INTERNATIONAL STANDARD

ISO/IEC 11179-32

First edition
2023-01

Information technology — Metadata registries (MDR) —

Part 32: **Metametamodel for concept system registration**

Technologies de l'information — Registres de métadonnées (RM) —

Partie 32: Métamodèle pour l'enregistrement de systèmes de concepts

Reference number
ISO/IEC 11179-32:2023(E)

Contents

Page

List of Figures

List of Tables

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC/JTC 1, *Information technology*, Subcommittee SC 32, *Data management and interchange*.

A list of all parts in the ISO/IEC 11179 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user’s national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

ISO/IEC 11179-3 specifies the structure of a Metadata Registry (MDR) and provides a metamodel for registry common facilities. That metamodel is intended to be extended by other parts of ISO/IEC 11179 for specific purposes.

This first edition of ISO/IEC 11179-32, is part of a restructuring of ISO/IEC 11179-3:2013, which has now been broken into multiple parts. This document provides a metamodel for registering metadata about concept systems and binary relations in a Metadata Registry (MDR), as extensions to the registry metamodel specified in ISO/IEC 11179-3.

In Clauses 7 and 8, this document uses:

— **bold** font to highlight terms which represent metadata objects specified by the metamodel;

— normal text for terms which represent concepts defined in Clause 3.

EXAMPLE **Concept** (7.2.2.1) is a class each instance of which models a concept.

Information technology — Metadata registries (MDR) —

Part 32: Metamodel for concept system registration

1 Scope

This document provides a specification for an extension to a metadata registry (MDR), as specified in ISO/IEC 11179-3:2023, in which metadata that describes concept systems can be registered.

The specification in this document, together with the relevant clauses of the specification in ISO/IEC 11179-3, provides the ability to record the following metadata:

— concept systems and associated concepts;

— relations among concepts in a concept system;

— assertions about concepts in a concept system.

The metamodel in this document is intended to support the full description of a concept system, including ontologies.

Where there is a requirement to register an ontology where the details are defined elsewhere, consider using ISO/IEC 19763-3[8] instead.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 11179-3:2023, *Information technology — Metadata registries (MDR) — Part 3: Metamodel for registry common facilities*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 11179-3 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1
concept
unit of knowledge created by a unique combination of characteristics

Note 1 to entry: Concepts are not necessarily bound to particular natural languages. They are, however, influenced by the social or cultural background which often leads to different categorizations.

Note 2 to entry: This is the concept "concept" as used and designated by the term "concept" in terminology work. It is a very different concept from that designated by other domains such as industrial automation or marketing.

Note 3 to entry: A concept is independent of its representation.

[SOURCE: ISO 1087:2019, 3.2.7, modified — Reference to definition of *characteristics* removed. Note 3 to entry added.]

3.2
concept system
system of concepts
set of *concepts* (3.1) structured in one or more related domains according to the *concept relations* (3.3) among its concepts

[SOURCE: ISO 1087:2019, 3.2.28]

3.3
relation
concept relation
sense in which *concepts* (3.1) may be connected, via constituent *relation roles* (3.4)

EXAMPLE Causality is a relation with two constituent roles: cause and effect.

Note 1 to entry: The related concepts may be general or individual concepts.

3.4
relation role
role (3.5) that a *concept* (3.1) plays in a *relation* (3.3)

3.5
role
specified responsibilities

3.6
link
member of a *relation* (3.3)

3.7
link end
end of a *link* (3.6), identifying the *relation role* (3.4) played by a *concept* (3.1) in the link

3.8
binary relation
relation (3.3) with *arity* (3.9) equal to 2 (i.e. whose members all have two ends)

Note 1 to entry: Most common semantic relations are binary, e.g. “equals”, “less than”, “greater than”, “is part of”, etc. An example of a relation which is not binary is “betweenness” (e.g. A is between B and C.).

3.9
arity
number of arguments that a function takes

3.10
reflexivity
characterization of a *binary relation* (3.8) as reflexive, irreflexive or antireflexive

Note 1 to entry: A binary relation, R, is reflexive if for all x, R(x,x) is true. Equality is an example of a reflexive relation.

Note 2 to entry: A binary relation, R, is irreflexive if it is not reflexive. i.e., R(x,x) is not necessarily true for all x.

Note 3 to entry: A binary relation, R, is antireflexive if for all x, R(x,x) is false. Inequality is an example of an antireflexive relation. An antireflexive relation is also irreflexive, but antireflexive is a more specific characterization.

3.11
symmetry
characterization of a *binary relation* (3.8) as symmetric, asymmetric or antisymmetric

Note 1 to entry: A binary relation, R, is symmetric if for all x, y: R(x,y) implies R(y,x).

EXAMPLE 1 Symmetric relations include: “equals”, “not equals”, “within-2-miles-of”, etc.

Note 2 to entry: Symmetry does not imply *reflexivity* (3.18).

EXAMPLE 2 the “inequality” relation is symmetric, but antireflexive.

Note 3 to entry: A binary relation, R, is asymmetric if for all x,y: R(x,y) does not imply R(y,x).

EXAMPLE 3 Asymmetric relations include: “less than”, “likes”, “father of”, etc.

Note 4 to entry: A binary relation, R, is anti-symmetric if for all x,y: R(x,y) implies not R(y,x). An antisymmetric relation is also asymmetric, but antisymmetric is a more specific characterization.

EXAMPLE 4 “less than” is an antisymmetric relation.

Note 5 to entry: An asymmetric relation is not necessarily antisymmetric.

EXAMPLE 5 Less than or equals.

3.12
transitivity
characterization of a *binary relation* (3.8) as: transitive, intransitive or antitransitive

Note 1 to entry: A binary relation, R, is transitive, if for all x,y,z: R(x,y) and R(y,z) implies R(x,z). Examples of transitive relations include equality, less than and less than or equals.

Note 2 to entry: A binary relation, R, is intransitive if it is not transitive i.e. R(x,y) and R(y,z) does not imply R(x,z).

Note 3 to entry: A binary relation, R, is antitransitive if for all x,y,z: R(x,y) and R(y,z) implies not R(x,z).

Note 4 to entry: An antitransitive relation is also intransitive, but antitransitive is a more specific characterization.

3.13
object
anything perceivable or conceivable

Note 1 to entry: Objects can be material (e.g. “engine”, “sheet of paper”, “diamond”), immaterial (e.g. “conversion ratio”, “project plan”) or imagined (e.g. “unicorn”, “scientific hypothesis”).

[SOURCE: ISO 1087:2019, 3.1.1]

3.14
property
feature of an *object* (3.13)

EXAMPLE 1 “Being made of wood” as a property of a given “table”.

EXAMPLE 2 “Belonging to person A” as a property of a given “pet”.

EXAMPLE 3 “Having been formulated by Einstein” as a property of the equation “$E = mc^2$”.

EXAMPLE 4 “Being compassionate” as a property of a given “person”.

EXAMPLE 5 “Having a given cable” as a property of a given “computer mouse”.

Note 1 to entry: One or more objects can have the same property.

[SOURCE: ISO 1087:2019, 3.1.3]

3.15
characteristic
abstraction of a *property* (3.14)

EXAMPLE “Having a cable for connecting with a computer” as a characteristic of the concept “cord mouse”.

Note 1 to entry: Characteristics are used for describing *concepts* (3.6).

[SOURCE: ISO 1087:2019, 3.2.1]

3.16
notation
formal syntax and associated semantics for the representation of information

EXAMPLE UML,[5],[6],[7] MOF, OCL, OWL[11]/RDF,[12] SKOS,[13] CGIF,[9] XCL,[9] XTM[17] or ISO/IEC 11404[4]

Note 1 to entry: A formal syntax consists of a set of symbols and the rules for their use.

Note 2 to entry: Formal syntax is often intended for machine processing.

[SOURCE: ISO/IEC 11179-3:2023, 3.2.36]

3.17
assertion
sentence or proposition in logic which is asserted (or assumed) to be true

3.18
cardinality
number of elements in a set

Note 1 to entry: cf. *multiplicity* (3.19)

Note 2 to entry: Adapted from ISO/IEC 19501:2005,[5] Glossary.

3.19
multiplicity
specification of the range of allowable *cardinalities* (3.18) that a set may assume

Note 1 to entry: Multiplicity specifications may be given for roles within *associations* (ISO/IEC 11179-3:2023, 3.1.5)

Note 2 to entry: A multiplicity is a (possibly infinite) subset of the nonnegative integers

Note 3 to entry: Adapted from ISO/IEC 19501:2005,[5] Glossary.

3.20
taxonomy
type of hierarchy which deals with generalization/specialization relationships

Note 1 to entry: cf. *meronomy* (3.21)

3.21
meronomy
type of hierarchy which deals with part-whole relationships

Note 1 to entry: cf. *taxonomy* (3.20)

4 Abbreviated terms

CD	Conceptual Domain
CL[9]	Common Logic

CLIF[9]	Common Logic Interchange Format
OWL[11]	Web Ontology Language
OWL-DL	OWL Description Logic
RDF[12]	Resource Description Framework
SKOS[13]	Simple Knowledge Organization System
UML[5][6][7]	Unified Modeling Language
URI	Uniform Resource Identifier
W3C[15]	World Wide Web Consortium
XCL[9]	eXtended Common Logic markup language
XML[16]	eXtensible Markup Language
XTM[17]	XML Topic Maps

5 Conformance

5.1 Overview of conformance

Conformance rules for a Metadata Registry are specified in ISO/IEC 11179-3:2023, Clause 4. The clause "Degree of Conformance" is repeated here for convenience. The subsequent subclauses extend the rules from ISO/IEC 11179-3.

5.2 Degree of conformance

5.2.1 General

The distinction between "strictly conforming" and "conforming" implementations is necessary to address the simultaneous needs for interoperability and extensions. This document describes specifications that promote interoperability. Extensions are motivated by needs of users, vendors, institutions and industries, and:

a) are not directly specified by this document;

b) are specified and agreed to outside this document;

c) may serve as trial usage for future editions of this document.

A strictly conforming implementation can be limited in usefulness but is maximally interoperable with respect to this document. A conforming implementation can be more useful but can be less interoperable with respect to this document.

5.2.2 Strictly conforming implementations

A strictly conforming implementation:

a) shall support all mandatory, optional and conditional classes, attributes, datatypes and associations;

b) shall not use, test, access or probe for any extension features nor extensions to classes, attributes, datatypes, associations or any combination thereof;

c) shall not recognize, nor act on, nor allow the production of classes, attributes, datatypes, associations or any combination thereof that are dependent on any unspecified, undefined or implementation-defined behaviour.

NOTE The use of extensions to the metamodel can cause undefined behaviour.

5.2.3 Conforming implementations

A conforming implementation:

a) shall support all mandatory, optional and conditional classes, attributes, datatypes and associations;

b) as permitted by the implementation, may use, test, access or probe for extension features or extensions to classes, attributes, datatypes, associations or any combination thereof;

c) may recognize, act on or allow the production of classes, attributes, datatypes, associations or any combination thereof that are dependent on implementation-defined behaviour.

NOTE 1 All strictly conforming implementations are also conforming implementations.

NOTE 2 The use of extensions to the metamodel can cause undefined behaviour.

5.3 Conformance by feature

Conformance claims may be made to Clause 7 and optionally Clause 8, or to specific features within these clauses. Those clauses are also dependent upon one or more other clauses of ISO/IEC 11179-3, so conformance to all or part of those clauses shall be understood to imply conformance also to relevant provisions specified in one or more of the clauses in ISO/IEC 11179-3.

A conformance statement shall specify exactly the features supported and not supported.

5.4 Registry conformance

5.4.1 Standard registry profiles

This document specifies the following standard profiles in addition to those specified in ISO/IEC 11179-3:2023, 4.4.2.

— **Concept System Registry**: Implements Clause 7 in addition to all provisions of the "Basic Registry" profile of ISO/IEC 11179-3:2023, 4.4.2;

— **Concept System and Binary Relations Registry**: Implements Clauses 7 and 8 in addition to all provisions of the "Basic Registry" profile of ISO/IEC 11179-3:2023, 4.4.2;

— **Concept System Registry with mapping**: Implements Clause 7 in addition to all provisions of the "Basic Registry with mapping" profile of ISO/IEC 11179-3:2023, 4.4.2;

— **Concept System and Binary Relations Registry with mapping**: Implements Clauses 7 and 8 in addition to all provisions of the "Basic Registry with mapping" profile of ISO/IEC 11179-3:2023, 4.4.2.

5.4.2 Conformance labels

Conformance to the profiles specified in 5.4.1 may be claimed using the following labels, respectively:

— ISO/IEC 11179-32:2023 Concept System Registry;

— ISO/IEC 11179-32:2023 Concept System and Binary Relations Registry;

— ISO/IEC 11179-32:2023 Concept System Registry with mapping;

— ISO/IEC 11179-32:2023 Concept System Registry and Binary Relations with mapping.

5.5 Implementation conformance statement (ICS)

An implementation claiming conformance to this document shall include an Implementation Conformance Statement stating:

a) whether it conforms or strictly conforms;

b) which clauses are or are not supported;

c) what extensions, if any, are supported or used.

A standard profile may be referenced, if applicable.

EXAMPLE Product Z strictly conforms to ISO/IEC 11179-32:2023 Concept System Registry with Mapping.

5.6 Obligation

Properties and relationships specified in this document are one of: Mandatory, Conditional or Optional. The obligation is not explicitly stated but is to be inferred from the multiplicity of the property or relationship, and the presence or absence of a condition. In addition, a Registration Authority can specify additional constraints to be applied to particular **Administered_Items** (see ISO/IEC 11179-3:2023, 9.4.2), using **Constraint_Sets** (see ISO/IEC 11179-3:2023, 9.4.6). See 6.5.

For the purpose of conformance:

a) Mandatory properties and relationships shall exist and shall conform to the provisions of this document.

b) Anything specified as Conditional within this document shall be treated as Mandatory if the associated condition is satisfied and shall otherwise be not present.

c) Optional properties and relationships are not required to exist, but if they do exist, they shall conform to the provisions of this document.

Such obligation is enforced if and only if the Registration Status of the associated registry items is Recorded or higher (see ISO/IEC 11179-3:2023, 9.4.6.3 and ISO/IEC 11179-6:2023,[3] 4.4).

6 Relationship to ISO/IEC 11179-3

6.1 Metamodel for a metadata registry

A metamodel is a model that describes other models. A metamodel provides a mechanism for understanding the precise structure and components of the specified models, which are needed for the successful sharing of the models by users, software facilities or both.

ISO/IEC 11179-3 uses a metamodel to describe the information model of a metadata registry. The registry in turn will be used to describe and model other data, for example about enterprise, public administration or business applications. The registry metamodel is specified as a conceptual data model, i.e., one that describes how relevant information is structured in the natural world. In other words, it is how the human mind is accustomed to thinking of the information.

6.2 Specification of the metamodel

The conventions used in specifying the metamodel are described in ISO/IEC 11179-3:2023, 5.3. Many of the classes specified in this document inherit from **Item**, which is specified in ISO/IEC 11179-3:2023, 6.4.2.1. As **Items**, instances of these classes may be identified, registered, administered, named, defined and classified.

6.3 Use of UML Class diagrams and textual description

This document uses both text and UML class diagrams to describe the metamodel. Both are normative and are intended to be complementary. However, if a conflict exists between what is specified in UML and what is specified in text, the text takes precedence until a correction is made to make them consistent. Further, if a conflict exists between a formal definition and other normative text, the formal definition takes precedence until a correction is made to make them consistent.

A consolidated UML class hierarchy is included as Annex A.

While the model diagrams are presented in UML notation, this document does not assume nor endorse any specific system environment, database management system, database design paradigm, system development methodology, data definition language, command language, system interface, user interface, computing platform or any technology required for implementation.

6.4 Package dependencies

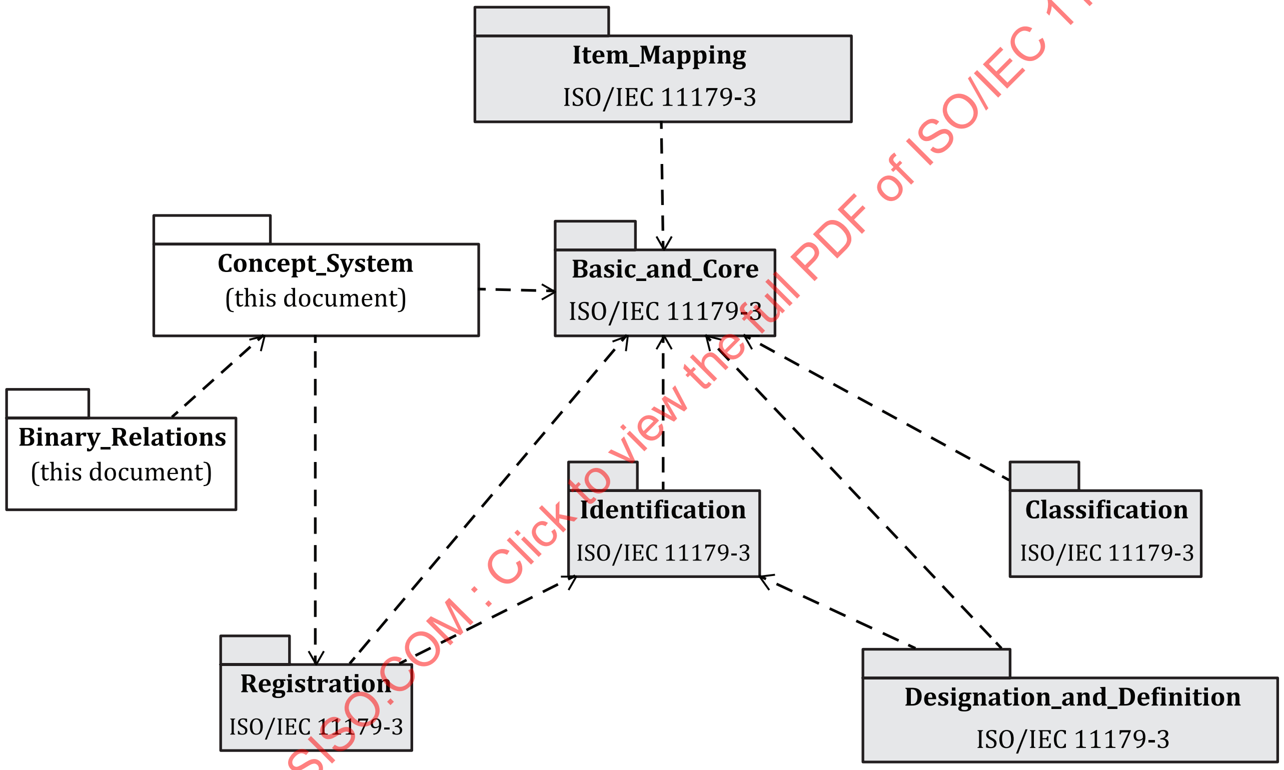

Figure 1 — Package dependencies

Figure 1 illustrates the dependencies among the packages. The **Concept_System** and **Binary_Relations** packages are specified in this document. All the other packages are specified in ISO/IEC 11179-3:2023.

The lines in the figure illustrate dependencies in the direction of the arrow. In order to implement a package that has dependencies, the packages on which it is dependent shall also be implemented. The dependencies are of three types:

a) Subclassing from classes in another package, e.g. **Relation** (7.2.2.3) in the **Concept_System** package is subclassed from the **Concept** class in the **Basic and Core** package (ISO/IEC 11179-3:2023, 6.4.2.2), and **Concept_Constraint_Set** (7.2.2.8) in the **Concept_System** package is subclassed from the **Constraint_Set** class in the **Registration** package (ISO/IEC 11179-3:2023, 9.4.6).

b) Relationship between classes, e.g. **Registered_Item** in the **Registration** package (ISO/IEC 11179-3:2023, 9.4.1) has a relationship with **Reference_Document** in the **Basic_and_Core** package (ISO/IEC 11179-3:2023, 6.3.8).

c) Some attributes use a predefined datatype or a class from another package as a datatype, e.g. the contact attribute of the **Stewardship_Record** class in the **Registration** package (ISO/IEC 11179-3:2023, 9.4.7) uses the **Contact** class of the **Basic_and_Core** package (ISO/IEC 11179-3:2023, 6.3.5) as a datatype.

Conformance options are specified in Clause 5 and standard conformance profiles in 5.3.

6.5 Subclassing the Constraint_Set class

This document extends the **Constraint_Set** class (ISO/IEC 11179-3:2023, 9.4.6) by specifying subclasses to support constraints specified in this document. See Figure 2.

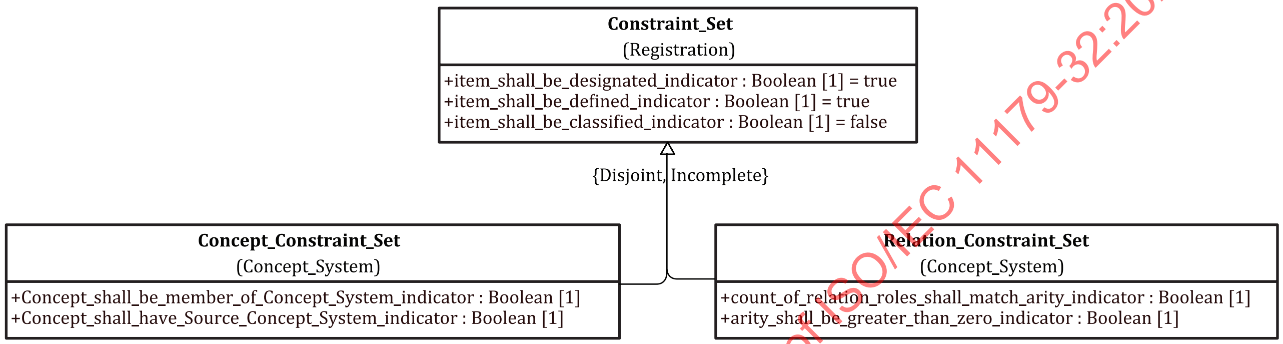
Constraint_Set
(Registration)
+item_shall_be_designated_indicator : Boolean [1] = true
+item_shall_be_defined_indicator : Boolean [1] = true
+item_shall_be_classified_indicator : Boolean [1] = false
{Disjoint, Incomplete}
Concept_Constraint_Set
(Concept_System)
+Concept_shall_be_member_of_Concept_System_indicator : Boolean [1]
+Concept_shall_have_Source_Concept_System_indicator : Boolean [1]
Relation_Constraint_Set
(Concept_System)
+count_of_relation_roles_shall_match_arity_indicator : Boolean [1]
+arity_shall_be_greater_than_zero_indicator : Boolean [1]

Figure 2 — Subclassing Constraint_Set

Concept_Constraint_Set is specified in 7.2.2.8.

Relation_Constraint_Set is specified in 7.2.2.9.

Any registry implementation shall provide a mechanism to enforce these constraints.

6.6 Relationship to Classification region in ISO/IEC 11179-3:2023

There are some overlaps in functionality between the **Concept_System** package specified in this document and the **Classification** package specified in ISO/IEC 11179-3:2023, Clause 10. Annex C provides a mapping between the two facilities and an explanation for the duplication.

7 Concept_System package

7.1 Overview of the Concept_System package

The **Concept_System** package consists of a single metamodel region, the Concept System metamodel region.

7.2 Concept System metamodel region

7.2.1 Overview

The Concept System metamodel region is illustrated in Figure 3. The purpose of this metamodel region is to describe concepts (abstract units of knowledge), represented by instances of **Concept** (7.2.2.1), and the various relations, represented by instances of **Relation** (7.2.2.3), which hold among the concepts. Support for more advanced concept systems, such as ontologies with formal semantics is provided through the use of assertions, represented by instances of **Assertion** (7.2.2.5). Annex B provides examples using SKOS, ORM, OWL and CLIF.

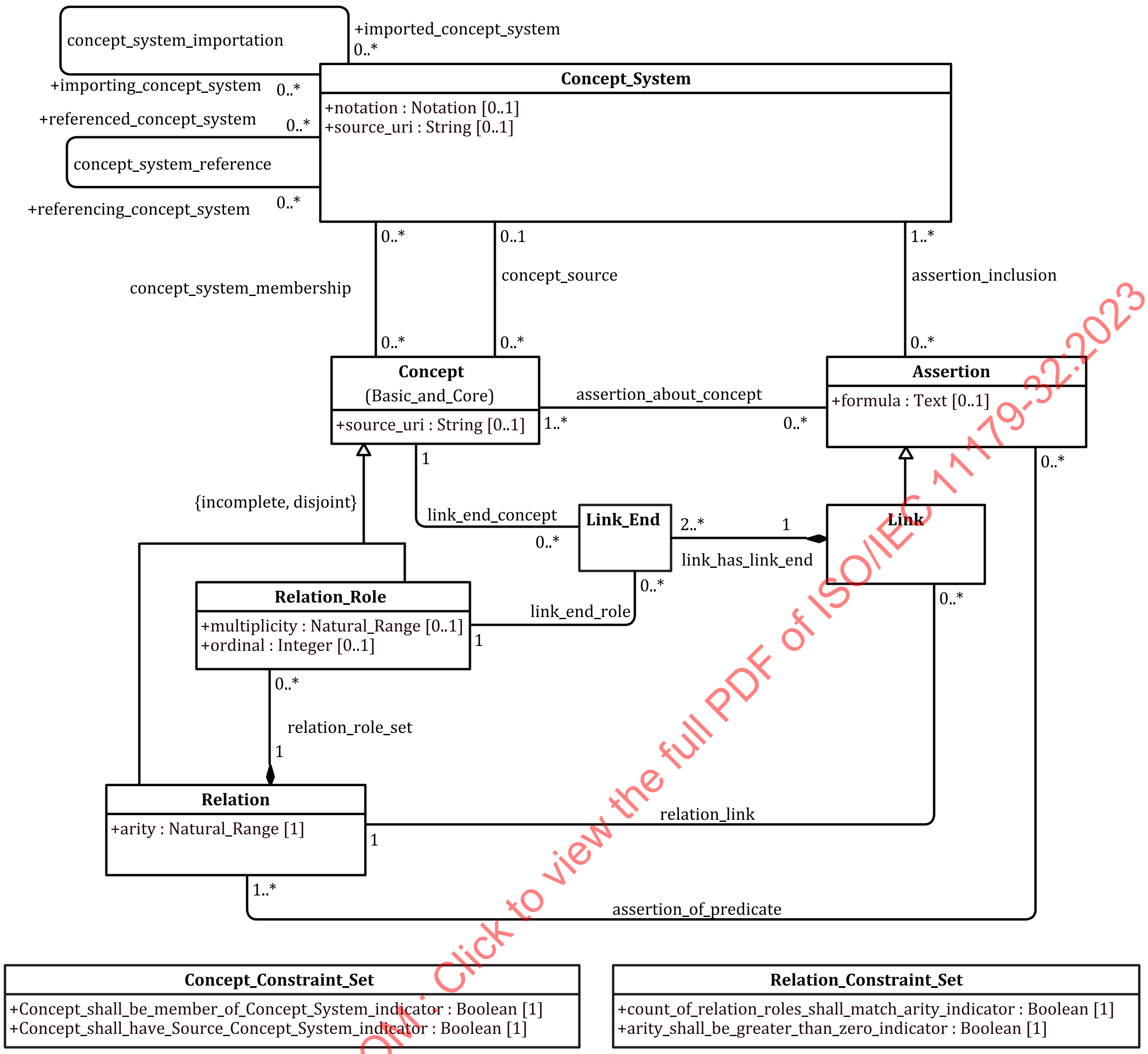
concept_system_importation
+imported_concept_system
0..*
+importing_concept_system
0..*
+referenced_concept_system
0..*
concept_system_reference
+referencing_concept_system
0..*
Concept_System
+notation : Notation [0..1]
+source_uri : String [0..1]
0..*
0..1
1..*
concept_system_membership
concept_source
assertion_inclusion
0..*
0..*
0..*
Concept
(Basic_and_Core)
+source_uri : String [0..1]
assertion_about_concept
1..*
0..*
Assertion
+formula : Text [0..1]
1
0..*
{incomplete, disjoint}
link_end_concept
0..*
Link_End
2..*
1
link_has_link_end
Link
0..*
0..*
Relation_Role
+multiplicity : Natural_Range [0..1]
+ordinal : Integer [0..1]
link_end_role
1
0..*
relation_role_set
1
Relation
+arity : Natural_Range [1]
relation_link
1
1..*
assertion_of_predicate
Concept_Constraint_Set
+Concept_shall_be_member_of_Concept_System_indicator : Boolean [1]
+Concept_shall_have_Source_Concept_System_indicator : Boolean [1]
Relation_Constraint_Set
+count_of_relation_roles_shall_match_arity_indicator : Boolean [1]
+arity_shall_be_greater_than_zero_indicator : Boolean [1]

Figure 3 — Concept system metamodel region

7.2.2 Classes in the Concept System metamodel region

7.2.2.1 Concept class

7.2.2.1.1 Direct superclass

Concept is a subclass of **Item** (ISO/IEC 11179-3:2023, 6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

7.2.2.1.2 Description of Concept

Concept is part of the Core metamodel and is specified in ISO/IEC 11179-3:2023, 6.4.2.2. Additional associations are specified in this metamodel region.

Concept is a class, each instance of which models a concept, a unit of knowledge created by a unique combination of characteristics. A concept is independent of representation.

Concept is a superclass of both **Relation** (7.2.2.3) and **Relation_Role** (7.2.2.4). These specializations are disjoint, and the specialization hierarchy is incomplete.

7.2.2.1.3 Associations of Concept

As a subclass of **Item**, **Concept** inherits **Item**'s associations (ISO/IEC 11179-3:2023, 6.4.2.1.2). This metamodel region specifies the following associations:

— **concept_system_membership** (7.2.3.1);

— **concept_source** (7.2.3.2);

— **assertion_about_concept** (7.2.3.7);

— **link_end_concept** (7.2.3.11).

7.2.2.1.4 Attributes of Concept

The attributes of **Concept** are specified in ISO/IEC 11179-3:2023, 6.4.2.2.4.

7.2.2.2 Concept_System class

7.2.2.2.1 Direct superclass

Concept_System is a subclass of **Item** (ISO/IEC 11179-3:2023, 6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

7.2.2.2.2 Description of Concept_System

Concept_System is a class, each instance of which models a concept system, a set of concepts structured according to the relations among them.

A minimal concept system can simply be a collection of concepts. A more elaborate concept system could be a collection of concepts organized into a taxonomy or meronomy specified by means of various relations (e.g. semantic relations) among the concepts.

NOTE 1 Examples of concept systems are included in Annex B.

NOTE 2 If a concept system is available in multiple notations, it is good practice to register the concept system only once and to use reference documents to record the various notations.

7.2.2.2.3 Associations of *Concept_System*

A **Concept_System** has the following associations:

— **concept_system_membership** (7.2.3.1);

— **concept_source** (7.2.3.2);

— **concept_system_reference** (7.2.3.3);

— **concept_system_importation** (7.2.3.4);

— **assertion_inclusion** (7.2.3.6).

7.2.2.2.4 Attributes of Concept_System

See Table 1.

Table 1 — Attributes of Concept_System

Attribute name	Multiplicity	Datatype	Description
notation	0..1	**Notation** (ISO/IEC 11179-3:2023, 6.2.7)	Definition: notation used in specification of this concept system. Comment: Examples of such notations include XCL Common Logic (ISO/IEC 24707) or OWL-DL XML notation (Ontology Web Language from W3C).
source_uri	0..1	**String** (ISO/IEC 11179-3:2023, 6.2.11)	Definition: URI that enables access to the concept system Provides access to details of the concept system that are external to the registry.

7.2.2.3 Relation class

7.2.2.3.1 Direct superclass

Relation is a subclass of **Concept** (specified in ISO/IEC 11179-3:2023, 6.4.2.2). **Concept** in turn is a subclass of **Item** (specified in ISO/IEC 11179-3:2023, 6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

7.2.2.3.2 Description of Relation

Relation is a class, each instance of which models a relation, a sense in which concepts may be connected via constituent relation roles.

Relation is a superclass of the **Binary_Relation** class which is described in 8.2.2.2.

Any relation is also a concept in its own right (and hence **Relation** is a subclass of **Concept**). A unary relation is just a concept and should be registered as such. Only relations with arity 2 or greater can be registered as **Relations**.

A reflexive relation (i.e. one which refers to itself) needs to be modelled using an **Assertion**.

NOTE 1 A **Relation** is a subset of the powerset of RxUD, for some role set R, where UD is the universe of discourse.

NOTE 2 An *n*-ary **Relation** on sets $A_1, ..., A_n$ is a set of ordered *n*-tuples $<a_1, ..., a_n>$ where a_i is an element of A_i for all *i*, *i between 1 and n*. Thus, an *n*-ary **Relation** on sets $A_1, ..., A_n$ is a subset of Cartesian product A_1 x ... x A_n. Membership of an n-tuple in the **Relation** is specified through **Assertions**, the simplest form of which is a **Link** representing exactly one tuple of the **Relation**. In this metamodel, **Relations** are defined over sets of **Concepts**.

NOTE 3 In this metamodel, unordered n-tuples are used with named **Relation_Roles** rather than positional elements of the n-tuple. The ordering can optionally be specified using the **ordinal** attribute on **Relation_Role**.

7.2.2.3.3 Associations of Relation

As a subclass of **Concept**, **Relation** inherits **Concept**'s associations. **Relation** has the following additional associations:

— **relation_role_set** (7.2.3.5);

— **relation_link** (7.2.3.9);

— **assertion_of_predicate** (7.2.3.8).

7.2.2.3.4 Attributes of Relation

See Table 2.

Table 2 — Attributes of Relation

Attribute name	Multiplicity	Datatype	Description
arity	1..1	**Natural_Range** (ISO/IEC 11179-3:2023, 6.2.6)	Definition: number of elements in the relation Example: A binary relation has an arity of 2. Constraint: **arity** should be >= 2.

7.2.2.4 Relation_Role class

7.2.2.4.1 Direct superclass

Relation_Role is a subclass of **Concept** (ISO/IEC 11179-3:2023, 6.4.2.2; see 7.2.2.1). **Concept** in turn is a subclass of **Item** (specified in ISO/IEC 11179-3:2023, 6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

7.2.2.4.2 Description of Relation_Role

Relation_Role is a class, each instance of which models a relation role. A relation role is an argument (element) of a relation.

NOTE 1 In relational database terms, the relation role represents a column in a relational table (for an asymmetric relation).

Relation roles permit position independent naming of the arguments of a relation.

NOTE 2 This is similar to the distinction between positional and named arguments to procedures in programming languages.

For symmetric **Binary_Relations** (8.2.2.2), relation roles are reused to indicate multiple arguments (**Link_Ends** (7.2.2.7)), since the arguments (link ends) are to be treated identically.

7.2.2.4.3 Associations of Relation_Role

As a subclass of **Concept**, **Relation_Role** inherits **Concept**'s associations. **Relation_Role** has the following additional associations:

— **relation_role_set** (7.2.3.5);

— **link_end_role** (7.2.3.12).

7.2.2.4.4 Attributes of Relation_Role

See Table 3.

Table 3 — Attributes of Relation_Role

Attribute name	Multiplicity	Datatype	Description
multiplicity	0..1	**Natural_Range** (ISO/IEC 11179-3:2023, 6.2.6)	Definition: number of links which have to (logically) be members of the source relation of this relation role, differing only by a link end linked to this relation role. EXAMPLE: If a relation *purchase* with an arity of 3 has relation roles *buyer*, *seller* and *item*, then a multiplicity of 0..1 on the *buyer* role means that it is not permitted for more than one member of the *purchase* relation to involve both the same *seller* and the same *item* (differing only in the *buyer*).

Table 3 *(continued)*

Attribute name	Multiplicity	Datatype	Description
ordinal	0..1	**Integer** (ISO/ IEC 11179-3:2023, 6.2.5)	Definition: order of the relation role among other relation roles in the relation. Comment: the ordinal allows the ordering of the concepts, represented in the relation by the relation roles, to be specified. This can be necessary if the ordering of the concepts changes the meaning of the relation.

7.2.2.4.5 Constraint on *Relation_Role*

If an instance of the **Relation** (7.2.2.3) class is deleted, then the linked instances of the **Relation_Role** class shall also be deleted.

7.2.2.5 Assertion class

7.2.2.5.1 Direct superclass

Assertion is a subclass of **Item** (ISO/IEC 11179-3:2023, 6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

7.2.2.5.2 Description of Assertion

Assertion is a class, each instance of which models an assertion, a sentence or proposition in logic which is asserted (or assumed) to be true. **Assertion** is also the superclass of **Link** (7.2.2.6).

7.2.2.5.3 Associations of Assertion

As a subclass of **Item, Assertion** inherits **Item**'s associations (ISO/IEC 11179-3:2023, 6.4.2.1.2). **Assertion** has the following additional associations:

— **assertion_inclusion** (7.2.3.6);

— **assertion_about_concept** (7.2.3.7);

— **assertion_of_predicate** (7.2.3.8).

7.2.2.5.4 Attributes of Assertion

See Table 4.

Table 4 — Attributes of Assertion

Attribute name	Multiplicity	Datatype	Description
formula	0..1	**Text** (ISO/IEC 11179-3:2023, 6.2.12)	Optional. Definition: text which expresses this assertion.

7.2.2.5.5 Constraint on Assertions

If **formula** is used, all **Assertions** within a **Concept_System** shall use the same notation for the formulas.

7.2.2.6 Link class

7.2.2.6.1 Direct superclass

Link is a subclass of **Assertion** (7.2.2.5). **Assertion** in turn is a subclass of **Item** (specified in ISO/IEC 11179-3:2023, 6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

7.2.2.6.2 Description of Link

Link is a class each instance of which models a link. A link is a member (not an instance) of a relation. In relational database parlance, a link would be a tuple (row) in a relation (table). **Link** is a subclass of **Assertion** (7.2.2.5), and as such is included in one or more **Concept_Systems** (7.2.2.2) through the **assertion_inclusion** (7.2.3.6) association.

7.2.2.6.3 Associations of Link

As a subclass of **Assertion**, **Link** inherits **Assertion**'s associations (7.2.2.5.3). **Link** has the following additional associations:

— **relation_link** (7.2.3.9);

— **link_has_link_end** (7.2.3.10).

7.2.2.6.4 Attributes of Link

No attributes are specified in this metamodel region.

7.2.2.6.5 Constraint on Link class

The number of **Link_Ends** (7.2.2.7) associated with a **Link** shall match the **arity** (7.2.2.3.4) of the **Relation** (7.2.2.3) of which the **Link** is a member, if the **arity** is specified.

7.2.2.7 Link_End class

7.2.2.7.1 Direct superclass

Link_End is a subclass of **Item** (ISO/IEC 11179-3:2023, 6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

7.2.2.7.2 Description of Link_End

Link_End is a class, each instance of which models a link end. A link end identifies the relation role played by a concept in the associated link.

The **Link_End** class models the association among **Links** (7.2.2.6), **Concepts** (7.2.2.1) (ends) and **Relation_Roles** (7.2.2.4). This is used to represent the relationship between an n-tuple (row) of a relation and the values for the fields (arguments) of the n-tuple. Hence, a **Link_End** is used to model the instantiation of a **Relation_Role** for a particular **Link** (tuple, row) of a **Relation** (7.2.2.3).

7.2.2.7.3 Associations of Link_End

As a subclass of **Item**, **Link_End** inherits **Item**'s associations (ISO/IEC 11179-3:2023, 6.4.2.1.2). **Link_End** has the following additional associations:

— **link_has_link_end** (7.2.3.10);

— **link_end_concept** (7.2.3.11);

— **link_end_role** (7.2.3.12).

7.2.2.7.4 Attributes of Link_End

No attributes are specified in this metamodel region.

7.2.2.7.5 Constraints on Link_End class

7.2.2.7.5.1 Constraint 1

It shall be the case that the **Relation_Role** (7.2.2.4) specified in the **link_end_role** association (7.2.3.12) shall correspond to a **Relation_Role** which is a role [via the **relation_role_set** association (7.2.3.5)] of the **Relation** (7.2.2.3) of the **Link** (7.2.2.6) (via the **relation_link** association (7.2.3.9)) to which the **Link_End** is associated via the **link_has_link_end** association (7.2.3.10).

7.2.2.7.5.2 Constraint 2

If an instance of the **Link** (7.2.2.6) class is deleted, then the linked instances of the **Link_End** class shall also be deleted.

7.2.2.8 Concept_Constraint_Set class

7.2.2.8.1 Direct superclass

Concept_Constraint_Set is a subclass of **Constraint_Set** (ISO/IEC 11179-3:2023, 9.4.6). **Constraint_Set** in turn is a subclass of **Item** (specified in ISO/IEC 11179-3:2023, 6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

7.2.2.8.2 Description of Concept_Constraint_Set

Concept_Constraint_Set extends **Constraint_Set** with constraints applicable to the use of the **Concept** class in the **Concept_System** package.

7.2.2.8.3 Associations of Concept_Constraint_Set

As a subclass of **Constraint_Set**, **Concept_Constraint_Set** inherits **Constraint_Set**'s associations.

7.2.2.8.4 Attributes of Concept_Constraint_Set

See Table 5.

Table 5 — Attributes of Concept_Constraint_Set

Attribute name	Multiplicity	Datatype	Description
Concept_shall_be_member_of_Concept_System_indicator	1	**Boolean** (ISO/IEC 11179-3:2023, 6.2.2)	Definition: indicator as to whether a concept is required to be a member of a concept system; that is, whether an instance of the **Concept** class is required to be linked to an instance of the **Concept_System**, class via an instance of the **concept_system_membership** association (7.2.3.1). NOTE Best practice is to set this to TRUE.

Table 5 *(continued)*

Attribute name	Multiplicity	Datatype	Description
Concept_shall_have_source_Concept_System_indicator	1	**Boolean** (ISO/IEC 11179-3:2023, 6.2.2)	Definition: indicator as to whether a concept is required to have a source concept system; that is, whether an instance of the **Concept** class is required to be linked to an instance of the **Concept_System**, class via an instance of the **concept_source** association (7.2.3.2). NOTE Best practice is to set this to TRUE.

7.2.2.9 Relation_Constraint_Set class

7.2.2.9.1 Direct superclass

Relation_Constraint_Set is a subclass of **Constraint_Set** (ISO/IEC 11179-3:2023, 9.4.6). **Constraint_Set** in turn is a subclass of **Item** (specified in ISO/IEC 11179-3:2023, 6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

7.2.2.9.2 Description of Relation_Constraint_Set

Relation_Constraint_Set extends **Constraint_Set** with constraints applicable to the use of the **Relation** class in the ***Concept_System*** package.

7.2.2.9.3 Associations of Relation_Constraint_Set

As a subclass of **Constraint_Set**, **Relation_Constraint_Set** inherits **Constraint_Set**'s associations.

7.2.2.9.4 Attributes of Relation_Constraint_Set

See Table 6.

Table 6 — Attributes of Relation_Constraint_Set

Attribute name	Multiplicity	Datatype	Description
count_of_Relation_Roles_shall_match_arity_indicator	1	**Boolean** (ISO/IEC 11179-3:2023, 6.2.2)	Definition: indicator as to whether the count of relation roles of a relation is required to match the arity of that relation; that is, whether the count of instances of the **Relation_Role** (7.2.2.4) class linked to an instance of the **Relation** (7.2.2.3) class via the **relation_role_set** (7.2.3.5) association is required to match the value of the **arity** attribute of that instance of the **Relation** class. NOTE Best practice is to set this to TRUE.
arity_shall_be_greater_than_zero_indicator	1	**Boolean** (ISO/IEC 11179-3:2023, 6.2.2)	Definition: indicator as to whether the arity of a relation is required to be greater than zero; that is, whether the value of the **arity** attributes of the instances of the **Relation** (7.2.2.3) class are required to be greater than zero. NOTE Best practice is to set this to TRUE.

7.2.3 Associations of the Concept System metamodel region

7.2.3.1 concept_system_membership association

The **concept_system_membership** association records the binding of zero, one or more instances of the **Concept** class (7.2.2.1) to zero, one or more instances of the **Concept_System** class (7.2.2.2).

This association registers the inclusion of a concept within a concept system.

Because the **Concept** class is specified in the Core metamodel (ISO/IEC 11179-3:2023, 6.4.2.2), a concept is not required to be a member of a concept system. However, if the Concept System metamodel region is supported, it is best practice for each concept to be a member of one or more concept systems. A registration authority may enforce this by setting the **Concept_shall_be_member_of_Concept_System_indicator** attribute (7.2.2.8.4) of the **Concept_Constraint_Set** class (7.2.2.8) to TRUE. The registry may be specified as a concept system if no more appropriate concept system is defined.

7.2.3.2 concept_source association

The **concept_source** association records the binding of zero or one instances of the **Concept_System** class (7.2.2.2) to zero, one or more instances of the **Concept** class (7.2.2.1).

This association registers the concept system that is the source of a set of concepts.

Because the **Concept** class is specified in the Core metamodel (ISO/IEC 11179-3:2023, 6.4.2.2), a concept is not required to be sourced from a concept system. However, if the Concept System metamodel region is supported, it is best practice for each concept to be sourced from exactly one concept system. A registration authority may enforce this by setting the indicator **Concept_shall_have_ source_Concept_System_indicator** (7.2.2.8.4) of the **Concept_Constraint_Set** class (7.2.2.8) to TRUE.

The **concept_system_membership** association and the **concept_source** association are not mutually exclusive. Thus, a concept may be specified as a member of the concept system that is its source.

The source concept system establishes an explicit minimum scope within which the identity of the concept can be taken to have been determined, in the sense that there should be no other concept within that same scope which represents the same meaning. In some registries (including implementations based on ISO/IEC 11179-3:2003 or ISO/IEC 11179-3:2013), this scope can always be the registry concept system, thus making explicit the expectation that there shall always be at most one concept within that entire registry representing any given meaning. In other registries, the scope can generally be much narrower, reflecting a lack of determination having (necessarily) been made as to whether the same meaning may or may not also be represented by one or more other concepts in the registry, from a different source(s).

The source concept system also provides a scoping mechanism for assertions pertaining to that concept (generally including many assertions that the concept does not participate in directly). In some registries in which all concepts have a distinguished registry concept system as their source, it is possible that all assertions also be included in that same registry concept system, thus indicating that all assertions pertaining to any concept are valid across the whole registry context.

NOTE Within the ontology community, this is called a “uniform ontological commitment”.

In other registries, it is possible that all concepts have the registry as their source, but discrimination can be made between assertions included in that registry concept system, and other assertions which are registered can be valid only in (a) narrower context(s). In yet other registries, it is possible there are many different concept instances representing either similar or even arguably identical meanings, but with some potentially critical difference(s) in semantics represented by the distinct sets of assertions included in their respective source concept systems.

7.2.3.3 concept_system_reference association

The **concept_system_reference** association specifies the binding of zero, one or more instances of the **Concept_System** (7.2.2.2) class by zero, one or more other instances of the **Concept_System** class.

This association registers where one or more concept systems (each being a referenced concept system) is referenced by one or more other concept systems (each being a referencing concept system).

A referenced concept system is not considered to be part of the referencing concept system.

Navigation from a referencing concept system to a referenced concept system shall be supported by a conforming or strictly conforming implementation. Navigation from the referenced concept system to the referencing concept system is permitted but not required.

7.2.3.4 concept_system_importation association

The **concept_system_importation** association records the binding of zero, one or more instances of the **Concept_System** class (7.2.2.2) to zero, one or more other instances of the **Concept_System** class.

This association registers where one or more concept systems (each being an imported concept system) is imported by one or more other concept systems (each being an importing concept system).

Such importation specifies that all concepts and assertions, if applicable, included in the imported concept system are also to be included in the importing concept system.

An imported concept system is considered to be an integral part of the importing concept system.

Navigation from the importing concept system to the imported concept system shall be supported by a conforming or strictly conforming implementation. Navigation from the imported concept system to the importing concept system is permitted but not required.

7.2.3.5 relation_role_set association

The **relation_role_set** association records the binding of zero, one or more instances of the **Relation_Role** class (7.2.2.4) to exactly one instance of the **Relation** class (7.2.2.3).

This association registers the set of relation roles that participate in a relation.

It is best practice that the number of relation roles participating in *a* relation matches the arity of the relation, as represented by the value assigned to the **arity** attribute (7.2.2.3.4) of the relevant instance of the **Relation** class. However, the multiplicity of the association at the **Relation_Role** end of the association is “0..*”, allowing for the situation where the relation roles are not yet known.

Hence, if an instance of the **Relation** class is deleted, all of the linked instances of the **Relation_Role** class are also to be deleted.

7.2.3.6 assertion_inclusion association

The **assertion_inclusion** association records the binding of zero, one or more instances of the **Assertion** class (7.2.2.5) to one or more instances of the **Concept_System** class (7.2.2.2).

This association registers the inclusion of an assertion in one or more concept systems. Each concept system may include one or more assertions, but not every concept system has to include assertions.

NOTE Since the **Link** class (7.2.2.6) is a subclass of the **Assertion** class, links are also included in one or more concept systems.

7.2.3.7 assertion_about_concept association

The **assertion_about_concept** association records the binding of zero, one or more instances of the **Assertion** class (7.2.2.5) to one or more instances of the **Concept** class (7.2.2.1).

This association registers the use of concepts within assertions. Each assertion shall use one or more concepts within their specification. Each concept may be used within one or more assertions, but not every concept has to be used within assertions.

7.2.3.8 assertion_of_predicate association

The **assertion_of_predicate** association records the binding of zero, one or more instances of the **Assertion** class (7.2.2.5) to one or more instances of the **Relation** class (7.2.2.3).

This association registers the use of relations as predicates in assertions. Each assertion shall use one or more relations as predicates. Each relation may be used as a predicate in one or more assertions, but not every relation has to be used as a predicate within assertions.

7.2.3.9 relation_link association

The **relation_link** association records the binding of zero, one or more instances of the **Link** class (7.2.2.6) to exactly one instance of the **Relation** class (7.2.2.3).

This association registers the membership of links in assertions. Each link shall be a member of exactly one relation. Each relation may have as members one or more links, but not every relation has to have a link as a member.

7.2.3.10 link_has_link_end association

The **link_has_link_end** association records the binding of exactly one instance of the **Link** class (7.2.2.6) to two or more instances of the **Link_End** class (7.2.2.7).

This association registers the link ends specified for a link. Each link shall include two or more link ends. Each link end shall be part of exactly one link.

The number of link ends included within a link depends on the arity of the relation that is linked to the link.

7.2.3.11 link_end_concept association

The **link_end_concept** association records the binding of zero, one or more instances of the **Link_End** class (7.2.2.7) to exactly one instance of the **Concept** class (7.2.2.1).

This association registers the concept that plays a role for a set of link ends. The specific role is identified through an instance of the **link_end_role** association (7.2.3.12).

Each link end shall have exactly one concept that plays a role. Each concept may play a role in one or more link ends, but not every concept has to play a role in link ends.

7.2.3.12 link_end_role association

The **link_end_role** association records the binding of zero, one or more instances of the **Link_End** class (7.2.2.7) to exactly one instance of the **Relation_Role** class (7.2.2.4).

This association registers the relation role that is the end role for a set of link ends. Each link end shall have a role played by exactly one relation role. Each relation role may be the role for one or more link ends, but not every relation role has to be a role for link ends.

8 Binary_Relations package

8.1 Overview of Binary_Relations package

The **Binary_Relations** package consists of a single metamodel region, the Binary Relations metamodel region.

8.2 Binary Relations metamodel region

8.2.1 Overview

The Binary Relations metamodel region supports characteristics particular to binary relations, as illustrated in Figure 4.

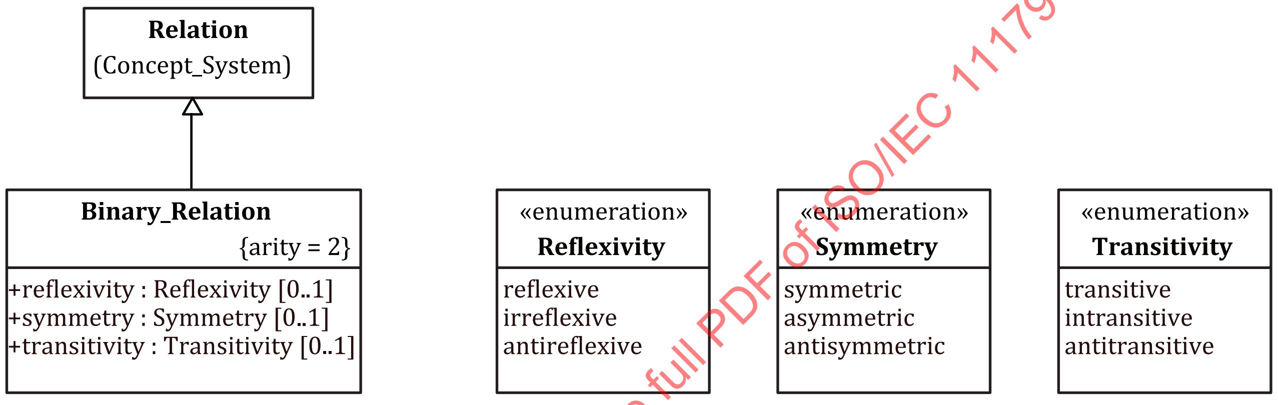

Figure 4 — Binary Relations metamodel region

8.2.2 Classes in the Binary Relations metamodel region

8.2.2.1 Relation class

The **Relation** class is described in 7.2.2.3.

8.2.2.2 Binary_Relation class

8.2.2.2.1 Direct superclass

Binary_Relation is a subclass of **Relation** (7.2.2.3). **Relation** in turn is ultimately a subclass of **Item** (specified in ISO/IEC 11179-3:2023, 6.4.2.1), allowing instances to be identified, registered, administered, named, defined and classified.

8.2.2.2.2 Description of Binary_Relation

Binary_Relation is a class each instance of which models a binary relation, a relation of arity 2 (i.e. having two link ends).

Most common semantic relations are binary, e.g. equals, less than, greater than, is-a, part-of, etc. An example of a relation which is not binary is betweenness. Binary relations are commonly represented as edges (or directed edges for asymmetric binary relations) in graphs, cf. the Resource Description Framework (RDF[12]) of the W3C[15].

Table 7 shows examples of some binary relationships and their characterization.

Table 7 — Examples of binary relations and their characterization

Relation	Symmetry	Reflexivity	Transitivity
equals	symmetric	reflexive	transitive
not equals	symmetric	antireflexive	intransitive
less than	antisymmetric	antireflexive	transitive
less than or equal	asymmetric	reflexive	transitive
similar	symmetric	reflexive	intransitive

The **Binary_Relation** class has three enumeration attributes: reflexivity, symmetry and transitivity.

A symmetric **Binary_Relation** may use the same **Relation_Role** (7.2.2.4) for both **Link_Ends** (7.2.2.7).

8.2.2.2.3 Attributes of Binary_Relation

See Table 8.

Table 8 — Attributes of Binary_Relation

Attribute name	Multiplicity	Datatype	Description
reflexivity	0..1	**Reflexivity** (8.2.3.1)	Optional. Definition: characterization of the binary relation (3.8) as: reflexive, irreflexive or antireflexive.
symmetry	0..1	**Symmetry** (8.2.3.2)	Optional. Definition: characterization of the binary relation (3.8) as: symmetric, asymmetric or antisymmetric
transitivity	0..1	**Transitivity** (8.2.3.3)	Optional. Definition: characterization of the binary relation (3.8) as: transitive, intransitive or antitransitive

8.2.3 Datatypes in the Binary_Relation metamodel region

8.2.3.1 Reflexivity enumeration

Reflexivity is an enumeration with values listed in Table 9. **Reflexivity** is used as the datatype of the **reflexivity** attribute (8.2.2.2.3) of **Binary_Relation** (8.2.2.2).

Table 9 — Values of Reflexivity enumeration

Value	Description
reflexive	A binary relation, R, is reflexive if for all x, R(x,x) is true. Equality is an example of a reflexive relation.
irreflexive	A binary relation, R, is irreflexive if it is not reflexive. i.e. R(x,x) is not necessarily true for all x.
antireflexive	A binary relation, R, is antireflexive if for all x, R(x,x) is false. Inequality is an example of an antireflexive relation. An antireflexive relation is also irreflexive, but antireflexive is a more specific characterization.

8.2.3.2 Symmetry enumeration

Symmetry is an enumeration of the values shown in Table 10. **Symmetry** is used as the datatype of the **symmetry** attribute (8.2.2.2.3) of **Binary_Relation** (8.2.2.2).

Table 10 — Values of Symmetry enumeration

Value	Description
symmetric	A binary relation, R, is symmetric if for all x, y: R(x,y) implies R(y,x). Examples of symmetric relations are “equals”, “not equals”, “within-2-miles-of”, etc. Symmetry does not imply reflexivity. For example, the inequality relation is symmetric, but antireflexive.
asymmetric	A binary relation, R, is asymmetric if for all x,y: R(x,y) does not imply R(y,x). In terms of this metamodel, asymmetric *Relations* have two distinguishable (non-identical) roles, one for each end of each Link. Examples of asymmetric relations include: less than, likes, father of, etc.
antisymmetric	A binary relation, R, is anti-symmetric if for all x,y: R(x,y) implies not R(y,x). “Less than” is an example of an antisymmetric relation. An antisymmetric relation is also asymmetric, but antisymmetric is a more specific characterization. An asymmetric relation is not necessarily antisymmetric (consider less than or equals).

8.2.3.3 Transitivity enumeration

Transitivity is an enumeration of the values shown in Table 11. **Transitivity** is used as the datatype of the **transitivity** attribute (8.2.2.2.3) of **Binary_Relation** (8.2.2.2).

Table 11 — Values of Reflexivity enumeration

Value	Description
transitive	A binary relation, R, is transitive, if for all x,y,z: R(x,y) and R(y,z) implies R(x,z). Examples of transitive relations include equality, less than and less than or equals.
intransitive	A binary relation, R, is intransitive if it is not transitive i.e. R(x,y) and R(y,z) does not imply R(x,z).
antitransitive	A binary relation, R, is antitransitive if for all x,y,z: R(x,y) and R(y,z) implies not R(x,z). An antitransitive relation is also intransitive, but antitransitive is a more specific characterization.

Annex A
(informative)

Consolidated Class Hierarchy

Figure A.1 shows all classes specified in this document or referenced from ISO/IEC 11179-3, that participate in a class hierarchy. Classes that do not participate in a class hierarchy are not shown.

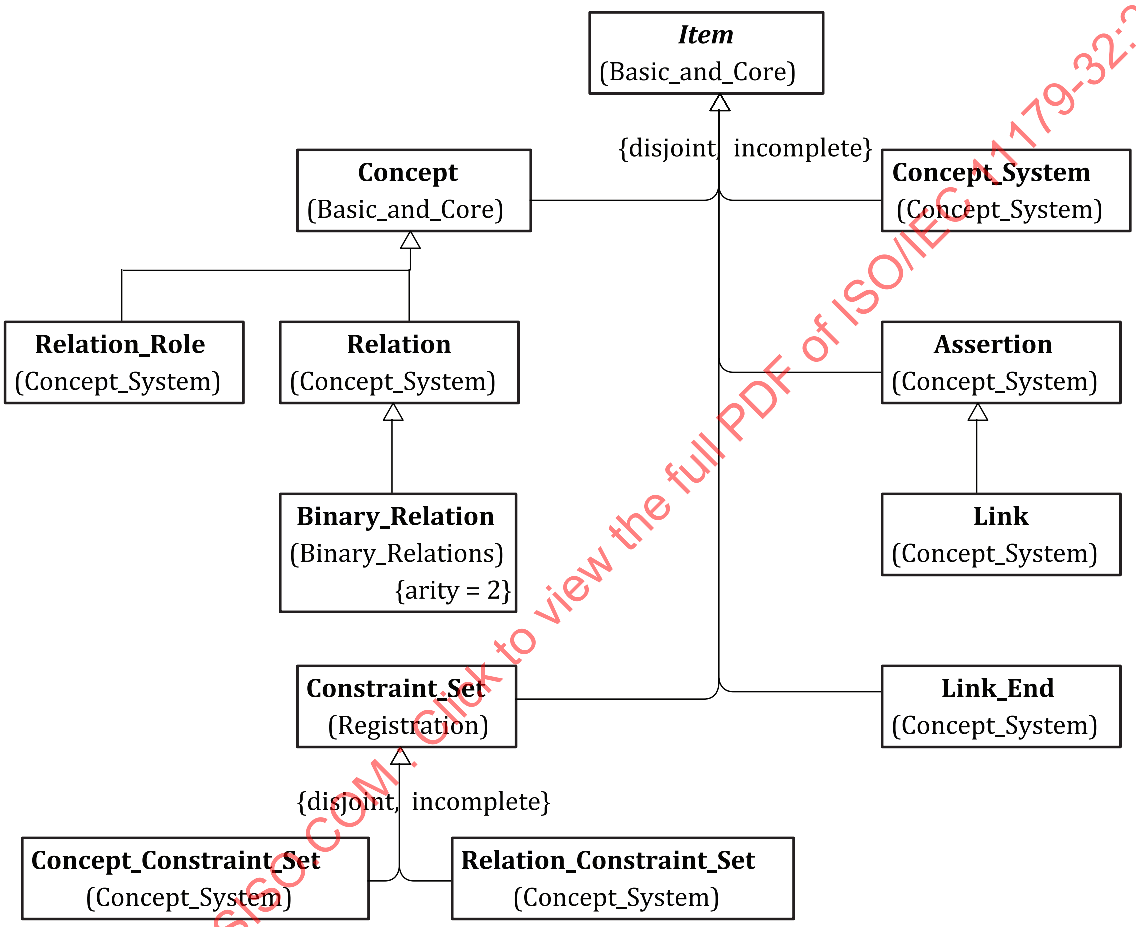

Figure A.1 — Consolidated Class Hierarchy

Standards or implementations which extend this document may extend the above class hierarchy.

Annex B
(informative)

Concept System Examples

B.1 Overview

This annex illustrates the use of the Concept System metamodel region (see 7.2).

B.2 Concept System Metamodels

The concept system metamodel specified in 7.2 is very generic, so that it may be used to register concept systems that are defined in a wide range of formalisms. Most formalisms will have some built-in constructs which are not built-in to the metamodel specified in 7.2. The concept system metamodel is generic enough to also support registration of such built-in constructs, in a "notation" concept system. For example, an OWL concept system can be used to define OWL ontological relations such as rdf:type, rdfs:range and owl:disjointWith.

By registering a full metamodel of each such formalism also as a concept system, the same facility can also be used to describe mappings between them, as well as to the concept system metamodel specified in 7.2. Table B.1 summarizes some suggested primary mappings between the ISO/IEC 11179-32 metamodel and a selection of notations. Note that in the RDF-based notations (SKOS and OWL) it is suggested to treat inverse properties as roles of an implicit binary relation. Some further details are given in this subclause.

Table B.1 — Correspondences of ISO/IEC 11179-32 concept system metamodel to selected notations

Notation	Concept	Relation	Relation_Role	Link	Assertion
SKOS	Concept	N/a	semantic relations	N/a	Statement
OWL	Class *or* Thing	N/a	ObjectProperty	N/a	Statement
UML	Class *or* Object	Association	AssociationEnd	Link	N/a
ORM	non-lexical object *or* non-lexical object type	idea type	role	idea	N/a
XTM	Topic	Association Type	Association Role	Association	N/a
SBVR	concept *or* characteristic	(non-unary) fact type	fact type role	N/a	fact
CLIF[9]	N/a	=	N/a	N/a	Statement

NOTE 1 N/a = Not applicable.

NOTE 2 **Relation** and **Relation_Role** are shown separately, even though they are subclasses of **Concept**.

— SKOS[13]

There is no metaclass in SKOS for semantic relations, instead there are only the foundational broader, narrower and related properties, and a semantic relation is one defined by those properties or by subproperties of those properties.

— OWL[11]

Some OWL built-in constructs are most naturally described as ternary relations, and others as variable arity relations with two roles. It is also reasonable to describe object properties as relations rather than relation roles. See OWL Example below.

— UML[5],[6],[7]

Because the ISO/IEC 11179-32 metamodel does not impose a meta-level hierarchy, the Generalization metaclass in UML can be interpreted as a built-in relation, and generalization relationships thus as links (in the ISO/IEC 11179-32 sense) between UML Classes.

— ORM

There is no official standard for ORM, but in this appendix the ORM metamodel provided in ISO/TR 9007:1987[2] is taken as normative.

— XTM[17]

An XML syntax for Topic Maps (ISO/IEC 13250).

— SBVR[10]

In the ISO/IEC 11179-32 metamodel, it is most natural to treat SBVR characteristics (unary fact types) as concepts rather than as relations, because links in this metamodel are required to have at least two ends.

B.3 SKOS Example

B.3.1 Overview

This is a very simple example using SKOS (Simple Knowledge Organization System)1).

B.3.2 SKOS Metamodel

The core of the SKOS (meta)model2) contains only two semantic relations3), defined by three RDF properties. Describing these two semantic relations in terms of the ISO/IEC 11179-32 metamodel is very straightforward, see Table B.2 and Table B.3.

Table B.2 — SKOS-CORE as an ISO/IEC 11179-32 Concept System

<Concept_System>			
	notation	**referencedConceptSystem**	**importedConceptSystem**
SKOS-CORE			

Table B.3 — SKOS relations as ISO/IEC 11179-32 Binary Relations

<Binary_Relation>					
	source	**role**	**reflexivity**	**symmetry**	**transitivity**
generalization	SKOS-CORE	skos:broader		antisymmetric	transitive
		skos:narrower			
association	SKOS-CORE	skos:related		symmetric	intransitive

B.3.3 SKOS example Thesaurus

Our SKOS example is a simple thesaurus of marital statuses. Here is its expression in Turtle4).

1) See http://www.w3.org/TR/skos-primer/

2) See http://www.w3.org/TR/skos-primer/#secsimple

3) See http://www.w3.org/TR/skos-primer/#secrel

4) See http://www.w3.org/TeamSubmission/2008/SUBM-turtle-20080114/

```
:MaritalStatus rdf:type skos:ConceptScheme .

:Married rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;
  skos:related :Single .

:Single rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;
  skos:narrower :NeverMarried .

:NeverMarried rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;

:Widowed rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;
  skos:broader :Single .

:Divorced rdf:type skos:Concept ;
  skos:inScheme :MaritalStatus ;
  skos:broader :Single .
```

Tables B.4, B.5 and B.6 provide a description of the above thesaurus in terms of the ISO/IEC 11179-32 metamodel.

Table B.4 — SKOS Thesaurus example — ISO/IEC 11179-32 Concept System

<Concept_System>			
	notation	**referencedConceptSystem**	**importedConceptSystem**
MaritalStatus	SKOS/Turtle	SKOS-CORE	

Table B.5 — SKOS Thesaurus Example — ISO/IEC 11179-32 Concepts

<Concept>	
	source
ms:Married	MaritalStatus
ms:Single	MaritalStatus
ms:NeverMarried	MaritalStatus
ms:Widowed	MaritalStatus
ms:Divorced	MaritalStatus

NOTE In Table B.5 and in the following tables, the concepts are represented by their designations.

Table B.6 — SKOS Thesaurus Example — ISO/IEC 11179-32 Links

<Link>			
source	**relation**	**link_end**	
		end_role	**end**
MaritalStatus	association	skos:related	ms:Married
		skos:related	ms:Single
MaritalStatus	generalization	skos:broader	ms:Single
		skos:narrower	ms:NeverMarried
MaritalStatus	generalization	skos:broader	ms:Single
		skos:narrower	ms:Widowed
MaritalStatus	generalization	skos:broader	ms:Single
		skos:narrower	ms:Divorced

B.3.4 Example Value Domain References

To illustrate one possible connection to data description, Table B.7 and Table B.8 show an example pair of enumerated value domains described with references to the above SKOS thesaurus.

NOTE Unused attributes have been omitted from these descriptions.

Table B.7 — SKOS Thesaurus Example — ISO/IEC 11179-31 Conceptual Domains

<Enumerated_Conceptual_Domain>	
	member
binary_ms_CD	ms:Single
	ms:Married
specific_ms_CD	ms:NeverMarried
	ms:Married
	ms:Widowed
	ms:Divorced

Table B.8 — SKOS Thesaurus Example — ISO/IEC 11179-31 Value Domains

<Enumerated_Value_Domain>				
	datatype	**meaning**	**member**	
			value	**meaning**
binary_marital_status	Bit	binary_ms_CD	0	ms:Single
			1	ms:Married
marital_status_code	Character	specific_ms_CD	S	ms:NeverMarried
			M	ms:Married
			W	ms:Widowed
			D	ms:Divorced

B.4 ORM Example

B.4.1 Overview

This example uses the Object Role Modelling[5)] (ORM) model from ISO/TR 9007:1987, Appendix E.

B.4.2 ORM Metamodel

There are two relations built-into ORM which need to be registered first. They are described in TR 9007 using the PASCAL syntax defined in ISO/TR 9007:1987, Appendix C. The subtype relation is defined by the declaration:

```
(R5) nolot-subtype = "NOLOT called"
                          (nolot-name-1 | nolot-name-1-list)
                    "is subtype-of NOLOT called" nolot-name-2 ";".
```

The other relation is implicitly defined within the idea-declaration definition:

```
(R7) idea-declaration = "IDEA (with-first ROLE (called" role-name-1
                                "and on NOLOT called" nolot-name-1 ")"
                        "and with-second ROLE (called" role-name-2
                                "and on NOLOT called" nolot-name-2 "))"
                        "is called" idea-name ";".
```

For clarity, first rewrite the idea-declaration rule as follows:

5) see http://www.orm.net/

```
(R7a) idea-declaration = "IDEA (with-first" idea-role-1
                         "and with-second" idea-role-2 ")"
                         "is called " idea-name ";".
(R7b) idea-role        = "ROLE (called" role-name
                         "and on NOLOT called" nolot-name ")".
(R7c) idea-role-1      = idea-role.
(R7d) idea-role-2      = idea-role.
```

Rule (R7a) corresponds to relation_role_set in the ISO/IEC 11179-3 metamodel; rule (R7b) defines a relation which is not built-in to the ISO/IEC 11179-3 metamodel. The ORM relations defined by rules (R5) and (R7b) thus can be described in terms of the ISO/IEC 11179-3 metamodel as in Table B.9, Table B.10 and Table B.11.

Table B.9 — ORM as an ISO/IEC 11179-32 Concept System

<Concept_System>			
	notation	**referencedConceptSystem**	**importedConceptSystem**
ORM	PASCAL		

Table B.10 — ORM Relations as ISO/IEC 11179-32 Binary Relations

<Binary_Relation>					
	source	**role**	**reflexivity**	**symmetry**	**transitivity**
subtype	ORM	subtype-of		antisymmetric	transitive
		supertype-of			
role-on	ORM	on			
		role			

Table B.11 — ORM Roles as ISO/IEC 11179-32 Relation Roles

<Relation_Role>		
	multiplicity	**ordinal**
subtype-of		1
supertype-of		2
on	1	1
role		2

B.4.3 Car Registration Model

The example model is depicted graphically in Figure B.1.

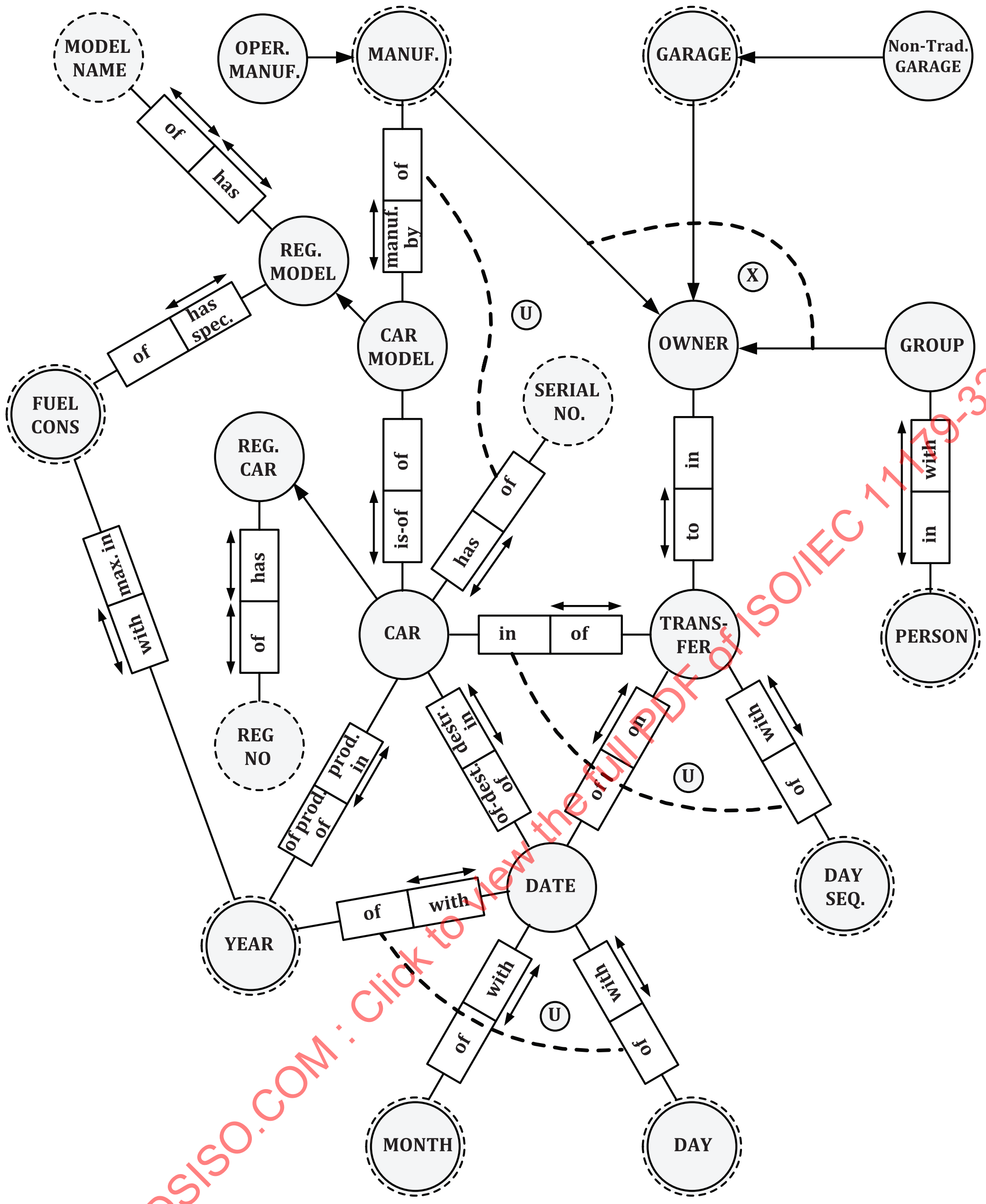

Figure B.1 — Car Registration Model in ORM

The textual definition of the Car Registration Model is reproduced below:

```
begin
add CONCEPTUAL-SCHEMA called 'CAR-REGISTRATION' ;
add NOLOT called ('MANUFACTURER' 'OPERATING-MANUFACTURER' 'REG-CAR'
                  'CAR' 'REG-MODEL' 'CAR-MODEL' 'FUEL-CONSUMPTION'
                  'DATE' 'YEAR' 'MONTH' 'DAY' 'TRANSFER' 'DAY-SEQ'
                  'OWNER' 'GARAGE' 'NON-TRADING-GARAGE' 'GROUP'
                  'PERSON');
add LOT called {'MANUFACTURER-NAME' 'REG-NO' 'SERIAL-NO' 'MODEL-NAME'
                'FUEL-CONSUMPTION-AMOUNT' 'YEAR-NO' 'MONTH-NO'
                'DAY-NO' 'SEQ-NO' 'GARAGE-NAME' 'PERSON-NAME');
add NOLOT called 'OPERATING-MANUFACTURER'
          is subtype-of NOLOT called 'MANUFACTURER';
add NOLOT called ('MANUFACTURER' 'GARAGE' 'GROUP')
          is subtype-of NOLOT called 'OWNER';
```

NOTE 1 Three other subtype declarations omitted here.

```
add IDEA (with-first ROLE (called 'manuf-by'
                           and on NOLOT called 'CAR-MODEL')
          and with-second ROLE (called 'of'
                           and on NOLOT called 'MANUFACTURER'))
          is called 'builds';
```

NOTE 2 Thirteen other idea declarations omitted here.

```
add BRIDGE (with-first ROLE (called 'called'
                             and on NOLOT called 'REG-CAR')
            and with-second ROLE (called 'of'
                             and on LOT called 'REG-NO'))
            is called 'registration';
```

NOTE 3 Two other explicit bridge declarations omitted here.

```
add BRIDGE (with-first ROLE (called 'called'
                             and on NOLOT called 'MANUFACTURER')
            and with-second ROLE (called 'of'
                             and on LOT called 'MANUFACTURER-NAME'))
            is called 'naming-of-model';
```

NOTE 4 Seven other implicit bridge declarations omitted here.

NOTE 5 The list of constraints is given on the next pages

```
end.
```

It has been chosen to regard only the non-lexical object types as concepts, and thus only the subtype declarations as links, and the idea types as relations. For the omitted subtype declarations, assume:

```
add NOLOT called 'NON-TRADING-GARAGE'
          is subtype-of NOLOT called 'GARAGE';
add NOLOT called 'CAR-MODEL'
          is subtype-of NOLOT called 'REG-MODEL';
add NOLOT called 'CAR'
          is subtype-of NOLOT called 'REG-CAR';
```

This document will not go through all of the omitted idea type declarations, but will assume:

```
add IDEA (with-first ROLE (called 'is-of'
                           and on NOLOT called 'CAR-MODEL')
          and with-second ROLE (called 'of'
                           and on NOLOT called 'CAR'))
          is called 'model';

add IDEA (with-first ROLE (called 'to'
                           and on NOLOT called 'OWNER')
          and with-second ROLE (called 'in'
                           and on NOLOT called 'TRANSFER'))
          is called 'transfer-owner';

add IDEA (with-first ROLE (called 'of'
                           and on NOLOT called 'CAR')
          and with-second ROLE (called 'in'
                           and on NOLOT called 'TRANSFER'))
          is called 'transfer-car';

add IDEA (with-first ROLE (called 'in'
                           and on NOLOT called 'GROUP')
          and with-second ROLE (called 'with'
                           and on NOLOT called 'PERSON'))
          is called 'group-member';
```

Table B.12, Table B.13, Table B.14 and Table B.15 express the model in terms of the 11179-32 Concept System metamodel.

Table B.12 — Car Registration Model in ORM — ISO/IEC 11179-32 Concept System

<Concept_System>			
	notation	referencedConceptSystem	importedConceptSystem
CAR-REGISTRATION	ISO9007-E3	ORM	

Table B.13 — Car Registration Model in ORM — ISO/IEC 11179-32 Concepts

<Concept>	
	source
MANUFACTURER	CAR-REGISTRATION
OPERATING-MANUFACTURER	CAR-REGISTRATION
REG-CAR	CAR-REGISTRATION
CAR	CAR-REGISTRATION
REG-MODEL	CAR-REGISTRATION
CAR-MODEL	CAR-REGISTRATION
FUEL-CONSUMPTION	CAR-REGISTRATION
DATE	CAR-REGISTRATION
YEAR	CAR-REGISTRATION
MONTH	CAR-REGISTRATION
DAY	CAR-REGISTRATION
TRANSFER	CAR-REGISTRATION
DAY-SEQ	CAR-REGISTRATION
OWNER	CAR-REGISTRATION
GARAGE	CAR-REGISTRATION
NON-TRADING-GARAGE	CAR-REGISTRATION
GROUP	CAR-REGISTRATION
PERSON	CAR-REGISTRATION

NOTE 6 In Table B.13, Table B.14 and Table B.15, the concepts are represented by their designations.

Table B.14 — Car Registration Model in ORM — ISO/IEC 11179-32 Binary Relations

<Binary_Relation>					
	source	role	reflexivity	symmetry	transitivity
builds	CAR-REGISTRATION	manuf-by			
		of			
model	CAR-REGISTRATION	is-of			
		of			
transfer-owner	CAR-REGISTRATION	to			
		in			
transfer-car	CAR-REGISTRATION	of			
		in			
group-member	CAR-REGISTRATION	in			
		with			

Table B.15 — Car Registration Model in ORM — ISO/IEC 11179-32 Links

<Link>			
source	relation	link_end	
		end_role	end
CAR-REGISTRATION	subtype	supertype-of	OPERATING_MANUFACTURER
		subtype-of	MANUFACTURER
CAR-REGISTRATION	subtype	supertype-of	MANUFACTURER
		subtype-of	OWNER
CAR-REGISTRATION	subtype	supertype-of	GARAGE
		subtype-of	OWNER
CAR-REGISTRATION	subtype	supertype-of	GROUP
		subtype-of	OWNER
CAR-REGISTRATION	subtype	supertype-of	NON-TRADING-GARAGE
		subtype-of	GARAGE
CAR-REGISTRATION	subtype	supertype-of	REG-CAR
		subtype-of	CAR
CAR-REGISTRATION	subtype	supertype-of	REG-MODEL
		subtype-of	CAR-MODEL
CAR-REGISTRATION	role-on	role	builds.manuf-by
		on	MANUFACTURER
CAR-REGISTRATION	role-on	role	builds.of
		on	CAR-MODEL
CAR-REGISTRATION	role-on	role	model.is-of
		on	CAR-MODEL
CAR-REGISTRATION	role-on	role	model.of
		on	CAR
CAR-REGISTRATION	role-on	role	transfer-owner.to
		on	OWNER
CAR-REGISTRATION	role-on	role	transfer-owner.in
		on	TRANSFER
CAR-REGISTRATION	role-on	role	transfer-car.of
		on	CAR
CAR-REGISTRATION	role-on	role	transfer-car.in
		on	TRANSFER
CAR-REGISTRATION	role-on	role	group-member.in
		on	GROUP
CAR-REGISTRATION	role-on	role	group-member.of
		on	PERSON

B.5 OWL Example

B.5.1 Overview

Because OWL (Web Ontology Language[6]) is founded upon the very simple binary predicate model of RDF, there is more than one reasonable way to map OWL into the 11179-3 Concept System metamodel. Perhaps the more obvious is to treat each ObjectProperty as a relation, each with relation roles rdf:

6) see http://www.w3.org/TR/owl-features/

subject and rdf:object. However, an analogy to Properties in UML and MOF will instead suggest treating each ObjectProperty as representing a relation role, of an underlying binary relation taken to be implicit in the OWL representation. Either approach is workable, and indeed it is possible to even mix the two approaches, treating some ObjectProperties as relations and others as relation roles, based on some case-by-case evaluation of the relative merits of each treatment.

B.5.2 OWL Metamodel

A convenient synopsis of OWL built-in constructs is provided in the OWL Web Ontology Language Overview. Some of the OWL built-in constructs correspond directly to elements of the 11179-3 Concept System metamodel. These are shown in Table B.16.

Table B.16 — OWL constructs with directly corresponding ISO/IEC 11179-32 metamodel elements

OWL Constructs	11179-3 metamodel description type
Ontology	Concept_System
imports	Importation
minCardinality, maxCardinality, cardinality, FunctionalProperty, InverseFunctionalProperty	multiplicity
TransitiveProperty	transitivity

OWL *inverseOf* assertions can also be captured using only built-in 11179-3 metamodel constructs, as will be shown below. (All *domain* assertions can also be described as *range* assertions on the opposing role, and therefore omit *domain* from our OWL metamodel as well.) And since assertions of membership in *SymmetricProperty* can also be regarded as simply an alternate syntax for expressing reflexive *inverseOf* assertions, they too may be captured in the same way.

Many other OWL built-in constructs do not have corresponding elements built-in to the 11179-3 Concept System metamodel, as summarized in Table B.17.

Table B.17 — OWL built-in constructs described in OWL metamodel

Group of OWL Constructs	11179-3 metamodel description type
metaclasses Class, Property, ObjectProperty and DatatypeProperty	Concept
classes Thing and Nothing	Concept
datatypes	Concept
equivalentClass, equivalentProperty, sameAs	Binary_Relation (symmetric, transitive)
differentFrom, complementOf, disjointWith	Binary_Relation (symmetric, intransitive)
subClassOf, subPropertyOf	Binary_Relation (asymmetric, transitive)
type, range	Binary_Relation (asymmetric, intransitive)
AllDifferent (and distinctMembers)	Relation (variable arity, 1 role)
intersectionOf, oneOf, unionOf	Relation (variable arity, 2 roles)
allValuesFrom, someValuesFrom, hasValue	Relation (arity=3, 3 roles)

Some of these constructs are actually reused from RDFS, which in turn is defined on top of RDF, and most of the OWL datatypes are taken from XML Schema. To describe this explicitly, four interrelated metamodels to support OWL are described: one each for RDF, RDFS, and the subset of XML Schema used in OWL; and one for OWL proper. The resulting description is shown in Table B.18 through Table B.23.

Table B.18 — OWL as an ISO/IEC 11179-32 Concept System

<Concept_System>			
	notation	**referencedConceptSystem**	**importedConceptSystem**
RDF			
RDFS			RDF
XSD		RDFS	
OWL		RDFS	XSD

Table B.19 — OWL Concepts as ISO/IEC 11179-32 Concepts

<Concept> (excluding Relations)	
	source
rdf:Property	RDF
rdfs:Resource	RDFS
rdfs:Class	RDFS
rdfs:Datatype	RDFS
rdfs:Literal	RDFS
xsd:string	XSD
xsd:decimal	XSD
xsd:integer	XSD
xsd:boolean	XSD
xsd:date	XSD
...other XSD datatypes...	
owl:Class	OWL
owl:Thing	OWL
owl:Nothing	OWL
owl:ObjectProperty	OWL
owl:DatatypeProperty	OWL

NOTE In Table B.19 through Table B.23, the concepts are represented by their designations.

Table B.20 — OWL Binary Relations as ISO/IEC 11179-32 Binary Relations

<Binary_Relation>					
	source	**role**	**reflexivity**	**symmetry**	**transitivity**
instance-type	RDF	instance		asymmetric	intransitive
		rdf:type			
role-range	RDFS	role		asymmetric	intransitive
		rdfs:range			
class-subsumption	RDFS	subclass		asymmetric	transitive
		rdfs:subclassOf			
property-subsumption	RDFS	subproperty		asymmetric	transitive
		rdfs:subpropertyOf			
class-equivalence	OWL	owl:equivalentClass		symmetric	transitive
property-equivalence	OWL	owl:equivalentProperty		symmetric	transitive
individual-equivalence	OWL	owl:sameAs		symmetric	transitive
inequality	OWL	owl:differentFrom		symmetric	intransitive
disjointness	OWL	owl:disjointFrom		symmetric	intransitive

Table B.20 *(continued)*

<Binary_Relation>					
	source	**role**	**reflexivity**	**symmetry**	**transitivity**
complementarity	OWL	owl:complementOf		symmetric	intransitive

Table B.21 — OWL Relations (except Binary Relations) as ISO/IEC 11179-32 Relations

<Relation> (excluding Binary_Relations)			
	source	**arity**	**role**
owl:AllDifferent	OWL		operand
owl:allValuesFrom	OWL	3	class
			role
			range
owl:someValuesFrom	OWL	3	class
			role
			range
owl:hasValue	OWL	3	class
			role
			value
owl:intersectionOf	OWL		intersection
			operand
owl:unionOf	OWL		union
			operand
owl:oneOf	OWL		enumeration
			member

Table B.22 — OWL Constructs as ISO/IEC 11179-32 Relation Roles

<Relation_Role>		
	multiplicity	**ordinal**
instance		1
rdf:type		2
subclass		1
rdfs:subclassOf		2
subproperty		1
rdfs:subpropertyOf		2
role		1
rdfs:range		2
owl:equivalentClass		
owl:equivalentProperty		
owl:sameAs		
owl:differentFrom		
owl:disjointFrom		
owl:complementOf		
owl:AllDifferent.operand		
owl:allValuesFrom.class		1
owl:allValuesFrom.role		2

Table B.22 *(continued)*

<Relation_Role>		
	multiplicity	**ordinal**
owl:allValuesFrom.range		3
owl:someValuesFrom.class		1
owl:someValuesFrom.role		2
owl:someValuesFrom.range		3
owl:hasValue.class		1
owl:hasValue.role		2
owl:hasValue.value		3
owl:intersectionOf.intersection		1
owl:intersectionOf.operand		2
owl:unionOf.union		1
owl:unionOf.operand		2
owl:oneOf.enumeration		1
owl:oneOf.member		2

Table B.23 — OWL Constructs as ISO/IEC 11179-32 Links

<Link>			
source	**relation**	**link_end**	
		end_role	**end**
RDF	instance-type	instance	rdf:type
		rdf:type	rdf:Property
RDFS	instance-type	instance	rdfs:Class
		rdf:type	rdfs:Class
RDFS	instance-type	instance	rdfs:range
		rdf:type	rdf:Property
RDFS	role-range	role	rdfs:range
		rdf:range	rdf:Class
RDFS	role-range	role	role
		rdf:range	rdf:Property
RDFS	role-range	role	rdf:type
		rdf:range	rdfs:Class
RDFS	instance-type	instance	rdfs:subclassOf
		rdf:type	rdf:Property
RDFS	role-range	role	rdfs:subclassOf
		rdf:range	rdfs:Class
RDFS	role-range	role	subclass
		rdf:range	rdfs:Class
RDFS	instance-type	instance	rdfs:subpropertyOf
		rdf:type	rdf:Property
RDFS	role-range	role	rdfs:subpropertyOf
		rdf:range	rdf:Property
RDFS	role-range	role	subproperty
		rdf:range	rdf:Property

Table B.23 *(continued)*

<Link>			
source	**relation**	**link_end**	
		end_role	**end**
RDFS	instance-type	instance	rdfs:Resource
		rdf:type	rdfs:Class
RDFS	class-subsumption	subclass	rdfs:Class
		rdfs:subclassOf	rdfs:Resource
RDFS	instance-type	instance	rdf:Property
		rdf:type	rdfs:Class
RDFS	class-subsumption	subclass	rdf:Property
		rdfs:subclassOf	rdfs:Resource
RDFS	instance-type	instance	rdfs:Datatype
		rdf:type	rdfs:Class
RDFS	class-subsumption	subclass	rdfs:Datatype
		rdfs:subclassOf	rdfs:Class
RDFS	instance-type	instance	rdfs:Literal
		rdf:type	rdfs:Class
RDFS	class-subsumption	subclass	rdfs:Literal
		rdfs:subclassOf	rdfs:Resource
XSD	instance-type	instance	xsd:string
		rdf:type	rdfs:Datatype
XSD	instance-type	instance	xsd:decimal
		rdf:type	rdfs:Datatype
XSD	instance-type	instance	xsd:integer
		rdf:type	rdfs:Datatype
XSD	instance-type	instance	xsd:boolean
		rdf:type	rdfs:Datatype
XSD	instance-type	instance	xsd:date
		rdf:type	rdfs:Datatype
...other XSD datatype links...			
OWL	instance-type	instance	owl:Thing
		rdf:type	owl:Class
OWL	instance-type	instance	owl:Nothing
		rdf:type	owl:Class
...other OWL metamodel links...			

B.5.3 Car Registration Ontology

An OWL ontology has been developed for illustration, for the application described in ISO/TR 9007:1987, Appendix B. Figure B.2 shows a graphical depiction of the ontology, as rendered with OntoViz.

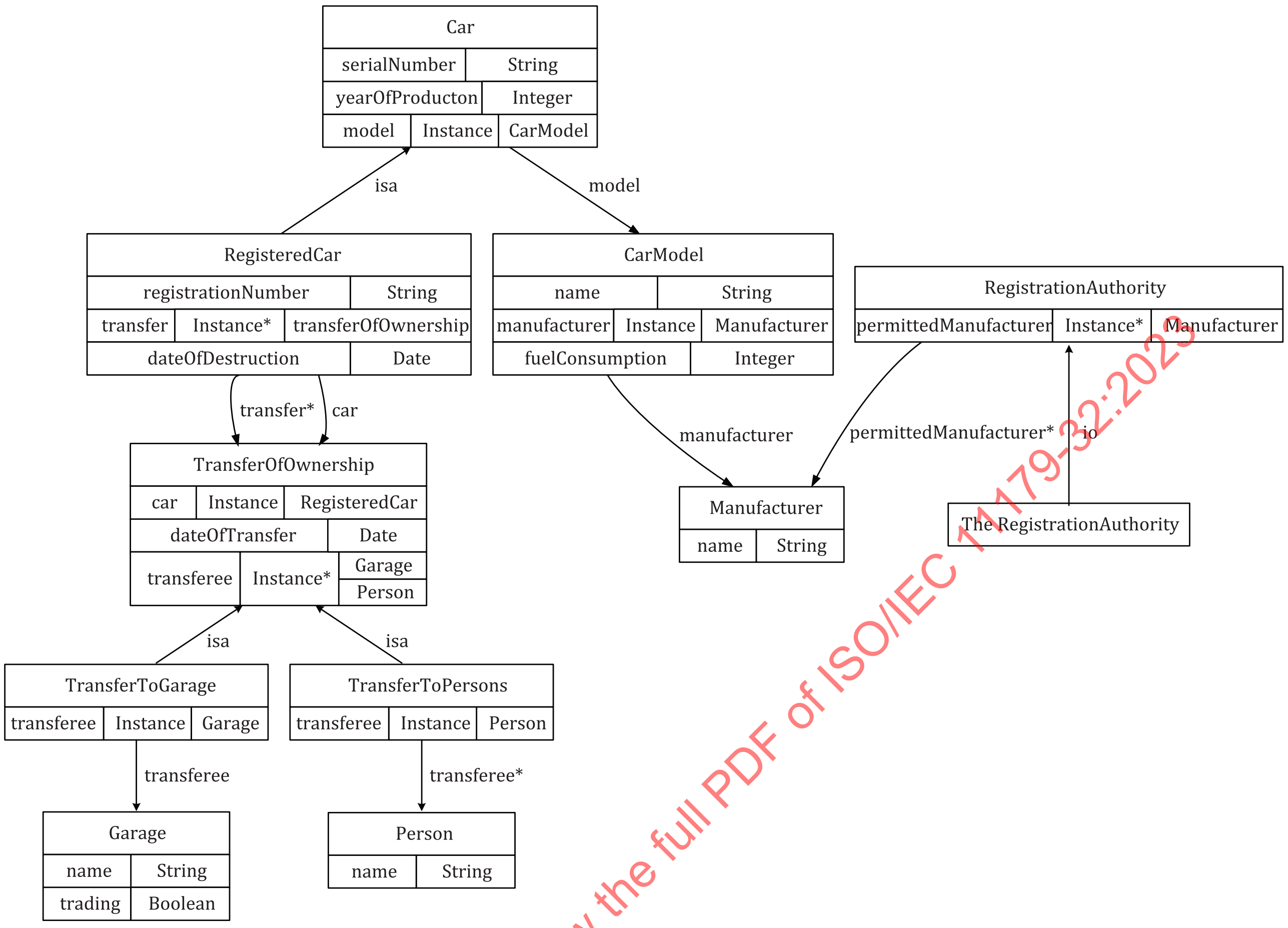
Car
serialNumber | String
yearOfProducton | Integer
model | Instance | CarModel
isa
model
RegisteredCar
registrationNumber | String
transfer | Instance* | transferOfOwnership
dateOfDestruction | Date
CarModel
name | String
manufacturer | Instance | Manufacturer
fuelConsumption | Integer
RegistrationAuthority
permittedManufacturer | Instance* | Manufacturer
transfer*
car
manufacturer
permittedManufacturer*
io
TransferOfOwnership
car | Instance | RegisteredCar
dateOfTransfer | Date
transferee | Instance* | Garage Person
Manufacturer
name | String
The RegistrationAuthority
isa
isa
TransferToGarage
transferee | Instance | Garage
TransferToPersons
transferee | Instance | Person
transferee
transferee*
Garage
name | String
trading | Boolean
Person
name | String

Figure B.2 — Car Registration Ontology in OWL

Below is the complete text of the ontology, in Turtle syntax:

```
@prefix :  <http://xmdr.org/ont/ISO9007.owl#> .
@prefix rdfs:    <http://www.w3.org/2000/01/rdf-schema#> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix owl:     <http://www.w3.org/2002/07/owl#> .
@prefix xsd:     <http://www.w3.org/2001/XMLSchema#> .

<http://xmdr.org/ont/ISO9007.owl>
      rdf:type owl:Ontology .

:Car
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :yearOfProduction
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :model
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :serialNumber
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Garage ,
       :Manufacturer , :RegistrationAuthority , :CarModel .
```

```
:CarModel
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :name
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :fuelConsumption
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :manufacturer
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Garage ,
       :Manufacturer , :Car , :RegistrationAuthority .

:Manufacturer
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :name
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Garage ,
       :Car , :RegistrationAuthority , :CarModel .

:RegistrationAuthority
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:maxCardinality "5"^^xsd:int ;
                owl:onProperty :permittedManufacturer
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Garage ,
       :Manufacturer , :Car , :CarModel ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:oneOf (:TheRegistrationAuthority)
              ] .

:TheRegistrationAuthority
      rdf:type :RegistrationAuthority .

:RegisteredCar
      rdf:type owl:Class ;
      rdfs:subClassOf :Car ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :registrationNumber
              ] .

:TransferOfOwnership
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:minCardinality "1"^^xsd:int ;
                owl:onProperty :transferee
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :dateOfTransfer
              ] ;
      owl:disjointWith :Person , :Garage , :Manufacturer , :Car ,
       :RegistrationAuthority , :CarModel ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:unionOf (:TransferToGarage :TransferToPersons)
              ] .

:TransferToGarage
      rdf:type owl:Class ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :transferee
              ] ;
      owl:disjointWith :TransferToPersons ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:intersectionOf ([ rdf:type owl:Restriction ;
                            owl:allValuesFrom :Garage ;
                            owl:onProperty :transferee
                          ] :TransferOfOwnership)
              ] .

:Garage
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :name
              ] ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :trading
              ] ;
      owl:disjointWith :TransferOfOwnership , :Person , :Manufacturer ,
       :Car , :RegistrationAuthority , :CarModel .

:TransferToPersons
      rdf:type owl:Class ;
      owl:disjointWith :TransferToGarage ;
      owl:equivalentClass
              [ rdf:type owl:Class ;
                owl:intersectionOf (:TransferOfOwnership [ rdf:type owl:Restriction ;
                            owl:allValuesFrom :Person ;
                            owl:onProperty :transferee
                          ])
              ] .

:Person
      rdf:type owl:Class ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ rdf:type owl:Restriction ;
                owl:cardinality "1"^^xsd:int ;
                owl:onProperty :name
              ] ;
      owl:disjointWith :TransferOfOwnership , :Garage , :Manufacturer ,
       :Car , :RegistrationAuthority , :CarModel .

:yearOfProduction
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :Car ;
      rdfs:range xsd:int .

:model
      rdf:type owl:ObjectProperty , owl:FunctionalProperty ;
      rdfs:domain :Car ;
      rdfs:range :CarModel .

:serialNumber
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :Car ;
      rdfs:range xsd:string .

:name
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain
              [ rdf:type owl:Class ;
                owl:unionOf (:CarModel :Manufacturer :Garage :Person)
              ] ;
      rdfs:range xsd:string .

:fuelConsumption
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:comment
            "number of litres of hydrocarbon fuel per 100 kilometres. Ranges from 4 to
25."^^xsd:string ;
      rdfs:domain :CarModel ;
      rdfs:range xsd:int .

:manufacturer
      rdf:type owl:ObjectProperty , owl:FunctionalProperty ;
      rdfs:domain :CarModel ;
      rdfs:range :Manufacturer .

:permittedManufacturer
      rdf:type owl:ObjectProperty ;
      rdfs:domain :RegistrationAuthority ;
      rdfs:range :Manufacturer .

:registrationNumber
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :RegisteredCar ;
      rdfs:range xsd:string .

:dateOfDestruction
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :RegisteredCar ;
      rdfs:range xsd:date .

:transfer
      rdf:type owl:ObjectProperty , owl:InverseFunctionalProperty ;
      rdfs:domain :RegisteredCar ;
      rdfs:range :TransferOfOwnership ;
      owl:inverseOf :car .

:car
      rdf:type owl:ObjectProperty , owl:FunctionalProperty ;
      rdfs:domain :TransferOfOwnership ;
      rdfs:range :RegisteredCar ;
      owl:inverseOf :transfer .

:transferee
      rdf:type owl:ObjectProperty ;
      rdfs:domain :TransferOfOwnership ;
      rdfs:range
              [ rdf:type owl:Class ;
                owl:unionOf (:Garage :Person)
              ] .

:dateOfTransfer
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :TransferOfOwnership ;
      rdfs:range xsd:date .

:trading
      rdf:type owl:DatatypeProperty , owl:FunctionalProperty ;
      rdfs:domain :Garage ;
```

```
    rdfs:range xsd:boolean .
```

Using the metamodels above, the following description in terms of the 11179-32 Concept System metamodel results, as shown in Table B.24 through Table B.29.

Table B.24 — Car Registration Model in OWL — ISO/IEC 11179-32 Concept System

<Concept_System>			
	notation	**referencedConceptSystem**	**importedConceptSystem**
ISO9007.B	OWL/Turtle	OWL	

Table B.25 — Car Registration Model in OWL — ISO/IEC 11179-32 Concepts

<Concept> (excluding Relations and Relation_Roles)	
	source
:Car	ISO9007.B
:CarModel	ISO9007.B
:Manufacturer	ISO9007.B
:RegistrationAuthority	ISO9007.B
:TheRegistrationAuthority	ISO9007.B
:RegisteredCar	ISO9007.B
:TransferOfOwnership	ISO9007.B
:TransferToGarage	ISO9007.B
:Garage	ISO9007.B
:TransferToPersons	ISO9007.B
:Person	ISO9007.B
:yearOfProduction	ISO9007.B
:serialNumber	ISO9007.B
:name	ISO9007.B
:fuelConsumption	ISO9007.B
:registrationNumber	ISO9007.B
:dateOfDestruction	ISO9007.B
:dateOfTransfer	ISO9007.B
:trading	ISO9007.B

Table B.26 — Car Registration Model in OWL — ISO/IEC 11179-32 Binary Relations

<Binary_Relation>					
	source	**role**	**reflexivity**	**symmetry**	**transitivity**
relation1	ISO9007.B	:model		asymmetric	instransitive
		inverse of :model			
relation2	ISO9007.B	:manufacturer		asymmetric	instransitive
		inverse of :manufacturer			
relation3	ISO9007.B	:permittedManufacturer		asymmetric	instransitive
		inverse of :permittedManufacturer			
relation4	ISO9007.B	:transfer		asymmetric	instransitive
		:car			
relation5	ISO9007.B	:transferee		asymmetric	instransitive
		inverse of :transferee			