RDF Constraint Types to Validate Metadata on Highly-Complex Person-Level and Aggregated Data

Thomas Bosch[1], Benjamin Zapilko[1], Joachim Wackerow[1], and Kai Eckert[2]

[1] GESIS – Leibniz Institute for the Social Sciences, Germany
{firstname.lastname}@gesis.org,
[2] University of Mannheim, Germany
kai@informatik.uni-mannheim.de

Abstract. ...

Keywords: RDF Validation, RDF Constraints, DDI-RDF Discovery Vocabulary, Disco, RDF Data Cube Vocabulary, Linked Data, Semantic Web

1 Introduction

Bosch and Eckert initiated a comprehensive database[3] on RDF validation requirements to collect case studies, use cases, and requirements [1]. It is continuously updated and used to evaluate and to compare various existing solutions for RDF constraint formulation and validation. Requirements are classified to provide a high-level view on different solutions and to facilitate a better understanding of the problem domain. Bosch et al. identified in total 74 requirements to formulate RDF constraints; each of them corresponding to a constraint type. They recently published a technical report[4] in which they explain each requirement (constraint type) in detail and give examples for each (represented by listed constraint languages) [2]. We state at least one Disco constraint for the majority of the constraint types. Where appropriate constraint types are related to complementary requirements of the RDF validation requirements database. When constraint types are mapped to DL, we do not state namespace prefixes for simplicity reasons.

2 Validation Types

Content-Driven Validation / Data Model Consistency. Is the data consistent with the intended semantics of the data model? Such validation rules ensure the integrity of the data according to the data model.

[3] Publicly available at http://purl.org/net/rdf-validation.
[4] Available at: http://arxiv.org/abs/1501.03933

Technology-Driven Validation. Some validation rules can be generated automatically out of the defined data model, such as cardinality restrictions, universal quantifications, domains, ranges.

3 Validation of Metadata on Person-Level and Aggregated Data

- Constraint types marked with an asterisk (*) can be used as OWL 2 axioms. Thus, reasoners may be used to infer implicit triples resolving constraint violations caused when these constraints are validated.
- <u>Underlined subsections</u> are data model specific constraint types and do not correspond to RDF validation requirements (constraint types).

3.1 <u>Comparison</u>

- ***DISCO-C-COMPARISON-VARIABLES-01***: are compared variables represented in a compatible way, i.e. are the variables' code lists theoretically comparable?
  - severity level: WARNING
- ***DISCO-C-COMPARISON-VARIABLES-02***: are variable definitions (*dcterms:description*) available for each variable (*disco:Variable*) to compare?
  - severity level: ERROR
- ***DISCO-C-COMPARISON-VARIABLES-03***: are code lists structured properly for each variable (*disco:Variable*) to compare?
  - severity level: WARNING
- ***DISCO-C-COMPARISON-VARIABLES-04***: is for each code (for each variable (*disco:Variable*) to compare) an associated category (a human-readable label) specified?
  - severity level: INFO
- ***DISCO-C-COMPARISON-VARIABLES-05***: each (*disco:Variable*) to compare must be present.
  - severity level: ERROR

3.2 <u>Data Model Consistency</u>

Is the data consistent with the intended semantics of the data model? Such validation rules ensure the integrity of the data according to the data model.

- ***DATA-CUBE-C-DATA-MODEL-CONSISTENCY-01:*** Only attributes may be optional (*IC-6* [3]) - The only components of a *qb:DataStructureDefinition* that may be marked as optional, using *qb:componentRequired* are attributes.
- ***DATA-CUBE-C-DATA-MODEL-CONSISTENCY-02:*** No duplicate observations (*IC-12* [3]) - No two *qb:Observations* in the same *qb:DataSet* may have the same value for all dimensions.

– ***DATA-CUBE-C-DATA-MODEL-CONSISTENCY-03:*** Slice Keys consistent with DSD (*IC-8* [3]) - Every *qb:componentProperty* on a *qb:SliceKey* must also be declared as a *qb:component* of the associated *qb:DataStructureDefinition*.
– ***DATA-CUBE-C-DATA-MODEL-CONSISTENCY-04:*** Required attributes (*IC-13* [3]) - Every *qb:Observation* has a value for each declared attribute that is marked as required.
– ***DATA-CUBE-C-DATA-MODEL-CONSISTENCY-05:*** All measures present (*IC-14* [3]) - In a *qb:DataSet* which does not use a Measure dimension then each individual *qb:Observation* must have a value for every declared measure.
– ***DATA-CUBE-C-DATA-MODEL-CONSISTENCY-06:*** Measure dimension consistent (*IC-15* [3]) - In a *qb:DataSet* which uses a Measure dimension then each qb:Observation must have a value for the measure corresponding to its given *qb:measureType*.
– ***DATA-CUBE-C-DATA-MODEL-CONSISTENCY-07:*** Single measure on measure dimension observation (*IC-16* [3]) - In a *qb:DataSet* which uses a Measure dimension then each *qb:Observation* must only have a value for one measure (by *IC-15* this will be the measure corresponding to its *qb:measureType*).
– ***DATA-CUBE-C-DATA-MODEL-CONSISTENCY-08:*** All measures present in measures dimension cube (*IC-17* [3]) - In a *qb:DataSet* which uses a Measure dimension then if there is a Observation for some combination of non-measure dimensions then there must be other Observations with the same non-measure dimension values for each of the declared measures.
– ***DATA-CUBE-C-DATA-MODEL-CONSISTENCY-09:*** Consistent data set links (*IC-18* [3]) - If a *qb:DataSet* D has a *qb:slice* S, and S has an *qb:observation* O, then the *qb:dataSet* corresponding to O must be D.

– ***SKOS-C-DATA-MODEL-CONSISTENCY-01[5]:*** Relation Clashes: Covers condition S27 from the SKOS reference document, that has not been defined formally.
  - Implementation: In a first step, all pairs of concepts are found that are associatively connected, using a SPARQL query. In the second step, a graph is created, containing only hierarchically related concepts and the respective relations. For each concept pair from the first step, we check for a path in the graph from step two. If such a path is found, a clash has been identified and the causing concepts are returned.
  - Severity level:

– ***SKOS-C-DATA-MODEL-CONSISTENCY-02[6]:*** Mapping Clashes: Covers condition S46 from the SKOS reference document, that has not been defined formally.
  - Implementation: Can be solved by issuing a SPARQL query.

[5] Corresponds to qSKOS Quality Issues - SKOS Semi-Formal Consistency Issues - Relation Clashes
[6] Corresponds to qSKOS Quality Issues - SKOS Semi-Formal Consistency Issues - Mapping Clashes

- Severity level:

– ***SKOS-C-DATA-MODEL-CONSISTENCY-03[7]:*** Mapping Relations Misuse: According to the SKOS reference documentation, mapping relations (e.g., *skos:broadMatch* or *skos:relatedMatch*) should be asserted to concepts being members of different concept schemes. This check finds concepts that are related by a mapping property and are either members of the same concept scheme or members of no concept scheme at all.
  - Severity level:

3.3 Subsumption*

A *subclass axiom*[8] (*concept inclusion* in DL) states that the class *C1* is a subclass of the class *C2* - *C1* is more specific than *C2*, i.e. each resource of the class *C1* must also be part of the class extension of *C2*.

– ***DISCO-C-SUBSUMPTION-01:*** All *disco:Universe*s must also be *skos:Concept*s (`Universe` $\sqsubseteq$ `Concept`).

3.4 Class Equivalence*

Class Equivalence[9] asserts that two concepts have the same instances. While synonyms are an obvious example of equivalent concepts, in practice one more often uses concept equivalence to give a name to complex expressions [5]. Concept equivalence is indeed subsumption from left and right ($A \sqsubseteq B$ and $B \sqsubseteq A$ implies $A \equiv B$).

– ***DISCO-C-CLASS-EQUIVALENCE-01:*** All *sio:SIO_000367* resources must also be *disco:Variable*s (`Variable` $\equiv$ `SIO_000367`). The Semanticscience Integrated Ontology (SIO)[10] provides a simple, integrated ontology of types and relations for rich description of objects, processes and their attributes. *sio:SIO_000367* is a variable defined as a value that may change within the scope of a given problem or set of operations. Thus, *sio:SIO_000367* is equivalent to *disco:Variable*.

3.5 Sub Properties*

Sub Properties[11] state that the property *P1* is a sub property of the property *P2* - that is, if an individual x is connected by *P1* to an individual or a literal y, then x is also connected by *P2* to y.

– ***DISCO-C-SUB-PROPERTIES-01:*** If an individual x is connected by *disco:fundedBy* to an individual y, then x is also connected by *dcterms:contributor* to y (`fundedBy` $\sqsubseteq$ `contributor`).

[7] Corresponds to qSKOS Quality Issues - SKOS Semi-Formal Consistency Issues - Mapping Relations Misuse
[8] *R-100-SUBSUMPTION*
[9] *R-3-EQUIVALENT-CLASSES*
[10] https://code.google.com/p/semanticscience/wiki/SIO
[11] *R-54-SUB-OBJECT-PROPERTIES*, *R-64-SUB-DATA-PROPERTIES*

3.6 Property Domains*

Property Domains[12] (*domain restrictions on roles* in DL) restrict the domain of object and data properties. The purpose is to declare that a given property is associated with a class. In OO terms this is the declaration of a member, field, attribute or association. $\exists R.\top \sqsubseteq C$ is the object property restriction where R is the object property (role) whose domain is restricted to concept C.

- ***DISCO-C-PROPERTY-DOMAIN-01:*** *Property Domain* constraints are defined for each *Disco* object and data property. Only *disco:Question*s, e.g., can have *disco:responseDomain* relationships ($\exists$ `responseDomain.`$\top \sqsubseteq$ `Question`).
  - Severity level: ERROR
- ***DATA-CUBE-C-PROPERTY-DOMAIN-01:*** *Property Domain* constraints are defined for each *Data Cube* object and data property. Only *qb:Observation*s, e.g., can have *qb:dataSet* relationships ($\exists$ `dataSet.`$\top \sqsubseteq$ `Observation`).
  - Severity level: ERROR
- ***DCAT-C-PROPERTY-DOMAIN-01:*** *Property Domain* constraints are defined for each *DCAT* object and data property. Only *dcat:Catalog*s, e.g., can have *dcat:dataset* relationships ($\exists$ `dataset.`$\top \sqsubseteq$ `Catalog`).
  - Severity level: ERROR
- ***PHDD-C-PROPERTY-DOMAIN-01:*** *Property Domain* constraints are defined for each *PHDD* object and data property. Only *phdd:Table*s, e.g., can have *phdd:isStructuredBy* relationships ($\exists$ `isStructuredBy.`$\top \sqsubseteq$ `Table`).
  - Severity level: ERROR
- ***SKOS-C-PROPERTY-DOMAIN-01:*** *Property Domain* constraints are defined for each *SKOS* object and data property. Only *skos:ConceptSchemes*, e.g., can have *skos:hasTopConcept* relationships ($\exists$ `hasTopConcept.`$\top \sqsubseteq$ `ConceptScheme`).
  - Severity level: ERROR
- ***XKOS-C-PROPERTY-DOMAIN-01:*** *Property Domain* constraints are defined for each *XKOS* object and data property.
  - Severity level: ERROR

3.7 Property Ranges*

Property Ranges[13] (*range restrictions on roles* in DL) restrict the range of object and data properties. $\top \sqsubseteq \forall R.C$ is the range restriction to the object property R (restricted by the concept C).

---

[12] *R-25-OBJECT-PROPERTY-DOMAIN*, *R-26-DATA-PROPERTY-DOMAIN*
[13] *R-28-OBJECT-PROPERTY-RANGE*, *R-35-DATA-PROPERTY-RANGE*

– ***DISCO-C-PROPERTY-RANGES-01:*** *Property Range* constraints are defined for each *Disco* object and data property. *disco:caseQuantity* relationships, e.g., can only point to literals of the datatype *xsd:nonNegativeInteger* ($\top \sqsubseteq \forall$ `caseQuantity.nonNegativeInteger`).
  - Severity level: ERROR
– ***DATA-CUBE-C-PROPERTY-RANGES-01:*** *Property Range* constraints are defined for each *Data Cube* object and data property. *qb:order* relationships, e.g., can only point to literals of the datatype *xsd:string* ($\top \sqsubseteq \forall$ `order.string`).
  - Severity level: ERROR
– ***DCAT-C-PROPERTY-RANGES-01:*** *Property Range* constraints are defined for each *DCAT* object and data property. *dcat:bytes* relationships, e.g., can only point to literals of the datatype *xsd:integer* ($\top \sqsubseteq \forall$ `bytes.integer`).
  - Severity level: ERROR
– ***PHDD-C-PROPERTY-RANGES-01:*** *Property Range* constraints are defined for each *PHDD* object and data property. *phdd:caseQuantity* relationships, e.g., can only point to literals of the datatype *xsd:nonNegativeInteger* ($\top \sqsubseteq \forall$ `caseQuantity.nonNegativeInteger`).
  - Severity level: ERROR
– ***SKOS-C-PROPERTY-RANGES-01:*** *Property Range* constraints are defined for each *SKOS* object and data property.
  - Severity level: ERROR
– ***XKOS-C-PROPERTY-RANGES-01:*** *Property Range* constraints are defined for each *XKOS* object and data property. *xkos:belongsTo* relationships, e.g., can only point to instances of the class *skos:Concept* ($\top \sqsubseteq \forall$ `belongsTo.Concept`).
  - Severity level: ERROR

3.8 Inverse Object Properties*

In many cases, properties are used bi-directionally and then accessed in the inverse direction, e.g. parent $\equiv$ child^{-}. There should be a way to declare value type, cardinality etc of those inverse relations without having to declare a new property URI. The object property *OP1* is an inverse[14] of the object property *OP2*. Thus, if an individual x is connected by *OP1* to an individual y, then y is also connected by *OP2* to x, and vice versa.

– ***DISCO-C-INVERSE-OBJECT-PROPERTIES-01***: *disco:CategoryStatistics* resources are accessed from codes (*skos:Concepts*) via *disco:statisticsCategory*$^{-}$.
– ***DISCO-C-INVERSE-OBJECT-PROPERTIES-02***: *disco:SummaryStatistics* resources are accessed from *disco:Variables* via *disco:statisticsVariable*$^{-}$.
– ***DISCO-C-INVERSE-OBJECT-PROPERTIES-03***: *disco:Variables* are accessed from *disco:Questions* via *disco:question*$^{-}$.

[14] *R-56-INVERSE-OBJECT-PROPERTIES*

3.9 Symmetric Object Properties*

A role is symmetric if it is equivalent to its own inverse [5]. An object property symmetry axiom[15] states that the object property expression *OPE* is symmetric - that is, if an individual x is connected by *OPE* to an individual y, then y is also connected by *OPE* to x.

3.10 Asymmetric Object Properties*

A property is asymmetric[16] if it is disjoint from its own inverse [5]. An object property asymmetry axiom states that the object property *OP* is asymmetric - that is, if an individual x is connected by *OP* to an individual y, then y cannot be connected by *OP* to x.

- ***DISCO-C-ASYMMETRIC-OBJECT-PROPERTIES-01:*** A *disco:Variable* may be based on a *disco:RepresentedVariable*. A *disco:RepresentedVariable*, however, cannot be based on a *disco:Variable*. This is a kind of mistake which may occur as a semantically equivalent object property for the other direction may also be possible (*disco:basisOf*) ($basedOn \sqcap basedOn^{-} \sqsubseteq \bot$).

3.11 Reflexive Object Properties*

Reflexive Object Properties[17] (*reflexive roles*, *global reflexivity* in DL) can be expressed by imposing local reflexivity on the top concept [5].

3.12 Irreflexive Object Properties*

An object property is irreflexive[18] (*irreflexive role* in DL) if it is never locally reflexive [5]. An object property irreflexivity axiom *IrreflexiveObjectProperty(OPE)* states that the object property expression *OPE* is irreflexive - that is, no individual is connected by *OPE* to itself.

- ***DISCO-C-IRREFLEXIVE-OBJECT-PROPERTIES-01:*** In *Disco*, every object property is irreflexive. No individual is connected by the object property *instrument* to itself ($\top \sqsubseteq \neg \exists instrument.Self$).

3.13 Class-Specific Irreflexive Object Properties*

A property is *irreflexive* if it is never locally reflexive [5]. An object property irreflexivity axiom states that the object property *OP* is irreflexive - that is, no individual is connected by *OP* to itself. *Class-Specific Irreflexive Object Properties* are object properties which are irreflexive within a given context, e.g. a class.

[15] *R-61-SYMMETRIC-OBJECT-PROPERTIES*
[16] *R-62-ASYMMETRIC-OBJECT-PROPERTIES*
[17] *R-59-REFLEXIVE-OBJECT-PROPERTIES*
[18] *R-60-IRREFLEXIVE-OBJECT-PROPERTIES*

– ***DISCO-C-CLASS-SPECIFIC-IRREFLEXIVE-OBJECT-PROPERTIES-01:*** Within the Disco context, *skos:Concept*s cannot be related via the object property *skos:boader* to themselves (`Concept` $\sqsubseteq$ $\neg\exists$ `broader.Self.`).
– ***DISCO-C-CLASS-SPECIFIC-IRREFLEXIVE-OBJECT-PROPERTIES-02:*** Within the Disco context, *skos:Concept*s cannot be related via the object property *skos:narrower* to themselves (`Concept` $\sqsubseteq$ $\neg\exists$ `narrower.Self.`).

3.14 Disjoint Properties

A *disjoint properties axiom*[19] states that all of the properties are pairwise disjoint; that is, no individual x can be connected to an individual/literal y by these properties.

– ***DATA-CUBE-C-DISJOINT-PROPERTIES-01:*** All *Data Cube* properties (not having the same domain and range classes) are defined to be pairwise disjoint. The properties *qb:dataSet* and *qb:structure* are disjoint ($dataSet \sqsubseteq \neg structure$).

– ***DCAT-C-DISJOINT-PROPERTIES-01:*** All *DCAT* properties (not having the same domain and range classes) are defined to be pairwise disjoint.

– ***DISCO-C-DISJOINT-PROPERTIES-01:*** All *Disco* properties (not having the same domain and range classes) are defined to be pairwise disjoint. The properties *disco:variable* and *disco:question* are disjoint ($variable \sqsubseteq \neg question$).

– ***PHDD-C-DISJOINT-PROPERTIES-01:*** All *PHDD* properties (not having the same domain and range classes) are defined to be pairwise disjoint. The properties *phdd:isStructuredBy* and *phdd:column* are disjoint ($isStructuredBy \sqsubseteq \neg column$)

– ***SKOS-C-DISJOINT-PROPERTIES-01:*** All *SKOS* properties (not having the same domain and range classes) are defined to be pairwise disjoint.
– ***SKOS-C-DISJOINT-PROPERTIES-02[20]:*** Disjoint Labels Violation: Covers condition S13 from the SKOS reference document stating that "*skos:prefLabel*, *skos:altLabel* and *skos:hiddenLabel* are pairwise disjoint properties".
  • Implementation: A SPARQL query collects all labels of all concepts, building an in-memory structure. This structure is then checked for disjoint entries.
  • Severity level:

– ***XKOS-C-DISJOINT-PROPERTIES-01:*** All *XKOS* properties (not having the same domain and range classes) are defined to be pairwise disjoint.

[19] *R-9-DISJOINT-PROPERTIES*
[20] Corresponds to qSKOS Quality Issues - SKOS Semi-Formal Consistency Issues - Disjoint Labels Violation

3.15 Disjoint Classes

Disjoint Classes[21] state that all of the classes are pairwise disjoint; that is, no individual can be at the same time an instance of these disjoint classes.

- ***DATA-CUBE-C-DISJOINT-CLASSES-01:*** All *Data Cube* classes are defined to be pairwise disjoint.

- ***DCAT-C-DISJOINT-CLASSES-01:*** All *DCAT* classes are defined to be pairwise disjoint.

- ***DISCO-C-DISJOINT-CLASSES-01:*** All *Disco* classes are defined to be pairwise disjoint (e.g. $\texttt{Study} \sqcap \texttt{Variable} \sqsubseteq \bot$).

- ***PHDD-C-DISJOINT-CLASSES-01:*** All *PHDD* classes are defined to be pairwise disjoint.

- ***SKOS-C-DISJOINT-CLASSES-01:*** All *SKOS* classes are defined to be pairwise disjoint.

- ***XKOS-C-DISJOINT-CLASSES-01:*** All *XKOS* classes are defined to be pairwise disjoint.

3.16 Context-Specific Property Groups

The *Context-Specific Property Groups*[22] constraint groups data and object properties within a context (e.g. a class).

3.17 Context-Specific Inclusive OR of Properties

Inclusive or is a logical connective joining two or more predicates that yields the logical value "true" when at least one of the predicates is true. *Context-Specific Inclusive OR of Properties*[23] constraints specify that individuals are valid if they have at least one property relationship of one or multiple properties stated within a given context. The context can be an application profile, a shape, or a class, i.e., the constraint applies for individuals of this specific class.

3.18 Context-Specific Inclusive OR of Property Groups

At least one property group must match for individuals of a specific context. Context may be a class, a shape, or an application profile.

---

[21] *R-7-DISJOINT-CLASSES*

[22] *R-66-PROPERTY-GROUPS*

[23] *R-202-CONTEXT-SPECIFIC-INCLUSIVE-OR-OF-PROPERTIES*

3.19 Recursive Queries

Resource Shapes is a recursive language[24] (the value shape of a Resource Shape is in turn another Resource Shape). There is no way to express that in SPARQL without hand-waving "and then you call the function again here" or "and then you embed this operation here" text. The embedding trick doesn't work in the general case because SPARQL can't express recursive queries, e.g. "test that this Issue is valid and all of the Issues that references, recursively". Most SPARQL engines already have functions that go beyond the official SPARQL 1.1 spec. The cost of that sounds manageable.

3.20 Individual Inequality

An *individual inequality axiom*[25] DifferentIndividuals(a_1 ... a_n) states that all of the individuals a_i, $1 \leqslant i \leqslant n$, are different from each other; that is, no individuals a_i and a_j with $i \neq j$ can be derived to be equal. This axiom can be used to axiomatize the unique name assumption — the assumption that all different individual names denote different individuals.

3.21 Equivalent Properties*

An *equivalent object properties axiom*[26] *EquivalentObjectProperties(OPE_1 ... OPE_n)* states that all of the object property expressions *OPE_i, $1 \leqslant i \leqslant n$*, are semantically equivalent to each other. This axiom allows one to use each OPE_i as a synonym for each OPE_j — that is, in any expression in the ontology containing such an axiom, OPE_i can be replaced with OPE_j without affecting the meaning of the ontology. The axiom *EquivalentObjectProperties(OPE_1 OPE_2)* is equivalent to the following two axioms *SubObjectPropertyOf(OPE_1 OPE_2)* and *SubObjectPropertyOf(OPE_1 OPE_2)*.

An *equivalent data properties axiom*[27] *EquivalentDataProperties(DPE_1 ... DPE_n)* states that all the data property expressions *DPE_i, $1 \leqslant i \leqslant n$*, are semantically equivalent to each other. This axiom allows one to use each DPE_i as a synonym for each DPE_j — that is, in any expression in the ontology containing such an axiom, DPE_i can be replaced with DPE_j without affecting the meaning of the ontology. The axiom *EquivalentDataProperties(DPE_1 DPE_2)* can be seen as a syntactic shortcut for the following axiom *SubDataPropertyOf(DPE_1 DPE_2)* and *SubDataPropertyOf(DPE_1 DPE_2)*.

- ***DATA-CUBE-C-EQUIVALENT-PROPERTIES-01:*** Equivalent properties from different versions of *Data Cube* can be marked as equivalent. As a consequence, the properties can be replaced by each other without affecting the meaning.

[24] *R-222-RECURSIVE-QUERIES*
[25] *R-14-DISJOINT-INDIVIDUALS*
[26] *R-4-EQUIVALENT-OBJECT-PROPERTIES*
[27] *R-5-EQUIVALENT-DATA-PROPERTIES*

– ***DCAT-C-EQUIVALENT-PROPERTIES-01:*** Equivalent properties from different versions of *DCAT* can be marked as equivalent. As a consequence, the properties can be replaced by each other without affecting the meaning.

– ***DISCO-C-EQUIVALENT-PROPERTIES-01:*** Equivalent properties from different versions of *Disco* can be marked as equivalent, e.g. *disco:containsVariable* and *disco:variable*. As a consequence, the properties can be replaced by each other without affecting the meaning.

– ***PHDD-C-EQUIVALENT-PROPERTIES-01:*** Equivalent properties from different versions of *PHDD* can be marked as equivalent. As a consequence, the properties can be replaced by each other without affecting the meaning.

– ***SKOS-C-EQUIVALENT-PROPERTIES-01:*** Equivalent properties from different versions of *SKOS* can be marked as equivalent. As a consequence, the properties can be replaced by each other without affecting the meaning.

– ***XKOS-C-EQUIVALENT-PROPERTIES-01:*** Equivalent properties from different versions of *XKOS* can be marked as equivalent. As a consequence, the properties can be replaced by each other without affecting the meaning.

3.22 Property Assertions

Property Assertions[28] and includes positive property assertions and negative property assertions. A *positive object property assertion ObjectPropertyAssertion(OPE* a_1 a_2 *)* states that the individual a_1 is connected by the object property expression *OPE* to the individual a_2. A *negative object property assertion NegativeObjectPropertyAssertion(OPE* a_1 a_2 *)* states that the individual a_1 is not connected by the object property expression *OPE* to the individual a_2. A *positive data property assertion DataPropertyAssertion(DPE a lt)* states that the individual a is connected by the data property expression DPE to the literal *lt*. A *negative data property assertion NegativeDataPropertyAssertion(DPE a lt)* states that the individual a is not connected by the data property expression *DPE* to the literal *lt*.

3.23 Data Property Facets

For datatype properties it should be possible to declare frequently needed *facets*[29] to drive user interfaces and validate input against simple conditions, including min/max value, regular expressions, string length - similar to XSD datatypes. Constraining facets, to restrict datatypes of RDF literals, may be: *xsd:length*, *xsd:minLength*, *xsd:maxLength*, *xsd:pattern*, *xsd:enumeration*, *xsd:whiteSpace*, *xsd:maxInclusive*, *xsd:maxExclusive*, *xsd:minExclusive*, *xsd:minInclusive*, *xsd:totalDigits*, *xsd:fractionDigits*.

– ***DISCO-C-DATA-PROPERTY-FACETS-01:*** The abstract of a study (*disco:purpose*) should have a minimum length (*xsd:minLength*) of x.

[28] *R-96-PROPERTY-ASSERTIONS*
[29] *R-46-CONSTRAINING-FACETS*

3.24 Literal Pattern Matching

There are multiple use cases associated with the requirement to match literals according to given patterns[30].

– ***DISCO-C-LITERAL-PATTERN-MATCHING-01:*** Each *disco:Variable* of a given *disco:LogicalDataSet* must have a given prefix for its variable name (*skos:notation*).

3.25 Negative Literal Pattern Matching

Literals of given data properties within given contexts do not have to match given patterns[31].

– ***DISCO-C-NEGATIVE-LITERAL-PATTERN-MATCHING-01:***

3.26 Object Property Paths*

Object Property Paths[32] (or *Object Property Chains* and in DL terminology *complex role inclusion axiom* or *role composition*) is the more complex form of sub properties. This axiom states that, if an individual x is connected by a sequence of object property expressions OPE_1, ..., OPE_n with an individual y, then x is also connected with y by the object property expression *OPE*. Role composition can only appear on the left-hand side of complex role inclusions [5].

3.27 Intersection*

Concept inclusions allow us to state that all mothers are female and that all mothers are parents, but what we really mean is that mothers are exactly the female parents. DLs support such statements by allowing us to form complex concepts such as the *intersection*[33] (also called *conjunction*) which denotes the set of individuals that are both female and parents. A complex concept can be used in axioms in exactly the same way as an atomic concept, e.g., in the equivalence $\text{Mother} \equiv \text{Female} \sqcap \text{Parent}$.

3.28 Disjunction*

A *union class expression*[34] contains all individuals that are instances of at least one class C_i for $1 \leqslant \text{i} \leqslant \text{n}$. A *union data range* contains all tuples of literals that are contained in at least one data range DR_i for $1 \leqslant \text{i} \leqslant \text{n}$. Synonyms of *disjunction* are *union* and *inclusive or*.

---

30 *R-44-PATTERN-MATCHING-ON-RDF-LITERALS*

31 *R-44-PATTERN-MATCHING-ON-RDF-LITERALS*

32 *R-55-OBJECT-PROPERTY-PATHS*

33 *R-15-CONJUNCTION-OF-CLASS-EXPRESSIONS*, *R-16-CONJUNCTION-OF-DATA-RANGES*

34 *R-17-DISJUNCTION-OF-CLASS-EXPRESSIONS*, *R-18-DISJUNCTION-OF-DATA-RANGES*

– ***DISCO-C-DISJUNCTION-01:*** Only *disco:Variable*s or *disco:Question*s or *disco:RepresentedVariable*s can have *disco:concept* relationships to *skos:Concept*s. $\text{Variable} \sqcup \text{Question} \sqcup \text{RepresentedVariable} \sqsubseteq \forall\ \text{concept.Concept}$

3.29 Negation*

A *complement class expression*[35] *ObjectComplementOf(CE)* contains all individuals that are not instances of the class expression *CE*.

3.30 Existential Quantifications*

An *existential class expression*[36] (*existential restriction* in DL terminology) contains all those individuals that are connected by the property P to an individual x that is an instance of the class C or to literals that are in the data range DR.

– ***DISCO-C-EXISTENTIAL-QUANTIFICATIONS-01:*** There must be at least one *disco:universe* relationship from *disco:Studies* or *disco:StudyGroups* to *disco:Universe* ($\texttt{Study} \sqcup \texttt{StudyGroup} \sqsubseteq \exists\ \texttt{universe.Universe}$).

– ***DATA-CUBE-C-EXISTENTIAL-QUANTIFICATIONS-01:*** Dimensions have range (*IC-4* [3]) - Every dimension declared in a *qb:DataStructureDefinition* must have a declared *rdfs:range*.
– ***DATA-CUBE-C-EXISTENTIAL-QUANTIFICATIONS-02:*** Concept dimensions have code lists (*IC-5* [3]) - Every dimension with range *skos:Concept* must have a *qb:codeList*.
– ***DATA-CUBE-C-EXISTENTIAL-QUANTIFICATIONS-03:*** Slice dimensions complete (*IC-10* [3]) - Every *qb:Slice* must have a value for every dimension declared in its *qb:sliceStructure*.
– ***DATA-CUBE-C-EXISTENTIAL-QUANTIFICATIONS-04:*** All dimensions required (*IC-11* [3]) - Every *qb:Observation* has a value for each dimension declared in its associated *qb:DataStructureDefinition*.
– ***DATA-CUBE-C-EXISTENTIAL-QUANTIFICATIONS-05:*** DSD includes measure (*IC-3* [3]) - Every *qb:DataStructureDefinition* must include (*qb:component*, *qb:componentProperty*) at least one declared measure.
– ***DATA-CUBE-C-EXISTENTIAL-QUANTIFICATIONS-06:*** Slice Keys must be declared (*IC-7* [3]) - Every *qb:SliceKey* must be associated with (*qb:sliceKey*) a *qb:DataStructureDefinition* ($\texttt{SliceKey} \sqsubseteq \exists\ \texttt{sliceKey}^{-}\texttt{.DataStructureDefinition}$).

3.31 Universal Quantifications*

A *universal class expression*[37] (*value restriction* in DL) contains all those individuals that are connected by an object property only to individuals that are instances of a particular class.

[35] *R-19-NEGATION-OF-CLASS-EXPRESSIONS*, *R-20-NEGATION-OF-DATA-RANGES*
[36] *R-86-EXISTENTIAL-QUANTIFICATION-ON-PROPERTIES*
[37] *R-91-UNIVERSAL-QUANTIFICATION-ON-PROPERTIES*

– ***DATA-CUBE-C-UNIVERSAL-QUANTIFICATIONS-01:*** *Universal quantifications* are defined for each *Data Cube* object and data property.
  - Severity level: ERROR
– ***DCAT-C-UNIVERSAL-QUANTIFICATIONS-01:*** *Universal quantifications* are defined for each *DCAT* object and data property. Only *dcat:Catalogs* can have *dcat:dataset* relationships to *dcat:Datasets* (`Catalog` $\sqsubseteq$ $\forall$ `dataset.Dataset`).
  - Severity level: ERROR
– ***DISCO-C-UNIVERSAL-QUANTIFICATIONS-01:*** *Universal quantifications* are defined for each *Disco* object and data property. Only *disco:LogicalDataSet*s can have *disco:aggregation* relationships to *qb:DataSet*s (`LogicalDataSet` $\sqsubseteq$ $\forall$ `aggregation.DataSet`).
  - Severity level: ERROR
– ***PHDD-C-UNIVERSAL-QUANTIFICATIONS-01:*** *Universal quantifications* are defined for each *PHDD* object and data property.
  - Severity level: ERROR
– ***SKOS-C-UNIVERSAL-QUANTIFICATIONS-01:*** *Universal quantifications* are defined for each *SKOS* object and data property.
  - Severity level: ERROR
– ***XKOS-C-UNIVERSAL-QUANTIFICATIONS-01:*** *Universal quantifications* are defined for each *XKOS* object and data property.
  - Severity level: ERROR

3.32 Minimum Unqualified Cardinality Restrictions*

A *minimum cardinality restriction*[38] contains all those individuals that are connected by a property to at least n different individuals/literals that are instances of a particular class or data range. If the class is missing, it is taken to be *owl:Thing*. If the data range is missing, it is taken to be *rdfs:Literal*. $\leqslant nR.\top$ is the minimum unqualified cardinality restriction where $n \in \mathbb{N}$ (written $\leqslant nR$ in short). For unqualified cardinality restrictions, classes respective data ranges are not stated.

3.33 Minimum Qualified Cardinality Restrictions*

A *minimum cardinality restriction*[39] contains all those individuals that are connected by a property to at least n different individuals/literals that are instances of a particular class or data range. If the class is missing, it is taken to be *owl:Thing*. If the data range is missing, it is taken to be *rdfs:Literal*. $\geqslant nR.C$ is a minimum qualified cardinality restriction where $n \in \mathbb{N}$.

[38] *R-81-MINIMUM-UNQUALIFIED-CARDINALITY-ON-PROPERTIES*, *R-211-CARDINALITY-CONSTRAINTS*

[39] *R-75-MINIMUM-QUALIFIED-CARDINALITY-ON-PROPERTIES*, *R-211-CARDINALITY-CONSTRAINTS*

– ***DATA-CUBE-C-MINIMUM-QUALIFIED-CARDINALITY-RESTRICTIONS-01:*** *Minimum Qualified Cardinality Restrictions* constraints are defined for each *Data Cube* object and data property.
  • Severity level: ERROR

– ***DCAT-C-MINIMUM-QUALIFIED-CARDINALITY-RESTRICTIONS-01:*** *Minimum Qualified Cardinality Restrictions* constraints are defined for each *DCAT* object and data property.
  • Severity level: ERROR

– ***DISCO-C-MINIMUM-QUALIFIED-CARDINALITY-RESTRICTIONS-01:*** *Minimum Qualified Cardinality Restrictions* constraints are defined for each *Disco* object and data property. A *disco:Questionnaire*, e.g., has at least one *disco:question* relationship to *disco:Questions* ($\texttt{Questionnaire} \sqsubseteq\ \geqslant 1\ \texttt{question.Question}$).
  • Severity level: ERROR

– ***PHDD-C-MINIMUM-QUALIFIED-CARDINALITY-RESTRICTIONS-01:*** *Minimum Qualified Cardinality Restrictions* constraints are defined for each *PHDD* object and data property.
  • Severity level: ERROR

3.34 Maximum Unqualified Cardinality Restrictions*

A *maximum cardinality restriction* contains all those individuals that are connected by a property to at most n different individuals/literals that are instances of a particular class or data range. If the class is missing, it is taken to be *owl:Thing*. If the data range is not present, it is taken to be *rdfs:Literal*. Unqualified means that the class respective the data range is not stated. $\geqslant nR.\top$ is a *maximum unqualified cardinality restriction*[40] where $n \in \mathbb{N}$ (written $\geqslant nR$ in short).

3.35 Maximum Qualified Cardinality Restrictions*

A *maximum cardinality restriction* contains all those individuals that are connected by a property to at most n different individuals/literals that are instances of a particular class or data range. If the class is missing, it is taken to be *owl:Thing*. If the data range is not present, it is taken to be *rdfs:Literal*. Qualified means that the class respective the data range is stated. $\leqslant nR.C$ is a *maximum qualified cardinality restriction*[41] where $n \in \mathbb{N}$.

– ***DISCO-C-MAXIMUM-QUALIFIED-CARDINALITY-RESTRICTIONS-01:*** A *disco:Variable* has at most one *disco:concept* relationship to a theoretical concept (*skos:Concept*) ($\texttt{Variable} \sqsubseteq\ \leqslant 1\ \texttt{concept.Concept}$).
  • Severity level: ERROR

[40] *R-82-MAXIMUM-UNQUALIFIED-CARDINALITY-ON-PROPERTIES, R-211-CARDINALITY-CONSTRAINTS*
[41] *R-76-MAXIMUM-QUALIFIED-CARDINALITY-ON-PROPERTIES, R-211-CARDINALITY-CONSTRAINTS*

3.36 Exact Unqualified Cardinality Restrictions*

An *exact cardinality restriction*[42] contains all those individuals that are connected by a property to exactly n different individuals that are instances of a particular class or data range. If the class is missing, it is taken to be *owl:Thing*. If the data range is not present, it is taken to be *rdfs:Literal*. Unqualified means that the class respective data range is not stated. $\geqslant nR.\top \sqcap \leqslant nR.\top$ is an exact unqualified cardinality restriction where $n \in \mathbb{N}$.

– ***DATA-CUBE-C-EXACT-UNQUALIFIED-CARDINALITY-RESTRICTIONS-01:*** Unique slice structure (*IC-9* [3]) - Each *qb:Slice* must have exactly one associated *qb:sliceStructure*.
  - Severity level: ERROR

3.37 Exact Qualified Cardinality Restrictions*

An *exact cardinality restriction*[43] contains all those individuals that are connected by a property to exactly n different individuals that are instances of a particular class or data range. If the class is missing, it is taken to be *owl:Thing*. If the data range is not present, it is taken to be *rdfs:Literal*. $\geqslant nR.C \sqcap \leqslant nR.C$ is an exact qualified cardinality restriction where $n \in \mathbb{N}$.

– ***DISCO-C-EXACT-QUALIFIED-CARDINALITY-RESTRICTIONS-01:*** A *disco:Question* has exactly 1 *disco:universe* relationship to *disco:Universe* (`Question` $\sqsubseteq$ $\geqslant$`1 universe.Universe` $\sqcap$ $\leqslant$`1 universe.Universe`).
  - Severity level: ERROR

– ***DATA-CUBE-C-EXACT-QUALIFIED-CARDINALITY-RESTRICTIONS-01:*** Unique data set (*IC-1* [3]) - Every *qb:Observation* has (*qb:dataSet*) exactly one associated *qb:DataSet* (`Observation` $\sqsubseteq$ $\geqslant$`1 dataSet.DataSet` $\sqcap$ $\leqslant$`1 dataSet.DataSet`).
  - Severity level: ERROR

– ***DATA-CUBE-C-EXACT-QUALIFIED-CARDINALITY-RESTRICTIONS-02:*** Unique DSD (*IC-2* [3]) - Every *qb:DataSet* has (*qb:structure*) exactly one associated *qb:DataStructureDefinition* (`DataSet` $\sqsubseteq$ $\geqslant$`1 structure.DataStructureDefinition` $\sqcap$ $\leqslant$`1 structure.DataStructureDefinition`).
  - Severity level: ERROR

3.38 Transitive Object Properties*

Transitivity is a special form of *complex role inclusion*. An *object property transitivity axiom*[44] states that the object property is transitive — that is, if an individual x is connected by the object property to an individual y that is connected by the object property to an individual z, then x is also connected by the object property to z.

[42] *R-80-EXACT-UNQUALIFIED-CARDINALITY-ON-PROPERTIES*, *R-211-CARDINALITY-CONSTRAINTS*

[43] *R-74-EXACT-QUALIFIED-CARDINALITY-ON-PROPERTIES*, *R-211-CARDINALITY-CONSTRAINTS*

[44] *R-63-TRANSITIVE-OBJECT-PROPERTIES*

3.39 Context-Specific Exclusive OR of Properties

Exclusive or is a logical operation that outputs true whenever both inputs differ (one is true, the other is false). Only one of multiple properties within some context (e.g. a class, a shape, or an application profile) leads to valid data[45]. This constraint is generally expressed in DL as follows: $C \sqsubseteq (\neg A \sqcap B) \sqcup (A \sqcap \neg B)$.

3.40 Context-Specific Exclusive OR of Property Groups

Exclusive or is a logical operation that outputs true whenever both inputs differ (one is true, the other is false). Only one of multiple property groups leads to valid data[46].

– ***DISCO-C-CONTEXT-SPECIFIC-EXCLUSIVE-OR-OF-PROPERTY-GROUPS-01:*** Within the context of *Disco*, *skos:Concept*s can have either *skos:definition* (when interpreted as theoretical concepts) or *skos:notation* and *skos:prefLabel* properties (when interpreted as codes and categories), but not both.

$$\begin{aligned} \text{Concept} \sqsubseteq (\neg\ \text{D} \sqcap \text{C}) \sqcup (\text{D} \sqcap \neg\ \text{C}) \\ \text{D} \equiv \text{A} \sqcap \text{B} \\ \text{A} \sqsubseteq\ \geqslant 1\ \text{notation.string} \sqcap \leqslant 1\ \text{notation.string} \\ \text{B} \sqsubseteq\ \geqslant 1\ \text{prefLabel.string} \sqcap \leqslant 1\ \text{prefLabel.string} \\ \text{C} \sqsubseteq\ \geqslant 1\ \text{definition.string} \sqcap \leqslant 1\ \text{definition.string} \end{aligned}$$

3.41 Allowed Values

It is a common requirement to narrow down the value space of a property by an exhaustive enumeration of the valid values (both literals or resources). This is often rendered in drop down boxes or radio buttons in user interfaces. *Allowed values*[47] for properties can be IRIs, IRIs (matching one or multiple patterns), (any) literals, literals of a list of allowed literals (e.g. 'red' 'blue' 'green'), typed literals of one or multiple type(s) (e.g. *xsd:string*).

– ***DISCO-C-ALLOWED-VALUES-01***. *disco:CategoryStatistics* can only have *disco:computationBase* relationships to the values *valid* and *invalid* of the datatype *rdf:langString* ($\texttt{CategoryStatistics} \equiv \forall\ \texttt{computationBase}.\{\texttt{valid,invalid}\} \sqcap \texttt{langString}$).

3.42 Not Allowed Values

A matching triple has any literal / object except those explicitly excluded[48].

---

45 *R-11-CONTEXT-SPECIFIC-EXCLUSIVE-OR-OF-PROPERTIES*

46 *R-13-DISJOINT-GROUP-OF-PROPERTIES-CLASS-SPECIFIC*

47 *R-30-ALLOWED-VALUES-FOR-RDF-OBJECTS* and *R-37-ALLOWED-VALUES-FOR-RDF-LITERALS*

48 *R-33-NEGATIVE-OBJECT-CONSTRAINTS*, *R-200-NEGATIVE-LITERAL-CONSTRAINTS*

3.43 Literal Ranges

P1 is a data property (of an instance of class *C1*) and its literal value must be between the range of $[V_{min}, V_{max}]$[49].

– ***DISCO-C-LITERAL-RANGES-01:*** *disco:percentage* (domain: *disco:CategoryStatistics*) literals must be of the datatype *xsd:double* whose range should be restricted to be between 0 and 100.

3.44 Negative Literal Ranges

P1 is a data property (of an instance of class *C1*) and its literal value must not be between the range of $[V_{min}, V_{max}]$[50].

3.45 Required Properties

Properties may be required[51].

3.46 Optional Properties

Properties may be optional[52].

3.47 Repeatable Properties

Properties may be repeatable[53].

3.48 Negative Property Constraints

Instances of a specific class must not have some object property[54].

3.49 Individual Equality*

Individual equality[55] states that two different names are known to refer to the same individual [5].

---

[49] *R-45-RANGES-OF-RDF-LITERAL-VALUES*

[50] *R-142-NEGATIVE-RANGES-OF-RDF-LITERAL-VALUES*

[51] *R-68-REQUIRED-PROPERTIES*

[52] *R-69-OPTIONAL-PROPERTIES*

[53] *R-70-REPEATABLE-PROPERTIES*

[54] *R-52-NEGATIVE-OBJECT-PROPERTY-CONSTRAINTS*, *R-53-NEGATIVE-DATA-PROPERTY-CONSTRAINTS*

[55] *R-6-EQUIVALENT-INDIVIDUALS*

3.50 Functional Properties*

An *object property functionality axiom*[56] *FunctionalObjectProperty(OPE)* states that the object property expression *OPE* is functional — that is, for each individual x, there can be at most one distinct individual y such that x is connected by *OPE* to y. Each such axiom can be seen as a syntactic shortcut for the following axiom: *SubClassOf(owl:Thing ObjectMaxCardinality(1 OPE))*.

3.51 Inverse-Functional Properties*

An *object property inverse functionality axiom*[57] *InverseFunctionalObjectProperty(OPE)* states that the object property expression *OPE* is inverse-functional - that is, for each individual x, there can be at most one individual y such that y is connected by *OPE* with x. Each such axiom can be seen as a syntactic shortcut for the following axiom: *SubClassOf(owl:Thing ObjectMaxCardinality(1 ObjectInverseOf(OPE)))*.

– ***DISCO-C-INVERSE-FUNCTIONAL-PROPERTIES-01:*** for each *rdfs:Resource* x, there can be at most one distinct *rdfs:Resource* y such that y is connected by *adms:identifier* to x (`funct identifier`$^-$).

Thomas: Achim, we created an open issue for DDI 4 how keys should be defined

3.52 Value Restrictions*

Individual Value Restrictions[58]: A has-value class expression *ObjectHasValue(OPE a)* consists of an object property expression *OPE* and an individual a, and it contains all those individuals that are connected by *OPE* to a. Each such class expression can be seen as a syntactic shortcut for the class expression *ObjectSomeValuesFrom(OPE ObjectOneOf(a))*. *Literal Value Restrictions*: A has-value class expression *DataHasValue(DPE lt)* consists of a data property expression *DPE* and a literal *lt*, and it contains all those individuals that are connected by *DPE* to *lt*. Each such class expression can be seen as a syntactic shortcut for the class expression *DataSomeValuesFrom(DPE DataOneOf(lt))*.

3.53 Self Restrictions*

A *self-restriction ObjectHasSelf(OPE)* consists of an object property expression *OPE*, and it contains all those individuals that are connected by *OPE* to themselves.

[56] *R-57-FUNCTIONAL-OBJECT-PROPERTIES*
[57] *R-58-INVERSE-FUNCTIONAL-OBJECT-PROPERTIES*
[58] *R-88-VALUE-RESTRICTIONS*

3.54 Primary Key Properties

The *Primary Key Properties*[59] constraint is often useful to declare a given (datatype) property as the "primary key" of a class, so that a system can enforce uniqueness and also automatically build URIs from user input and data imported from relational databases or spreadsheets.. Starfleet officers, e.g., are uniquely identified by their command authorization code (e.g. to activate and cancel auto-destruct sequences). It means that the property *commandAuthorizationCode* is inverse functional - mapped to DL as follows: (`funct commandAuthorizationCode`$^{-}$) Keys, however, are even more general, i.e., a generalization of inverse functional properties [7]. A key can be a datatype property, an object property, or a chain of properties. For this generalization purposes, as there are different sorts of key, and as keys can lead to undecidability, DL is extended with *key boxes* and a special *keyfor* construct[6]. This leads to the following DL mapping (only one simple property constraint): `commandAuthorizationCode keyfor StarfleetOfficer`

– see *inverse-functional properties*

3.55 Class-Specific Property Range*

Class-Specific Property Range[60] restricts the range of object and data properties for individuals within a specific context (e.g. class, shape, application profile). The values of each member property of a class may be limited by their value type, such as *xsd:string* or *foaf:Person.*

– ***DISCO-C-CLASS-SPECIFIC-PROPERTY-RANGE-01:*** Only *disco:Question*s can have *disco:questionText* relationships to literals of the datatype *rdf:langString* ($\neg$`Question` $\sqsubseteq$ $\neg\exists$ `questionText.langString`).

3.56 Class-Specific Reflexive Object Properties*

Using DL terminology *Class-Specific Reflexive Object Properties* is called local reflexivity - a set of individuals (of a specific class) that are related to themselves via a given role [5].

3.57 Membership in Controlled Vocabularies.

Resources can only be members of listed controlled vocabularies[61].

[59] R-226-PRIMARY-KEY-PROPERTIES
[60] *R-29-CLASS-SPECIFIC-RANGE-OF-RDF-OBJECTS*, *R-36-CLASS-SPECIFIC-RANGE-OF-RDF-LITERALS*
[61] *R-32-MEMBERSHIP-OF-RDF-OBJECTS-IN-CONTROLLED-VOCABULARIES*, *R-39-MEMBERSHIP-OF-RDF-LITERALS-IN-CONTROLLED-VOCABULARIES*

– ***DISCO-C-MEMBERSHIP-IN-CONTROLLED-VOCABULARIES-01***: *disco:SummaryStatistics* can only have *disco:summaryStatisticType* relationships to *skos:Concept*s which must be members of the controlled vocabulary *ddicv:SummaryStatisticType* which is a *skos:ConceptScheme*.

$$\text{SummaryStatistics} \sqsubseteq \forall summaryStatisticType.A$$

$$A \equiv Concept \sqcap \forall inScheme.B$$

$$B \equiv ConceptScheme \sqcap \{SummaryStatisticType\}$$

– ***DATA-CUBE-C-MEMBERSHIP-IN-CONTROLLED-VOCABULARIES-01:*** Codes from code list (*IC-19* [3]) - If a dimension property has a *qb:codeList*, then the value of the dimension property on every *qb:Observation* must be in the code list.

3.58 IRI Pattern Matching

IRI pattern matching applied on subjects, properties, and objects[62].

– ***DISCO-C-IRI-PATTERN-MATCHING***: *disco:Study* resources must match a given IRI pattern.

3.59 Literal Value Comparison

Depending on the property semantics, there are cases where two different literal values must have a specific ordering with respect to an operator. *P1* and *P2* are the datatype properties we need to compare and *OP* is the comparison operator (<, <=, >, >=, =, !=)[63]. The *COMP Pattern*, one of the Data Quality Test Patterns, can be used to validate the *Literal Value Comparison* constraint [4]:

```
SELECT ?s WHERE {
    ?s %%P1%% ?v1 .
    ?s %%P2%% ?v2 .
    FILTER ( ?v1 %%OP%% ?v2 ) }
```

– ***DISCO-C-LITERAL-VALUE-COMPARISON-01***: *disco:startDate*s must be before ('<') *disco:endDate*s. To validate this constraint we bind the variables as follows (P1: *disco:startDate*, P2: *disco:endDate*, OP: <).

3.60 Ordering

With this constraint objects of object properties can be ordered as well as literals of data properties[64].

---

[62] *R-21-IRI-PATTERN-MATCHING-ON-RDF-SUBJECTS*, *R-22-IRI-PATTERN-MATCHING-ON-RDF-OBJECTS*, *R-23-IRI-PATTERN-MATCHING-ON-RDF-PROPERTIES*

[63] *R-43-LITERAL-VALUE-COMPARISON*

[64] *R-121-SPECIFY-ORDER-OF-RDF-RESOURCES*, *R-217-DEFINE-ORDER-FOR-FORMS/DISPLAY*

In DDI, variables, questions, and codes/categories are typically organized in a particular order. For obtaining this order, *skos:OrderedCollection* resources are used.

- ***DISCO-C-ORDERING-01***: If *disco:Variable*s of a given *disco:LogicalDataSet* should be ordered, a collection of variables must be present in the data and connected with the data set. The collection of variables is of the type *skos:OrderedCollection* containing multiple variables (each represented as *skos:Concept*) in a *skos:memberList*.
- ***DISCO-C-ORDERING-02***: If *disco:Question*s of a given *disco:Questionnaire* should be ordered, a collection of questions must be present in the data and connected with the questionnaire. The collection of questions is of the type *skos:OrderedCollection* containing multiple questions (each represented as *skos:Concept*) in a *skos:memberList*.
- ***DISCO-C-ORDERING-03***: If codes/categories (*skos:Concept*s) of a given *disco:Representation* of a given *disco:Variable* should be ordered, the variable representation should also be of the type *skos:OrderedCollection* containing multiple codes/categories (each represented as *skos:Concept*) in a *skos:memberList*.

3.61 Validation Levels

Different levels of severity (priority)[65] should be assigned to constraints. Possible validation levels could be: informational, warning, error, fail, should, recommended, must, may, optional, closed (only this) constraints, open (at least this) constraint.

For *Disco* each constraint should be assigned to exactly one *validation level*.

3.62 String Operations

Many different *string operations*[66] are possible. Some constraints require building new strings out of other strings. Calculating the string length would also be another constraint of this type.

- ***DISCO-C-STRING-OPERATIONS-01***: The title of a study (*dcterms:title*) (e.g. 'EU-SILC 2005') may be calculated out of the title of the containing series (*dcterms:title*) (e.g. 'EU-SILC') and the human-readable label of the study (*rdfs:label*) (e.g. '2005').

[65] *R-205-VARYING-LEVELS-OF-ERROR*, *R-135-CONSTRAINT-LEVELS*, *R-158-SEVERITY-LEVELS-OF-CONSTRAINT-VIOLATIONS*, *R-193-MULTIPLE-CONSTRAINT-VALIDATION-EXECUTION-LEVELS*

[66] *R-194-PROVIDE-STRING-FUNCTIONS-FOR-RDF-LITERALS*

3.63 Context-Specific Valid Classes

What types are valid in a specific context?[67] Context can be an input stream, a data creation function, or an API.

– ***DATA-CUBE-C-CONTEXT-SPECIFIC-VALID-CLASSES-01***: For future versions of *Data Cube*, out-dated classes can be marked as deprecated.
  • severity level: RECOMMENDED

– ***DCAT-C-CONTEXT-SPECIFIC-VALID-CLASSES-01***: For future versions of *DCAT*, out-dated classes can be marked as deprecated.
  • severity level: RECOMMENDED

– ***DISCO-C-CONTEXT-SPECIFIC-VALID-CLASSES-01***: For future versions of *Disco*, out-dated classes can be marked as deprecated.
  • severity level: RECOMMENDED

– ***PHDD-C-CONTEXT-SPECIFIC-VALID-CLASSES-01***: For future versions of *PHDD*, out-dated classes can be marked as deprecated.
  • severity level: RECOMMENDED

– ***SKOS-C-CONTEXT-SPECIFIC-VALID-CLASSES-01***: For future versions of *SKOS*, out-dated classes can be marked as deprecated.
  • severity level: RECOMMENDED

– ***XKOS-C-CONTEXT-SPECIFIC-VALID-CLASSES-01***: For future versions of *XKOS*, out-dated classes can be marked as deprecated.
  • severity level: RECOMMENDED

3.64 Context-Specific Valid Properties

What properties can be used within this context?[68] Context can be an data receipt function, data creation function, or API.

– ***DATA-CUBE-C-CONTEXT-SPECIFIC-VALID-PROPERTIES-01***: For future versions of *Data Cube*, out-dated properties can be marked as deprecated.
  • severity level: RECOMMENDED

– ***DCAT-C-CONTEXT-SPECIFIC-VALID-PROPERTIES-01***: For future versions of *DCAT*, out-dated properties can be marked as deprecated.
  • severity level: RECOMMENDED

– ***DISCO-C-CONTEXT-SPECIFIC-VALID-PROPERTIES-01***: For future versions of *Disco*, out-dated properties can be marked as deprecated.
  • severity level: RECOMMENDED

67 *R-209-VALID-CLASSES*
68 *R-210-VALID-PROPERTIES*

– ***PHDD-C-CONTEXT-SPECIFIC-VALID-PROPERTIES-01***: For future versions of *PHDD*, out-dated properties can be marked as deprecated.
  - severity level: RECOMMENDED

– ***SKOS-C-CONTEXT-SPECIFIC-VALID-PROPERTIES-01***: For future versions of *SKOS*, out-dated properties can be marked as deprecated.
  - severity level: RECOMMENDED

– ***XKOS-C-CONTEXT-SPECIFIC-VALID-PROPERTIES-01***: For future versions of *XKOS*, out-dated properties can be marked as deprecated.
  - severity level: RECOMMENDED

3.65 Default Values*

Default values[69] for objects and literals are inferred automatically. It should be possible to declare the default value for a given property, e.g. so that input forms can be pre-populated and to insert a required property that is missing in a web service call.

– ***DISCO-C-DEFAULT-VALUES-01:*** The value 'true' for the property *disco:isPublic* (*xsd:boolean*) indicates that the data set (*disco:LogicalDataSet*) can be accessed (usually downloaded) by anyone. Per default, access to data sets should be restricted ('false').

3.66 Mathematical Operations

Examples for *Mathematical Operations*[70] are the addition of two dates, the addition of days to a start date, and statistical computations (e.g. average, mean, sum).

– ***DISCO-C-MATHEMATICAL-OPERATIONS-01:*** The sum of *disco:percentage* (datatype: *xsd:double*) values of all codes (represented as *skos:Concept*s) of a code list (*skos:ConceptScheme* or *skos:OrderedCollection*), serving as representation of a particular *disco:Variable*, must exactly be 100.
  - severity level: ERROR

– ***DISCO-C-MATHEMATICAL-OPERATIONS-02***: For a given variable, the sum of all category statistics (frequency) has to be equal to the 'total' value of summary statistics.
  - severity level: ERROR

– ***DISCO-C-MATHEMATICAL-OPERATIONS-03***: For a given variable, the sum of 'valid total' and 'missing total' has to be equal to 'total'.
  - severity level: ERROR

[69] *R-31-DEFAULT-VALUES-OF-RDF-OBJECTS*, *R-38-DEFAULT-VALUES-OF-RDF-LITERALS*

[70] *R-42-MATHEMATICAL-OPERATIONS*, *R-41-STATISTICAL-COMPUTATIONS*

– ***DISCO-C-MATHEMATICAL-OPERATIONS-04***: For a given variable, the 'total' value for country 'All' must be equal to the sum of the 'total' value for all Countries.
  - severity level: ERROR
– ***DISCO-C-MATHEMATICAL-OPERATIONS-05***: Minimum values do not have to be greater than maximum values (*disco:SummaryStatistics*).
  - severity level: ERROR

3.67 Language Tag Matching

For particular data properties, values must be stated for predefined languages[71].

– ***DISCO-C-LANGUAGE-TAG-MATCHING:*** There must be an English variable name (*skos:notation*) for each *disco:Variable* within *disco:LogicalDataSet*s.

3.68 Language Tag Cardinality

For particular data properties, values of predefined languages must be stated for determined number of times[72].

– ***DISCO-C-LANGUAGE-TAG-CARDINALITY-01:*** There must be at least one English *disco:questionText* for each *disco:Question* within *disco:LogicalDataSet*s.
– ***DISCO-C-LANGUAGE-TAG-CARDINALITY-02:*** There should be at most one English literal value for variable names (*skos:notation*, domain: *disco:Variable*).
– ***DISCO-C-LANGUAGE-TAG-CARDINALITY-03***: For each question (*disco:Question*), there must be at least one question text (*disco:questionText*) associated with a language tag of an arbitrary language or with an English language tag.

– ***SKOS-C-LANGUAGE-TAG-CARDINALITY-01[73]:*** Omitted or Invalid Language Tags: Some controlled vocabularies contain literals in natural language, but without information what language has actually been used. Language tags might also not conform to language standards, such as RFC 3066.
  - Implementation: Iteration over all triples in the vocabulary that have a predicate which is a (subclass of) *rdfs:label* or *skos:note*.
  - Severity level: WARNING

[71] *R-47-LANGUAGE-TAG-MATCHING*
[72] *R-49-RDF-LITERALS-HAVING-AT-MOST-ONE-LANGUAGE-TAG*, *R-48-MISSING-LANGUAGE-TAGS*
[73] Corresponds to qSKOS Quality Issues - Labeling and Documentation Issues - Omitted or Invalid Language Tags

– ***SKOS-C-LANGUAGE-TAG-CARDINALITY-02***[74]***:*** Incomplete Language Coverage: Some concepts in a thesaurus are labeled in only one language, some in multiple languages. It may be desirable to have each concept labeled in each of the languages that also are used on the other concepts. This is not always possible, but incompleteness of language coverage for some concepts can indicate shortcomings of the vocabulary.
  - Severity level: INFO

– ***SKOS-C-LANGUAGE-TAG-CARDINALITY-03***[75]***:*** No Common Language: Checks if all concepts have at least one common language, i.e. they have assigned at least one literal in the same language.
  - Severity level: INFO

– ***SKOS-C-LANGUAGE-TAG-CARDINALITY-04***[76]***:*** Inconsistent Preferred Labels: According to the SKOS reference document, "A resource has no more than one value of skos:prefLabel per language tag".
  - Implementation: A SPARQL query is used to find concepts with at least two prefLabels. In a second step, the language tags of these prefLabels are analyzed and an ambiguity is detected if they are equal.
  - Severity level: INFO

3.69 Whitespace Handling

Avoid whitespaces in literals neither leading nor trailing white spaces[77].

– ***DISCO-C-WHITESPACE-HANDLING-01:*** Delete whitespaces of series and study abstracts (*dcterms:abstract*; domain: *disco:StudyGroup*, *disco:Study*) automatically.

3.70 HTML Handling

Check if all HTML tags, included in literals (of specific data properties within the context of specific classes)[78], are closed properly.

– ***DISCO-C-HTML-HANDLING-01:*** Check if all HTML tags, included in literals of all *Disco* data properties, are closed properly.

– ***DISCO-C-HTML-HANDLING-02:*** Check if all HTML tags, included in literals of all data properties whose domains are *Disco* classes, are closed properly.

3.71 Conditional Properties

If specific properties exist, then specific other properties must also be present[79].

[74] Corresponds to qSKOS Quality Issues - Labeling and Documentation Issues - Incomplete Language Coverage

[75] Corresponds to qSKOS Quality Issues - Labeling and Documentation Issues - No Common Language

[76] Corresponds to qSKOS Quality Issues - SKOS Semi-Formal Consistency Issues - Inconsistent Preferred Labels

[77] *R-50-WHITESPACE-HANDLING-OF-RDF-LITERALS*

[78] *R-51-HTML-HANDLING-OF-RDF-LITERALS*

[79] *R-71-CONDITIONAL-PROPERTIES*

– ***DISCO-C-CONDITIONAL-PROPERTIES-01***: If a *skos:Concept* represents a code (having a *skos:notation* property) and a category (having a *skos:prefLabel* property), then the property *disco:isValid* has to be stated indicating if the code is valid ('true') or missing ('false').
– ***DISCO-C-CONDITIONAL-PROPERTIES-02***: If the abstract (*dcterms:abstract*) of a *disco:Study* is missing, a study title (*dcterms:title*) has to be stated.

3.72 Recommended Properties

Which properties are not necessarily required but recommended within a particular context[80].

– ***DATA-CUBE-C-RECOMMENDED-PROPERTIES-01:***
  • severity level: RECOMMENDED

– ***DCAT-C-RECOMMENDED-PROPERTIES-01:***
  • severity level: RECOMMENDED

– ***DISCO-C-RECOMMENDED-PROPERTIES-01:*** The property *skos:notation* is not mandatory for *disco:Variable*s, but recommended to indicate variable names.
  • severity level: RECOMMENDED

– ***PHDD-C-RECOMMENDED-PROPERTIES-01:***
  • severity level: RECOMMENDED

– ***SKOS-C-RECOMMENDED-PROPERTIES-01:***
  • severity level: RECOMMENDED

– ***XKOS-C-RECOMMENDED-PROPERTIES-01:***
  • severity level: RECOMMENDED

3.73 Handle RDF Collections

Examples of the *Handle RDF Collections*[81] constraint are: a collection must have a specific size; the first/last element of a given list must be a specific literal; the elements of collections are compared; are collections identical?; actions on RDF lists[82]; the 2. list element must be equal to 'XXX'; does the list have more than 10 elements?

– ***DISCO-C-HANDLE-RDF-COLLECTIONS-01***: Have comparable *disco:Variable*s the same number of codes in their code lists?
– ***DISCO-C-HANDLE-RDF-COLLECTIONS-02***: Does the actual number of *disco:Variable*s within an (un)ordered collection of a given *disco:LogicalDataSet* match the expected number?

---

80 *R-72-RECOMMENDED-PROPERTIES*
81 *R-120-HANDLE-RDF-COLLECTIONS*
82 See http://www.snee.com/bobdc.blog/2014/04/rdf-lists-and-sparql.html

3.74 Value is Valid for Datatype

Make sure that a value is valid for its datatype. It has to be ensured, e.g., that a date is really a date, or that a *xsd:nonNegativeInteger* value is not negative.

- ***DISCO-C-VALUE-IS-VALID-FOR-DATATYPE-01***: Check if all literal values of properties used within the *Disco* context of the datatype *xsd:date* (e.g. *disco:startDate*, *disco:endDate*, *dcterms:date*) are really of the datatype *xsd:date*.

- ***DATA-CUBE-C-VALUE-IS-VALID-FOR-DATATYPE-01:*** Datatype consistency (*IC-0* [3]) - The RDF graph must be consistent under RDF D-entailment using a datatype map containing all the datatypes used within the graph.

3.75 Use Sub-Super Relations in Validation

83

The validation of instances data (direct or indirect) exploits the sub-class or sub-property link in a given ontology. This validation can indicate when the data is verbose (redundant) or expressed at a too general level, and could be improved. If *dcterms:date* and one of its sub-properties *dcterms:created* or *dcterms:issued* are present, e.g., check that the value in *dcterms:date* is not redundant with *dcterms:created* or *dcterms:issued* for ingestion.

- ***DISCO-C-USE-SUB-SUPER-RELATIONS-IN-VALIDATION-01***: If one or more *dcterms:coverage* properties are present, suggest the use of one of its sub-properties *dcterms:spatial* or *dcterms:temporal*.
- ***DISCO-C-USE-SUB-SUPER-RELATIONS-IN-VALIDATION-02***: If the *dcterms:contributor* property is present, suggest the use of one of its sub-properties, e.g. *disco:fundedBy*.

3.76 Cardinality Shortcuts*

In most library applications, cardinality shortcuts tend to appear in pairs, with repeatable/non-repeatable establishing maximum cardinality and optional/mandatory establishing minimum cardinality. These are shortcuts for more detailed *cardinality restrictions*.

3.77 Aggregation

Some constraints require aggregating multiple values, especially via *COUNT*, *MIN* and *MAX*.

- ***DISCO-C-AGGREGATION-01***: calculate the number of theoretical concepts in the thematic classification of a given study.

[83] *R-224-USE-SUB-SUPER-RELATIONS-IN-VALIDATION*

- ***DISCO-C-AGGREGATION-02***: calculate the number of variables of a data set.
- ***DISCO-C-AGGREGATION-03***: calculate the number of questions in a given questionnaire.
- ***DISCO-C-AGGREGATION-04***: the number of codes of a given variable must be below a maximum value.
- ***DISCO-C-AGGREGATION-05***: the number of questions of a given questionnaire must exactly be a given value.
- ***DISCO-C-AGGREGATION-06***: the cumulative percentage of all codes of a given variable must be 100.
- ***DISCO-C-AGGREGATION-07***: the absolute frequency of all valid codes of a given variable must be equal to a given value.

3.78 Structure

SKOS is based on RDF, which is a graph-based data model. Therefore we can concentrate on the vocabulary's graph-based structure for assessing the quality of SKOS vocabularies and apply graph- and network-analysis techniques.

- ***DISCO-C-STRUCTURE-01***: there must be exactly one root in the hierarchy of DDI concepts.

- ***DATA-CUBE-C-STRUCTURE-01:*** Codes from hierarchy (*IC-20* [3]) - If a dimension property has a *qb:HierarchicalCodeList* with a non-blank *qb:parentChildProperty* then the value of that dimension property on every *qb:Observation* must be reachable from a root of the hierarchy using zero or more hops along the *qb:parentChildProperty* links.
- ***DATA-CUBE-C-STRUCTURE-02:*** Codes from hierarchy (inverse) (*IC-21* [3]) - If a dimension property has a *qb:HierarchicalCodeList* with an inverse *qb:parentChildProperty* then the value of that dimension property on every *qb:Observation* must be reachable from a root of the hierarchy using zero or more hops along the inverse *qb:parentChildProperty* links.

- ***SKOS-C-STRUCTURE-01[84]:*** Orphan Concepts: An orphan concept is a concept without any associative or hierarchical relations. It might have attached literals like e.g., labels, but is not connected to any other resource, lacking valuable context information. A controlled vocabulary that contains many orphan concepts is less usable for search and retrieval use cases, because, e.g., no hierarchical query expansion can be performed on search terms to find documents with more general content.
  - Implementation: Iteration over all concepts in the vocabulary and returning that don't have associated resources using (sub-properties of) *skos:semanticRelation*.
  - Severity level: WARNING

[84] Corresponds to qSKOS Quality Issues - Structural Issues - Orphan Concepts

- ***SKOS-C-STRUCTURE-02***[85]***:*** Disconnected Concept Clusters: Checking the connectivity of the graph, it is possible to identify all weakly connected components. These datasets form "islands" in the vocabulary and might be caused by incomplete data acquisition, "forgotten" test data, outdated terms and the like.
  - Implementation: Creation of an undirected graph that includes all non-orphan concepts as nodes and all semantic relations as edges. Tarjan's algorithm then finds and returns all weakly connected components.
  - Severity level: INFO
- ***SKOS-C-STRUCTURE-03***[86]***:*** Cyclic Hierarchical Relations: Although perfectly consistent with the SKOS data model, cyclic relations may reveal a logical problem in the thesaurus. Consider the following example: "decision" → "problem resolution" → "problem" (→ "decision": here the cycle is closed). The concepts are connected using *skos:broader* relationships (indicated with "→"). Due to the fact that a thesaurus is in many cases a product of consensus between the contributors (or just the decision of one dedicated thesaurus manager), it will be almost impossible to automatically resolve the cycle (i.e. deleting an edge).
  - Implementation: Construction of a graph having all concepts as nodes and the set of edges being *skos:broader* relations.
  - Severity level: WARNING
- ***SKOS-C-STRUCTURE-04***[87]***:*** Valueless Associative Relations: Two concepts are sibling, but also connected by an associative relation. In that context, the associative relation is not necessary. See ISO_DIS_25964-1, 11.3.2.2
  - Implementation: Identification of all pairs of concepts that have the same broader or narrower concepts, i.e. they are "sibling terms". All siblings that are related by a *skos:related* property are returned.
  - Severity level: INFO
- ***SKOS-C-STRUCTURE-05***[88]***:*** Solely Transitively Related Concepts: *skos:broaderTransitive* and *skos:narrowerTransitive* are, according to the SKOS reference document, "not used to make assertions", so they should not be the only relations hierarchically relating two concepts.
  - Implementation: Identification of all concept pairs that are related by *skos:broaderTransitive* or *skos:narrowerTransitive* properties but not by their *skos:broader* and *skos:narrower* subproperties.
  - Severity level: INFO

[85] Corresponds to qSKOS Quality Issues - Structural Issues - Disconnected Concept Clusters
[86] Corresponds to qSKOS Quality Issues - Structural Issues - Cyclic Hierarchical Relations
[87] Corresponds to qSKOS Quality Issues - Structural Issues - Valueless Associative Relations
[88] Corresponds to qSKOS Quality Issues - Structural Issues - Solely Transitively Related Concepts

- ***SKOS-C-STRUCTURE-06*[89]*:*** Unidirectionally Related Concepts: Reciprocal relations (e.g., *broader/narrower, related, hasTopConcept/topConceptOf*) should be included in the controlled vocabularies to achieve better search results using SPARQL in systems without reasoner support.
  - Implementation: This issue is checked without inference of *owl:inverseOf* properties. We iterate over all triples and check for each property if an inverse property is defined in the SKOS ontology and if the respective statement using this property is included in the vocabulary. If not, the resources associated with this property are returned.
  - Severity level: INFO
- ***SKOS-C-STRUCTURE-07*[90]*:*** Omitted Top Concepts: A vocabulary should provide "entry points" to the data to provide "efficient access" (SKOS primer) and guidance for human users.
  - Implementation: For every ConceptScheme in the controlled vocabulary, a SPARQL query is issued finding resources that are associated with this ConceptScheme by one of the properties *skos:hasTopConcept* or *skos:topConceptOf*. Top concepts are also concepts having no broader concept.
  - Severity level: WARNING
- ***SKOS-C-STRUCTURE-08*[91]*:*** Top Concepts Having Broader Concepts: Concepts "internal to the tree" should not be indicated as top concepts.
  - Implementation: A SPARQL query finds all top concepts (being defined by one of the properties *skos:hasTopConcept* or *skos:topConceptOf*) having associated a broader concept.
  - Severity level: ERROR
- ***SKOS-C-STRUCTURE-09*[92]*:*** Hierarchical Redundancy: As stated in the SKOS reference document, *skos:broader* and *skos:narrower* are not transitive properties. However, they are sub-properties of *skos:broaderTransitive* and *skos:narrowerTransitive* which enables inference of a "transitive closure". This, in fact, leaves it up to the user to interpret whether a vocabulary's hierarchical structure is seen as transitive or not. In the former case, this check can be useful. It finds pairs of concepts (A,B) that are directly hierarchically related but there also exits an hierarchical path through a concept C that connects A and B.
  - Severity level: INFO
- ***SKOS-C-STRUCTURE-10*[93]*:*** Reflexive Relations: Concepts related to themselves.
  - Severity level: WARNING

[89] Corresponds to qSKOS Quality Issues - Structural Issues - Unidirectionally Related Concepts
[90] Corresponds to qSKOS Quality Issues - Structural Issues - Omitted Top Concepts
[91] Corresponds to qSKOS Quality Issues - Structural Issues - Top Concepts Having Broader Concepts
[92] Corresponds to qSKOS Quality Issues - Structural Issues - Hierarchical Redundancy
[93] Corresponds to qSKOS Quality Issues - Structural Issues - Reflexive Relations

3.79 Labeling and Documentation

- ***SKOS-C-LABELING-AND-DOCUMENTATION-01[94]:*** Undocumented Concepts: The SKOS standard defines a number of properties useful for documenting the meaning of the concepts in a thesaurus also in a human-readable form. Intense use of these properties leads to a well-documented thesaurus which should also improve its quality.
  - Implementation: Iteration over all concepts in the vocabulary and find those not using one of *skos:note*, *skos:changeNote*, *skos:definition*, *skos:editorialNote*, *skos:example*, *skos:historyNote*, or *skos:scopeNote*.
  - Severity level: INFO
- ***SKOS-C-LABELING-AND-DOCUMENTATION-02[95]:*** Overlapping Labels: This is a generalization of a recommendation in the SKOS primer, that "no two concepts have the same preferred lexical label in a given language when they belong to the same concept scheme". This could indicate missing disambiguation information and thus lead to problems in autocompletion application.
  - Severity level: INFO
- ***SKOS-C-LABELING-AND-DOCUMENTATION-03[96]:*** Missing Labels: To make the vocabulary more convenient for humans to use, instances of SKOS classes (Concept, ConceptScheme, Collection) should be labeled using e.g., *skos:prefLabel*, *altLabel*, *rdfs:label*, *dc:title*.
  - Severity level: INFO
- ***SKOS-C-LABELING-AND-DOCUMENTATION-04[97]:*** Unprintable Characters in Labels: *pref/alt/hiddenlabels* contain characters that are not alphanumeric characters or blanks.
  - Severity level: INFO
- ***SKOS-C-LABELING-AND-DOCUMENTATION-05[98]:*** Empty Labels: Labels also need to contain textual information to be useful, thus we find all SKOS labels with length 0 (after removing whitespaces).
  - Severity level: INFO
- ***SKOS-C-LABELING-AND-DOCUMENTATION-06[99]:*** Ambiguous Notation References: Concepts within the same concept scheme should not have identical *skos:notation* literals.
  - Severity level: INFO

[94] Corresponds to qSKOS Quality Issues - Labeling and Documentation Issues - Undocumented Concepts

[95] Corresponds to qSKOS Quality Issues - Labeling and Documentation Issues - Overlapping Labels

[96] Corresponds to qSKOS Quality Issues - Labeling and Documentation Issues - Missing Labels

[97] Corresponds to qSKOS Quality Issues - Labeling and Documentation Issues - Unprintable Characters in Labels

[98] Corresponds to qSKOS Quality Issues - Labeling and Documentation Issues - Empty Labels

[99] Corresponds to qSKOS Quality Issues - Labeling and Documentation Issues - Ambiguous Notation References

3.80 Vocabulary

Vocabularies should not invent any new terms or use deprecated elements.

- ***DATA-CUBE-C-VOCABULARY-01***
  - Severity level: FATALERROR
- ***DCAT-C-VOCABULARY-01***
  - Severity level: FATALERROR
- ***DISCO-C-VOCABULARY-01***
  - Severity level: FATALERROR
- ***PHDD-C-VOCABULARY-01***
  - Severity level: FATALERROR
- ***SKOS-C-VOCABULARY-01[100]:*** Undefined SKOS Resources: The vocabulary should not invent any new terms within the SKOS namespace or use deprecated SKOS elements.
  - Severity level: FATALERROR
- ***XKOS-C-VOCABULARY-01***
  - Severity level: FATALERROR

3.81 Linked Data

- ***R-HTTP-URI-SCHEME-VIOLATION[101]***: In the context of Linked Data, we restrict ourselves to using HTTP URIs only and avoid other URI schemes such as URNs and DOIs.
  - Severity level: ERROR

4 SKOS-Related Constraint Types

5 Validation in Combination with other Vocabularies

5.1 RDF Data Cube Vocabulary

5.2 SKOS and XKOS

5.3 PHDD

5.4 DCAT

6 Conclusion and Future Work

References

1. Thomas Bosch and Kai Eckert. Requirements on rdf constraint formulation and validation. *Proceedings of the DCMI International Conference on Dublin Core and Metadata Applications (DC 2014)*, 2014.

[100] Corresponds to qSKOS Quality Issues - Linked Data Specific Issues - Undefined SKOS Resources

[101] Corresponds to qSKOS Quality Issues - Linked Data Specific Issues - HTTP URI Scheme Violation

2. Thomas Bosch, Andreas Nolle, Erman Acar, and Kai Eckert. Rdf validation requirements - evaluation and logical underpinning. 2015.
3. Richard Cyganiak and Dave Reynolds. The rdf data cube vocabulary. W3C recommendation, W3C, January 2014.
4. Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, Roland Cornelissen, and Amrapali Zaveri. Test-driven evaluation of linked data quality. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 747–758, Republic and Canton of Geneva, Switzerland, 2014. International World Wide Web Conferences Steering Committee.
5. Markus Krötzsch, Frantisek Simancik, and Ian Horrocks. A description logic primer. *CoRR*, abs/1201.4089, 2012.
6. Carsten Lutz, Carlos Areces, Ian Horrocks, and Ulrike Sattler. Keys, nominals, and concrete domains. *Journal of Artificial Intelligence Research*, 23(1):667–726, June 2005.
7. Michael Schneider. OWL 2 Web Ontology Language RDF-Based Semantics. W3C recommendation, W3C, October 2009.