FACT BASED MODELLING

Exchanging Conceptual Data Models

This document is under production by the so-called “exchanging FBM Working Group” made of experts and users members of the Fact Based Modelling community.

This is the first official Draft for review by the FBM WG

The review will last up to 31 January 2012

DISCLAIMER

This document is Draft. It is subject to change without any notice and may not be referred without an explicit authorisation of the authors.

For any information, please contact:

serge.valera@esa.int (convenor)

inge.lemmens@pna-group.nl (secretariat)
8.36 Equality.........................................................................................................................63
8.37 Exclusion........................................................................................................................64
8.38 Join path..........................................................................................................................65
8.39 Join path projection........................................................................................................68
8.40 Join path multibranch....................................................................................................69
8.41 Cardinality ....................................................................................................................71
8.42 Value constraint ............................................................................................................72
8.43 Adhoc constraint ..........................................................................................................75
8.44 Sample population.........................................................................................................77
8.45 Sample population and instance compatibility .............................................................79
Introduction

1 Scope

2 Conformance

3 Normative references

4 Terms and definitions

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

4.1 alethic
concerning truth or possibility

EXAMPLE An alethic constraint is a restriction that cannot be violated by any state or state transition of the conceptual data model. An alethic constraint may be expressed positively as a necessity, or negatively by indicating an impossibility e.g.a) It is necessary that each Person was born on at most one Date; b) It is impossible that some Person was born on more than one Date.

4.2 alethic modality
modality of necessity and possibility

NOTE The alethic modal operators used are: “it is necessary that”, “it is possible that”, and “it is impossible that”.

4.3 arity
number of roles in a fact type or predicate

NOTE 1 A fact type containing exactly one role is called a unary fact type.

NOTE 2 A fact type containing exactly two roles is called a binary fact type.

NOTE 3 A fact type containing more than two roles is called a n-ary fact type.

4.4 asserted fact
fact that is declared to be the case rather than being derived from other facts

SYNONYMS primitive fact, base fact, extensional fact

4.5 asserted fact type
fact type, each of whose instances is an asserted fact

SYNONYMS primitive fact type, base fact type, extensional fact type

4.6 asserted subtype
subtype, all of whose instances are asserted
EXAMPLE Simply assert that FemalePerson is a subtype of Person, rather than (a) adding a fact type such as "Person is of Gender" and (b) deriving membership in FemalePerson by a derivation rule such as "Each FemalePerson is a Person who is of Gender 'F'."

4.7 atomic fact fact that cannot be decomposed, without loss of information, into multiple facts involving exactly the same objects

NOTE An atomic fact is either an elementary fact or an existential fact.

SYNONYM irreducible fact

4.8 atomic fact type fact type, each of whose instances is an atomic fact

4.9 cardinality constraint constraint on an object type or role that determines the possible number of instances that the object type or the role may contain for any given state of the fact base

NOTE 1 The full terms "object cardinality constraint" and "role cardinality constraint" should be used to distinguish them from cardinality constraints in other approaches (e.g. ER and OWL).

NOTE 2 The possible number of instances can be expressed by means of one or more ranges specified by their minimum and/or maximum values.

4.10 compatible object types object types whose populations may have some instances in common

NOTE 1 Two object types are compatible if and only if one of the following applies: (1) the object types are the same, (2) one is a subtype of the other; (3) the object types share a common supertype and are not declared mutually exclusive.

NOTE 2 Value types based on the same data type are considered to be compatible value types.

4.11 compatible roles roles whose associated object types are compatible

4.12 compound fact fact that can be decomposed without loss of information into multiple facts involving exactly the same objects

NOTE A compound fact is not atomic.

4.13 compound fact type fact type, each of whose instances is a compound fact

4.14 conceptual schema schema that specifies the possible and permitted populations and the possible and permitted transitions of the fact base, as well as an explanation of every term that could be misunderstood by the intended audience

NOTE A fact based conceptual schema declares the fact types, constraints, derivation rules, and concept definitions relevant to the universe of discourse.
4.15 **conceptualization principle**
principle governing how the conceptual data model maps the universe of discourse

NOTE Conceptual data models include only conceptually relevant aspects, both static and dynamic, of the universe of discourse, thus excluding all aspects of (external or internal) data representation, physical data organization and access as well as all aspects of a particular external user representation such as message format, data structures, refer to ISO TC97/SC5/WG3 TR9007 (1987).

4.16 **constraint**
restriction on what populations or transitions of the fact base are possible for or permitted by the conceptual schema

4.17 **constraint arity**
number of roles in a sequence of roles

4.18 **deontic**
pertaining to necessity, duty or obligation, or expressions conveying this

NOTE A deontic constraint is a constraint that could be violated but ought to be obeyed. A deontic constraint is an obligation. A deontic constraint can also be phrased in a negative way as a prohibition to indicate what is not permitted.

4.19 **deontic modality**
modality of obligation and permission

NOTE The deontic modal operators used are “it is obligatory that”, “it is permitted that”, and “it is forbidden that”.

4.20 **derivation rule**
rule that specifies how to derive instances of a derived fact type or semiderived fact type from other facts, or how to derive instances in a derived subtype or semiderived subtype

4.21 **derivation status**
indication that the instances of a fact type (a) are all asserted, or (b) are all derived, or (c) may include some asserted facts and some derived facts

4.22 **derived fact**
fact that is deduced from other facts by means of a derivation rule

NOTE A fact that is not derived is an asserted fact.

4.23 **derived fact type**
fact type, each of whose instances is a derived fact

4.24 **derived subtype**
subtype, all of whose instances are derived from facts of its supertype(s) by means of a derivation rule

4.25 **elementary fact**
fact that declares that an object has a property or that one or more objects participate in a relationship, where the fact cannot be split into simpler facts with the same object(s) without information loss
NOTE 1  An object in an elementary fact is an individual, in the sense in which this term is used in classical logic.

NOTE 2  An elementary fact is expressed by a fact reading that predicates over one or more individuals (i.e. applies a logical predicate to a non-empty sequence of individuals). Fact readings that mean the same express the same fact. Existence is not a property or predicate. Simply asserting that an object exists is an atomic (and existential) fact, but not an elementary fact.

EXAMPLE  Both of the following fact readings express the same fact: 1/ The Politician named “Julia Gillard” governs the Country named “Australia”. The Country named “Australia” is governed by the Politician named “Julia Gillard”.

4.26  
**elementary fact type**
atomic fact type, each of whose instances are elementary facts

4.27  
**entity**
object that is referenced by a definite description that relates it to other objects

NOTE  An entity can typically change its state over time (e.g. by participating in new facts). Not a value (lexical constant), such as a name or numeral.

SYNONYM non-lexical object

EXAMPLE 1  The Country named “Australia”.

EXAMPLE 2  The President named “Barack Obama”.

4.28  
**entity type**
object type, each of whose instances are entities

4.29  
**equality constraint**
set-comparison constraint that specifies that, for each state of the fact base, the populations of the constrained sequences of role occurrences must be equal

4.30  
**exclusion constraint**
set-comparison constraint that specifies that, for each state of the fact base, the populations of the constrained sequences of role occurrences must be mutually exclusive i.e. do not overlap

4.31  
**exclusive-or constraint**
combination of an inclusive-or constraint and an exclusion constraint over the same set of roles

SYNONYM XOR constraint

4.32  
**existential fact**
atomic fact that asserts the existence of an object

EXAMPLE 1  There exists a Country named “Australia”.

EXAMPLE 2  There exists a Country code “AU”.

4.33  
**existential fact type**
atomic fact type, each of whose instances is an existential fact
4.34 fact
proposition that is taken to be true by the relevant community whereby the proposition is represented by a predicate with an associated ordered set of objects

NOTE A fact is neither a constraint nor a derivation rule.

4.35 fact base
set of facts of interest to the relevant community and that satisfies the rules of the conceptual schema

NOTE The set of asserted facts is called the extensional database. The set of derived facts is called the intensional database.

4.36 fact reading
sentence expressing a fact

4.37 fact type
type, each of whose instances are facts that express the same kind of information

NOTE 1 A fact type is a non-empty set of typed predicates, i.e. predicates where each placeholder is applied to a specific object type, with the same semantics.

NOTE 2 Each instance of a fact type includes one object for each role.

4.38 fact type reading
reading for a fact type that inserts the object type names in the placeholders within a predicate reading

SYNONYM sentential form

EXAMPLE Person was born in Country.

4.39 far role of an object type
If an object type O plays role r1 of a binary fact type whose other role is r2, then r2 is a far role of O.

NOTE For any given object type, its far role names must be distinct.

EXAMPLE 1 The role named “birthCountry” in the fact type “Person was born in Country [birthCountry]” is a far role of Person.

EXAMPLE 2 The roles named “parent” and “child” in the fact type “Person [parent] is a parent of Person [child]” are far roles of Person.

4.40 frequency constraint
constraint that restricts, for each state of the fact base, the number of times any given sequence of objects that instantiate the constrained role sequence appears in the population of that role sequence

NOTE 1 The allowed number of instances can be expressed by means of one or more ranges, i.e. minimum and/or maximum values.

NOTE 2 A uniqueness constraint should always be used instead of a frequency constraint having a maximum frequency of 1.

4.41 functional role
role with an alethic uniqueness constraint that applies to that role only
4.42
identity binary fact type
<to be defined>

4.43
inclusive-or constraint
constraint over two or more roles played by the same object type or by compatible object types, which
specifies that all instances of the object type(s) must play at least one of the constrained roles

EXAMPLE  disjunctive mandatory role constraint (mandatory constraint over a logical disjunction of two or more
roles)

4.44
instance
member of a specified type

4.45
join path
connected path that navigates from one object type occurrence to another by traversing through multiple
predicates, conceptually joining the role used to exit a predicate with the compatible role used to enter the
next predicate

EXAMPLE  Given the fact types F1 "Advisor speaks Language" and F2 "Language is used by Country", the join path
"Advisor speaks some Language that is used by Country" navigates from advisor’s role in F1, then conceptual inner joins
language’s role in F1 to language’s role in F2, and then ends at country’s role in F2.

4.46
mandatory role
role that must be played by all instances in the population of the role’s associated object type

SYNONYM total role

4.47
mandatory role constraint
constraint requiring that each instance in the population of a given object type must play the constrained role
or at least one of the constrained roles

NOTE 1  A mandatory role constraint applying to one role only is also known as simple mandatory role constraint.

NOTE 2  A mandatory role constraint applying to more than one role is also known as disjunctive mandatory role
constraint or inclusive-or constraint.

4.48
modality
construct from modal logic that classifies the mode in which a proposition applies

NOTE  Fact-orientation uses modality as one way to classify constraints, and currently considers only alethic and
deontic modalities.

4.49
object
ground level single thing, individual or value, about which the community wants to exchange facts

NOTE  An object is either an entity or a value. It cannot be both.

EXAMPLES  the person named “Barack Obama”, the country code “AU”

4.50
object type
concept used to classify the objects in the universe of discourse into different kinds
NOTE Object type is either an entity type or a value type.

4.51 objectification
process of making an object from a fact in order to talk about the state of affairs that corresponds to, but is not identical to, the fact

NOTE 1 Propositional nominalization is not supported. Propositional nominalization is the use of a noun phrase to refer to the proposition expressed by a declarative sentence. For example, in the sentence “I know that it’s snowing outside”, the noun phrase “that it’s snowing outside” is used to nominalize the proposition expressed by the sentence “It’s snowing outside 1”.

NOTE 2 Situational nominalization is the use of a noun phrase to refer to the state of affairs denoted by another linguistic expression. For example, in the sentence “That snowing is beautiful”, the noun phrase “that snowing” is used to nominalize the state of affairs referenced in the sentence “It’s snowing outside”.

4.52 objectified fact type
object type resulting from the objectification of a fact type

4.53 population
set of instances in a given state of the fact base, of an object type, of a fact type, or a role under discussion

NOTE The population of a type may change over time.

4.54 predicate
result of replacing the objects of interest in a proposition by placeholders for those objects

NOTE 1 A predicate is denoted by a declarative sentence with the object terms of interest replaced by placeholders.

NOTE 2 Predicate is a semantic, not a syntactic construct.

SYNONYM logical predicate

4.55 predicate reading
sentence with placeholders for object(s) that expresses a logical predicate

NOTE A predicate must have at least one predicate reading.

EXAMPLES “… employs …”, “… was born in …”, “… played … for …”

4.56 preferred identifier of an entity type
reference scheme chosen by the domain expert as the preferred way of identifying instances of the entity type

EXAMPLE The entity type Country may have its instances identified using the injective fact types “Country has CountryCode” or “Country has CountryName”. If Country’s relationship to CountryCode is chosen for its preferred reference scheme, then its relationship to CountryName may still be used as an alternative way to refer to countries.

4.57 property
 unary predicate in which an object participates

EXAMPLE “drinking” is a property of “Jan” as specified by the unary fact “Jan drinks”.

4.58 proposition
that which is asserted by a declarative sentence
NOTE   A proposition is true or false, but not both.

4.59 reference mode
mode or manner in which a single value refers to a single entity

4.60 reference scheme
means for identifying instances of the entity type under discussion

NOTE    A reference mode is a simple kind of reference scheme.

4.61 relationship
predicate in which two or more objects participate

4.62 ring constraint
logical constraint between two type-compatible role occurrences that specifies how the populations of these role occurrences may be related

EXAMPLES  irreflexive, asymmetric, intransitive, antisymmetric, acyclic ring constraints.

4.63 role
constituting part of a fact type that indicates the function played by the instances of the associated object type

4.64 role occurrence
instance of navigating through the role of a fact type

EXAMPLE    The derived fact type definition “Person1 is a grandparent of Person2 if and only if Person1 is a parent of some Person3 who is a parent of Person2” involves a defining role path that includes two occurrences of each of the roles in the fact type “Person is a parent of Person”.

4.65 semiderived fact type
fact type that may have a population that includes some asserted facts and some derived facts

4.66 semiderived subtype
subtype, that may have a population that includes some asserted objects and some derived objects

4.67 set-comparison constraint
constraint that specifies a condition to be satisfied when comparing the populations of compatible role-sequences

NOTE 1   The set-comparison constraints are: “subset constraint”, “equality constraint” and “exclusion constraint”.

NOTE 2   Compatible role-sequences imply that the role occurrences across matching positions in the sequences must be compatible.

4.68 subset constraint
set-comparison constraint that specifies that, for each state of the fact base, the population of a sequence of one or more role occurrences must be a subset of the population of another compatible sequence of role occurrences
4.69 subtype
object type, each of whose instances belong to an encompassing type

EXAMPLE Woman is a (proper) subtype of Person.

4.70 supertype
object type that has at least one subtype and may have some instances not in that subtype

4.71 uniqueness constraint
constraint over a sequence of one or more roles that requires that in each state of the fact base, each instantiation of that role sequence occurs only once

4.72 value
object that identifies itself and is unchangeable

EXAMPLES 12, “Amsterdam”

SYNONYMS lexical object, label

4.73 value comparison constraint
constraint that specifies how the values of instances of two roles with co-roles played by the same object are related by one of the following comparison operators: <, ≤, >, ≥

4.74 value constraint
constraint that specifies the possible values for the instances of a role or a value type

4.75 value role player
role played by an object type that ultimately is identified by a single value

4.76 value type
object type, each of whose instances must be values

5 Symbols (and abbreviated terms)

CogNIAM cognition enhanced natural language information analysis method

FBM fact based modelling

FCO-IM fully communication oriented information modelling

IT information technology

NIAM natural language information analysis method

ORM object role modelling

OWL web ontology language

SBVR semantics of business vocabulary and business rules

UoD universe of discourse
6 Conventions

6.1 Data definition

This Standard defines the FBM exchange schema that can be used by which suppliers and customers for exchanging any conceptual data model. The different semantic elements that constitute the FBM exchange schema are specified in natural language. Where possible, a graphical representation of these requirements is also provided. When found necessary, examples are also given.

Mapping with existing modelling languages and notations are provided in other documents, refer to FBM1005-nn.

6.2 Case sensitivity

All words used in expressing the data requirements in this Standard are case insensitive.

6.3 Graphical notation

Fact based modelling notations and tools provide graphical means to represent semantic definitions. Although there is no agreement on a common graphical representation for use in the FBM specification, for the purpose of facilitating the review of the textually expressed requirements, for WD01 of the Standard the ORM2 notation (refer to Annex B) is provided. It is noted that ORM2 cannot represent everything graphically, the textual representation of the requirements remains the only applicable baseline.

6.4 UUID

Taking into account the distributed approach in developing conceptual data models promoted by this Standard, the uniqueness of names assigned by a given conceptual data model supplier cannot be guaranteed at customer level (e.g. when merging supplied conceptual data models). For this reason, to uniquely identify data to be exchanged, this Standard uses universally unique identifiers (UUID), in compliance with ITU-T Rec. X.667 - ISO/IEC 9834-8.

UUIDs are strictly speaking not of conceptual nature. They are defined to ensure the unique identification of semantic elements involved in relationships and are especially required for sharing and reusing (part of) conceptual data models.

7 Exchanging FBM conceptual data models

This Standard specifies what information shall be delivered by a supplier for the purpose of exchanging conceptual data model(s). According to the data modelling tool used to produce the conceptual data model(s) and the specific needs of suppliers and customers, the exchange schema to apply to the supplier/customer interaction corresponds to:

- a subset of the standard exchange schema, supplemented by
- tool-specific add-ons.

The standard exchange schema to be used for the exchange of conceptual data models is specified in Annex A.

Any tool-specific extension to the standard exchange schema shall be backward compatible with the standard exchange schema.

The standard exchange schema is suitable for communication, interpretation and processing by computers.
Formal languages are used within this Standard for the definition of data types and constants. Data delivered by suppliers should comply with these languages. Such compliance is encouraged to optimize the process of sharing and reuse of product data.

If other languages are used, these languages shall be formally specified using the extended Backus-Naur form (EBNF), see ISO/IEC 14977.

The standard exchange schema provides implicit ordering of data to be exchanged. Unless explicitly mentioned i.e. in clause 5.2, the order in which data is organised is not semantically relevant.

For verification purpose, where there are no explicit definitions of which order is to be produced in the exchange schema, ordering the exported data alpha-numerically is recommended.

Within the exchange schema, references to semantic elements are realised by use of the related UUID.

8 Requirements

8.1 Conventions

In the following clauses, the specification of the fact based modelling is decomposed into sets of concepts for which requirements are given. Each set consists of:

8.x Set Name
8.x.1 Introduction
8.x.2 Assertions
8.x.3 Derivations
8.x.4 Other constraints
8.x.5 Other considerations
8.x.6 Graphical representation
8.x.6.1 ORM 2
8.x.6.y other graphical representation if required…

8.2 Core elements

8.2.1 Introduction

Note 'Object Type plays Role' implies that instances of that object type may play that role.

8.2.2 Assertions

a) Object type plays role

1) Each role is played by exactly one object type.

2) It is possible that the same object type plays more than one role.

b) Fact type contains role

1) Each fact type contains some role.
2) Each role belongs to exactly one fact type.
3) It is possible that the same fact type contains more than one role.

8.2.3 Derivations

8.2.4 Other constraints

8.2.5 Other considerations

a) Object type is an entity type.
   1) Reference Scheme: object type has UUID.
   2) Reference Mode: UUID.

b) Role is an entity type.
   1) Reference Scheme: role has UUID.
   2) Reference Mode: UUID.

c) Fact type is an entity type.
   1) Reference Scheme: fact type has UUID.
   2) Reference Mode: UUID.

8.2.6 Graphical representation

8.2.6.1 ORM2 representation

8.3 Object type

8.3.1 Introduction

8.3.2 Assertions

a) Object type has primary object type name
   1) Each object type has exactly one primary object type name.
   2) Each object type name is primary name of at most one object type.

b) Value type draws values from data type
   1) Each value type draws values from exactly one data type.
   2) It is possible that more than one value type draws values from the same data type.
8.3.3 Derivations

8.3.4 Other constraints

a) For each semantic object type, exactly one of the following holds: that semantic object type is some entity type; that semantic object type is some value type.

b) For each object type, exactly one of the following holds: that object type is some semantic object type; that object type is some data type.

8.3.5 Other considerations

a) Object type name is a value type.
   1) Portable data type: Text: Variable Length.

b) Object type is an entity type.
   1) Reference Scheme: object type has UUID.
   2) Reference Mode: UUID.

c) Entity type is an entity type.
   1) Reference Scheme: object type has UUID.
   2) Reference Mode: UUID.

d) Value type is an entity type.
   1) Reference Scheme: object type has UUID.
   2) Reference Mode: UUID.

e) Data type is an entity type.
   1) Reference Scheme: object type has UUID.
   2) Reference Mode: UUID.

f) Semantic object type is an entity type.
   1) Reference Scheme: object type has UUID.
   2) Reference Mode: UUID.
8.3.6 Graphical representation

8.3.6.1 ORM2 representation

8.4 Role name

8.4.1 Introduction

Note Note on derived fact types: These fact types exist to describe the far role and dot role concepts, and to apply the internal uniqueness constraints, which are not implied. This model assumes set semantics for all fact types, including derived fact types, so a spanning uniqueness constraint on a binary 'ObjectType has far RoleName' fact type would be implied by the set projection and add no additional restriction. The ternaries are necessary to form a set without an implied uniqueness constraint, so that the applied uniqueness constraints can enforce additional restrictions.

8.4.2 Assertions

a) Role has role name

1) Each role has at most one role name.

2) It is possible that the same role name is of more than one role.

b) Fact type contains role

1) Each fact type contains some role.

2) Each role belongs to exactly one fact type.

3) It is possible that the same fact type contains more than one role.

8.4.3 Derivations

a) Object type has far role name from fact type

1) *Object type has far role name from fact type if and only if that object type plays some role1 that belongs to that fact type that is some binary fact type and contains some role2 that has that role name where role1 <> role2.

2) For each object type and far role name, that object type has that far role name from at most one fact type.
b) Object type has dot role name from fact type

1) *Object type has dot role name from fact type if and only if that object type has that far role name from that fact type or plays some role that has that role name and belongs to that fact type that is some unary fact type.*

2) For each object type and dot role name, that object type has that dot role name from at most one fact type.

8.4.4 Other constraints

a) For each fact type and role name, at most one role belongs to that fact type and has that role name.

8.4.5 Other considerations

a) Role name is a value type.

1) Portable data type: Text: Variable Length.

8.4.6 Graphical representation

8.4.6.1 ORM2 representation

8.5 Fact type name

8.5.1 Introduction

8.5.2 Assertions

a) Fact type has fact type name

1) Each fact type has at most one fact type name.

2) Each fact type name is of at most one fact type.

8.5.3 Derivations

8.5.4 Other constraints

8.5.5 Other considerations

a) Fact type name is a value type.
1) **Portable data type**: Text: Variable Length.

### 8.5.6 Graphical representation

#### 8.5.6.1 ORM2 representation

![Graphical representation of ORM2]

### 8.6 Fact type and arity

#### 8.6.1 Introduction

#### 8.6.2 Assertions

#### 8.6.3 Derivations

a) **Fact type has arity**

1) *Fact type has arity if and only if that fact type contains some role where arity = count(each role for that fact type).*

2) Each fact type has exactly one arity.

3) It is possible that more than one fact type has the same arity.

#### 8.6.4 Other constraints

a) For each fact type, exactly one of the following holds: that fact type is some unary fact type; that fact type is some binary fact type; that fact type is some N ary fact type.

#### 8.6.5 Other considerations

a) **Binary fact type is an entity type.**

1) *Each binary fact type is some fact type that has some arity where the possible value of that arity is 2.*

2) Reference Scheme: fact type has UUID.

3) Reference Mode: UUID.

b) **Unary fact type is an entity type.**

1) *Each unary fact type is some fact type that has some arity where the possible value of that arity is 1.*

2) Reference Scheme: fact type has UUID.

3) Reference Mode: UUID.

c) **N ary fact type is an entity type.**
1) *Each N ary fact type is some fact type that has some arity where the possible values of that arity are at least 3.*

2) Reference Scheme: fact type has UUID.

3) Reference Mode: UUID.

8.6.6 Graphical representation

8.6.6.1 ORM2 representation

8.7 Fact type and predicate

8.7.1 Introduction

Note A Predicate is an ordered set of roles. Hence, for each Predicate, the set of Role and Ordinal tuples consisting of the Roles at Ordinals in that Predicate determines that Predicate.

8.7.2 Assertions

a) Fact type has predicate

1) Each fact type has some predicate.

2) Each predicate orders roles of exactly one fact type.

3) It is possible that the same fact type has more than one predicate.

b) Predicate positions role at ordinal

1) For each predicate and role, that predicate positions that role at at most one ordinal.

2) This association with predicate, role provides the preferred identification scheme for predicate role ordinal.

3) For each predicate and ordinal, that predicate positions at most one role at that ordinal.

4) Each predicate positions some role at some ordinal.

5) Each role occupies some ordinal in some predicate.

8.7.3 Derivations

a) Predicate has arity
1) *Predicate has arity if and only if that predicate positions some role at some ordinal where arity = count(each role for that predicate).

2) Each predicate has exactly one arity.

3) It is possible that more than one predicate has the same arity.

8.7.4 Other constraints

a) For each predicate and arity, that predicate has that arity if and only if that predicate orders roles of some fact type that has that arity.

b) If some fact type has some predicate that positions some role at some ordinal then that fact type contains that role.

8.7.5 Other considerations

a) Predicate is an entity type.

1) Reference Scheme: predicate has UUID.

2) Reference Mode: UUID.

8.7.6 Graphical representation

8.7.6.1 ORM2 representation

8.8 Fact type and predicate reading text

8.8.1 Introduction

8.8.2 Assertions

a) Predicate reading attaches text to role

1) It is possible that more than one predicate reading attaches text to the same role and that the same predicate reading attaches text to more than one role.
2) In each population of predicate reading attaches text to role, each predicate reading, role combination occurs at most once.

3) This association with predicate reading, role provides the preferred identification scheme for reading role text.

b) Reading role text has pre-bound reading text
   1) Each reading role text has at most one pre-bound reading text.
   2) It is possible that more than one reading role text has the same pre-bound reading text.

c) Predicate reading has front reading text
   1) Each predicate reading has at most one front reading text.
   2) It is possible that more than one predicate reading has the same front reading text.

d) Predicate reading is of predicate
   1) Each predicate has some predicate reading.
   2) Each predicate reading is of exactly one predicate.
   3) It is possible that the same predicate has more than one predicate reading.

e) Reading role text has post-bound reading text
   1) Each reading role text has at most one post-bound reading text.
   2) It is possible that more than one reading role text has the same post-bound reading text.

f) Reading role text has following reading text
   1) Each reading role text has at most one following reading text.
   2) It is possible that more than one reading role text has the same following reading text.

8.8.3 Derivations

8.8.4 Other constraints

a) Each predicate reading attaches text to some role or has some front reading text.

b) Each reading role text has some pre-bound reading text or has some post-bound reading text or has some following reading text.

c) If for some role, some predicate reading attaches text to that role and that predicate reading is of some predicate then that role occupies some ordinal in that predicate.

8.8.5 Other considerations

a) Reading text is a value type.
   1) Portable data type: Text: Variable Length.

b) Predicate reading is an entity type.
1) **Reference Scheme:** predicate reading has UUID.

2) **Reference Mode:** UUID.

### 8.8.6 Graphical representation

#### 8.8.6.1 ORM2 representation

![ORM2 Diagram]

### 8.9 Fact type and predicate reading ordering

#### 8.9.1 Introduction

#### 8.9.2 Assertions

a) **Predicate reading has ordinal rank**

1) *Each* predicate reading *has exactly one* ordinal rank.

2) *It is possible that more than one* predicate reading *has the same ordinal rank.*

#### 8.9.3 Derivations

a) **Predicate reading is preferred for fact type**

1) *Predicate reading is preferred for fact type if and only if that predicate reading describes that fact type and has some ordinal rank where the possible value of that ordinal is 1.*

2) *Each* predicate reading *is preferred for at most one* fact type.

3) *For each fact type, exactly one* predicate reading *is preferred for that fact type.*

b) **Predicate reading describes fact type**

1) *Predicate reading describes fact type if and only if that predicate reading is of some predicate that orders roles of that fact type.*

2) *Each* fact type *has some* predicate reading.

3) *Each* predicate reading *describes exactly one* fact type.
4) It is possible that the same fact type has more than one predicate reading.

c) Predicate reading1 is preferred for predicate

1) Predicate reading1 is preferred for predicate if and only if that predicate reading1 has some ordinal1 rank and is of that predicate where it is not true that that predicate has some predicate reading2 that has some ordinal2 rank where ordinal2 < ordinal1.

2) Each predicate reading is preferred for at most one predicate.

3) For each predicate, exactly one predicate reading is preferred for that predicate.

8.9.4 Other constraints

a) For each fact type and ordinal rank, at most one predicate reading describes that fact type and has that ordinal rank.

8.9.5 Other considerations

8.9.6 Graphical representation

8.9.6.1 ORM2 representation

8.10 Fact type and world assumption

8.10.1 Introduction

8.10.2 Assertions

a) Unary fact type has world assumption

1) Each unary fact type has exactly one world assumption.

2) It is possible that more than one unary fact type has the same world assumption.

8.10.3 Derivations

8.10.4 Other constraints

8.10.5 Other considerations

a) World assumption is an entity type.
1) Reference Scheme: world assumption has world assumption.name.

2) Reference Mode: .name.

3) The possible values of world assumption are 'ClosedWorld', 'OpenWorld', 'OpenWorldWithNegation'.

8.10.6 Graphical representation

8.10.6.1 ORM2 representation

8.11 Role classification

8.11.1 Introduction

8.11.2 Assertions

8.11.3 Derivations

a) Role is mandatory

1) *Role is mandatory if and only if some mandatory constraint restricts that role and has some arity where the possible value of that arity is 1 and that mandatory constraint is some constraint that has some modality where the possible value of that modality is 'Alethic'.

2) In each population of role is mandatory, each role occurs at most once.

b) Role is functional

1) *Role is functional if and only if some uniqueness constraint restricts that role at some ordinal and has some arity where the possible value of that arity is 1 and that uniqueness constraint is some constraint that has some modality where the possible value of that modality is 'Alethic'.

2) In each population of role is functional, each role occurs at most once.

8.11.4 Other constraints

8.11.5 Other considerations

8.11.6 Graphical representation

8.11.6.1 ORM2 representation
8.12 Subtyping

8.12.1 Introduction

8.12.2 Assertions

a) Object type is a direct subtype of object type

1) It is possible that more than one object type is a direct subtype of the same object type and that more than one object type is a direct supertype of the same object type.

2) In each population of object type is a direct subtype of object type, each object type, object type combination occurs at most once.

3) This association with object type, object type provides the preferred identification scheme for subtyping.

4) No object type may cycle back to itself via one or more traversals through object type is a direct subtype of object type.

5) If object type1 is a direct subtype of some object type2 then it is not true that object type1 is indirectly related to object type2 by repeatedly applying this fact type.

8.12.3 Derivations

8.12.4 Other constraints

a) No object type may cycle back to itself via one or more traversals through object type is a direct subtype of object type. If object type1 is a direct subtype of some object type2 then it is not true that object type1 is indirectly related to object type2 by repeatedly applying this fact type.

8.12.5 Other considerations

8.12.6 Graphical representation

8.12.6.1 ORM2 representation

8.13 Subtyping and type consistency]

8.13.1 Introduction

8.13.2 Assertions

8.13.3 Derivations

a) Subtyping applies to entity types
1) *Subtyping applies to entity types if and only if that subtyping involves some subtype object type1 that is some semantic object type1 that is some entity type or that subtyping involves some supertype object type2 that is some semantic object type2 that is some entity type.

2) In each population of subtyping applies to entity types, each subtyping occurs at most once.

b) Subtyping applies to value types

1) *Subtyping applies to value types if and only if that subtyping involves some subtype object type1 that is some semantic object type1 that is some value type or that subtyping involves some supertype object type2 that is some semantic object type2 that is some value type.

2) In each population of subtyping applies to value types, each subtyping occurs at most once.

c) Subtyping applies to data types

1) *Subtyping applies to data types if and only if that subtyping involves some subtype object type1 that is some data type or that subtyping involves some supertype object type2 that is some data type.

2) In each population of subtyping applies to data types, each subtyping occurs at most once.

8.13.4 Other constraints

a) For each subtyping, exactly one of the following holds: that subtyping applies to entity types; that subtyping applies to value types; that subtyping applies to data types.

8.13.5 Other considerations

8.13.6 Graphical representation

8.13.6.1 ORM2 representation

8.14 Subtyping and identity fact type

8.14.1 Introduction

Note Implementation note: It is expected that the IdentityFactType structure may be automatically created by a tool when a Subtyping relationship is specified. Although a formal creation rule describing this operation is beyond the scope of this model, the requirements for such a creation rule can be clearly inferred from the subtype and subset constraints on this diagram.

8.14.2 Assertions

a) Subtyping corresponds to identity fact type
1) Each subtyping corresponds to at most one identity fact type.

2) For each identity fact type, exactly one subtyping corresponds to that identity fact type.

### 8.14.3 Derivations

**a)** Identity fact type has supertype role

1) *Identity fact type has supertype role if and only if some subtyping corresponds to that identity fact type and involves some supertype object type and that identity fact type is some fact type that contains that role that is played by that object type.*

2) Each identity fact type has exactly one supertype role.

3) For each supertype role, at most one identity fact type has that supertype role.

**b)** Identity fact type has subtype role

1) *Identity fact type has subtype role if and only if some subtyping corresponds to that identity fact type and involves some subtype object type and that identity fact type is some fact type that contains that role that is played by that object type.*

2) Each identity fact type has exactly one subtype role.

3) For each subtype role, at most one identity fact type has that subtype role.

### 8.14.4 Other constraints

**a)** For each role, if some identity fact type has that subtype role then that role is functional.

**b)** For each role, if some identity fact type has that supertype role then that role is functional.

**c)** For each role, if some identity fact type has that subtype role then that role is mandatory.

### 8.14.5 Other considerations

**a)** Identity fact type is an entity type.

1) **Reference Scheme:** fact type has UUID.

2) **Reference Mode:** UUID.
8.14.6 Graphical representation

8.14.6.1 ORM2 representation

8.15 Subtyping and compatibility

8.15.1 Introduction

8.15.2 Assertions

8.15.3 Derivations

a) Object type1 is compatible with object type2

1) *Object type1 is compatible with object type2 if and only if that object type1 is a subtype of that object type2 or is a supertype of that object type2 or is a subtype of some object type3 that is a supertype of that object type2 where it is not true that object type2 is a disjoint relative of that object type1 or that object type1 exists where that object type2 exists where object type1 = object type2.

2) It is possible that more than one object type is compatible with the same object type and that the same object type is compatible with more than one object type.

3) In each population of object type is compatible with object type, each object type, object type combination occurs at most once.

b) Object type1 is a subtype of object type2

1) *Object type1 is a subtype of object type2 if and only if that object type1 is a direct subtype of that object type2 or is a direct subtype of some object type3 that is a subtype of that object type2.

2) It is possible that more than one object type is a subtype of the same object type and that more than one object type is a supertype of the same object type.
3) In each population of object type is a subtype of object type, each object type, object type combination occurs at most once.

c) Object type1 is a disjoint relative of object type2

1) *Object type1 is a disjoint relative of object type2 if and only if that object type1 shares supertype with that object type2 and has some ancestor supertype role1 and that object type2 has some ancestor supertype role2 and for some exclusion constraint, that exclusion constraint is some set comparison constraint that has some column arity where the possible value of that arity is 1 and that exclusion constraint is some constraint that has some modality where the possible value of that modality is ‘Alethic’ and that set comparison constraint compares some compared role list occurrence1 that includes that role1 at some ordinal and that set comparison constraint compares some compared role list occurrence2 that includes that role2 at some ordinal where role1 <> role2.

2) It is possible that more than one object type is a disjoint relative of the same object type and that the same object type is a disjoint relative of more than one object type.

3) In each population of object type is a disjoint relative of object type, each object type, object type combination occurs at most once.

d) Object type1 has ancestor supertype role

1) *Object type1 has ancestor supertype role if and only if that object type1 is involved as subtype in some subtyping that corresponds to some identity fact type that has that supertype role or that subtyping involves some supertype object type2 that has that ancestor supertype role.

2) It is possible that more than one object type has the same ancestor supertype role and that the same object type has more than one ancestor supertype role.

3) In each population of object type has ancestor supertype role, each object type, role combination occurs at most once.

e) Role1 is compatible with role2

1) *Role1 is compatible with role2 if and only if that role1 is played by some object type1 and that role2 is played by some object type2 that is compatible with that object type1 where role1 <> role2.

2) It is possible that more than one role is compatible with the same role and that the same role is compatible with more than one role.

3) In each population of role is compatible with role, each role, role combination occurs at most once.

f) Object type1 shares supertype with object type2

1) *Object type1 shares supertype with object type2 if and only if some object type3 is a supertype of that object type1 and is a supertype of that object type2 and it is not true that that object type2 is a subtype of that object type1 and it is not true that that object type2 is a supertype of that object type1.

2) It is possible that more than one object type shares supertype with the same object type and that the same object type shares supertype with more than one object type.

3) In each population of object type shares supertype with object type, each object type, object type combination occurs at most once.

g) Object type1 has direct supertype object type2 distinct from other direct supertype object type3
1) *Object type1 has direct supertype object type2 distinct from other direct supertype object type3 if and only if that object type1 is a direct subtype of that object type2 and is a direct subtype of that object type3 where object type2 <> object type3.

2) It is possible that more than one object type1 has the same direct supertype object type2 distinct from the same other direct supertype object type3 and that the same object type1 has more than one direct supertype object type2 distinct from the same other direct supertype object type3 and that the same object type1 has the same direct supertype object type2 distinct from more than one other direct supertype object type3.

3) In each population of object type1 has direct supertype object type2 distinct from other direct supertype object type3, each object type1, object type2, object type3 combination occurs at most once.

8.15.4 Other constraints

a) If some object type has some direct supertype object type1 distinct from some other direct supertype object type2 then that object type1 is compatible with that object type2.

8.15.5 Other considerations

8.15.6 Graphical representation

8.15.6.1 ORM2 representation

8.16 Semantic object type identification

8.16.1 Introduction

8.16.2 Assertions

a) Entity type is directly identified by uniqueness constraint

1) Each entity type is directly identified by at most one uniqueness constraint.

2) Each uniqueness constraint directly identifies at most one entity type.

b) Subtyping provides identification path

1) In each population of subtyping provides identification path, each subtyping occurs at most once.
8.16.3 Derivations

a) Entity type is indirectly identified through subtyping

1) *Entity type is indirectly identified through subtyping if and only if that entity type is some object type1 that is a direct subtype of some object type and is involved as subtype in that subtyping that provides identification path.

2) Each subtyping provides identification path for at most one entity type.

3) It is possible that the same entity type is indirectly identified through more than one subtyping.

b) Entity type1 has identifying uniqueness constraint

1) *Entity type1 has identifying uniqueness constraint if and only if that entity type1 is directly identified by that uniqueness constraint or is indirectly identified through some subtyping that involves some supertype object type that is some entity type2 that has that identifying uniqueness constraint.

2) Each entity type has exactly one identifying uniqueness constraint.

3) It is possible that the same uniqueness constraint identifies more than one entity type.

c) Semantic object type1 has a simple identifier based on value type1

1) *Semantic object type1 has a simple identifier based on value type1 if and only if that value type1 is that semantic object type1 that is some value type or that semantic object type1 is some entity type that has some identifying uniqueness constraint that has some arity where the possible value of that arity is 1 and restricts some role at some ordinal where that role is played by some object type that is some semantic object type2 that has a simple identifier based on that value type1.

2) Each semantic object type has a simple identifier based on at most one value type.

3) It is possible that more than one semantic object type has a simple identifier based on the same value type.

8.16.4 Other constraints

a) For each entity type, exactly one of the following holds: that entity type is directly identified by some uniqueness constraint; that entity type is indirectly identified through some subtyping.
8.16.5 Other considerations

8.16.6 Graphical representation

8.16.6.1 ORM2 representation

8.17 Semantic object type independent

8.17.1 Introduction

Note  Note that each element in a DataType is inherently independent, so we do not record independence for DataType.

8.17.2 Assertions

a) Semantic object type is independent

1) In each population of semantic object type is independent, each semantic object type occurs at most once.

8.17.3 Derivations

a) Semantic object type can be independent

1) *Semantic object type can be independent if and only if it is not true that that semantic object type is some object type that plays some role1 that is restricted by some mandatory constraint that is some constraint that has some modality where the possible value of that modality is 'Alethic' and that role1 belongs to no fact type that contains some role2 where some uniqueness constraint restricts that role2 at some ordinal and identifies some entity type that is that semantic object type.

2) In each population of semantic object type can be independent, each semantic object type occurs at most once.

8.17.4 Other constraints

a) If some semantic object type is independent then that semantic object type can be independent.
8.18 Derivation

8.18.1 Introduction

Informal Description: DerivationEvaluation determines when derived instances are calculated:

- Immediate evaluation calculates and stores derived instances immediately when data changes.
- OnDemand evaluation recalculates derived instances for every request.

Note The structure of the DerivationRule is out of scope for this metamodel version.

8.18.2 Assertions

a) Derivation rule has derivation evaluation

1) Each derivation rule has at most one derivation evaluation.

2) It is possible that more than one derivation rule has the same derivation evaluation.

b) Derived or semiderived fact type is derived using derivation rule

1) It is possible that more than one derived or semiderived fact type is derived using the same derivation rule and that more than one derivation rule derives the same derived or semiderived fact type.

2) In each population of derived or semiderived fact type is derived using derivation rule, each derived or semiderived fact type, derivation rule combination occurs at most once.

3) This association with derived or semiderived fact type, derivation rule provides the preferred identification scheme for fact type derivation.

4) Each derived or semiderived fact type is derived using some derivation rule.

c) Fact type has instantiation method

1) Each fact type has exactly one instantiation method.

2) It is possible that more than one fact type has the same instantiation method.

d) Subtype has instantiation method
1) Each subtype has exactly one instantiation method.

2) It is possible that more than one subtype has the same instantiation method.

e) Derived or semiderived subtype is derived using derivation rule

1) It is possible that more than one derived or semiderived subtype is derived using the same derivation rule and that more than one derivation rule derives the same derived or semiderived subtype.

2) In each population of derived or semiderived subtype is derived using derivation rule, each derived or semiderived subtype, derivation rule combination occurs at most once.

3) This association with derived or semiderived subtype, derivation rule provides the preferred identification scheme for subtype derivation.

4) Each derived or semiderived subtype is derived using some derivation rule.

f) Derivation rule has derivation description

1) Each derivation rule has at most one derivation description.

2) It is possible that more than one derivation rule has the same derivation description.

8.18.3 Derivations

8.18.4 Other constraints

a) Each derivation rule derives some derived or semiderived fact type or derives some derived or semiderived subtype.

8.18.5 Other considerations

a) Derived or semiderived fact type is an entity type.

1) Each derived or semiderived fact type is some fact type that has some instantiation method where the possible values of that instantiation method are 'Derived', 'Semiderived'.

2) Reference Scheme: fact type has UUID.

3) Reference Mode: UUID.

b) Derivation evaluation is an entity type.

1) Reference Scheme: derivation evaluation has derivation evaluation_name.

2) Reference Mode: .name.

3) The possible values of derivation evaluation are 'Immediate', 'OnDemand'.

c) Derivation rule is an entity type.

1) Reference Scheme: derivation rule has UUID.

2) Reference Mode: UUID.

d) Instantiation method is an entity type.
1) Reference Scheme: instantiation method has instantiation method_name.
2) Reference Mode: .name.
3) The possible values of instantiation method are 'Asserted', 'Derived', 'Semiderived'.

c) Derived or semiderived subtype is an entity type.
1) *Each derived or semiderived subtype is some subtype that has some instantiation method where the possible values of that instantiation method are 'Derived', 'Semiderived'.
2) Reference Scheme: object type has UUID.
3) Reference Mode: UUID.

f) Subtype is an entity type.
1) *Each subtype is some object type that is a direct subtype of some object type.
2) Reference Scheme: object type has UUID.
3) Reference Mode: UUID.

g) Derivation description is a value type.
1) Portable data type: Text: Variable Length.

8.18.6 Graphical representation

8.18.6.1 ORM2 representation

8.19 Data type

8.19.1 Introduction

Note TBD: A list of standard data type names based on, e.g., ISO 11179-3 or ISO 9075-2.
8.19.2 Assertions

a) Data type has standard data type name
   1) Each data type has at most one standard data type name.
   2) For each standard data type name, at most one data type has that standard data type name.

8.19.3 Derivations

a) Data type is topmost
   1) * Data type is topmost if and only if it is not true that data type is some object type1 that is a direct subtype of some object type.
   2) In each population of data type is topmost, each data type occurs at most once.

b) Value type uses values from data type1
   1) * Value type uses values from data type1 if and only if that value type draws values from that data type1 or draws values from some data type2 that is some object type1 that is a subtype of some object type2 that is that data type1.
   2) It is possible that more than one value type uses values from the same data type and that the same value type uses values from more than one data type.
   3) In each population of value type uses values from data type, each value type, data type combination occurs at most once.

c) Object type1 can restrict values
   1) * Object type1 can restrict values if and only if that object type1 is some data type and is a direct subtype of some object type or that object type1 is some semantic object type that has a simple identifier based on some value type.
   2) In each population of object type can restrict values, each object type occurs at most once.

d) Object type1 uses values from data type1
   1) * Object type1 uses values from data type1 if and only if that object type1 is some semantic object type that has a simple identifier based on some value type that uses values from that data type1 or that object type1 is some data type and is a subtype of some object type2 that is that data type1.
   2) It is possible that more than one object type uses values from the same data type and that the same object type uses values from more than one data type.
   3) In each population of object type uses values from data type, each object type, data type combination occurs at most once.

8.19.4 Other constraints

a) If some data type is topmost then that data type has some standard data type name.

8.19.5 Other considerations

a) Standard data type name is a value type.
8.20 Facet

8.20.1 Introduction

**Informal Description:** The fact restriction style defines how instances of the facet can be used to form a more restrictive set.

Note that 'FacetRestriction is fixed' can also fix the value(s) of that facet.

- **FixedValue** indicates that a facet instance has a single value that cannot be changed by a further restriction of the data type.

- **DecreasingSet** indicates that the facet can have multiple discrete values and that a further restriction of the data type can reduce the number of values in the set.

- **AscendingRestriction** indicates that the facet has a single value and larger values are more restrictive.

The facet must draw values from a comparable data type to use this style.

- **DescendingRestriction** indicates that the facet has a single value and smaller values are more restrictive.

The facet must draw values from a comparable data type to use this style.

Note: The fact restriction style defines how instances of the facet can be used to form a more restrictive set. Note that 'FacetRestriction is fixed' can also fix the value(s) of that facet, preventing more restrictive types from adjusting its value. - FixedValue indicates that a facet instance has a single value that cannot be changed by a further restriction of the data type. - DecreasingSet indicates that the facet can have multiple discrete values and that a further restriction of the data type can reduce the number of values in the set. - AscendingRestriction indicates that the facet has a single value and larger values are more restrictive. The facet must draw values from a comparable data type to use this style. - DescendingRestriction indicates that the facet has a single value and smaller values are more restrictive. The facet must draw values from a comparable data type to use this style.

Note: TBD: Add restrictions to limit facets to standard data types for v1, and provide a populated model containing the standard data types and corresponding facets.

8.20.2 Assertions

a) **Facet draws values from data type**
1) Each facet draws values from exactly one data type.

2) It is possible that more than one facet draws values from the same data type.

b) Object type restricts facet

1) It is possible that more than one object type restricts the same facet and that the same object type restricts more than one facet.

2) In each population of object type restricts facet, each object type, facet combination occurs at most once.

3) This association with object type, facet provides the preferred identification scheme for facet restriction.

c) Facet has default value instance

1) Each facet has at most one default value instance.

2) It is possible that more than one facet has the same default value instance.

d) Facet has facet name

1) Each facet has exactly one facet name.

2) It is possible that more than one facet has the same facet name.

e) Facet requires restriction

1) In each population of facet requires restriction, each facet occurs at most once.

f) Facet has facet restriction style

1) Each facet has exactly one facet restriction style.

2) It is possible that more than one facet has the same facet restriction style.

g) Facet restriction has value instance

1) It is possible that more than one facet restriction has the same value instance and that the same facet restriction has more than one value instance.

2) In each population of facet restriction has value instance, each facet restriction, value instance combination occurs at most once.

3) This association with facet restriction, value instance provides the preferred identification scheme for facet restriction has value instance.

4) Each facet restriction has some value instance.

h) Facet restriction is fixed

1) In each population of facet restriction is fixed, each facet restriction occurs at most once.

i) Data type defines facet

1) For each facet, exactly one data type defines that facet.
2) It is possible that the same data type defines more than one facet.

8.20.3 Derivations

8.20.4 Other constraints

a) If some object type restricts some facet then that object type uses values from some data type that defines that facet.

b) For each facet, at most one of the following holds: that facet has some default value instance; that facet requires restriction.

c) If some object type restricts some facet then that object type can restrict values.

8.20.5 Other considerations

a) Facet name is a value type.

   1) Portable data type: Text: Variable Length.

b) Facet restriction style is an entity type.

   1) Reference Scheme: facet restriction style has facet restriction style_name.

   2) Reference Mode: .name.

   3) The possible values of facet restriction style are FixedValue, DecreasingSet, AscendingRestriction, DescendingRestriction.

c) Facet is an entity type.

   1) Reference Scheme: facet has UUID.

   2) Reference Mode: UUID.

d) Context: for some facet, some data type defines that facet and that facet has some facet name.

   1) In this context, each data type, facet name combination is associated with at most one facet.
8.20.6 Graphical representation

8.20.6.1 ORM2 representation

8.21 Calculated fact type

8.21.1 Introduction

8.21.2 Assertions

8.21.3 Derivations

a) Fact type is calculated

1) *Fact type is calculated if and only if that fact type contains some role that is played by some object type that is some data type.*

2) In each population of fact type is calculated, each fact type occurs at most once.

b) Fact type is semantic

1) *Fact type is semantic if and only if that fact type contains some role that is played by some object type that is some semantic object type.*

2) In each population of fact type is semantic, each fact type occurs at most once.

8.21.4 Other constraints

a) For each fact type, exactly one of the following holds: that fact type is calculated; that fact type is semantic.
8.22 Comparison operation

8.22.1 Introduction

8.22.2 Assertions

a) Comparison operation uses standard comparator
   1) Each comparison operation uses at most one standard comparator.
   2) It is possible that more than one comparison operation uses the same standard comparator.

b) Fact type is comparison operation
   1) In each population of fact type is comparison operation, each fact type occurs at most once.

8.22.3 Derivations

a) Data type is comparable
   1) *Data type is comparable if and only if some comparison operation compares values for that data type and uses some standard comparator where the possible values of that standard comparator are 'LessThan', 'GreaterThan'.
   2) In each population of data type is comparable, each data type occurs at most once.

b) Data type is equatable
   1) *Data type is equatable if and only if some comparison operation compares values for that data type and uses some standard comparator where the possible values of that standard comparator are 'Equals', 'NotEquals'.
   2) In each population of data type is equatable, each data type occurs at most once.

c) Fact type can be comparison operation
   1) *Fact type can be comparison operation if and only if that fact type is calculated and is some binary fact type and has some spanning uniqueness constraint and contains some role1 that is played by some object type and that fact type contains some role2 that is played by that object type where role1 <> role2.
   2) In each population of fact type can be comparison operation, each fact type occurs at most once.
d) **Comparison operation compares values for data type1**

1) *Comparison operation compares values for data type1 if and only if that comparison operation is defined on that data type1 or that data type1 is some object type1 that is a subtype of some object type2 where that comparison operation is defined on some data type2 that is that object type2.*

2) **It is possible that more than one comparison operation compares values for the same data type and that the same comparison operation compares values for more than one data type.**

3) **In each population of comparison operation compares values for data type, each comparison operation, data type combination occurs at most once.**

4) **Each comparison operation compares values for some data type.**

e) **Comparison operation is defined on data type**

1) *Comparison operation is defined on data type if and only if that comparison operation is some fact type that contains some role that is played by some object type that is that data type.*

2) **Each comparison operation is defined on exactly one data type.**

3) **It is possible that more than one comparison operation is defined on the same data type.**

**8.22.4 Other constraints**

a) **If some fact type is comparison operation then that fact type can be comparison operation.**

**8.22.5 Other considerations**

a) **Standard comparator is an entity type.**

1) **Reference Scheme:** standard comparator has standard comparator_name.

2) **Reference Mode:** .name.

3) **The possible values of standard comparator are 'Equals', 'NotEquals', 'LessThan', 'LessThanOrEqual', 'GreaterThanOrEqual', 'GreaterThan'.**

b) **Comparison operation is an entity type.**

1) *Each comparison operation is some fact type that is comparison operation.*

2) **Reference Scheme:** fact type has UUID.

3) **Reference Mode:** UUID.
8.22.6 Graphical representation

8.22.6.1 ORM2 representation

8.23 Value type comparable

8.23.1 Introduction

8.23.2 Assertions

8.23.3 Derivations

a) Value type1 is comparable to value type2

1) *Value type1 is comparable to value type2 if and only if for some data type, that value type1 uses values from that data type and that value type2 uses values from that data type and that data type is comparable.

2) It is possible that more than one value type is comparable to the same value type and that the same value type is comparable to more than one value type.

3) In each population of value type is comparable to value type, each value type, value type combination occurs at most once.

4) If value type1 is comparable to value type2 then value type2 is comparable to value type1.

b) Semantic object type1 is value comparable with semantic object type2

1) *Semantic object type1 is value comparable with semantic object type2 if and only if that semantic object type1 has a simple identifier based on some value type1 and that semantic object type2 has a simple identifier based on some value type2 that is comparable to that value type1.

2) It is possible that more than one semantic object type is value comparable with the same semantic object type and that the same semantic object type is value comparable with more than one semantic object type.
3) In each population of semantic object type is value comparable with semantic object type, each semantic object type combination occurs at most once.

c) Role1 is value comparable with role2

1) *Role1 is value comparable with role2 if and only if that role1 is played by some object type1 that is some semantic object type1 and that role2 is played by some object type2 that is some semantic object type2 that is value comparable with that semantic object type1 where role1 <> role2.

2) It is possible that more than one role is value comparable with the same role and that the same role is value comparable with more than one role.

3) In each population of role is value comparable with role, each role, role combination occurs at most once.

8.23.4 Other constraints

a) If value type1 is comparable to value type2 then value type2 is comparable to value type1.

8.23.5 Other considerations

8.23.6 Graphical representation

8.23.6.1 ORM2 representation

8.24 Objectification

8.24.1 Introduction

8.24.2 Assertions

a) Fact type is objectified by semantic object type

1) Each fact type is objectified by at most one semantic object type.

2) Each semantic object type objectifies at most one fact type.

b) Fact type contains role

1) Each fact type contains some role.

2) Each role belongs to exactly one fact type.

3) It is possible that the same fact type contains more than one role.
8.24.3 Derivations

8.24.4 Other constraints

a) For each fact type and object type, at most one of the following holds: that fact type is objectified by some semantic object type that is that object type; that fact type contains some role that is played by that object type.

8.24.5 Other considerations

8.24.6 Graphical representation

8.24.6.1 ORM2 representation

8.25 Objectification and link fact type

8.25.1 Introduction

Note This model is currently silent on the constraints associated with a mirror role. The associated discussion is placed outside this model. Similarly, no fact type is provided for implied objectification. Objectification may be implied for all fact types that can be asserted that are either n-ary or binary with a spanning uniqueness constraint.

8.25.2 Assertions

a) Role implies link binary fact type

1) Each role implies at most one link binary fact type.

2) Each link binary fact type is implied by at most one role.

8.25.3 Derivations

a) Role1 has objectifying role2 in link fact type

1) *Role1 has objectifying role2 in link fact type if and only if that role1 implies some link binary fact type that is some fact type1 that contains that role2 that is played by some object type that is some semantic object type that objectifies some fact type2 that contains that role1.

2) Each role has at most one objectifying role in link fact type.

3) For each objectifying role, at most one role has that objectifying role in link fact type.

b) Role1 has mirror role2 in link fact type

1) *Role1 has mirror role2 in link fact type if and only if that role1 is played by some object type and implies some link binary fact type that is some fact type that contains that role2 that is played by that object type.
2) For each mirror role, at most one role has that mirror role in link fact type.

3) Each role has at most one mirror role in link fact type.

c) Role is in objectified fact type

1) *Role is in objectified fact type if and only if that role belongs to some fact type that is objectified by some semantic object type.

2) In each population of role is in objectified fact type, each role occurs at most once.

8.25.4 Other constraints

a) For each role, if some role has that objectifying role in link fact type then that role is functional.

b) If some role implies some link binary fact type then that role is in objectified fact type.

c) For each fact type and role, at most one of the following holds: that fact type contains that role; that role implies some link binary fact type that is that fact type.

d) For each role, if some role has that objectifying role in link fact type then that role is mandatory.

e) For each role, that role has some mirror role in link fact type if and only if that role implies some link binary fact type.

f) For each role, that role has some objectifying role in link fact type if and only if that role implies some link binary fact type.

8.25.5 Other considerations

8.25.6 Graphical representation

8.25.6.1 ORM2 representation

8.26 Constraint

8.26.1 Introduction

8.26.2 Assertions

a) Constraint has constraint name
1) Each constraint has at most one constraint name.

2) Each constraint name is of at most one constraint.

b) Constraint has modality

1) Each constraint has exactly one modality.

2) It is possible that more than one constraint has the same modality.

8.26.3 Derivations

8.26.4 Other constraints

a) For each set comparison constraint, exactly one of the following holds: that set comparison constraint is some equality constraint; that set comparison constraint is some exclusion constraint; that set comparison constraint is some subset constraint.

b) For each constraint, exactly one of the following holds: that constraint is some mandatory constraint; that constraint is some uniqueness constraint; that constraint is some frequency constraint; that constraint is some ring constraint; that constraint is some value comparison constraint; that constraint is some set comparison constraint; that constraint is some value constraint; that constraint is some cardinality constraint; that constraint is some ad hoc constraint.

8.26.5 Other considerations

a) Constraint is an entity type.

1) Reference Scheme: constraint has UUID.

2) Reference Mode: UUID.

b) Constraint name is a value type.

1) Portable data type: Text: Variable Length.

c) Modality is an entity type.

1) Reference Scheme: modality has modality_name.

2) Reference Mode: .name.

3) The possible values of modality are 'Alethic', 'Deontic'.

d) Frequency constraint is an entity type.

1) Reference Scheme: constraint has UUID.

2) Reference Mode: UUID.

e) Ring constraint is an entity type.

1) Reference Scheme: constraint has UUID.

2) Reference Mode: UUID.
f) Value comparison constraint is an entity type.

1) Reference Scheme: constraint has UUID.

2) Reference Mode: UUID.

8.26.6 Graphical representation

8.26.6.1 ORM2 representation

8.27 Mandatory

8.27.1 Introduction

8.27.2 Assertions

a) Mandatory constraint restricts role

1) It is possible that more than one mandatory constraint restricts the same role and that more than one role is restricted by the same mandatory constraint.

2) In each population of mandatory constraint restricts role, each mandatory constraint, role combination occurs at most once.

3) This association with mandatory constraint, role provides the preferred identification scheme for mandatory constrained role.

4) Each mandatory constraint restricts some role.

8.27.3 Derivations

a) Mandatory constraint is a simple mandatory constraint

1) *Mandatory constraint is a simple mandatory constraint if and only if that mandatory constraint has some arity where the possible value of that arity is 1.

2) In each population of mandatory constraint is a simple mandatory constraint, each mandatory constraint occurs at most once.
b) Mandatory constraint is an inclusive-or constraint

1) Mandatory constraint is an inclusive-or constraint if and only if that mandatory constraint has some arity where the possible values of that arity are at least 2.

2) In each population of mandatory constraint is an inclusive-or constraint, each mandatory constraint occurs at most once.

c) Role1 is in inclusive-or constraint with role2

1) Role1 is in inclusive-or constraint with role2 if and only if some mandatory constraint restricts that role1 and restricts that role2 where role1 <> role2.

2) It is possible that more than one role is in inclusive-or constraint with the same role and that the same role is in inclusive-or constraint with more than one role.

3) In each population of role is in inclusive-or constraint with role, each role, role combination occurs at most once.

d) Mandatory constraint has arity

1) Mandatory constraint has arity if and only if that mandatory constraint restricts some role where arity = count(each role for that mandatory constraint).

2) Each mandatory constraint has exactly one arity.

3) It is possible that more than one mandatory constraint has the same arity.

e) Role1 is compatible with role2

1) Role1 is compatible with role2 if and only if that role1 is played by some object type1 and that role2 is played by some object type2 that is compatible with that object type1 where role1 <> role2.

2) It is possible that more than one role is compatible with the same role and that the same role is compatible with more than one role.

3) In each population of role is compatible with role, each role, role combination occurs at most once.

8.27.4 Other constraints

a) For each mandatory constraint, exactly one of the following holds: that mandatory constraint is a simple mandatory constraint; that mandatory constraint is an inclusive-or constraint.

b) If some role1 is in inclusive-or constraint with some role2 then that role1 is compatible with that role2.

8.27.5 Other considerations

a) Mandatory constraint is an entity type.

1) Reference Scheme: constraint has UUID.

2) Reference Mode: UUID.
8.27.6 Graphical representation

8.27.6.1 ORM2 representation

8.28 Exclusive or

8.28.1 Introduction

8.28.2 Assertions

a) Mandatory constraint forms exclusive-or constraint with exclusion constraint

1) Each mandatory constraint forms exclusive-or constraint with at most one exclusion constraint.

2) Each exclusion constraint forms exclusive-or constraint with at most one mandatory constraint.

8.28.3 Derivations

a) Mandatory constraint can form exclusive-or constraint with exclusion constraint

1) *Mandatory constraint can form exclusive-or constraint with exclusion constraint if and only if that mandatory constraint has some arity1 and that mandatory constraint is some constraint1 that has some modality and that exclusion constraint is some set comparison constraint that has that row arity1 and has some column arity2 where the possible value of that arity2 is 1 and compares no compared role list occurrence that includes some role at some ordinal where it is not true that that role is restricted by that mandatory constraint and that exclusion constraint is some constraint2 that has that modality.

2) Each mandatory constraint can form exclusive-or constraint with at most one exclusion constraint.

3) Each exclusion constraint can form exclusive-or constraint with at most one mandatory constraint.

8.28.4 Other constraints

a) If some mandatory constraint forms exclusive-or constraint with some exclusion constraint then that mandatory constraint can form exclusive-or constraint with that exclusion constraint.

8.28.5 Other considerations

a) Exclusion constraint is an entity type.
1) Reference Scheme: constraint has UUID.
2) Reference Mode: UUID.

b) Mandatory constraint is an entity type.
   1) Reference Scheme: constraint has UUID.
   2) Reference Mode: UUID.

8.28.6 Graphical representation

8.28.6.1 ORM2 representation

8.29 Uniqueness

8.29.1 Introduction

8.29.2 Assertions

a) Uniqueness constraint restricts role at ordinal

1) Each uniqueness constraint restricts some role at some ordinal.
2) For each uniqueness constraint and role, that uniqueness constraint restricts that role at at most one ordinal.
3) For each uniqueness constraint and ordinal, that uniqueness constraint restricts at most one role at that ordinal.
4) This association with uniqueness constraint, ordinal provides the preferred identification scheme for uniquely constrained role.

8.29.3 Derivations

a) Uniqueness constraint has arity

1) *Uniqueness constraint has arity if and only if that uniqueness constraint restricts some role at some ordinal where arity = count(each role for that uniqueness constraint).*
2) Each uniqueness constraint has exactly one arity.
3) It is possible that more than one uniqueness constraint has the same arity.

b) Uniqueness constraint is internal
1) *Uniqueness constraint is internal if and only if that uniqueness constraint restricts some role at some ordinal where that role belongs to some fact type and it is not true that that uniqueness constraint restricts some role at some ordinal where that role belongs to some fact type where fact type <> fact type.

2) In each population of uniqueness constraint is internal, each uniqueness constraint occurs at most once.

c) Fact type has spanning uniqueness constraint

1) *Fact type has spanning uniqueness constraint if and only if that uniqueness constraint is internal and has some arity and restricts some role at some ordinal where that role belongs to that fact type that has that arity.

2) Each fact type has at most one spanning uniqueness constraint.

3) Each uniqueness constraint spans at most one fact type.

8.29.4 Other constraints

8.29.5 Other considerations

a) Uniqueness constraint is an entity type.

1) Reference Scheme: constraint has UUID.

2) Reference Mode: UUID.

8.29.6 Graphical representation

8.29.6.1 ORM2 representation

8.30 Frequency

8.30.1 Introduction

8.30.2 Assertions

a) Frequency constraint allows frequency range

1) Each frequency range is allowed in exactly one frequency constraint.
2) It is possible that the same frequency constraint allows more than one frequency range.

b) Frequency constraint allows discrete frequency bound

1) It is possible that more than one frequency constraint allows the same discrete frequency bound and that the same frequency constraint allows more than one discrete frequency bound.

2) In each population of frequency constraint allows discrete frequency bound, each frequency constraint, frequency bound combination occurs at most once.

3) This association with frequency constraint, frequency bound provides the preferred identification scheme for frequency constraint discrete bound.

c) Frequency range has minimum frequency bound

1) Each frequency range has at most one minimum frequency bound.

2) It is possible that more than one frequency range has the same minimum frequency bound.

d) Frequency range has maximum frequency bound

1) The possible values of maximum frequency bound in frequency range has maximum frequency bound are at least 2.

2) Each frequency range has at most one maximum frequency bound.

3) It is possible that more than one frequency range has the same maximum frequency bound.

e) Frequency constraint restricts role

1) It is possible that more than one frequency constraint restricts the same role and that more than one role is restricted by the same frequency constraint.

2) In each population of frequency constraint restricts role, each frequency constraint, role combination occurs at most once.

3) This association with frequency constraint, role provides the preferred identification scheme for frequency constrained role.

4) Each frequency constraint restricts some role.

8.30.3 Derivations

a) Frequency constraint has arity

1) Frequency constraint has arity if and only if that frequency constraint restricts some role where arity = count(each role for that frequency constraint).

2) Each frequency constraint has exactly one arity.

3) It is possible that more than one frequency constraint has the same arity.

8.30.4 Other constraints

a) Each frequency range has some minimum frequency bound or has some maximum frequency bound.
b) Each frequency constraint allows some frequency range or allows some discrete frequency bound.

c) For each frequency constraint, minimum frequency bound1, and maximum frequency bound2, at most one frequency range is allowed in that frequency constraint and has that minimum frequency bound1 and has that maximum frequency bound2. This association with frequency constraint, frequency bound1, and frequency bound2 provides the preferred identification scheme for frequency range.

8.30.5 Other considerations

a) Frequency range is an entity type.

1) Reference Scheme: frequency constraint allows frequency range; frequency range has minimum frequency bound; frequency range has maximum frequency bound.

b) Frequency constraint is an entity type.

1) Reference Scheme: constraint has UUID.

2) Reference Mode: UUID.

c) Frequency bound is a value type.

1) Portable data type: Numeric: Signed Integer.

2) The possible values of frequency bound are at least 1.

8.30.6 Graphical representation

8.30.6.1 ORM2 representation
8.31 Ring

8.31.1 Introduction

8.31.2 Assertions

a) Ring constraint restricts first role
   1) Each ring constraint restricts exactly one first role.
   2) It is possible that more than one ring constraint restricts the same first role.

b) Ring constraint restricts second role
   1) Each ring constraint restricts exactly one second role.
   2) It is possible that more than one ring constraint restricts the same second role.

c) Ring constraint is irreflexive
   1) In each population of ring constraint is irreflexive, each ring constraint occurs at most once.

d) Ring constraint is antisymmetric
   1) In each population of ring constraint is antisymmetric, each ring constraint occurs at most once.

e) Ring constraint is asymmetric
   1) In each population of ring constraint is asymmetric, each ring constraint occurs at most once.

f) Ring constraint is intransitive
   1) In each population of ring constraint is intransitive, each ring constraint occurs at most once.

g) Ring constraint is strongly intransitive
   1) In each population of ring constraint is strongly intransitive, each ring constraint occurs at most once.

h) Ring constraint is acyclic
   1) In each population of ring constraint is acyclic, each ring constraint occurs at most once.

i) Ring constraint is purely reflexive
   1) In each population of ring constraint is purely reflexive, each ring constraint occurs at most once.

j) Ring constraint is reflexive
   1) In each population of ring constraint is reflexive, each ring constraint occurs at most once.

k) Ring constraint is symmetric
   1) In each population of ring constraint is symmetric, each ring constraint occurs at most once.

l) Ring constraint is transitive
1) In each population of ring constraint is transitive, each ring constraint occurs at most once.

8.31.3 Derivations

8.31.4 Other constraints

a) For each ring constraint and role, at most one of the following holds: that ring constraint restricts that first role; that ring constraint restricts that second role.

b) If for some role1, some ring constraint restricts that first role1 and that ring constraint restricts some second role2 then that role1 is compatible with that role2.

c) Each ring constraint is irreflexive or is asymmetric or is asymmetric or is transitive or is strongly intransitive or is acyclic or is purely reflexive or is reflexive or is symmetric or is intransitive.

8.31.5 Other considerations

a) Ring constraint is an entity type.

1) Reference Scheme: constraint has UUID.

2) Reference Mode: UUID.

8.31.6 Graphical representation

8.31.6.1 ORM2 representation

8.32 Ring combination

8.32.1 Introduction

Note Ring Combination Note The exclusions and implications on this page allow 16 valid combinations in addition to the 10 single ring types. The legal combinations are: reflexive+symmetric reflexive+antisymmetric reflexive+transitive reflexive+transitive+antisymmetric symmetric+transitive symmetric+irreflexive symmetric+transitive symmetric+strongly intransitive symmetric+transitive asymmetric+intransitive asymmetric+strongly intransitive transitive+antisymmetric transitive+antisymmetric acyclic+transitive acyclic+transitive acyclic+transitive acyclic+strongly intransitive Additional implications not explicitly shown are: acyclic=>asymmetric purely reflexive=>symmetric irreflexive+antisymmetric=>asymmetric An exclusion is provided for the latter case to force use of asymmetric instead of both irreflexive and antisymmetric.
8.32.2 Assertions

a) Ring constraint is irreflexive

   1) In each population of ring constraint is irreflexive, each ring constraint occurs at most once.

8.32.3 Derivations

a) Ring constraint is implicitly reflexive

   1) *Ring constraint is implicitly reflexive if and only if that ring constraint is purely reflexive or is symmetric and is transitive.

   2) In each population of ring constraint is implicitly reflexive, each ring constraint occurs at most once.

b) Ring constraint is implicitly irreflexive

   1) *Ring constraint is implicitly irreflexive if and only if that ring constraint is asymmetric or is intransitive or is strongly intransitive or is acyclic.

   2) In each population of ring constraint is implicitly irreflexive, each ring constraint occurs at most once.

c) Ring constraint is implicitly antisymmetric

   1) *Ring constraint is implicitly antisymmetric if and only if that ring constraint is asymmetric or is acyclic.

   2) In each population of ring constraint is implicitly antisymmetric, each ring constraint occurs at most once.

8.32.4 Other constraints

a) For each ring constraint, at most one of the following holds: that ring constraint is antisymmetric; that ring constraint is implicitly antisymmetric; that ring constraint is symmetric; that ring constraint is purely reflexive.

b) For each ring constraint, at most one of the following holds: that ring constraint is irreflexive; that ring constraint is antisymmetric.

c) For each ring constraint, at most one of the following holds: that ring constraint is transitive; that ring constraint is purely reflexive.

d) For each ring constraint, at most one of the following holds: that ring constraint is irreflexive; that ring constraint is implicitly irreflexive; that ring constraint is reflexive; that ring constraint is implicitly reflexive.

e) For each ring constraint, at most one of the following holds: that ring constraint is asymmetric; that ring constraint is symmetric; that ring constraint is purely reflexive.

f) For each ring constraint, at most one of the following holds: that ring constraint is intransitive; that ring constraint is strongly intransitive; that ring constraint is transitive.

8.32.5 Other considerations

a) Ring constraint is an entity type.

   1) Reference Scheme: constraint has UUID.

   2) Reference Mode: UUID.
8.32.6 Graphical representation

8.32.6.1 ORM2 representation

8.33 Value comparison

8.33.1 Introduction

8.33.2 Assertions

a) Value comparison constraint compares values with standard comparator

1) Each value comparison constraint compares values with exactly one standard comparator.

2) It is possible that more than one value comparison constraint compares values with the same standard comparator.

b) Value comparison constraint compares first role

1) Each value comparison constraint compares exactly one first role.

2) It is possible that more than one value comparison constraint compares the same first role.

c) Value comparison constraint compares second role

1) Each value comparison constraint compares exactly one second role.

2) It is possible that more than one value comparison constraint compares the same second role.

8.33.3 Derivations

8.33.4 Other constraints

a) If for some role1, some value comparison constraint compares that first role1 and that value comparison constraint compares some second role2 then that role1 is value comparable with that role2.
b) For each value comparison constraint and role, at most one of the following holds: that value comparison constraint compares that first role; that value comparison constraint compares that second role.

c) It is possible that more than one role is value comparable with the same role and that the same role is value comparable with more than one role. In each population of role is value comparable with role, each role, role combination occurs at most once.

8.33.5 Other considerations

a) Value comparison constraint is an entity type.
   1) Reference Scheme: constraint has UUID.
   2) Reference Mode: UUID.

8.33.6 Graphical representation

8.33.6.1 ORM2 representation

8.34 Set comparison

8.34.1 Introduction

Note: Undisplayed Constraints: (ComparisonRoleListOccurrenceHasExactlyOneOwner Exclusive Or Constraint) For each ComparisonRoleListOccurrence, exactly one of the following holds: that ComparisonRoleListOccurrence is compared by some EqualityConstraint; that ComparisonRoleListOccurrence is compared by some ExclusionConstraint; that ComparisonRoleListOccurrence is superset for some SubsetConstraint; that ComparisonRoleListOccurrence is subset for some SubsetConstraint.

8.34.2 Assertions

a) Compared role list occurrence includes role at ordinal
   1) For each compared role list occurrence and role, that compared role list occurrence includes that role at most one ordinal.
   2) Each compared role list occurrence includes some role at some ordinal.
   3) For each compared role list occurrence and ordinal, that compared role list occurrence includes at most one role at that ordinal.
   4) This association with compared role list occurrence, ordinal provides the preferred identification scheme for comparison constrained role.

8.34.3 Derivations

a) Role1 is in set comparison constraint column with role2
1) *Role1 is in set comparison constraint column with role2 if and only if that role1 is included in some compared role list occurrence1 at some ordinal where that compared role list occurrence1 is compared by some set comparison constraint that compares some compared role list occurrence2 that includes that role2 at that ordinal where compared role list occurrence1 <-> compared role list occurrence2.

2) It is possible that more than one role is in set comparison constraint column with the same role and that the same role is in set comparison constraint column with more than one role.

3) In each population of role is in set comparison constraint column with role, each role, role combination occurs at most once.

b) Set comparison constraint compares compared role list occurrence

1) *Set comparison constraint compares compared role list occurrence if and only if that set comparison constraint is some equality constraint that compares that compared role list occurrence or that set comparison constraint is some exclusion constraint that compares that compared role list occurrence or that set comparison constraint is some subset constraint that has that subset compared role list occurrence or has that superset compared role list occurrence.

2) Each set comparison constraint compares some compared role list occurrence.

3) Each compared role list occurrence is compared by exactly one set comparison constraint.

4) It is possible that the same set comparison constraint compares more than one compared role list occurrence.

c) Compared role list occurrence has arity

1) *Compared role list occurrence has arity if and only if that compared role list occurrence includes some role at some ordinal where arity = \text{count}(\text{each role for that compared role list occurrence}).

2) Each compared role list occurrence has exactly one arity.

3) It is possible that more than one compared role list occurrence has the same arity.

d) Set comparison constraint has column arity1

1) *Set comparison constraint has column arity1 if and only if that set comparison constraint compares some compared role list occurrence that has some arity2 where arity1 = \text{max}(\text{each arity2 for that set comparison constraint}).

2) Each set comparison constraint has exactly one column arity.

3) It is possible that more than one set comparison constraint has the same column arity.

e) Set comparison constraint has row arity

1) *Set comparison constraint has row arity if and only if that set comparison constraint compares some compared role list occurrence where arity = \text{count}(\text{each compared role list occurrence for that set comparison constraint}).

2) Each set comparison constraint has exactly one row arity.

3) It is possible that more than one set comparison constraint has the same row arity.

8.34.4 Other constraints

a) If some role1 is in set comparison constraint column with some role2 then that role1 is compatible with that role2.
b) For each set comparison constraint and arity, that set comparison constraint has that column arity if and only if that set comparison constraint compares some compared role list occurrence that has that arity.

c) It is possible that more than one role is compatible with the same role and that the same role is compatible with more than one role. In each population of role is compatible with role, each role, role combination occurs at most once.

8.34.5 Other considerations

a) Set comparison constraint is an entity type.
   1) Reference Scheme: constraint has UUID.
   2) Reference Mode: UUID.

b) Compared role list occurrence is an entity type.
   1) Reference Scheme: compared role list occurrence has UUID.
   2) Reference Mode: UUID.

8.34.6 Graphical representation

8.34.6.1 ORM2 representation

8.35 Subset

8.35.1 Introduction

8.35.2 Assertions

a) Subset constraint has superset compared role list occurrence
   1) Each subset constraint has exactly one superset compared role list occurrence.
   2) Each compared role list occurrence is subset for at most one subset constraint.

b) Subset constraint has subset compared role list occurrence
   1) Each subset constraint has exactly one subset compared role list occurrence.
   2) Each compared role list occurrence is superset for at most one subset constraint.
8.35.3 Derivations

8.35.4 Other constraints

8.35.5 Other considerations

a) Subset constraint is an entity type.
   1) Reference Scheme: constraint has UUID.
   2) Reference Mode: UUID.

8.35.6 Graphical representation

8.36 Equality

8.36.1 Introduction

8.36.2 Assertions

a) Equality constraint compares compared role list occurrence
   1) Each equality constraint compares some compared role list occurrence.
   2) Each compared role list occurrence is compared by at most one equality constraint.
   3) It is possible that the same equality constraint compares more than one compared role list occurrence.
   4) Each equality constraint in the population of “equality constraint compares compared role list occurrence” occurs there at least 2 times.

8.36.3 Derivations

8.36.4 Other constraints

a) Each equality constraint in the population of “equality constraint compares compared role list occurrence” occurs there at least 2 times.

8.36.5 Other considerations

a) Equality constraint is an entity type.
   1) Reference Scheme: constraint has UUID.
   2) Reference Mode: UUID.
8.36.6 Graphical representation

8.36.6.1 ORM2 representation

8.37 Exclusion

8.37.1 Introduction

8.37.2 Assertions

a) Exclusion constraint compares compared role list occurrence

   1) Each exclusion constraint compares some compared role list occurrence.

   2) Each compared role list occurrence is compared by at most one exclusion constraint.

   3) It is possible that the same exclusion constraint compares more than one compared role list occurrence.

   4) Each exclusion constraint in the population of “exclusion constraint compares compared role list occurrence” occurs there at least 2 times.

8.37.3 Derivations

8.37.4 Other constraints

a) Each exclusion constraint in the population of “exclusion constraint compares compared role list occurrence” occurs there at least 2 times.

8.37.5 Other considerations

a) Exclusion constraint is an entity type.

   1) Reference Scheme: constraint has UUID.

   2) Reference Mode: UUID.

8.37.6 Graphical representation

8.37.6.1 ORM2 representation
8.38 Join path

8.38.1 Introduction

8.38.2 Assertions

a) Join path applies to uniqueness constraint
   1) Each join path applies to at most one uniqueness constraint.
   2) For each uniqueness constraint, at most one join path applies to that uniqueness constraint.

b) Join path applies to frequency constraint
   1) Each join path applies to at most one frequency constraint.
   2) For each frequency constraint, at most one join path applies to that frequency constraint.

c) Join path applies to ring constraint
   1) Each join path applies to at most one ring constraint.
   2) For each ring constraint, at most one join path applies to that ring constraint.

d) Join path applies to value comparison constraint
   1) Each join path applies to at most one value comparison constraint.
   2) For each value comparison constraint, at most one join path applies to that value comparison constraint.

e) Join path applies to compared role list occurrence
   1) Each join path applies to at most one compared role list occurrence.
   2) For each compared role list occurrence, at most one join path applies to that compared role list occurrence.

f) Join step is explicitly projected on role
   1) Each join step is explicitly projected on at most one role.
   2) It is possible that more than one join step is explicitly projected on the same role.

g) Join step steps over role
   1) Each join step steps over exactly one role.
   2) It is possible that more than one join step steps over the same role.

h) Join step is in join path
   1) Each join step is in exactly one join path.
   2) Each join path has some join step.
   3) It is possible that the same join path has more than one join step.
i) Join step occurs in branch at ordinal
   1) Each join step occurs in branch at exactly one ordinal.
   2) It is possible that more than one join step occurs in branch at the same ordinal.

j) Join step is in branch ordinal
   1) Each join step is in exactly one branch ordinal.
   2) It is possible that more than one join step is in the same branch ordinal.

8.38.3 Derivations

a) Join path applies to role
   1) *Join path applies to role if and only if that join path applies to some uniqueness constraint that restricts that role at some ordinal or that join path applies to some frequency constraint that restricts that role or that join path applies to some ring constraint that restricts that first role or restricts that second role or that join path applies to some value comparison constraint that compares that first role or compares that second role or that join path applies to some compared role list occurrence that includes that role at some ordinal.
   2) It is possible that more than one join path applies to the same role and that the same join path applies to more than one role.
   3) In each population of join path applies to role, each join path, role combination occurs at most once.
   4) Each join path applies to some role.

b) Role1 in join path branch ordinal1 at ordinal2 has following role2
   1) *Role1 in join path branch ordinal1 at ordinal2 has following role2 if and only if some join step1 steps over that role1 and is in that branch ordinal1 and occurs in branch at that ordinal2 and is in that join path that has some join step2 that is in that branch ordinal1 and occurs in branch at some ordinal3 and steps over that role2 where ordinal3 = ordinal2 + 1.
   2) For each role1, join path, ordinal1, and ordinal2, that role1 in that join path branch that ordinal1 at that ordinal2 has following at most one role2.

c) Role1 is compatible with or in same fact type as role2
   1) *Role1 is compatible with or in same fact type as role2 if and only if that role1 belongs to some fact type that contains that role2 or that role1 is compatible with that role2 where role1 <> role2.
   2) It is possible that more than one role is compatible with or in same fact type as the same role and that the same role is compatible with or in same fact type as more than one role.
   3) In each population of role is compatible with or in same fact type as role, each role, role combination occurs at most once.

8.38.4 Other constraints

a) For each join path, exactly one of the following holds: that join path applies to some uniqueness constraint; that join path applies to some frequency constraint; that join path applies to some ring constraint; that join path applies to some value comparison constraint; that join path applies to some compared role list occurrence.
b) If some join path has some join step that is explicitly projected on some role then that join path applies to that role.

c) If some role1 in some join path branch some ordinal at some ordinal has following some role2 then that role1 is compatible with or in same fact type as that role2.

d) For each join path, branch ordinal1, and ordinal2, at most one join step is in that join path and is in that branch ordinal1 and occurs in branch at that ordinal2. This association with join path, ordinal1, ordinal2 provides the preferred identification scheme for join step.

8.38.5 Other considerations

a) Join path is an entity type.
   1) Reference Scheme: join path has UUID.
   2) Reference Mode: UUID.

b) Join step is an entity type.
   1) Reference Scheme: join step is in join path; join step is in branch ordinal; join step occurs in branch at ordinal.

8.38.6 Graphical representation

8.38.6.1 ORM2 representation
8.39 Join path projection

8.39.1 Introduction

Note: The non-emphasized subset constraint on this page is implied by the derivation rules.

Note: Join Constraints: (JoinPathProjectionIsComplete Equality Constraint) For each JoinPath and Role, that JoinPath applies to that Role if and only if that JoinStep has some JoinStep that is projected on that Role. (JoinPathProjectionIsCompatible Subset Constraint) If some JoinStep is projected on some Role1 and steps over some Role2 then that Role1 is compatible with that Role2.

8.39.2 Assertions

8.39.3 Derivations

a) Join step1 can implicitly project on role

1) *Join step1 can implicitly project on role if and only if some join path has that join step1 that steps over that role and that join path applies to that role and has no join step2 that steps over that role where join step1 <> join step2.

2) Each join step can implicitly project on at most one role.

3) It is possible that more than one join step can implicitly project on the same role.

b) Join step is implicitly projected on role

1) *Join step is implicitly projected on role if and only if it is not true that that join step is explicitly projected on that role and that join step can implicitly project on that role.

2) Each join step is implicitly projected on at most one role.

3) It is possible that more than one join step is implicitly projected on the same role.

c) Join step is projected on role

1) *Join step is projected on role if and only if that join step is explicitly projected on that role or is implicitly projected on that role.

2) Each join step is projected on at most one role.

3) It is possible that more than one join step is projected on the same role.

d) Role1 is compatible with role2

1) *Role1 is compatible with role2 if and only if that role1 is played by some object type1 and that role2 is played by some object type2 that is compatible with that object type1 where role1 <> role2.

2) It is possible that more than one role is compatible with the same role and that the same role is compatible with more than one role.

3) In each population of role is compatible with role, each role, role combination occurs at most once.

8.39.4 Other constraints

a) For each join path and role, that join path applies to that role if and only if that join path has some join step that is projected on that role.
b) If some join step is implicitly projected on some role then that join step can implicitly project on that role.

c) If some join step is projected on some role1 and steps over some role2 then that role1 is compatible with that role2.

8.39.5 Other considerations

8.39.6 Graphical representation

8.39.6.1 ORM2 representation

8.40 Join path multibranch

8.40.1 Introduction

8.40.2 Assertions

8.40.3 Derivations

a) Join path has multiple branches

1) *Join path has multiple branches if and only if that join path has some join step that is in some branch ordinal where ordinal <> 1.

2) In each population of join path has multiple branches, each join path occurs at most once.

b) Join path supports multiple branches

1) *Join path supports multiple branches if and only if that join path applies to some uniqueness constraint or applies to some frequency constraint.

2) In each population of join path supports multiple branches, each join path occurs at most once.

c) Role1 is compatible with role2
1) *Role1 is compatible with role2 if and only if that role1 is played by some object type1 and that role2 is played by some object type2 that is compatible with that object type1 where role1 <> role2.

2) It is possible that more than one role is compatible with the same role and that the same role is compatible with more than one role.

3) In each population of role is compatible with role, each role, role combination occurs at most once.

d) Join path lead role1 is paired with other join path lead role2

1) *Join path lead role1 is paired with other join path lead role2 if and only if some join path has some join step1 that is in some branch ordinal1 and occurs in branch at some ordinal2 where the possible value of that ordinal2 is 1 and steps over that role1 and that join path has some join step2 that is in some branch ordinal3 and occurs in branch at some ordinal4 where the possible value of that ordinal4 is 1 and steps over that role2 where ordinal1 < ordinal3.

2) It is possible that more than one join path lead role is paired with the same other join path lead role and that the same join path lead role is paired with more than one other join path lead role.

3) In each population of join path lead role is paired with other join path lead role, each role, role combination occurs at most once.

8.40.4 Other constraints

a) If some join path has multiple branches then that join path supports multiple branches.

8.40.5 Other considerations

a) If some join path lead role1 is paired with some other join path lead role2 then that role1 is compatible with that role2.

8.40.6 Graphical representation

8.40.6.1 ORM2 representation
8.41 Cardinality

8.41.1 Introduction

Note For each CardinalityRange, minimumCardinality. <= maximumCardinality For each CardinalityConstraint, CardinalityRanges do not overlap.

8.41.2 Assertions

a) Cardinality range specifies minimum cardinality
   1) The possible values of minimum cardinality in cardinality range specifies minimum cardinality are at least 1.
   2) Each cardinality range specifies at most one minimum cardinality.
   3) It is possible that more than one cardinality range specifies the same minimum cardinality.

b) Cardinality range specifies maximum cardinality
   1) The possible values of maximum cardinality in cardinality range specifies maximum cardinality are at least 1.
   2) Each cardinality range specifies at most one maximum cardinality.
   3) It is possible that more than one cardinality range specifies the same maximum cardinality.

c) Cardinality constraint applies to role
   1) Each cardinality constraint applies to at most one role.
   2) Each role has at most one cardinality constraint.

d) Cardinality constraint applies to object type
   1) Each cardinality constraint applies to at most one object type.
   2) Each object type has at most one cardinality constraint.

e) Cardinality constraint has cardinality range
   1) Each cardinality constraint has some cardinality range.
   2) Each cardinality range is for exactly one cardinality constraint.
   3) It is possible that the same cardinality constraint has more than one cardinality range.

8.41.3 Derivations

8.41.4 Other constraints

a) Each cardinality range specifies some minimum cardinality or specifies some maximum cardinality.

b) For each cardinality constraint, minimum cardinality1, and maximum cardinality2, at most one cardinality range is for that cardinality constraint and specifies that minimum cardinality1 and specifies that maximum cardinality2. This association with cardinality constraint, cardinality1, cardinality2 provides the preferred identification scheme for cardinality range.
c) For each cardinality constraint, exactly one of the following holds: that cardinality constraint applies to some object type; that cardinality constraint applies to some role.

8.41.5 Other considerations

a) Cardinality constraint is an entity type.
   1) Reference Scheme: constraint has UUID.
   2) Reference Mode: UUID.

b) Cardinality is a value type.
   1) Portable data type: Numeric: Signed Integer.
   2) The possible values of cardinality are at least 0.

c) Cardinality range is an entity type.
   1) Reference Scheme: cardinality constraint has cardinality range; cardinality range specifies minimum cardinality; cardinality range specifies maximum cardinality.

8.41.6 Graphical representation

8.41.6.1 ORM2 representation

8.42 Value constraint

8.42.1 Introduction

8.42.2 Assertions

a) Value constraint restricts role
   1) Each value constraint restricts at most one role.
   2) Each role is restricted by at most one value constraint.
b) Value constraint restricts object type
   1) Each value constraint restricts at most one object type.
   2) Each object type is restricted by at most one value constraint.

c) Value constraint allows value range
   1) Each value range is allowed in exactly one value constraint.
   2) It is possible that the same value constraint allows more than one value range.

d) Value constraint allows value instance
   1) It is possible that more than one value constraint allows the same value instance and that the same value constraint allows more than one value instance.
   2) In each population of value constraint allows value instance, each value instance, value constraint combination occurs at most once.
   3) This association with value instance, value constraint provides the preferred identification scheme for value constraint value.

e) Value instance has normalized lexical value
   1) Each value instance has exactly one normalized lexical value.
   2) It is possible that more than one value instance has the same normalized lexical value.

f) Value instance has user specified lexical value
   1) Each value instance has at most one user specified lexical value.
   2) It is possible that more than one value instance has the same user specified lexical value.

g) Value range has minimum value instance
   1) Each value range has at most one minimum value instance.
   2) This association with value range provides the preferred identification scheme for value range minimum.
   3) It is possible that more than one value range has the same minimum value instance.

h) Value range has maximum value instance
   1) Each value range has at most one maximum value instance.
   2) This association with value range provides the preferred identification scheme for value range maximum.
   3) It is possible that more than one value range has the same maximum value instance.

i) Value range minimum is included
   1) In each population of value range minimum is included, each value range minimum occurs at most once.
j) Value range maximum is included
   1) In each population of value range maximum is included, each value range maximum occurs at most once.

8.42.3 Derivations

a) Role has value role player
   1) *Role has value role player if and only if that role is played by some object type that is some semantic object type that has a simple identifier based on some value type.

   2) In each population of role has value role player, each role occurs at most once.

8.42.4 Other constraints

a) For each value constraint, exactly one of the following holds: that value constraint restricts some role; that value constraint restricts some object type.

b) If some role is restricted by some value constraint then that role has value role player.

c) Each value range has some minimum value instance or has some maximum value instance.

d) Each value constraint allows some value range or allows some value instance.

e) For each value constraint, minimum value instance1, and maximum value instance2, at most one value range is allowed in that value constraint and has that minimum value instance1 and has that maximum value instance2. This association with value constraint, value instance1, value instance2 provides the preferred identification scheme for value range.

f) If some object type is restricted by some value constraint then that object type can restrict values.

8.42.5 Other considerations

a) Value range is an entity type.
   1) Reference Scheme: value constraint allows value range; value range has minimum value instance; value range has maximum value instance.

b) Value instance is an entity type.
   1) Reference Scheme: value instance has normalized lexical value; value instance has user specified lexical value.

c) Lexical value is a value type.
   1) Portable data type: Text: Variable Length.

d) For each normalized lexical value1 and user specified lexical value2, at most one value instance has that normalized lexical value1 and has that user specified lexical value2.
   1) This association with lexical value1, lexical value2 provides the preferred identification scheme for value instance.

e) Value constraint is an entity type.
   1) Reference Scheme: constraint has UUID.

   2) Reference Mode: UUID.
8.42.6 Graphical representation

8.42.6.1 ORM2 representation

8.43 Adhoc constraint

8.43.1 Introduction

8.43.2 Assertions

a) **Ad hoc constraint applies to fact type**

   1) It is possible that more than one ad hoc constraint applies to the same fact type and that more than one fact type has the same ad hoc constraint.

   2) In each population of ad hoc constraint applies to fact type, each ad hoc constraint, fact type combination occurs at most once.

   3) This association with ad hoc constraint, fact type provides the preferred identification scheme for ad hoc constraint applies to fact type.

b) **Ad hoc constraint applies to object type**

   1) It is possible that more than one ad hoc constraint applies to the same object type and that more than one object type has the same ad hoc constraint.

   2) In each population of ad hoc constraint applies to object type, each ad hoc constraint, object type combination occurs at most once.
3) This association with ad hoc constraint, object type provides the preferred identification scheme for ad hoc constraint applies to object type.

c) Ad hoc constraint applies to role

1) It is possible that more than one ad hoc constraint applies to the same role and that more than one role has the same ad hoc constraint.

2) In each population of ad hoc constraint applies to role, each ad hoc constraint, role combination occurs at most once.

3) This association with ad hoc constraint, role provides the preferred identification scheme for ad hoc constraint applies to role.

d) Ad hoc constraint has constraint description

1) Each ad hoc constraint has exactly one constraint description.

2) It is possible that more than one ad hoc constraint has the same constraint description.

8.43.3 Derivations

8.43.4 Other constraints

a) Each ad hoc constraint applies to some object type or applies to some role or applies to some fact type.

8.43.5 Other considerations

a) Constraint description is a value type.

1) Portable data type: Text: Large Length.

8.43.6 Graphical representation

8.43.6.1 ORM2 representation
8.44 Sample population

8.44.1 Introduction

Note: Join Constraints: (ObjectificationRequiresTypedObjectInstanceIdentifiesFactInstance Equality Constraint) For each TypedObjectInstance and FactInstance, that TypedObjectInstance identifies that objectified FactInstance if and only if that TypedObjectInstance is typed as some SemanticObjectType that objectifies some FactType where that FactInstance is an instance of that FactType. (FactTypeInstanceOfFactInstancePopulatesAllFactRoles Equality Constraint) For each FactInstance and Role, that FactInstance populates that fact Role with some ObjectInstance if and only if that FactInstance is an instance of some FactType that contains that Role. (PopulatedRoleInstanceOfRolePlayerType Subset Constraint) If some FactInstance populates some fact Role with some ObjectInstance then that ObjectInstance is an instance of some SemanticObjectType that is some ObjectType that plays that Role.

Notes: (SharedInstancesMustBeOfTypeExclusion Constraint) For each ObjectInstance and ObjectType1, at most one of the following holds: that ObjectInstance is an instance of some SemanticObjectType1 that is that ObjectType1; that ObjectInstance is an instance of some SemanticObjectType2 where it is not true that that SemanticObjectType2 is some ObjectType2 that is compatible with that ObjectType1.

8.44.2 Assertions

a) Fact instance is an instance of fact type
   
   1) Each fact instance is an instance of exactly one fact type.
   
   2) It is possible that more than one fact instance is an instance of the same fact type.

b) Typed object instance identifies objectified fact instance
   
   1) Each typed object instance identifies at most one objectified fact instance.
   
   2) Each fact instance is identified by at most one objectifying typed object instance.

c) Object instance is an instance of semantic object type
   
   1) It is possible that more than one object instance is an instance of the same semantic object type and that the same object instance is an instance of more than one semantic object type.
   
   2) In each population of object instance is an instance of semantic object type, each object instance, semantic object type combination occurs at most once.
   
   3) This association with object instance, semantic object type provides the preferred identification scheme for typed object instance.
   
   4) Each object instance is an instance of some semantic object type.

d) Fact instance populates fact role with object instance
   
   1) For each fact instance and fact role, that fact instance populates that fact role with at most one object instance.
   
   2) This association with fact instance, role provides the preferred identification scheme for fact role instance.

e) Self identified object instance has value instance
   
   1) Each self identified object instance has exactly one value instance.
   
   2) It is possible that more than one self identified object instance has the same value instance.

f) Typed object instance is typed as semantic object type
1) Each typed object instance is typed as exactly one semantic object type.

2) It is possible that more than one typed object instance is typed as the same semantic object type.

### 8.44.3 Derivations

### 8.44.4 Other constraints

a) For each typed object instance and fact instance, that typed object instance identifies that objectified fact instance if and only if that typed object instance is typed as some semantic object type that objectifies some fact type where that fact instance is an instance of that fact type.

b) For each fact instance and role, that fact instance populates that fact role with some object instance if and only if that fact instance is an instance of some fact type that contains that role.

c) If some fact instance populates some fact role with some object instance then that object instance is an instance of some semantic object type that is some object type that plays that role.

d) Each role is played by at most one object type.

### 8.44.5 Other considerations

a) Fact instance is an entity type.

1) Reference Scheme: fact instance has UUID.

2) Reference Mode: UUID.

b) Object instance is an entity type.

1) Reference Scheme: object instance has UUID.

2) Reference Mode: UUID.

c) Self identified object instance is an entity type.

1) *Each self identified object instance is some object instance that is an instance of some semantic object type that is some value type.*

2) Reference Scheme: object instance has UUID.

3) Reference Mode: UUID.
8.44.6 Graphical representation

8.44.6.1 ORM2 representation

8.45 Sample population and instance compatibility

8.45.1 Introduction

Note (SubtypeInstanceMustBeSupertypeInstance Subset Constraint) If some ObjectInstance is an instance of some SemanticObjectType that is some ObjectType1 that is a direct supertype of some ObjectType then that ObjectInstance is an instance of that SemanticObjectType.

Note (The constraint display on this page is unclear without highlighted roles because the ObjectInstance role is in both compared role lists on both constraints.).

8.45.2 Assertions

8.45.3 Derivations

8.45.4 Other constraints

a) If some object instance is an instance of some semantic object type that is some object type1 that is a direct supertype of some object type then that object instance is an instance of that semantic object type.

b) For each object instance and object type1, at most one of the following holds: that object instance is an instance of some semantic object type1 that is that object type1; that object instance is an instance of some semantic object type2 where it is not true that that semantic object type2 is some object type2 that is compatible with that object type1.
8.45.5 Other considerations

8.45.6 Graphical representation

8.45.6.1 ORM2 representation