
**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





COPYRIGHT PROTECTED DOCUMENT

© ISO 2011

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

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

Published in Switzerland

Contents	Page
Foreword	vii
Introduction	viii
1 Scope	1
2 Terms, definitions, and abbreviated terms	1
2.1 Terms and definitions	1
2.2 Abbreviated terms	5
3 Fundamental concepts and assumptions	6
3.1 General	6
3.2 Concepts and models	6
3.2.1 ISO 15926-2 data model	6
3.2.2 ISO/TS 15926-4 reference data	7
3.2.3 User-defined taxonomy	7
3.2.4 Templates	7
4 Modelling basics	7
4.1 ISO 15926-2 data model in first-order logic	7
4.2 Logical template definition	10
4.3 Proto-templates	10
4.3.1 entityTriple	11
4.4 Diagrams	11
5 Template specification	13
5.1 Requirements on template, general	13
5.2 Template signatures	14
5.3 Template specialization	14
5.4 Verification of compliance with ISO 15926-2	15
6 Templates for individuals	15
6.1 Purpose	15
6.2 Reference items needed	15
6.3 Initial set	16
6.3.1 Template ClassificationOfIndividual	16
6.3.2 Template ClassificationOfRelationship	16
6.3.3 Template InstanceOfRelation	17
6.3.4 Template IdentificationByNumber	18
6.3.5 Template ClassifiedIdentification	19
6.3.6 Template LocationOfActivity	20
6.3.7 Template BeginningOfIndividual	21
6.3.8 Template BeginningEndOfIndividual	21
6.3.9 Template BeginningOfTemporalPart	22
6.3.10 Template BeginningEndLocationOfActivity	22
6.3.11 Template InstanceOfIndirectProperty	23
6.3.12 Template RealMagnitudeOfProperty	23
6.3.13 Template IndirectPropertyScaleReal	24
6.3.14 Template StatusApproval	24
6.3.15 Template ClassifiedInvolvement	25
6.3.16 Template InvolvementStatus	25
6.3.17 Template InvolvementStatusBeginning	26

6.3.18	Template SuccessionOfInvolvementByReference	27
6.3.19	Template SuccessionOfInvolvementInActivity	27
7	Templates for classes	27
7.1	Purpose	27
7.2	Reference data items needed	28
7.2.1	Reference classes	28
7.2.2	Reference relations	29
7.3	Representing complex classes	29
7.4	Relation constraints	30
7.4.1	Relations: Domain and co-domain	30
7.4.2	Existential and universal constraints	32
7.4.3	<i>At-most n</i> subrelation constraints	33
7.5	Initial set	37
7.5.1	Template ClassificationOfClass	37
7.5.2	Template ClassificationOfClassOfIndividual	37
7.5.3	Template ClassificationOfClassOfRelationship	38
7.5.4	Template RelationOfIndividualsToIndividuals	39
7.5.5	Template SpecializationOfIndividualRelation	40
7.5.6	Template EnumeratedSetOf2Classes	41
7.5.7	Template EnumeratedSetOf3Classes	42
7.5.8	Template UnionOf2Classes	42
7.5.9	Template IntersectionOf2Classes	43
7.5.10	Template DifferenceOf2Classes	44
7.5.11	Template RelativeComplementOf2Classes	44
7.5.12	Template DisjointnessOf2Classes	45
7.5.13	Templates SpecializationAsEnd1UniversalRestriction, Specializa- tionAsEnd2UniversalRestriction	46
7.5.14	Templates CardinalityMin, CardinalityMax, CardinalityMinMax	47
7.5.15	Cardinality Assignment Templates	48
7.5.16	Template TimeRepresentation	51
7.5.17	Template MagnitudeOfProperty	52
7.5.18	Template LowerUpperOfNumberRange	52
7.5.19	Template LowerUpperOfPropertyRange	53
7.5.20	Template LowerUpperMagnitudeOfPropertyRange	54
7.5.21	Template PropertyRangeRestrictionOfClass	56
7.5.22	Template PropertyRangeMagnitudeRestrictionOfClass	56
7.5.23	Template SymbolOfScale	58
7.5.24	Template DimensionUnitNumberRangeOfScale	59
7.5.25	Template ClassInvolvementStatusBeginning	60
7.5.26	Template ClassInvolvementSuccession	61
8	Templates as reference data	62
8.1	Template signatures and template axioms	62
8.2	Representation as class_of_multidimensional_object	63
8.2.1	Roles constrained by RDL constructs	63
8.2.2	Roles constrained by entity type only	63
Annex A	(normative) Information object registration	64
A.1	Document identification	64
Annex B	(normative) Listing: ISO 15926-2 in first-order logic	65
B.1	General	65

B.2	Universe axiom	65
B.3	Subtype axioms	65
B.4	Abstract axioms	68
B.5	Disjoint axioms	68
B.6	Role axioms	73
B.7	Additional range restriction axioms	81
Annex C	(normative) Listing: proto-templates	83
C.1	Proto-templates for relational entity types	84
C.2	Proto-templates for subtypes of relational entity types	86
C.3	entityTriple	88
Annex D	(informative) Table: proto-templates	89
Annex E	(informative) Recursive vs. non-recursive template expansion	96
E.1	Nomenclature	96
E.2	Recursion or not?	96
E.3	Further technical issues	98
Annex F	(informative) Template expansion: example	99
F.1	Example of template expansion	99
F.2	Result of expansion according to template axioms	99
F.3	Instantiating existential quantifiers	101
F.4	Verification of consistency	103
Annex G	(informative) Consistency checking via coherent logic	104
Annex H	(informative) Formal constraints beyond templates	107
Annex J	(normative) Semantics of templates	108
J.1	Rewrite rule	108
J.2	Pattern rewriting system corresponding to a template set	108
J.3	Template expansion	109
Annex K	(normative) Properties of template expansion	110
K.1	Logical readings of template definitions	110
K.2	Decidability of consistency with ISO 15926-2	110
K.2.1	Translation of ISO 15926-2 language into description logic	111
K.2.2	Consistency Check for Template Expansion	112
Bibliography		113
Index		115

Figures

Figure 1	— An overview of the integration architecture of ISO/TS 18876-1	6
Figure 2	— Diagram of a class	11
Figure 3	— Diagram of a relation	11
Figure 4	— Diagram of a relationship	12
Figure 5	— Diagram showing relation roles	12
Figure 6	— Diagram showing cardinalities	12
Figure 7	— Diagram of specialization	13
Figure 8	— Diagram of classification	13

Figure 9 — Example template ClassificationOfIndividual	17
Figure 10 — Example template ClassificationOfRelationship	17
Figure 11 — Example template InstanceOfRelation	18
Figure 12 — Example template IdentificationByNumber	19
Figure 13 — Example template ClassifiedIdentification	20
Figure 14 — Example template LocationOfActivity	20
Figure 15 — Example template BeginningEndOfIndividual	22
Figure 16 — Use of EmptyClass	29
Figure 17 — Relation: Permitted Ambient Temperature	31
Figure 18 — Relation: Celsius	31
Figure 19 — Existential constraint: Permitted Ambient Temperature	33
Figure 20 — Precisely-one constraint: Permitted Ambient Temperature	34
Figure 21 — Classification for universal constraint	34
Figure 22 — Universal constraint pattern	35
Figure 23 — Universal constraint: Equipment, Temperature	35
Figure 24 — Example for <i>at most</i> constraints	36
Figure 25 — Combining universal constraints and cardinalities	36
Figure 26 — Example template ClassificationOfClass	37
Figure 27 — Example template ClassificationOfClassOfIndividual	38
Figure 28 — Example template ClassificationOfClassOfRelationship	39
Figure 29 — Example template RelationOfIndividualsToIndividuals	40
Figure 30 — Example template SpecializationOfIndividualRelation	41
Figure 31 — Example template EnumeratedSetOf2Classes	42
Figure 32 — Example template UnionOf2Classes	43
Figure 33 — Example template RelativeComplementOf2Classes	45
Figure 34 — Example template DisjointnessOf2Classes	46
Figure 35 — Example template SpecializationAsEnd2UniversalRestriction	47
Figure 36 — Example template CardinalityMinMax	48
Figure 37 — Example template CardinalityEnd1MinMax	51
Figure 38 — Example template MagnitudeOfProperty	52
Figure 39 — Example template LowerUpperOfNumberRange	53
Figure 40 — Example template LowerUpperOfPropertyRange	54
Figure 41 — Example template LowerUpperMagnitudeOfPropertyRange	55
Figure 42 — Example template PropertyRangeRestrictionOfClass	56
Figure 43 — Example template PropertyRangeMagnitudeRestrictionOfClass	58
Figure 44 — Example template SymbolOfScale	59
Figure 45 — Example template DimensionUnitNumberRangeOfScale	60

Tables

Table 1 — Reference items: Templates for individuals	15
Table 2 — Reference items: Templates for classes	28
Table D.1 — Compact listing of Proto-templates	90

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 criteria 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, *individual* 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.

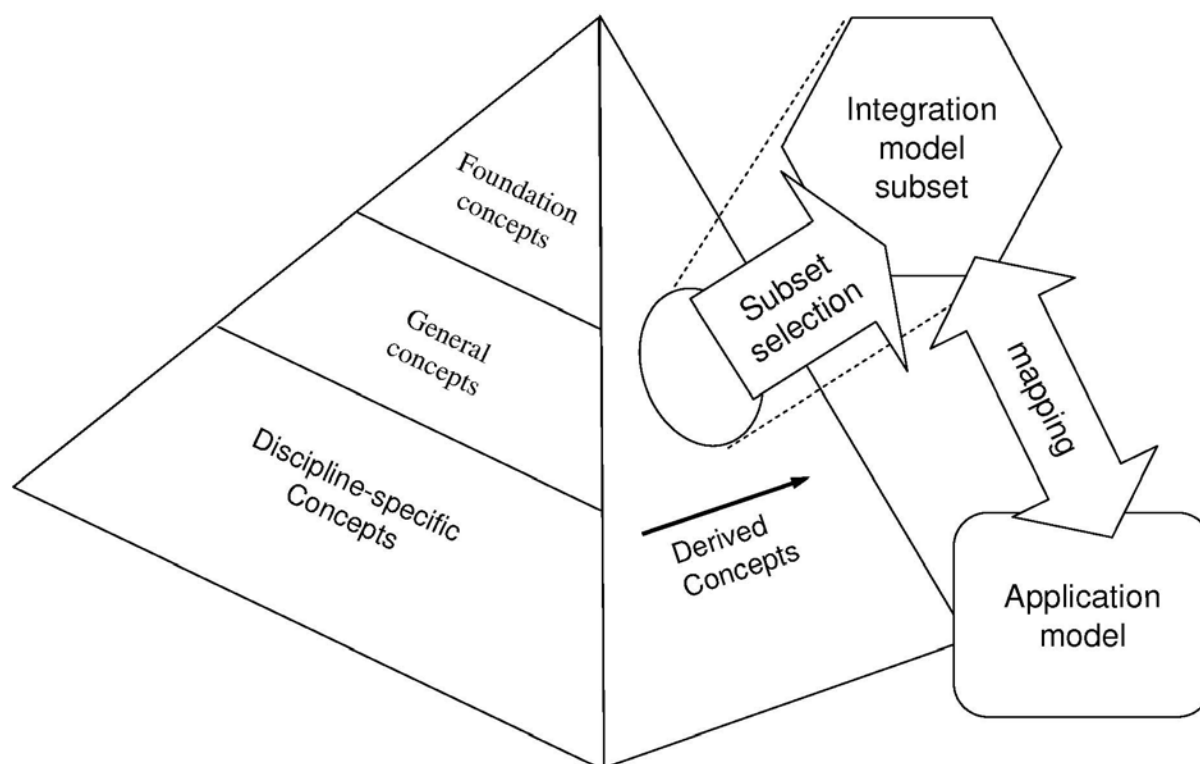


Figure 1 — An overview of the integration architecture of ISO/TS 18876-1

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, “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

$$A(x)$$

where $A(c)$ 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 allows 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 $\mathbf{A}(c)$ 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.

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

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:

$$\mathbf{A}(x) \rightarrow \exists y(\mathbf{hasR}(x, y)) \tag{1}$$

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

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:

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

where $N \notin \Sigma$ is a predicate symbol called the name of the template, $x, y, \dots$ 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 $x, y, \dots$ 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 $\text{names}(S) := \{N \mid N(\dots) \leftrightarrow \phi \in S\}$ 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 $N(\dots) \leftrightarrow \phi$ 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

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

is a valid template set, but

$$\left\{ \begin{array}{l} A(x) \leftrightarrow \exists y.(B(y) \wedge \mathbf{R}(x, y)), \\ B(x) \leftrightarrow \mathbf{C}(x) \vee A(x) \end{array} \right\}$$

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

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

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

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

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.

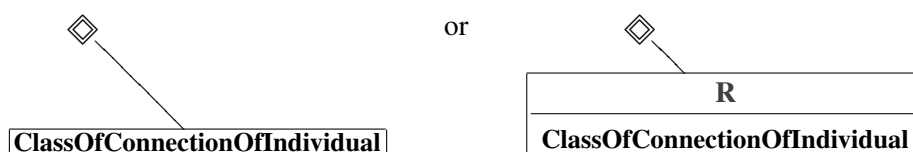


Figure 3 — Diagram of a relation

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.

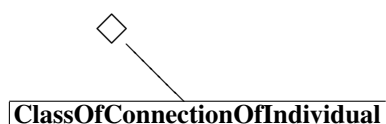


Figure 4 — Diagram of a relationship

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.

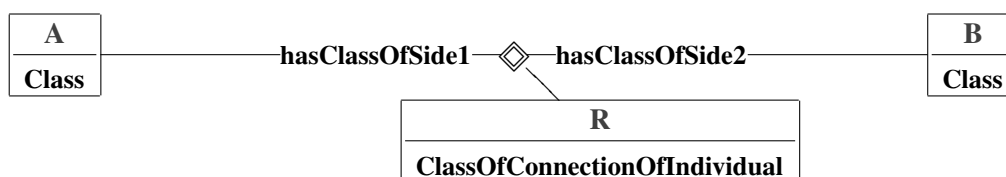


Figure 5 — Diagram showing relation roles

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.

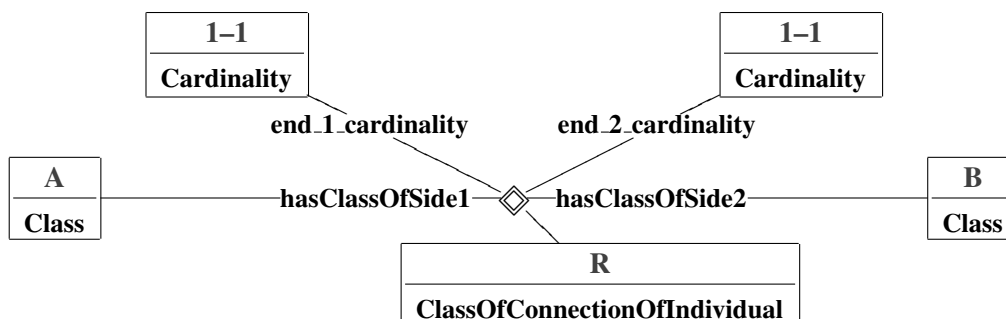


Figure 6 — Diagram showing cardinalities

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**.

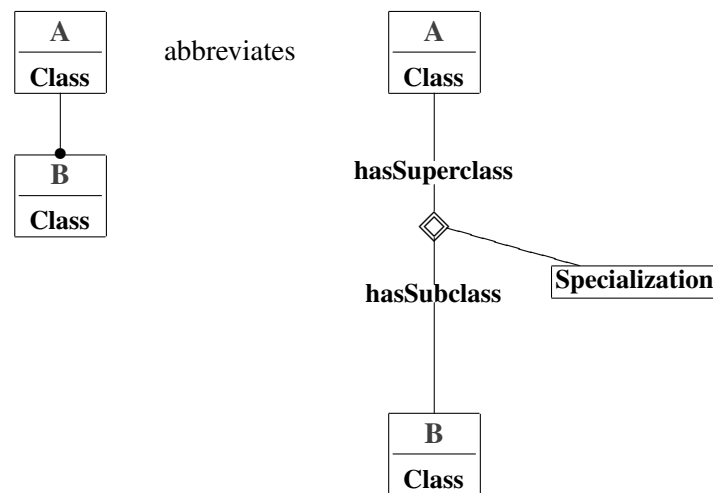


Figure 7 — Diagram of specialization

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

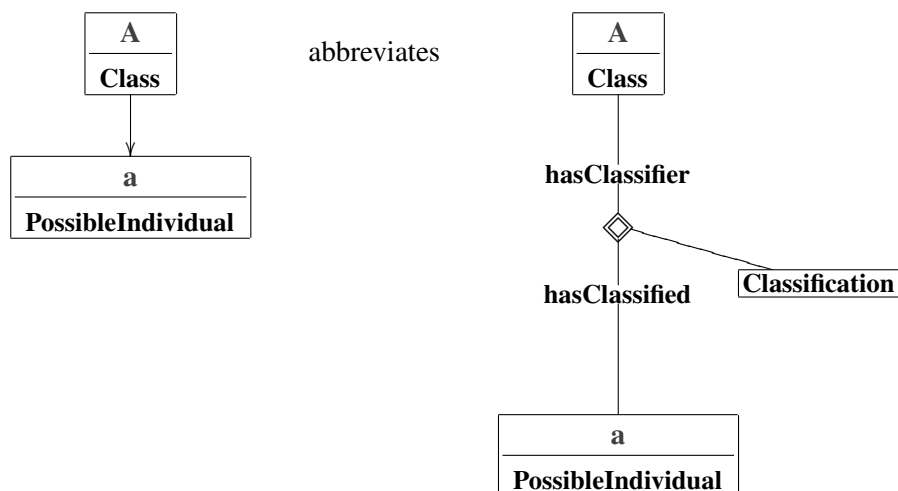


Figure 8 — Diagram of classification

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 be 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 criterion 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_i is type of relationship_i that indicates that the classified thing_i is a member of the classifier class_i .

Order	Role name	Role type
1	Individual	PossibleIndividual
2	Class	ClassOfIndividual

ClassificationOfIndividual(x₁, x₂) ↔

PossibleIndividual(x₁) ∧

ClassOfIndividual(x₂) ∧

ClassificationTemplate(x₁, x₂)

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*.

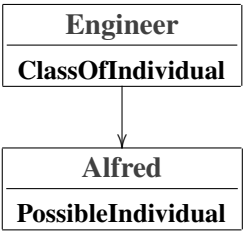


Figure 9 — Example template ClassificationOfIndividual

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 $ClassificationOfRelationship(\langle \text{Alfred}, \text{ACME Co.} \rangle, \text{Employment})$ 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, ' $\langle \dots \rangle$ ', to name an ordered pair is not normative in this part of ISO 15926.

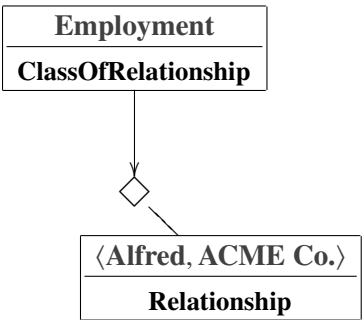


Figure 10 — Example template ClassificationOfRelationship

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

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

ClassOfRelationshipWithSignature(x_1) $\wedge$

Thing(x_2) $\wedge$

Thing(x_3) $\wedge$

$\exists u(OtherRelationshipTriple(u, x_2, x_3) \wedge$

$ClassificationOfRelationship(u, x_1))$

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.

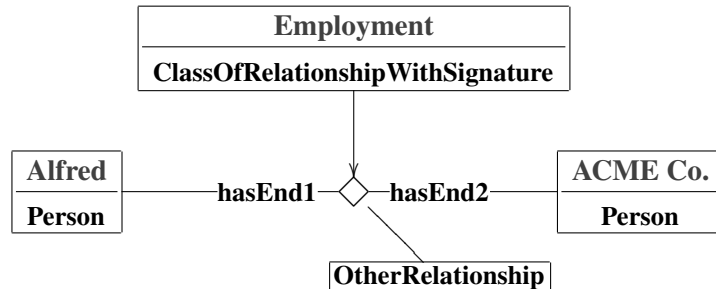


Figure 11 — Example template InstanceOfRelation

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, π).

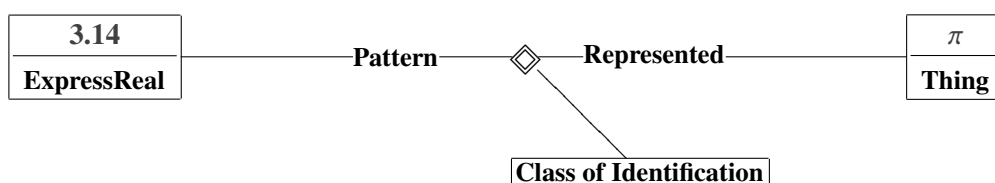


Figure 12 — Example template *IdentificationByNumber*

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) $\wedge$

$\exists u(\text{ClassOfIdentificationTriple}(u, x_2, x_1) \wedge$

ClassificationTemplate(u, x_3))

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

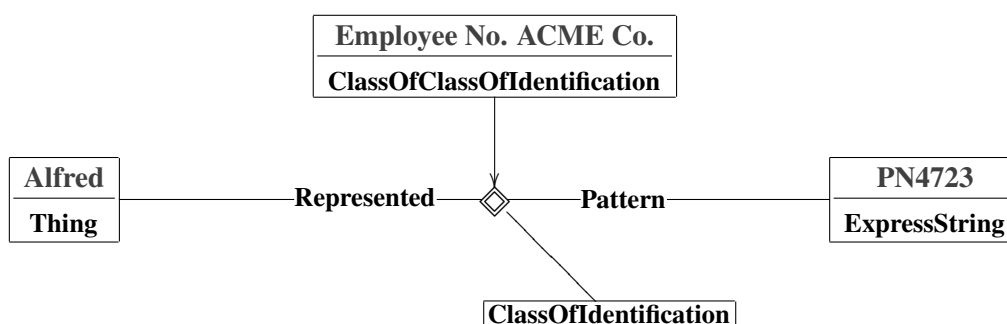


Figure 13 — Example template ClassifiedIdentification

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.

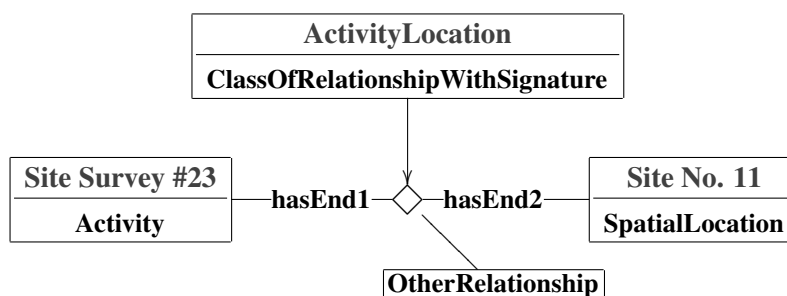


Figure 14 — Example template LocationOfActivity

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	RepresentationOfGregorianCalendarAndUtcTime

BeginningOfIndividual(x₁, x₂) ↔

PossibleIndividual(*x₁*) ∧
RepresentationOfGregorianCalendarAndUtcTime(*x₂*) ∧
 $\exists u(\text{PointInTime}(u) \wedge \text{BeginningTemplate}(u, x_1) \wedge$
ClassOfRepresentationOfThingTemplate(*x₂*, *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	RepresentationOfGregorianCalendarAndUtcTime
3	End time	RepresentationOfGregorianCalendarAndUtcTime

BeginningEndOfIndividual(x₁, x₂, x₃) ↔

PossibleIndividual(*x₁*) ∧
RepresentationOfGregorianCalendarAndUtcTime(*x₂*) ∧
RepresentationOfGregorianCalendarAndUtcTime(*x₃*) ∧
 $\exists u(\text{PointInTime}(u) \wedge \text{BeginningTemplate}(u, x_1) \wedge$
ClassOfRepresentationOfThingTemplate(*x₂*, *u*) ∧
 $\exists v(\text{PointInTime}(v) \wedge \text{EndTemplate}(v, x_1) \wedge$
ClassOfRepresentationOfThingTemplate(*x₃*, *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.

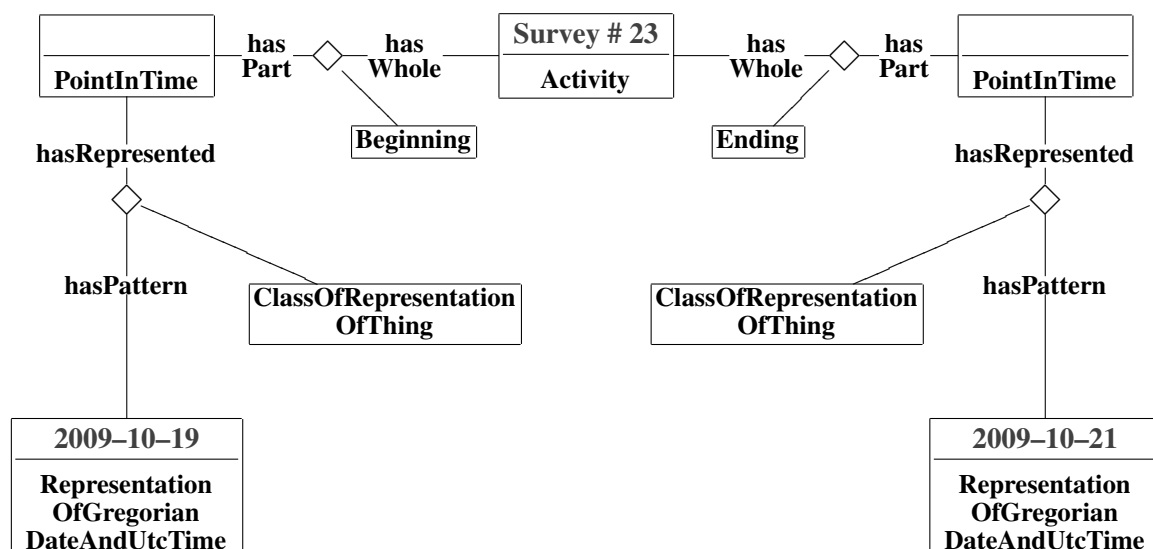


Figure 15 — Example template BeginningEndOfIndividual

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(x₁, x₂, x₃) ↔

PossibleIndividual(x₁) ∧

PossibleIndividual(x₂) ∧

RepresentationOfGregorianDateAndUtcTime(x₃) ∧

TemporalWholePartTemplate(x₁, x₂) ∧

BeginningOfIndividual(x₁, x₃)

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 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) $\wedge$
 $\exists u(\text{MagnitudeOfProperty}(x_1, u, x_3) \wedge$
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 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) $\wedge$
 $\exists u(\text{InstanceOfIndirectProperty}(x_1, x_2, u) \wedge$
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

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

Relationship(x_1) $\wedge$

ClassOfApprovalByStatus(x_2) $\wedge$

PossibleIndividual(x_3) $\wedge$

$\exists u(\text{ApprovalTriple}(u, x_1, x_3) \wedge$

ClassificationTemplate(u, x_2))

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

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

Thing(x_1) $\wedge$

Activity(x_2) $\wedge$

ClassOfInvolvementByReference(x_3) $\wedge$

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

ClassificationTemplate(u, x_3))

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

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

Thing(x_1) $\wedge$

Activity(x_2) $\wedge$

ClassOfInvolvementByReference(x_3) $\wedge$

ClassOfApprovalByStatus(x_4) $\wedge$

PossibleIndividual(x_5) $\wedge$

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

$\text{ClassificationTemplate}(u, x_3) \wedge$

$\text{StatusApproval}(u, x_4, x_5))$

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.

$\text{InvolvementStatusBeginning}(a, b, c, d, e, f)$ 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

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

Thing(x_1) $\wedge$

Activity(x_2) $\wedge$

ClassOfInvolvementByReference(x_3) $\wedge$

ClassOfApprovalByStatus(x_4) $\wedge$

PossibleIndividual(x_5) $\wedge$

RepresentationOfGregorianDateAndUtcTime(x_6) $\wedge$

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

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

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

SuccessionOfInvolvementByReference(x_1, x_2) $\leftrightarrow$

InvolvementByReference(x_1) $\wedge$

InvolvementByReference(x_2) $\wedge$

InstanceOfRelation(**InvolvementSuccession**, x_1, x_2)

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

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

Thing(x_1) $\wedge$

Thing(x_2) $\wedge$

Activity(x_3) $\wedge$

$\exists u_1 \exists u_2$ (

InvolvementByReferenceTriple(u_1, x_1, x_3) $\wedge$

InvolvementByReferenceTriple(u_2, x_2, x_3) $\wedge$

SuccessionOfInvolvementByReference(u_1, u_2))

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.

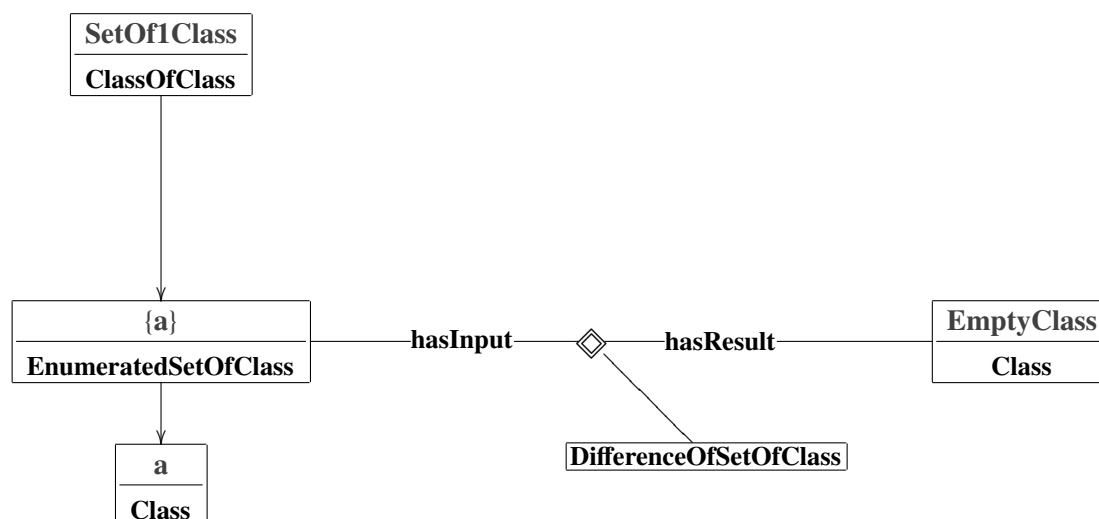


Figure 16 — Use of EmptyClass

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 $a_1, a_2, \dots, a_n$. With A in the **hasInput** role of an instance of **UnionOfSetOfClass**,