope of this part of ISO 15926:

— representation of the ISO 15926-2 EXPRESS model in formal logic;

— criteria for template definitions;

— methods of template expansion and verification;

— initial set of template definitions.

NOTE For practical guidelines to information representation using templates, see Reference [17].

The following are outside the scope of this part of ISO 15926:

— implementation in computer-interpretable languages;

— storage and retrieval;

— security.

2 Terms, definitions, and abbreviated terms

2.1 Terms and definitions

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

2.1.1
base template
template with only entity types in the expansion of its template axiom

2.1.2
class
category or division of things based on one or more critera for inclusion and exclusion

NOTE 1 A class need not have any known members (things that satisfy its criteria for membership).

NOTE 2 Because of the spatio-temporal paradigm used to define individuals in ISO 15926, all classes are non-well-founded sets.

NOTE 3 Adapted from ISO 15926-1:2004, definition 3.1.1.

2.1.3
class template
template for making statements about classes

2.1.4
conceptual data model
data model in the three-schema architecture defined in ISO/TR 9007, in which the structure of data is represented in a form independent of any physical storage or external presentation format

2.1.5
core class
class that is a commonly used subdivision corresponding to terms used in common language

NOTE The conditions for membership are often not formally defined; understanding of the class may be conveyed by example.

EXAMPLE Pipe, floor, pump, and light bulb are all core classes.

[ISO 15926-1:2004, definition 3.1.4]

2.1.6
core template
RDL template for which all reference data items in the expansion of its template axiom are core classes

2.1.7
data store
computer system that allows data to be stored for future reference

[ISO 15926-1:2004, definition 3.1.6]

2.1.8
data type
domain of values

[ISO 10303-11:2004, definition 3.3.5]

2.1.9
data warehouse
data store in which related data are merged to provide an integrated set of data containing no duplication or redundancy of information, and which supports many different application viewpoints

[ISO 15926-1:2004, definition 3.1.7]

2.1.10
document
thing serving as a representation of information by means of symbolic marks

NOTE The word “document” is used in a wider sense. Next to the information content of customary (paper) documents (not a paper document itself, because that is an instance of PhysicalObject), such as equipment data sheets or purchase orders, it can also be used for other sets of data, like the transaction data that are input to an engineering program or data sets that are exchanged between systems of business partners.

2.1.11
entity
class of information defined by common properties

[ISO 10303-11:2004, definition 3.3.6]

2.1.12
entity instance
named unit of data which represents a unit of information within the class defined by an entity

NOTE 1 It is a member of the domain established by an entity data type.

NOTE 2 Adapted from ISO 10303-11:2004, definition 3.3.8.

2.1.13
first-order logic
symbolized reasoning in which each sentence, or statement, is broken down into a subject and a predicate

NOTE 1 The predicate modifies or defines the properties of the subject. In first-order logic, a predicate can only refer to a single subject.

NOTE 2 First-order logic is also known as first-order predicate calculus or first-order functional calculus.

2.1.14
individual template
template for making statements about individuals

2.1.15
instance
named value

[ISO 10303-11:2004, definition 3.3.10]

2.1.16
ISO 15926-2 language
first-order language in which the ISO 15926-2 data model is expressed

NOTE The ISO 15926-2 language is specified in 4.1.

2.1.17
life-cycle information
information about a **possible_individual**, collected at any point in time during the life-cycle of that individual

NOTE In ISO 15926-2:2003, definition 3.1.6, ñindividualò is defined as: “thing that exists in space and time”.

2.1.18
RDL template
template with at least one reference data item in the expansion of its template axiom

2.1.19
reference data
process plant life-cycle data that represents information about classes or individuals which are common to many process plants or of interest to many users

[ISO 15926-1:2004, definition 3.1.18]

2.1.20
reference data library
RDL
managed collection of reference data

[ISO 15926-1:2004, definition 3.1.19]

NOTE In ISO/TS 15926-8 "RDL" and "ontology" are used interchangeably. An alternative term is "information model".

2.1.21
reification
modelling style in which a relationship is expressed as an object class

EXAMPLE The relation Employed-by is reified by the object Employment which is connected to the objects Employee and Organization. The meaning of the relation with cardinalities at both ends is "an organization has zero or more employees". The reified Employment object can be subject in other relations, defining it.

NOTE The relational entity types of ISO 15926 are all the entity types which have exactly two attributes, except **class_of_relationship**.

2.1.22
template
set comprising of a first-order logic predicate for which a definition is stated as an axiom, a template signature and a template axiom expansion

2.1.23
template expansion
template axiom expansion
statement expressed in ISO 15926-2 entity data types, equivalent to a template axiom

NOTE The expansion of a template axiom refers to a typically complex, biconditional in the ISO 15926-2 language. It is obtained by repeated application of template biconditionals until the template's interpretation is expressed directly in terms of simple ISO 15926-2 language constructions.

2.1.24
template instance expansion
set of simple statements in the ISO 15926-2 language, obtained by instantiating variables in the expanded template axiom with entity instances

2.1.25
template axiom
axiom in the template language defining the interpretation of template statements

2.1.26
template instance
ordered list of entity instances of which a template is true

2.1.27
template language
axioms in first-order logic extending the ISO 15926-2 data model

2.1.28
template role
named and numbered argument in a template with required type given as entity type, data type, or reference data class

EXAMPLE Template InstanceOfIndirectProperty(a, b, c) means that a is a ClassOfIndirectProperty, b a (temporal part of) PossibleIndividual to which the relation applies and c the instance of Property. b has an a type of ClassOfIndirectProperty, which has a c instance of Property. It has the following roles:

a) role name: Property type;

b) role name: Property possessor;

c) role name: Property.

2.1.29
template signature
named, ordered and typed list of template roles

EXAMPLE Template InstanceOfIndirectProperty(a, b, c) means that a is a ClassOfIndirectProperty, b a (temporal part of) PossibleIndividual to which the relation applies and c the instance of Property. b has an a type of ClassOfIndirectProperty, which has a c instance of Property. The template signature is:

a) role name: Property type, role type: **ClassOfIndirectProperty**;

b) role name: Property possessor, role type: **PossibleIndividual**;

c) role name: Property, role type: **Property**.

2.1.30
template statement
statement made by instantiating the roles of a template with entity instances

2.1.31
value
unit of data

[ISO 10303-11:2004, definition 3.3.22]

2.2 Abbreviated terms

FOL	first order logic
DL	description logic
RDL	reference data library

3 Fundamental concepts and assumptions

3.1 General

The ISO 15926-2 data model is generic and highly normalized. Whilst this enables considerable flexibility in what can be said, it can also give rise to complexity in how it is said. This part of ISO 15926 specifies templates that are expressions of predefined units of semantics allowing the use of the model in a convenient way. The approach taken in this part of ISO 15926 is based on the architecture described in ISO/TS 18876-1 and is illustrated in Figure 1.

3.2 Concepts and models

The concepts and models shown in Figure 1 are implemented in this part of ISO 15926.

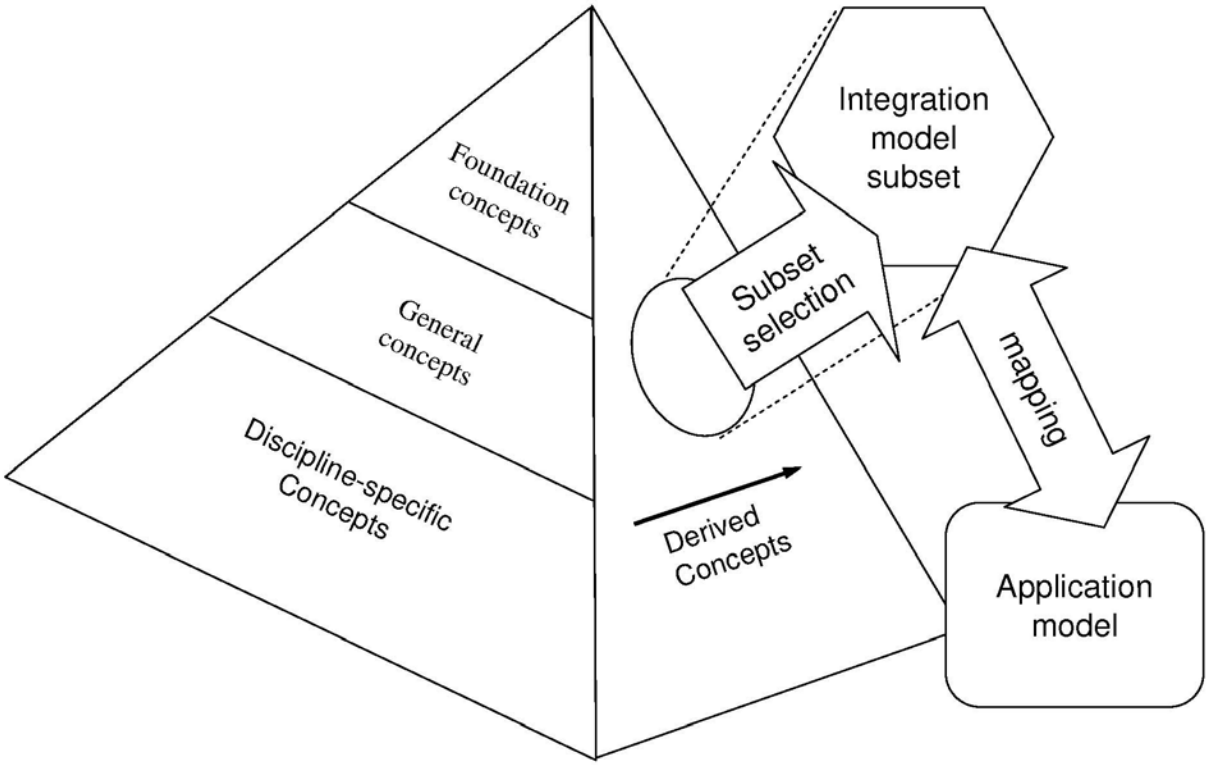

Figure 1 — An overview of the integration architecture of ISO/TS 18876-1

3.2.1 ISO 15926-2 data model

The foundation concepts are represented in ISO 15926-2 by a generic, conceptual data model that is suitable as the basis for implementation in a shared database or data warehouse. The data model is designed to be used in conjunction with reference data. The support for a specific life-cycle activity depends on the use of appropriate reference data in conjunction with the data model (see 4.1).

The ISO 15926-2 EXPRESS model is translated in this part of ISO 15926 into first order logic (FOL), with each entity type translated into an unary predicate, and each attribute translated into a binary predicate (see 4.1).

3.2.2 ISO/TS 15926-4 reference data

ISO/TS 15926-4 provides reference data that defines a taxonomy of core classes representing the entity data types defined in ISO 15926-2.

For this part of ISO 15926, reference data is treated as constant terms in the sense of first order logic. The notion of reference data instantiating the entity types of the data model is thereby reduced to truth of first order representations of ISO 15926-2 entity types about constant reference terms, or, for ISO 15926-2 attributes, ordered pairs of terms.

EXAMPLE 1 A template makes a relation between an instance of PUMP, an identifying string "PU101" and an identification type TAG NUMBER.

$$ClassifiedIdentification(myPump, \text{“}PU101'', TagNumber)$$

These three terms are reference data, and the template is defined in first order logic.
When an implementation verifies the template, the entity types of the reference data are checked by lifting the template, and these reference data are treated as constants while testing this verification to be true or false.

3.2.3 User-defined taxonomy

Where the ISO/TS 15926-4 taxonomy defines the general concepts in Figure 1, a user organization often needs to define discipline-specific concepts. These are defined in the form of specializations of the general concepts in the ISO/TS 15926-4 taxonomy.

EXAMPLE CP-834833 ('Model AK/150 pump') is a specialized class in the supplier catalog of the XYZ Corporation. This supplier-owned class resides in the RDL of the supplier, but the class must be a specialization of another class, following the taxonomy up the tree until inside the ISO/TS 15926-4 taxonomy.

3.2.4 Templates

Templates (see Clauses 4 and 5), defined in this part of ISO 15926 and in user-defined specializations form the "derived concepts" in Figure 1.

4 Modelling basics

4.1 ISO 15926-2 data model in first-order logic

This clause defines the ISO 15926-2 language, a formulation of ISO 15926-2, as standardized in EXPRESS format, in first order logic. This language serves for this part of ISO 15926 as the expression of the ISO 15926 data model.

In ISO 15926-2 EXPRESS, names of entity types and attributes are written in lower case with the _ character dividing the words in names. For the ISO 15926-2 language, entity type and attribute names shall be written in upper camel case, with attributes given an additional prefix "**has**". For example, the entity type `class_of_class_of_relationship` is named **ClassOfClassOfRelationship**, and the attribute `shape_dimension` is named **hasShapeDimension** in this part of ISO 15926.

In the ISO 15926-2 language, entity types are represented as unary first-order predicates. Let `a` be an EXPRESS entity type. This shall be represented in the ISO 15926-2 language as

$$\mathbf{A}(x)$$

where $\mathbf{A}(c)$ is true iff (if and only if) c is an instance of `a`.

Attributes shall be represented in the ISO 15926-2 language as binary predicates: with `r` an EXPRESS attribute, this becomes

$$\mathbf{hasR}(x, y)$$

with the interpretation that $\mathbf{hasR}(c, d)$ is true iff the value of the attribute of `r` of the entity instance c is d.

NOTE 1 The reason for adding the prefix **has** to attributes is that some names in the EXPRESS implementation of ISO 15926-2 are used for both entity types and attributes. The prefix is needed because entities and attributes occupy different namespaces in EXPRESS, but the same namespace in the ISO 15926-2 language. Use of this convention allowes a consistent mapping to be used while avoiding collisions.

In EXPRESS models the native datatypes are usually written in upper case, e.g., `INTEGER`. In the ISO 15926-2 language these datatypes shall also be written in upper case, e.g., **INTEGER**. All EXPRESS native datatypes except `LIST` shall be represented as unary predicates. If `A` is an EXPRESS datatype, write

$$\mathbf{A}(x)$$

with the interpretation that $A(c)$ is true iff c is of datatype `A`.

The use of `LIST` in ISO 15926-2 is limited to the definition of the entity types `multidimensional_object` and `class_of_multidimensional_object`. In this part of ISO 15926, these types are considered part of the definition of the template language. Therefore `LIST` is not itself a predicate of the ISO 15926-2 language.

Subtyping among entity types in EXPRESS shall be represented in the ISO 15926-2 language by conditionals. If `a` is a subtype of `b`, the ISO 15926-2 language states that all `a`s are `b`s:

$$\mathbf{A}(x) \rightarrow \mathbf{B}(x)$$

The `ABSTRACT` entity types of ISO 15926-2 are represented in the ISO 15926-2 language by stating that every instance of an abstract entity type also instantiates at least one of the immediate subtypes of the abstract entity type. If the entity type `a` is `ABSTRACT` and `b`, `c` and `d` are all the immediate subtypes of `a`, then the fact that `a` is abstract is represented in FOL as follows.

$$\begin{aligned}\mathbf{A}(x) \rightarrow (&\mathbf{B}(x) \vee \mathbf{C}(x) \vee \mathbf{D}(x))\\ \wedge(&\mathbf{B}(x) \rightarrow \mathbf{A}(x))\\ \wedge(&\mathbf{C}(x) \rightarrow \mathbf{A}(x))\\ \wedge(&\mathbf{D}(x) \rightarrow \mathbf{A}(x))\end{aligned}$$

ISO 10303-11 describes `ONEOF` statements as follows.

> The ONEOF constraint states that the populations of the operands in the ONEOF list are mutually exclusive; no instance of any of the populations of the operands in the ONEOF list shall appear in the population of any other operand in the ONEOF list. The ONEOF constraint may be combined with the other supertype constraints to enable the writing of complex constraints. When an ONEOF constraint occurs as an operand in another constraint, it represents the set of entity instances that is the union of the populations of the operands in the ONEOF list.
>
> [ISO 10303-11:2004, 9.2.5.2]

`ONEOF` statements are represented as disjointness axioms in the ISO 15926-2 language. The EXPRESS

statement `ONEOF(a, b, c)` is represented with the following formula.

$$\neg(\mathbf{A}(x) \wedge (\mathbf{B}(x))$$
$$\wedge\neg(\mathbf{A}(x) \wedge (\mathbf{C}(x))$$
$$\wedge\neg(\mathbf{B}(x) \wedge (\mathbf{C}(x))$$

EXPRESS attributes are represented as binary predicates in FOL. The set of allowable values in the first position of a binary predicate is called the domain and the set of allowable values in the second position the range.

NOTE 2 EXPRESS attributes are often called "roles".

The set of ISO 15926-2 entity types to which an attribute applies is represented in the ISO 15926-2 language by a restriction on the domain of the binary predicate representing the attribute. If the attribute `r` is defined for the entity types `a` and `b`, write

$$\mathbf{hasR}(x, y) \rightarrow (\mathbf{A}(x) \vee \mathbf{B}(x)) .$$

The set of allowable values of an attribute is represented in the ISO 15926-2 language by restricting the range of the binary predicate representing the attribute. In EXPRESS, attribute values are always restricted with respect to a given entity type. These restrictions are represented by local range restrictions in the ISO 15926-2 language, meaning that a range restriction always includes a domain restriction of the binary predicate.

Suppose `r` is an attribute, for which values are limited to entity type `f` for the entity type `a`, and to `g` for entity type `b`. In the ISO 15926-2 language this shall be written as follows:

$$\mathbf{A}(x) \wedge \mathbf{hasR}(x, y) \rightarrow \mathbf{F}(y)$$
$$\mathbf{B}(x) \wedge \mathbf{hasR}(x, y) \rightarrow \mathbf{G}(y)$$

Every attribute in the EXPRESS implementation of ISO 15926-2 has a cardinality constraint $[0, 1]$ or $[1, 1]$. Cardinality constraints for attributes are given per entity type, as for range restrictions. Cardinality constraints on binary predicates are represented using two axioms. Suppose that the attribute `r` has cardinality constraint $[1, 1]$ for the entity type `a`. In the ISO 15926-2 language this shall be written as a pair of axioms:

$$\mathbf{A}(x) \rightarrow \exists y(\mathbf{hasR}(x, y)) \quad (1)$$
$$\mathbf{A}(x) \wedge \mathbf{hasR}(x, y) \wedge \mathbf{hasR}(x, z) \rightarrow y = z \quad (2)$$

Formula (1) represents the cardinality constraint $[1, *)$. The second axiom represents the cardinality constraint $[0, 1]$. Together they express the cardinality constraint $[1, 1]$. A cardinality constraint $[0, 1]$ is represented by a formula of the form (2) only.

Where an attribute is restricted by an EXPRESS `UNIQUE` rule, this shall be represented in the ISO 15926-2 language by requiring that the predicate is inverse functional. If attribute `r` is `UNIQUE` for the entity type `A` write:

$$\mathbf{A}(x) \wedge \mathbf{A}(y) \wedge \mathbf{hasR}(x, z) \wedge \mathbf{hasR}(y, z) \rightarrow x = y .$$

The set of axioms of the ISO 15926-2 language is listed in Annex B.

NOTE 3 The set of axioms has been proven to be logically consistent by the use of an FOL model checker.

4.2 Logical template definition

This clause defines logical templates. Templates as defined in Clause 5 additionally have signatures, which will not be discussed here.

Given a first-order formula Σ_0 which uses the predicate names of the ISO 15926-2 language axiomatisation for a basic language, a logical template definition over Σ is a formula of first order logic of the form:

$$N(x, y, \ldots) \leftrightarrow \phi$$

where $N \notin \Sigma$ is a predicate symbol called the name of the template, $x, y, \ldots$ are pairwise different variables called the formal arguments, and ϕ is a $\wedge$-$\vee$-$\exists$-formula over symbols in Σ and containing only the formal arguments $x, y, \ldots$ as free variables, and which is called the body of the template.

The Set $TS(\Sigma_0)$ of logical template sets over Σ_0 is inductively defined as the least set such that

— $\emptyset \in TS(\Sigma_0)$ is a logical template set over Σ_0.

— If $S \in TS(\Sigma_0)$ is a logical template set over Σ_0, and d is a logical template definition over $\Sigma_0 \cup \text{names}(S)$, then $S \cup \{d\} \in TS(\Sigma_0)$ is also a logical template set over Σ_0

where $\text{names}(S) := \{N \mid N(\ldots) \leftrightarrow \phi \in S\}$ is the set of names of templates defined in S.

NOTE The intuition behind template sets is as follows: To ensure that template expansion terminates, we want to forbid sets of cyclic template definitions, e.g. where a template A gets expanded to a formula containing a template B, which in turn gets expanded to a formula containing A. Our definition ensures that a template definition $N(\ldots) \leftrightarrow \phi$ can only be added to a template set if all templates named in ϕ have previously been defined without reference to N.

EXAMPLE The set of definitions

$$\begin{array}{lll} \{ \; A(x) & \leftrightarrow & \exists y.(\, B(y) \wedge \mathbf{R}(x, y)\,), \\ \; B(x) & \leftrightarrow & \mathbf{C}(x) \vee \mathbf{D}(x) \qquad \} \end{array}$$

is a valid template set, but

$$\begin{array}{lll} \{ \; A(x) & \leftrightarrow & \exists y.(\, B(y) \wedge \mathbf{R}(x, y)\,), \\ \; B(x) & \leftrightarrow & \mathbf{C}(x) \vee A(x) \qquad \} \end{array}$$

is not.

4.3 Proto-templates

Proto-templates are a very basic form of templates. They provide a layer of abstraction immediately above the relational entity types of ISO 15926-2 by hiding the reified (see 2.1.21) relationships of ISO 15926-2.

Each proto-template is constructed of two FOL axioms. The first axiom provides a short form for the relational entity type. The proto-template predicate is called a *proto-triple*.

Suppose that `R` is a relational entity type in ISO 15926-2 and that `R` has two roles `r1` and `r2`.

The proto-triple for `R`, *RTriple*, is defined as

$$\mathit{RTriple}(z, x, y) \leftrightarrow \mathbf{R}(z) \wedge \mathbf{hasR1}(z, x) \wedge \mathbf{hasR2}(z, y)$$

If the relational entity `T` inherits its roles from a supertype `R`, the proto-triple *TTriple* is defined as

$$\mathit{TTriple}(z, x, y) \leftrightarrow \mathbf{T}(z) \wedge \mathit{RTriple}(z, x, y)$$

A (binary) proto-template sentence expresses a relationship between the two role fillers of the corresponding relational entity type. The proto-template for R, *RTemplate*, is defined as

$$RTemplate(x, y) \leftrightarrow \exists z(RTriple(z, x, y)) .$$

The axioms defining the proto-templates for the ISO 15926-2 relational entity types are given in Annex C. A concise listing of all proto-templates is given in Annex D.

4.3.1 entityTriple

The predicate *entityTriple* is defined in clause C.3, as the disjunction of all proto-templates given in clauses C.1 and C.2. This triple allows for a concise expression of statements that apply to all relational entity types.

4.4 Diagrams

This clause specifies how diagrams should be interpreted.

A class is displayed as a box with two compartments. The upper compartment shows the designation, the lower compartment the ISO 15926-2 entity type of the class. A class with designation **A** and entity type **Class** is shown in Figure 2.

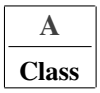

Figure 2 — Diagram of a class

A relation is displayed using a double diamond, with a connector to a box showing the entity type, and optionally the designation of the relation. A relation with designation **R** and entity type **ClassOfConnectionOfIndividual** is shown in Figure 3, with and without designation.

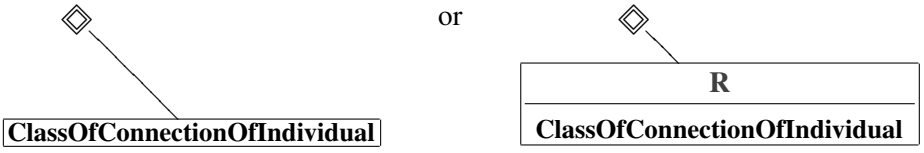

Figure 3 — Diagram of a relation

A relationship (an instance of a relation) is displayed similarly to a relation, but with a single diamond. Relationships are in general not given designations. A relationship with entity type **ConnectionOfIndividual** is shown in Figure 4.

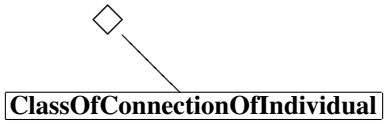

Figure 4 — Diagram of a relationship

The roles of a relation or relationship are shown as connectors from the relation (relationship), annotated with ISO 15926-2 role names. A **ClassOfConnectionOfIndividual** relation **R** between classes **A** and **B**, with **A** in the **hasClassOfSide1** and **B** in the **hasClassOfSide2** roles, is shown in Figure 5.

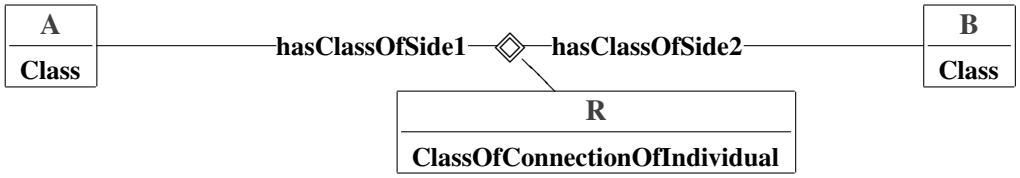

Figure 5 — Diagram showing relation roles

Cardinality that applies to a relation is attached to the role with connectors indicating which role it apply to as **end_1_cardinality** or **end_2_cardinality**. See Figure 6.

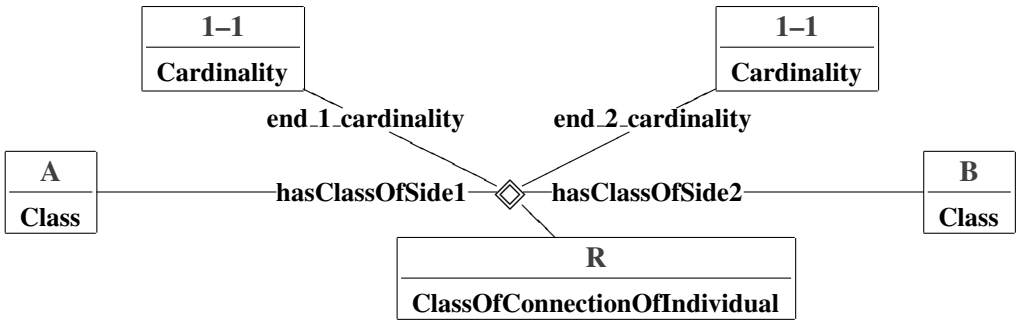

Figure 6 — Diagram showing cardinalities

With the frequently used relation types Classification (membership) and Specialization (subclassing), the relationship symbol and role indicators can be eliminated in favour of directed arrows. The relations are represented by pointed and bullet-shaped arrowheads, respectively. Figure 7 illustrates specialization between classes **A** and **B**.

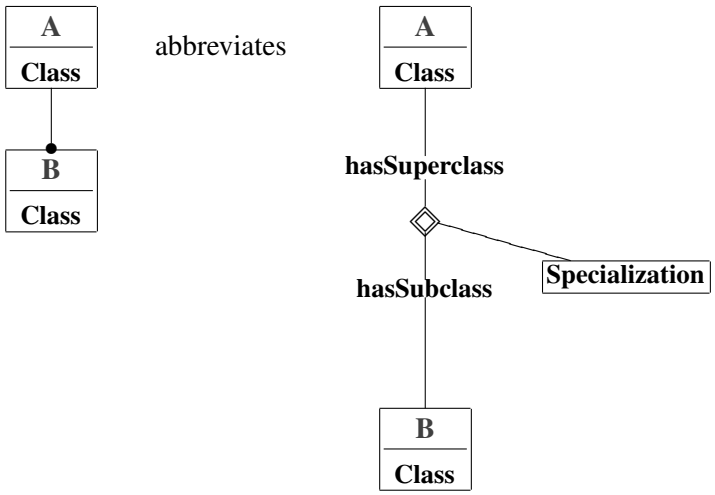

Figure 7 — Diagram of specialization

Figure 8 illustrates classification of an individual **a** as member of class **A**.

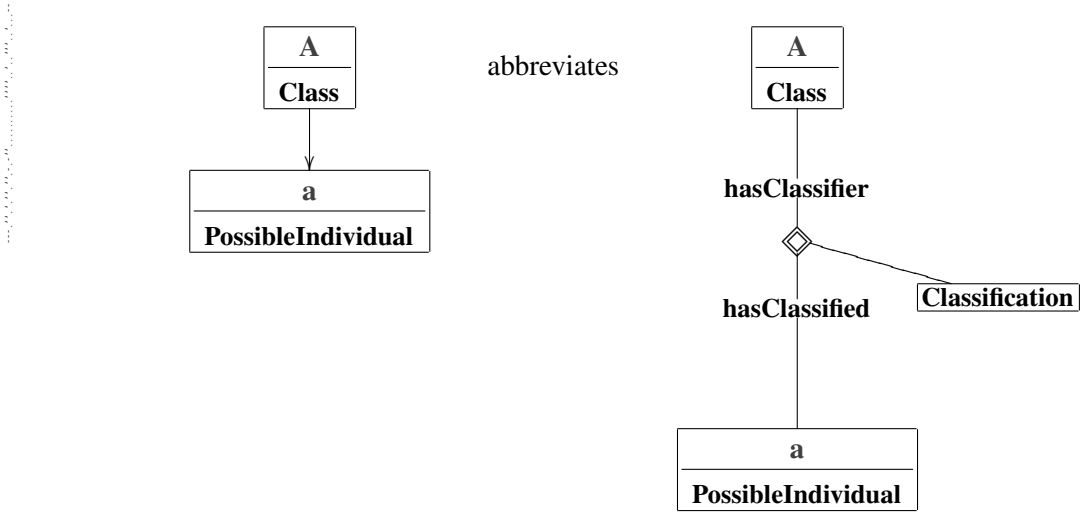

Figure 8 — Diagram of classification

5 Template specification

5.1 Requirements on template, general

To introduce a template, the following shall be provided.

— a name;

— an informal textual explanation of intended use and meaning;

— a signature listing roles and types of each argument (see 5.2);

— a formal definition in the form of a biconditional in the template language.

The formal definition of the template shall expand to an ISO 15926-2 language pattern (see 4.1) that is appropriate for the expressive purpose. The definition should be tested for consistency with the data model (see 5.4).

5.2 Template signatures

A template signature shall specify a constraint, for each role in a template, on the permitted type of the individuals filling the role. Each role constraint shall given by a unary predicate.

NOTE 1 The signature can be seen as a representation of the full template definition for which relations between the role-fillers are not included.

Template signatures are significant for the use of templates to represent and store data. The choice of constraints, from the template's definition, to include in the signature is only determined by considerations of practical usability.

NOTE 2 See Clause 8 for further information on the representation of templates as reference data.

A template signature shall be specified as an ordered list of template roles. For each role the following shall be given.

a) a role name. Role names are required to be unique within a signature;

b) a type constraint for individuals filling the role, given as an

   1) entity type,

   2) RDL class, or

   3) data type,

   meaning, respectively, that any individual that fills the role must have the specified entity type; is classified by the specified RDL class; is an instance of the specified data type.

NOTE Each role in a signature gives a constraint on the type of individuals instantiating the role. When used in reference data implementations, unary templates are in general not suitable for specifying role types, even though they may be stored in an RDL as classes. This is because templates range over ordered lists of individuals rather than individuals. A role whose type is given by a unary template may therefore only be instantiated by singleton lists.

5.3 Template specialization

A special case of template definition is the introduction of a template as a specialized version of another template. Let T name a ternary template, with roles constrained by R_1, R_2, and R_3 and further "modelling" requirements on the relations between arguments expressed in a formula φ:

$$T(x, y, z) \leftrightarrow R_1(x) \wedge R_2(y) \wedge R_3(z) \wedge \varphi\,.$$

Suppose that T_1 is a specialization of T, distinguished by having R_4 constraining the first role instead of R_1. The following formula suffices to capture this in a formal definition of T'.

$$\begin{aligned} &T' \rightarrow T\,,\\ &R_4(x) \rightarrow R_1(x)\,,\\ &T'(x, y, z) \leftrightarrow T(x, y, z) \wedge R_4(x)\,. \end{aligned}$$

It follows that T' is a specialized template of T: every instance of T' is an instance of T, and relations between individuals that participate in an instance will be just as implemented for T.

NOTE This covers only the logical definition of a specialized template. For a template compliant with this part of ISO 15926, the requirements to give an explanation and signature (see 5.1) still apply.

5.4 Verification of compliance with ISO 15926-2

Each template definition should be tested for consistency with the constraints laid down by the ISO 15926-2 data model. It should be verified that each template can be instantiated in a way that is consistent with the ISO 15926-2 language.

NOTE 1 The following procedure can be used to test whether a template T is consistent with ISO 15926-2.

Instantiate T with pairwise distinct arbitrary individuals, then expand the statement according to its template axiom. The expansion typically requires expanding other templates as well: any templates that occur in the definition of T, and in definitions of further templates to which these in turn expand. Expand the template instance, and consider whether the expansion contains predicates that are not in the ISO 15926-2 language (i.e., it contains template language predicates)

— if it does, the template definition appeals to predicates for which formal ISO 15926-2 definition is missing, and the definition of T is incomplete.

— if not, the expansion of the template instance is an expression in the ISO 15926-2 language. This expression should be tested for consistency with the ISO 15926-2 model using generic first-order proof methods.

A successful test of consistency proves that the template as defined has an interpretation in terms of the ISO 15926 data model, and thus meets a formal crierion of adequacy.

NOTE 2 Tests of suitability for a purpose are outside the scope of this part of ISO 15926.

6 Templates for individuals

6.1 Purpose

Templates that serve to characterise individuals, as opposed to classes, are included under the heading “templates for individuals”. The typical use for these templates is to record information about a single, physical object.

NOTE 1 ISO 15926 does not require a sharp stratification of the universe into individuals, classes, metaclasses and so forth. What constitutes an individual is accordingly not strictly defined.

6.2 Reference items needed

The following reference data item is used in template axioms in this section. It extends the ISO 15926-2 language with an individual term. For practical applications of templates whose definitions refer to this item, the item should be recorded in a reference data library.

Reference individual	Entity type
ActivityLocation	**ClassOfRelationshipWithSignature**
InvolvementSuccession	**ClassOfRelationshipWithSignature**

Table 1 — Reference items: Templates for individuals

ActivityLocation and **InvolvementSuccession** are relations. RDL records for these should assign classes to roles as required by the entity type: **hasClassOfEnd1** and **hasClassOfEnd2**, representing *domain* and *range* of the relations, respectively.

For **ActivityLocation**, domain and range should be given by reference items representing the entity types **Activity** and **SpatialLocation**, respectively, in accordance with the intended use in assigning locations to activities.

For **InvolvementSuccession**, domain and range should be given by a reference item representing the entity type **InvolvementByReference**.

6.3 Initial set

6.3.1 Template ClassificationOfIndividual

This is a template for classification of individuals (as opposed to pairs of individuals, classes, or relations).

ClassificationOfIndividual(a, b) means that a is an individual, b is a class of individuals, and a is a member of b.

To quote from ISO 15926-2, classification:

> A ¡classification¿ is type of ¡relationship¿ that indicates that the classified ¡thing¿ is a member of the classifier ¡class¿.

Order	Role name	Role type
1	Individual	**PossibleIndividual**
2	Class	**ClassOfIndividual**

ClassificationOfIndividual$(x_1, x_2) \leftrightarrow$

PossibleIndividual$(x_1) \wedge$

ClassOfIndividual$(x_2) \wedge$

ClassificationTemplate(x_1, x_2)

EXAMPLE The classification of **Alfred** as an **Engineer** can be expressed using this template. The expansion of the statement *ClassificationOfIndividual*(**Alfred**, **Person**) corresponds to statements in the ISO 15926-2 language as shown in the following diagram.

Note that the reference individuals **Alfred** and **Engineer** are not themselves defined by the template statement; each may have a more specialized entity type than shown in the diagram.

6.3.2 Template ClassificationOfRelationship

This is a template for giving type to relationships. It is a classification template, restricted to classifying pairs of things as members of relations.

ClassificationOfRelationship(a, b) means that a is an ordered pair, b is a relation, and a is a member of b.

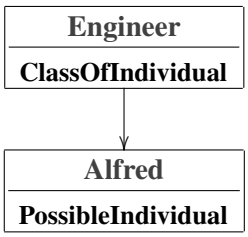

Figure 9 — Example template ClassificationOfIndividual

Order	Role name	Role type
1	Pair	**Relationship**
2	Relation	**ClassOfRelationship**

ClassificationOfRelationship$(x_1, x_2) \leftrightarrow$

Relationship$(x_1) \wedge$

ClassOfRelationship$(x_2) \wedge$

ClassificationTemplate(x_1, x_2)

NOTE See also *InstanceOfRelation*.

EXAMPLE *ClassificationOfRelationship*(⟨**Alfred**, **ACME Co.**⟩, **Employment**) expands to a representation structured as in the following diagram. Note that

— the definition of the first argument (the specification of the entity type and members for the ordered pair) is not given as part of the template statement;

— the entity type **Relationship** is abstract. In a valid instantiation of this template, the ordered pair will have an entity type that is a subtype of **Relationship**;

— the use of angular brackets, '⟨. . . , . . .⟩', to name an ordered pair is not normative in this part of ISO 15926.

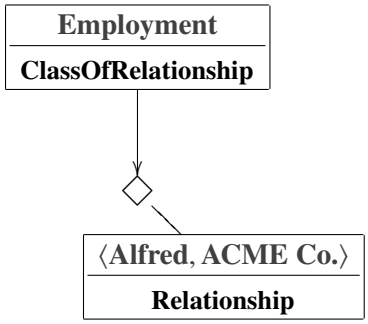

Figure 10 — Example template ClassificationOfRelationship

6.3.3 Template InstanceOfRelation

This is a template for expressing relationships that do not have a predefined ISO 15926-2 type.

InstanceOfRelation(a, b, c) means that a is a custom-defined relation by which b is related to c.

Order	Role name	Role type
1	Relation	**ClassOfRelationshipWithSignature**
2	First element	**Thing**
3	Second element	**Thing**

$$\begin{aligned}&\mathit{InstanceOfRelation}(x_1, x_2, x_3) \leftrightarrow \\ &\quad \mathbf{ClassOfRelationshipWithSignature}(x_1) \wedge \\ &\quad \mathbf{Thing}(x_2) \wedge \\ &\quad \mathbf{Thing}(x_3) \wedge \\ &\quad \exists u(\mathit{OtherRelationshipTriple}(u, x_2, x_3) \wedge \\ &\quad \mathit{ClassificationOfRelationship}(u, x_1))\end{aligned}$$

NOTE This template uses the template *ClassificationOfRelationship*. Where the latter template takes an ordered pair (a relationship) as a single argument to be classified by a relation, this template takes the elements of the ordered pair themselves as arguments. The composition of these elements into an ordered pair is given by the template axiom.

EXAMPLE Let **Alfred** and **ACME Co.** be instances of **Person**, and **Employment** a custom-defined relation (i.e., a **ClassOfRelationshipWithSignature**). *InstanceOfRelation*(**Employment**, **Alfred**, **ACME Co.**) then expands to the following representation, which may be compared to the one shown in the example for *ClassificationOfRelationship*. Note that

— no designation for the classified ordered pair is defined;

— the definition of the **ClassOfRelationshipWithSignature**, which should include specification of roles for the ends of the relation, is not given in the template statement.

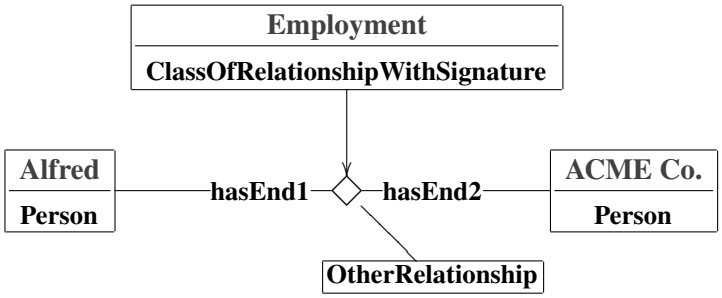

Figure 11 — Example template InstanceOfRelation

6.3.4 Template IdentificationByNumber

This is a template for naming things by real numbers.

IdentificationByNumber(a, b) means that a is *a* real number, and a refers to b.

Order	Role name	Role type
1	Identifier	**ExpressReal**
2	Identified	**Thing**

IdentificationByNumber$(x_1, x_2) \leftrightarrow$

ExpressReal$(x_1) \wedge$

Thing$(x_2) \wedge$

ClassOfIdentificationTemplate(x_1, x_2)

NOTE This template is a specialized version of the *ClassOfIdentificationTemplate*, constraining the type of the first argument from **ClassOfInformationRepresentation** to its subtype **ExpressReal**.

EXAMPLE The statement that the number π is identified by the decimal number 3.14 may be expressed as *IdentificationByNumber*$(3.14, \pi)$.

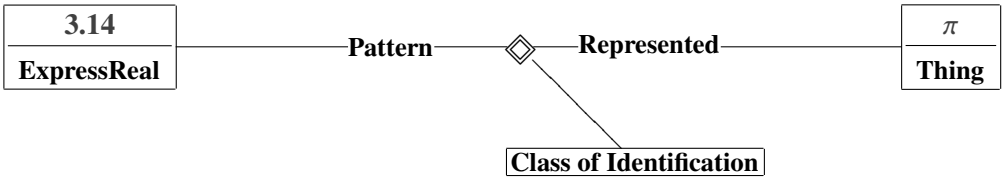

Figure 12 — Example template IdentificationByNumber

6.3.5 Template ClassifiedIdentification

This is a template for typed naming of things.

ClassifiedIdentification(a, b, c) means that b is a string and c a type of name assignment, and that b is a c-type name for a.

Order	Role name	Role type
1	Object	**Thing**
2	Identifier	**ExpressString**
3	Context	**ClassOfClassOfIdentification**

ClassifiedIdentification$(x_1, x_2, x_3) \leftrightarrow$

Thing$(x_1) \wedge$

ExpressString$(x_2) \wedge$

ClassOfClassOfIdentification$(x_3) \wedge$

$\exists u($*ClassOfIdentificationTriple*$(u, x_2, x_1) \wedge$

ClassificationTemplate$(u, x3))$

EXAMPLE The statement *ClassifiedIdentification*(**Alfred**, **PN4723**, **Employee No. ACME Co.**) (e.g., an assignment of employee number) expands as follows.

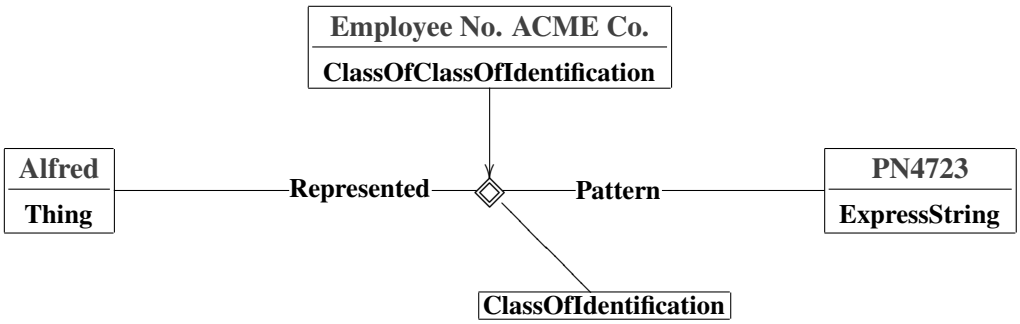

Figure 13 — Example template ClassifiedIdentification

NOTE This template assigns names to things, and a type (a classifier) to the assignments themselves. An intended use is for the classifier to represent a context in which the name assignment is valid.

6.3.6 Template LocationOfActivity

This is a template for stating where activities take place.

LocationOfActivity(a, b) means that a is an activity, b is a location, and that a takes place at b.

Order	Role name	Role type
1	Activity	**Activity**
2	Location	**SpatialLocation**

LocationOfActivity$(x_1, x_2) \leftrightarrow$

Activity$(x_1) \wedge$

SpatialLocation$(x_2) \wedge$

InstanceOfRelation(**ActivityLocation**, x_1, x_2)

EXAMPLE The statement *LocationOfActivity*(**Site Survey #23**, **Site No. 11**) expands to the following representation.

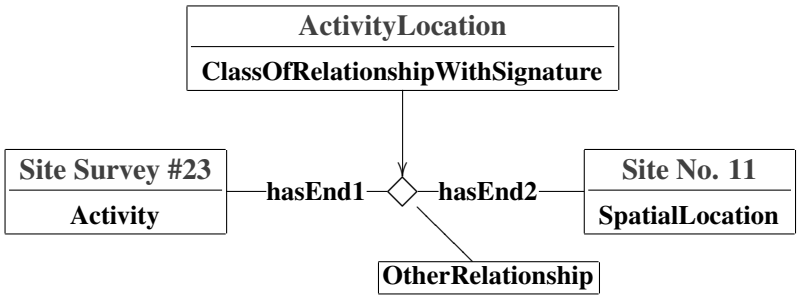

Figure 14 — Example template LocationOfActivity

NOTE *LocationOfActivity* is an *RDL template* because the reference individual **ActivityLocation** (a **ClassOfRelationshipWithSignature**) appears in its template axiom.

6.3.7 Template BeginningOfIndividual

This is a template for stating start time of existence.

BeginningOfIndividual(a, b) means that a is an individual and b is a point in time, and that a begins to exist at b.

Order	Role name	Role type
1	Individual	**PossibleIndividual**
2	Start time	**RepresentationOfGregorianDateAndUtcTime**

BeginningOfIndividual$(x_1, x_2) \leftrightarrow$

PossibleIndividual$(x_1) \wedge$

RepresentationOfGregorianDateAndUtcTime$(x_2) \wedge$

$\exists u($**PointInTime**$(u) \wedge$ *BeginningTemplate*$(u, x_1) \wedge$

ClassOfRepresentationOfThingTemplate(x_2, u)

$)$

6.3.8 Template BeginningEndOfIndividual

This is a template for stating start and end times of existence.

BeginningEndOfIndividual(a, b, c) means that a is an individual and b and c are points in time, and that a begins to exist at b and ceases to exist at c.

Order	Role name	Role type
1	Individual	**PossibleIndividual**
2	Start time	**RepresentationOfGregorianDateAndUtcTime**
3	End time	**RepresentationOfGregorianDateAndUtcTime**

BeginningEndOfIndividual$(x_1, x_2, x_3) \leftrightarrow$

PossibleIndividual$(x_1) \wedge$

RepresentationOfGregorianDateAndUtcTime$(x_2) \wedge$

RepresentationOfGregorianDateAndUtcTime$(x_3) \wedge$

$\exists u($**PointInTime**$(u) \wedge$ *BeginningTemplate*$(u, x_1) \wedge$

ClassOfRepresentationOfThingTemplate$(x_2, u)) \wedge$

$\exists u($**PointInTime**$(u) \wedge$ *EndTemplate*$(u, x_1) \wedge$

ClassOfRepresentationOfThingTemplate$(x_3, u))$

EXAMPLE The statement *BeginningEndOfIndividual*(**Survey # 23**, **2009–10–19**, **2009–10–21**) expands to the following representation. Note that no designation for the **PointInTime** instances is defined.

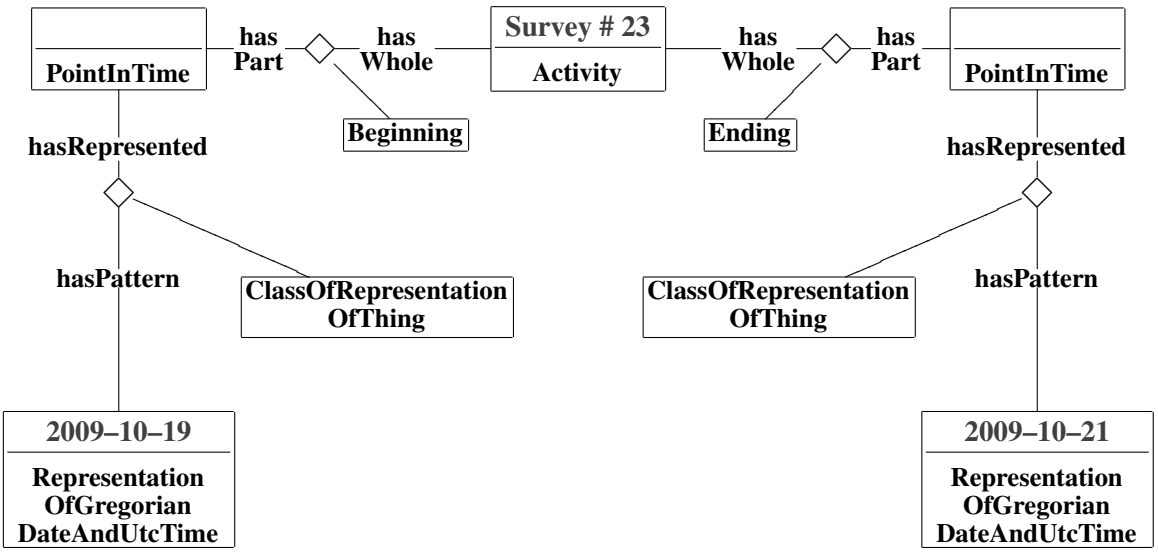

Figure 15 — Example template BeginningEndOfIndividual

6.3.9 Template BeginningOfTemporalPart

This template is for stating that an individual is a temporal part of another, initiated at a time represented by a timestamp data value.

BeginningOfTemporalPart(a, b, c) means that a is an individual and b is an individual and c is a point in time, that a is a temporal part of b, and that a begins to exist at c.

Order	Role name	Role type
1	Part	**PossibleIndividual**
2	Whole	**PossibleIndividual**
3	Start time	**RepresentationOfGregorianDateAndUtcTime**

BeginningOfTemporalPart$(x_1, x_2, x_3) \leftrightarrow$

PossibleIndividual$(x_1) \wedge$

PossibleIndividual$(x_2) \wedge$

RepresentationOfGregorianDateAndUtcTime$(x_3) \wedge$

TemporalWholePartTemplate$(x_1, x_2) \wedge$

BeginningOfIndividual(x_1, x_3)

6.3.10 Template BeginningEndLocationOfActivity

This is a template for expressing where and when an activity takes place.

BeginningEndLocationOfActivity(a, b, c, d) means that a is an activity, b and c are points in time, and d is a location, and that a takes place at d, starting at b and ending at c.

Order	Role name	Role type
1	Activity	**Activity**
2	Start time	**RepresentationOfGregorianDateAndUtcTime**
3	End time	**RepresentationOfGregorianDateAndUtcTime**
4	Location	**SpatialLocation**

BeginningEndLocationOfActivity$(x_1, x_2, x_3, x_4) \leftrightarrow$

Activity$(x_1) \wedge$

RepresentationOfGregorianDateAndUtcTime$(x_2) \wedge$

RepresentationOfGregorianDateAndUtcTime$(x_3) \wedge$

SpatialLocation$(x_4) \wedge$

BeginningEndOfIndividual$(x_1, x_2, x_3) \wedge$

LocationOfActivity(x_1, x_4)

6.3.11 Template InstanceOfIndirectProperty

This template is for expressing the classified possession by an individual of an indirect property.

InstanceOfIndirectProperty(a, b, c) means that a is is a ClassOfIndirectProperty, b a (temporal part of) PossibleIndividual to which the relation applies and c is the instance of Property. b has a a type of ClassOfIndirectProperty, which has c instance of Property.

Order	Role name	Role type
1	Property type	**ClassOfIndirectProperty**
2	Property possessor	**PossibleIndividual**
3	Property	**Property**

InstanceOfIndirectProperty$(x_1, x_2, x_3) \leftrightarrow$

ClassOfIndirectProperty$(x_1) \wedge$

PossibleIndividual$(x_2) \wedge$

Property$(x_3) \wedge$

$\exists u($*ClassificationOfRelationship*$(u, x_1) \wedge$

IndirectPropertyTriple(u, x_2, x_3)

)

6.3.12 Template RealMagnitudeOfProperty

This template provides a version of **MagnitudeOfProperty** for which the magnitude is given by a datatyped representation rather than a number object.

RealMagnitudeOfProperty(a, b, c) means that a is is an instance of Property, b is a floating point number with the property value and d the Scale as unit of measurement.

Order	Role name	Role type
1	Property	**Property**
2	Property value	**ExpressReal**
3	Property scale	**Scale**

Template RealMagnitudeOfProperty$(x_1, x_2, x_3) \leftrightarrow$

Property$(x_1) \wedge$

ExpressReal$(x_2) \wedge$

Scale$(x_3) \wedge$

$\exists u($*MagnitudeOfProperty*$(x_1, u, x_3) \wedge$

IdentificationByNumber(x_2, u)

)

6.3.13 Template IndirectPropertyScaleReal

This template is for assigning a typed indirect property to an individual, with magnitude given as a real number and a scale.

IndirectPropertyScaleReal(a, b, c, d) means that a is is a ClassOfIndirectProperty, b a (temporal part of) PossibleIndividual to which the relation applies, c is a floating point number with the property value and d the Scale as unit of measurement. b has a a type of ClassOfIndirectProperty, which has c value and d unit of measurement.

Order	Role name	Role type
1	Property type	**ClassOfIndirectProperty**
2	Property possessor	**PossibleIndividual**
3	Property value	**ExpressReal**
4	Property scale	**Scale**

IndirectPropertyScaleReal$(x_1, x_2, x_3, x_4) \leftrightarrow$

ClassOfIndirectProperty$(x_1) \wedge$

PossibleIndividual$(x_2) \wedge$

ExpressReal$(x_3) \wedge$

Scale$(x_4) \wedge$

$\exists u($*InstanceOfIndirectProperty*$(x_1, x_2, u) \wedge$

RealMagnitudeOfProperty(u, x_3, x_4)

)

6.3.14 Template StatusApproval

This template is for stating that an approver assigns a status to a relationship.

StatusApproval(a, b, c) means that a is an relationship and b is a class of approval by status and c is a approver and that c assigns a with status b.

Order	Role name	Role type
1	Relationship	**Relationship**
2	Status	**ClassOfApprovalByStatus**
3	Approver	**PossibleIndividual**

$$\begin{aligned}&\textit{StatusApproval}(x_1, x_2, x_3) \leftrightarrow \\ &\quad \textbf{Relationship}(x_1) \wedge \\ &\quad \textbf{ClassOfApprovalByStatus}(x_2) \wedge \\ &\quad \textbf{PossibleIndividual}(x_3) \wedge \\ &\quad \exists u(\textit{ApprovalTriple}(u, x_1, x_3) \wedge \\ &\quad \textit{ClassificationTemplate}(u, x_2))\end{aligned}$$

6.3.15 Template ClassifiedInvolvement

This template is for stating that a thing is involved in an activity, where the kind of involvement is classified.

NOTE 1 This template is also suitable for weak kinds of involvement, "by reference".

ClassifiedInvolvement(a, b, c) means that a is an thing and b is an activity and c is a type of involvement. a is involved in activity b and c is the type of involvement.

Order	Role name	Role type
1	Involved	**Thing**
2	Involver activity	**Activity**
3	Involvement type	**ClassOfInvolvementByReference**

$$\begin{aligned}&\textit{ClassifiedInvolvement}(x_1, x_2, x_3) \leftrightarrow \\ &\quad \textbf{Thing}(x_1) \wedge \\ &\quad \textbf{Activity}(x_2) \wedge \\ &\quad \textbf{ClassOfInvolvementByReference}(x_3) \wedge \\ &\quad \exists u(\textit{InvolvementByReferenceTriple}(u, x_1, x_2) \wedge \\ &\quad \textit{ClassificationTemplate}(u, x_3))\end{aligned}$$

6.3.16 Template InvolvementStatus

This template is for stating that a thing is involved in an activity, that the involvement is classified to be of a certain kind, and that the involvement is approved by an approver to have a certain status.

InvolvementStatus(a, b, c, d, e) means that a is an thing and b is an activity and c is a type of involvement, d is an approval status and e is the approver. a is involved in activity b and c is the type of involvement, the activity is approved, d is type of status of the approval and e is the approver.

Order	Role name	Role type
1	Involved	**Thing**
2	Involver activity	**Activity**
3	Involvement type	**ClassOfInvolvementByReference**
4	Status	**ClassOfApprovalByStatus**
5	Approver	**PossibleIndividual**

$\mathit{InvolvementStatus}(x_1, x_2, x_3, x_4, x_5) \leftrightarrow$

$\quad \mathbf{Thing}(x_1) \wedge$

$\quad \mathbf{Activity}(x_2) \wedge$

$\quad \mathbf{ClassOfInvolvementByReference}(x_3) \wedge$

$\quad \mathbf{ClassOfApprovalByStatus}(x_4) \wedge$

$\quad \mathbf{PossibleIndividual}(x_5) \wedge$

$\quad \exists u(\mathit{InvolvementByReferenceTriple}(u, x_1, x_2) \wedge$

$\qquad \mathit{ClassificationTemplate}(u, x_3) \wedge$

$\qquad \mathit{StatusApproval}(u, x_4, x_5))$

6.3.17 Template InvolvementStatusBeginning

This template is for stating that starting at a certain time, a thing is involved in an activity, and that the involvement is classified to be of a certain kind, and that the involvement is approved by an approver to have a certain status.

$\mathit{InvolvementStatusBeginning}(a, b, c, d, e, f)$ means that a is an thing and b is an activity and c is a type of involvement, d is an approval status and e is the approver and f is a point in time. a is involved in activity b and c is the type of involvement, the activity is approved, d is type of status of the approval and e is the approver, f is the start time of the activity.

Order	Role name	Role type
1	Involved	**Thing**
2	Involver activity	**Activity**
3	Involvement type	**ClassOfInvolvementByReference**
4	Status	**ClassOfApprovalByStatus**
5	Approver	**PossibleIndividual**
6	Start time	**RepresentationOfGregorianDateAndUtcTime**

$\mathit{InvolvementStatusBeginning}(x_1, x_2, x_3, x_4, x_5) \leftrightarrow$

$\quad \mathbf{Thing}(x_1) \wedge$

$\quad \mathbf{Activity}(x_2) \wedge$

$\quad \mathbf{ClassOfInvolvementByReference}(x_3) \wedge$

$\quad \mathbf{ClassOfApprovalByStatus}(x_4) \wedge$

$\quad \mathbf{PossibleIndividual}(x_5) \wedge$

$\quad \mathbf{RepresentationOfGregorianDateAndUtcTime}(x_6) \wedge$

$\quad \exists u(\mathit{BeginningOfTemporalPart}(u, x_2, x_6) \wedge$

$\qquad \mathit{InvolvementStatus}(x_1, u, x_3, x_4, x_5))$

6.3.18 Template SuccessionOfInvolvementByReference

This template is for expressing that one involvement is succeeded by another. *SuccessionOfInvolvementByReference*(a, b) means that a is a involvement and b is an involvement, and that a is succeeded by b.

NOTE *SuccessionOfInvolvementByReference* is an *RDL template* because the reference individual **InvolvementSuccession** appears in its template axiom.

Order	Role name	Role type
1	Predecessor	**InvolvementByReference**
2	Successor	**InvolvementByReference**

$$\begin{aligned}&\textit{SuccessionOfInvolvementByReference}(x_1, x_2) \leftrightarrow \\ &\quad \textbf{InvolvementByReference}(x_1) \wedge \\ &\quad \textbf{InvolvementByReference}(x_2) \wedge \\ &\quad \textit{InstanceOfRelation}(\textbf{InvolvementSuccession}, x_1, x_2)\end{aligned}$$

6.3.19 Template SuccessionOfInvolvementInActivity

This template is for expressing that in an activity, one thing that's involved in the activity is succeeded by another.

NOTE 1 Here the involved things are arguments to the template, not the involvement relationships themselves.

SuccessionOfInvolvementInActivity(a, b, c) means that a is a involvement and b is an involvement, and c an activity. b succeeds a, and both are involved in activity c.

Order	Role name	Role type
1	Involved predecessor	**Thing**
2	Involved successor	**Thing**
3	Involver	**Activity**

$$\begin{aligned}&\textit{SuccessionOfInvolvementInActivity}(x_1, x_2, x_3) \leftrightarrow \\ &\quad \textbf{Thing}(x_1) \wedge \\ &\quad \textbf{Thing}(x_2) \wedge \\ &\quad \textbf{Activity}(x_3) \wedge \\ &\quad \exists u_1 \exists u_2 (\\ &\qquad \textit{InvolvementByReferenceTriple}(u_1, x_1, x_3) \wedge \\ &\qquad \textit{InvolvementByReferenceTriple}(u_2, x_2, x_3) \wedge \\ &\qquad \textit{SuccessionOfInvolvementByReference}(u_1, u_2))\end{aligned}$$

7 Templates for classes

7.1 Purpose

Templates that serve to characterise types of entities are included under the heading "templates for classes". The characteristic case is to express that the (individual) members of a class, or the (ordered

pair) members of a relation, satisfy general constraints.

NOTE ISO 15926 does not require a sharp stratification of the universe into individuals, classes, metaclasses and so forth. What constitutes a class, a relation, etc. may be indeterminate in a given case.

7.2 Reference data items needed

The following reference data items are used in template axioms in this section. They constitute an extension of the ISO 15926-2 language with individual terms. For practical applications of templates whose definitions refer to these items, the items should be recorded in a reference data library.

Reference individual	Entity type
EmptyClass	**Class**
SetOf1Class	**ClassOfClass**
SetOf2Classes	**ClassOfClass**
SetOf3Classes	**ClassOfClass**
End1UniversalRestriction	**ClassOfSpecialization**
End2UniversalRestriction	**ClassOfSpecialization**
*** Cardinality**	**INTEGER**
Infinity	**ArithmeticNumber**
−Infinity	**ArithmeticNumber**
UomSymbolAssignment	**ClassOfClassOfIdentification**

Table 2 — Reference items: Templates for classes

7.2.1 Reference classes

The reference items **SetOf1Class**, **SetOf2Classes**, and **SetOf3Classes** are classes. Their intended use is as classifiers for **EnumeratedSetOfClass** instances that classify precisely one, two, or three classes, respectively. (Classes with higher cardinalities may be added to reference data.)

NOTE The reference items **SetOf1Class**, etc., are used in this part of ISO 15926 to represent that instances of the entity type **EnumeratedSetOfClass** have all members "explicitly given". For an **EnumeratedSetOfClass** *A*, membership in *A* is expressed by classification relationships. The classification of *A* by, e.g., **SetOf3Classes**, represents that there are precisely 3 such members.

The reference individual **EmptyClass** is a class which is defined to have no members. Semantically, the class can be *defined* as the result of applying **DifferenceOfSetOfClass** to an **EnumeratedSetOfClass** containing just a single class (which class is chosen is insignificant), as illustrated in the following diagram.

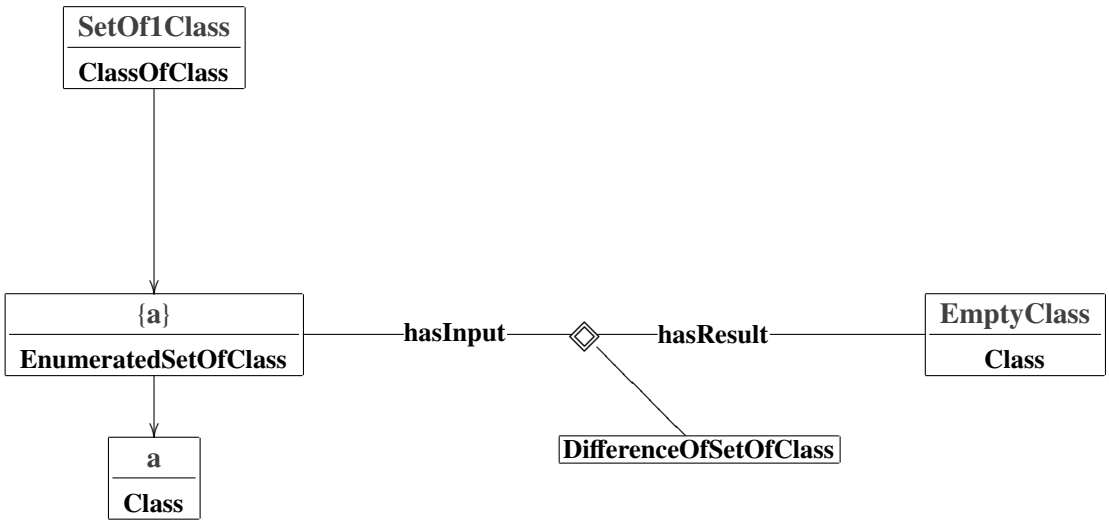

Figure 16 — Use of EmptyClass

NOTE In Figure 16, the **EnumeratedSetOfClass** instance {a} is named using brace set notation, to emphasise its natural interpretation as a set containing the element a. This pattern of naming is not intended to be normative.

The reference item * **Cardinality** is used in this part of ISO 15926 to represent unlimited maximum cardinality.

The reference items **Infinity** and **−Infinity** are used in this part of ISO 15926 to represent positive and negative infinity, as required for the definition of number ranges.

7.2.2 Reference relations

The items **End1UniversalRestriction**, **End2UniversalRestriction**, and **UomSymbolAssignment** are relations (their entity types are subtypes of **ClassOfRelationship**). This means an RDL record needs to assign classes to the roles of these items, as required by the respective entity types. For present purposes, stating general requirements on such assignment in an RDL is sufficient.

End1UniversalRestriction and **End2UniversalRestriction** are intended to be used for characterising subrelation relationships. For each item, the required roles are **hasClassOfSubclass** and **hasClassOfSuperclass**. The roles should both be assigned a reference item representing the entity type **ClassOfRelationship**.

UomSymbolAssignment is intended to be used for assigning symbols to scales. The required roles are **hasClassOfRepresented** and **hasClassOfPattern**. They should be assigned reference items representing the entity types **Scale** and **ExpressString**, respectively.

7.3 Representing complex classes

In ISO 15926, the class operations *union*, *intersection*, and *complement* are expressed by means of the functional relations **UnionOfSetOfClass**, **IntersectionOfSetOfClass**, and **DifferenceOfSetOfClass**, respectively. Every instance of these entity types relates a set of classes (an item of type **EnumeratedSetOfClass**) in the **hasInput** role to a single class, in the **hasResult** role.

Let A be a set of classes $a_1, a_2, \ldots, a_n$. With A in the **hasInput** role of an instance of **UnionOfSetOfClass**,

IntersectionOfSetOfClass, or **DifferenceOfSetOfClass**, the **hasResult** role is a class with the following definition.

for **UnionOfSetOfClass**: The union of the sets in A, i.e., the set of elements that belong to to a_1, or to a_2, etc.;

for **IntersectionOfSetOfClass**: The intersection of the sets in A, i.e., the set of elements that belong to a_1, and also to a_2, etc.;

for **DifferenceOfSetOfClass**: The members of the union of the sets in A that do not belong to the intersection of the sets in A.

Templates that provide for expressing complex class definitions include *UnionOf2Classes*, *IntersectionOf2Classes*, and *RelativeComplementOf2Classes*.

EXAMPLE Let A, B, and C be entities of type **EnumeratedSetOfClass**: A is the class containing just the class **MOTOR**. B is the class containing the classes **ELECTRIC MOTOR** and **HYDRAULIC MOTOR**. C is the class containing the classes **PUMP** and **PIPE** (In brace notation, we say A is {**MOTOR**}, B is {**ELECTRIC MOTOR**, **HYDRAULIC MOTOR**}, and C is {**PUMP**, **PIPE**}.)

Proto-templates that support the set operations are **UnionOfSetOfClassTemplate**, **IntersectionOfSetOfClassTemplate**, and **DifferenceOfSetOfClassTemplate**. Below, we abbreviate using set notation, writing $\bigcup X$, $\bigcap X$, and $\bigcup X \setminus \bigcap X$ to designate the value of the second argument of these respective templates, given X as the first argument. With default interpretations saying **MOTOR** is the class of motors, etc., we have:

$\bigcup A$ is the class of motors (the class **MOTOR**).

$\bigcup B$ is the class of motors that are electric or hydraulic.

$\bigcap B$ is the class of motors that are both electric and hydraulic.

$\bigcup${**MOTOR**, **ELECTRIC MOTOR**} $\setminus \bigcap$ {**MOTOR**, **ELECTRIC MOTOR**} is the class of non-electric motors.

$\bigcap C$ is the (typically, empty) class of things that are both **PUMP** and **PIPE**.

$\bigcup$ {**MOTOR**, $\bigcup B$} $\setminus \bigcap$ {**MOTOR**, $\bigcup B$} is the class of motors that are neither electric nor hydraulic.

NOTE Together with **IntersectionOfSetOfClass**, the **DifferenceOfSetOfClass** entity type is suitable for representing the common notion of set complement. The (relative) complement of two sets a and b is defined as $\{x \in a | x \notin b\}$. The **DifferenceOfSetOfClass** of $\{a, b\}$ is $\{x \in a \cup b \mid x \notin a \cap b\}$; let c designate this set. Then the relative complement of a and b is the **IntersectionOfSetOfClass** of $\{a, c\}$. This is captured in the template *RelativeComplementOf2Classes*.

7.4 Relation constraints

This section accounts describes how complex constraints on classes and relations can be represented using this part of ISO 15926.

7.4.1 Relations: Domain and co-domain

Introducing a relation between a pair of classes C and D doesn't in itself apply constraints on C's or on D's. We now consider how this part of ISO 15926 can be used to constrain classes according to the relations its members can enter into.

EXAMPLE A typical set of reference data for mechanical equipment is described in Figures 17 (*Permitted Ambient Temperature*) and 18 (*Celsius*). As illustrated, **Permitted Ambient Temperature** is a relation with **Equipment** in its **hasClassOfPossessor** role and **Temperature** in its **hasPropertySpace** role, and **Celsius** is a scale that relates **Temperature** properties to numbers.

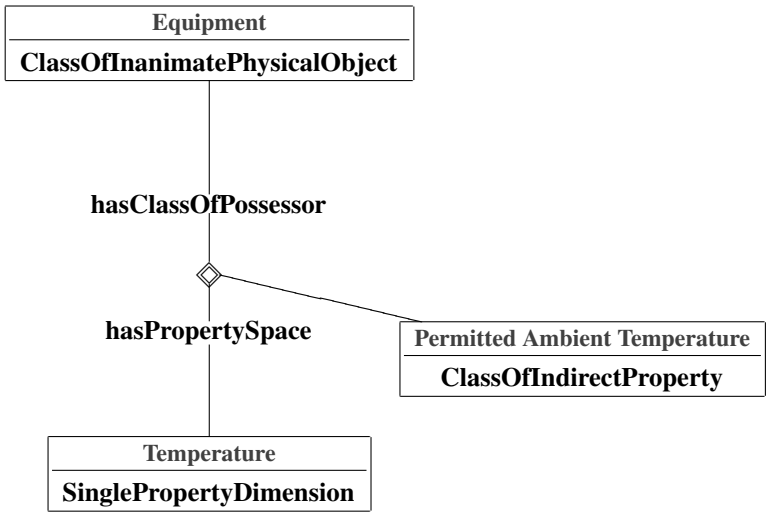

Figure 17 — Relation: Permitted Ambient Temperature

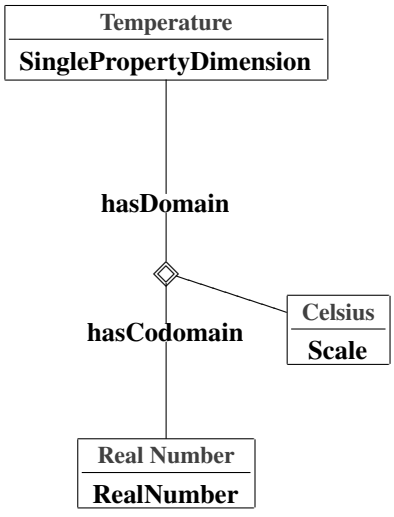

Figure 18 — Relation: Celsius

Figure 17 corresponds to the following set of statements in the ISO 15926-2 language.

ClassOfInanimatePhysicalObject(**Equipment**)
SinglePropertyDimension(**Temperature**)
ClassOfIndirectProperty(**Permitted Ambient Temperature**)
hasClassOfPossessor(**Permitted Ambient Temperature**, **Equipment**)
hasPropertySpace(**Permitted Ambient Temperature**, **Temperature**)

It is natural to consider the two directions (left–right and right–left) of a relation such as **Permitted Ambient Temperature** as distinct, directed relations:

a) a relation with domain **Equipment** and codomain[1)] **Temperature**

b) a relation with domain **Temperature** and codomain **Equipment**.

Different constraints may be applied to each direction.

7.4.2 Existential and universal constraints

Existential ("some") and universal ("all") constraints are of particular interest for reference data. In FOL notation, the characteristic cases correspond to expressions of the following forms, with C and D standing for classes and R for a relation.

$$C(x) \supset \exists y(R(x, y) \wedge D(y)) \tag{1}$$
$$C(x) \supset \forall y(R(x, y) \supset D(y)) \tag{2}$$

Formula (1) says that every C is related by R to at least one D; (2), that whenever a C is related by R, it is related to a D.

Existential constraints serve to express that instances of a given class are required to participate in some minimum number of relationships. In ISO 15926, they are expressed as cardinality restrictions on relations. Applied to the example of Figure 17, the one-to-many constraint pattern (1) on the left-to-right direction of the relation can be expressed as shown in Figure 19. Restriction to a maximum number of related instances can also be covered using cardinalities, as illustrated in Figure 20, where a cardinality constraint of $1 : 1$ ("precisely 1") is expressed. *Universal* constraints can not be expressed using cardinality constraints, and ISO 15926-2 doesn't provide primitives for them. In this part of ISO 15926, a conventional extension is provided for adding the expressive capability, with the reference data items **End1UniversalRestriction** and **End2UniversalRestriction**. They are used to classify relation specialisations, one item for each direction of a **ClassOfRelationship** (Figure 21). The relations to which universal constraints are applied are typically subrelations of generic relations. To represent a universal constraint, the **Specialization** of a relation is itself classified as an **End2UniversalRestriction**. The construction is interpreted according to the following convention (and accordingly for **End1UniversalRestriction**):

> If a **Specialization** S from R to subrelation R', where C' is the domain of R', is classified by **End2UniversalRestriction**, then every pair (x, y) in R for which $C'(x)$ holds is a member of R'. (3)

The pattern for classification of a relation specialisation by **End2UniversalRestriction** is illustrated in Figure 22. By convention in this part of ISO 15926, the classification expresses that restriction (4) holds.

$$C'(x) \supset \forall y(R(x, y) \supset D'(y)), \tag{4}$$

[1)] The term "codomain" is a synonym of "range", as used to refer to the set of second elements in a binary relation;

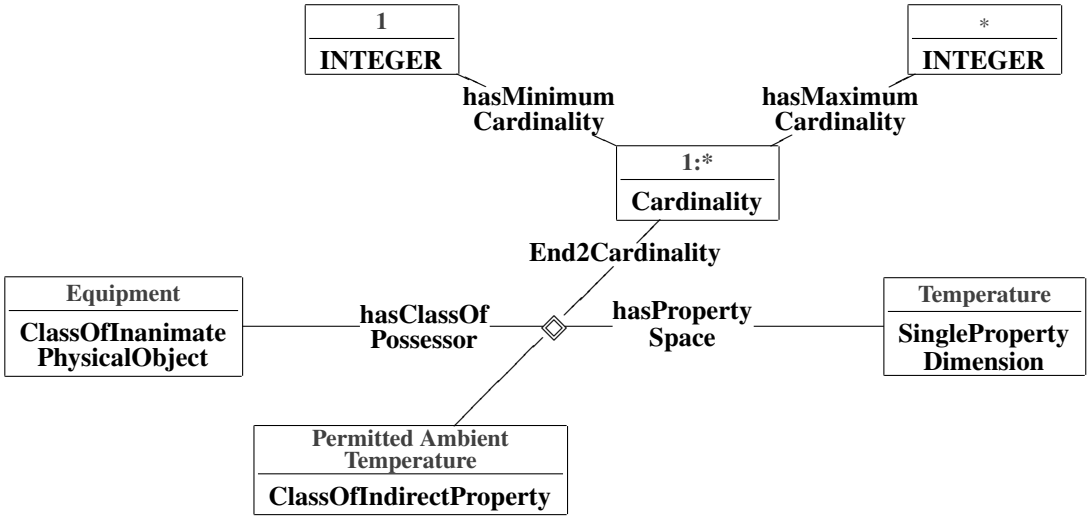

Figure 19 — Existential constraint: Permitted Ambient Temperature

EXAMPLE An example of the pattern is given in Figure 23, where an (unnamed) subrelation between **Type N Engine** and a **Temperature** range is given as a **Specialization** of the generic relationship **Permitted Ambient Temperature**.[2)] The classification as universal means that every assignment of **Permitted Ambient Temperature** to a **Type N Engine** is restricted to properties in the range **Temperature 0 to 70 degrees C.**

7.4.3 *At-most n* subrelation constraints

A common constraint pattern limits the number of relationships, of a given type, into which a given class may enter. Given a generic relation R from C to D, and a subclass C' of C (Figure 24), *at most* cardinality restrictions with regard to R may be applied to C' according to pattern (5), which says that a C' may be related by R to at most n things.

$$\forall x(C'(x) \supset \exists_{\leq n} y(R(x, y))) \tag{5}$$

As for universal constraints, the reference items **End1UniversalRestriction** and **End2UniversalRestriction** are used in this part of ISO 15926 to express *at most* constraints. To limit the number of R relationships for a class C', apply a cardinality restriction on a subrelation R' for which the domain is restricted to C'. The subrelation specialisation is then classified as a universal restriction on the pattern introduced above.

EXAMPLE An example of the *at most* constraint representation pattern is given in Figure 25, according to which every member of C' participates in between 1 and 3 R' relationships. The **End2UniversalRestriction** classification ensures that every R relationship with a C' in the left role is an R' relationship. Hence, C' is related by R to at most 3 D's.

[2)] See ISO 15926-2, 5.2.27.6, on the representation of a property range as a subclass of a property dimension.

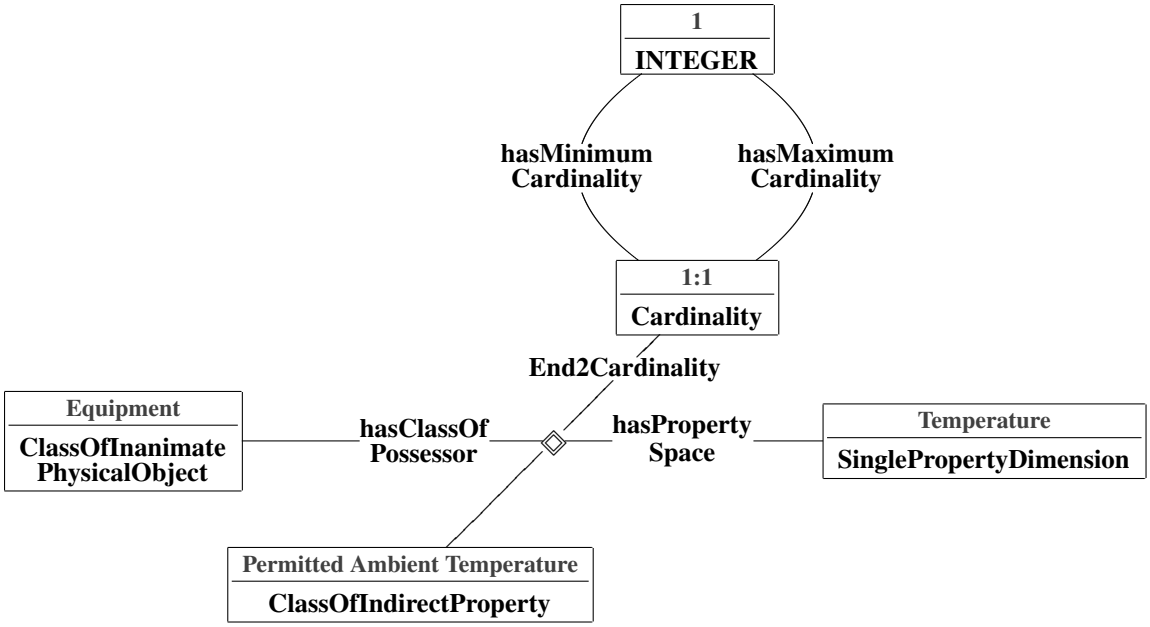

Figure 20 — Precisely-one constraint: Permitted Ambient Temperature

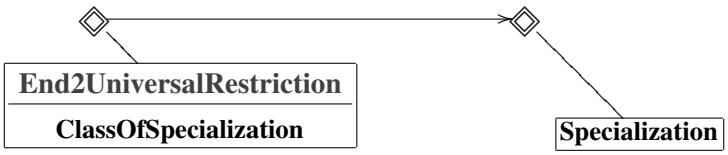

Figure 21 — Classification for universal constraint

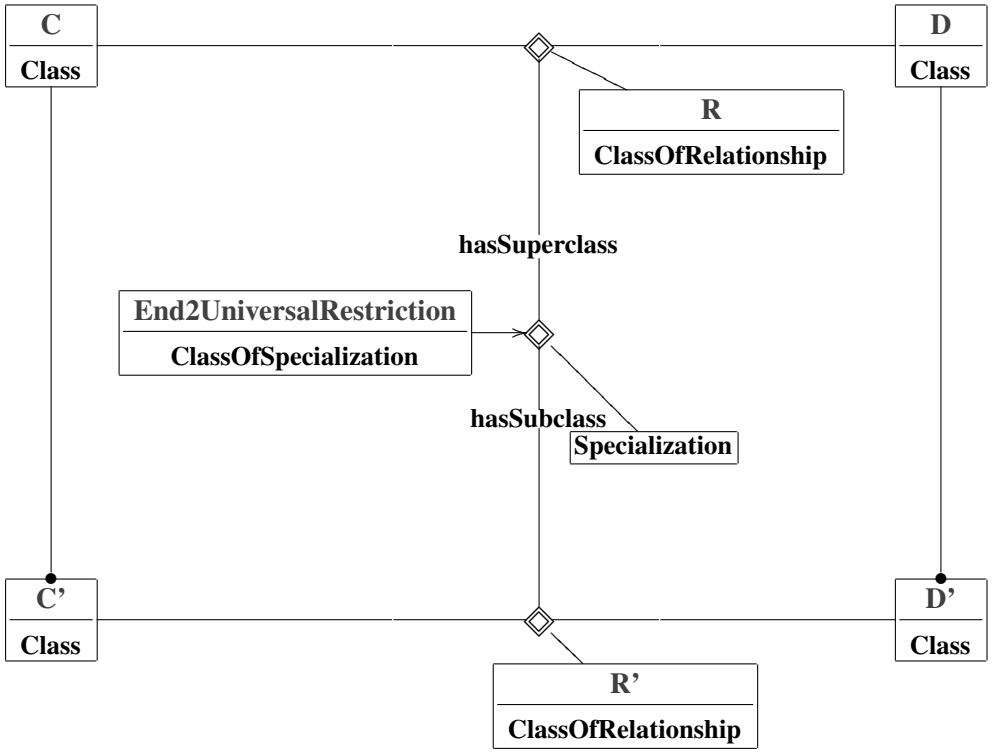

Figure 22 — Universal constraint pattern

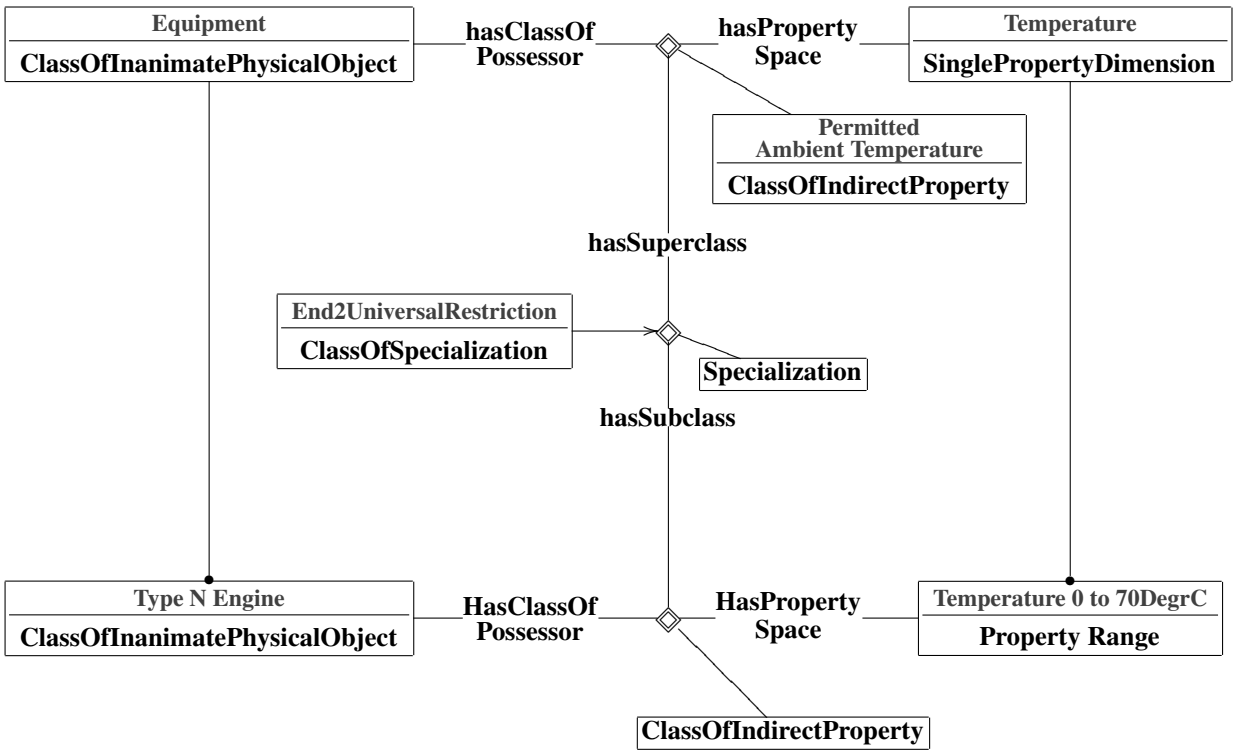

Figure 23 — Universal constraint: Equipment, Temperature

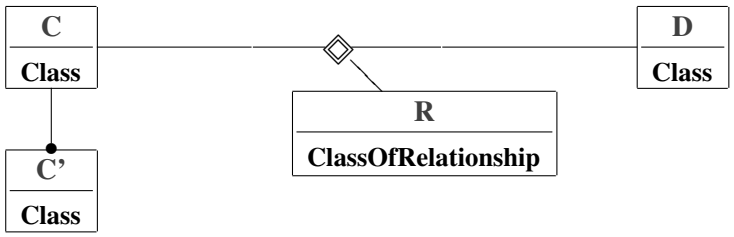

Figure 24 — Example for *at most* constraints

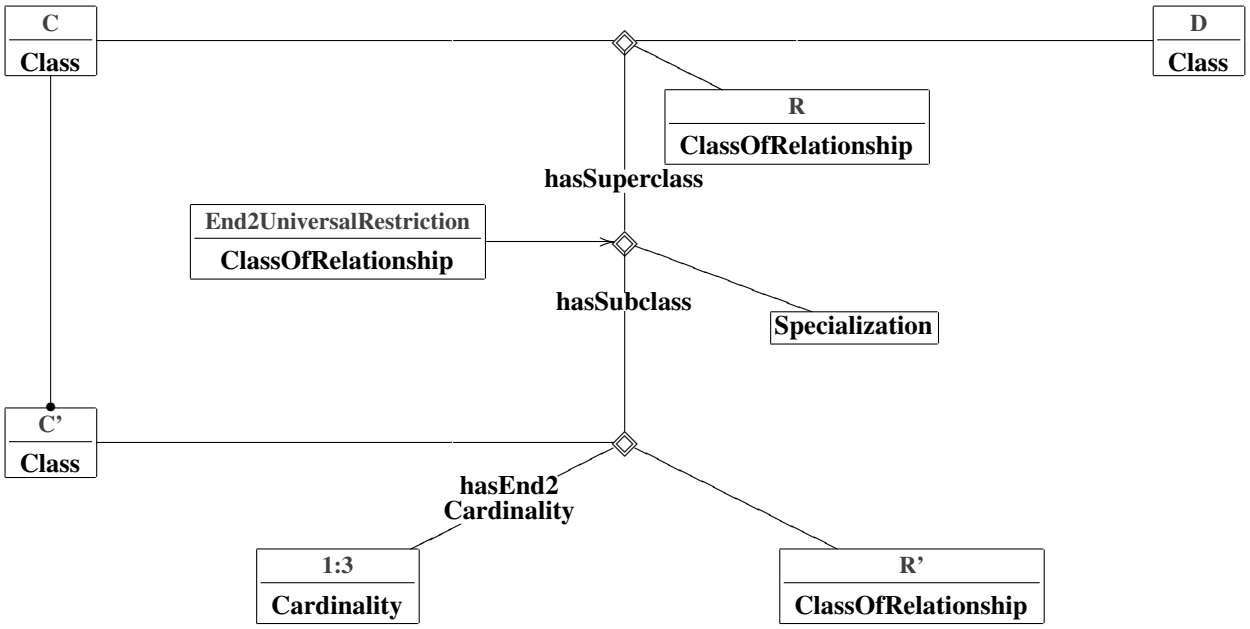

Figure 25 — Combining universal constraints and cardinalities

7.5 Initial set

7.5.1 Template ClassificationOfClass

This is a template for classifying classes.

ClassificationOfClass(a, b) means that a is a class, that b is a class of classes, and that a is a member of b.

Order	Role name	Role type
1	Class	**Class**
2	Class Classifier	**ClassOfClass**

ClassificationOfClass$(x_1, x_2) \leftrightarrow$

Class$(x_1) \wedge$

ClassOfClass$(x_2) \wedge$

ClassificationTemplate(x_1, x_2)

EXAMPLE Typical uses of this template is in classification of classes that are used by particular entities or which are defined in domain standards. No restriction on the order of the classes is imposed. As an example of classification of a second-order class by a third-order class, consider "the types of drill string belong to the drilling domain". This could be expressed by *ClassificationOfClass*(**Drilling Domain Class**, **Drill String Type**), which expands to statements in the ISO 15926-2 language as shown in the following diagram.

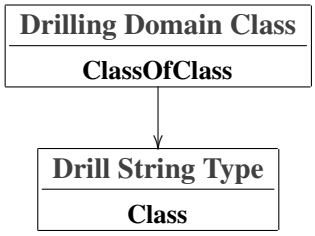

Figure 26 — Example template ClassificationOfClass

7.5.2 Template ClassificationOfClassOfIndividual

This is a template for classifying classes that only have individuals as members.

ClassificationOfClassOfIndividual(a, b) means that a is a first-order class, that b is a second-order class, and that a is a member of b.

Order	Role name	Role type
1	Class	**ClassOfIndividual**
2	Class Classifier	**ClassOfClassOfIndividual**

ClassificationOfClassOfIndividual$(x_1, x_2) \leftrightarrow$

ClassOfIndividual$(x_1) \wedge$

ClassOfClassOfIndividual$(x_2) \wedge$

ClassificationOfClass(x_1, x_2)

NOTE This template is a specialised version of *ClassificationOfClass*, with the added restriction that the first argument is a first-order class (a **ClassOfIndividual**), and the second argument is a second-order class (a **ClassOfClassOfIndividual**).

EXAMPLE The expansion of the statement *ClassificationOfClass*(**Drilling Class**, **Drill String**) is shown in the following diagram.

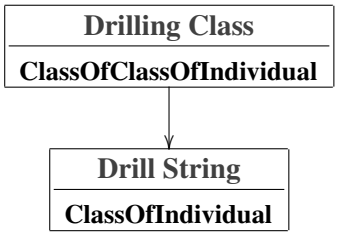

Figure 27 — Example template ClassificationOfClassOfIndividual

7.5.3 Template ClassificationOfClassOfRelationship

This is a template for classifying relations.

ClassificationOfClassOfRelationship(a, b) means that a is a relation and b a class of relations, and that a is a member of b.

Order	Role name	Role type
1	Class	**ClassOfRelationship**
2	Class Classifier	**ClassOfClassOfRelationship**

ClassificationOfClassOfRelationship$(x_1, x_2) \leftrightarrow$

ClassOfRelationship$(x_1) \wedge$

ClassOfClassOfRelationship$(x_2) \wedge$

ClassificationOfClass(x_1, x_2)

NOTE This template is a specialised version of *ClassificationOfClass*, with the added restriction that the first argument is a relation (a **ClassOfRelationship**), and the second argument is a class of relations (a **ClassOfClassOfRelationship**).

EXAMPLE The expansion of the statement *ClassificationOfClass*(**Shaft Seal Connection**, **Machinery Relation**) is shown in the following diagram.

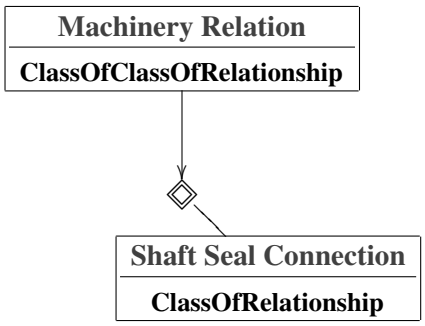

Figure 28 — Example template ClassificationOfClassOfRelationship

7.5.4 Template RelationOfIndividualsToIndividuals

This is a template for expressing that a relation relates individuals only.

RelationOfIndividualsToIndividuals(a) means that a is a relation of one of the subtypes of **ClassOfRelationship**, and that its domain and range (as determined by attributes according to the entity type) are both first-order classes.

Order	Role name	Role type
1	Relation	**ClassOfRelationship**

RelationOfIndividualsToIndividuals(x) ↔

ClassOfRelationship(x) ∧

$\exists y_1 \exists y_2$(*entityTriple*(x, y_1, y_2) ∧

ClassOfIndividual(y_1) ∧ **ClassOfIndividual**(y_2))

NOTE The purpose of this unary template is in expressing a constraint on relations. The use of the disjunctive *entityTriple* template (annex C.3) in the defining axiom means this template is not suitable for introducing relations.

NOTE The template language lacks the expressive power to fully capture the constraint intended for this template. A full expression would require universal quantification, stating that for *every* entity type to which the subject relation belongs, the attributes of the relation are first-order classes. Universal statements are beyond the scope of template definitions (cf. 4.2, annex H). The template still captures a useful approximation, because the intended constraint is satisfied given that the the subject relation has only one pair of attributes (i.e., has unique domain and range). This requirement is not captured in the ISO 15926-2 or template languages.

EXAMPLE The expansion of *RelationOfIndividualsToIndividuals*(**Shaft Seal Connection**) is a disjunctive statement that **Shaft Seal Connection** belongs to one of the subtypes of **ClassOfRelationship**, with the appropriate attributes filled by first-order classes. This statement is not properly suited for representation in a single diagram, but the following is offered as an informal illustration (where connectors have not been annotated with attribute names).

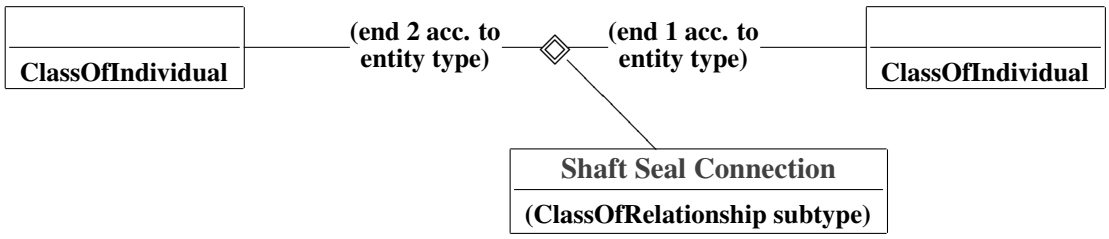

Figure 29 — Example template RelationOfIndividualsToIndividuals

7.5.5 Template SpecializationOfIndividualRelation

This is a template for expressing that one relation is a subrelation of another, constrained to relations between individuals.

SpecializationOfIndividualRelation(a, b) means that a and b are relations between individuals, and that a is a subrelation of b.

Order	Role name	Role type
1	Subrelation	**ClassOfRelationship**
2	Superrelation	**ClassOfRelationship**

SpecializationOfIndividualRelation$(x_1, x_2) \leftrightarrow$

ClassOfRelationship$(x_1) \wedge$

ClassOfRelationship$(x_2) \wedge$

RelationOfIndividualsToIndividuals$(x_1) \wedge$

RelationOfIndividualsToIndividuals$(x_2) \wedge$

SpecializationTemplate(x_1, x_2)

EXAMPLE The expansion of *SpecializationOfIndividualRelation*(**Shaft Seal Connection**, **Seal Connection**) is a disjunctive statement: cf. the example for *RelationOfIndividualsToIndividuals* for an explanation. The following diagram is offered as an informal illustration.

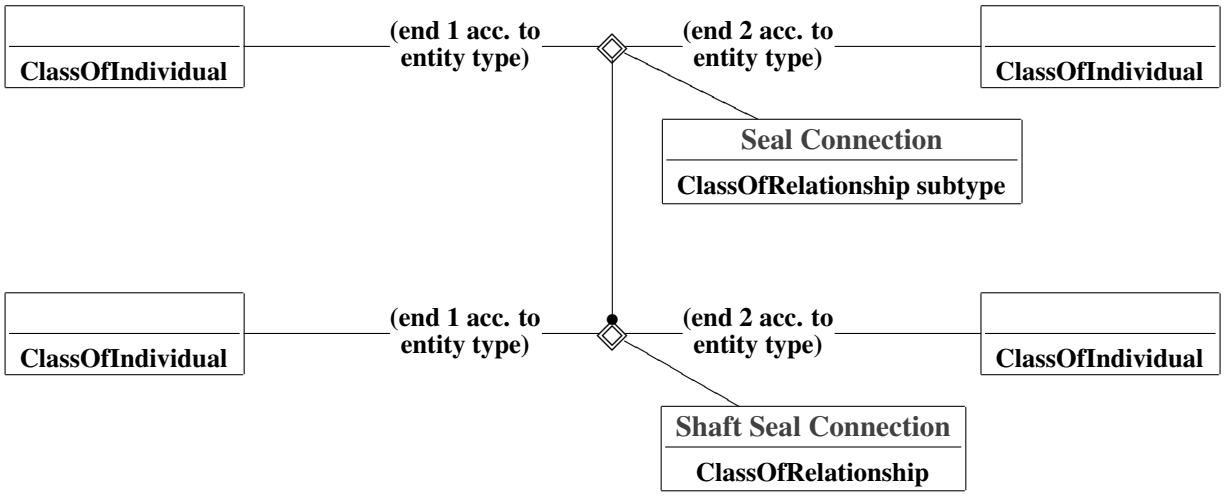

Figure 30 — Example template SpecializationOfIndividualRelation

7.5.6 Template EnumeratedSetOf2Classes

EnumeratedSetOf2Classes is a template for collecting two classes into a third.

EnumeratedSetOf2Classes(a, b, c) means that a, b, and c are classes, and that a has precisely b and c as members.

Order	Role name	Role type
1	Classified 1	**Class**
2	Classified 2	**Class**
3	Enumerated Set Of Class	**EnumeratedSetOfClass**

$$
\begin{aligned}
&\textit{EnumeratedSetOf2Classes}(x_1, x_2, x_3) \leftrightarrow \\
&\quad \textbf{Class}(x_1) \wedge \\
&\quad \textbf{Class}(x_2) \wedge \\
&\quad \textbf{EnumeratedSetOfClass}(x_3) \wedge \\
&\quad \textit{ClassificationTemplate}(x_1, x_3) \wedge \\
&\quad \textit{ClassificationTemplate}(x_2, x_3) \wedge \\
&\quad \textit{ClassificationTemplate}(x_3, \textbf{SetOf2Classes})
\end{aligned}
$$

NOTE The order in which the first two arguments is given is insignificant. The numbering in role names (“Classified 1”, “Classified 2”) is only applied in order to distinguish the roles from each other.

NOTE See 7.2.1 for further information on the representation used in this template.

EXAMPLE The statement *EnumeratedSetOf2Classes*(**Pump**, **Pipe**, {**Pump, Pipe**}) expands to the following representation.

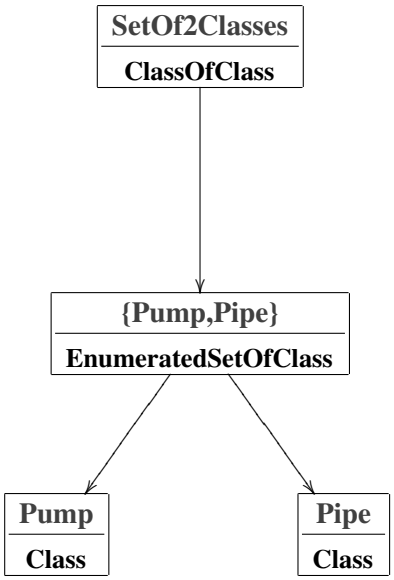

Figure 31 — Example template EnumeratedSetOf2Classes

7.5.7 Template EnumeratedSetOf3Classes

EnumeratedSetOf2Classes is a template for collecting three classes into a fourth.

EnumeratedSetOf2Classes(a, b, c, d) means that a, b, c, and d are classes, and that a has precisely b, c, and d as members.

Order	Role name	Role type
1	Classified 1	**Class**
2	Classified 2	**Class**
3	Classified 3	**Class**
4	Enumerated Set Of Class	**EnumeratedSetOfClass**

EnumeratedSetOf3Classes$(x_1, x_2, x_3, x_4) \leftrightarrow$

Class$(x_1) \wedge$

Class$(x_2) \wedge$

Class$(x_3) \wedge$

EnumeratedSetOfClass$(x_4) \wedge$

ClassificationTemplate$(x_1, x_4) \wedge$

ClassificationTemplate$(x_2, x_4) \wedge$

ClassificationTemplate$(x_3, x_4) \wedge$

ClassificationTemplate$(x_4,$ **SetOf3Classes**$)$

NOTE See 7.2.1 for further information on the representation used in this template.

7.5.8 Template UnionOf2Classes

UnionOf2Classes is a template for expressing that a class is the union of two classes.

UnionOf2Classes(a, b, c) means that a, b, and c are classes, and that c is the union of a and b.

Order	Role name	Role type
1	Class 1	**Class**
2	Class 2	**Class**
3	Class Union	**Class**

UnionOf2Classes(x_1, x_2, x_3) ↔

Class(x_1) ∧

Class(x_2) ∧

Class(x_3) ∧

∃y(*EnumeratedSetOf2Classes*(x_1, x_2, y) ∧

UnionOfSetOfClassTemplate(y, x_3))

NOTE See 7.3 for further information on the representation used in this template.

EXAMPLE The statement *UnionOf2Classes*(**Pump**, **Pipe**, **Pump∪Pipe**) expands to the following representation. Note: The designation of the item {**Pump,Pipe**} is included to make the diagram more readable, but is not defined by the expansion.

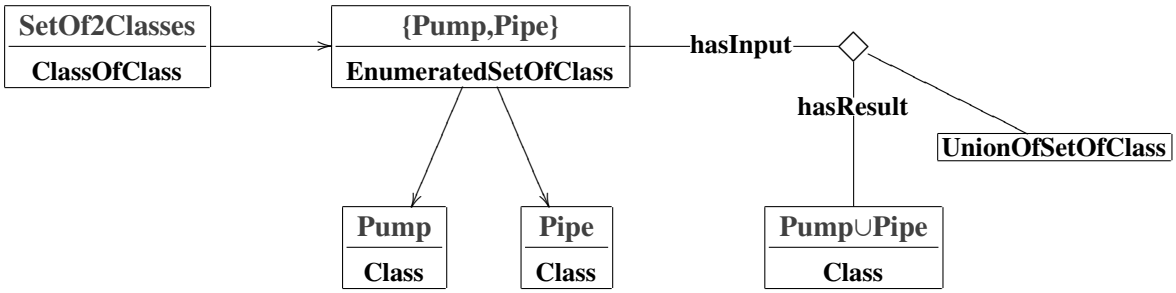

Figure 32 — Example template UnionOf2Classes

7.5.9 Template IntersectionOf2Classes

IntersectionOf2Classes is a template for expressing that a class is the intersection of two classes.

IntersectionOf2Classes(a, b, c) means that c is the intersection of a and b.

Order	Role name	Role type
1	Class 1	**Class**
2	Class 2	**Class**
3	Class Intersection	**Class**

IntersectionOf2Classes(x_1, x_2, x_3) ↔

Class(x_1) ∧

Class(x_2) ∧

Class(x_3) ∧

∃y(*EnumeratedSetOf2Classes*(x_1, x_2, y) ∧

IntersectionOfSetOfClassTemplate(y, x_3))

NOTE See 7.3 for further information on the representation used in this template.

7.5.10 Template DifferenceOf2Classes

DifferenceOf2Classes is a template for expressing that a class is the *difference*, as given in ISO 15926-2, of two classes.

DifferenceOf2Classes(a, b, c) means that c is the difference of a and b.

Order	Role name	Role type
1	Class 1	**Class**
2	Class 2	**Class**
3	Class Difference	**Class**

$$\textit{DifferenceOf2Classes}(x_1, x_2, x_3) \leftrightarrow$$
$$\textbf{Class}(x_1) \wedge$$
$$\textbf{Class}(x_2) \wedge$$
$$\textbf{Class}(x_3) \wedge$$
$$\exists y(\textit{EnumeratedSetOf2Classes}(x_1, x_2, y) \wedge$$
$$\textit{DifferenceOfSetOfClassTemplate}(y, x_3))$$

NOTE ISO 15926-2 defines the difference of two classes a and b as $(a \cup b)n(a \cap b)$. See 7.3 for further information on the representation used in this template.

7.5.11 Template RelativeComplementOf2Classes

RelativeComplementOf2Classes is a template for expressing that a class is the relative complement of two classes.

RelativeComplementOf2Classes(a, b, c) means that c is the relative complement of a and b.

Order	Role name	Role type
1	Class 1	**Class**
2	Class 2	**Class**
3	Class Relative Complement	**Class**

$$\textit{RelativeComplementOf2Classes}(x_1, x_2, x_3) \leftrightarrow$$
$$\textbf{Class}(x_1) \wedge$$
$$\textbf{Class}(x_2) \wedge$$
$$\textbf{Class}(x_3) \wedge$$
$$\exists y(\textit{DifferenceOf2Classes}(x_1, x_2, y) \wedge$$
$$\textit{IntersectionOf2Classes}(x_1, y, x_3))$$

NOTE The relative complement of two classes a and b is commonly written anb. See 7.3 for further information on the representation used in this template.

EXAMPLE The following diagram shows the expansion of *RelativeComplementOf2Classes*(**a**, **b**, **a\b**). Note: The designations of the items {**a,b**}, (**a** ∪ **b**)*n*(**a** ∩ **b**), and {**a**, (**a** ∪ **b**)*n*(**a** ∩ **b**)} are included here to make the diagram more readable, but are not defined by the expansion.

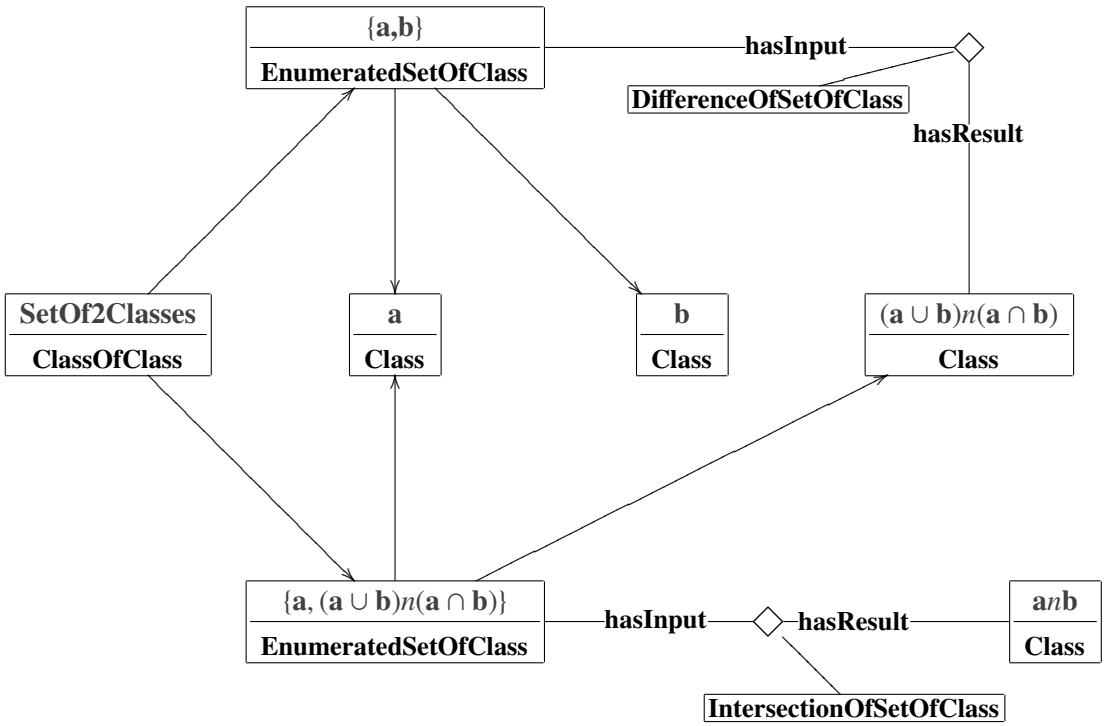

Figure 33 — Example template RelativeComplementOf2Classes

7.5.12 Template DisjointnessOf2Classes

This template expresses that two classes have no members in common (that the classes are disjoint). *DisjointnessOf2Classes*(x, y) means that the intersection of x and y is empty.

Order	Role name	Role type
1	Class 1	**Class**
2	Class 2	**Class**

DisjointnessOf2Classes(x_1, x_2) ↔

Class(x_1) ∧

Class(x_2) ∧

IntersectionOf2Classes(x_1, x_2, **EmptyClass**)

NOTE See 7.3 for further information on the representation used in this template. See 7.2.1 for a description of the item **EmptyClass** used in this template.

EXAMPLE The statement *DisjointnessOf2Classes*(**Pump**, **Pipe**, **EmptyClass**) expands to the following representation. Note: The designation of the item {**Pump,Pipe**} is included to make the diagram more readable, but is not defined by the expansion.

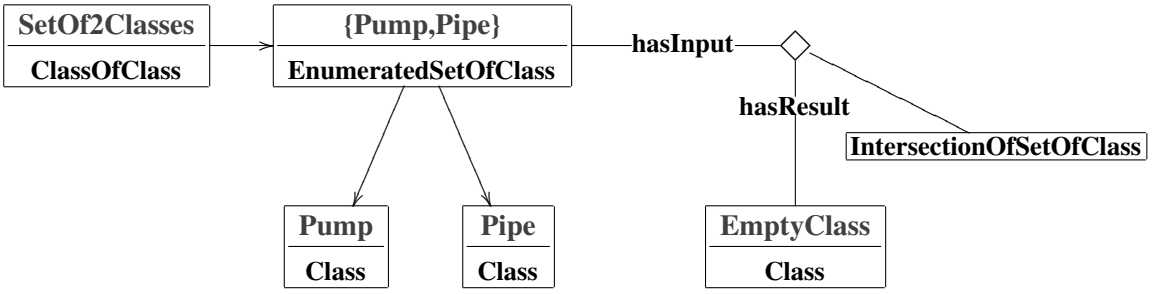

Figure 34 — Example template DisjointnessOf2Classes

7.5.13 Templates SpecializationAsEnd1UniversalRestriction, SpecializationAsEnd2UniversalRestriction

SpecializationAsEnd1UniversalRestriction and *SpecializationAsEnd2UniversalRestriction* are templates for expressing relation specialisations with the force of universal restriction.

These templates have the same roles, and similar definitions except for the use of reference items **End1UniversalRestriction**, resp. **End2UniversalRestriction**.

SpecializationAsEnd1UniversalRestriction means that *a* and *b* are relations, that *a* is a subrelation of *b*, and that the specialization relation between *a* and *b* is a member of **End1UniversalRestriction**.

Order	Role name	Role type
1	Subrelation	**ClassOfRelationship**
2	Superrelation	**ClassOfRelationship**

$$\textit{SpecializationAsEnd1UniversalRestriction}(x_1, x_2) \leftrightarrow$$
$$\textbf{ClassOfRelationship}(x_1) \wedge$$
$$\textbf{ClassOfRelationship}(x_2) \wedge$$
$$\exists y(\textit{SpecializationTriple}(y, x_1, x_2) \wedge$$
$$\textit{ClassificationTemplate}(y, \textbf{End1UniversalRestriction}))$$

$$\textit{SpecializationAsEnd2UniversalRestriction}(x_1, x_2) \leftrightarrow$$
$$\textbf{ClassOfRelationship}(x_1) \wedge$$
$$\textbf{ClassOfRelationship}(x_2) \wedge$$
$$\exists y(\textit{SpecializationTriple}(y, x_1, x_2) \wedge$$
$$\textit{ClassificationTemplate}(y, \textbf{End2UniversalRestriction}))$$

NOTE A statement *SpecializationAsEnd1UniversalRestriction*(a, b) implies that the specialization of *b* to *a* has the force of a universal restriction on the class that is the domain of *a*: any pair in *b* for which the first element is a member of the domain of *a* is a member of *a*. (Accordingly, *SpecializationAsEnd2UniversalRestriction*(a, b) constrains the range of *a*).

See 7.4 for further information on the representation used in this template.

EXAMPLE The expansion of the statement *SpecializationAsEnd2UniversalRestriction* (**Allowed Ambient Temp. 330A-3-874**, **Allowed Ambient Temperature**) is shown in the following diagram.

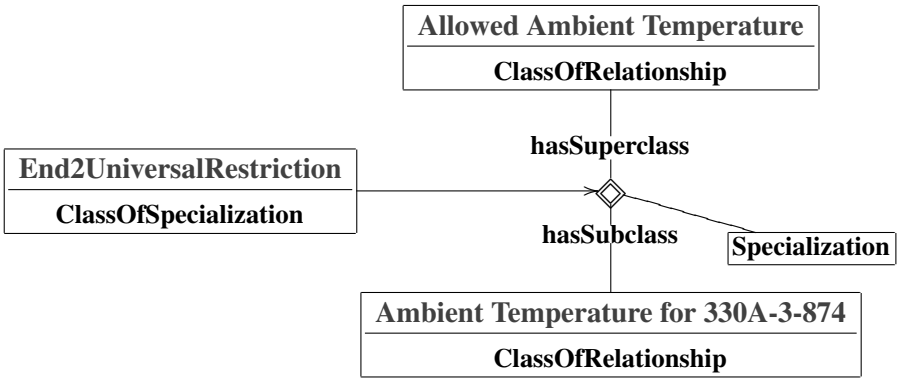

Figure 35 — Example template SpecializationAsEnd2UniversalRestriction

7.5.14 Templates CardinalityMin, CardinalityMax, CardinalityMinMax

CardinalityMin, *CardinalityMax*, and *CardinalityMinMax* are templates for expressing the values of cardinalities.

CardinalityMinMax(a, b, c) means that a is a cardinality and b and c are integers, and that b is the minimal, c the maximal, constraint of a. *CardinalityMin* and *CardinalityMax* are similar, and apply to just the minimal, resp. the maximal constraint.

CardinalityMin

Order	Role name	Role type
1	Cardinality	**Cardinality**
2	Minimum Cardinality	**INTEGER**

CardinalityMax

Order	Role name	Role type
1	Cardinality	**Cardinality**
2	Maximum Cardinality	**INTEGER**

CardinalityMinMax

Order	Role name	Role type
1	Cardinality	**Cardinality**
2	Minimum Cardinality	**INTEGER**
3	Maximum Cardinality	**INTEGER**

CardinalityMin$(x_1, x_2) \leftrightarrow$

Cardinality$(x_1) \wedge$

INTEGER$(x_2) \wedge$

hasMinimumCardinality(x_1, x_2)

CardinalityMax$(x_1, x_2) \leftrightarrow$

Cardinality$(x_1) \wedge$

INTEGER$(x_2) \wedge$

hasMaximumCardinality(x_1, x_2)

CardinalityMinMax$(x_1, x_2, x_3) \leftrightarrow$

Cardinality$(x_1) \wedge$

INTEGER$(x_2) \wedge$

INTEGER$(x_3) \wedge$

CardinalityMin$(x_1, x_2) \wedge$

CardinalityMax(x_1, x_3)

NOTE In ISO 15926, cardinalities are first-class objects. In ISO 15926-2, it is stated that an absence of specified minimum or maximum values for a cardinality should be interpreted as a absence of constraints (clause 5.2.13.1). The nature of the representation of ISO 15926-2 in first-order logic, with an *open world* assumption, mandates that both lower and upper bounds be given explicitly. Where no minimal constraint applies, the value 0 should be assigned. Where no maximum constraint applies, the reference item *** Cardinality** should be assigned (see 7.2.1).

EXAMPLE The statement *CardinalityMin*(**2:***, **2**, *** Cardinality**), as suitable for a cardinality with minimum 2 and unlimited maximum, is shown in the following diagram..

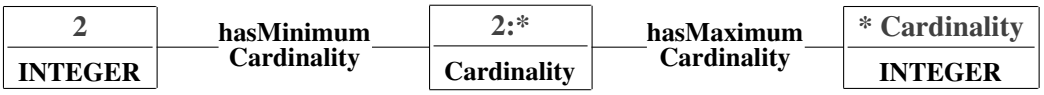

Figure 36 — Example template CardinalityMinMax

7.5.15 Cardinality Assignment Templates

CardinalityEnd1Min, *CardinalityEnd1Max*, *CardinalityEnd1MinMax*, *CardinalityEnd2Min*, *CardinalityEnd2Max*, and *CardinalityEnd2MinMax* are templates for expressing cardinality constraints on relations.

CardinalityEnd1MinMax(a, b) means that a is a relation and b and c are integers, and that the first role of a has b as minimum, c as maximum cardinality. The other templates in this group follow the same pattern.

CardinalityEnd1Min

Order	Role name	Role type
1	Relationship	**ClassOfRelationship**
2	Minimum cardinality	**INTEGER**

CardinalityEnd1Max

Order	Role name	Role type
1	Relationship	**ClassOfRelationship**
2	Maximum cardinality	**INTEGER**

CardinalityEnd1MinMax

Order	Role name	Role type
1	Relationship	**ClassOfRelationship**
2	Minimum cardinality	**INTEGER**
3	Maximum cardinality	**INTEGER**

CardinalityEnd2Min

Order	Role name	Role type
1	Relationship	**ClassOfRelationship**
2	Minimum cardinality	**INTEGER**

CardinalityEnd2Max

Order	Role name	Role type
1	Relationship	**ClassOfRelationship**
2	Maximum cardinality	**INTEGER**

CardinalityEnd2MinMax

Order	Role name	Role type
1	Relationship	**ClassOfRelationship**
2	Minimum cardinality	**INTEGER**
3	Maximum cardinality	**INTEGER**

$$\textit{CardinalityEnd1Min}(x_1, x_2) \leftrightarrow$$
$$\quad \textbf{ClassOfRelationship}(x_1) \wedge$$
$$\quad \textbf{INTEGER}(x_2) \wedge$$
$$\quad \exists u(\textit{CardinalityMin}(u, x_2) \wedge \textbf{hasEnd1Cardinality}(x_1, u))$$

$$\textit{CardinalityEnd1Max}(x_1, x_2) \leftrightarrow$$
$$\quad \textbf{ClassOfRelationship}(x_1) \wedge$$
$$\quad \textbf{INTEGER}(x_2) \wedge$$
$$\quad \exists u(\textit{CardinalityMax}(u, x_2) \wedge \textbf{hasEnd1Cardinality}(x_1, u))$$

$CardinalityEnd1MinMax(x_1, x_2, x_3) \leftrightarrow$
ClassOfRelationship$(x_1) \wedge$
INTEGER$(x_2) \wedge$
INTEGER$(x_3) \wedge$
$\exists u(CardinalityMinMax(u, x_2, x_3) \wedge$
hasEnd1Cardinality$(x_1, u))$

$CardinalityEnd2Min(x_1, x_2) \leftrightarrow$
ClassOfRelationship$(x_1) \wedge$
INTEGER$(x_2) \wedge$
$\exists u(CardinalityMin(u, x_2) \wedge$
hasEnd2Cardinality$(x_1, u))$

$CardinalityEnd2Max(x_1, x_2) \leftrightarrow$
ClassOfRelationship$(x_1) \wedge$
INTEGER$(x_2) \wedge$
$\exists u(CardinalityMax(u, x_2) \wedge$
hasEnd2Cardinality$(x_1, u))$

$CardinalityEnd2MinMax(x_1, x_2, x_3) \leftrightarrow$
ClassOfRelationship$(x_1) \wedge$
INTEGER$(x_2) \wedge$
INTEGER$(x_3) \wedge$
$\exists u(CardinalityMinMax(u, x_2, x_3) \wedge$
hasEnd2Cardinality$(x_1, u))$

NOTE For unlimited minimum or maximum cardinalities, assign the values 0 and * **Cardinality**, respectively. See the definitions of templates *CardinalityMin*, etc., above.

NOTE What constitutes the first and second roles of a relation is stated in the table of proto-templates (D.1, page 95).

NOTE Let R be a relation to which a cardinality $n_1 : m_1$ is assigned to the first role, and a cardinality $n_2 : m_2$ to the second. This means that (1) each instance of the first role is R-related to at least n_2 and at most m_2 distinct instances of the second role, and (2) each instance of the second role is related to at least n_1 and at most m_1 distinct instances of the first role.

EXAMPLE The statement *CardinalityEnd1MinMax*(**6 M8 bolt assembly**, **6**, **6**) is suitable for expressing an example of a constraint given in ISO 15926-2, clause 4.10.3, that "[t]he **class_of_relationship_with_signature** '6 M8 bolt assembly' has a cardinality such that each '6 of M8 bolts' is linked by exactly 6 relationships to different M8 bolts at all times". The expansion of this statement is shown in the following diagram.

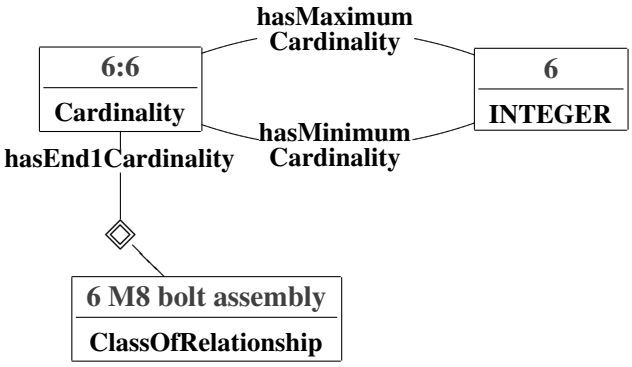

Figure 37 — Example template CardinalityEnd1MinMax

7.5.16 Template TimeRepresentation

This is a template for stating the coordinates of points in time.

PointInTime(a, b, c, d, e, f, g) means that a is a representation of a point in time, b, c, d, e, and f are integer numbers, and g a real number, and that the coordinates of a are given by b representing the year, c the month, d the day, e the hour, f the minute, and g the second of a.

Order	Role name	Role type
1	Time	**RepresentationOfGregorianDateAndUtcTime**
2	Year	**INTEGER**
3	Month	**INTEGER**
4	Day	**INTEGER**
5	Hour	**INTEGER**
6	Minute	**INTEGER**
7	Second	**REAL**

TimeRepresentation$(x_1, x_2, x_3, x_4, x_5, x_6, x_7) \leftrightarrow$

RepresentationOfGregorianDateAndUtcTime$(x_1) \wedge$

INTEGER$(x_2) \wedge$

INTEGER$(x_3) \wedge$

INTEGER$(x_4) \wedge$

INTEGER$(x_5) \wedge$

INTEGER$(x_6) \wedge$

REAL$(x_7) \wedge$

hasYear$(x_1, x_2) \wedge$

hasMonth$(x_1, x_3) \wedge$

hasDay$(x_1, x_4) \wedge$

hasHour$(x_1, x_5) \wedge$

hasMinute$(x_1, x_6) \wedge$

hasSecond(x_1, x_7)

NOTE This template provides a pattern for defining points in time. While the entity type **RepresentationOfGregorianDateAndUtcTime** has all attributes except **hasYear** optional, none are optional in the template.

7.5.17 Template MagnitudeOfProperty

MagnitudeOfProperty is a template for stating magnitude of properties.

MagnitudeOfProperty(a, b, c) means that a is a property, b is a number, and c is a scale, and that b is the value of a as measured on the scale c.

Order	Role name	Role type
1	Property	**Property**
2	Property value	**ArithmeticNumber**
3	Property scale	**Scale**

$$\begin{aligned}&\textit{MagnitudeOfProperty}(x_1, x_2, x_3) \leftrightarrow \\ &\quad \textbf{Property}(x_1) \wedge \\ &\quad \textbf{ArithmeticNumber}(x_2) \wedge \\ &\quad \textbf{Scale}(x_3) \wedge \\ &\quad \exists u(\textit{PropertyQuantificationTriple}(u, x_1, x_2) \wedge \\ &\quad \textit{ClassificationTemplate}(u, x_3))\end{aligned}$$

EXAMPLE The expansion of the statement *MagnitudeOfProperty*(**Mass 2 kg**, **2**, **Kilogramme**) is shown in the following diagram.

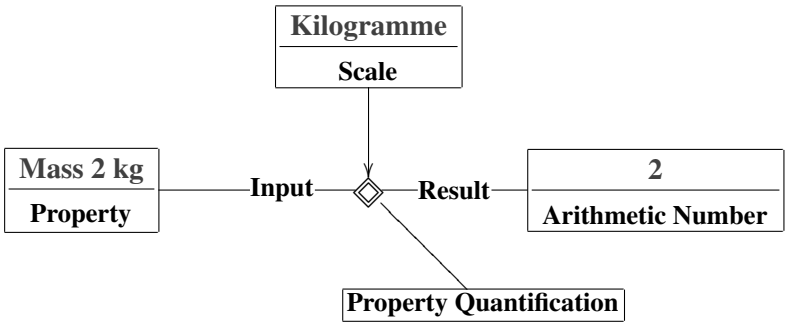

Figure 38 — Example template MagnitudeOfProperty

7.5.18 Template LowerUpperOfNumberRange

LowerUpperOfNumberRange is a template for stating what the upper and lower bounds of a number range are.

LowerUpperOfNumberRange(a, b, c) means that a is a number range and b and c are numbers, and that b is the lower, c the upper bound of a.

Order	Role name	Role type
1	Range	**NumberRange**
2	Lower bound	**ArithmeticNumber**
3	Upper bound	**ArithmeticNumber**

$$\textit{LowerUpperOfNumberRange}(x_1, x_2, x_3) \leftrightarrow$$
$$\textbf{NumberRange}(x_1) \wedge$$
$$\textbf{ArithmeticNumber}(x_2) \wedge$$
$$\textbf{ArithmeticNumber}(x_3) \wedge$$
$$\textit{LowerBoundOfNumberRangeTemplate}(x_2, x_1) \wedge$$
$$\textit{UpperBoundOfNumberRangeTemplate}(x_3, x_1)$$

NOTE To represent number ranges which are not bounded, use the reference items **Infinity** and **−Infinity** (see 7.2.1).

EXAMPLE The expansion of the statement *LowerUpperOfNumberRange*(**[-273.1 to Infinity]**, **-273.1**, **Infinity**) is shown in the following diagram.

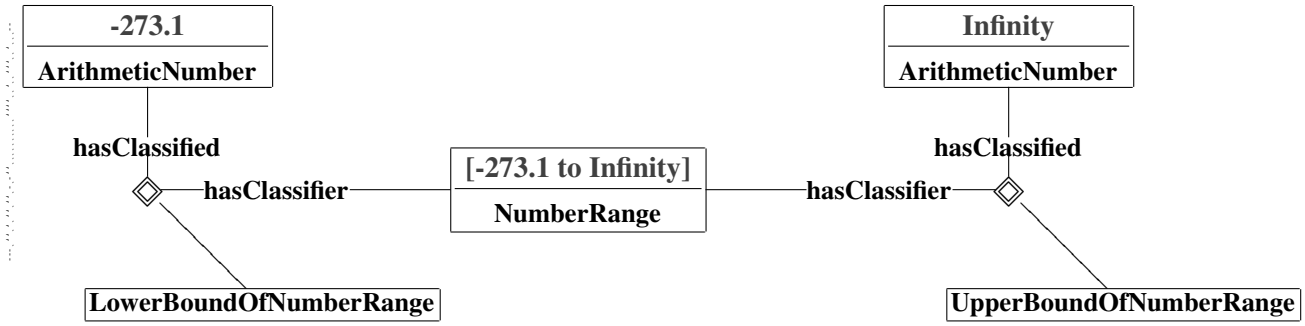

Figure 39 — Example template LowerUpperOfNumberRange

7.5.19 Template LowerUpperOfPropertyRange

LowerUpperOfPropertyRange is a template for stating what the upper and lower bounds of a property range are.

LowerUpperOfPropertyRange(a, b, c) means that a is a property range and b and c are properties, and that b is the lower, c the upper bound of a.

Order	Role name	Role type
1	Property Range	**PropertyRange**
2	Lower Bound	**Property**
3	Upper Bound	**Property**

LowerUpperOfPropertyRange(x_1, x_2, x_3) ↔

PropertyRange(x_1) ∧

Property(x_2) ∧

Property(x_3) ∧

LowerBoundOfPropertyRangeTemplate(x_2, x_1) ∧

UpperBoundOfPropertyRangeTemplate(x_3, x_1)

EXAMPLE The expansion of the statement *LowerUpperOfPropertyRange*(**0 to 20 Fahrenheit**,**0 Fahrenheit**, **20 Fahrenheit**) is shown in the following diagram.

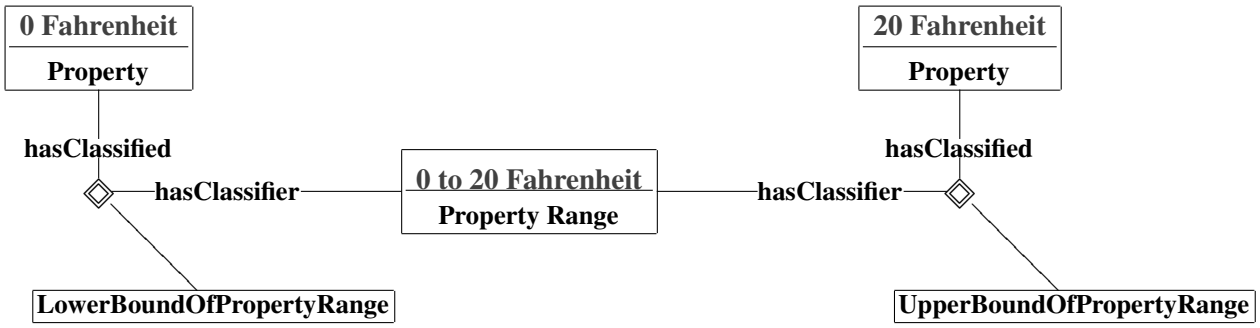

Figure 40 — Example template LowerUpperOfPropertyRange

7.5.20 Template LowerUpperMagnitudeOfPropertyRange

LowerUpperMagnitudeOfPropertyRange is a template for expressing the extent of a property range, in terms of maximum and minimum values on a scale.

LowerUpperMagnitudeOfPropertyRange(a, b, c, d) means that a is a property range, b is a scale, and c and d are numbers, and that a has c and d as, respectively, lower and upper bounds, as measured on the scale b.

Order	Role name	Role type
1	Property range	**PropertyRange**
2	Scale	**Scale**
3	Lower bound	**ArithmeticNumber**
4	Upper bound	**ArithmeticNumber**

$$\begin{aligned}&\textit{LowerUpperMagnitudeOfPropertyRange}(x_1, x_2, x_3, x_4) \leftrightarrow \\ &\quad \textbf{PropertyRange}(x_1) \wedge \\ &\quad \textbf{Scale}(x_2) \wedge \\ &\quad \textbf{ArithmeticNumber}(x_3) \wedge \\ &\quad \textbf{ArithmeticNumber}(x_4) \wedge \\ &\quad \exists y_1, y_2(\textit{LowerUpperOfPropertyRange}(x_1, y_1, y_2) \wedge \\ &\quad \textit{MagnitudeOfProperty}(y_1, x_3, x_2) \wedge \\ &\quad \textit{MagnitudeOfProperty}(y_2, x_4, x_2))\end{aligned}$$

EXAMPLE The expansion of the statement *LowerUpperMagnitudeOfPropertyRange*(**-40 to 85 Celsius**,**Celsius**,**-40**,**85**) is shown in the following diagram. Note: The designations of items **Temperature -40° C** and **Temperature 85° C** are included to make the diagram more readable, but are not defined by the expansion.

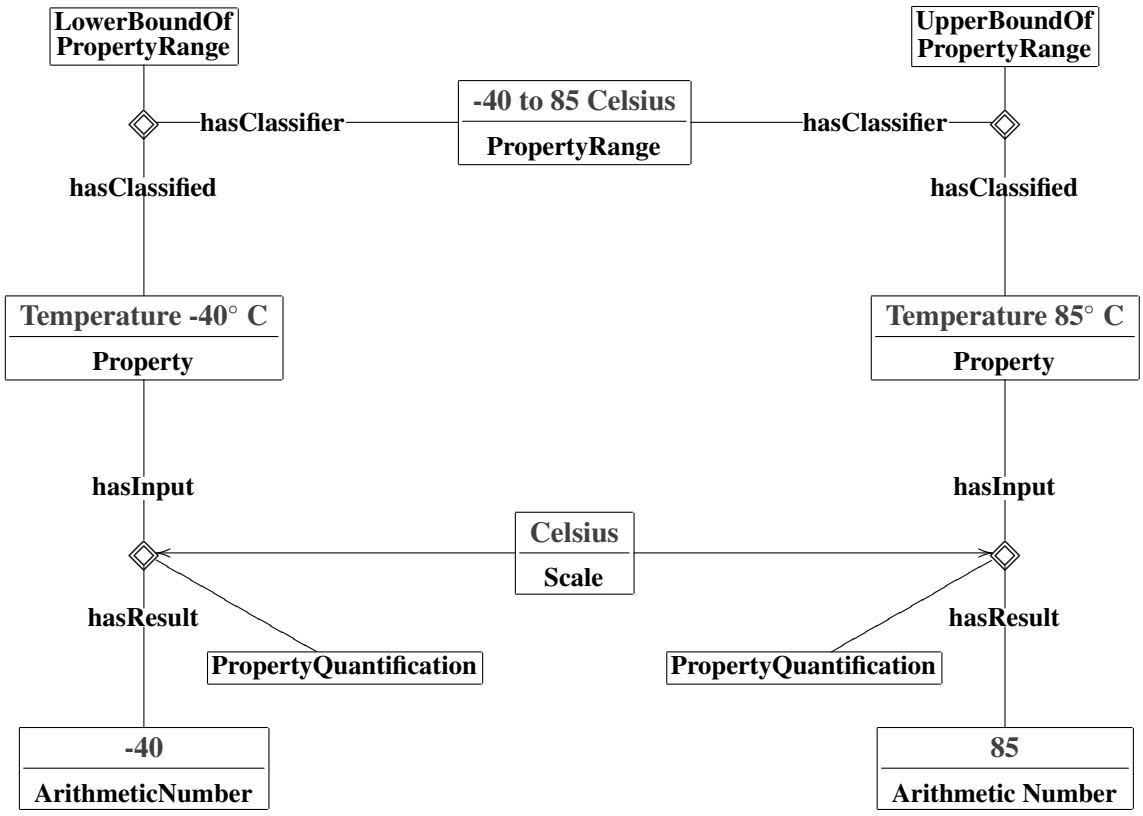

Figure 41 — Example template LowerUpperMagnitudeOfPropertyRange

7.5.21 Template PropertyRangeRestrictionOfClass

PropertyRangeRestrictionOfClass is a template for stating about a class of individuals that the magnitude of a property is restricted to a range of values.

PropertyRangeRestrictionOfClass(a, b, c) means that a is a class, b is a property relation, and c is a range of properties, and that every b property assignment to an a belongs to a subrelation of b for which the range is restricted to c.

Order	Role name	Role type
1	Class	**ClassOfIndividual**
2	Property	**ClassOfIndirectProperty**
3	Range	**PropertyRange**

PropertyRangeRestrictionOfClass$(x_1, x_2, x_3) \leftrightarrow$

ClassOfIndividual$(x_1) \wedge$

ClassOfIndirectProperty$(x_2) \wedge$

PropertyRange$(x_3) \wedge$

$\exists u($*ClassOfIndirectPropertyTriple*$(u, x_1, x_3) \wedge$

SpecializationAsEnd2UniversalRestriction$(u, x_2))$

EXAMPLE The statement *PropertyRangeRestrictionOfClass*(**330A-3-874**, **Working Pressure**, **0–300 psi**) (as might represent the statement, "pumps of the type 330A-3-874 have an allowed working pressure of 0–300 psi") expands to the following representation.

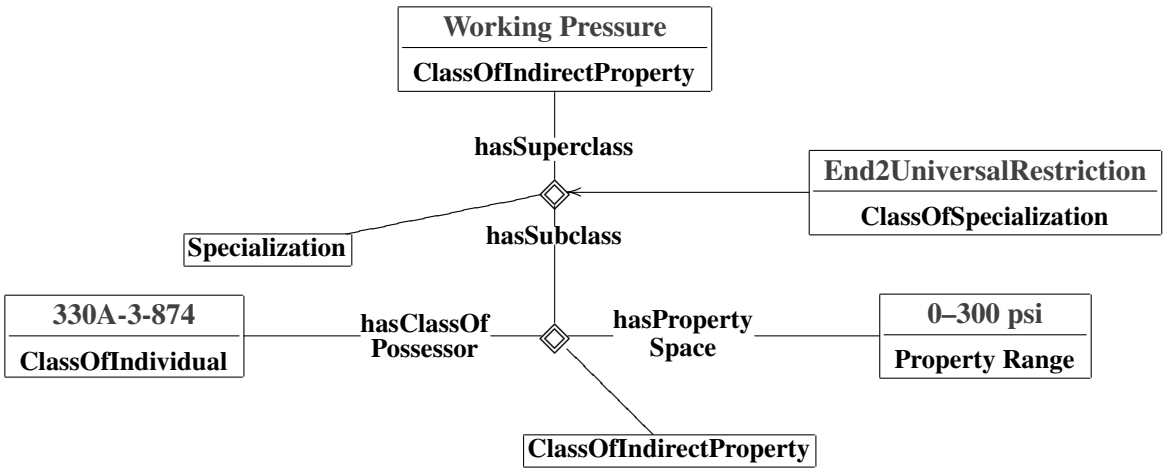

Figure 42 — Example template PropertyRangeRestrictionOfClass

7.5.22 Template PropertyRangeMagnitudeRestrictionOfClass

PropertyRangeMagnitudeRestrictionOfClass is a template for stating which range of values of a property can apply to a class of individuals.

PropertyRangeMagnitudeRestrictionOfClass(a, b, c, d, e) means that a is a class of individuals, b is a property relation, c is a scale, and d and e are real numbers, and that every b property of an a has a value in the d to e range, measured on the c scale.

Order	Role name	Role type
1	Class	**ClassOfIndividual**
2	Restricted Property	**ClassOfIndirectProperty**
3	Scale	**Scale**
4	Upper Bound	**ExpressReal**
5	Lower Bound	**ExpressReal**

PropertyRangeMagnitudeRestrictionOfClass$(x_1, x_2, x_3, x_4, x_5) \leftrightarrow$

ClassOfIndividual$(x_1)\wedge$

ClassOfIndirectProperty$(x_2)\wedge$

Scale$(x_3)\wedge$

ExpressReal$(x_4)\wedge$

ExpressReal$(x_5)\wedge$

$\exists u($*PropertyRangeRestrictionOfClass*$(x_1, x_2, u)\wedge$

$\exists y_1 \exists y_2($*IdentificationByNumber*$(x_4, y_1)\wedge$

IdentificationByNumber$(x_5, y_2)\wedge$

LowerUpperMagnitudeOfPropertyRange$(u, x_3, y_1, y_2)))$

EXAMPLE The statement *PropertyRangeMagnitudeRestrictionOfClass*(**330A-3-874, Working Pressure, pound-force per square inch, 0, 300**) has the following expansion.

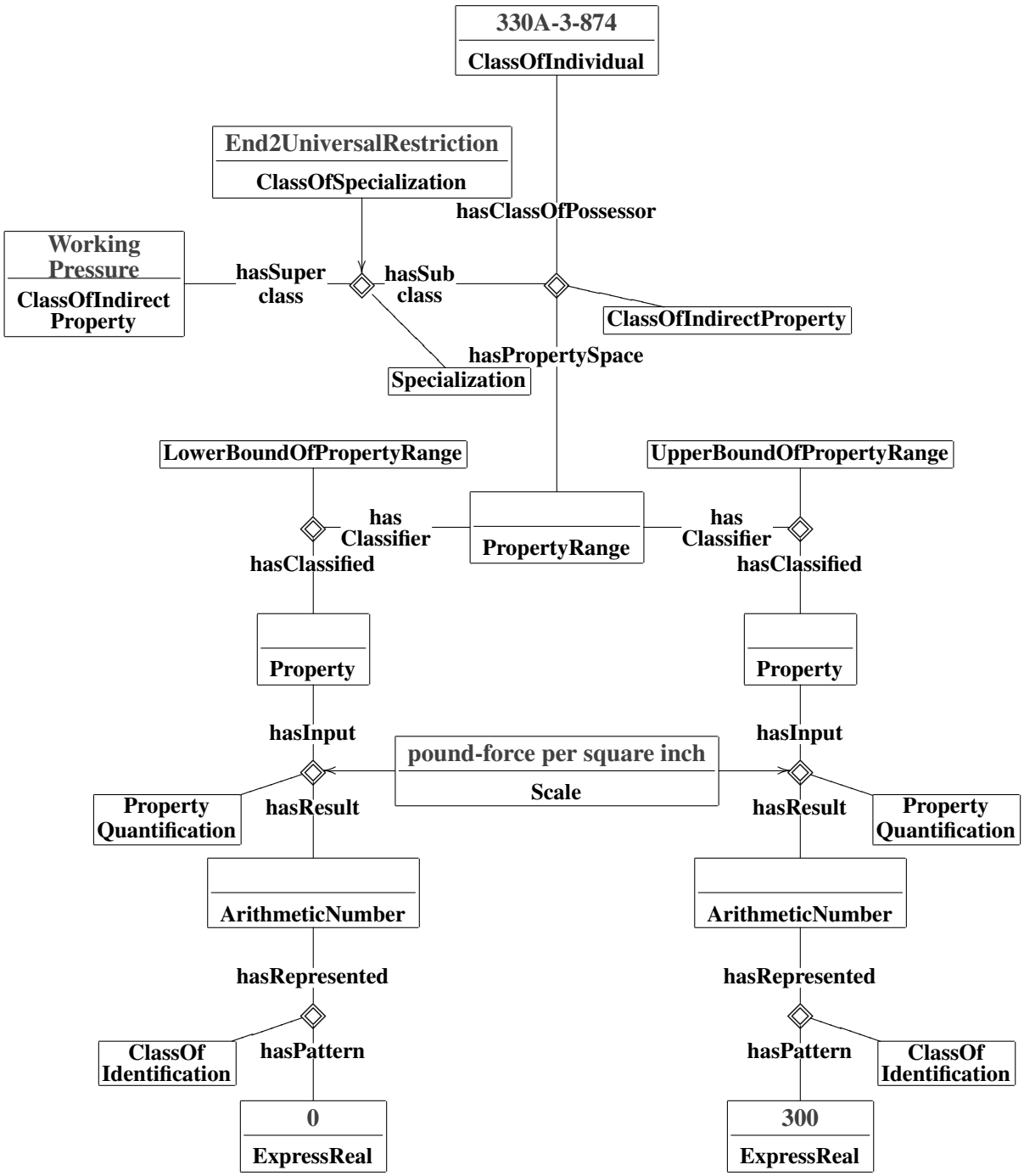

Figure 43 — Example template PropertyRangeMagnitudeRestrictionOfClass

7.5.23 Template SymbolOfScale

SymbolOfScale is a template for expressing that a symbol represents a scale.

SymbolOfScale(a, b, c) means that a is a scale, b is a string and that b is an identifier for a designated as a unit of measure symbol.

Order	Role name	Role type
1	Scale	**Scale**
2	Symbol	**ExpressString**

SymbolOfScale$(x_1, x_2) \leftrightarrow$

Scale$(x_1) \wedge$

ExpressString$(x_2) \wedge$

ClassifiedIdentification$(x_1, x_2,$ **UomSymbolAssignment**$)$

NOTE This template uses the reference item **UomSymbolAssignment**, a relation introduced for symbol assignment as a distinguished kind of naming. See 7.2.2.

EXAMPLE The expansion of the statement *SymbolOfScale*(**Celsius**, **DegrC**) is shown in the following diagram.

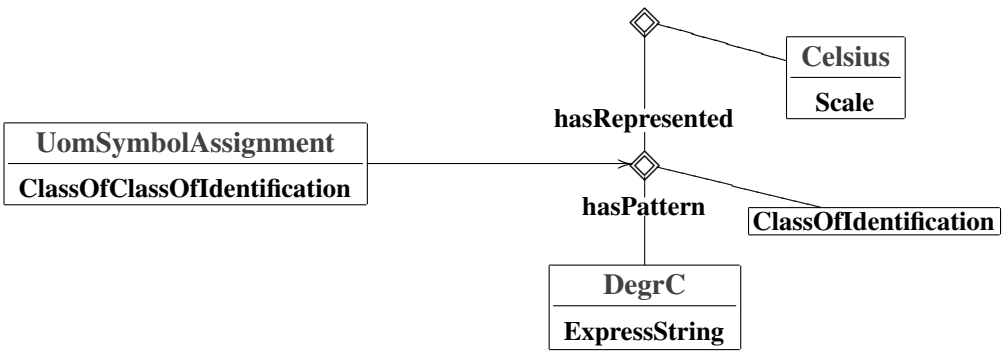

Figure 44 — Example template SymbolOfScale

7.5.24 Template DimensionUnitNumberRangeOfScale

DimensionUnitNumberRangeOfScale is a template for stating which dimension, number range and symbol applies to a scale.

DimensionUnitNumberRangeOfScale(a, b, c, d) means that a is a scale, b is a string, c is a property dimension, and d is a number range, that b is a unit of measure symbol for the scale and that c is the dimension and d the number range of the scale.

Order	Role name	Role type
1	Scale	**Scale**
2	Symbol	**ClassOfIdentification**
3	Dimension	**SinglePropertyDimension**
4	Number Range	**NumberRange**

DimensionUnitNumberRangeOfScale$(x_1, x_2, x_3, x_4) \leftrightarrow$

Scale$(x_1) \wedge$

ExpressString$(x_2) \wedge$

SinglePropertyDimension$(x_3) \wedge$

NumberRange$(x_4) \wedge$

SymbolOfScale$(x_1, x_2) \wedge$

ScaleTriple(x_1, x_4, x_3)

EXAMPLE The expansion of the statement *DimensionUnitNumberRangeOfScale*(**Celsius**, **DegrC**, **Temperature**, **[-273.1 to Infinity]** is shown in the following diagram.

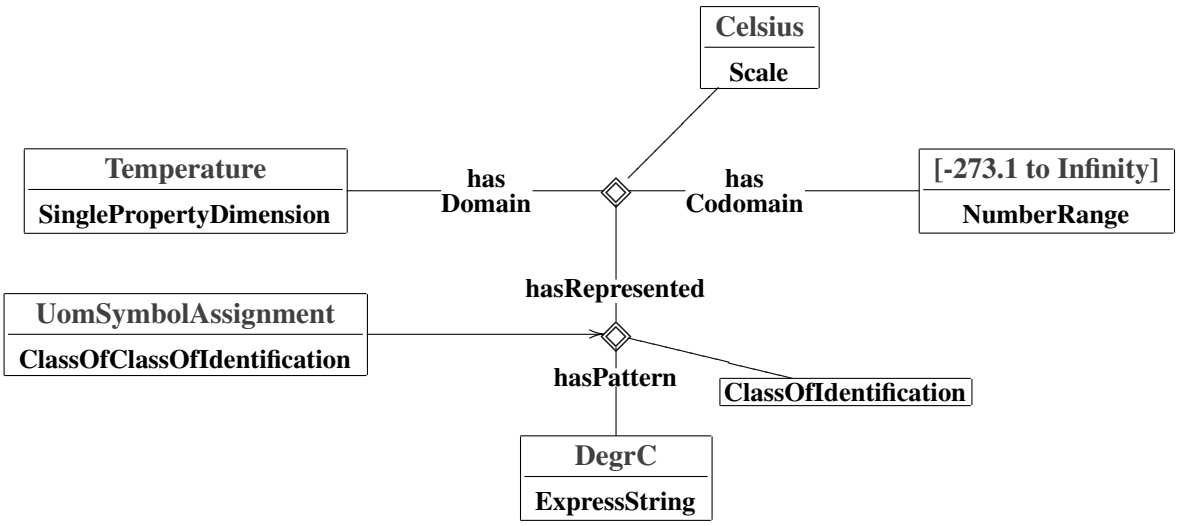

Figure 45 — Example template DimensionUnitNumberRangeOfScale

7.5.25 Template ClassInvolvementStatusBeginning

This template specializes template InvolvementStatusBeginning by restricting the thing involved in an activity to be a class.

ClassInvolvementStatusBeginning(a, b, c, d, e, f) means that a is a class and b is an activity and c is a type of involvement, d is an approval status and e is the approver and f is a point in time. a is involved in activity b and c is the type of involvement, the activity is approved, d is type of status of the approval and e is the approver, f is the start time of the activity.

Order	Role name	Role type
1	Involved class	**Class**
2	Involver activity	**Activity**
3	Involvement type	**ClassOfInvolvementByReference**
4	Status	**ClassOfApprovalByStatus**
5	Approver	**PossibleIndividual**
6	Start time	**RepresentationOfGregorianDateAndUtcTime**

ClassInvolvementStatusBeginning$(x_1, x_2, x_3, x_4, x_5) \leftrightarrow$

Class$(x_1) \wedge$

Activity$(x_2) \wedge$

ClassOfInvolvementByReference$(x_3) \wedge$

ClassOfApprovalByStatus$(x_4) \wedge$

PossibleIndividual$(x_5) \wedge$

RepresentationOfGregorianDateAndUtcTime$(x_6) \wedge$

InvolvementStatusBeginning$(x_1, u, x_3, x_4, x_5, x_6)$

7.5.26 Template ClassInvolvementSuccession

ClassInvolvementSuccession(a, b, c, d, e, f, g, h) means two classes a, b are involved in the same capacity in activity c. From time h, the involvement of a is succeeded by involvement of b; and approver g assigns status e to a's involvement, status f to b's involvement.

NOTE 1 In the phylosophy of ISO 15926 classes are eternal; e.g. can have no life-cycle. However some highly specialized classes can get to be "revised" ("to revise" is an Activity). This template is used to connect succeeding classes and qualify the succession with an activity, approver and time stamp.

Order	Role name	Role type
1	Predecessor class	**Class**
2	Successor class	**Class**
3	Involver activity	**Activity**
4	Involvement type	**ClassOfInvolvementByReference**
5	Status of predecessor	**ClassOfApprovalByStatus**
6	Status of successor	**ClassOfApprovalByStatus**
7	Status approver	**PossibleIndividual**
8	Start time	**RepresentationOfGregorianDateAndUtcTime**

$$\begin{aligned}
&\textit{ClassInvolvementSuccession}(x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8) \leftrightarrow \\
&\quad \textbf{Class}(x_1) \wedge \\
&\quad \textbf{Class}(x_2) \wedge \\
&\quad \textbf{Activity}(x_3) \wedge \\
&\quad \textbf{ClassOfInvolvementByReference}(x_4) \wedge \\
&\quad \textbf{ClassOfApprovalByStatus}(x_5) \wedge \\
&\quad \textbf{ClassOfApprovalByStatus}(x_6) \wedge \\
&\quad \textbf{PossibleIndividual}(x_7) \wedge \\
&\quad \textbf{RepresentationOfGregorianDateAndUtcTime}(x_8) \wedge \\
&\quad \exists u_1 \exists u_2 \exists u_3 (\\
&\qquad \textit{BeginningOfTemporalPart}(u_1, x_3, x_8) \wedge \\
&\qquad \textit{InvolvementByReferenceTriple}(u_2, x_1, u_1) \wedge \\
&\qquad \textit{InvolvementByReferenceTriple}(u_3, x_2, u_1) \wedge \\
&\qquad \textit{ClassificationTemplate}(u_2, x_4) \wedge \\
&\qquad \textit{ClassificationTemplate}(u_3, x_4) \wedge \\
&\qquad \textit{StatusApproval}(u_2, x_5, x_7) \wedge \\
&\qquad \textit{StatusApproval}(u_3, x_6, x_7) \wedge \\
&\qquad \textit{SuccessionOfInvolvementByReference}(u_2, u_3))
\end{aligned}$$

8 Templates as reference data

When recording a template in an ISO 15926 reference data library (RDL), it is provided with an ISO 15926 identifier. Storage in an RDL enables efficient sharing of ISO 15926 information representation patterns.

8.1 Template signatures and template axioms

An ISO 15926 RDL stores reference data expressed in the language of ISO 15926-2. The ISO 15926-2 and template languages used in this part of ISO 15926 to express template axioms include logical operators and quantifiers. The expressive power of these languages goes beyond what is covered in ISO 15926-2. Formal requirements on the representation of templates in an RDL are therefore limited to the *signature* aspect of templates.

An RDL representation of a template should include the template axiom as an annotation. No rigid requirement is imposed on the syntax in which axioms are expressed, beyond standard adequacy of expression as first-order formulae in accordance with the ISO 15926-2 language for this part of ISO 15926, as given in annex B.

8.2 Representation as class_of_multidimensional_object

When represented as reference data, a template should have the entity type **ClassOfMultidimensionalObject** (ISO 15926-2). Instances of templates have, accordingly, the entity type **MultidimensionalObject**.

A template signature as defined in this part of ISO 15926 is a restricted form of **ClassOfMultidimensionalObject**. As given in the EXPRESS specification, the attributes of this entity type are the following lists: `cardinalities`, `optional_element`, `parameters`, `parameter_position`, and `roles`. Of these, the optional attributes `cardinalities`, `parameters`, and `parameter_position` are not in use for template signatures. An RDL representation should leave these attributes empty.

For requirements on the introduction of roles in a template, see clause 5.2. In an RDL, each role should be represented by a reference item of type **RoleAndDomain**.

NOTE The entity type **RoleAndDomain** is defined in ISO 15926-2 as follows (p. 142): "A *role_and_domain* is a *class* that specifies the domain and role for an end of a *class_of_relationship*, or *class_of_multidimensional_object*."

For each role, the role *name* is given as the designation of the requisite RDL item of entity type **RoleAndDomain**. The *type* of the role, which serves to constrain the range of permitted values, is given by the constraints that apply to that RDL item.

The roles of a template signature should be recorded in the `roles` attribute, as an ordered list of reference items of type **RoleAndDomain**. A template as defined in this part of ISO 15926 has no optional roles, so the `optional_element` attribute of a template signature should be a list of length equal to that of the `roles` attribute, with the value TRUE in every element.

8.2.1 Roles constrained by RDL constructs

Each role for which type constraints are given by constructs using RDL items should be represented by a reference item, of entity type **RoleAndDomain**, that is subject to the requisite constraints.

8.2.2 Roles constrained by entity type only

This part of ISO 15926 permits the definition of template roles which are constrained by entity type only. In the signature of a base template, *every* role is constrained this way.

As defined in ISO 15926-2, every role in a **ClassOfMultidimensionalObject** is constrained by a class of type **RoleAndDomain**. There is no provision for constraining roles directly by entity type. Therefore, an RDL that represents templates defined according to this part of ISO 15926 should provide a representation pattern for entity types as **RoleAndDomain** reference classes. For each role that is constrained by entity type only, an RDL should represent that entity type as a **RoleAndDomain** reference item. The representation should follow the method of "punning" used in knowledge representation (cf. [9]).

NOTE This approach is also employed in ISO/TS 15926-4. In that part of ISO 15926, a selection of entity types is given as a set of classes, suitable for use as superclasses for standardized core reference classes.

Annex A
(normative)
Information object registration

A.1 Document identification

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

{ iso standard 15926 part(7) version(1) }

is assigned to this part of ISO 15926. The meaning of this value is defined in ISO/IEC 8824-1, and is described in ISO 10303-1.

Annex B
(normative)
Listing: ISO 15926-2 in first-order logic

B.1 General

This annex lists the set of FOL axioms representing the EXPRESS implementation of ISO 15926-2 used throughout this part of ISO 15926. The translation from the EXPRESS implementation of ISO 15926-2 to this FOL representation is explained in 4.1.

B.2 Universe axiom

$\forall x(\mathbf{Thing}(x))$

B.3 Subtype axioms

$\mathbf{AbstractObject}(x) \rightarrow \mathbf{Thing}(x)$
$\mathbf{Activity}(x) \rightarrow \mathbf{PossibleIndividual}(x)$
$\mathbf{ActualIndividual}(x) \rightarrow \mathbf{PossibleIndividual}(x)$
$\mathbf{Approval}(x) \rightarrow \mathbf{Relationship}(x)$
$\mathbf{ArithmeticNumber}(x) \rightarrow \mathbf{ClassOfClass}(x)$
$\mathbf{ArrangedIndividual}(x) \rightarrow \mathbf{PossibleIndividual}(x)$
$\mathbf{ArrangementOfIndividual}(x) \rightarrow \mathbf{CompositionOfIndividual}(x)$
$\mathbf{AssemblyOfIndividual}(x) \rightarrow \mathbf{ArrangementOfIndividual}(x)$
$\mathbf{Beginning}(x) \rightarrow \mathbf{TemporalBounding}(x)$
$\mathbf{BoundaryOfNumberSpace}(x) \rightarrow \mathbf{Specialization}(x)$
$\mathbf{BoundaryOfPropertySpace}(x) \rightarrow \mathbf{Specialization}(x)$
$\mathbf{Cardinality}(x) \rightarrow \mathbf{Class}(x)$
$\mathbf{CauseOfEvent}(x) \rightarrow \mathbf{Relationship}(x)$
$\mathbf{Class}(x) \rightarrow \mathbf{AbstractObject}(x)$
$\mathbf{ClassOfAbstractObject}(x) \rightarrow \mathbf{Class}(x)$
$\mathbf{ClassOfActivity}(x) \rightarrow \mathbf{ClassOfArrangedIndividual}(x)$
$\mathbf{ClassOfApproval}(x) \rightarrow \mathbf{ClassOfRelationship}(x)$
$\mathbf{ClassOfApprovalByStatus}(x) \rightarrow \mathbf{ClassOfRelationship}(x)$
$\mathbf{ClassOfArrangedIndividual}(x) \rightarrow \mathbf{ClassOfIndividual}(x)$
$\mathbf{ClassOfArrangementOfIndividual}(x) \rightarrow \mathbf{ClassOfCompositionOfIndividual}(x)$
$\mathbf{ClassOfAssemblyOfIndividual}(x) \rightarrow \mathbf{ClassOfArrangementOfIndividual}(x)$
$\mathbf{ClassOfAssertion}(x) \rightarrow \mathbf{ClassOfRelationship}(x)$
$\mathbf{ClassOfAtom}(x) \rightarrow \mathbf{ClassOfArrangedIndividual}(x)$
$\mathbf{ClassOfBiologicalMatter}(x) \rightarrow \mathbf{ClassOfArrangedIndividual}(x)$
$\mathbf{ClassOfCauseOfBeginningOfClassOfIndividual}(x) \rightarrow \mathbf{ClassOfRelationship}(x)$
$\mathbf{ClassOfCauseOfEndingOfClassOfIndividual}(x) \rightarrow \mathbf{ClassOfRelationship}(x)$
$\mathbf{ClassOfClass}(x) \rightarrow \mathbf{ClassOfAbstractObject}(x)$
$\mathbf{ClassOfClassOfComposition}(x) \rightarrow \mathbf{ClassOfClassOfRelationship}(x)$
$\mathbf{ClassOfClassOfDefinition}(x) \rightarrow \mathbf{ClassOfClassOfRepresentation}(x)$
$\mathbf{ClassOfClassOfDescription}(x) \rightarrow \mathbf{ClassOfClassOfRepresentation}(x)$
$\mathbf{ClassOfClassOfIdentification}(x) \rightarrow \mathbf{ClassOfClassOfRepresentation}(x)$
$\mathbf{ClassOfClassOfIndividual}(x) \rightarrow \mathbf{ClassOfClass}(x)$
$\mathbf{ClassOfClassOfInformationRepresentation}(x) \rightarrow \mathbf{ClassOfClassOfIndividual}(x)$
$\mathbf{ClassOfClassOfRelationship}(x) \rightarrow \mathbf{ClassOfClass}(x)$
$\mathbf{ClassOfClassOfRelationshipWithSignature}(x) \rightarrow \mathbf{ClassOfClassOfRelationship}(x)$
$\mathbf{ClassOfClassOfRelationshipWithSignature}(x) \rightarrow \mathbf{ClassOfRelationshipWithSignature}(x)$
$\mathbf{ClassOfClassOfRepresentation}(x) \rightarrow \mathbf{ClassOfClassOfRelationship}(x)$
$\mathbf{ClassOfClassOfRepresentationTranslation}(x) \rightarrow \mathbf{ClassOfClassOfRelationship}(x)$
$\mathbf{ClassOfClassOfResponsibilityForRepresentation}(x) \rightarrow \mathbf{ClassOfClassOfRelationship}(x)$
$\mathbf{ClassOfClassOfUsageOfRepresentation}(x) \rightarrow \mathbf{ClassOfClassOfRelationship}(x)$
$\mathbf{ClassOfClassification}(x) \rightarrow \mathbf{ClassOfRelationship}(x)$
$\mathbf{ClassOfCompositeMaterial}(x) \rightarrow \mathbf{ClassOfArrangedIndividual}(x)$
$\mathbf{ClassOfCompositionOfIndividual}(x) \rightarrow \mathbf{ClassOfRelationship}(x)$
$\mathbf{ClassOfCompound}(x) \rightarrow \mathbf{ClassOfArrangedIndividual}(x)$
$\mathbf{ClassOfConnectionOfIndividual}(x) \rightarrow \mathbf{ClassOfRelationship}(x)$
$\mathbf{ClassOfContainmentOfIndividual}(x) \rightarrow \mathbf{ClassOfRelativeLocation}(x)$
$\mathbf{ClassOfDefinition}(x) \rightarrow \mathbf{ClassOfRepresentationOfThing}(x)$
$\mathbf{ClassOfDescription}(x) \rightarrow \mathbf{ClassOfRepresentationOfThing}(x)$
$\mathbf{ClassOfDimensionForShape}(x) \rightarrow \mathbf{ClassOfClassOfRelationship}(x)$
$\mathbf{ClassOfDirectConnection}(x) \rightarrow \mathbf{ClassOfConnectionOfIndividual}(x)$

ClassOfEvent(x) → **ClassOfIndividual**(x)
ClassOfExpressInformationRepresentation(x) → **ClassOfInformationRepresentation**(x)
ClassOfFeature(x) → **ClassOfArrangedIndividual**(x)
ClassOfFeatureWholePart(x) → **ClassOfArrangementOfIndividual**(x)
ClassOfFunctionalMapping(x) → **ClassOfRelationship**(x)
ClassOfFunctionalObject(x) → **ClassOfArrangedIndividual**(x)
ClassOfIdentification(x) → **ClassOfRepresentationOfThing**(x)
ClassOfInanimatePhysicalObject(x) → **ClassOfArrangedIndividual**(x)
ClassOfIndirectConnection(x) → **ClassOfConnectionOfIndividual**(x)
ClassOfIndirectProperty(x) → **ClassOfRelationship**(x)
ClassOfIndividual(x) → **Class**(x)
ClassOfIndividualUsedInConnection(x) → **ClassOfRelationship**(x)
ClassOfInformationObject(x) → **ClassOfArrangedIndividual**(x)
ClassOfInformationPresentation(x) → **ClassOfArrangedIndividual**(x)
ClassOfInformationRepresentation(x) → **ClassOfArrangedIndividual**(x)
ClassOfIntendedRoleAndDomain(x) → **ClassOfRelationship**(x)
ClassOfInvolvementByReference(x) → **ClassOfRelationship**(x)
ClassOfIsomorphicFunctionalMapping(x) → **ClassOfFunctionalMapping**(x)
ClassOfLeftNamespace(x) → **ClassOfNamespace**(x)
ClassOfLifecycleStage(x) → **ClassOfRelationship**(x)
ClassOfMolecule(x) → **ClassOfArrangedIndividual**(x)
ClassOfMultidimensionalObject(x) → **ClassOfAbstractObject**(x)
ClassOfNamespace(x) → **ClassOfClassOfRelationship**(x)
ClassOfNumber(x) → **ClassOfClass**(x)
ClassOfOrganism(x) → **ClassOfArrangedIndividual**(x)
ClassOfOrganization(x) → **ClassOfArrangedIndividual**(x)
ClassOfParticipation(x) → **ClassOfCompositionOfIndividual**(x)
ClassOfParticulateMaterial(x) → **ClassOfArrangedIndividual**(x)
ClassOfPeriodInTime(x) → **ClassOfIndividual**(x)
ClassOfPerson(x) → **ClassOfOrganism**(x)
ClassOfPointInTime(x) → **ClassOfEvent**(x)
ClassOfPossibleRoleAndDomain(x) → **ClassOfRelationship**(x)
ClassOfProperty(x) → **ClassOfClassOfIndividual**(x)
ClassOfPropertySpace(x) → **ClassOfClass**(x)
ClassOfRecognition(x) → **ClassOfRelationship**(x)
ClassOfRelationship(x) → **ClassOfAbstractObject**(x)
ClassOfRelationshipWithRelatedEnd1(x) → **ClassOfRelationship**(x)
ClassOfRelationshipWithRelatedEnd2(x) → **ClassOfRelationship**(x)
ClassOfRelationshipWithSignature(x) → **ClassOfRelationship**(x)
ClassOfRelationshipWithSignature(x) → **Relationship**(x)
ClassOfRelativeLocation(x) → **ClassOfRelationship**(x)
ClassOfRepresentationOfThing(x) → **ClassOfRelationship**(x)
ClassOfRepresentationTranslation(x) → **ClassOfRelationship**(x)
ClassOfResponsibilityForRepresentation(x) → **ClassOfRelationship**(x)
ClassOfRightNamespace(x) → **ClassOfNamespace**(x)
ClassOfScale(x) → **ClassOfClassOfRelationship**(x)
ClassOfScaleConversion(x) → **ClassOfIsomorphicFunctionalMapping**(x)
ClassOfShape(x) → **PropertySpace**(x)
ClassOfShapeDimension(x) → **ClassOfClass**(x)
ClassOfSpecialization(x) → **ClassOfRelationship**(x)
ClassOfStatus(x) → **ClassOfClassOfIndividual**(x)
ClassOfSubAtomicParticle(x) → **ClassOfArrangedIndividual**(x)
ClassOfTemporalSequence(x) → **ClassOfRelationship**(x)
ClassOfTemporalWholePart(x) → **ClassOfCompositionOfIndividual**(x)
ClassOfUsageOfRepresentation(x) → **ClassOfRelationship**(x)
Classification(x) → **Relationship**(x)
ComparisonOfProperty(x) → **Relationship**(x)
CompositionOfIndividual(x) → **Relationship**(x)
ConnectionOfIndividual(x) → **Relationship**(x)
ContainmentOfIndividual(x) → **RelativeLocation**(x)
CoordinateSystem(x) → **MultidimensionalScale**(x)
CrystallineStructure(x) → **ClassOfArrangedIndividual**(x)
Definition(x) → **RepresentationOfThing**(x)
Description(x) → **RepresentationOfThing**(x)
DifferenceOfSetOfClass(x) → **FunctionalMapping**(x)

DimensionOfIndividual(x) → **ClassOfRelationship**(x)
DimensionOfShape(x) → **ClassOfClassOfRelationship**(x)
DirectConnection(x) → **ConnectionOfIndividual**(x)
DocumentDefinition(x) → **ClassOfClassOfInformationRepresentation**(x)
Ending(x) → **TemporalBounding**(x)
EnumeratedNumberSet(x) → **ClassOfNumber**(x)
EnumeratedNumberSet(x) → **EnumeratedSetOfClass**(x)
EnumeratedPropertySet(x) → **ClassOfProperty**(x)
EnumeratedPropertySet(x) → **EnumeratedSetOfClass**(x)
EnumeratedSetOfClass(x) → **ClassOfClass**(x)
Event(x) → **PossibleIndividual**(x)
ExpressBinary(x) → **ClassOfExpressInformationRepresentation**(x)
ExpressBoolean(x) → **ClassOfExpressInformationRepresentation**(x)
ExpressInteger(x) → **ClassOfExpressInformationRepresentation**(x)
ExpressLogical(x) → **ClassOfExpressInformationRepresentation**(x)
ExpressReal(x) → **ClassOfExpressInformationRepresentation**(x)
ExpressString(x) → **ClassOfExpressInformationRepresentation**(x)
FeatureWholePart(x) → **ArrangementOfIndividual**(x)
FunctionalMapping(x) → **Relationship**(x)
FunctionalPhysicalObject(x) → **PhysicalObject**(x)
Identification(x) → **RepresentationOfThing**(x)
IndirectConnection(x) → **ConnectionOfIndividual**(x)
IndirectProperty(x) → **Relationship**(x)
IndividualDimension(x) → **ClassOfIndividual**(x)
IndividualUsedInConnection(x) → **Relationship**(x)
IntegerNumber(x) → **ArithmeticNumber**(x)
IntendedRoleAndDomain(x) → **Relationship**(x)
IntersectionOfSetOfClass(x) → **FunctionalMapping**(x)
InvolvementByReference(x) → **Relationship**(x)
Language(x) → **ClassOfClassOfInformationRepresentation**(x)
LeftNamespace(x) → **Namespace**(x)
LifecycleStage(x) → **Relationship**(x)
LowerBoundOfNumberRange(x) → **Classification**(x)
LowerBoundOfPropertyRange(x) → **Classification**(x)
MaterializedPhysicalObject(x) → **PhysicalObject**(x)
MultidimensionalNumber(x) → **ArithmeticNumber**(x)
MultidimensionalNumber(x) → **MultidimensionalObject**(x)
MultidimensionalNumberSpace(x) → **MultidimensionalObject**(x)
MultidimensionalNumberSpace(x) → **NumberSpace**(x)
MultidimensionalObject(x) → **AbstractObject**(x)
MultidimensionalProperty(x) → **MultidimensionalObject**(x)
MultidimensionalProperty(x) → **Property**(x)
MultidimensionalPropertySpace(x) → **MultidimensionalObject**(x)
MultidimensionalPropertySpace(x) → **PropertySpace**(x)
MultidimensionalScale(x) → **MultidimensionalObject**(x)
MultidimensionalScale(x) → **Scale**(x)
Namespace(x) → **ClassOfArrangementOfIndividual**(x)
NumberRange(x) → **NumberSpace**(x)
NumberSpace(x) → **ClassOfNumber**(x)
OtherRelationship(x) → **Relationship**(x)
ParticipatingRoleAndDomain(x) → **ClassOfIndividual**(x)
ParticipatingRoleAndDomain(x) → **RoleAndDomain**(x)
Participation(x) → **CompositionOfIndividual**(x)
PeriodInTime(x) → **PossibleIndividual**(x)
Phase(x) → **ClassOfArrangedIndividual**(x)
PhysicalObject(x) → **PossibleIndividual**(x)
PointInTime(x) → **Event**(x)
PossibleIndividual(x) → **Thing**(x)
PossibleRoleAndDomain(x) → **Relationship**(x)
Property(x) → **ClassOfIndividual**(x)
PropertyForShapeDimension(x) → **ClassOfRelationship**(x)
PropertyQuantification(x) → **FunctionalMapping**(x)
PropertyRange(x) → **PropertySpace**(x)
PropertySpace(x) → **ClassOfProperty**(x)
PropertySpaceForClassOfShapeDimension(x) → **ClassOfClassOfRelationship**(x)

RealNumber(x) → **ArithmeticNumber**(x)
Recognition(x) → **Relationship**(x)
Relationship(x) → **AbstractObject**(x)
RelativeLocation(x) → **Relationship**(x)
RepresentationForm(x) → **ClassOfClassOfInformationRepresentation**(x)
RepresentationOfGregorianDateAndUtcTime(x) → **ClassOfInformationRepresentation**(x)
RepresentationOfThing(x) → **Relationship**(x)
ResponsibilityForRepresentation(x) → **Relationship**(x)
RightNamespace(x) → **Namespace**(x)
Role(x) → **RoleAndDomain**(x)
RoleAndDomain(x) → **Class**(x)
Scale(x) → **ClassOfIsomorphicFunctionalMapping**(x)
Shape(x) → **Property**(x)
ShapeDimension(x) → **ClassOfClassOfIndividual**(x)
SinglePropertyDimension(x) → **PropertySpace**(x)
SpatialLocation(x) → **PhysicalObject**(x)
Specialization(x) → **Relationship**(x)
SpecializationByDomain(x) → **Specialization**(x)
SpecializationByRole(x) → **Specialization**(x)
SpecializationOfIndividualDimensionFromProperty(x) → **Specialization**(x)
Status(x) → **ClassOfIndividual**(x)
Stream(x) → **PhysicalObject**(x)
TemporalBounding(x) → **CompositionOfIndividual**(x)
TemporalSequence(x) → **Relationship**(x)
TemporalWholePart(x) → **CompositionOfIndividual**(x)
UnionOfSetOfClass(x) → **FunctionalMapping**(x)
UpperBoundOfNumberRange(x) → **Classification**(x)
UpperBoundOfPropertyRange(x) → **Classification**(x)
UsageOfRepresentation(x) → **Relationship**(x)
WholeLifeIndividual(x) → **PossibleIndividual**(x)

B.4 Abstract axioms

AbstractObject(x) → (**Class**(x) ∨ **MultidimensionalObject**(x) ∨ **Relationship**(x))
ClassOfAbstractObject(x) →
(**ClassOfClass**(x) ∨ **ClassOfMultidimensionalObject**(x) ∨ **ClassOfRelationship**(x))
ClassOfConnectionOfIndividual(x) → (**ClassOfDirectConnection**(x) ∨ **ClassOfIndirectConnection**(x))
ClassOfExpressInformationRepresentation(x) → (**ExpressBinary**(x) ∨ **ExpressBoolean**(x) ∨
ExpressInteger(x) ∨ **ExpressLogical**(x) ∨ **ExpressReal**(x) ∨ **ExpressString**(x))
Namespace(x) → (**LeftNamespace**(x) ∨ **RightNamespace**(x))
Relationship(x) → (**Approval**(x) ∨ **CauseOfEvent**(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨
Classification(x) ∨ **ComparisonOfProperty**(x) ∨ **CompositionOfIndividual**(x) ∨ **ConnectionOfIndividual**(x) ∨
FunctionalMapping(x) ∨ **IndirectProperty**(x) ∨ **IndividualUsedInConnection**(x) ∨
IntendedRoleAndDomain(x) ∨ **InvolvementByReference**(x) ∨ **LifecycleStage**(x) ∨ **OtherRelationship**(x) ∨
PossibleRoleAndDomain(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨
ResponsibilityForRepresentation(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x))
TemporalBounding(x) → (**Beginning**(x) ∨ **Ending**(x))
Thing(x) → (**AbstractObject**(x) ∨ **PossibleIndividual**(x))

B.5 Disjoint axioms

¬(**IntegerNumber**(x) ∧ (**MultidimensionalNumber**(x)))
¬(**RealNumber**(x) ∧ (**IntegerNumber**(x) ∨ **MultidimensionalNumber**(x)))
¬(**AssemblyOfIndividual**(x) ∧ (**FeatureWholePart**(x)))
¬(**ClassOfIndividual**(x) ∧ (**ClassOfAbstractObject**(x)))
¬(**ClassOfAtom**(x) ∧ (**ClassOfBiologicalMatter**(x) ∨ **ClassOfCompositeMaterial**(x) ∨ **ClassOfCompound**(x) ∨
ClassOfFunctionalObject(x) ∨ **ClassOfInformationPresentation**(x) ∨
ClassOfInformationRepresentation(x) ∨ **ClassOfMolecule**(x) ∨ **ClassOfParticulateMaterial**(x) ∨
ClassOfSubAtomicParticle(x) ∨ **CrystallineStructure**(x) ∨ **Phase**(x)))
¬(**ClassOfBiologicalMatter**(x) ∧ (**ClassOfCompositeMaterial**(x) ∨ **ClassOfCompound**(x) ∨
ClassOfFunctionalObject(x) ∨ **ClassOfInformationPresentation**(x) ∨
ClassOfInformationRepresentation(x) ∨ **ClassOfMolecule**(x) ∨ **ClassOfParticulateMaterial**(x) ∨
ClassOfSubAtomicParticle(x) ∨ **CrystallineStructure**(x) ∨ **Phase**(x)))
¬(**ClassOfCompositeMaterial**(x) ∧ (**ClassOfCompound**(x) ∨ **ClassOfFunctionalObject**(x) ∨
ClassOfInformationPresentation(x) ∨ **ClassOfInformationRepresentation**(x) ∨ **ClassOfMolecule**(x) ∨
ClassOfParticulateMaterial(x) ∨ **ClassOfSubAtomicParticle**(x) ∨ **CrystallineStructure**(x) ∨ **Phase**(x)))
¬(**ClassOfCompound**(x) ∧ (**ClassOfFunctionalObject**(x) ∨ **ClassOfInformationPresentation**(x) ∨
ClassOfInformationRepresentation(x) ∨ **ClassOfMolecule**(x) ∨ **ClassOfParticulateMaterial**(x) ∨

ClassOfSubAtomicParticle(x) ∨ **CrystallineStructure**(x) ∨ **Phase**(x)))
¬(**ClassOfFunctionalObject**(x) ∧ (**ClassOfInformationPresentation**(x) ∨
ClassOfInformationRepresentation(x) ∨ **ClassOfMolecule**(x) ∨ **ClassOfParticulateMaterial**(x) ∨
ClassOfSubAtomicParticle(x) ∨ **CrystallineStructure**(x) ∨ **Phase**(x)))
¬(**ClassOfInformationPresentation**(x) ∧ (**ClassOfInformationRepresentation**(x) ∨ **ClassOfMolecule**(x) ∨
ClassOfParticulateMaterial(x) ∨ **ClassOfSubAtomicParticle**(x) ∨ **CrystallineStructure**(x) ∨ **Phase**(x)))
¬(**ClassOfInformationRepresentation**(x) ∧ (**ClassOfMolecule**(x) ∨ **ClassOfParticulateMaterial**(x) ∨
ClassOfSubAtomicParticle(x) ∨ **CrystallineStructure**(x) ∨ **Phase**(x)))
¬(**ClassOfMolecule**(x) ∧ (**ClassOfParticulateMaterial**(x) ∨ **ClassOfSubAtomicParticle**(x) ∨
CrystallineStructure(x) ∨ **Phase**(x)))
¬(**ClassOfParticulateMaterial**(x) ∧ (**ClassOfSubAtomicParticle**(x) ∨ **CrystallineStructure**(x) ∨ **Phase**(x)))
¬(**ClassOfSubAtomicParticle**(x) ∧ (**CrystallineStructure**(x) ∨ **Phase**(x)))
¬(**CrystallineStructure**(x) ∧ (**Phase**(x)))
¬(**ClassOfOrganism**(x) ∧ (**ClassOfInanimatePhysicalObject**(x)))
¬(**ClassOfAssemblyOfIndividual**(x) ∧ (**Namespace**(x)))
¬(**ClassOfFeatureWholePart**(x) ∧ (**ClassOfAssemblyOfIndividual**(x) ∨ **Namespace**(x)))
¬(**ArithmeticNumber**(x) ∧ (**ClassOfClassOfIndividual**(x) ∨ **ClassOfClassOfRelationship**(x) ∨
ClassOfNumber(x) ∨ **ClassOfPropertySpace**(x) ∨ **ClassOfShapeDimension**(x)))
¬(**ClassOfClassOfIndividual**(x) ∧ (**ClassOfClassOfRelationship**(x) ∨ **ClassOfNumber**(x) ∨
ClassOfPropertySpace(x) ∨ **ClassOfShapeDimension**(x)))
¬(**ClassOfClassOfRelationship**(x) ∧ (**ClassOfNumber**(x) ∨ **ClassOfPropertySpace**(x) ∨
ClassOfShapeDimension(x)))
¬(**ClassOfNumber**(x) ∧ (**ClassOfPropertySpace**(x) ∨ **ClassOfShapeDimension**(x)))
¬(**ClassOfPropertySpace**(x) ∧ (**ClassOfShapeDimension**(x)))
¬(**ClassOfClassOfInformationRepresentation**(x) ∧ (**ClassOfProperty**(x) ∨ **ClassOfStatus**(x) ∨
ShapeDimension(x)))
¬(**ClassOfProperty**(x) ∧ (**ClassOfStatus**(x) ∨ **ShapeDimension**(x)))
¬(**ClassOfStatus**(x) ∧ (**ShapeDimension**(x)))
¬(**Language**(x) ∧ (**DocumentDefinition**(x)))
¬(**RepresentationForm**(x) ∧ (**DocumentDefinition**(x) ∨ **Language**(x)))
¬(**ClassOfClassOfComposition**(x) ∧ (**ClassOfClassOfRelationshipWithSignature**(x) ∨
ClassOfClassOfRepresentation(x) ∨ **ClassOfClassOfRepresentationTranslation**(x) ∨
ClassOfClassOfResponsibilityForRepresentation(x) ∨ **ClassOfClassOfUsageOfRepresentation**(x) ∨
ClassOfDimensionForShape(x) ∨ **ClassOfNamespace**(x) ∨ **ClassOfScale**(x) ∨ **DimensionOfShape**(x) ∨
PropertySpaceForClassOfShapeDimension(x)))
¬(**ClassOfClassOfRelationshipWithSignature**(x) ∧ (**ClassOfClassOfRepresentation**(x) ∨
ClassOfClassOfRepresentationTranslation(x) ∨ **ClassOfClassOfResponsibilityForRepresentation**(x) ∨
ClassOfClassOfUsageOfRepresentation(x) ∨ **ClassOfDimensionForShape**(x) ∨ **ClassOfNamespace**(x) ∨
ClassOfScale(x) ∨ **DimensionOfShape**(x) ∨ **PropertySpaceForClassOfShapeDimension**(x)))
¬(**ClassOfClassOfRepresentation**(x) ∧ (**ClassOfClassOfRepresentationTranslation**(x) ∨
ClassOfClassOfResponsibilityForRepresentation(x) ∨ **ClassOfClassOfUsageOfRepresentation**(x) ∨
ClassOfDimensionForShape(x) ∨ **ClassOfNamespace**(x) ∨ **ClassOfScale**(x) ∨ **DimensionOfShape**(x) ∨
PropertySpaceForClassOfShapeDimension(x)))
¬(**ClassOfClassOfRepresentationTranslation**(x) ∧ (**ClassOfClassOfResponsibilityForRepresentation**(x) ∨
ClassOfClassOfUsageOfRepresentation(x) ∨ **ClassOfDimensionForShape**(x) ∨ **ClassOfNamespace**(x) ∨
ClassOfScale(x) ∨ **DimensionOfShape**(x) ∨ **PropertySpaceForClassOfShapeDimension**(x)))
¬(**ClassOfClassOfResponsibilityForRepresentation**(x) ∧ (**ClassOfClassOfUsageOfRepresentation**(x) ∨
ClassOfDimensionForShape(x) ∨ **ClassOfNamespace**(x) ∨ **ClassOfScale**(x) ∨ **DimensionOfShape**(x) ∨
PropertySpaceForClassOfShapeDimension(x)))
¬(**ClassOfClassOfUsageOfRepresentation**(x) ∧ (**ClassOfDimensionForShape**(x) ∨ **ClassOfNamespace**(x) ∨
ClassOfScale(x) ∨ **DimensionOfShape**(x) ∨ **PropertySpaceForClassOfShapeDimension**(x)))
¬(**ClassOfDimensionForShape**(x) ∧ (**ClassOfNamespace**(x) ∨ **ClassOfScale**(x) ∨ **DimensionOfShape**(x) ∨
PropertySpaceForClassOfShapeDimension(x)))
¬(**ClassOfNamespace**(x) ∧ (**ClassOfScale**(x) ∨ **DimensionOfShape**(x) ∨
PropertySpaceForClassOfShapeDimension(x)))
¬(**ClassOfScale**(x) ∧ (**DimensionOfShape**(x) ∨ **PropertySpaceForClassOfShapeDimension**(x)))
¬(**DimensionOfShape**(x) ∧ (**PropertySpaceForClassOfShapeDimension**(x)))
¬(**ClassOfArrangementOfIndividual**(x) ∧ (**ClassOfParticipation**(x) ∨ **ClassOfTemporalWholePart**(x)))
¬(**ClassOfTemporalWholePart**(x) ∧ (**ClassOfParticipation**(x)))
¬(**ClassOfDirectConnection**(x) ∧ (**ClassOfIndirectConnection**(x)))
¬(**ExpressBoolean**(x) ∧ (**ExpressBinary**(x)))
¬(**ExpressInteger**(x) ∧ (**ExpressBinary**(x) ∨ **ExpressBoolean**(x) ∨ **ExpressLogical**(x) ∨ **ExpressReal**(x)))
¬(**ExpressLogical**(x) ∧ (**ExpressBinary**(x) ∨ **ExpressBoolean**(x)))
¬(**ExpressReal**(x) ∧ (**ExpressBinary**(x) ∨ **ExpressBoolean**(x) ∨ **ExpressLogical**(x)))
¬(**ExpressString**(x) ∧ (**ExpressBinary**(x) ∨ **ExpressBoolean**(x) ∨ **ExpressInteger**(x) ∨ **ExpressLogical**(x) ∨

ExpressReal(x)))
¬(**ClassOfArrangedIndividual**(x) ∧ (**ClassOfPeriodInTime**(x) ∨ **IndividualDimension**(x) ∨ **Property**(x) ∨ **Status**(x)))
¬(**ClassOfEvent**(x) ∧ (**ClassOfArrangedIndividual**(x) ∨ **ClassOfPeriodInTime**(x) ∨ **IndividualDimension**(x) ∨ **Property**(x) ∨ **Status**(x)))
¬(**ClassOfPeriodInTime**(x) ∧ (**IndividualDimension**(x) ∨ **Property**(x) ∨ **Status**(x)))
¬(**IndividualDimension**(x) ∧ (**Property**(x) ∨ **Status**(x)))
¬(**Property**(x) ∧ (**Status**(x)))
¬(**ClassOfExpressInformationRepresentation**(x) ∧ (**RepresentationOfGregorianDateAndUtcTime**(x)))
¬(**Scale**(x) ∧ (**ClassOfScaleConversion**(x)))
¬(**ClassOfLeftNamespace**(x) ∧ (**ClassOfRightNamespace**(x)))
¬(**NumberSpace**(x) ∧ (**EnumeratedNumberSet**(x)))
¬(**PropertySpace**(x) ∧ (**EnumeratedPropertySet**(x)))
¬(**ClassOfApproval**(x) ∧ (**ClassOfApprovalByStatus**(x) ∨ **ClassOfAssertion**(x) ∨ **ClassOfCauseOfBeginningOfClassOfIndividual**(x) ∨ **ClassOfCauseOfEndingOfClassOfIndividual**(x) ∨ **ClassOfClassification**(x) ∨ **ClassOfCompositionOfIndividual**(x) ∨ **ClassOfConnectionOfIndividual**(x) ∨ **ClassOfFunctionalMapping**(x) ∨ **ClassOfIndirectProperty**(x) ∨ **ClassOfIndividualUsedInConnection**(x) ∨ **ClassOfIntendedRoleAndDomain**(x) ∨ **ClassOfInvolvementByReference**(x) ∨ **ClassOfLifecycleStage**(x) ∨ **ClassOfPossibleRoleAndDomain**(x) ∨ **ClassOfRecognition**(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨ **ClassOfRelativeLocation**(x) ∨ **ClassOfRepresentationOfThing**(x) ∨ **ClassOfRepresentationTranslation**(x) ∨ **ClassOfResponsibilityForRepresentation**(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨ **ClassOfUsageOfRepresentation**(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfApprovalByStatus**(x) ∧ (**ClassOfAssertion**(x) ∨ **ClassOfCauseOfBeginningOfClassOfIndividual**(x) ∨ **ClassOfCauseOfEndingOfClassOfIndividual**(x) ∨ **ClassOfClassification**(x) ∨ **ClassOfCompositionOfIndividual**(x) ∨ **ClassOfConnectionOfIndividual**(x) ∨ **ClassOfFunctionalMapping**(x) ∨ **ClassOfIndirectProperty**(x) ∨ **ClassOfIndividualUsedInConnection**(x) ∨ **ClassOfIntendedRoleAndDomain**(x) ∨ **ClassOfInvolvementByReference**(x) ∨ **ClassOfLifecycleStage**(x) ∨ **ClassOfPossibleRoleAndDomain**(x) ∨ **ClassOfRecognition**(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨ **ClassOfRelativeLocation**(x) ∨ **ClassOfRepresentationOfThing**(x) ∨ **ClassOfRepresentationTranslation**(x) ∨ **ClassOfResponsibilityForRepresentation**(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨ **ClassOfUsageOfRepresentation**(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfAssertion**(x) ∧ (**ClassOfPossibleRoleAndDomain**(x) ∨ **ClassOfRecognition**(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨ **ClassOfRelativeLocation**(x) ∨ **ClassOfRepresentationOfThing**(x) ∨ **ClassOfRepresentationTranslation**(x) ∨ **ClassOfResponsibilityForRepresentation**(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨ **ClassOfUsageOfRepresentation**(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfCauseOfBeginningOfClassOfIndividual**(x) ∧ (**ClassOfAssertion**(x) ∨ **ClassOfCauseOfEndingOfClassOfIndividual**(x) ∨ **ClassOfClassification**(x) ∨ **ClassOfCompositionOfIndividual**(x) ∨ **ClassOfConnectionOfIndividual**(x) ∨ **ClassOfFunctionalMapping**(x) ∨ **ClassOfIndirectProperty**(x) ∨ **ClassOfIndividualUsedInConnection**(x) ∨ **ClassOfIntendedRoleAndDomain**(x) ∨ **ClassOfInvolvementByReference**(x) ∨ **ClassOfLifecycleStage**(x) ∨ **ClassOfPossibleRoleAndDomain**(x) ∨ **ClassOfRecognition**(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨ **ClassOfRelativeLocation**(x) ∨ **ClassOfRepresentationOfThing**(x) ∨ **ClassOfRepresentationTranslation**(x) ∨ **ClassOfResponsibilityForRepresentation**(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨ **ClassOfUsageOfRepresentation**(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfCauseOfEndingOfClassOfIndividual**(x) ∧ (**ClassOfAssertion**(x) ∨ **ClassOfClassification**(x) ∨ **ClassOfCompositionOfIndividual**(x) ∨ **ClassOfConnectionOfIndividual**(x) ∨ **ClassOfFunctionalMapping**(x) ∨ **ClassOfIndirectProperty**(x) ∨ **ClassOfIndividualUsedInConnection**(x) ∨ **ClassOfIntendedRoleAndDomain**(x) ∨ **ClassOfInvolvementByReference**(x) ∨ **ClassOfLifecycleStage**(x) ∨ **ClassOfPossibleRoleAndDomain**(x) ∨ **ClassOfRecognition**(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨ **ClassOfRelativeLocation**(x) ∨ **ClassOfRepresentationOfThing**(x) ∨ **ClassOfRepresentationTranslation**(x) ∨ **ClassOfResponsibilityForRepresentation**(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨ **ClassOfUsageOfRepresentation**(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfClassification**(x) ∧ (**ClassOfAssertion**(x) ∨ **ClassOfCompositionOfIndividual**(x) ∨ **ClassOfConnectionOfIndividual**(x) ∨ **ClassOfFunctionalMapping**(x) ∨ **ClassOfIndirectProperty**(x) ∨ **ClassOfIndividualUsedInConnection**(x) ∨ **ClassOfIntendedRoleAndDomain**(x) ∨ **ClassOfInvolvementByReference**(x) ∨ **ClassOfLifecycleStage**(x) ∨ **ClassOfPossibleRoleAndDomain**(x) ∨ **ClassOfRecognition**(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨ **ClassOfRelativeLocation**(x) ∨ **ClassOfRepresentationOfThing**(x) ∨ **ClassOfRepresentationTranslation**(x) ∨ **ClassOfResponsibilityForRepresentation**(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨ **ClassOfUsageOfRepresentation**(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfCompositionOfIndividual**(x) ∧ (**ClassOfAssertion**(x) ∨ **ClassOfConnectionOfIndividual**(x) ∨ **ClassOfFunctionalMapping**(x) ∨ **ClassOfIndirectProperty**(x) ∨ **ClassOfIndividualUsedInConnection**(x) ∨ **ClassOfIntendedRoleAndDomain**(x) ∨ **ClassOfInvolvementByReference**(x) ∨ **ClassOfLifecycleStage**(x) ∨ **ClassOfPossibleRoleAndDomain**(x) ∨ **ClassOfRecognition**(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨

ClassOfRelativeLocation(x) ∨ **ClassOfRepresentationOfThing**(x) ∨ **ClassOfRepresentationTranslation**(x) ∨
ClassOfResponsibilityForRepresentation(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨
ClassOfUsageOfRepresentation(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfConnectionOfIndividual**(x) ∧ (**ClassOfAssertion**(x) ∨ **ClassOfFunctionalMapping**(x) ∨
ClassOfIndirectProperty(x) ∨ **ClassOfIndividualUsedInConnection**(x) ∨
ClassOfIntendedRoleAndDomain(x) ∨ **ClassOfInvolvementByReference**(x) ∨ **ClassOfLifecycleStage**(x) ∨
ClassOfPossibleRoleAndDomain(x) ∨ **ClassOfRecognition**(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨
ClassOfRelativeLocation(x) ∨ **ClassOfRepresentationOfThing**(x) ∨ **ClassOfRepresentationTranslation**(x) ∨
ClassOfResponsibilityForRepresentation(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨
ClassOfUsageOfRepresentation(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfFunctionalMapping**(x) ∧ (**ClassOfAssertion**(x) ∨ **ClassOfIndirectProperty**(x) ∨
ClassOfIndividualUsedInConnection(x) ∨ **ClassOfIntendedRoleAndDomain**(x) ∨
ClassOfInvolvementByReference(x) ∨ **ClassOfLifecycleStage**(x) ∨ **ClassOfPossibleRoleAndDomain**(x) ∨
ClassOfRecognition(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨ **ClassOfRelativeLocation**(x) ∨
ClassOfRepresentationOfThing(x) ∨ **ClassOfRepresentationTranslation**(x) ∨
ClassOfResponsibilityForRepresentation(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨
ClassOfUsageOfRepresentation(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfIndirectProperty**(x) ∧ (**ClassOfAssertion**(x) ∨ **ClassOfIndividualUsedInConnection**(x) ∨
ClassOfIntendedRoleAndDomain(x) ∨ **ClassOfInvolvementByReference**(x) ∨ **ClassOfLifecycleStage**(x) ∨
ClassOfPossibleRoleAndDomain(x) ∨ **ClassOfRecognition**(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨
ClassOfRelativeLocation(x) ∨ **ClassOfRepresentationOfThing**(x) ∨ **ClassOfRepresentationTranslation**(x) ∨
ClassOfResponsibilityForRepresentation(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨
ClassOfUsageOfRepresentation(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfIndividualUsedInConnection**(x) ∧ (**ClassOfAssertion**(x) ∨ **ClassOfIntendedRoleAndDomain**(x) ∨
ClassOfInvolvementByReference(x) ∨ **ClassOfLifecycleStage**(x) ∨ **ClassOfPossibleRoleAndDomain**(x) ∨
ClassOfRecognition(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨ **ClassOfRelativeLocation**(x) ∨
ClassOfRepresentationOfThing(x) ∨ **ClassOfRepresentationTranslation**(x) ∨
ClassOfResponsibilityForRepresentation(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨
ClassOfUsageOfRepresentation(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfIntendedRoleAndDomain**(x) ∧ (**ClassOfAssertion**(x) ∨ **ClassOfInvolvementByReference**(x) ∨
ClassOfLifecycleStage(x) ∨ **ClassOfPossibleRoleAndDomain**(x) ∨ **ClassOfRecognition**(x) ∨
ClassOfRelationshipWithSignature(x) ∨ **ClassOfRelativeLocation**(x) ∨ **ClassOfRepresentationOfThing**(x) ∨
ClassOfRepresentationTranslation(x) ∨ **ClassOfResponsibilityForRepresentation**(x) ∨
ClassOfSpecialization(x) ∨ **ClassOfTemporalSequence**(x) ∨ **ClassOfUsageOfRepresentation**(x) ∨
DimensionOfIndividual(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfInvolvementByReference**(x) ∧ (**ClassOfAssertion**(x) ∨ **ClassOfLifecycleStage**(x) ∨
ClassOfPossibleRoleAndDomain(x) ∨ **ClassOfRecognition**(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨
ClassOfRelativeLocation(x) ∨ **ClassOfRepresentationOfThing**(x) ∨ **ClassOfRepresentationTranslation**(x) ∨
ClassOfResponsibilityForRepresentation(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨
ClassOfUsageOfRepresentation(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfLifecycleStage**(x) ∧ (**ClassOfAssertion**(x) ∨ **ClassOfPossibleRoleAndDomain**(x) ∨
ClassOfRecognition(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨ **ClassOfRelativeLocation**(x) ∨
ClassOfRepresentationOfThing(x) ∨ **ClassOfRepresentationTranslation**(x) ∨
ClassOfResponsibilityForRepresentation(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨
ClassOfUsageOfRepresentation(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfPossibleRoleAndDomain**(x) ∧ (**ClassOfRecognition**(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨
ClassOfRelativeLocation(x) ∨ **ClassOfRepresentationOfThing**(x) ∨ **ClassOfRepresentationTranslation**(x) ∨
ClassOfResponsibilityForRepresentation(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨
ClassOfUsageOfRepresentation(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfRecognition**(x) ∧ (**ClassOfRelationshipWithSignature**(x) ∨ **ClassOfRelativeLocation**(x) ∨
ClassOfRepresentationOfThing(x) ∨ **ClassOfRepresentationTranslation**(x) ∨
ClassOfResponsibilityForRepresentation(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨
ClassOfUsageOfRepresentation(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfRelationshipWithSignature**(x) ∧ (**ClassOfRelativeLocation**(x) ∨
ClassOfRepresentationOfThing(x) ∨ **ClassOfRepresentationTranslation**(x) ∨
ClassOfResponsibilityForRepresentation(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨
ClassOfUsageOfRepresentation(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfRelativeLocation**(x) ∧ (**ClassOfRepresentationOfThing**(x) ∨ **ClassOfRepresentationTranslation**(x) ∨
ClassOfResponsibilityForRepresentation(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨
ClassOfUsageOfRepresentation(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfRepresentationOfThing**(x) ∧ (**ClassOfRepresentationTranslation**(x) ∨
ClassOfResponsibilityForRepresentation(x) ∨ **ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨
ClassOfUsageOfRepresentation(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfRepresentationTranslation**(x) ∧ (**ClassOfResponsibilityForRepresentation**(x) ∨
ClassOfSpecialization(x) ∨ **ClassOfTemporalSequence**(x) ∨ **ClassOfUsageOfRepresentation**(x) ∨

DimensionOfIndividual(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfResponsibilityForRepresentation**(x) ∧ (**ClassOfSpecialization**(x) ∨ **ClassOfTemporalSequence**(x) ∨ **ClassOfUsageOfRepresentation**(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfSpecialization**(x) ∧ (**ClassOfTemporalSequence**(x) ∨ **ClassOfUsageOfRepresentation**(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfTemporalSequence**(x) ∧ (**ClassOfUsageOfRepresentation**(x) ∨ **DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**ClassOfUsageOfRepresentation**(x) ∧ (**DimensionOfIndividual**(x) ∨ **PropertyForShapeDimension**(x)))
¬(**DimensionOfIndividual**(x) ∧ (**PropertyForShapeDimension**(x)))
¬(**ClassOfRelationshipWithRelatedEnd1**(x) ∧ (**ClassOfRelationshipWithRelatedEnd2**(x)))
¬(**ArrangementOfIndividual**(x) ∧ (**Participation**(x) ∨ **TemporalBounding**(x) ∨ **TemporalWholePart**(x)))
¬(**Participation**(x) ∧ (**TemporalBounding**(x)))
¬(**TemporalWholePart**(x) ∧ (**Participation**(x) ∨ **TemporalBounding**(x)))
¬(**DirectConnection**(x) ∧ (**IndirectConnection**(x)))
¬(**MultidimensionalNumber**(x) ∧ (**MultidimensionalNumberSpace**(x) ∨ **MultidimensionalProperty**(x) ∨ **MultidimensionalScale**(x)))
¬(**MultidimensionalNumberSpace**(x) ∧ (**MultidimensionalScale**(x)))
¬(**MultidimensionalProperty**(x) ∧ (**MultidimensionalNumberSpace**(x) ∨ **MultidimensionalScale**(x)))
¬(**MultidimensionalPropertySpace**(x) ∧ (**MultidimensionalNumber**(x) ∨ **MultidimensionalNumberSpace**(x) ∨ **MultidimensionalProperty**(x) ∨ **MultidimensionalScale**(x)))
¬(**RightNamespace**(x) ∧ (**LeftNamespace**(x)))
¬(**NumberRange**(x) ∧ (**MultidimensionalNumberSpace**(x)))
¬(**Approval**(x) ∧ (**CauseOfEvent**(x) ∨ **ClassOfRelationshipWithSignature**(x) ∨ **Classification**(x) ∨ **ComparisonOfProperty**(x) ∨ **CompositionOfIndividual**(x) ∨ **ConnectionOfIndividual**(x) ∨ **FunctionalMapping**(x) ∨ **IndirectProperty**(x) ∨ **IndividualUsedInConnection**(x) ∨ **IntendedRoleAndDomain**(x) ∨ **InvolvementByReference**(x) ∨ **LifecycleStage**(x) ∨ **OtherRelationship**(x) ∨ **PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**CauseOfEvent**(x) ∧ (**ClassOfRelationshipWithSignature**(x) ∨ **Classification**(x) ∨ **ComparisonOfProperty**(x) ∨ **CompositionOfIndividual**(x) ∨ **ConnectionOfIndividual**(x) ∨ **FunctionalMapping**(x) ∨ **IndirectProperty**(x) ∨ **IndividualUsedInConnection**(x) ∨ **IntendedRoleAndDomain**(x) ∨ **InvolvementByReference**(x) ∨ **LifecycleStage**(x) ∨ **OtherRelationship**(x) ∨ **PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**ClassOfRelationshipWithSignature**(x) ∧ (**Classification**(x) ∨ **ComparisonOfProperty**(x) ∨ **CompositionOfIndividual**(x) ∨ **ConnectionOfIndividual**(x) ∨ **FunctionalMapping**(x) ∨ **IndirectProperty**(x) ∨ **IndividualUsedInConnection**(x) ∨ **IntendedRoleAndDomain**(x) ∨ **InvolvementByReference**(x) ∨ **LifecycleStage**(x) ∨ **OtherRelationship**(x) ∨ **PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**Classification**(x) ∧ (**ComparisonOfProperty**(x) ∨ **CompositionOfIndividual**(x) ∨ **ConnectionOfIndividual**(x) ∨ **FunctionalMapping**(x) ∨ **IndirectProperty**(x) ∨ **IndividualUsedInConnection**(x) ∨ **IntendedRoleAndDomain**(x) ∨ **InvolvementByReference**(x) ∨ **LifecycleStage**(x) ∨ **OtherRelationship**(x) ∨ **PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**ComparisonOfProperty**(x) ∧ (**CompositionOfIndividual**(x) ∨ **ConnectionOfIndividual**(x) ∨ **FunctionalMapping**(x) ∨ **IndirectProperty**(x) ∨ **IndividualUsedInConnection**(x) ∨ **IntendedRoleAndDomain**(x) ∨ **InvolvementByReference**(x) ∨ **LifecycleStage**(x) ∨ **OtherRelationship**(x) ∨ **PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**CompositionOfIndividual**(x) ∧ (**ConnectionOfIndividual**(x) ∨ **FunctionalMapping**(x) ∨ **IndirectProperty**(x) ∨ **IndividualUsedInConnection**(x) ∨ **IntendedRoleAndDomain**(x) ∨ **InvolvementByReference**(x) ∨ **LifecycleStage**(x) ∨ **OtherRelationship**(x) ∨ **PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**ConnectionOfIndividual**(x) ∧ (**FunctionalMapping**(x) ∨ **IndirectProperty**(x) ∨ **IndividualUsedInConnection**(x) ∨ **IntendedRoleAndDomain**(x) ∨ **InvolvementByReference**(x) ∨ **LifecycleStage**(x) ∨ **OtherRelationship**(x) ∨ **PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**FunctionalMapping**(x) ∧ (**IndirectProperty**(x) ∨ **IndividualUsedInConnection**(x) ∨ **IntendedRoleAndDomain**(x) ∨ **InvolvementByReference**(x) ∨ **LifecycleStage**(x) ∨ **OtherRelationship**(x) ∨ **PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**IndirectProperty**(x) ∧ (**IndividualUsedInConnection**(x) ∨ **IntendedRoleAndDomain**(x) ∨

InvolvementByReference(x) ∨ **LifecycleStage**(x) ∨ **OtherRelationship**(x) ∨ **PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**IndividualUsedInConnection**(x) ∧ (**IntendedRoleAndDomain**(x) ∨ **InvolvementByReference**(x) ∨ **LifecycleStage**(x) ∨ **OtherRelationship**(x) ∨ **PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**IntendedRoleAndDomain**(x) ∧ (**InvolvementByReference**(x) ∨ **LifecycleStage**(x) ∨ **OtherRelationship**(x) ∨ **PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**InvolvementByReference**(x) ∧ (**LifecycleStage**(x) ∨ **OtherRelationship**(x) ∨ **PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**LifecycleStage**(x) ∧ (**OtherRelationship**(x) ∨ **PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**OtherRelationship**(x) ∧ (**PossibleRoleAndDomain**(x) ∨ **Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**PossibleRoleAndDomain**(x) ∧ (**Recognition**(x) ∨ **RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**Recognition**(x) ∧ (**RelativeLocation**(x) ∨ **RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**RelativeLocation**(x) ∧ (**RepresentationOfThing**(x) ∨ **ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**RepresentationOfThing**(x) ∧ (**ResponsibilityForRepresentation**(x) ∨ **Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**ResponsibilityForRepresentation**(x) ∧ (**Specialization**(x) ∨ **TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**Specialization**(x) ∧ (**TemporalSequence**(x) ∨ **UsageOfRepresentation**(x)))
¬(**TemporalSequence**(x) ∧ (**UsageOfRepresentation**(x)))
¬(**BoundaryOfNumberSpace**(x) ∧ (**BoundaryOfPropertySpace**(x) ∨ **SpecializationByDomain**(x) ∨ **SpecializationByRole**(x) ∨ **SpecializationOfIndividualDimensionFromProperty**(x)))
¬(**BoundaryOfPropertySpace**(x) ∧ (**SpecializationByDomain**(x) ∨ **SpecializationByRole**(x) ∨ **SpecializationOfIndividualDimensionFromProperty**(x)))
¬(**SpecializationByDomain**(x) ∧ (**SpecializationByRole**(x) ∨ **SpecializationOfIndividualDimensionFromProperty**(x)))
¬(**SpecializationByRole**(x) ∧ (**SpecializationOfIndividualDimensionFromProperty**(x)))
¬(**Ending**(x) ∧ (**Beginning**(x)))
¬(**PossibleIndividual**(x) ∧ (**AbstractObject**(x)))

B.6 Role axioms

hasApproved(x, y) → (**Approval**(x))
Approval(x) ∧ **hasApproved**(x, y) → **Relationship**(y)
Approval(x) → ∃y(**hasApproved**(x, y))
Approval(x) ∧ **hasApproved**(x, y) ∧ **hasApproved**(x, z) → $y = z$
hasApprover(x, y) → (**Approval**(x))
Approval(x) ∧ **hasApprover**(x, y) → **PossibleIndividual**(y)
Approval(x) → ∃y(**hasApprover**(x, y))
Approval(x) ∧ **hasApprover**(x, y) ∧ **hasApprover**(x, z) → $y = z$
hasCardinalities(x, y) → (**ClassOfMultidimensionalObject**(x))
ClassOfMultidimensionalObject(x) ∧ **hasCardinalities**(x, y) ∧ **hasCardinalities**(x, z) → $y = z$
hasCaused(x, y) → (**CauseOfEvent**(x))
CauseOfEvent(x) ∧ **hasCaused**(x, y) → **Event**(y)
CauseOfEvent(x) → ∃y(**hasCaused**(x, y))
CauseOfEvent(x) ∧ **hasCaused**(x, y) ∧ **hasCaused**(x, z) → $y = z$
hasCauser(x, y) → (**CauseOfEvent**(x))
CauseOfEvent(x) ∧ **hasCauser**(x, y) → **Activity**(y)
CauseOfEvent(x) → ∃y(**hasCauser**(x, y))
CauseOfEvent(x) ∧ **hasCauser**(x, y) ∧ **hasCauser**(x, z) → $y = z$
hasClassOfApproved(x, y) → (**ClassOfApproval**(x))
ClassOfApproval(x) ∧ **hasClassOfApproved**(x, y) → **ClassOfRelationship**(y)
ClassOfApproval(x) → ∃y(**hasClassOfApproved**(x, y))
ClassOfApproval(x) ∧ **hasClassOfApproved**(x, y) ∧ **hasClassOfApproved**(x, z) → $y = z$
hasClassOfApprover(x, y) → (**ClassOfApproval**(x))

ClassOfApproval$(x) \wedge$ **hasClassOfApprover**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfApproval$(x) \rightarrow \exists y($**hasClassOfApprover**$(x, y))$
ClassOfApproval$(x) \wedge$ **hasClassOfApprover**$(x, y) \wedge$ **hasClassOfApprover**$(x, z) \rightarrow y = z$
hasClassOfBegun$(x, y) \rightarrow$ (**ClassOfCauseOfBeginningOfClassOfIndividual**(x))
ClassOfCauseOfBeginningOfClassOfIndividual$(x) \wedge$ **hasClassOfBegun**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfCauseOfBeginningOfClassOfIndividual$(x) \rightarrow \exists y($**hasClassOfBegun**$(x, y))$
ClassOfCauseOfBeginningOfClassOfIndividual$(x) \wedge$ **hasClassOfBegun**$(x, y) \wedge$ **hasClassOfBegun**$(x, z) \rightarrow y = z$
hasClassOfCauser$(x, y) \rightarrow$
(**ClassOfCauseOfBeginningOfClassOfIndividual**$(x) \vee$ **ClassOfCauseOfEndingOfClassOfIndividual**(x))
ClassOfCauseOfBeginningOfClassOfIndividual$(x) \wedge$ **hasClassOfCauser**$(x, y) \rightarrow$ **ClassOfActivity**(y)
ClassOfCauseOfBeginningOfClassOfIndividual$(x) \rightarrow \exists y($**hasClassOfCauser**$(x, y))$
ClassOfCauseOfBeginningOfClassOfIndividual$(x) \wedge$ **hasClassOfCauser**$(x, y) \wedge$ **hasClassOfCauser**$(x, z) \rightarrow y = z$
ClassOfCauseOfEndingOfClassOfIndividual$(x) \wedge$ **hasClassOfCauser**$(x, y) \rightarrow$ **ClassOfActivity**(y)
ClassOfCauseOfEndingOfClassOfIndividual$(x) \rightarrow \exists y($**hasClassOfCauser**$(x, y))$
ClassOfCauseOfEndingOfClassOfIndividual$(x) \wedge$ **hasClassOfCauser**$(x, y) \wedge$ **hasClassOfCauser**$(x, z) \rightarrow y = z$
hasClassOfClassOfControlled$(x, y) \rightarrow$ (**ClassOfClassOfResponsibilityForRepresentation**(x))
ClassOfClassOfResponsibilityForRepresentation$(x) \wedge$ **hasClassOfClassOfControlled**$(x, y) \rightarrow$
ClassOfClassOfRepresentation(y)
ClassOfClassOfResponsibilityForRepresentation$(x) \rightarrow \exists y($**hasClassOfClassOfControlled**$(x, y))$
ClassOfClassOfResponsibilityForRepresentation$(x) \wedge$ **hasClassOfClassOfControlled**$(x, y) \wedge$
hasClassOfClassOfControlled$(x, z) \rightarrow y = z$
hasClassOfClassOfPart$(x, y) \rightarrow$ (**ClassOfClassOfComposition**(x))
ClassOfClassOfComposition$(x) \wedge$ **hasClassOfClassOfPart**$(x, y) \rightarrow$ **ClassOfClassOfIndividual**(y)
ClassOfClassOfComposition$(x) \rightarrow \exists y($**hasClassOfClassOfPart**$(x, y))$
ClassOfClassOfComposition$(x) \wedge$ **hasClassOfClassOfPart**$(x, y) \wedge$ **hasClassOfClassOfPart**$(x, z) \rightarrow y = z$
hasClassOfClassOfUsed$(x, y) \rightarrow$ (**ClassOfClassOfUsageOfRepresentation**(x))
ClassOfClassOfUsageOfRepresentation$(x) \wedge$ **hasClassOfClassOfUsed**$(x, y) \rightarrow$
ClassOfClassOfRepresentation(y)
ClassOfClassOfUsageOfRepresentation$(x) \rightarrow \exists y($**hasClassOfClassOfUsed**$(x, y))$
ClassOfClassOfUsageOfRepresentation$(x) \wedge$ **hasClassOfClassOfUsed**$(x, y) \wedge$ **hasClassOfClassOfUsed**$(x, z) \rightarrow$
$y = z$
hasClassOfClassOfWhole$(x, y) \rightarrow$ (**ClassOfClassOfComposition**$(x) \vee$ **ClassOfNamespace**(x))
ClassOfClassOfComposition$(x) \wedge$ **hasClassOfClassOfWhole**$(x, y) \rightarrow$ **ClassOfClassOfIndividual**(y)
ClassOfClassOfComposition$(x) \rightarrow \exists y($**hasClassOfClassOfWhole**$(x, y))$
ClassOfClassOfComposition$(x) \wedge$ **hasClassOfClassOfWhole**$(x, y) \wedge$ **hasClassOfClassOfWhole**$(x, z) \rightarrow y = z$
ClassOfNamespace$(x) \wedge$ **hasClassOfClassOfWhole**$(x, y) \rightarrow$ **ClassOfClassOfInformationRepresentation**(y)
ClassOfNamespace$(x) \rightarrow \exists y($**hasClassOfClassOfWhole**$(x, y))$
ClassOfNamespace$(x) \wedge$ **hasClassOfClassOfWhole**$(x, y) \wedge$ **hasClassOfClassOfWhole**$(x, z) \rightarrow y = z$
hasClassOfClassified$(x, y) \rightarrow$ (**ClassOfClassification**(x))
ClassOfClassification$(x) \wedge$ **hasClassOfClassified**$(x, y) \rightarrow$ **Class**(y)
ClassOfClassification$(x) \rightarrow \exists y($**hasClassOfClassified**$(x, y))$
ClassOfClassification$(x) \wedge$ **hasClassOfClassified**$(x, y) \wedge$ **hasClassOfClassified**$(x, z) \rightarrow y = z$
hasClassOfClassifier$(x, y) \rightarrow$ (**ClassOfClassification**(x))
ClassOfClassification$(x) \wedge$ **hasClassOfClassifier**$(x, y) \rightarrow$ **ClassOfClass**(y)
ClassOfClassification$(x) \rightarrow \exists y($**hasClassOfClassifier**$(x, y))$
ClassOfClassification$(x) \wedge$ **hasClassOfClassifier**$(x, y) \wedge$ **hasClassOfClassifier**$(x, z) \rightarrow y = z$
hasClassOfConnection$(x, y) \rightarrow$ (**ClassOfIndividualUsedInConnection**(x))
ClassOfIndividualUsedInConnection$(x) \wedge$ **hasClassOfConnection**$(x, y) \rightarrow$ **ClassOfConnectionOfIndividual**(y)
ClassOfIndividualUsedInConnection$(x) \rightarrow \exists y($**hasClassOfConnection**$(x, y))$
ClassOfIndividualUsedInConnection$(x) \wedge$ **hasClassOfConnection**$(x, y) \wedge$ **hasClassOfConnection**$(x, z) \rightarrow y = z$
hasClassOfControlled$(x, y) \rightarrow$ (**ClassOfResponsibilityForRepresentation**(x))
ClassOfResponsibilityForRepresentation$(x) \wedge$ **hasClassOfControlled**$(x, y) \rightarrow$
ClassOfRepresentationOfThing(y)
ClassOfResponsibilityForRepresentation$(x) \rightarrow \exists y($**hasClassOfControlled**$(x, y))$
ClassOfResponsibilityForRepresentation$(x) \wedge$ **hasClassOfControlled**$(x, y) \wedge$ **hasClassOfControlled**$(x, z) \rightarrow y = z$
hasClassOfDimension$(x, y) \rightarrow$ (**ClassOfDimensionForShape**(x))
ClassOfDimensionForShape$(x) \wedge$ **hasClassOfDimension**$(x, y) \rightarrow$ **ClassOfShapeDimension**(y)
ClassOfDimensionForShape$(x) \rightarrow \exists y($**hasClassOfDimension**$(x, y))$
ClassOfDimensionForShape$(x) \wedge$ **hasClassOfDimension**$(x, y) \wedge$ **hasClassOfDimension**$(x, z) \rightarrow y = z$
hasClassOfEnd1$(x, y) \rightarrow$ (**ClassOfRelationshipWithSignature**(x))
ClassOfRelationshipWithSignature$(x) \wedge$ **hasClassOfEnd1**$(x, y) \rightarrow$ **RoleAndDomain**(y)
ClassOfRelationshipWithSignature$(x) \wedge$ **hasClassOfEnd1**$(x, y) \wedge$ **hasClassOfEnd1**$(x, z) \rightarrow y = z$
hasClassOfEnd2$(x, y) \rightarrow$ (**ClassOfRelationshipWithSignature**(x))
ClassOfRelationshipWithSignature$(x) \wedge$ **hasClassOfEnd2**$(x, y) \rightarrow$ **RoleAndDomain**(y)
ClassOfRelationshipWithSignature$(x) \wedge$ **hasClassOfEnd2**$(x, y) \wedge$ **hasClassOfEnd2**$(x, z) \rightarrow y = z$

hasClassOfEnded$(x, y) \rightarrow ($**ClassOfCauseOfEndingOfClassOfIndividual**$(x))$
ClassOfCauseOfEndingOfClassOfIndividual$(x) \wedge$ **hasClassOfEnded**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfCauseOfEndingOfClassOfIndividual$(x) \rightarrow \exists y($**hasClassOfEnded**$(x, y))$
ClassOfCauseOfEndingOfClassOfIndividual$(x) \wedge$ **hasClassOfEnded**$(x, y) \wedge$ **hasClassOfEnded**$(x, z) \rightarrow y = z$
hasClassOfFirst$(x, y) \rightarrow$
$($**ClassOfClassOfRepresentationTranslation**$(x) \vee$ **ClassOfRepresentationTranslation**$(x))$
ClassOfClassOfRepresentationTranslation$(x) \wedge$ **hasClassOfFirst**$(x, y) \rightarrow$
ClassOfClassOfInformationRepresentation(y)
ClassOfClassOfRepresentationTranslation$(x) \rightarrow \exists y($**hasClassOfFirst**$(x, y))$
ClassOfClassOfRepresentationTranslation$(x) \wedge$ **hasClassOfFirst**$(x, y) \wedge$ **hasClassOfFirst**$(x, z) \rightarrow y = z$
ClassOfRepresentationTranslation$(x) \wedge$ **hasClassOfFirst**$(x, y) \rightarrow$ **ClassOfInformationRepresentation**(y)
ClassOfRepresentationTranslation$(x) \rightarrow \exists y($**hasClassOfFirst**$(x, y))$
ClassOfRepresentationTranslation$(x) \wedge$ **hasClassOfFirst**$(x, y) \wedge$ **hasClassOfFirst**$(x, z) \rightarrow y = z$
hasClassOfInvolved$(x, y) \rightarrow ($**ClassOfInvolvementByReference**$(x))$
ClassOfInvolvementByReference$(x) \wedge$ **hasClassOfInvolved**$(x, y) \rightarrow$ **RoleAndDomain**(y)
ClassOfInvolvementByReference$(x) \rightarrow \exists y($**hasClassOfInvolved**$(x, y))$
ClassOfInvolvementByReference$(x) \wedge$ **hasClassOfInvolved**$(x, y) \wedge$ **hasClassOfInvolved**$(x, z) \rightarrow y = z$
hasClassOfInvolver$(x, y) \rightarrow ($**ClassOfInvolvementByReference**$(x))$
ClassOfInvolvementByReference$(x) \wedge$ **hasClassOfInvolver**$(x, y) \rightarrow$ **ClassOfActivity**(y)
ClassOfInvolvementByReference$(x) \rightarrow \exists y($**hasClassOfInvolver**$(x, y))$
ClassOfInvolvementByReference$(x) \wedge$ **hasClassOfInvolver**$(x, y) \wedge$ **hasClassOfInvolver**$(x, z) \rightarrow y = z$
hasClassOfLocated$(x, y) \rightarrow ($**ClassOfRelativeLocation**$(x))$
ClassOfRelativeLocation$(x) \wedge$ **hasClassOfLocated**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfRelativeLocation$(x) \rightarrow \exists y($**hasClassOfLocated**$(x, y))$
ClassOfRelativeLocation$(x) \wedge$ **hasClassOfLocated**$(x, y) \wedge$ **hasClassOfLocated**$(x, z) \rightarrow y = z$
hasClassOfLocator$(x, y) \rightarrow ($**ClassOfRelativeLocation**$(x))$
ClassOfRelativeLocation$(x) \wedge$ **hasClassOfLocator**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfRelativeLocation$(x) \rightarrow \exists y($**hasClassOfLocator**$(x, y))$
ClassOfRelativeLocation$(x) \wedge$ **hasClassOfLocator**$(x, y) \wedge$ **hasClassOfLocator**$(x, z) \rightarrow y = z$
hasClassOfPart$(x, y) \rightarrow ($**ClassOfCompositionOfIndividual**$(x) \vee$ **ClassOfNamespace**$(x))$
ClassOfCompositionOfIndividual$(x) \wedge$ **hasClassOfPart**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfCompositionOfIndividual$(x) \rightarrow \exists y($**hasClassOfPart**$(x, y))$
ClassOfCompositionOfIndividual$(x) \wedge$ **hasClassOfPart**$(x, y) \wedge$ **hasClassOfPart**$(x, z) \rightarrow y = z$
ClassOfNamespace$(x) \wedge$ **hasClassOfPart**$(x, y) \rightarrow$ **ClassOfInformationRepresentation**(y)
ClassOfNamespace$(x) \rightarrow \exists y($**hasClassOfPart**$(x, y))$
ClassOfNamespace$(x) \wedge$ **hasClassOfPart**$(x, y) \wedge$ **hasClassOfPart**$(x, z) \rightarrow y = z$
hasClassOfPattern$(x, y) \rightarrow ($**ClassOfClassOfRepresentation**$(x))$
ClassOfClassOfRepresentation$(x) \wedge$ **hasClassOfPattern**$(x, y) \rightarrow$
ClassOfClassOfInformationRepresentation(y)
ClassOfClassOfRepresentation$(x) \rightarrow \exists y($**hasClassOfPattern**$(x, y))$
ClassOfClassOfRepresentation$(x) \wedge$ **hasClassOfPattern**$(x, y) \wedge$ **hasClassOfPattern**$(x, z) \rightarrow y = z$
hasClassOfPlayer$(x, y) \rightarrow ($**ClassOfIntendedRoleAndDomain**$(x) \vee$ **ClassOfPossibleRoleAndDomain**$(x))$
ClassOfIntendedRoleAndDomain$(x) \wedge$ **hasClassOfPlayer**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfIntendedRoleAndDomain$(x) \rightarrow \exists y($**hasClassOfPlayer**$(x, y))$
ClassOfIntendedRoleAndDomain$(x) \wedge$ **hasClassOfPlayer**$(x, y) \wedge$ **hasClassOfPlayer**$(x, z) \rightarrow y = z$
ClassOfPossibleRoleAndDomain$(x) \wedge$ **hasClassOfPlayer**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfPossibleRoleAndDomain$(x) \rightarrow \exists y($**hasClassOfPlayer**$(x, y))$
ClassOfPossibleRoleAndDomain$(x) \wedge$ **hasClassOfPlayer**$(x, y) \wedge$ **hasClassOfPlayer**$(x, z) \rightarrow y = z$
hasClassOfPossessor$(x, y) \rightarrow ($**ClassOfIndirectProperty**$(x))$
ClassOfIndirectProperty$(x) \wedge$ **hasClassOfPossessor**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfIndirectProperty$(x) \rightarrow \exists y($**hasClassOfPossessor**$(x, y))$
ClassOfIndirectProperty$(x) \wedge$ **hasClassOfPossessor**$(x, y) \wedge$ **hasClassOfPossessor**$(x, z) \rightarrow y = z$
hasClassOfPredecessor$(x, y) \rightarrow ($**ClassOfTemporalSequence**$(x))$
ClassOfTemporalSequence$(x) \wedge$ **hasClassOfPredecessor**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfTemporalSequence$(x) \rightarrow \exists y($**hasClassOfPredecessor**$(x, y))$
ClassOfTemporalSequence$(x) \wedge$ **hasClassOfPredecessor**$(x, y) \wedge$ **hasClassOfPredecessor**$(x, z) \rightarrow y = z$
hasClassOfRecognized$(x, y) \rightarrow ($**ClassOfRecognition**$(x))$
ClassOfRecognition$(x) \wedge$ **hasClassOfRecognized**$(x, y) \rightarrow$ **Class**(y)
ClassOfRecognition$(x) \rightarrow \exists y($**hasClassOfRecognized**$(x, y))$
ClassOfRecognition$(x) \wedge$ **hasClassOfRecognized**$(x, y) \wedge$ **hasClassOfRecognized**$(x, z) \rightarrow y = z$
hasClassOfRecognizing$(x, y) \rightarrow ($**ClassOfRecognition**$(x))$
ClassOfRecognition$(x) \wedge$ **hasClassOfRecognizing**$(x, y) \rightarrow$ **ClassOfActivity**(y)
ClassOfRecognition$(x) \rightarrow \exists y($**hasClassOfRecognizing**$(x, y))$
ClassOfRecognition$(x) \wedge$ **hasClassOfRecognizing**$(x, y) \wedge$ **hasClassOfRecognizing**$(x, z) \rightarrow y = z$
hasClassOfRepresented$(x, y) \rightarrow ($**ClassOfClassOfRepresentation**$(x))$

ClassOfClassOfRepresentation$(x) \wedge$ **hasClassOfRepresented**$(x, y) \rightarrow$ **Class**(y)
ClassOfClassOfRepresentation$(x) \rightarrow \exists y($**hasClassOfRepresented**$(x, y))$
ClassOfClassOfRepresentation$(x) \wedge$ **hasClassOfRepresented**$(x, y) \wedge$ **hasClassOfRepresented**$(x, z) \rightarrow y = z$
hasClassOfSecond$(x, y) \rightarrow$
(**ClassOfClassOfRepresentationTranslation**$(x) \vee$ **ClassOfRepresentationTranslation**(x))
ClassOfClassOfRepresentationTranslation$(x) \wedge$ **hasClassOfSecond**$(x, y) \rightarrow$
ClassOfClassOfInformationRepresentation(y)
ClassOfClassOfRepresentationTranslation$(x) \rightarrow \exists y($**hasClassOfSecond**$(x, y))$
ClassOfClassOfRepresentationTranslation$(x) \wedge$ **hasClassOfSecond**$(x, y) \wedge$ **hasClassOfSecond**$(x, z) \rightarrow y = z$
ClassOfRepresentationTranslation$(x) \wedge$ **hasClassOfSecond**$(x, y) \rightarrow$ **ClassOfInformationRepresentation**(y)
ClassOfRepresentationTranslation$(x) \rightarrow \exists y($**hasClassOfSecond**$(x, y))$
ClassOfRepresentationTranslation$(x) \wedge$ **hasClassOfSecond**$(x, y) \wedge$ **hasClassOfSecond**$(x, z) \rightarrow y = z$
hasClassOfShape$(x, y) \rightarrow$ (**ClassOfDimensionForShape**(x))
ClassOfDimensionForShape$(x) \wedge$ **hasClassOfShape**$(x, y) \rightarrow$ **ClassOfShape**(y)
ClassOfDimensionForShape$(x) \rightarrow \exists y($**hasClassOfShape**$(x, y))$
ClassOfDimensionForShape$(x) \wedge$ **hasClassOfShape**$(x, y) \wedge$ **hasClassOfShape**$(x, z) \rightarrow y = z$
hasClassOfShapeDimension$(x, y) \rightarrow$ (**PropertySpaceForClassOfShapeDimension**(x))
PropertySpaceForClassOfShapeDimension$(x) \wedge$ **hasClassOfShapeDimension**$(x, y) \rightarrow$
ClassOfShapeDimension(y)
PropertySpaceForClassOfShapeDimension$(x) \rightarrow \exists y($**hasClassOfShapeDimension**$(x, y))$
PropertySpaceForClassOfShapeDimension$(x) \wedge$ **hasClassOfShapeDimension**$(x, y) \wedge$
hasClassOfShapeDimension$(x, z) \rightarrow y = z$
hasClassOfSide1$(x, y) \rightarrow$ (**ClassOfConnectionOfIndividual**(x))
ClassOfConnectionOfIndividual$(x) \wedge$ **hasClassOfSide1**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfConnectionOfIndividual$(x) \rightarrow \exists y($**hasClassOfSide1**$(x, y))$
ClassOfConnectionOfIndividual$(x) \wedge$ **hasClassOfSide1**$(x, y) \wedge$ **hasClassOfSide1**$(x, z) \rightarrow y = z$
hasClassOfSide2$(x, y) \rightarrow$ (**ClassOfConnectionOfIndividual**(x))
ClassOfConnectionOfIndividual$(x) \wedge$ **hasClassOfSide2**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfConnectionOfIndividual$(x) \rightarrow \exists y($**hasClassOfSide2**$(x, y))$
ClassOfConnectionOfIndividual$(x) \wedge$ **hasClassOfSide2**$(x, y) \wedge$ **hasClassOfSide2**$(x, z) \rightarrow y = z$
hasClassOfSubclass$(x, y) \rightarrow$ (**ClassOfSpecialization**(x))
ClassOfSpecialization$(x) \wedge$ **hasClassOfSubclass**$(x, y) \rightarrow$ **ClassOfClass**(y)
ClassOfSpecialization$(x) \rightarrow \exists y($**hasClassOfSubclass**$(x, y))$
ClassOfSpecialization$(x) \wedge$ **hasClassOfSubclass**$(x, y) \wedge$ **hasClassOfSubclass**$(x, z) \rightarrow y = z$
hasClassOfSuccessor$(x, y) \rightarrow$ (**ClassOfTemporalSequence**(x))
ClassOfTemporalSequence$(x) \wedge$ **hasClassOfSuccessor**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfTemporalSequence$(x) \rightarrow \exists y($**hasClassOfSuccessor**$(x, y))$
ClassOfTemporalSequence$(x) \wedge$ **hasClassOfSuccessor**$(x, y) \wedge$ **hasClassOfSuccessor**$(x, z) \rightarrow y = z$
hasClassOfSuperclass$(x, y) \rightarrow$ (**ClassOfSpecialization**(x))
ClassOfSpecialization$(x) \wedge$ **hasClassOfSuperclass**$(x, y) \rightarrow$ **ClassOfClass**(y)
ClassOfSpecialization$(x) \rightarrow \exists y($**hasClassOfSuperclass**$(x, y))$
ClassOfSpecialization$(x) \wedge$ **hasClassOfSuperclass**$(x, y) \wedge$ **hasClassOfSuperclass**$(x, z) \rightarrow y = z$
hasClassOfUsage$(x, y) \rightarrow$ (**ClassOfIndividualUsedInConnection**(x))
ClassOfIndividualUsedInConnection$(x) \wedge$ **hasClassOfUsage**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfIndividualUsedInConnection$(x) \rightarrow \exists y($**hasClassOfUsage**$(x, y))$
ClassOfIndividualUsedInConnection$(x) \wedge$ **hasClassOfUsage**$(x, y) \wedge$ **hasClassOfUsage**$(x, z) \rightarrow y = z$
hasClassOfUsed$(x, y) \rightarrow$ (**ClassOfUsageOfRepresentation**(x))
ClassOfUsageOfRepresentation$(x) \wedge$ **hasClassOfUsed**$(x, y) \rightarrow$ **ClassOfRepresentationOfThing**(y)
ClassOfUsageOfRepresentation$(x) \rightarrow \exists y($**hasClassOfUsed**$(x, y))$
ClassOfUsageOfRepresentation$(x) \wedge$ **hasClassOfUsed**$(x, y) \wedge$ **hasClassOfUsed**$(x, z) \rightarrow y = z$
hasClassOfWhole$(x, y) \rightarrow$ (**ClassOfCompositionOfIndividual**(x))
ClassOfCompositionOfIndividual$(x) \wedge$ **hasClassOfWhole**$(x, y) \rightarrow$ **ClassOfIndividual**(y)
ClassOfCompositionOfIndividual$(x) \rightarrow \exists y($**hasClassOfWhole**$(x, y))$
ClassOfCompositionOfIndividual$(x) \wedge$ **hasClassOfWhole**$(x, y) \wedge$ **hasClassOfWhole**$(x, z) \rightarrow y = z$
hasClassified$(x, y) \rightarrow$ (**Classification**(x))
Classification$(x) \wedge$ **hasClassified**$(x, y) \rightarrow$ **Thing**(y)
Classification$(x) \rightarrow \exists y($**hasClassified**$(x, y))$
Classification$(x) \wedge$ **hasClassified**$(x, y) \wedge$ **hasClassified**$(x, z) \rightarrow y = z$
hasClassifier$(x, y) \rightarrow$ (**Classification**(x))
Classification$(x) \wedge$ **hasClassifier**$(x, y) \rightarrow$ **Class**(y)
Classification$(x) \rightarrow \exists y($**hasClassifier**$(x, y))$
Classification$(x) \wedge$ **hasClassifier**$(x, y) \wedge$ **hasClassifier**$(x, z) \rightarrow y = z$
hasCodomain$(x, y) \rightarrow$ (**ClassOfFunctionalMapping**(x))
ClassOfFunctionalMapping$(x) \wedge$ **hasCodomain**$(x, y) \rightarrow$ **Class**(y)
ClassOfFunctionalMapping$(x) \rightarrow \exists y($**hasCodomain**$(x, y))$

ClassOfFunctionalMapping(x) ∧ **hasCodomain**(x, y) ∧ **hasCodomain**(x, z) → $y = z$
hasConnection(x, y) → (**IndividualUsedInConnection**(x))
IndividualUsedInConnection(x) ∧ **hasConnection**(x, y) → **ConnectionOfIndividual**(y)
IndividualUsedInConnection(x) → ∃y(**hasConnection**(x, y))
IndividualUsedInConnection(x) ∧ **hasConnection**(x, y) ∧ **hasConnection**(x, z) → $y = z$
hasContent(x, y) → (**ExpressBinary**(x) ∨ **ExpressBoolean**(x) ∨ **ExpressInteger**(x) ∨ **ExpressLogical**(x) ∨
ExpressReal(x) ∨ **ExpressString**(x))
ExpressBinary(x) ∧ **hasContent**(x, y) → **BINARY**(y)
ExpressBinary(x) → ∃y(**hasContent**(x, y))
ExpressBinary(x) ∧ **hasContent**(x, y) ∧ **hasContent**(x, z) → $y = z$
ExpressBinary(x) ∧ **ExpressBinary**(y) ∧ **hasContent**(x, z) ∧ **hasContent**(y, z) → $x = y$
ExpressBoolean(x) ∧ **hasContent**(x, y) → **BOOLEAN**(y)
ExpressBoolean(x) → ∃y(**hasContent**(x, y))
ExpressBoolean(x) ∧ **hasContent**(x, y) ∧ **hasContent**(x, z) → $y = z$
ExpressBoolean(x) ∧ **ExpressBoolean**(y) ∧ **hasContent**(x, z) ∧ **hasContent**(y, z) → $x = y$
ExpressInteger(x) ∧ **hasContent**(x, y) → **INTEGER**(y)
ExpressInteger(x) → ∃y(**hasContent**(x, y))
ExpressInteger(x) ∧ **hasContent**(x, y) ∧ **hasContent**(x, z) → $y = z$
ExpressInteger(x) ∧ **ExpressInteger**(y) ∧ **hasContent**(x, z) ∧ **hasContent**(y, z) → $x = y$
ExpressLogical(x) ∧ **hasContent**(x, y) → **LOGICAL**(y)
ExpressLogical(x) → ∃y(**hasContent**(x, y))
ExpressLogical(x) ∧ **hasContent**(x, y) ∧ **hasContent**(x, z) → $y = z$
ExpressLogical(x) ∧ **ExpressLogical**(y) ∧ **hasContent**(x, z) ∧ **hasContent**(y, z) → $x = y$
ExpressReal(x) ∧ **hasContent**(x, y) → **REAL**(y)
ExpressReal(x) → ∃y(**hasContent**(x, y))
ExpressReal(x) ∧ **hasContent**(x, y) ∧ **hasContent**(x, z) → $y = z$
ExpressReal(x) ∧ **ExpressReal**(y) ∧ **hasContent**(x, z) ∧ **hasContent**(y, z) → $x = y$
ExpressString(x) ∧ **hasContent**(x, y) → **STRING**(y)
ExpressString(x) → ∃y(**hasContent**(x, y))
ExpressString(x) ∧ **hasContent**(x, y) ∧ **hasContent**(x, z) → $y = z$
ExpressString(x) ∧ **ExpressString**(y) ∧ **hasContent**(x, z) ∧ **hasContent**(y, z) → $x = y$
hasControlled(x, y) → (**ResponsibilityForRepresentation**(x))
ResponsibilityForRepresentation(x) ∧ **hasControlled**(x, y) → **RepresentationOfThing**(y)
ResponsibilityForRepresentation(x) → ∃y(**hasControlled**(x, y))
ResponsibilityForRepresentation(x) ∧ **hasControlled**(x, y) ∧ **hasControlled**(x, z) → $y = z$
hasController(x, y) → (**ClassOfClassOfResponsibilityForRepresentation**(x) ∨
ClassOfResponsibilityForRepresentation(x) ∨ **ResponsibilityForRepresentation**(x))
ClassOfClassOfResponsibilityForRepresentation(x) ∧ **hasController**(x, y) → **PossibleIndividual**(y)
ClassOfClassOfResponsibilityForRepresentation(x) → ∃y(**hasController**(x, y))
ClassOfClassOfResponsibilityForRepresentation(x) ∧ **hasController**(x, y) ∧ **hasController**(x, z) → $y = z$
ClassOfResponsibilityForRepresentation(x) ∧ **hasController**(x, y) → **PossibleIndividual**(y)
ClassOfResponsibilityForRepresentation(x) → ∃y(**hasController**(x, y))
ClassOfResponsibilityForRepresentation(x) ∧ **hasController**(x, y) ∧ **hasController**(x, z) → $y = z$
ResponsibilityForRepresentation(x) ∧ **hasController**(x, y) → **PossibleIndividual**(y)
ResponsibilityForRepresentation(x) → ∃y(**hasController**(x, y))
ResponsibilityForRepresentation(x) ∧ **hasController**(x, y) ∧ **hasController**(x, z) → $y = z$
hasDay(x, y) → (**RepresentationOfGregorianDateAndUtcTime**(x))
RepresentationOfGregorianDateAndUtcTime(x) ∧ **hasDay**(x, y) → **INTEGER**(y)
RepresentationOfGregorianDateAndUtcTime(x) ∧ **hasDay**(x, y) ∧ **hasDay**(x, z) → $y = z$
hasDimension(x, y) → (**DimensionOfShape**(x))
DimensionOfShape(x) ∧ **hasDimension**(x, y) → **ShapeDimension**(y)
DimensionOfShape(x) → ∃y(**hasDimension**(x, y))
DimensionOfShape(x) ∧ **hasDimension**(x, y) ∧ **hasDimension**(x, z) → $y = z$
hasDomain(x, y) → (**ClassOfFunctionalMapping**(x))
ClassOfFunctionalMapping(x) ∧ **hasDomain**(x, y) → **Class**(y)
ClassOfFunctionalMapping(x) → ∃y(**hasDomain**(x, y))
ClassOfFunctionalMapping(x) ∧ **hasDomain**(x, y) ∧ **hasDomain**(x, z) → $y = z$
hasElements(x, y) → (**MultidimensionalObject**(x))
MultidimensionalObject(x) → ∃y(**hasElements**(x, y))
MultidimensionalObject(x) ∧ **hasElements**(x, y) ∧ **hasElements**(x, z) → $y = z$
hasEnd1(x, y) → (**OtherRelationship**(x))
OtherRelationship(x) ∧ **hasEnd1**(x, y) → **Thing**(y)
OtherRelationship(x) → ∃y(**hasEnd1**(x, y))
OtherRelationship(x) ∧ **hasEnd1**(x, y) ∧ **hasEnd1**(x, z) → $y = z$
hasEnd1Cardinality(x, y) → (**ClassOfRelationship**(x))

ClassOfRelationship$(x) \wedge$ **hasEnd1Cardinality**$(x, y) \rightarrow$ **Cardinality**(y)
ClassOfRelationship$(x) \wedge$ **hasEnd1Cardinality**$(x, y) \wedge$ **hasEnd1Cardinality**$(x, z) \rightarrow y = z$
hasEnd2$(x, y) \rightarrow$ (**OtherRelationship**(x))
OtherRelationship$(x) \wedge$ **hasEnd2**$(x, y) \rightarrow$ **Thing**(y)
OtherRelationship$(x) \rightarrow \exists y$(**hasEnd2**$(x, y)$)
OtherRelationship$(x) \wedge$ **hasEnd2**$(x, y) \wedge$ **hasEnd2**$(x, z) \rightarrow y = z$
hasEnd2Cardinality$(x, y) \rightarrow$ (**ClassOfRelationship**(x))
ClassOfRelationship$(x) \wedge$ **hasEnd2Cardinality**$(x, y) \rightarrow$ **Cardinality**(y)
ClassOfRelationship$(x) \wedge$ **hasEnd2Cardinality**$(x, y) \wedge$ **hasEnd2Cardinality**$(x, z) \rightarrow y = z$
hasGreaterElement$(x, y) \rightarrow$ (**ComparisonOfProperty**(x))
ComparisonOfProperty$(x) \wedge$ **hasGreaterElement**$(x, y) \rightarrow$ **Property**(y)
ComparisonOfProperty$(x) \rightarrow \exists y$(**hasGreaterElement**$(x, y)$)
ComparisonOfProperty$(x) \wedge$ **hasGreaterElement**$(x, y) \wedge$ **hasGreaterElement**$(x, z) \rightarrow y = z$
hasHour$(x, y) \rightarrow$ (**RepresentationOfGregorianDateAndUtcTime**(x))
RepresentationOfGregorianDateAndUtcTime$(x) \wedge$ **hasHour**$(x, y) \rightarrow$ **INTEGER**(y)
RepresentationOfGregorianDateAndUtcTime$(x) \wedge$ **hasHour**$(x, y) \wedge$ **hasHour**$(x, z) \rightarrow y = z$
hasId$(x, y) \rightarrow$ (**Thing**(x))
Thing$(x) \wedge$ **hasId**$(x, y) \rightarrow$ **STRING**(y)
Thing$(x) \rightarrow \exists y$(**hasId**$(x, y)$)
Thing$(x) \wedge$ **hasId**$(x, y) \wedge$ **hasId**$(x, z) \rightarrow y = z$
Thing$(x) \wedge$ **Thing**$(y) \wedge$ **hasId**$(x, z) \wedge$ **hasId**$(y, z) \rightarrow x = y$
hasIndividual$(x, y) \rightarrow$ (**DimensionOfIndividual**(x))
DimensionOfIndividual$(x) \wedge$ **hasIndividual**$(x, y) \rightarrow$ **PossibleIndividual**(y)
DimensionOfIndividual$(x) \rightarrow \exists y$(**hasIndividual**$(x, y)$)
DimensionOfIndividual$(x) \wedge$ **hasIndividual**$(x, y) \wedge$ **hasIndividual**$(x, z) \rightarrow y = z$
hasIndividualDimension$(x, y) \rightarrow$ (**DimensionOfIndividual**(x))
DimensionOfIndividual$(x) \wedge$ **hasIndividualDimension**$(x, y) \rightarrow$ **IndividualDimension**(y)
DimensionOfIndividual$(x) \rightarrow \exists y$(**hasIndividualDimension**(x, y))
DimensionOfIndividual$(x) \wedge$ **hasIndividualDimension**$(x, y) \wedge$ **hasIndividualDimension**$(x, z) \rightarrow y = z$
hasInput$(x, y) \rightarrow$ (**FunctionalMapping**(x))
FunctionalMapping$(x) \wedge$ **hasInput**$(x, y) \rightarrow$ **Thing**(y)
FunctionalMapping$(x) \rightarrow \exists y$(**hasInput**$(x, y)$)
FunctionalMapping$(x) \wedge$ **hasInput**$(x, y) \wedge$ **hasInput**$(x, z) \rightarrow y = z$
hasInterest$(x, y) \rightarrow$ (**LifecycleStage**(x))
LifecycleStage$(x) \wedge$ **hasInterest**$(x, y) \rightarrow$ **PossibleIndividual**(y)
LifecycleStage$(x) \rightarrow \exists y$(**hasInterest**$(x, y)$)
LifecycleStage$(x) \wedge$ **hasInterest**$(x, y) \wedge$ **hasInterest**$(x, z) \rightarrow y = z$
hasInterested$(x, y) \rightarrow$ (**LifecycleStage**(x))
LifecycleStage$(x) \wedge$ **hasInterested**$(x, y) \rightarrow$ **PossibleIndividual**(y)
LifecycleStage$(x) \rightarrow \exists y$(**hasInterested**$(x, y)$)
LifecycleStage$(x) \wedge$ **hasInterested**$(x, y) \wedge$ **hasInterested**$(x, z) \rightarrow y = z$
hasInvolved$(x, y) \rightarrow$ (**InvolvementByReference**(x))
InvolvementByReference$(x) \wedge$ **hasInvolved**$(x, y) \rightarrow$ **Thing**(y)
InvolvementByReference$(x) \rightarrow \exists y$(**hasInvolved**$(x, y)$)
InvolvementByReference$(x) \wedge$ **hasInvolved**$(x, y) \wedge$ **hasInvolved**$(x, z) \rightarrow y = z$
hasInvolver$(x, y) \rightarrow$ (**InvolvementByReference**(x))
InvolvementByReference$(x) \wedge$ **hasInvolver**$(x, y) \rightarrow$ **Activity**(y)
InvolvementByReference$(x) \rightarrow \exists y$(**hasInvolver**$(x, y)$)
InvolvementByReference$(x) \wedge$ **hasInvolver**$(x, y) \wedge$ **hasInvolver**$(x, z) \rightarrow y = z$
hasLesserElement$(x, y) \rightarrow$ (**ComparisonOfProperty**(x))
ComparisonOfProperty$(x) \wedge$ **hasLesserElement**$(x, y) \rightarrow$ **Property**(y)
ComparisonOfProperty$(x) \rightarrow \exists y$(**hasLesserElement**$(x, y)$)
ComparisonOfProperty$(x) \wedge$ **hasLesserElement**$(x, y) \wedge$ **hasLesserElement**$(x, z) \rightarrow y = z$
hasLocated$(x, y) \rightarrow$ (**RelativeLocation**(x))
RelativeLocation$(x) \wedge$ **hasLocated**$(x, y) \rightarrow$ **PossibleIndividual**(y)
RelativeLocation$(x) \rightarrow \exists y$(**hasLocated**$(x, y)$)
RelativeLocation$(x) \wedge$ **hasLocated**$(x, y) \wedge$ **hasLocated**$(x, z) \rightarrow y = z$
hasLocator$(x, y) \rightarrow$ (**RelativeLocation**(x))
RelativeLocation$(x) \wedge$ **hasLocator**$(x, y) \rightarrow$ **PossibleIndividual**(y)
RelativeLocation$(x) \rightarrow \exists y$(**hasLocator**$(x, y)$)
RelativeLocation$(x) \wedge$ **hasLocator**$(x, y) \wedge$ **hasLocator**$(x, z) \rightarrow y = z$
hasMaximumCardinality$(x, y) \rightarrow$ (**Cardinality**(x))
Cardinality$(x) \wedge$ **hasMaximumCardinality**$(x, y) \rightarrow$ **INTEGER**(y)
Cardinality$(x) \wedge$ **hasMaximumCardinality**$(x, y) \wedge$ **hasMaximumCardinality**$(x, z) \rightarrow y = z$
hasMinimumCardinality$(x, y) \rightarrow$ (**Cardinality**(x))

Cardinality$(x) \wedge$ **hasMinimumCardinality**$(x, y) \rightarrow$ **INTEGER**(y)
Cardinality$(x) \wedge$ **hasMinimumCardinality**$(x, y) \wedge$ **hasMinimumCardinality**$(x, z) \rightarrow y = z$
hasMinute$(x, y) \rightarrow$ (**RepresentationOfGregorianDateAndUtcTime**(x))
RepresentationOfGregorianDateAndUtcTime$(x) \wedge$ **hasMinute**$(x, y) \rightarrow$ **INTEGER**(y)
RepresentationOfGregorianDateAndUtcTime$(x) \wedge$ **hasMinute**$(x, y) \wedge$ **hasMinute**$(x, z) \rightarrow y = z$
hasMonth$(x, y) \rightarrow$ (**RepresentationOfGregorianDateAndUtcTime**(x))
RepresentationOfGregorianDateAndUtcTime$(x) \wedge$ **hasMonth**$(x, y) \rightarrow$ **INTEGER**(y)
RepresentationOfGregorianDateAndUtcTime$(x) \wedge$ **hasMonth**$(x, y) \wedge$ **hasMonth**$(x, z) \rightarrow y = z$
hasOptionalElement$(x, y) \rightarrow$ (**ClassOfMultidimensionalObject**(x))
ClassOfMultidimensionalObject$(x) \rightarrow \exists y$(**hasOptionalElement**$(x, y)$)
ClassOfMultidimensionalObject$(x) \wedge$ **hasOptionalElement**$(x, y) \wedge$ **hasOptionalElement**$(x, z) \rightarrow y = z$
hasParameterPosition$(x, y) \rightarrow$ (**ClassOfMultidimensionalObject**(x))
ClassOfMultidimensionalObject$(x) \wedge$ **hasParameterPosition**$(x, y) \wedge$ **hasParameterPosition**$(x, z) \rightarrow y = z$
hasParameters$(x, y) \rightarrow$ (**ClassOfMultidimensionalObject**(x))
ClassOfMultidimensionalObject$(x) \wedge$ **hasParameters**$(x, y) \wedge$ **hasParameters**$(x, z) \rightarrow y = z$
hasPart$(x, y) \rightarrow$ (**CompositionOfIndividual**(x))
CompositionOfIndividual$(x) \wedge$ **hasPart**$(x, y) \rightarrow$ **PossibleIndividual**(y)
CompositionOfIndividual$(x) \rightarrow \exists y$(**hasPart**$(x, y)$)
CompositionOfIndividual$(x) \wedge$ **hasPart**$(x, y) \wedge$ **hasPart**$(x, z) \rightarrow y = z$
hasPattern$(x, y) \rightarrow$ (**ClassOfRepresentationOfThing**(x))
ClassOfRepresentationOfThing$(x) \wedge$ **hasPattern**$(x, y) \rightarrow$ **ClassOfInformationRepresentation**(y)
ClassOfRepresentationOfThing$(x) \rightarrow \exists y$(**hasPattern**$(x, y)$)
ClassOfRepresentationOfThing$(x) \wedge$ **hasPattern**$(x, y) \wedge$ **hasPattern**$(x, z) \rightarrow y = z$
hasPlayed$(x, y) \rightarrow$ (**ClassOfIntendedRoleAndDomain**$(x) \vee$ **ClassOfPossibleRoleAndDomain**$(x) \vee$
IntendedRoleAndDomain$(x) \vee$ **PossibleRoleAndDomain**(x))
ClassOfIntendedRoleAndDomain$(x) \wedge$ **hasPlayed**$(x, y) \rightarrow$ **RoleAndDomain**(y)
ClassOfIntendedRoleAndDomain$(x) \rightarrow \exists y$(**hasPlayed**$(x, y)$)
ClassOfIntendedRoleAndDomain$(x) \wedge$ **hasPlayed**$(x, y) \wedge$ **hasPlayed**$(x, z) \rightarrow y = z$
ClassOfPossibleRoleAndDomain$(x) \wedge$ **hasPlayed**$(x, y) \rightarrow$ **RoleAndDomain**(y)
ClassOfPossibleRoleAndDomain$(x) \rightarrow \exists y$(**hasPlayed**$(x, y)$)
ClassOfPossibleRoleAndDomain$(x) \wedge$ **hasPlayed**$(x, y) \wedge$ **hasPlayed**$(x, z) \rightarrow y = z$
IntendedRoleAndDomain$(x) \wedge$ **hasPlayed**$(x, y) \rightarrow$ **RoleAndDomain**(y)
IntendedRoleAndDomain$(x) \rightarrow \exists y$(**hasPlayed**$(x, y)$)
IntendedRoleAndDomain$(x) \wedge$ **hasPlayed**$(x, y) \wedge$ **hasPlayed**$(x, z) \rightarrow y = z$
PossibleRoleAndDomain$(x) \wedge$ **hasPlayed**$(x, y) \rightarrow$ **RoleAndDomain**(y)
PossibleRoleAndDomain$(x) \rightarrow \exists y$(**hasPlayed**$(x, y)$)
PossibleRoleAndDomain$(x) \wedge$ **hasPlayed**$(x, y) \wedge$ **hasPlayed**$(x, z) \rightarrow y = z$
hasPlayer$(x, y) \rightarrow$ (**IntendedRoleAndDomain**$(x) \vee$ **PossibleRoleAndDomain**(x))
IntendedRoleAndDomain$(x) \wedge$ **hasPlayer**$(x, y) \rightarrow$ **PossibleIndividual**(y)
IntendedRoleAndDomain$(x) \rightarrow \exists y$(**hasPlayer**$(x, y)$)
IntendedRoleAndDomain$(x) \wedge$ **hasPlayer**$(x, y) \wedge$ **hasPlayer**$(x, z) \rightarrow y = z$
PossibleRoleAndDomain$(x) \wedge$ **hasPlayer**$(x, y) \rightarrow$ **PossibleIndividual**(y)
PossibleRoleAndDomain$(x) \rightarrow \exists y$(**hasPlayer**$(x, y)$)
PossibleRoleAndDomain$(x) \wedge$ **hasPlayer**$(x, y) \wedge$ **hasPlayer**$(x, z) \rightarrow y = z$
hasPosition$(x, y) \rightarrow$ (**MultidimensionalObject**(x))
MultidimensionalObject$(x) \wedge$ **hasPosition**$(x, y) \wedge$ **hasPosition**$(x, z) \rightarrow y = z$
hasPossessor$(x, y) \rightarrow$ (**IndirectProperty**(x))
IndirectProperty$(x) \wedge$ **hasPossessor**$(x, y) \rightarrow$ **PossibleIndividual**(y)
IndirectProperty$(x) \rightarrow \exists y$(**hasPossessor**$(x, y)$)
IndirectProperty$(x) \wedge$ **hasPossessor**$(x, y) \wedge$ **hasPossessor**$(x, z) \rightarrow y = z$
hasPredecessor$(x, y) \rightarrow$ (**TemporalSequence**(x))
TemporalSequence$(x) \wedge$ **hasPredecessor**$(x, y) \rightarrow$ **PossibleIndividual**(y)
TemporalSequence$(x) \rightarrow \exists y$(**hasPredecessor**$(x, y)$)
TemporalSequence$(x) \wedge$ **hasPredecessor**$(x, y) \wedge$ **hasPredecessor**$(x, z) \rightarrow y = z$
hasProperty$(x, y) \rightarrow$ (**IndirectProperty**$(x) \vee$ **PropertyForShapeDimension**(x))
IndirectProperty$(x) \wedge$ **hasProperty**$(x, y) \rightarrow$ **Property**(y)
IndirectProperty$(x) \rightarrow \exists y$(**hasProperty**$(x, y)$)
IndirectProperty$(x) \wedge$ **hasProperty**$(x, y) \wedge$ **hasProperty**$(x, z) \rightarrow y = z$
PropertyForShapeDimension$(x) \wedge$ **hasProperty**$(x, y) \rightarrow$ **Property**(y)
PropertyForShapeDimension$(x) \rightarrow \exists y$(**hasProperty**$(x, y)$)
PropertyForShapeDimension$(x) \wedge$ **hasProperty**$(x, y) \wedge$ **hasProperty**$(x, z) \rightarrow y = z$
hasPropertySpace$(x, y) \rightarrow$ (**ClassOfIndirectProperty**$(x) \vee$ **PropertySpaceForClassOfShapeDimension**(x))
ClassOfIndirectProperty$(x) \wedge$ **hasPropertySpace**$(x, y) \rightarrow$ **PropertySpace**(y)
ClassOfIndirectProperty$(x) \rightarrow \exists y$(**hasPropertySpace**$(x, y)$)
ClassOfIndirectProperty$(x) \wedge$ **hasPropertySpace**$(x, y) \wedge$ **hasPropertySpace**$(x, z) \rightarrow y = z$

PropertySpaceForClassOfShapeDimension$(x) \wedge$ **hasPropertySpace**$(x, y) \rightarrow$ **PropertySpace**(y)
PropertySpaceForClassOfShapeDimension$(x) \rightarrow \exists y($**hasPropertySpace**$(x, y))$
PropertySpaceForClassOfShapeDimension$(x) \wedge$ **hasPropertySpace**$(x, y) \wedge$ **hasPropertySpace**$(x, z) \rightarrow y = z$
hasRecognized$(x, y) \rightarrow ($**Recognition**$(x))$
Recognition$(x) \wedge$ **hasRecognized**$(x, y) \rightarrow$ **Thing**(y)
Recognition$(x) \rightarrow \exists y($**hasRecognized**$(x, y))$
Recognition$(x) \wedge$ **hasRecognized**$(x, y) \wedge$ **hasRecognized**$(x, z) \rightarrow y = z$
hasRecognizing$(x, y) \rightarrow ($**Recognition**$(x))$
Recognition$(x) \wedge$ **hasRecognizing**$(x, y) \rightarrow$ **Activity**(y)
Recognition$(x) \rightarrow \exists y($**hasRecognizing**$(x, y))$
Recognition$(x) \wedge$ **hasRecognizing**$(x, y) \wedge$ **hasRecognizing**$(x, z) \rightarrow y = z$
hasRecordCopyCreated$(x, y) \rightarrow ($**Thing**$(x))$
Thing$(x) \wedge$ **hasRecordCopyCreated**$(x, y) \rightarrow$ **RepresentationOfGregorianDateAndUtcTime**(y)
Thing$(x) \wedge$ **hasRecordCopyCreated**$(x, y) \wedge$ **hasRecordCopyCreated**$(x, z) \rightarrow y = z$
hasRecordCreated$(x, y) \rightarrow ($**Thing**$(x))$
Thing$(x) \wedge$ **hasRecordCreated**$(x, y) \rightarrow$ **RepresentationOfGregorianDateAndUtcTime**(y)
Thing$(x) \wedge$ **hasRecordCreated**$(x, y) \wedge$ **hasRecordCreated**$(x, z) \rightarrow y = z$
hasRecordCreator$(x, y) \rightarrow ($**Thing**$(x))$
Thing$(x) \wedge$ **hasRecordCreator**$(x, y) \rightarrow$ **PossibleIndividual**(y)
Thing$(x) \wedge$ **hasRecordCreator**$(x, y) \wedge$ **hasRecordCreator**$(x, z) \rightarrow y = z$
hasRecordLogicallyDeleted$(x, y) \rightarrow ($**Thing**$(x))$
Thing$(x) \wedge$ **hasRecordLogicallyDeleted**$(x, y) \rightarrow$ **RepresentationOfGregorianDateAndUtcTime**(y)
Thing$(x) \wedge$ **hasRecordLogicallyDeleted**$(x, y) \wedge$ **hasRecordLogicallyDeleted**$(x, z) \rightarrow y = z$
hasRelated$(x, y) \rightarrow ($**ClassOfRelationshipWithRelatedEnd1**$(x) \vee$ **ClassOfRelationshipWithRelatedEnd2**$(x))$
ClassOfRelationshipWithRelatedEnd1$(x) \wedge$ **hasRelated**$(x, y) \rightarrow$ **Thing**(y)
ClassOfRelationshipWithRelatedEnd1$(x) \rightarrow \exists y($**hasRelated**$(x, y))$
ClassOfRelationshipWithRelatedEnd1$(x) \wedge$ **hasRelated**$(x, y) \wedge$ **hasRelated**$(x, z) \rightarrow y = z$
ClassOfRelationshipWithRelatedEnd2$(x) \wedge$ **hasRelated**$(x, y) \rightarrow$ **Thing**(y)
ClassOfRelationshipWithRelatedEnd2$(x) \rightarrow \exists y($**hasRelated**$(x, y))$
ClassOfRelationshipWithRelatedEnd2$(x) \wedge$ **hasRelated**$(x, y) \wedge$ **hasRelated**$(x, z) \rightarrow y = z$
hasRepresented$(x, y) \rightarrow ($**ClassOfRepresentationOfThing**$(x) \vee$ **RepresentationOfThing**$(x))$
ClassOfRepresentationOfThing$(x) \wedge$ **hasRepresented**$(x, y) \rightarrow$ **Thing**(y)
ClassOfRepresentationOfThing$(x) \rightarrow \exists y($**hasRepresented**$(x, y))$
ClassOfRepresentationOfThing$(x) \wedge$ **hasRepresented**$(x, y) \wedge$ **hasRepresented**$(x, z) \rightarrow y = z$
RepresentationOfThing$(x) \wedge$ **hasRepresented**$(x, y) \rightarrow$ **Thing**(y)
RepresentationOfThing$(x) \rightarrow \exists y($**hasRepresented**$(x, y))$
RepresentationOfThing$(x) \wedge$ **hasRepresented**$(x, y) \wedge$ **hasRepresented**$(x, z) \rightarrow y = z$
hasResult$(x, y) \rightarrow ($**FunctionalMapping**$(x))$
FunctionalMapping$(x) \wedge$ **hasResult**$(x, y) \rightarrow$ **Thing**(y)
FunctionalMapping$(x) \rightarrow \exists y($**hasResult**$(x, y))$
FunctionalMapping$(x) \wedge$ **hasResult**$(x, y) \wedge$ **hasResult**$(x, z) \rightarrow y = z$
hasRoles$(x, y) \rightarrow ($**ClassOfMultidimensionalObject**$(x))$
ClassOfMultidimensionalObject$(x) \rightarrow \exists y($**hasRoles**$(x, y))$
ClassOfMultidimensionalObject$(x) \wedge$ **hasRoles**$(x, y) \wedge$ **hasRoles**$(x, z) \rightarrow y = z$
hasSecond$(x, y) \rightarrow ($**RepresentationOfGregorianDateAndUtcTime**$(x))$
RepresentationOfGregorianDateAndUtcTime$(x) \wedge$ **hasSecond**$(x, y) \rightarrow$ **REAL**(y)
RepresentationOfGregorianDateAndUtcTime$(x) \wedge$ **hasSecond**$(x, y) \wedge$ **hasSecond**$(x, z) \rightarrow y = z$
hasShape$(x, y) \rightarrow ($**DimensionOfShape**$(x))$
DimensionOfShape$(x) \wedge$ **hasShape**$(x, y) \rightarrow$ **Shape**(y)
DimensionOfShape$(x) \rightarrow \exists y($**hasShape**$(x, y))$
DimensionOfShape$(x) \wedge$ **hasShape**$(x, y) \wedge$ **hasShape**$(x, z) \rightarrow y = z$
hasShapeDimension$(x, y) \rightarrow ($**PropertyForShapeDimension**$(x))$
PropertyForShapeDimension$(x) \wedge$ **hasShapeDimension**$(x, y) \rightarrow$ **ShapeDimension**(y)
PropertyForShapeDimension$(x) \rightarrow \exists y($**hasShapeDimension**$(x, y))$
PropertyForShapeDimension$(x) \wedge$ **hasShapeDimension**$(x, y) \wedge$ **hasShapeDimension**$(x, z) \rightarrow y = z$
hasSide1$(x, y) \rightarrow ($**ConnectionOfIndividual**$(x))$
ConnectionOfIndividual$(x) \wedge$ **hasSide1**$(x, y) \rightarrow$ **PossibleIndividual**(y)
ConnectionOfIndividual$(x) \rightarrow \exists y($**hasSide1**$(x, y))$
ConnectionOfIndividual$(x) \wedge$ **hasSide1**$(x, y) \wedge$ **hasSide1**$(x, z) \rightarrow y = z$
hasSide2$(x, y) \rightarrow ($**ConnectionOfIndividual**$(x))$
ConnectionOfIndividual$(x) \wedge$ **hasSide2**$(x, y) \rightarrow$ **PossibleIndividual**(y)
ConnectionOfIndividual$(x) \rightarrow \exists y($**hasSide2**$(x, y))$
ConnectionOfIndividual$(x) \wedge$ **hasSide2**$(x, y) \wedge$ **hasSide2**$(x, z) \rightarrow y = z$
hasSign$(x, y) \rightarrow ($**RepresentationOfThing**$(x))$
RepresentationOfThing$(x) \wedge$ **hasSign**$(x, y) \rightarrow$ **PossibleIndividual**(y)

RepresentationOfThing(x) → ∃y(**hasSign**(x, y))
RepresentationOfThing(x) ∧ **hasSign**(x, y) ∧ **hasSign**(x, z) → $y = z$
hasSubclass(x, y) → (**Specialization**(x))
Specialization(x) ∧ **hasSubclass**(x, y) → **Class**(y)
Specialization(x) → ∃y(**hasSubclass**(x, y))
Specialization(x) ∧ **hasSubclass**(x, y) ∧ **hasSubclass**(x, z) → $y = z$
hasSuccessor(x, y) → (**TemporalSequence**(x))
TemporalSequence(x) ∧ **hasSuccessor**(x, y) → **PossibleIndividual**(y)
TemporalSequence(x) → ∃y(**hasSuccessor**(x, y))
TemporalSequence(x) ∧ **hasSuccessor**(x, y) ∧ **hasSuccessor**(x, z) → $y = z$
hasSuperclass(x, y) → (**Specialization**(x))
Specialization(x) ∧ **hasSuperclass**(x, y) → **Class**(y)
Specialization(x) → ∃y(**hasSuperclass**(x, y))
Specialization(x) ∧ **hasSuperclass**(x, y) ∧ **hasSuperclass**(x, z) → $y = z$
hasUsage(x, y) → (**IndividualUsedInConnection**(x))
IndividualUsedInConnection(x) ∧ **hasUsage**(x, y) → **PossibleIndividual**(y)
IndividualUsedInConnection(x) → ∃y(**hasUsage**(x, y))
IndividualUsedInConnection(x) ∧ **hasUsage**(x, y) ∧ **hasUsage**(x, z) → $y = z$
hasUsed(x, y) → (**UsageOfRepresentation**(x))
UsageOfRepresentation(x) ∧ **hasUsed**(x, y) → **RepresentationOfThing**(y)
UsageOfRepresentation(x) → ∃y(**hasUsed**(x, y))
UsageOfRepresentation(x) ∧ **hasUsed**(x, y) ∧ **hasUsed**(x, z) → $y = z$
hasUser(x, y) → (**ClassOfClassOfUsageOfRepresentation**(x) ∨ **ClassOfUsageOfRepresentation**(x) ∨
UsageOfRepresentation(x))
ClassOfClassOfUsageOfRepresentation(x) ∧ **hasUser**(x, y) → **PossibleIndividual**(y)
ClassOfClassOfUsageOfRepresentation(x) → ∃y(**hasUser**(x, y))
ClassOfClassOfUsageOfRepresentation(x) ∧ **hasUser**(x, y) ∧ **hasUser**(x, z) → $y = z$
ClassOfUsageOfRepresentation(x) ∧ **hasUser**(x, y) → **PossibleIndividual**(y)
ClassOfUsageOfRepresentation(x) → ∃y(**hasUser**(x, y))
ClassOfUsageOfRepresentation(x) ∧ **hasUser**(x, y) ∧ **hasUser**(x, z) → $y = z$
UsageOfRepresentation(x) ∧ **hasUser**(x, y) → **PossibleIndividual**(y)
UsageOfRepresentation(x) → ∃y(**hasUser**(x, y))
UsageOfRepresentation(x) ∧ **hasUser**(x, y) ∧ **hasUser**(x, z) → $y = z$
hasWhole(x, y) → (**CompositionOfIndividual**(x))
CompositionOfIndividual(x) ∧ **hasWhole**(x, y) → **PossibleIndividual**(y)
CompositionOfIndividual(x) → ∃y(**hasWhole**(x, y))
CompositionOfIndividual(x) ∧ **hasWhole**(x, y) ∧ **hasWhole**(x, z) → $y = z$
hasWhyDeleted(x, y) → (**Thing**(x))
Thing(x) ∧ **hasWhyDeleted**(x, y) → **ClassOfInformationRepresentation**(y)
Thing(x) ∧ **hasWhyDeleted**(x, y) ∧ **hasWhyDeleted**(x, z) → $y = z$
hasYear(x, y) → (**RepresentationOfGregorianDateAndUtcTime**(x))
RepresentationOfGregorianDateAndUtcTime(x) ∧ **hasYear**(x, y) → **INTEGER**(y)
RepresentationOfGregorianDateAndUtcTime(x) → ∃y(**hasYear**(x, y))
RepresentationOfGregorianDateAndUtcTime(x) ∧ **hasYear**(x, y) ∧ **hasYear**(x, z) → $y = z$

B.7 Additional range restriction axioms

ClassOfParticipation(x) ∧ **hasClassOfPart**(x, y) → **ParticipatingRoleAndDomain**(y)
Namespace(x) ∧ **hasClassOfPart**(x, y) → **ClassOfInformationRepresentation**(y)
ClassOfArrangementOfIndividual(x) ∧ **hasClassOfWhole**(x, y) → **ClassOfArrangedIndividual**(y)
ClassOfParticipation(x) ∧ **hasClassOfWhole**(x, y) → **ClassOfActivity**(y)
Namespace(x) ∧ **hasClassOfWhole**(x, y) → **ClassOfInformationRepresentation**(y)
LowerBoundOfNumberRange(x) ∧ **hasClassified**(x, y) → **ArithmeticNumber**(y)
LowerBoundOfPropertyRange(x) ∧ **hasClassified**(x, y) → **Property**(y)
UpperBoundOfNumberRange(x) ∧ **hasClassified**(x, y) → **ArithmeticNumber**(y)
UpperBoundOfPropertyRange(x) ∧ **hasClassified**(x, y) → **Property**(y)
LowerBoundOfNumberRange(x) ∧ **hasClassifier**(x, y) → **NumberRange**(y)
LowerBoundOfPropertyRange(x) ∧ **hasClassifier**(x, y) → **PropertyRange**(y)
UpperBoundOfNumberRange(x) ∧ **hasClassifier**(x, y) → **NumberRange**(y)
UpperBoundOfPropertyRange(x) ∧ **hasClassifier**(x, y) → **PropertyRange**(y)
ClassOfScaleConversion(x) ∧ **hasCodomain**(x, y) → **Scale**(y)
Scale(x) ∧ **hasCodomain**(x, y) → **NumberSpace**(y)
ClassOfScaleConversion(x) ∧ **hasDomain**(x, y) → **Scale**(y)
Scale(x) ∧ **hasDomain**(x, y) → **PropertySpace**(y)
DifferenceOfSetOfClass(x) ∧ **hasInput**(x, y) → **EnumeratedSetOfClass**(y)
IntersectionOfSetOfClass(x) ∧ **hasInput**(x, y) → **EnumeratedSetOfClass**(y)

PropertyQuantification$(x) \wedge$ **hasInput**$(x, y) \rightarrow$ **Property**(y)
UnionOfSetOfClass$(x) \wedge$ **hasInput**$(x, y) \rightarrow$ **EnumeratedSetOfClass**(y)
TemporalBounding$(x) \wedge$ **hasPart**$(x, y) \rightarrow$ **Event**(y)
ClassOfDefinition$(x) \wedge$ **hasRepresented**$(x, y) \rightarrow$ **Class**(y)
Definition$(x) \wedge$ **hasRepresented**$(x, y) \rightarrow$ **Class**(y)
DifferenceOfSetOfClass$(x) \wedge$ **hasResult**$(x, y) \rightarrow$ **Class**(y)
IntersectionOfSetOfClass$(x) \wedge$ **hasResult**$(x, y) \rightarrow$ **Class**(y)
PropertyQuantification$(x) \wedge$ **hasResult**$(x, y) \rightarrow$ **ArithmeticNumber**(y)
UnionOfSetOfClass$(x) \wedge$ **hasResult**$(x, y) \rightarrow$ **Class**(y)
BoundaryOfNumberSpace$(x) \wedge$ **hasSubclass**$(x, y) \rightarrow$ **NumberSpace**(y)
BoundaryOfPropertySpace$(x) \wedge$ **hasSubclass**$(x, y) \rightarrow$ **PropertySpace**(y)
SpecializationByDomain$(x) \wedge$ **hasSubclass**$(x, y) \rightarrow$ **RoleAndDomain**(y)
SpecializationByRole$(x) \wedge$ **hasSubclass**$(x, y) \rightarrow$ **RoleAndDomain**(y)
SpecializationOfIndividualDimensionFromProperty$(x) \wedge$ **hasSubclass**$(x, y) \rightarrow$ **IndividualDimension**(y)
BoundaryOfNumberSpace$(x) \wedge$ **hasSuperclass**$(x, y) \rightarrow$ **NumberSpace**(y)
BoundaryOfPropertySpace$(x) \wedge$ **hasSuperclass**$(x, y) \rightarrow$ **PropertySpace**(y)
SpecializationByRole$(x) \wedge$ **hasSuperclass**$(x, y) \rightarrow$ **Role**(y)
SpecializationOfIndividualDimensionFromProperty$(x) \wedge$ **hasSuperclass**$(x, y) \rightarrow$ **Property**(y)
ArrangementOfIndividual$(x) \wedge$ **hasWhole**$(x, y) \rightarrow$ **ArrangedIndividual**(y)
Participation$(x) \wedge$ **hasWhole**$(x, y) \rightarrow$ **Activity**(y)

Annex C
(normative)
Listing: proto-templates

This annex lists the complete set of proto-templates defined in this part of ISO 15926. See also 4.3.

C.1 Proto-templates for relational entity types

ApprovalTriple$(x, y, z) \leftrightarrow$ **Approval**$(x) \wedge$ **hasApproved**$(x, y) \wedge$ **hasApprover**(x, z)
ApprovalTemplate$(y, z) \leftrightarrow \exists u($*ApprovalTriple*$(u, y, z))$
BoundaryOfNumberSpaceTriple$(x, y, z) \leftrightarrow$ **BoundaryOfNumberSpace**$(x) \wedge$ **hasSubclass**$(x, y) \wedge$ **hasSuperclass**(x, z)
BoundaryOfNumberSpaceTemplate$(y, z) \leftrightarrow \exists u($*BoundaryOfNumberSpaceTriple*$(u, y, z))$
BoundaryOfPropertySpaceTriple$(x, y, z) \leftrightarrow$ **BoundaryOfPropertySpace**$(x) \wedge$ **hasSubclass**$(x, y) \wedge$ **hasSuperclass**(x, z)
BoundaryOfPropertySpaceTemplate$(y, z) \leftrightarrow \exists u($*BoundaryOfPropertySpaceTriple*$(u, y, z))$
CauseOfEventTriple$(x, y, z) \leftrightarrow$ **CauseOfEvent**$(x) \wedge$ **hasCaused**$(x, y) \wedge$ **hasCauser**(x, z)
CauseOfEventTemplate$(y, z) \leftrightarrow \exists u($*CauseOfEventTriple*$(u, y, z))$
ClassOfApprovalTriple$(x, y, z) \leftrightarrow$ **ClassOfApproval**$(x) \wedge$ **hasClassOfApproved**$(x, y) \wedge$ **hasClassOfApprover**(x, z)
ClassOfApprovalTemplate$(y, z) \leftrightarrow \exists u($*ClassOfApprovalTriple*$(u, y, z))$
ClassOfCauseOfBeginningOfClassOfIndividualTriple$(x, y, z) \leftrightarrow$
ClassOfCauseOfBeginningOfClassOfIndividual$(x) \wedge$ **hasClassOfBegun**$(x, y) \wedge$ **hasClassOfCauser**(x, z)
ClassOfCauseOfBeginningOfClassOfIndividualTemplate$(y, z) \leftrightarrow$
$\exists u($*ClassOfCauseOfBeginningOfClassOfIndividualTriple*$(u, y, z))$
ClassOfCauseOfEndingOfClassOfIndividualTriple$(x, y, z) \leftrightarrow$
ClassOfCauseOfEndingOfClassOfIndividual$(x) \wedge$ **hasClassOfCauser**$(x, y) \wedge$ **hasClassOfEnded**(x, z)
ClassOfCauseOfEndingOfClassOfIndividualTemplate$(y, z) \leftrightarrow \exists u($*ClassOfCauseOfEndingOfClassOfIndividualTriple*$(u, y, z))$
ClassOfClassOfCompositionTriple$(x, y, z) \leftrightarrow$
ClassOfClassOfComposition$(x) \wedge$ **hasClassOfClassOfPart**$(x, y) \wedge$ **hasClassOfClassOfWhole**(x, z)
ClassOfClassOfCompositionTemplate$(y, z) \leftrightarrow \exists u($*ClassOfClassOfCompositionTriple*$(u, y, z))$
ClassOfClassOfRepresentationTriple$(x, y, z) \leftrightarrow$
ClassOfClassOfRepresentation$(x) \wedge$ **hasClassOfPattern**$(x, y) \wedge$ **hasClassOfRepresented**(x, z)
ClassOfClassOfRepresentationTemplate$(y, z) \leftrightarrow \exists u($*ClassOfClassOfRepresentationTriple*$(u, y, z))$
ClassOfClassOfRepresentationTranslationTriple$(x, y, z) \leftrightarrow$
ClassOfClassOfRepresentationTranslation$(x) \wedge$ **hasClassOfFirst**$(x, y) \wedge$ **hasClassOfSecond**(x, z)
ClassOfClassOfRepresentationTranslationTemplate$(y, z) \leftrightarrow \exists u($*ClassOfClassOfRepresentationTranslationTriple*$(u, y, z))$
ClassOfClassOfResponsibilityForRepresentationTriple$(x, y, z) \leftrightarrow$
ClassOfClassOfResponsibilityForRepresentation$(x) \wedge$**hasClassOfClassOfControlled**$(x, y) \wedge$**hasController**(x, z)
ClassOfClassOfResponsibilityForRepresentationTemplate$(y, z) \leftrightarrow$
$\exists u($*ClassOfClassOfResponsibilityForRepresentationTriple*$(u, y, z))$
ClassOfClassOfUsageOfRepresentationTriple$(x, y, z) \leftrightarrow$
ClassOfClassOfUsageOfRepresentation$(x) \wedge$ **hasClassOfClassOfUsed**$(x, y) \wedge$ **hasUser**(x, z)
ClassOfClassOfUsageOfRepresentationTemplate$(y, z) \leftrightarrow \exists u($*ClassOfClassOfUsageOfRepresentationTriple*$(u, y, z))$
ClassOfClassificationTriple$(x, y, z) \leftrightarrow$
ClassOfClassification$(x) \wedge$ **hasClassOfClassified**$(x, y) \wedge$ **hasClassOfClassifier**(x, z)
ClassOfClassificationTemplate$(y, z) \leftrightarrow \exists u($*ClassOfClassificationTriple*$(u, y, z))$
ClassOfCompositionOfIndividualTriple$(x, y, z) \leftrightarrow$
ClassOfCompositionOfIndividual$(x) \wedge$ **hasClassOfPart**$(x, y) \wedge$ **hasClassOfWhole**(x, z)
ClassOfCompositionOfIndividualTemplate$(y, z) \leftrightarrow \exists u($*ClassOfCompositionOfIndividualTriple*$(u, y, z))$
ClassOfConnectionOfIndividualTriple$(x, y, z) \leftrightarrow$
ClassOfConnectionOfIndividual$(x) \wedge$ **hasClassOfSide1**$(x, y) \wedge$ **hasClassOfSide2**(x, z)
ClassOfConnectionOfIndividualTemplate$(y, z) \leftrightarrow \exists u($*ClassOfConnectionOfIndividualTriple*$(u, y, z))$
ClassOfDimensionForShapeTriple$(x, y, z) \leftrightarrow$
ClassOfDimensionForShape$(x) \wedge$ **hasClassOfDimension**$(x, y) \wedge$ **hasClassOfShape**(x, z)
ClassOfDimensionForShapeTemplate$(y, z) \leftrightarrow \exists u($*ClassOfDimensionForShapeTriple*$(u, y, z))$
ClassOfFunctionalMappingTriple$(x, y, z) \leftrightarrow$ **ClassOfFunctionalMapping**$(x) \wedge$ **hasCodomain**$(x, y) \wedge$ **hasDomain**(x, z)
ClassOfFunctionalMappingTemplate$(y, z) \leftrightarrow \exists u($*ClassOfFunctionalMappingTriple*$(u, y, z))$
ClassOfIndirectPropertyTriple$(x, y, z) \leftrightarrow$
ClassOfIndirectProperty$(x) \wedge$ **hasClassOfPossessor**$(x, y) \wedge$ **hasPropertySpace**(x, z)
ClassOfIndirectPropertyTemplate$(y, z) \leftrightarrow \exists u($*ClassOfIndirectPropertyTriple*$(u, y, z))$
ClassOfIndividualUsedInConnectionTriple$(x, y, z) \leftrightarrow$
ClassOfIndividualUsedInConnection$(x) \wedge$ **hasClassOfConnection**$(x, y) \wedge$ **hasClassOfUsage**(x, z)
ClassOfIndividualUsedInConnectionTemplate$(y, z) \leftrightarrow \exists u($*ClassOfIndividualUsedInConnectionTriple*$(u, y, z))$
ClassOfIntendedRoleAndDomainTriple$(x, y, z) \leftrightarrow$
ClassOfIntendedRoleAndDomain$(x) \wedge$ **hasClassOfPlayer**$(x, y) \wedge$ **hasPlayed**(x, z)
ClassOfIntendedRoleAndDomainTemplate$(y, z) \leftrightarrow \exists u($*ClassOfIntendedRoleAndDomainTriple*$(u, y, z))$
ClassOfInvolvementByReferenceTriple$(x, y, z) \leftrightarrow$
ClassOfInvolvementByReference$(x) \wedge$ **hasClassOfInvolved**$(x, y) \wedge$ **hasClassOfInvolver**(x, z)
ClassOfInvolvementByReferenceTemplate$(y, z) \leftrightarrow \exists u($*ClassOfInvolvementByReferenceTriple*$(u, y, z))$
ClassOfNamespaceTriple$(x, y, z) \leftrightarrow$ **ClassOfNamespace**$(x) \wedge$ **hasClassOfClassOfWhole**$(x, y) \wedge$ **hasClassOfPart**(x, z)
ClassOfNamespaceTemplate$(y, z) \leftrightarrow \exists u($*ClassOfNamespaceTriple*$(u, y, z))$
ClassOfParticipationTriple$(x, y, z) \leftrightarrow$ **ClassOfParticipation**$(x) \wedge$ **hasClassOfPart**$(x, y) \wedge$ **hasClassOfWhole**(x, z)
ClassOfParticipationTemplate$(y, z) \leftrightarrow \exists u($*ClassOfParticipationTriple*$(u, y, z))$
ClassOfPossibleRoleAndDomainTriple$(x, y, z) \leftrightarrow$

ClassOfPossibleRoleAndDomain$(x) \wedge$ **hasClassOfPlayer**$(x, y) \wedge$ **hasPlayed**(x, z)
ClassOfPossibleRoleAndDomainTemplate$(y, z) \leftrightarrow \exists u($*ClassOfPossibleRoleAndDomainTriple*$(u, y, z))$
ClassOfRecognitionTriple$(x, y, z) \leftrightarrow$
ClassOfRecognition$(x) \wedge$ **hasClassOfRecognized**$(x, y) \wedge$ **hasClassOfRecognizing**(x, z)
ClassOfRecognitionTemplate$(y, z) \leftrightarrow \exists u($*ClassOfRecognitionTriple*$(u, y, z))$
ClassOfRelationshipWithSignatureTriple$(x, y, z) \leftrightarrow$
ClassOfRelationshipWithSignature$(x) \wedge$ **hasClassOfEnd1**$(x, y) \wedge$ **hasClassOfEnd2**(x, z)
ClassOfRelationshipWithSignatureTemplate$(y, z) \leftrightarrow \exists u($*ClassOfRelationshipWithSignatureTriple*$(u, y, z))$
ClassOfRelativeLocationTriple$(x, y, z) \leftrightarrow$
ClassOfRelativeLocation$(x) \wedge$ **hasClassOfLocated**$(x, y) \wedge$ **hasClassOfLocator**(x, z)
ClassOfRelativeLocationTemplate$(y, z) \leftrightarrow \exists u($*ClassOfRelativeLocationTriple*$(u, y, z))$
ClassOfRepresentationOfThingTriple$(x, y, z) \leftrightarrow$
ClassOfRepresentationOfThing$(x) \wedge$ **hasPattern**$(x, y) \wedge$ **hasRepresented**(x, z)
ClassOfRepresentationOfThingTemplate$(y, z) \leftrightarrow \exists u($*ClassOfRepresentationOfThingTriple*$(u, y, z))$
ClassOfRepresentationTranslationTriple$(x, y, z) \leftrightarrow$
ClassOfRepresentationTranslation$(x) \wedge$ **hasClassOfFirst**$(x, y) \wedge$ **hasClassOfSecond**(x, z)
ClassOfRepresentationTranslationTemplate$(y, z) \leftrightarrow \exists u($*ClassOfRepresentationTranslationTriple*$(u, y, z))$
ClassOfResponsibilityForRepresentationTriple$(x, y, z) \leftrightarrow$
ClassOfResponsibilityForRepresentation$(x) \wedge$ **hasClassOfControlled**$(x, y) \wedge$ **hasController**(x, z)
ClassOfResponsibilityForRepresentationTemplate$(y, z) \leftrightarrow \exists u($*ClassOfResponsibilityForRepresentationTriple*$(u, y, z))$
ClassOfScaleConversionTriple$(x, y, z) \leftrightarrow$ **ClassOfScaleConversion**$(x) \wedge$ **hasCodomain**$(x, y) \wedge$ **hasDomain**(x, z)
ClassOfScaleConversionTemplate$(y, z) \leftrightarrow \exists u($*ClassOfScaleConversionTriple*$(u, y, z))$
ClassOfSpecializationTriple$(x, y, z) \leftrightarrow$
ClassOfSpecialization$(x) \wedge$ **hasClassOfSubclass**$(x, y) \wedge$ **hasClassOfSuperclass**(x, z)
ClassOfSpecializationTemplate$(y, z) \leftrightarrow \exists u($*ClassOfSpecializationTriple*$(u, y, z))$
ClassOfTemporalSequenceTriple$(x, y, z) \leftrightarrow$
ClassOfTemporalSequence$(x) \wedge$ **hasClassOfPredecessor**$(x, y) \wedge$ **hasClassOfSuccessor**(x, z)
ClassOfTemporalSequenceTemplate$(y, z) \leftrightarrow \exists u($*ClassOfTemporalSequenceTriple*$(u, y, z))$
ClassOfUsageOfRepresentationTriple$(x, y, z) \leftrightarrow$
ClassOfUsageOfRepresentation$(x) \wedge$ **hasClassOfUsed**$(x, y) \wedge$ **hasUser**(x, z)
ClassOfUsageOfRepresentationTemplate$(y, z) \leftrightarrow \exists u($*ClassOfUsageOfRepresentationTriple*$(u, y, z))$
ClassificationTriple$(x, y, z) \leftrightarrow$ **Classification**$(x) \wedge$ **hasClassified**$(x, y) \wedge$ **hasClassifier**(x, z)
ClassificationTemplate$(y, z) \leftrightarrow \exists u($*ClassificationTriple*$(u, y, z))$
ComparisonOfPropertyTriple$(x, y, z) \leftrightarrow$
ComparisonOfProperty$(x) \wedge$ **hasGreaterElement**$(x, y) \wedge$ **hasLesserElement**(x, z)
ComparisonOfPropertyTemplate$(y, z) \leftrightarrow \exists u($*ComparisonOfPropertyTriple*$(u, y, z))$
CompositionOfIndividualTriple$(x, y, z) \leftrightarrow$ **CompositionOfIndividual**$(x) \wedge$ **hasPart**$(x, y) \wedge$ **hasWhole**(x, z)
CompositionOfIndividualTemplate$(y, z) \leftrightarrow \exists u($*CompositionOfIndividualTriple*$(u, y, z))$
ConnectionOfIndividualTriple$(x, y, z) \leftrightarrow$ **ConnectionOfIndividual**$(x) \wedge$ **hasSide1**$(x, y) \wedge$ **hasSide2**(x, z)
ConnectionOfIndividualTemplate$(y, z) \leftrightarrow \exists u($*ConnectionOfIndividualTriple*$(u, y, z))$
DifferenceOfSetOfClassTriple$(x, y, z) \leftrightarrow$ **DifferenceOfSetOfClass**$(x) \wedge$ **hasInput**$(x, y) \wedge$ **hasResult**(x, z)
DifferenceOfSetOfClassTemplate$(y, z) \leftrightarrow \exists u($*DifferenceOfSetOfClassTriple*$(u, y, z))$
DimensionOfIndividualTriple$(x, y, z) \leftrightarrow$
DimensionOfIndividual$(x) \wedge$ **hasIndividual**$(x, y) \wedge$ **hasIndividualDimension**(x, z)
DimensionOfIndividualTemplate$(y, z) \leftrightarrow \exists u($*DimensionOfIndividualTriple*$(u, y, z))$
DimensionOfShapeTriple$(x, y, z) \leftrightarrow$ **DimensionOfShape**$(x) \wedge$ **hasDimension**$(x, y) \wedge$ **hasShape**(x, z)
DimensionOfShapeTemplate$(y, z) \leftrightarrow \exists u($*DimensionOfShapeTriple*$(u, y, z))$
FunctionalMappingTriple$(x, y, z) \leftrightarrow$ **FunctionalMapping**$(x) \wedge$ **hasInput**$(x, y) \wedge$ **hasResult**(x, z)
FunctionalMappingTemplate$(y, z) \leftrightarrow \exists u($*FunctionalMappingTriple*$(u, y, z))$
IndirectPropertyTriple$(x, y, z) \leftrightarrow$ **IndirectProperty**$(x) \wedge$ **hasPossessor**$(x, y) \wedge$ **hasProperty**(x, z)
IndirectPropertyTemplate$(y, z) \leftrightarrow \exists u($*IndirectPropertyTriple*$(u, y, z))$
IndividualUsedInConnectionTriple$(x, y, z) \leftrightarrow$ **IndividualUsedInConnection**$(x) \wedge$ **hasConnection**$(x, y) \wedge$ **hasUsage**(x, z)
IndividualUsedInConnectionTemplate$(y, z) \leftrightarrow \exists u($*IndividualUsedInConnectionTriple*$(u, y, z))$
IntendedRoleAndDomainTriple$(x, y, z) \leftrightarrow$ **IntendedRoleAndDomain**$(x) \wedge$ **hasPlayed**$(x, y) \wedge$ **hasPlayer**(x, z)
IntendedRoleAndDomainTemplate$(y, z) \leftrightarrow \exists u($*IntendedRoleAndDomainTriple*$(u, y, z))$
IntersectionOfSetOfClassTriple$(x, y, z) \leftrightarrow$ **IntersectionOfSetOfClass**$(x) \wedge$ **hasInput**$(x, y) \wedge$ **hasResult**(x, z)
IntersectionOfSetOfClassTemplate$(y, z) \leftrightarrow \exists u($*IntersectionOfSetOfClassTriple*$(u, y, z))$
InvolvementByReferenceTriple$(x, y, z) \leftrightarrow$ **InvolvementByReference**$(x) \wedge$ **hasInvolved**$(x, y) \wedge$ **hasInvolver**(x, z)
InvolvementByReferenceTemplate$(y, z) \leftrightarrow \exists u($*InvolvementByReferenceTriple*$(u, y, z))$
LifecycleStageTriple$(x, y, z) \leftrightarrow$ **LifecycleStage**$(x) \wedge$ **hasInterest**$(x, y) \wedge$ **hasInterested**(x, z)
LifecycleStageTemplate$(y, z) \leftrightarrow \exists u($*LifecycleStageTriple*$(u, y, z))$
LowerBoundOfNumberRangeTriple$(x, y, z) \leftrightarrow$
LowerBoundOfNumberRange$(x) \wedge$ **hasClassified**$(x, y) \wedge$ **hasClassifier**(x, z)
LowerBoundOfNumberRangeTemplate$(y, z) \leftrightarrow \exists u($*LowerBoundOfNumberRangeTriple*$(u, y, z))$
LowerBoundOfPropertyRangeTriple$(x, y, z) \leftrightarrow$

LowerBoundOfPropertyRange$(x) \wedge$ **hasClassified**$(x, y) \wedge$ **hasClassifier**(x, z)
LowerBoundOfPropertyRangeTemplate$(y, z) \leftrightarrow \exists u($*LowerBoundOfPropertyRangeTriple*$(u, y, z))$
OtherRelationshipTriple$(x, y, z) \leftrightarrow$ **OtherRelationship**$(x) \wedge$ **hasEnd1**$(x, y) \wedge$ **hasEnd2**(x, z)
OtherRelationshipTemplate$(y, z) \leftrightarrow \exists u($*OtherRelationshipTriple*$(u, y, z))$
PossibleRoleAndDomainTriple$(x, y, z) \leftrightarrow$ **PossibleRoleAndDomain**$(x) \wedge$ **hasPlayed**$(x, y) \wedge$ **hasPlayer**(x, z)
PossibleRoleAndDomainTemplate$(y, z) \leftrightarrow \exists u($*PossibleRoleAndDomainTriple*$(u, y, z))$
PropertyForShapeDimensionTriple$(x, y, z) \leftrightarrow$
PropertyForShapeDimension$(x) \wedge$ **hasProperty**$(x, y) \wedge$ **hasShapeDimension**(x, z)
PropertyForShapeDimensionTemplate$(y, z) \leftrightarrow \exists u($*PropertyForShapeDimensionTriple*$(u, y, z))$
PropertyQuantificationTriple$(x, y, z) \leftrightarrow$ **PropertyQuantification**$(x) \wedge$ **hasInput**$(x, y) \wedge$ **hasResult**(x, z)
PropertyQuantificationTemplate$(y, z) \leftrightarrow \exists u($*PropertyQuantificationTriple*$(u, y, z))$
PropertySpaceForClassOfShapeDimensionTriple$(x, y, z) \leftrightarrow$
PropertySpaceForClassOfShapeDimension$(x) \wedge$ **hasClassOfShapeDimension**$(x, y) \wedge$ **hasPropertySpace**(x, z)
PropertySpaceForClassOfShapeDimensionTemplate$(y, z) \leftrightarrow \exists u($*PropertySpaceForClassOfShapeDimensionTriple*$(u, y, z))$
RecognitionTriple$(x, y, z) \leftrightarrow$ **Recognition**$(x) \wedge$ **hasRecognized**$(x, y) \wedge$ **hasRecognizing**(x, z)
RecognitionTemplate$(y, z) \leftrightarrow \exists u($*RecognitionTriple*$(u, y, z))$
RelativeLocationTriple$(x, y, z) \leftrightarrow$ **RelativeLocation**$(x) \wedge$ **hasLocated**$(x, y) \wedge$ **hasLocator**(x, z)
RelativeLocationTemplate$(y, z) \leftrightarrow \exists u($*RelativeLocationTriple*$(u, y, z))$
RepresentationOfThingTriple$(x, y, z) \leftrightarrow$ **RepresentationOfThing**$(x) \wedge$ **hasRepresented**$(x, y) \wedge$ **hasSign**(x, z)
RepresentationOfThingTemplate$(y, z) \leftrightarrow \exists u($*RepresentationOfThingTriple*$(u, y, z))$
ResponsibilityForRepresentationTriple$(x, y, z) \leftrightarrow$
ResponsibilityForRepresentation$(x) \wedge$ **hasControlled**$(x, y) \wedge$ **hasController**(x, z)
ResponsibilityForRepresentationTemplate$(y, z) \leftrightarrow \exists u($*ResponsibilityForRepresentationTriple*$(u, y, z))$
ScaleTriple$(x, y, z) \leftrightarrow$ **Scale**$(x) \wedge$ **hasCodomain**$(x, y) \wedge$ **hasDomain**(x, z) *ScaleTemplate*$(y, z) \leftrightarrow \exists u($*ScaleTriple*$(u, y, z))$
SpecializationTriple$(x, y, z) \leftrightarrow$ **Specialization**$(x) \wedge$ **hasSubclass**$(x, y) \wedge$ **hasSuperclass**(x, z)
SpecializationTemplate$(y, z) \leftrightarrow \exists u($*SpecializationTriple*$(u, y, z))$
SpecializationByRoleTriple$(x, y, z) \leftrightarrow$ **SpecializationByRole**$(x) \wedge$ **hasSubclass**$(x, y) \wedge$ **hasSuperclass**(x, z)
SpecializationByRoleTemplate$(y, z) \leftrightarrow \exists u($*SpecializationByRoleTriple*$(u, y, z))$
SpecializationOfIndividualDimensionFromPropertyTriple$(x, y, z) \leftrightarrow$
SpecializationOfIndividualDimensionFromProperty$(x) \wedge$ **hasSubclass**$(x, y) \wedge$ **hasSuperclass**(x, z)
SpecializationOfIndividualDimensionFromPropertyTemplate$(y, z) \leftrightarrow$
$\exists u($*SpecializationOfIndividualDimensionFromPropertyTriple*$(u, y, z))$
TemporalSequenceTriple$(x, y, z) \leftrightarrow$ **TemporalSequence**$(x) \wedge$ **hasPredecessor**$(x, y) \wedge$ **hasSuccessor**(x, z)
TemporalSequenceTemplate$(y, z) \leftrightarrow \exists u($*TemporalSequenceTriple*$(u, y, z))$
UnionOfSetOfClassTriple$(x, y, z) \leftrightarrow$ **UnionOfSetOfClass**$(x) \wedge$ **hasInput**$(x, y) \wedge$ **hasResult**(x, z)
UnionOfSetOfClassTemplate$(y, z) \leftrightarrow \exists u($*UnionOfSetOfClassTriple*$(u, y, z))$
UpperBoundOfNumberRangeTriple$(x, y, z) \leftrightarrow$
UpperBoundOfNumberRange$(x) \wedge$ **hasClassified**$(x, y) \wedge$ **hasClassifier**(x, z)
UpperBoundOfNumberRangeTemplate$(y, z) \leftrightarrow \exists u($*UpperBoundOfNumberRangeTriple*$(u, y, z))$
UpperBoundOfPropertyRangeTriple$(x, y, z) \leftrightarrow$
UpperBoundOfPropertyRange$(x) \wedge$ **hasClassified**$(x, y) \wedge$ **hasClassifier**(x, z)
UpperBoundOfPropertyRangeTemplate$(y, z) \leftrightarrow \exists u($*UpperBoundOfPropertyRangeTriple*$(u, y, z))$
UsageOfRepresentationTriple$(x, y, z) \leftrightarrow$ **UsageOfRepresentation**$(x) \wedge$ **hasUsed**$(x, y) \wedge$ **hasUser**(x, z)
UsageOfRepresentationTemplate$(y, z) \leftrightarrow \exists u($*UsageOfRepresentationTriple*$(u, y, z))$

C.2 Proto-templates for subtypes of relational entity types

ArrangementOfIndividualTriple$(x, y, z) \leftrightarrow$ **ArrangementOfIndividual**$(x) \wedge$ *CompositionOfIndividualTriple*(x, y, z)
ArrangementOfIndividualTemplate$(y, z) \leftrightarrow \exists u($*ArrangementOfIndividualTriple*$(u, y, z))$
AssemblyOfIndividualTriple$(x, y, z) \leftrightarrow$ **AssemblyOfIndividual**$(x) \wedge$ *ArrangementOfIndividualTriple*(x, y, z)
AssemblyOfIndividualTemplate$(y, z) \leftrightarrow \exists u($*AssemblyOfIndividualTriple*$(u, y, z))$
BeginningTriple$(x, y, z) \leftrightarrow$ **Beginning**$(x) \wedge$ *TemporalBoundingTriple*(x, y, z)
BeginningTemplate$(y, z) \leftrightarrow \exists u($*BeginningTriple*$(u, y, z))$
ClassOfArrangementOfIndividualTriple$(x, y, z) \leftrightarrow$
ClassOfArrangementOfIndividual$(x) \wedge$ *ClassOfCompositionOfIndividualTriple*(x, y, z)
ClassOfArrangementOfIndividualTemplate$(y, z) \leftrightarrow \exists u($*ClassOfArrangementOfIndividualTriple*$(u, y, z))$
ClassOfAssemblyOfIndividualTriple$(x, y, z) \leftrightarrow$
ClassOfAssemblyOfIndividual$(x) \wedge$ *ClassOfArrangementOfIndividualTriple*(x, y, z)
ClassOfAssemblyOfIndividualTemplate$(y, z) \leftrightarrow \exists u($*ClassOfAssemblyOfIndividualTriple*$(u, y, z))$
ClassOfClassOfDefinitionTriple$(x, y, z) \leftrightarrow$ **ClassOfClassOfDefinition**$(x) \wedge$ *ClassOfClassOfRepresentationTriple*(x, y, z)
ClassOfClassOfDefinitionTemplate$(y, z) \leftrightarrow \exists u($*ClassOfClassOfDefinitionTriple*$(u, y, z))$
ClassOfClassOfDescriptionTriple$(x, y, z) \leftrightarrow$ **ClassOfClassOfDescription**$(x) \wedge$ *ClassOfClassOfRepresentationTriple*(x, y, z)
ClassOfClassOfDescriptionTemplate$(y, z) \leftrightarrow \exists u($*ClassOfClassOfDescriptionTriple*$(u, y, z))$
ClassOfClassOfIdentificationTriple$(x, y, z) \leftrightarrow$
ClassOfClassOfIdentification$(x) \wedge$ *ClassOfClassOfRepresentationTriple*(x, y, z)
ClassOfClassOfIdentificationTemplate$(y, z) \leftrightarrow \exists u($*ClassOfClassOfIdentificationTriple*$(u, y, z))$

ClassOfClassOfRelationshipWithSignatureTriple$(x, y, z) \leftrightarrow$
ClassOfClassOfRelationshipWithSignature$(x) \wedge$ *ClassOfRelationshipWithSignatureTriple*(x, y, z)
ClassOfClassOfRelationshipWithSignatureTemplate$(y, z) \leftrightarrow \exists u($*ClassOfClassOfRelationshipWithSignatureTriple*$(u, y, z))$
ClassOfContainmentOfIndividualTriple$(x, y, z) \leftrightarrow$
ClassOfContainmentOfIndividual$(x) \wedge$ *ClassOfRelativeLocationTriple*(x, y, z)
ClassOfContainmentOfIndividualTemplate$(y, z) \leftrightarrow \exists u($*ClassOfContainmentOfIndividualTriple*$(u, y, z))$
ClassOfDefinitionTriple$(x, y, z) \leftrightarrow$ **ClassOfDefinition**$(x) \wedge$ *ClassOfRepresentationOfThingTriple*(x, y, z)
ClassOfDefinitionTemplate$(y, z) \leftrightarrow \exists u($*ClassOfDefinitionTriple*$(u, y, z))$
ClassOfDescriptionTriple$(x, y, z) \leftrightarrow$ **ClassOfDescription**$(x) \wedge$ *ClassOfRepresentationOfThingTriple*(x, y, z)
ClassOfDescriptionTemplate$(y, z) \leftrightarrow \exists u($*ClassOfDescriptionTriple*$(u, y, z))$
ClassOfDirectConnectionTriple$(x, y, z) \leftrightarrow$ **ClassOfDirectConnection**$(x) \wedge$ *ClassOfConnectionOfIndividualTriple*(x, y, z)
ClassOfDirectConnectionTemplate$(y, z) \leftrightarrow \exists u($*ClassOfDirectConnectionTriple*$(u, y, z))$
ClassOfFeatureWholePartTriple$(x, y, z) \leftrightarrow$ **ClassOfFeatureWholePart**$(x) \wedge$ *ClassOfArrangementOfIndividualTriple*(x, y, z)
ClassOfFeatureWholePartTemplate$(y, z) \leftrightarrow \exists u($*ClassOfFeatureWholePartTriple*$(u, y, z))$
ClassOfIdentificationTriple$(x, y, z) \leftrightarrow$ **ClassOfIdentification**$(x) \wedge$ *ClassOfRepresentationOfThingTriple*(x, y, z)
ClassOfIdentificationTemplate$(y, z) \leftrightarrow \exists u($*ClassOfIdentificationTriple*$(u, y, z))$
ClassOfIndirectConnectionTriple$(x, y, z) \leftrightarrow$
ClassOfIndirectConnection$(x) \wedge$ *ClassOfConnectionOfIndividualTriple*(x, y, z)
ClassOfIndirectConnectionTemplate$(y, z) \leftrightarrow \exists u($*ClassOfIndirectConnectionTriple*$(u, y, z))$
ClassOfIsomorphicFunctionalMappingTriple$(x, y, z) \leftrightarrow$
ClassOfIsomorphicFunctionalMapping$(x) \wedge$ *ClassOfFunctionalMappingTriple*(x, y, z)
ClassOfIsomorphicFunctionalMappingTemplate$(y, z) \leftrightarrow \exists u($*ClassOfIsomorphicFunctionalMappingTriple*$(u, y, z))$
ClassOfLeftNamespaceTriple$(x, y, z) \leftrightarrow$ **ClassOfLeftNamespace**$(x) \wedge$ *ClassOfNamespaceTriple*(x, y, z)
ClassOfLeftNamespaceTemplate$(y, z) \leftrightarrow \exists u($*ClassOfLeftNamespaceTriple*$(u, y, z))$
ClassOfRightNamespaceTriple$(x, y, z) \leftrightarrow$ **ClassOfRightNamespace**$(x) \wedge$ *ClassOfNamespaceTriple*(x, y, z)
ClassOfRightNamespaceTemplate$(y, z) \leftrightarrow \exists u($*ClassOfRightNamespaceTriple*$(u, y, z))$
ClassOfTemporalWholePartTriple$(x, y, z) \leftrightarrow$
ClassOfTemporalWholePart$(x) \wedge$ *ClassOfCompositionOfIndividualTriple*(x, y, z)
ClassOfTemporalWholePartTemplate$(y, z) \leftrightarrow \exists u($*ClassOfTemporalWholePartTriple*$(u, y, z))$
ContainmentOfIndividualTriple$(x, y, z) \leftrightarrow$ **ContainmentOfIndividual**$(x) \wedge$ *RelativeLocationTriple*(x, y, z)
ContainmentOfIndividualTemplate$(y, z) \leftrightarrow \exists u($*ContainmentOfIndividualTriple*$(u, y, z))$
CoordinateSystemTriple$(x, y, z) \leftrightarrow$ **CoordinateSystem**$(x) \wedge$ *MultidimensionalScaleTriple*(x, y, z)
CoordinateSystemTemplate$(y, z) \leftrightarrow \exists u($*CoordinateSystemTriple*$(u, y, z))$
DefinitionTriple$(x, y, z) \leftrightarrow$ **Definition**$(x) \wedge$ *RepresentationOfThingTriple*(x, y, z)
DefinitionTemplate$(y, z) \leftrightarrow \exists u($*DefinitionTriple*$(u, y, z))$
DescriptionTriple$(x, y, z) \leftrightarrow$ **Description**$(x) \wedge$ *RepresentationOfThingTriple*(x, y, z)
DescriptionTemplate$(y, z) \leftrightarrow \exists u($*DescriptionTriple*$(u, y, z))$
DirectConnectionTriple$(x, y, z) \leftrightarrow$ **DirectConnection**$(x) \wedge$ *ConnectionOfIndividualTriple*(x, y, z)
DirectConnectionTemplate$(y, z) \leftrightarrow \exists u($*DirectConnectionTriple*$(u, y, z))$
EndingTriple$(x, y, z) \leftrightarrow$ **Ending**$(x) \wedge$ *TemporalBoundingTriple*(x, y, z) *EndingTemplate*$(y, z) \leftrightarrow \exists u($*EndingTriple*$(u, y, z))$
FeatureWholePartTriple$(x, y, z) \leftrightarrow$ **FeatureWholePart**$(x) \wedge$ *ArrangementOfIndividualTriple*(x, y, z)
FeatureWholePartTemplate$(y, z) \leftrightarrow \exists u($*FeatureWholePartTriple*$(u, y, z))$
IdentificationTriple$(x, y, z) \leftrightarrow$ **Identification**$(x) \wedge$ *RepresentationOfThingTriple*(x, y, z)
IdentificationTemplate$(y, z) \leftrightarrow \exists u($*IdentificationTriple*$(u, y, z))$
IndirectConnectionTriple$(x, y, z) \leftrightarrow$ **IndirectConnection**$(x) \wedge$ *ConnectionOfIndividualTriple*(x, y, z)
IndirectConnectionTemplate$(y, z) \leftrightarrow \exists u($*IndirectConnectionTriple*$(u, y, z))$
LeftNamespaceTriple$(x, y, z) \leftrightarrow$ **LeftNamespace**$(x) \wedge$ *NamespaceTriple*(x, y, z)
LeftNamespaceTemplate$(y, z) \leftrightarrow \exists u($*LeftNamespaceTriple*$(u, y, z))$
MultidimensionalScaleTriple$(x, y, z) \leftrightarrow$ **MultidimensionalScale**$(x) \wedge$ *ScaleTriple*(x, y, z)
MultidimensionalScaleTemplate$(y, z) \leftrightarrow \exists u($*MultidimensionalScaleTriple*$(u, y, z))$
NamespaceTriple$(x, y, z) \leftrightarrow$ **Namespace**$(x) \wedge$ *ClassOfArrangementOfIndividualTriple*(x, y, z)
NamespaceTemplate$(y, z) \leftrightarrow \exists u($*NamespaceTriple*$(u, y, z))$
ParticipationTriple$(x, y, z) \leftrightarrow$ **Participation**$(x) \wedge$ *CompositionOfIndividualTriple*(x, y, z)
ParticipationTemplate$(y, z) \leftrightarrow \exists u($*ParticipationTriple*$(u, y, z))$
RightNamespaceTriple$(x, y, z) \leftrightarrow$ **RightNamespace**$(x) \wedge$ *NamespaceTriple*(x, y, z)
RightNamespaceTemplate$(y, z) \leftrightarrow \exists u($*RightNamespaceTriple*$(u, y, z))$
SpecializationByDomainTriple$(x, y, z) \leftrightarrow$ **SpecializationByDomain**$(x) \wedge$ *SpecializationTriple*(x, y, z)
SpecializationByDomainTemplate$(y, z) \leftrightarrow \exists u($*SpecializationByDomainTriple*$(u, y, z))$
TemporalBoundingTriple$(x, y, z) \leftrightarrow$ **TemporalBounding**$(x) \wedge$ *CompositionOfIndividualTriple*(x, y, z)
TemporalBoundingTemplate$(y, z) \leftrightarrow \exists u($*TemporalBoundingTriple*$(u, y, z))$
TemporalWholePartTriple$(x, y, z) \leftrightarrow$ **TemporalWholePart**$(x) \wedge$ *CompositionOfIndividualTriple*(x, y, z)
TemporalWholePartTemplate$(y, z) \leftrightarrow \exists u($*TemporalWholePartTriple*$(u, y, z))$

C.3 entityTriple

$entityTriple(x, y, z) \leftrightarrow (ApprovalTriple(x, y, z) \vee ArrangementOfIndividualTriple(x, y, z) \vee AssemblyOfIndividualTriple(x, y, z) \vee BeginningTriple(x, y, z) \vee BoundaryOfNumberSpaceTriple(x, y, z) \vee BoundaryOfPropertySpaceTriple(x, y, z) \vee CauseOfEventTriple(x, y, z) \vee ClassificationTriple(x, y, z) \vee ClassOfApprovalTriple(x, y, z) \vee ClassOfArrangementOfIndividualTriple(x, y, z) \vee ClassOfAssemblyOfIndividualTriple(x, y, z) \vee ClassOfCauseOfBeginningOfClassOfIndividualTriple(x, y, z) \vee ClassOfCauseOfEndingOfClassOfIndividualTriple(x, y, z) \vee ClassOfClassificationTriple(x, y, z) \vee ClassOfClassOfCompositionTriple(x, y, z) \vee ClassOfClassOfDefinitionTriple(x, y, z) \vee ClassOfClassOfDescriptionTriple(x, y, z) \vee ClassOfClassOfIdentificationTriple(x, y, z) \vee ClassOfClassOfRelationshipWithSignatureTriple(x, y, z) \vee ClassOfClassOfRepresentationTranslationTriple(x, y, z) \vee ClassOfClassOfRepresentationTriple(x, y, z) \vee ClassOfClassOfResponsibilityForRepresentationTriple(x, y, z) \vee ClassOfClassOfUsageOfRepresentationTriple(x, y, z) \vee ClassOfCompositionOfIndividualTriple(x, y, z) \vee ClassOfConnectionOfIndividualTriple(x, y, z) \vee ClassOfContainmentOfIndividualTriple(x, y, z) \vee ClassOfDefinitionTriple(x, y, z) \vee ClassOfDescriptionTriple(x, y, z) \vee ClassOfDimensionForShapeTriple(x, y, z) \vee ClassOfDirectConnectionTriple(x, y, z) \vee ClassOfFeatureWholePartTriple(x, y, z) \vee ClassOfFunctionalMappingTriple(x, y, z) \vee ClassOfIdentificationTriple(x, y, z) \vee ClassOfIndirectConnectionTriple(x, y, z) \vee ClassOfIndirectPropertyTriple(x, y, z) \vee ClassOfIndividualUsedInConnectionTriple(x, y, z) \vee ClassOfIntendedRoleAndDomainTriple(x, y, z) \vee ClassOfInvolvementByReferenceTriple(x, y, z) \vee ClassOfIsomorphicFunctionalMappingTriple(x, y, z) \vee ClassOfLeftNamespaceTriple(x, y, z) \vee ClassOfNamespaceTriple(x, y, z) \vee ClassOfParticipationTriple(x, y, z) \vee ClassOfPossibleRoleAndDomainTriple(x, y, z) \vee ClassOfRecognitionTriple(x, y, z) \vee ClassOfRelationshipWithSignatureTriple(x, y, z) \vee ClassOfRelativeLocationTriple(x, y, z) \vee ClassOfRepresentationOfThingTriple(x, y, z) \vee ClassOfRepresentationTranslationTriple(x, y, z) \vee ClassOfResponsibilityForRepresentationTriple(x, y, z) \vee ClassOfRightNamespaceTriple(x, y, z) \vee ClassOfScaleConversionTriple(x, y, z) \vee ClassOfSpecializationTriple(x, y, z) \vee ClassOfTemporalSequenceTriple(x, y, z) \vee ClassOfTemporalWholePartTriple(x, y, z) \vee ClassOfUsageOfRepresentationTriple(x, y, z) \vee ComparisonOfPropertyTriple(x, y, z) \vee CompositionOfIndividualTriple(x, y, z) \vee ConnectionOfIndividualTriple(x, y, z) \vee ContainmentOfIndividualTriple(x, y, z) \vee CoordinateSystemTriple(x, y, z) \vee DefinitionTriple(x, y, z) \vee DescriptionTriple(x, y, z) \vee DifferenceOfSetOfClassTriple(x, y, z) \vee DimensionOfIndividualTriple(x, y, z) \vee DimensionOfShapeTriple(x, y, z) \vee DirectConnectionTriple(x, y, z) \vee EndingTriple(x, y, z) \vee FeatureWholePartTriple(x, y, z) \vee FunctionalMappingTriple(x, y, z) \vee IdentificationTriple(x, y, z) \vee IndirectConnectionTriple(x, y, z) \vee IndirectPropertyTriple(x, y, z) \vee IndividualUsedInConnectionTriple(x, y, z) \vee IntendedRoleAndDomainTriple(x, y, z) \vee IntersectionOfSetOfClassTriple(x, y, z) \vee InvolvementByReferenceTriple(x, y, z) \vee LeftNamespaceTriple(x, y, z) \vee LifecycleStageTriple(x, y, z) \vee LowerBoundOfNumberRangeTriple(x, y, z) \vee LowerBoundOfPropertyRangeTriple(x, y, z) \vee MultidimensionalScaleTriple(x, y, z) \vee NamespaceTriple(x, y, z) \vee OtherRelationshipTriple(x, y, z) \vee ParticipationTriple(x, y, z) \vee PossibleRoleAndDomainTriple(x, y, z) \vee PropertyForShapeDimensionTriple(x, y, z) \vee PropertyQuantificationTriple(x, y, z) \vee PropertySpaceForClassOfShapeDimensionTriple(x, y, z) \vee RecognitionTriple(x, y, z) \vee RelativeLocationTriple(x, y, z) \vee RepresentationOfThingTriple(x, y, z) \vee ResponsibilityForRepresentationTriple(x, y, z) \vee RightNamespaceTriple(x, y, z) \vee ScaleTriple(x, y, z) \vee SpecializationByDomainTriple(x, y, z) \vee SpecializationByRoleTriple(x, y, z) \vee SpecializationOfIndividualDimensionFromPropertyTriple(x, y, z) \vee SpecializationTriple(x, y, z) \vee TemporalBoundingTriple(x, y, z) \vee TemporalSequenceTriple(x, y, z) \vee TemporalWholePartTriple(x, y, z) \vee UnionOfSetOfClassTriple(x, y, z) \vee UpperBoundOfNumberRangeTriple(x, y, z) \vee UpperBoundOfPropertyRangeTriple(x, y, z) \vee UsageOfRepresentationTriple(x, y, z))$

Annex D
(informative)
Table: proto-templates

Table D.1 lists all proto-templates and their roles. The format of the table is

Proto-template	First role	Second role
proto-template name	**role name** **type for role**	**role name** **type for role**

EXAMPLE 1 The listing of the proto-template *ApprovalTemplate* is as follows.

Proto-template	First role	Second role
ApprovalTemplate	**hasApproved** **Relationship**	**hasApprover** **PossibleIndividual**

This means that the first role of the template is **hasApproved** and instances of the first role must be instances of the entity type **Relationship**. The second role is **hasApprover** and instances of this role must be instances of the entity type **PossibleIndividual**.

Table D.1 – Compact listing of proto-templates

Proto-template	First role	Second role
ApprovalTemplate	**hasApproved Relationship**	**hasApprover PossibleIndividual**
ArrangementOfIndividualTemplate	**hasPart PossibleIndividual**	**hasWhole ArrangedIndividual**
AssemblyOfIndividualTemplate	**hasPart PossibleIndividual**	**hasWhole ArrangedIndividual**
BeginningTemplate	**hasPart Event**	**hasWhole PossibleIndividual**
BoundaryOfNumberSpaceTemplate	**hasSubclass NumberSpace**	**hasSuperclass NumberSpace**
BoundaryOfPropertySpaceTemplate	**hasSubclass PropertySpace**	**hasSuperclass PropertySpace**
CauseOfEventTemplate	**hasCaused Event**	**hasCauser Activity**
ClassOfApprovalTemplate	**hasClassOfApproved ClassOfRelationship**	**hasClassOfApprover ClassOfIndividual**
ClassOfArrangementOfIndividualTemplate	**hasClassOfPart ClassOfIndividual**	**hasClassOfWhole ClassOfArrangedIndividual**
ClassOfAssemblyOfIndividualTemplate	**hasClassOfPart ClassOfIndividual**	**hasClassOfWhole ClassOfArrangedIndividual**
ClassOfCauseOfBeginningOfClassOfIndividualTemplate	**hasClassOfBegun ClassOfIndividual**	**hasClassOfCauser ClassOfActivity**
ClassOfCauseOfEndingOfClassOfIndividualTemplate	**hasClassOfCauser ClassOfActivity**	**hasClassOfEnded ClassOfIndividual**
ClassOfClassOfCompositionTemplate	**hasClassOfClassOfPart ClassOfClassOfIndividual**	**hasClassOfClassOfWhole ClassOfClassOfIndividual**
ClassOfClassOfDefinitionTemplate	**hasClassOfPattern ClassOfClassOfInformationRepresentation**	**hasClassOfRepresented Class**
ClassOfClassOfDescriptionTemplate	**hasClassOfPattern ClassOfClassOfInformationRepresentation**	**hasClassOfRepresented Class**
ClassOfClassOfIdentificationTemplate	**hasClassOfPattern ClassOfClassOfInformationRepresentation**	**hasClassOfRepresented Class**
ClassOfClassOfRelationshipWithSignatureTemplate	**hasClassOfEnd1 RoleAndDomain**	**hasClassOfEnd2 RoleAndDomain**
ClassOfClassOfRepresentationTemplate	**hasClassOfPattern ClassOfClassOfInformationRepresentation**	**hasClassOfRepresented Class**
ClassOfClassOfRepresentationTranslationTemplate	**hasClassOfFirst ClassOfClassOfInformationRepresentation**	**hasClassOfSecond ClassOfClassOfInformationRepresentation**

Continued on next page

Table D.1 – continued from previous page

Proto-template	First role	Second role
ClassOfClassOfResponsibilityForRepresentationTemplate	**hasClassOfClassOfControlled** **ClassOfClassOfRepresentation**	**hasController** **PossibleIndividual**
ClassOfClassOfUsageOfRepresentationTemplate	**hasClassOfClassOfUsed** **ClassOfClassOfRepresentation**	**hasUser** **PossibleIndividual**
ClassOfClassificationTemplate	**hasClassOfClassified** **Class**	**hasClassOfClassifier** **ClassOfClass**
ClassOfCompositionOfIndividualTemplate	**hasClassOfPart** **ClassOfIndividual**	**hasClassOfWhole** **ClassOfIndividual**
ClassOfConnectionOfIndividualTemplate	**hasClassOfSide1** **ClassOfIndividual**	**hasClassOfSide2** **ClassOfIndividual**
ClassOfContainmentOfIndividualTemplate	**hasClassOfLocated** **ClassOfIndividual**	**hasClassOfLocator** **ClassOfIndividual**
ClassOfDefinitionTemplate	**hasPattern** **ClassOfInformationRepresentation**	**hasRepresented** **Class**
ClassOfDescriptionTemplate	**hasPattern** **ClassOfInformationRepresentation**	**hasRepresented** **Thing**
ClassOfDimensionForShapeTemplate	**hasClassOfDimension** **ClassOfShapeDimension**	**hasClassOfShape** **ClassOfShape**
ClassOfDirectConnectionTemplate	**hasClassOfSide1** **ClassOfIndividual**	**hasClassOfSide2** **ClassOfIndividual**
ClassOfFeatureWholePartTemplate	**hasClassOfPart** **ClassOfIndividual**	**hasClassOfWhole** **ClassOfArrangedIndividual**
ClassOfFunctionalMappingTemplate	**hasCodomain** **Class**	**hasDomain** **Class**
ClassOfIdentificationTemplate	**hasPattern** **ClassOfInformationRepresentation**	**hasRepresented** **Thing**
ClassOfIndirectConnectionTemplate	**hasClassOfSide1** **ClassOfIndividual**	**hasClassOfSide2** **ClassOfIndividual**
ClassOfIndirectPropertyTemplate	**hasClassOfPossessor** **ClassOfIndividual**	**hasPropertySpace** **PropertySpace**
ClassOfIndividualUsedInConnectionTemplate	**hasClassOfConnection** **ClassOfConnectionOfIndividual**	**hasClassOfUsage** **ClassOfIndividual**
ClassOfIntendedRoleAndDomainTemplate	**hasClassOfPlayer** **ClassOfIndividual**	**hasPlayed** **RoleAndDomain**
ClassOfInvolvementByReferenceTemplate	**hasClassOfInvolved** **RoleAndDomain**	**hasClassOfInvolver** **ClassOfActivity**
ClassOfIsomorphicFunctionalMappingTemplate	**hasCodomain**	**hasDomain**

Continued on next page

Table D.1 – continued from previous page

Proto-template	First role	Second role
	Class	**Class**
ClassOfLeftNamespaceTemplate	**hasClassOfClassOfWhole** **ClassOfClassOfInformationRepresentation**	**hasClassOfPart** **ClassOfInformationRepresentation**
ClassOfNamespaceTemplate	**hasClassOfClassOfWhole** **ClassOfClassOfInformationRepresentation**	**hasClassOfPart** **ClassOfInformationRepresentation**
ClassOfParticipationTemplate	**hasClassOfPart** **ParticipatingRoleAndDomain**	**hasClassOfWhole** **ClassOfActivity**
ClassOfPossibleRoleAndDomainTemplate	**hasClassOfPlayer** **ClassOfIndividual**	**hasPlayed** **RoleAndDomain**
ClassOfRecognitionTemplate	**hasClassOfRecognized** **Class**	**hasClassOfRecognizing** **ClassOfActivity**
ClassOfRelationshipWithSignatureTemplate	**hasClassOfEnd1** **RoleAndDomain**	**hasClassOfEnd2** **RoleAndDomain**
ClassOfRelativeLocationTemplate	**hasClassOfLocated** **ClassOfIndividual**	**hasClassOfLocator** **ClassOfIndividual**
ClassOfRepresentationOfThingTemplate	**hasPattern** **ClassOfInformationRepresentation**	**hasRepresented** **Thing**
ClassOfRepresentationTranslationTemplate	**hasClassOfFirst** **ClassOfInformationRepresentation**	**hasClassOfSecond** **ClassOfInformationRepresentation**
ClassOfResponsibilityForRepresentationTemplate	**hasClassOfControlled** **ClassOfRepresentationOfThing**	**hasController** **PossibleIndividual**
ClassOfRightNamespaceTemplate	**hasClassOfClassOfWhole** **ClassOfClassOfInformationRepresentation**	**hasClassOfPart** **ClassOfInformationRepresentation**
ClassOfScaleConversionTemplate	**hasCodomain** **Scale**	**hasDomain** **Scale**
ClassOfSpecializationTemplate	**hasClassOfSubclass** **ClassOfClass**	**hasClassOfSuperclass** **ClassOfClass**
ClassOfTemporalSequenceTemplate	**hasClassOfPredecessor** **ClassOfIndividual**	**hasClassOfSuccessor** **ClassOfIndividual**
ClassOfTemporalWholePartTemplate	**hasClassOfPart** **ClassOfIndividual**	**hasClassOfWhole** **ClassOfIndividual**
ClassOfUsageOfRepresentationTemplate	**hasClassOfUsed** **ClassOfRepresentationOfThing**	**hasUser** **PossibleIndividual**
ClassificationTemplate	**hasClassified** **Thing**	**hasClassifier** **Class**
ComparisonOfPropertyTemplate	**hasGreaterElement** **Property**	**hasLesserElement** **Property**
CompositionOfIndividualTemplate	**hasPart**	**hasWhole**

Continued on next page

Table D.1 – continued from previous page

Proto-template	First role	Second role
	PossibleIndividual	**PossibleIndividual**
ConnectionOfIndividualTemplate	**hasSide1** **PossibleIndividual**	**hasSide2** **PossibleIndividual**
ContainmentOfIndividualTemplate	**hasLocated** **PossibleIndividual**	**hasLocator** **PossibleIndividual**
CoordinateSystemTemplate	**hasCodomain** **NumberSpace**	**hasDomain** **PropertySpace**
DefinitionTemplate	**hasRepresented** **Class**	**hasSign** **PossibleIndividual**
DescriptionTemplate	**hasRepresented** **Thing**	**hasSign** **PossibleIndividual**
DifferenceOfSetOfClassTemplate	**hasInput** **EnumeratedSetOfClass**	**hasResult** **Class**
DimensionOfIndividualTemplate	**hasIndividual** **PossibleIndividual**	**hasIndividualDimension** **IndividualDimension**
DimensionOfShapeTemplate	**hasDimension** **ShapeDimension**	**hasShape** **Shape**
DirectConnectionTemplate	**hasSide1** **PossibleIndividual**	**hasSide2** **PossibleIndividual**
EndingTemplate	**hasPart** **Event**	**hasWhole** **PossibleIndividual**
FeatureWholePartTemplate	**hasPart** **PossibleIndividual**	**hasWhole** **ArrangedIndividual**
FunctionalMappingTemplate	**hasInput** **Thing**	**hasResult** **Thing**
IdentificationTemplate	**hasRepresented** **Thing**	**hasSign** **PossibleIndividual**
IndirectConnectionTemplate	**hasSide1** **PossibleIndividual**	**hasSide2** **PossibleIndividual**
IndirectPropertyTemplate	**hasPossessor** **PossibleIndividual**	**hasProperty** **Property**
IndividualUsedInConnectionTemplate	**hasConnection** **ConnectionOfIndividual**	**hasUsage** **PossibleIndividual**
IntendedRoleAndDomainTemplate	**hasPlayed** **RoleAndDomain**	**hasPlayer** **PossibleIndividual**
IntersectionOfSetOfClassTemplate	**hasInput** **EnumeratedSetOfClass**	**hasResult** **Class**
InvolvementByReferenceTemplate	**hasInvolved**	**hasInvolver**

Continued on next page

Table D.1 – continued from previous page

Proto-template	First role	Second role
	Thing	**Activity**
LeftNamespaceTemplate	**hasClassOfPart** **ClassOfInformationRepresentation**	**hasClassOfWhole** **ClassOfIndividual**
LifecycleStageTemplate	**hasInterest** **PossibleIndividual**	**hasInterested** **PossibleIndividual**
LowerBoundOfNumberRangeTemplate	**hasClassified** **ArithmeticNumber**	**hasClassifier** **NumberRange**
LowerBoundOfPropertyRangeTemplate	**hasClassified** **Property**	**hasClassifier** **PropertyRange**
MultidimensionalScaleTemplate	**hasCodomain** **NumberSpace**	**hasDomain** **PropertySpace**
NamespaceTemplate	**hasClassOfPart** **ClassOfInformationRepresentation**	**hasClassOfWhole** **ClassOfArrangedIndividual**
OtherRelationshipTemplate	**hasEnd1** **Thing**	**hasEnd2** **Thing**
ParticipationTemplate	**hasPart** **PossibleIndividual**	**hasWhole** **Activity**
PossibleRoleAndDomainTemplate	**hasPlayed** **RoleAndDomain**	**hasPlayer** **PossibleIndividual**
PropertyForShapeDimensionTemplate	**hasProperty** **Property**	**hasShapeDimension** **ShapeDimension**
PropertyQuantificationTemplate	**hasInput** **Property**	**hasResult** **ArithmeticNumber**
PropertySpaceForClassOfShapeDimensionTemplate	**hasClassOfShapeDimension** **ClassOfShapeDimension**	**hasPropertySpace** **PropertySpace**
RecognitionTemplate	**hasRecognized** **Thing**	**hasRecognizing** **Activity**
RelativeLocationTemplate	**hasLocated** **PossibleIndividual**	**hasLocator** **PossibleIndividual**
RepresentationOfThingTemplate	**hasRepresented** **Thing**	**hasSign** **PossibleIndividual**
ResponsibilityForRepresentationTemplate	**hasControlled** **RepresentationOfThing**	**hasController** **PossibleIndividual**
RightNamespaceTemplate	**hasClassOfPart** **ClassOfInformationRepresentation**	**hasClassOfWhole** **ClassOfIndividual**
ScaleTemplate	**hasCodomain** **NumberSpace**	**hasDomain** **PropertySpace**
SpecializationTemplate	**hasSubclass**	**hasSuperclass**

Continued on next page

Table D.1 – continued from previous page

Proto-template	First role	Second role
	Class	**Class**
SpecializationByDomainTemplate	**hasSubclass**	**hasSuperclass**
	RoleAndDomain	**Class**
SpecializationByRoleTemplate	**hasSubclass**	**hasSuperclass**
	RoleAndDomain	**Role**
SpecializationOfIndividualDimensionFromPropertyTemplate	**hasSubclass**	**hasSuperclass**
	IndividualDimension	**Property**
TemporalBoundingTemplate	**hasPart**	**hasWhole**
	Event	**PossibleIndividual**
TemporalSequenceTemplate	**hasPredecessor**	**hasSuccessor**
	PossibleIndividual	**PossibleIndividual**
TemporalWholePartTemplate	**hasPart**	**hasWhole**
	PossibleIndividual	**PossibleIndividual**
UnionOfSetOfClassTemplate	**hasInput**	**hasResult**
	EnumeratedSetOfClass	**Class**
UpperBoundOfNumberRangeTemplate	**hasClassified**	**hasClassifier**
	ArithmeticNumber	**NumberRange**
UpperBoundOfPropertyRangeTemplate	**hasClassified**	**hasClassifier**
	Property	**PropertyRange**
UsageOfRepresentationTemplate	**hasUsed**	**hasUser**
	RepresentationOfThing	**PossibleIndividual**

Table D.1: Compact listing of Proto-templates

Annex E
(informative)
Recursive vs. non-recursive template expansion

E.1 Nomenclature

In computer science, the words *macros*, *templates*, *inlining* have been used to denote various partly overlapping concepts.

macro:

— In LISP, a macro is a function that is executed during compile time and that returns a piece of program that is spliced in in place of the macro call. Since the function can produce arbitrary code, this is a very general mechanism.

— In the C preprocessor (cpp), a macro call is replaced by the expansion test, after which it is again scanned for occurrences of macro calls. Recursion is classically not prevented, but leads to an infinite loop of the preprocessor. The GNU[3)] cpp detects and blocks recursion.

— In TEX, macros are expanded similarly to cpp, also allowing for recursion. But there are also conditional data structures which allow terminating a recursion

templates:

— In C++, templates, originally meant as a means of describing parametric polymorphism of data types, may also be recursive. Several expansions for the same template with different arguments. The compiler prefers the most specific definition that matches a particular instance, which makes it possible to define a terminating "base case". This is heavily exploited in so-called "template metaprogramming".

— Code templates in IDEs[4)].

inlining: is a technique where function/procedure calls are not complied into code that uses the stack to remember its previous state, and then jumps into the code of the function; instead, the body of the function is expanded in the place of the call. This does not work for recursive calls, or at least one has to limit the depth of calls that get inlined. Some compilers do inlining under-the-hood. In GNU C, a function can be declared as inline, which prompts the compiler to used it in this way. Apart from the restrictions concerning recursion, the semantics of an inlined function call are like those of a normal call, but the code runs faster.

Since the standard is about templates, we will talk about *template mechanisms* when discussing different possibilities of defining such a thing.

E.2 Recursion or not?

There is an essential difference between a template or macro mechanism that allows for cyclic or recursive definitions, and one that does not.

[3)] GNU is a recursive acronym for "GNU's Not Unix".

[4)] Integrated development environment.

For some discussion of this topic, see Sects. 1.3.1 and 2.2.2 of the Description Logic Handbook [11]. The distinction between recursive and non-recursive macros is the same as that between cyclic and acyclic TBoxes.

An example of a thorough definition of a particular non-recursive macro expansion mechanism is given in [16].

Taking an abstract look at a set of template definitions, let N be a set of names available for templates, and maybe also for other parts of the language. Now a template definition gives a definition for some name $n \in N$, referring to (or using) the names of various other templates or other elements with names N, say $n_1, n_2, \ldots, n_k \ldots$. Write $n \succ n_i$ if n_i is referred to in the definition of n. For instance, with a template definition

$$\mathsf{SpecOrEqual(a, b)} := \mathsf{Spec(a, b)} \vee \mathsf{a = b} \tag{1}$$

one has $\mathsf{SpecOrEqual} \succ \mathsf{Spec}$, whereas with a definition

$$\mathsf{SpecStar(a, b)} := (\exists \mathsf{x.Spec(a, x)} \wedge \mathsf{SpecStar(x, b)}) \vee \mathsf{a = b} \tag{2}$$

one gets $\mathsf{SpecStar} \succ \mathsf{Spec}$ and $\mathsf{SpecStar} \succ \mathsf{SpecStar}$. Given several such definitions, let $\succ$ be the union of all the relations given by the individual definitions.

In a *non-recursive template mechanism*, the relation $\succ$ from all definitions is required to be acyclic. I.e. there must not be a sequence $n_0 \succ n_1 \succ \cdots \succ n_k$ with $n_0 = n_k$. In particular, there cannot be an n_0 with $n_0 \succ n_0$.

Equivalently, the transitive closure $\succ^+$ is required to be irreflexive, but that is just a pretty way of saying the same thing.

For instance, example (1) is permissible in a non-recursive mechanism but not (2).

One usually partitions N into a set P of primitive names and a set D of defined names, $N = P \cup D$, $P \cap D = \emptyset$, such that there must not be template definitions for names in P, while any name in D has to have a definition.

If the template mechanism works by repeatedly replacing references to defined names $n \in D$ by the body of the definition of n, this means that any input ultimately (after finitely many steps) gets reduced to something that contains only names in P, i.e. primitive names. Moreover, the result is easily seen to be independent of the order in which occurrences of defined names are expanded.

One can also see that the evaluation of templates through expansion is consistent with an axiomatic view as follows: For every template definition

$$N(x, y, \ldots) \leftrightarrow \phi$$

where the body b is a complex term possibly containing $x, y, \ldots$, define an axiom[5)]

$$\forall x, y, \ldots . N(x, y, \ldots) \leftrightarrow \phi$$

and collect all these axioms into a set Ax. Now if s is the result of repeatedly expanding all defined names in t, it holds that

$$Ax \models s = t$$

[5)] If these are formula templates, one can take $\leftrightarrow$ instead of =.

i.e. it follows from the axiomatic reading of the template definitions that the expansion s is semantically equivalent to the original term t.[6)] This can be shown by induction over the expansion steps and repeated applications of the substitution lemma.

Great care needs to be taken with template arguments: if "higher-order" arguments are allowed, i.e. arguments that are used as functions in the expansion, cycles can be introduced in unexpected ways. For instance, after defining

$$\mathsf{SelfApp(x)} := \mathsf{x(x)} \quad ,$$

the expansion of SelfApp(SelfApp) leads to an infinite cycle, destroying the nice properties of the mechanism. If one does not want to exclude higher-order arguments, one needs some kind of type system for the arguments to exclude this kind of problem.

In a *recursive template mechanism* on the other hand, there are no restrictions on $\succ$. In particular, one can use recursive definitions such as (2) above. The obvious advantage is that one can express complex properties such as transitive closure. Disadvantages include:

— no more guarantee that the outcome, in particular termination, is independent of the order of expansion;

— one gets a Turing-complete mechanism, basically the same as the untyped lambda calculus. I.e. one can program with templates! Programmers being programmers, this will be used. But a template mechanism is a very bad programming language in that it is error prone and hard to understand. C++ template meta-programming is the prime example of this. If one wants a programming language, one should design one, instead of using something that was meant as an abbreviation facility.

E.3 Further technical issues

Even for a non-recursive template mechanism, some precautions must be taken:

— it can be tempting to allow template bodies that are not valid terms. For instance, one could define

$$\& := \wedge$$

to use ones own favorite symbol for conjunction. If expansion is string-based, this would actually work. But it makes the analysis of a system of template definitions very difficult;

— in a first order context, template arguments should not be quantified in the body of the definition:

$$t(x) := \forall x.p(x)$$

should not be allowed, and doesn't make much sense either;

— the instantiation must take care to rename bound variables as needed. With a template

$$t(x) := \forall y.p(x, y) \quad ,$$

the instance $t(y)$ should not be instantiated to

$$\forall y.p(y, y) \quad .$$

This is known as *variable capture*. It can be avoided by first renaming the bound variable, e.g.

$$\forall y'.p(y, y') \quad .$$

[6)] The converse is not necessarily true, since lots of equations can follow from Ax without being provable by template expansion alone.

Annex F
(informative)
Template expansion: example

This section describes an example to illustrate how expressions using templates expand to, and correspond to, expressions in the ISO 15926-2 language. Three simple template statements are expanded into a first-order formula, using only ISO 15926-2 language predicates. The quantifiers of that formula are then replaced by constants, corresponding to identifiers of atomic data in the sense of ISO 15926-2. The results are checked for logical consistency.

F.1 Example of template expansion

The following statement is given in the template language, as defined in previous clauses. The following sections show the result of expanding this statement, eliminating templates in favour of statements in the ISO 15926-2 language.

LowerUpperOfNumberRange(**[-273.1 to Infinity]**, **-273.1**, **Infinity**) ∧
DimensionUnitNumberRangeOfScale(**Celsius**, **DegrC**,
  Temperature, **[-273.1 to Infinity]**) ∧
PropertyRangeMagnitudeRestrictionOfClass(**PressureTransmitter**,
  AmbientTemperature, **Celsius**, **-40**, **+40**)

F.2 Result of expansion according to template axioms

The templates are expanded according to their axiomatic definition. This produces, in general, an existentially quantified first-order formula, in which all template predicates are eliminated in favour of ISO 15926-2 entity types and attributes.

(**NumberRange**(**[-273.1 to Infinity]**)
  ∧ **ArithmeticNumber**(**-273.1**)
  ∧ **ArithmeticNumber**(**Infinity**)
  ∧ ∃*z*
    (**LowerBoundOfNumberRange**(*z*)
      ∧ **hasClassified**(*z*, **-273.1**)
      ∧ **hasClassifier**(*z*, **[-273.1 to Infinity]**))
  ∧ ∃*z*
    (**UpperBoundOfNumberRange**(*z*)
      ∧ **hasClassified**(*z*, **Infinity**)
      ∧ **hasClassifier**(*z*, **[-273.1 to Infinity]**)))
∧ (**Scale**(**Celsius**)
  ∧ **ExpressString**(**DegrC**)
  ∧ **SinglePropertyDimension**(**Temperature**)
  ∧ **NumberRange**(**[-273.1 to Infinity]**)
  ∧ (**Scale**(**Celsius**)
    ∧ **ExpressString**(**DegrC**)
    ∧ **Thing**(**Celsius**)
    ∧ **ExpressString**(**DegrC**)
    ∧ **ClassOfClassOfIdentification**(**UomSymbolAssignment**)
    ∧ ∃*u*

((**ClassOfIdentification**(u)
∧ **hasPattern**(u, **DegrC**)
∧ **hasRepresented**(u, **Celsius**))
∧ ∃z
(**Classification**(z)
∧ **hasClassified**(z, u)
∧ **hasClassifier**(z, **UomSymbolAssignment**))))
∧ **Scale**(**Celsius**)
∧ **hasCodomain**(**Celsius**, **[-273.1 to Infinity]**)
∧ **hasDomain**(**Celsius**, **Temperature**))
∧ **ClassOfIndividual**(**PressureTransmitter**)
∧ **ClassOfIndirectProperty**(**AmbientTemperature**)
∧ **Scale**(**Celsius**)
∧ **ExpressReal**(**-40**)
∧ **ExpressReal**(**+40**)
∧ ∃u
((**ClassOfIndividual**(**PressureTransmitter**)
∧ **ClassOfIndirectProperty**(**AmbientTemperature**)
∧ **PropertyRange**(u)
∧ ∃u_0
((**ClassOfIndirectProperty**(u_0)
∧ **hasClassOfPossessor**(u_0, **PressureTransmitter**)
∧ **hasPropertySpace**(u_0, u))
∧ **ClassOfRelationship**(u_0)
∧ **ClassOfRelationship**(**AmbientTemperature**)
∧ ∃y
((**Specialization**(y)
∧ **hasSubclass**(y, u_0)
∧ **hasSuperclass**(y, **AmbientTemperature**))
∧ ∃z
(**Classification**(z)
∧ **hasClassified**(z, y)
∧ **hasClassifier**(z, **End2UniversalRestriction**)))))
∧ ∃y_1
∃y_2
((**ExpressReal**(**-40**)
∧ **Thing**(y_1)
∧ ∃z
(**ClassOfIdentification**(z)
∧ **hasPattern**(z, **-40**
∧ **hasRepresented**(z, y_1)))
∧ (**ExpressReal**(**+40**
∧ **Thing**(y_2)
∧ ∃z
(**ClassOfIdentification**(z)
∧ **hasPattern**(z, **+40**
∧ **hasRepresented**(z, y_2)))
∧ **PropertyRange**(u)
∧ **Scale**(**Celsius**)
∧ **ArithmeticNumber**(y_1)
∧ **ArithmeticNumber**(y_2)

$$
\begin{aligned}
&\wedge\ \exists y_{10}\\
&\quad \exists y_{20}\\
&\qquad ((\mathbf{PropertyRange}(u)\\
&\qquad\quad \wedge\ \mathbf{Property}(y_{10})\\
&\qquad\quad \wedge\ \mathbf{Property}(y_{20})\\
&\qquad\quad \wedge\ \exists z\\
&\qquad\qquad (\mathbf{LowerBoundOfPropertyRange}(z)\\
&\qquad\qquad\quad \wedge\ \mathbf{hasClassified}(z, y_{10})\\
&\qquad\qquad\quad \wedge\ \mathbf{hasClassifier}(z, u))\\
&\qquad\quad \wedge\ \exists z\\
&\qquad\qquad (\mathbf{UpperBoundOfPropertyRange}(z)\\
&\qquad\qquad\quad \wedge\ \mathbf{hasClassified}(z, y_{20})\\
&\qquad\qquad\quad \wedge\ \mathbf{hasClassifier}(z, u)))\\
&\qquad \wedge\ (\mathbf{Property}(y_{10})\\
&\qquad\quad \wedge\ \mathbf{ArithmeticNumber}(y_1)\\
&\qquad\quad \wedge\ \mathbf{Scale}(\mathbf{Celsius})\\
&\qquad\quad \wedge\ \exists u\\
&\qquad\qquad ((\mathbf{PropertyQuantification}(u)\\
&\qquad\qquad\quad \wedge\ \mathbf{hasInput}(u, y_{10})\\
&\qquad\qquad\quad \wedge\ \mathbf{hasResult}(u, y_1))\\
&\qquad\qquad\quad \wedge\ \exists z\\
&\qquad\qquad\quad (\mathbf{Classification}(z)\\
&\qquad\qquad\qquad \wedge\ \mathbf{hasClassified}(z, u)\\
&\qquad\qquad\qquad \wedge\ \mathbf{hasClassifier}(z, \mathbf{Celsius}))))\\
&\qquad \wedge\ \mathbf{Property}(y_{20})\\
&\qquad \wedge\ \mathbf{ArithmeticNumber}(y_2)\\
&\qquad \wedge\ \mathbf{Scale}(\mathbf{Celsius})\\
&\qquad \wedge\ \exists u\\
&\qquad\quad ((\mathbf{PropertyQuantification}(u)\\
&\qquad\qquad\quad \wedge\ \mathbf{hasInput}(u, y_{20})\\
&\qquad\qquad\quad \wedge\ \mathbf{hasResult}(u, y_2))\\
&\qquad\qquad\quad \wedge\ \exists z\\
&\qquad\qquad\quad (\mathbf{Classification}(z)\\
&\qquad\qquad\qquad \wedge\ \mathbf{hasClassified}(z, u)\\
&\qquad\qquad\qquad \wedge\ \mathbf{hasClassifier}(z, \mathbf{Celsius})))))
\end{aligned}
$$

F.3 Instantiating existential quantifiers

The existential quantifiers are eliminated by instantiating them with constant values. This is analogous to assigning identifiers to the data elements, in a ISO15926 data store. The result is a set of atomic statements in the ISO 15926-2 language. The constants used correspond to data that instantiate ISO 15926-2 types, analogous to data in reference data libraries.

$$
\begin{aligned}
&\mathbf{NumberRange}(\mathbf{[-273.1\ to\ Infinity]})\\
&\mathbf{ArithmeticNumber}(\mathbf{-273.1})\\
&\mathbf{ArithmeticNumber}(\mathbf{Infinity})\\
&\mathbf{LowerBoundOfNumberRange}(\mathbf{X})\\
&\mathbf{hasClassified}(\mathbf{X}, \mathbf{-273.1})
\end{aligned}
$$

hasClassifier(X, [-273.1 to Infinity])
UpperBoundOfNumberRange(Y)
hasClassified(Y, Infinity)
hasClassifier(Y, [-273.1 to Infinity])
Scale(Celsius)
ExpressString(DegrC)
SinglePropertyDimension(Temperature)
Thing(Celsius)
ClassOfClassOfIdentification(UomSymbolAssignment)
ClassOfIdentification(Z)
hasPattern(Z, DegrC)
hasRepresented(Z, Celsius)
Classification(U)
hasClassified(U, Z)
hasClassifier(U, UomSymbolAssignment)
hasCodomain(Celsius, [-273.1 to Infinity])
hasDomain(Celsius, Temperature)
ClassOfIndividual(PressureTransmitter)
ClassOfIndirectProperty(AmbientTemperature)
ExpressReal(-40)
ExpressReal(+40)
PropertyRange(W)
ClassOfIndirectProperty(V5)
hasClassOfPossessor(V5, PressureTransmitter)
hasPropertySpace(V5, W)
ClassOfRelationship(V5)
ClassOfRelationship(AmbientTemperature)
Specialization(V6)
hasSubclass(V6, V5)
hasSuperclass(V6, AmbientTemperature)
Classification(V7)
hasClassified(V7, V6)
hasClassifier(V7, End2UniversalRestriction)
Thing(V8)
ClassOfIdentification(V9)
hasPattern(V9, -40)
hasRepresented(V9, V8)
Thing(V10)
ClassOfIdentification(V11)
hasPattern(V11, +40)
hasRepresented(V11, V10)
ArithmeticNumber(V8)
ArithmeticNumber(V10)
Property(V12)
Property(V13)
LowerBoundOfPropertyRange(V14)
hasClassified(V14, V12)
hasClassifier(V14, W)
UpperBoundOfPropertyRange(V15)
hasClassified(V15, V13)
hasClassifier(V15, W)

PropertyQuantification(V16)
hasInput(V16, V12)
hasResult(V16, V8)
Classification(V17)
hasClassified(V17, V16)
hasClassifier(V17, Celsius)
PropertyQuantification(V18)
hasInput(V18, V13)
hasResult(V18, V10)
Classification(V19)
hasClassified(V19, V18)
hasClassifier(V19, Celsius)

F.4 Verification of consistency

The data set resulting from expansion and instantiation (F.3) may be checked for consistency with ISO 15926-2. This is achieved by interpreting the data set as atomic statements extending the axiomatization of ISO 15926-2 (annex B). The resulting set of axioms can be checked using a theorem prover for generic first-order logic, or by more specialized deduction software. Examples of the latter include description logic provers.

Annex G
(informative)
Consistency checking via coherent logic

An inspection shows that the formal language of the formalization of ISO 15926-2 and ISO 15926-7 falls within the fragment of FOL known as Coherent Logic (CL), consisting of formulas of the form

$$\forall \vec{x}\,(A_1 \wedge \cdots \wedge A_n \rightarrow \exists \vec{y}_1.C_1 \vee \cdots \vee \exists \vec{y}_m.C_m)$$

where the A_i are atoms and the C_j are conjunctions of atoms. When writing CL formulae, the unversal quantifiers are usually omitted, so that free variables are implicitly universally quantified. Example from annex B:

— The universal axiom $\forall x(\mathbf{Thing}(x))$ is the CL formula

$$\rightarrow \mathbf{Thing}(x)$$

— The disjoint axioms can be encoded using $\perp$, e.g.,

$$\neg(\mathbf{IntegerNumber}(x) \wedge (\mathbf{MultidimensionalNumber}(x)))$$

goes into

$$\mathbf{IntegerNumber}(x) \wedge \mathbf{MultidimensionalNumber}(x) \rightarrow \perp$$

— All other axioms have already the CL form

Note that CL has both quantifiers $\forall$ and $\exists$, but their possible alternations are restricted: the only permitted quantifier shift is due to the existence quantifier in the consequent, which can occur within the scope of the universal quantifiers that encompass the whole formula.

$$\mathbf{hasClassOfClassOfWhole}(x, y) \rightarrow (\mathbf{ClassOfClassOfComposition}(x) \vee \mathbf{ClassOfNamespace}(x))$$

The problem we address here is the $\mathcal{T}, \Sigma$ consistency problem: *Given a set of CL formulae $\mathcal{T}$ (i.e. a coherent theory) and set of $\wedge$-$\vee$-$\exists$-formulae Σ, identify conditions on $\mathcal{T}$ and Σ such that the test for FOL-consistency of $\mathcal{T} \cup \Sigma$ is decidable.*[7)] The key observation is that it is sufficient to check $\mathcal{T} \cup \Sigma$ for CL-consistency.

An important point of CL is that coherent formulas can be used as generating rules, which forms a complete proof procedure for CL. In this note we use the same idea to give a solution to the $\mathcal{T}, \Sigma$ consistency problem. The procedure is driven by known constants and the known ground atomic formulae (both of which there are always finitely many). We assume initially that Σ contains at least one constant symbol. If there is an individual of which we know nothing, the representation Thing(a) is used.

Technically we introduce the notion of an application sequence, which is a string of rule applications. For each application sequence s, fml(s) is a set of formulae and and dom(s) is a set of constants. The set of *application sequences from* Σ is inductively defined as follows.

— The empty string ϵ is an application sequence; fml(ϵ) = Σ and dom(ϵ) is the set of constant symbols in fml(ϵ).

[7)] Note that this covers the consistency checking problem from Section K.2: Check whether $\Psi \wedge \bar{\phi}$ is consistent, where Ψ is a formula representing the axiomatization of ISO 15926-2, S is a template set, and $\bar{\phi}$ is the $R(S)$-normal form of ϕ, the closed $\wedge$-$\vee$-$\exists$-formula that we want to test for consistency.

— If s is an application sequence, $s.r$ is also provided r is either of the following:

  — a closed instance $A_1 \wedge \cdots \wedge A_n \rightarrow C$ of a formula in $\mathcal{T}$ such that each $A_i \in \text{fml}(s)$. Then $\text{fml}(s.r) = \text{fml}(s) \cup \{C\}$ and $\text{dom}(s.r) = \text{dom}(s)$

  — $A_1 \vee A_2 \rightarrow A_i$ where $A_1 \vee A_2 \in \text{fml}(s)$ and no member of s is $A_1 \vee A_2 \rightarrow A_{3-i}$. Then $\text{fml}(s.r) = \text{fml}(s) \cup \{A_i\}$ and $\text{dom}(s.r) = \text{dom}(s)$

  — $A_1 \wedge A_2 \rightarrow A_i$ where each $A_i \in \text{fml}(s)$. Then $\text{fml}(s.r) = \text{fml}(s) \cup \{A_i\}$ and $\text{dom}(s.r) = \text{dom}(s)$

  — $\exists x A(x) \rightarrow A(t)$ for $\exists x A(x) \in \text{fml}(s)$ such that no formula of the form $A(t')$ is in $\text{fml}(s)$. In this case t is a new constant symbol, $\text{fml}(s.r) = \text{fml}(s) \cup \{A(t)\}$ and $\text{dom}(s.r) = \text{dom}(s) \cup \{t\}$

  — $A(a) \rightarrow A(b)$ for $A(a)$ and either $a = b$ or $b = a$ in $\text{fml}(s)$. Then $\text{fml}(s.r) = \text{fml}(s) \cup \{A(b)\}$ and $\text{dom}(s.r) = \text{dom}(s)$

The proviso in or-elimination is necessary for consisteny. An application sequence for $\{\textbf{PossibleIndividual}(a) \vee \textbf{AbstractObject}(a)\}$ can derive either that a is a possible individual or that a is an abstract object, but not both.

The proviso in $\exists$-elimination is possible since in CL one can avoid skolemization. If, for instance, we have derived $R(a, b)$, $\exists x R(a, x)$ will not be expanded further since we already have a witness. This is a major source of improvement over full FOL (which is also used in DL reasoners).

It is now easy to see that the consistency of Σ with respect to $\mathcal{T}$ is decidable if, for any application sequence s from Σ,

$$\bigcup_{t \leq s} \text{dom}(t)$$

is finite, where $t \preceq s$ means that t is an initial sequence of s.

This is clearly the case if we can limit the number of new terms coming from $\exists$-elimination. The only axioms in the formalization of ISO 15926-2 that introduce formulae with existence quantifiers are the role axioms, e.g.,

$$\textbf{ClassOfClassOfComposition}(x) \wedge \textbf{hasClassOfClassOfWhole}(x, y) \rightarrow \textbf{ClassOfClassOfIndividual}(y)$$
$$\textbf{ClassOfClassOfComposition}(x) \rightarrow \exists y(\textbf{hasClassOfClassOfWhole}(x, y))$$

The latter axiom is potentially harmful. If a is a **ClassOfClassOfComposition**, it can generate a new term b as filler in **hasClassOfClassOfWhole**, from which it follows that **ClassOfClassOfIndividual**(b). But as long as this does entail that **ClassOfClassOfComposition**(b), there can be no unbounded generation of new terms stemming from these axioms.

The connection to CL is relevant for ISO 15926-2 / ISO 15926-7, and possibly worth exploring further, for several reasons.

— The syntactical form of CL formulas allows a very simple proof procedure known as *forward ground reasoning*. The procedure is effective for CL formulas and easy to reason about. It is likely that CL-based algorithms for consistency checking can be implemented for ISO 15926-2 / ISO 15926-7 that are more efficient than algorithms based on translation into DL since knowledge about the structure of ISO 15926-2 can be exploited in the search procedures.

— CL is more expressive than DLs. It is possible that one can identify decidable fragments of CL that are better suited than DLs for applications of ISO 15926 like, e.g., expressing constraints on data stores.

— CL is constructive, which means that it has more structure than FOL. In many situations this structure can facilitate the study of meta-properties.

Since coherent logic is not a very well known fragment of FOL we include some keywords about it:

— CL goes back to the Norwegian Thoralf Skolem, who developed it in 1920 to obtain metamathematical results in lattice theory and projective geometry [24], [25]. It is also known as *Geometrical logic*.

— Recently CL has been rediscovered: see [15] for its relevance to computer science and [12] for recent progress in computer science research on CL.

— The bottom-up proof procedure used today in deductive databases and the system SATCHMO [21] (for logics weaker than CL), can already be found in [24].

— CL extends Horn clause logic [19] on which the programming language Prolog [20] is based in that we may have a whole disjunction of existentially quantified conjunctions of atoms in the conclusion.

— The undecidability of CL without function symbols has been proved in [13].

— CL is somewhat less expressive than full FOL. However, there exists a natural translation of FOL to CL [14] which preserves satisfiability, but at the cost of blowing up formula size considerably. Every first-order theory has a conservative (definitional) extension which is equivalent to a coherent theory.

Annex H
(informative)
Formal constraints beyond templates

The template language may be embedded in more expressive first-order languages. Using richer formalisms, a wide range of constraints of practical interest can be expressed.

EXAMPLE 1 Transitivity of **Specialization** may be expressed as follows.

$$\forall xyz(SpecializationTemplate(x, y) \wedge SpecializationTemplate(y, z) \rightarrow SpecializationTemplate(x, z))$$

EXAMPLE 2 If a is a member of b, then b may not be a member of a. This constraint can be captured as follows.

$$\forall xy(ClassificationTemplate(x, y) \rightarrow \neg ClassificationTemplate(y, x))$$

Annex J
(normative)
Semantics of templates

J.1 Rewrite rule

The following definition is taken verbatim from Defs. 3.1 and 3.3 of [22]. They are originally from [23], where a pattern rewriting system is called a higher-order rewriting system.

A λ-term t in β-normal form is called a (higher-order) pattern if every free occurrence of a variable F is in a subterm $F(\overline{u_n})$ of t, such that $\overline{u_n}$ is η-equivalent to a list of distinct bound variables.

A rewrite rule is a pair of λ-terms $l \Rightarrow r$ such that l is not a free variable, l and r are of the same base type, and $fv(l) \supseteq fv(r)$. A pattern rewrite rule is a rewrite rule where l is a pattern. A pattern rewrite system (PRS) is a set of pattern rewrite rules.

NOTE These definitions are very general, but they imply that

— the first order atoms $N(x, y, \ldots)$ that are the left sides of template definitions are in fact patterns (the argument lists $\overline{u_n}$ are empty in this case);

— if $N(x, y, \ldots) \leftrightarrow \phi$ is a template definition, then $N(x, y, z) \Rightarrow \phi$ is a pattern rewrite rule.

These definitions are used instead of standard first-order rewriting notions to ensure that quantifiers in the body of a template definition can do no harm. They also provide a very general framework in case the template mechanism should be extended at some future point.

J.2 Pattern rewriting system corresponding to a template set

The pattern rewriting system $R(S)$ corresponding to a template set S is the set of all rules $N(x, y, z) \Rightarrow \phi$ for all $N(x, y, z) \leftrightarrow \phi \in S$.

The following lemma guarantees that if one repeatedly expands occurrences of templates in any given input formula, then

a) this process eventually stops, i.e. there are no 'infinite loops'. One can 'exhaustively expand' templates until one eventually comes to a formula which no longer refers to templates. Additionally;

b) if the formula contains several references to templates, it ultimately does not matter in which order one expands them, because the result after exhaustive expansion is always the same.

Lemma 1 *$R(S)$ is terminating and confluent for any logical template set S.*

Proof: To prove termination, first an ordering on the set $\Sigma_0 \cup \text{names}(S)$ of all basic symbols and template names is defined, letting $N \succ_0 A$ iff there is a template definition $N(\cdots) \leftrightarrow \phi \in S$ such that A occurs in ϕ. $\succ$ to be the transitive closure of $\succ_0$ is defined. The construction of logical template sets guarantees that $\succ$ is in fact an order, since $\succ_0$ cannot have cycles. Also, due to the inductive definition of logical template sets, $\succ$ is well-founded. It is well-known that this implies that the multi-set extension $\succ\!\!\succ$ of $\succ$ is a well-founded order on multi-sets of symbols.

Now define $\mu(\phi)$ to be the multi-set of basic symbols and template names occurring in ϕ. In particular, if a symbol occurs several times in ϕ, it must occur the same number of times in $\mu(\phi)$. If one use of a template N in ϕ is expanded by its definition, obtaining ϕ', then ϕ' has one occurrence of N less than ϕ,

but additional occurrences of the symbols in the body of the template definition, which are smaller than N w.r.t. $\succ_0$. By the definition of the multi-set extension of an ordering, this means that $\mu(\phi) \succ \mu(\phi')$. Since $\succ$ is well-founded, it immediately follows that the template expansion terminates.

To show confluence, refer to the critical pair lemma for pattern rewrite systems, Theorem 4.7 in [22]. Showing that all critical pairs in $R(S)$ converge. This is trivially the case, since there are no critical pairs in $R(S)$: no symbol occurs in more than one left side of a rule in $R(S)$. *Q.E.D.*

This is be used to define the semantics of templates simply as the final result of exhaustive expansion. The lemma guarantees that this is a valid definition.

EXAMPLE Given the template set

$$\begin{array}{lllll} \{ & A(x) & \leftrightarrow & \exists y.(\, B(y) \wedge \mathbf{R}(x,y)\,), \\ & B(x) & \leftrightarrow & \mathbf{C}(x) \vee \mathbf{D}(x) & \}, \end{array}$$

expanded to the formula $A(a) \wedge B(b)$ as

$$\begin{array}{ll} & \underline{A(a)} \wedge B(b) \\ \rightsquigarrow & \underline{\exists y.(\, B(y)} \wedge \mathbf{R}(a,y)\,) \wedge B(b) \\ \rightsquigarrow & \exists y.(\, (\underline{\mathbf{C}(y)} \vee \mathbf{D}(y)) \wedge \mathbf{R}(a,y)\,) \wedge \underline{B(b)} \\ \rightsquigarrow & \exists y.(\, (\mathbf{C}(y) \vee \mathbf{D}(y)) \wedge \mathbf{R}(a,y)\,) \wedge (\underline{\mathbf{C}(b)} \vee \mathbf{D}(b)) \end{array}$$

where the sub-formula expanded next is underlined, or as

$$\begin{array}{ll} & \underline{A(a)} \wedge B(b) \\ \rightsquigarrow & \underline{\exists y.(\, B(y)} \wedge \mathbf{R}(a,y)\,) \wedge \underline{B(b)} \\ \rightsquigarrow & \exists y.(\, \underline{B(y)} \wedge \mathbf{R}(a,y)\,) \wedge (\underline{\mathbf{C}(b)} \vee \mathbf{D}(b) \\ \rightsquigarrow & \exists y.(\, (\underline{\mathbf{C}(y)} \vee \mathbf{D}(y)) \wedge \mathbf{R}(a,y)\,) \wedge (\mathbf{C}(b) \vee \mathbf{D}(b)) \end{array}$$

or finally as

$$\begin{array}{ll} & A(a) \wedge \underline{B(b)} \\ \rightsquigarrow & \underline{A(a)} \wedge (\underline{\mathbf{C}(b)} \vee \mathbf{D}(b)) \\ \rightsquigarrow & \underline{\exists y.(\, B(y)} \wedge \mathbf{R}(a,y)\,) \wedge (\mathbf{C}(b) \vee \mathbf{D}(b)) \\ \rightsquigarrow & \exists y.(\, (\underline{\mathbf{C}(y)} \vee \mathbf{D}(y)) \wedge \mathbf{R}(a,y)\,) \wedge (\mathbf{C}(b) \vee \mathbf{D}(b)) \end{array}$$

and the result is always the same.

J.3 Template expansion

Given a template set $S \in TS(\Sigma_0)$ and a $\wedge$-$\vee$-$\exists$-formula ϕ over symbols in $\Sigma_0 \cup \text{names}(S)$, the *expansion of ϕ w.r.t. S* as normal form of ϕ under $R(S)$ is defined, i.e. the uniquely determined result of exhaustively applying rewrite rules from $R(S)$ to ϕ.

Annex K
(normative)
Properties of template expansion

K.1 Logical readings of template definitions

If the template definitions in a set are taken as equivalence axioms, then a formula and its expansion are equivalent with respect to these axioms.

Lemma 2 *Let* $S \in TS(\Sigma_0)$ *be a template set and* ϕ *and* ϕ' $\wedge$-$\vee$-$\exists$*-formulae over symbols in* $\Sigma_0 \cup names(S)$*. If* ϕ' *is the expansion of* ϕ*, then* $S \models \phi \leftrightarrow \phi'$*.*

Proof of Lemma 2: Since every template expansion step replaces a sub-formula by one that is equivalent under the axioms, the final result is still equivalent to the original one. *Q.E.D.*

EXAMPLE The axioms corresponding to the template set above are

$$\begin{array}{lll} \forall x. & (\, A(x) & \leftrightarrow \quad \exists y.(\, B(y) \wedge \mathbf{R}(x,y)\,)\,) \\ \forall y. & (\, B(x) & \leftrightarrow \quad \mathbf{C}(x) \vee \mathbf{D}(x)\,) \end{array}$$

According to Lemma 2, these two together imply the equivalence

$$\begin{array}{ll} & A(a) \wedge \overline{B(b)} \\ \leftrightarrow & \exists y.(\,(\overline{\mathbf{C}(y)} \vee \mathbf{D}(y)) \wedge \mathbf{R}(a,y)\,) \wedge (\mathbf{C}(b) \vee \mathbf{D}(b)) \end{array}$$

of the original formula and its expansion in our example.

K.2 Decidability of consistency with ISO 15926-2

In practical applications of template definitions, the following type of problem will be of interest. Given

— the axiomatisation of ISO 15926-2 (given as a formula Ψ),

— a template set S, and

— a $\wedge$-$\vee$-$\exists$-formula ϕ without free variables,

check whether $\Psi \wedge \bigwedge_{\xi \in S} \xi \wedge \phi$ is consistent.

NOTE ϕ might be a large formula that refers to many defined templates. This is to be sure that ϕ, together with the template definitions, does not contradict the basic ISO 15926-2 axiomatisation.

EXAMPLE The ISO 15926-2 axiomatisation contains axioms

$$\begin{array}{l} \mathbf{Activity}(x) \rightarrow \mathbf{PossibleIndividual}(x) \\ \mathbf{Relationship}(x) \rightarrow \mathbf{AbstractObject}(x) \\ \neg(\mathbf{PossibleIndividual}(x) \wedge \mathbf{AbstractObject}(x))\,. \end{array}$$

If we have template definitions

$$\begin{array}{l} AB(x) \rightarrow A(x) \wedge B(x) \\ A(x) \rightarrow \mathbf{Activity}(x) \\ B(x) \rightarrow \mathbf{Relationship}(x) \end{array}$$

and we then assert a formula

$$\phi \quad = \quad AB(a)\,,$$

it follows that a is both a **PossibleIndividual** and an **AbstractObject**, which contradicts ISO 15926-2. While this is easy to see in this case, for larger formulae ϕ, larger collections of templates, and in particular in view of the large number of axioms for ISO 15926-2, it will be more difficult to detect such problems in general. It will therefore be valuable to have a possibility of detecting them automatically.

The following lemma tells us that it is possible to consider this problem by first expanding all templates.

Lemma 3 *Let $\bar{\phi}$ be the expansion of ϕ with respect to the template set S. Then the formula $\Psi \wedge \bigwedge_{\xi \in S} \xi \wedge \phi$ is consistent iff $\Psi \wedge \bar{\phi}$ is consistent.*

Proof: Given any model that satisfies $\Psi \wedge \bigwedge_{\xi \in S} \xi \wedge \phi$, it follows from Lemma 2 that it also satisfies $\bar{\phi}$, and therefore $\Psi \wedge \bar{\phi}$. For the other direction, assume that there is a model $\mathcal{M}$ for $\Psi \wedge \bar{\phi}$. Neither Ψ nor ϕ contain template names, so $\mathcal{M}$ can be extended via an induction over the construction of S to a new model $\mathcal{M}'$ that interprets the basic symbols like $\mathcal{M}$ and all template names N such that the template axioms $N(\cdots) \leftrightarrow \cdots$ are satisfied in $\mathcal{M}'$. This implies that $\mathcal{M}'$ satisfies $\Psi \wedge \bigwedge_{\xi \in S} \xi \wedge \bar{\phi}$, and using Lemma 2 once again, we obtain that it also satisfies ϕ. *Q.E.D.*

The consistency of a $\wedge$-$\vee$-$\exists$-formula ϕ containing only basic symbols in the context of the ISO 15926-2 axiomatisation Ψ must be checked. Standard description logic technology can be used to answer this question. Alternatively, a hyper-tableau method could be preferable for efficiency, as described in annex G.

K.2.1 Translation of ISO 15926-2 language into description logic

The ISO 15926-2 axiomatisation can be translated into a TBox in $\mathcal{ALCIQ}$, a description logic with inverse roles and quantified number restrictions. This is a subset of the well-studied and implemented description logic $\mathcal{SHIQ}$. To see that this is possible, one has to analyse the various axioms used in the first order formalisation of ISO 15926-2, as described in subclause 4.1.

— the one-place predicates $\mathbf{A}(x)$ for EXPRESS entities are mapped to atomic concepts A in the description logic;

— The two-place predicates $\mathbf{hasR}(x)$ for EXPRESS attributes are mapped to roles $hasR$ in the DL;

— the EXPRESS list data type is not actually used in the axioms;

— The implementation $\mathbf{A}(x) \rightarrow \mathbf{B}(x)$ that represents an EXPRESS subtype relationship is mapped to an axiom $A \sqsubseteq B$ in the DL;

— an axiom $\mathbf{A}(x) \leftrightarrow (\mathbf{B}(x) \vee \mathbf{C}(x) \vee \mathbf{D}(x))$ for an ABSTRACT declaration in EXPRESS is mapped to an axiom $A \equiv B \sqcup C \sqcup D$ in the DL;

— an axiom $\neg(\mathbf{A}(x) \wedge (\mathbf{B}(x) \vee \mathbf{C}(x)))$ for a ONEOF declaration in EXPRESS is mapped to an axiom $A \sqsubseteq \neg(B \sqcup C)$;

— an axiom $\mathbf{hasR}(x, y) \rightarrow (\mathbf{A}(x) \vee \mathbf{B}(x))$ to represent the domain of an attribute in EXPRESS is mapped to an axiom $\top \sqsubseteq \forall hasR^{-}.(A \sqcup B)$;

— an axiom $\mathbf{A}(x) \wedge \mathbf{hasR}(x, y) \rightarrow \mathbf{F}(y)$ for a local range restriction in EXPRESS is mapped to an axiom $A \sqsubseteq \forall hasR.F$;

— an axiom $\mathbf{A}(x) \wedge \mathbf{hasR}(x, y) \wedge \mathbf{hasR}(x, z) \rightarrow y = z$ for a $[0, 1]$ cardinality restriction in EXPRESS is mapped to an axiom $A \sqsubseteq hasR \text{ max } 1 \top$ in the DL;

— an axiom $\mathbf{A}(x) \rightarrow \exists y(\mathbf{hasR}(x, y))$ additionally needed for a $[1, 1]$ cardinality restriction in EXPRESS is mapped to an axiom $A \sqsubseteq \exists hasR.\top$ in the DL;

— an axiom $\mathbf{A}(x) \wedge \mathbf{A}(y) \wedge \mathbf{hasR}(x, z) \wedge \mathbf{hasR}(y, z) \rightarrow x = y$ for a UNIQUE rule in EXPRESS is mapped to an axiom $\top \sqsubseteq \mathit{hasR}^{-} \max\ 1\ A$ in the DL.

This provides the description logic TBox axioms for ISO 15926-2.

K.2.2 Consistency Check for Template Expansion

To check consistency, the $\wedge$-$\vee$-$\exists$-formula ϕ is skolemised, replacing every existentially bound variable by a new constant symbol, producing $\hat{\phi}$. Then $\Psi \wedge \bar{\phi}$ is consistent iff $\Psi \wedge \hat{\phi}$ is consistent. This again can be checked by transforming $\hat{\phi}$ into disjunctive normal form

$$\hat{\phi} \leftrightarrow \hat{\phi}_1 \vee \cdots \vee \hat{\phi}_n$$

where the $\hat{\phi}_i$ are conjunctions of ground atoms over Σ_0. These conjunctions can be read as ABoxes over the previously defined TBox-translation of ISO 15926-2. Then $\Psi \wedge \hat{\phi}$ is consistent iff one of these ABoxes is consistent with the TBox. This type of problem is known as an *ABox consistency problem* and can be handled by state-of-the-art DL reasoning systems.

EXAMPLE An example of expansion and consistency check is provided in annex F.

Bibliography

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

[2] ISO/TR 9007, *Information processing systems — Concepts and terminology for the conceptual schema and the information base.*

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

[4] ISO 10303-11:2004, *Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.*

[5] ISO 15926-1:2004, *Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities — Part 1: Overview and fundamental principles.*

[6] ISO 15926-2:2003, *Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities — Part 2: Data model.*

[7] ISO/TS 15926-4, *Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities — Part 4: Initial reference data.*

[8] ISO/TS 15926-8:2011, *Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities — Part 8: Implementation methods for the integration of distributed systems: Web Ontology Language (OWL) implementation.*

[9] *OWL 2 web ontology language direct semantics.* [online] W3C Recommendation 27 October 2009. Available from World Wide Web: <http://www.w3.org/TR/owl2-direct-semantics/>.

[10] BAADER Franz. *DL terminology, DL handbook,* etc. Appendix 1, pages 495–505.

[11] BAADER Franz, CALVANESE Diego, McGUINNESS Deborah L., NARDI Daniele and PATEL-SCHNEIDER Peter F., editors. *The description logic handbook: theory, implementation, and applications.* Cambridge University Press, 2003.

[12] BEZEM M. *Website for geometric/coherent logic.* Available from World Wide Web: <http://www.ii.uib.no/~bezem/GL>.

[13] BEZEM M.A. *On the undecidability of coherent logic*. In A. Middeldorp e.a., editors, *Processes, terms and cycles: steps on the road to infinity.* LNCS 3838, pages 6–13, Springer-Verlag, Berlin, 2005.

[14] BEZEM M.A. and COQUAND T. *Automating coherent logic.* In G. Sutcliffe and A. Voronkov, editors, Proceedings LPAR-12, LNCS 3835, pages 246–260, Springer-Verlag, Berlin, 2005.

[15] BLASS A. *Topoi and computation.* Bulletin of the EATCS 36:57–65, 10-1998.

[16] BOVE, Ana and ARBILLA, Laura. *A confluent calculus of macro expansion and evaluation.* SIGPLAN lisp pointers, V(1):278–287, 1992. Available from World Wide Web: <http://www.cse.chalmers.se/~bove/Papers/papers.html>.

[17] GLENDINNING, Ian and VALEN-SENDSTAD, Magne. *Characterization methodology for ISO/TS 15926-7 templates.* [online]. In progress, 2008. Available from

World Wide Web: <https://www.posccaesar.org/browser/projects/IDS-ADI/Part7/Part7SpecificationsMethodologies/P7L_Characterization_Methodology_IDS-120-001_Iss_2x.doc>.

[18] HE L., CHAO Y. and ITOH H. *R-SATCHMO refinements on I-SATCHMO.* Journal of logic and computation 14(2):117–143, 2004.

[19] HORN A. *Sentences which are true of direct unions of algebras.* Journal of symbolic logic 16(1):14–21, 1951.

[20] KOWALSKI R.A. *Predicate logic as programming language.* Procedings IFIP congress, pages 569-574, 1974.10.

[21] MANTHEY R. and BRY F. *SATCHMO a theorem prover implemented in prolog.* In E. Lusk and R. Overbeek, editors, Proceedings of the 9th conference on automated deduction, lecture notes in computer science 310, pages 415–434, Springer, 1988.

[22] MAYR Richard and NIPKOW Tobias. *Higher-order rewrite systems and their confluence.* Theoretical computer science, vol. 192, 3–29, 1998.

[23] NIPKOW T. *Higher-order critical pairs*. Proceedings of the 6th annual symposium on logic in computer science, ed. G. Kahn, IEEE, pages 342–349, 1991.

[24] SKOLEM T. *Logisch-kombinatorische Untersuchungen "uber die Erfullbarkeit und Beweisbarkeit mathematischen Satze nebst einem Theoreme uber dichte Mengen."* Videnskapsselskapets skrifter I, Matematisk-naturvidenskabelig klasse, Videnskabsakademiet i Kristiania 4:1–36, 1920.

[25] SKOLEM T. *Selected works in logic*, edited by J.E. Fenstad. Universitetsforlaget, Oslo, 1970.

Index

ICS 25.040.40; 75.020

Price based on 116 pages