# XXXXX

XXXXX[1] and XXXXX[2]

[1] XXXXX
XXXXX,
[2] XXXXX
XXXXX

**Abstract.** **Keywords:** ..

## 1 Introduction

## 2 Ideas

- sind alle constraints abgedeckt?
- kann man alle constraints in SPARQL definieren?
- sind alle constraints mit Logik ausdrückbar?
- vollständig mit Reasoning — OW
- vollständig ohne Reasoning — CW
- es gibt keinen query rewriting mechanismus für OWL 2, nur für OWL-QL
- constraints in einer anderen constraint language definieren wenn constraints nicht in OWL beschrieben werden können
- durch reasoning entstehen Probleme, auf die man nicht gekommen wäre –¿ sofort nachvollziehbar
- zeigen, dass OWL-QL und constraint language einer konkreten Domäne zusammen vollständig sind
- System entwickeln, das effizient ist / Experimente

—

Nehmen wir nun an, dass dein Framework welches entsprechende SPARQL Queries generiert diese auf einem SPARQL Endpoint evaluiert der zu der vorliegenden Ontologie bzw. des darin verwendeten OWL 2 Profils das entsprechende Entailment Regime realisiert, wären die zurückgegebenen Resultsets vollständig. Wie das Entailment Regime im Endpoint realisiert ist, also durch Query Rewriting oder durch Vervollständigung der ABox, ist dabei irrelevant.

Wie allerdings bspw. in https://www.uni-ulm.de/fileadmin/website_uni_ulm/iui.inst.090/Lehre/WS_2011-2012/SemWebGrundlagen/LectureNotes.pdf auf Seite 51 veranschaulicht, ist die Komplexität des Reasoning abhängig von der zugrunde gelegten Sprache und kann daher nur in bestimmten Fällen effizient durchgeführt werden. Wie in unserem letzten Paper beschrieben zielt unter anderem die Definition von DL-Lite gerade darauf ab Reasoning Aufgaben und Query Answering effizient zu ermöglichen und ist Grundlage des OWL 2 QL

Profils. Nun ist allgemein bekannt, dass die logische Konsistenz für diese Art von Sprachen effizient geprüft werden kann.

Allerding wäre wie bspw. in http://www.aifb.kit.edu/images/d/d2/2005_925_Haase_Consistent_Evol_1.pdf beschrieben auch eine sogenannte 'User-defined Consistency' denkbar. Genau an dieser Stelle könnten wir ansetzen.

## 3 research questions

- for which RDF validation requirement the expressivity of DL-LiteA respectively OWL 2 QL is sufficient?
- for which RDF validation requirement additional constraint languages are needed?
- which constraint languages are suitable to express remaining requirements?
- what are the effects of these constraints regarding complexity?

## 4 OWL 2 QL

OWL 2 profiles specification: [2]

- OWL 2 QL constructs
- Difference between OWL 2 DL and OWL 2 QL

**Logical Underpinning for OWL 2 QL.** OWL 2 QL is based on the DL-Lite family of description logics. Several variants of DL-Lite have been described in the literature, and DL-Lite$_R$ provides the logical underpinning for OWL 2 QL. DL-Lite$_R$ does not require the unique name assumption (UNA), since making this assumption would have no impact on the semantic consequences of a DL-Lite$_R$ ontology. More expressive variants of DL-Lite, such as DL-Lite$_A$, extend DL-Lite$_R$ with functional properties, and these can also be extended with keys; however, for query answering to remain in LOGSPACE, these extensions require UNA and need to impose certain global restrictions on the interaction between properties used in different types of axiom. Basing OWL 2 QL on DL-Lite$_R$ avoids practical problems involved in the explicit axiomatization of UNA [2].

## 5 RDF Validation Requirements Not Covered By OWL 2 QL

### 5.1 Class-Specific Disjointness of Properties

requirements:

- R-11-DISJOINT-DATA-PROPERTIES-CLASS-SPECIFIC
- R-12-DISJOINT-OBJECT-PROPERTIES-CLASS-SPECIFIC
- R-13-DISJOINT-GROUP-OF-PROPERTIES-CLASS-SPECIFIC

#### 5.1.1 Class-Specific Disjoint Data Properties R-11-DISJOINT-DATA-PROPERTIES-CLASS-SPECIFIC

- exclusive OR of data properties
- with OWL 2, inclusive OR of properties would be possible to express, but not exclusive OR of data properties

constraint (ShEx):

```
(foaf:name xsd:string  | foaf:givenName xsd:string+, foaf:familyName xsd:string)
```

valid data (ShEx):

```
<Anakin>
    foaf:name "Anakin Skywalker" ;
                foaf:familyName "Skywalker" .
```

```
<Anakin>
    foaf:givenName "Anakin" ;
                foaf:givenName "Darth" ;
    foaf:familyName "Skywalker" .
```

invalid data (ShEx):

```
<Anakin>
    foaf:name "Anakin Skywalker" ;
    foaf:givenName "Anakin" ;
    foaf:familyName "Skywalker" .
```

#### 5.1.2 Class-Specific Disjoint Object Properties R-12-DISJOINT-OBJECT-PROPERTIES-CLASS-SPECIFIC

- see R-11-DISJOINT-DATA-PROPERTIES-CLASS-SPECIFIC

#### 5.1.3 Class-Specific Disjoint Group of Properties R-13-DISJOINT-GROUP-OF-PROPERTIES-CLASS-SPECIFIC

- exclusive OR of property groups
- with OWL 2, inclusive OR of property groups would be possible to express, but not exclusive OR of property groups

A <Human>has either a name or at least 1 given name and 1 family name.
constraint (ShEx):

```
<Human> {
    (
            foaf:name xsd:string
        |
            foaf:givenName xsd:string+,
            foaf:familyName xsd:string
    )
}
```

valid data (ShEx):

```
<Anakin>
    foaf:givenName "Anakin" ;
    foaf:familyName "Skywalker" .
```

```
<Anakin>
    foaf:name "Anakin Skywalker" .
```

invalid data (ShEx):

```
<Anakin>
    foaf:givenName "Anakin" ;
    foaf:familyName "Skywalker" ;
    foaf:name "Anakin Skywalker" .
```

## 5.2 Default Values

requirements:

- R-31-DEFAULT-VALUES-OF-RDF-OBJECTS
- R-38-DEFAULT-VALUES-OF-RDF-LITERALS

### 5.2.1 R-31-DEFAULT-VALUES-OF-RDF-OBJECTS

- see R-38-DEFAULT-VALUES-OF-RDF-LITERALS

### 5.2.2 R-38-DEFAULT-VALUES-OF-RDF-LITERALS

rule (SPIN):

Jedis have only 1 blue laser sword per default. Siths, in contrast, normally have 2 red laser swords.

```
owl:Thing
    spin:rule [
        a sp:Construct ;
            sp:text """
                CONSTRUCT {
                    ?this :laserSwordColor "blue"^^xsd:string ;
                    ?this :numberLaserSwords "1"^^xsd:nonNegativeInteger .
                }
                WHERE {
                    ?this a :Jedi .
                } """ ; ] .
owl:Thing
    spin:rule [
        a sp:Construct ;
            sp:text """
                CONSTRUCT {
                    ?this :laserSwordColor "red"^^xsd:string ;
                    ?this :numberLaserSwords "2"^^xsd:nonNegativeInteger .
                }
                WHERE {
                    ?this a :Sith .
                } """ ; ] .
```

data (SPIN):

```
:Joda a :Jedi .
:DarthSidious a :Sith .
```

inferred triples (SPIN):

```
:Joda
    :laserSwordColor "blue"^^xsd:string ;
    :numberLaserSwords "1"^^xsd:nonNegativeInteger .
:DarthSidious
    :laserSwordColor "red"^^xsd:string ;
    :numberLaserSwords "2"^^xsd:nonNegativeInteger .
```

### 5.3 Membership in Controlled Vocabularies

requirements:

- R-32-MEMBERSHIP-OF-RDF-OBJECTS-IN-CONTROLLED-VOCABULARIES
- R-39-MEMBERSHIP-OF-RDF-LITERALS-IN-CONTROLLED-VOCABULARIES

constraint (DSP):

```
:bookDescriptionTemplate
    a dsp:DescriptionTemplate ;
    dsp:resourceClass swrc:Book ;
    dsp:statementTemplate [
        a dsp:NonLiteralStatementTemplate ;
        dsp:property dcterms:subject ;
        dsp:nonLiteralConstraint [
            a dsp:NonLiteralConstraint ;
            dsp:valueClass skos:Concept ;
            dsp:vocabularyEncodingScheme :BookSubjects, :BookTopics, :BookCategories ] ] .
```

A DSP consists of `dsp:DescriptionTemplate`s that put constraints on instances of a certain class (`dsp:resourceClass`). `:bookDescriptionTemplate` describes resources of the type `swrc:Book`. The constraints can either be constraints on the description itself, e.g. a minimum occurrence of instances of this class. Additionally, constraints on single properties can be defined within a `dsp:StatementTemplate`. The `dsp:NonLiteralStatementTemplate` restricts books to have `dcterms:subject` (`dsp:property`) relationships to RDF objects which are further described by the `dsp:NonLiteralConstraint`. These RDF objects have to be of the class `skos:Concept` (`dsp:ValueClass`). Controlled vocabularies (like `:BookSubjects`) are represented as `skos:ConceptSchemes` in RDF and as `dsp:VocabularyEncodingSchemes` in DSP. `dsp:VocabularyEncodingScheme` points to a list of controlled vocabularies the `skos:Concept` resources must be members of. The controlled vocabulary members (`skos:Concepts`) are related to the controlled vocabulary (`skos:ConceptScheme`) via the object properties `skos:inScheme` and `dcam:memberOf`.

valid data (DSP):

```
:ArtficialIntelligence
    a swrc:Book ;
    dcterms:subject :ComputerScience .
:ComputerScience
    a skos:Concept ;
    dcam:memberOf :BookSubjects ;
    skos:inScheme :BookSubjects .
:BookSubjects
    a skos:ConceptScheme .
```

invalid data (DSP):

```
:ArtficialIntelligence
    a swrc:Book ;
    dcterms:subject :ComputerScience .
:ComputerScience
    a skos:Concept ;
    dcam:memberOf :BooksAboutBirds ;
    skos:inScheme :BooksAboutBirds ;
    dcam:memberOf :BookSubjects ;
    skos:inScheme :BookSubjects .
:BookSubjects
    a skos:ConceptScheme .
```

The related subject (`:ComputerScience`) is a member of a controlled vocabulary (`:BooksAboutBirds`) which is not part of the list of allowed controlled vocabularies.

### 5.4 Pattern Matching

requirements:

- R-21-IRI-PATTERN-MATCHING-ON-RDF-SUBJECTS
- R-22-IRI-PATTERN-MATCHING-ON-RDF-OBJECTS
- R-23-IRI-PATTERN-MATCHING-ON-RDF-PROPERTIES
- R-44-PATTERN-MATCHING-ON-RDF-LITERALS
- R-141-NEGATIVE-PATTERN-MATCHING-ON-RDF-LITERALS

#### 5.4.1 R-44-PATTERN-MATCHING-ON-RDF-LITERALS

Covering approaches:

DQTP (MATCH Pattern), OWL 2 DL, RS, ShEx, SPARQL, SPIN

constraints (OWL 2 DL) [description logics abstract syntax]:

```

```

constraints (OWL 2 DL) [functional-style syntax]:

```
Declaration( Datatype( :SSN ) )
DatatypeDefinition(
    :SSN
    DatatypeRestriction( xsd:string xsd:pattern "[0-9]{3}-[0-9]{2}-[0-9]{4}" ) )
DataPropertyRange( :hasSSN :SSN )
```

constraints (OWL 2 DL) [turtle syntax]:

```
:SSN
    a rdfs:Datatype ;
    owl:equivalentClass [
        a rdfs:Datatype ;
        owl:onDatatype xsd:string ;
        owl:withRestrictions (
            [ xsd:pattern "[0-9]{3}-[0-9]{2}-[0-9]{4}" ] ) ] .
```

– OWL 2 construct 'DatatypeRestriction' not allowed for OWL 2 QL

A social security number is a string that matches the given regular expression. The second axiom defines :SSN as an abbreviation for a datatype restriction on xsd:string. The first axiom explicitly declares :SSN to be a datatype. The datatype :SSN can be used just like any other datatype; for example, it is used in the third axiom to define the range of the :hasSSN property.

valid data (OWL 2 DL):

```
:IimBernersLee
    :hasSSN "123-45-6789"^^:SSN .
```

invalid data (OWL 2 DL):

```
:IimBernersLee
    :hasSSN "123456789"^^:SSN .
```

### 5.4.2 R-141-NEGATIVE-PATTERN-MATCHING-ON-RDF-LITERALS

Covering approaches: DQTP (MATCH Pattern), OWL 2 DL, SPARQL, SPIN

constraints (DQTP):

MATCH Pattern [1]

Application logic or real world constraints may put restrictions on the form of a literal value. P1 is the property we need to check against REGEX and NOP can be a not operator (!) or empty.

```
SELECT DISTINCT ?s WHERE { ?s %%P1%% ?value .
    FILTER ( %%NOP%% regex(str(?value), %%REGEX%) ) }
```

example test bindings (DQTP):

```
1. dbo:isbn format is different '!' from '^([iIsSbBnN 0-9-])*$'
2. dbo:postCode format is different '!' from '^[0-9]{5}$'
3. foaf:phone contains any letters ('[A-Za-z]')
```

test binding (DQTP):

```
dbo:isbn format is different '!' from '^([iIsSbBnN 0-9-])*$'

P1 => dbo:isbn
NOP => !
REGEX => '^([iIsSbBnN 0-9-])*$'
```

valid data (DQTP):

```
:FoundationsOfSWTechnologies
    dbo:isbn 'ISBN-13 978-1420090505' .
```

invalid data (DQTP):

```
:HandbookOfSWTechnologies
    dbo:isbn 'DOI 10.1007/978-3-540-92913-0' .
```

### 5.5 Calculations on and Comparisons of RDF Literals

requirements:

- R-41-STATISTICAL-COMPUTATIONS
- R-42-COMPUTATIONS-BASED-ON-DATATYPE
- R-43-COMPARISONS-BASED-ON-DATATYPE
- R-194-PROVIDE-STRING-FUNCTIONS-FOR-RDF-LITERALS

### 5.6 Constraining Facets on RDF Literals

XSD constraining facets: http://www.w3.org/TR/2001/REC-xmlschema-2-20010502/#rf-facets

requirements:

- R-44-PATTERN-MATCHING-ON-RDF-LITERALS
- R-45-RANGES-OF-RDF-LITERAL-VALUES
- R-46-CONSTRAINING-FACETS
- R-50-WHITESPACE-HANDLING-OF-RDF-LITERALS
- R-142-NEGATIVE-RANGES-OF-RDF-LITERAL-VALUES

#### 5.6.1 R-44-PATTERN-MATCHING-ON-RDF-LITERALS

- see 'Pattern Matching'

#### 5.6.2 R-45-RANGES-OF-RDF-LITERAL-VALUES

constraints (OWL 2 DL) [description logics abstract syntax]:

```
ToDO
```

constraints (OWL 2 DL) [functional-style syntax]:

```
Declaration( Datatype( :NumberPlayersPerWorldCupTeam ) )
DatatypeDefinition(
    :NumberPlayersPerWorldCupTeam
    DatatypeRestriction(
        xsd:nonNegativeInteger
        xsd:minInclusive "1"^^xsd:nonNegativeInteger
        xsd:maxInclusive "23"^^xsd:nonNegativeInteger ) )
DataPropertyRange( :position :NumberPlayersPerWorldCupTeam )
```

constraints (OWL 2 DL) [turtle syntax]:

```
:NumberPlayersPerWorldCupTeam
    a rdfs:Datatype ;
    owl:equivalentClass [
        a rdfs:Datatype ;
        owl:onDatatype xsd:nonNegativeInteger ;
        owl:withRestrictions (
            [ xsd:minInclusive "1"^^xsd:nonNegativeInteger ]
            [ xsd:maxInclusive "23"^^xsd:nonNegativeInteger ] ) ] .
```

The data range 'NumberPlayersPerWorldCupTeam' contains the non negative integers 1 to 23, as each world cup team can only have 23 football players at most.

valid data (OWL 2 DL):

```
:MarioGoetze
    :position "19"^^:NumberPlayersPerWorldCupTeam .
```

invalida data (OWL 2 DL):

```
:MarioGoetze
    :position "99"^^:NumberPlayersPerWorldCupTeam .
```

### 5.7 Language of RDF Literals

requirements:

- R-47-LANGUAGE-TAG-MATCHING
- R-48-MISSING-LANGUAGE-TAGS
- R-49-RDF-LITERALS-HAVING-AT-MOST-ONE-LANGUAGE-TAG

#### 5.7.1 R-49-RDF-LITERALS-HAVING-AT-MOST-ONE-LANGUAGE-TAG

constraints (DQTP):

ONELANGPattern [1]

A literal value should contain at most 1 literal for a language. P1 is the property containing the literal and V1 is the language we want to check.

```
SELECT DISTINCT ?s WHERE { ?s %%P1%% ?c
    BIND ( lang(?c) AS ?l )
    FILTER (isLiteral (?c) && lang(?c) = %%V1%%)}
GROUP BY ?s HAVING COUNT (?l) > 1
```

test binding (DQTP):

a single English ("en") foaf:name

```
P1 => foaf:name
V1 => en
```

valid data (DQTP):

```
:LeiaSkywalker
    foaf:name 'Leia Skywalker'@en .
```

invalid data (DQTP):

```
:LeiaSkywalker
    foaf:name 'Leia Skywalker'@en ;
    foaf:name 'Leia'@en .
```

### 5.8 Property Occurrences

requirements:

- R-52-NEGATIVE-OBJECT-PROPERTY-CONSTRAINTS
- R-53-NEGATIVE-DATA-PROPERTY-CONSTRAINTS
- R-67-CLASSIFY-PROPERTIES-ACCORDING-TO-OCCURRENCE

### 5.9 Property Groups

requirements:

- R-13-DISJOINT-GROUP-OF-PROPERTIES-CLASS-SPECIFIC
- R-66-PROPERTY-GROUPS

### 5.10 RDF-Specific Validation

requirements:

- R-120-HANDLE-RDF-COLLECTIONS

### 5.11 RDF Shapes

requirements:

- R-125-RDF-SHAPE-CHECKING

### 5.12 RDF Validation Results

requirements:

- R-150-RDF-REPRESENTATION-OF-VALIDATION-RESULTS
- R-151-USEFUL-MESSAGE-VALIDATION-RESULTS
- R-152-FIND-NOT-VALIDATED-TRIPLES
- R-153-RDF-REPRESENTATION-OF-CONSTRAINT-VIOLATIONS
- R-154-HANDLE-CONSTRAINT-VIOLATIONS
- R-155-GUIDANCE-HOW-TO-BECOME-VALID-DATA
- R-156-REFERENCES-TO-TRIPLES-CAUSING-THE-CONSTRAINT-VIOLATIONS
- R-157-REFERENCES-TO-VALIDATION-RULES-CAUSING-CONSTRAINT-VIOLATIONS
- R-158-SEVERITY-LEVELS-OF-CONSTRAINT-VIOLATIONS
- R-159-EXPLAIN-REASONS-OF-CONSTRAINT-VIOLATIONS

### 5.13 Requirements Covered by OWL 2 DL But Not by OWL 2 QL

## 6 Related Work

## 7 Evaluation

### 7.1 Evaluation Using Practical Data Set

### 7.2 RDF Validation Requirements Covered by OWL 2 QL

Table 2

### 7.3 RDF Validation Requirements Not Covered by OWL 2 QL

Table 3 Table 4 Table 5

## 8 Conclusion and Future Work

## 9 Appendix

### 9.1 Allowed Usage of Constructs in Class Expressions in OWL 2 QL

**Subclass Expressions**

```
subClassExpression :=
    Class |
    subObjectSomeValuesFrom | DataSomeValuesFrom
subObjectSomeValuesFrom := 'ObjectSomeValuesFrom' '(' ObjectPropertyExpression owl:Thing ')'
```

**Superclass Expressions**

```
superClassExpression :=
    Class |
    superObjectIntersectionOf | superObjectComplementOf |
    superObjectSomeValuesFrom | DataSomeValuesFrom
```

### 9.2 Supported Constructs in OWL 2 QL

- subclass axioms (SubClassOf)
- class expression equivalence (EquivalentClasses)
- class expression disjointness (DisjointClasses)
- inverse object properties (InverseObjectProperties)
- property inclusion (SubObjectPropertyOf not involving property chains and SubDataPropertyOf)
- property equivalence (EquivalentObjectProperties and EquivalentDataProperties)
- property domain (ObjectPropertyDomain and DataPropertyDomain)
- property range (ObjectPropertyRange and DataPropertyRange)

**Table 1.** RDF Validation Requirements Covered by OWL 2 QL

| Requirements Classification | Requirements |
|---|---|
| | R-1-UNIQUENESS-OF-URIS |
| | R-2-UNIQUE-INSTANCES |
| | R-3-EQUIVALENT-CLASSES (EquivalentClasses) |
| | R-4-EQUIVALENT-OBJECT-PROPERTIES (EquivalentObjectProperties) |
| | R-5-EQUIVALENT-DATA-PROPERTIES (EquivalentDataProperties) |
| | R-7-DISJOINT-CLASSES (DisjointClasses) |
| | R-9-DISJOINT-OBJECT-PROPERTIES (DisjointObjectProperties) |
| | R-10-DISJOINT-DATA-PROPERTIES (DisjointDataProperties) |
| | R-14-DISJOINT-INDIVIDUALS (DifferentIndividuals) |
| | R-25-OBJECT-PROPERTY-DOMAIN (ObjectPropertyDomain) |
| | R-26-DATA-PROPERTY-DOMAIN (DataPropertyDomain) |
| | R-27-CLASS-SPECIFIC-VALIDATION |
| | R-28-OBJECT-PROPERTY-RANGE (ObjectPropertyRange) |
| | R-35-DATA-PROPERTY-RANGE (DataPropertyRange) |
| | R-54-SUB-OBJECT-PROPERTIES (SubObjectPropertyOf) |
| | R-56-INVERSE-OBJECT-PROPERTIES (ObjectInverseOf) |
| | R-59-REFLEXIVE-OBJECT-PROPERTIES (ReflexiveObjectProperty) |
| | R-60-IRREFLEXIVE-OBJECT-PROPERTIES (IrreflexiveObjectProperty) |
| | R-61-SYMMETRIC-OBJECT-PROPERTIES (SymmetricObjectProperty) |
| | R-62-ASYMMETRIC-OBJECT-PROPERTIES (AsymmetricObjectProperty) |
| | R-64-SUB-DATA-PROPERTIES (SubDataPropertyOf) |
| | R-93-DIFFERENCE-BETWEEN-CONSTRAINTS-ON-OBJECT-AND-DATA-PROPERT |
| | R-94-POSITIVE-OBJECT-PROPERTY-ASSERTIONS (ObjectPropertyAssertion) |
| | R-95-POSITIVE-DATA-PROPERTY-ASSERTIONS (DataPropertyAssertion) |
| | R-99-STABLE-IDENTIFICATION-OF-CONSTRAINTS |
| | R-100-SUBSUMPTION (SubClassOf) |
| | R-101-DECLARATIVE-CONSTRAINT-LANGUAGE |
| | R-102-INTUITIVE-CONSTRAINT-LANGUAGE |
| | R-103-HIGH-LEVEL-CONSTRAINT-LANGUAGE |
| | R-104-CONSTRAINT-LANGUAGE-HAVING-IMPLEMENTATION-LANGUAGE |
| | R-105-CONSTRAINT-LANGUAGE-TRANSLATABLE-TO-IMPLEMENTATION-LANG |
| | R-107-TRANSFORMATIONS-BETWEEN-CONSTRAINT-LANGUAGE-AND-UML |
| | R-108-TRANSFORMATIONS-BETWEEN-CONSTRAINT-LANGUAGE-AND-XML-SC |
| | R-109-TRANSFORMATIONS-BETWEEN-CONSTRAINT-LANGUAGE-AND-OCL |
| | R-110-TRANSFORMATIONS-BETWEEN-CONSTRAINT-LANGUAGE-AND-SPARQL |

**Table 2.** RDF Validation Requirements Covered by OWL 2 QL

| Requirements Classification | Requirements |
|---|---|
| | R-111-BASIC-USE-CASES-COVERED-BY-CONSTRAINT-LANGUAGE |
| | R-113-INTERACTION-OF-VALIDATION-WITH-REASONING |
| | R-115-CLOSED-WORLD-ASSUMPTION-CWA |
| | R-116-UNIQUE-NAME-ASSUMPTION-UNA |
| | R-117-CONTEXT-SENSITIVE-CONSTRAINTS |
| | R-118-NAMESPACE-SENSITIVE-CONSTRAINTS |
| | R-122-TRADE-OFF-BETWEEN-DIMENSIONS-EXPRESSIVITY-COMPLEXITY-PREI |
| | R-124-DESCRIBE-DATA |
| | R-126-CUSTOMIZABLE-VALIDATION-PROCESS |
| | R-128-HUMAN-UNDERSTANDABLE-CONCRETE-SYNTAXES-FORMULATING-CON |
| | R-129-MACHINE-UNDERSTANDABLE-CONCRETE-SYNTAXES-FORMULATION-CO |
| | R-130-CONCISE-CONCRETE-SYNTAXES-FORMULATING-CONSTRAINTS |
| | R-131-OWL-AS-CONCRETE-SYNTAX-FORMULATING-CONSTRAINTS |
| | R-132-MULTIPLE-CONCRETE-SYNTAXES-FORMULATING-CONSTRAINTS |
| | R-133-MULTIPLE-CONCRETE-SYNTAXES-FORMULATING-DATA |
| | R-136-MODULARITY-OF-CONSTRAINT-DEFINITIONS |
| | R-137-LEVERAGE-ON-EXISTING-TECHNOLOGIES |
| | R-138-CONSTRAINT-LANGUAGE-COMPATIBLE-WITH-SPARQL |
| | R-139-CONSTRAINT-LANGUAGE-DRIVES-USER-INTERFACE-FORM-GENERATIO |
| | R-140-SEPARATE-ONTOLOGIES-FROM-VALIDATION-SCHEMAS |
| | R-143-CONDITIONAL-TYPED-VALIDATION |
| | R-147-DISTRIBUTION-OF-CONSTRAINT-SCHEMAS |
| | R-148-DISTRIBUTED-VALIDATION-IN-COLLABORATIVE-ENVIRONMENTS |
| | R-149-MANAGEMENT-OF-CONSTRAINT-SCHEMA-EVOLUTION |
| | R-160-OPEN-SOURCE-CONSTRAINT-VALIDATION |
| | R-161-ACCEPTABLE-PERFORMANCE-OF-VALIDATION-ALGORITHM |
| | R-162-SPECIFICATION-PUBLICLY-AVAILABLE |
| | R-163-IMPLEMENTATION-EXISTS |
| | R-164-IMPLEMENTATION-PUBLICLY-AVAILABLE |
| | R-165-EXECUTABLE-DEMOS-EXAMPLES-USE-CASES |
| | R-168-PERFORM-BIG-DATASETS |
| | R-169-RDF-REPRESENTATION-OF-CONSTRAINT-LANGUAGE |
| | R-173-SEPARATE-CONSTRAINTS-FROM-VOCABULARIES-AND-ONTOLOGIES |
| | R-174-REUSE-CONSTRAINTS |
| | R-175-DISCOVER-CONSTRAINTS |
| | R-177-DEFINE-SEMANTICS-FOR-CONSTRAINTS |
| | R-178-ASSOCIATE-CONSTRAINTS-WITH-VOCABULARIES |
| | R-182-USE-KNOWN-CONCRETE-SYNTAX |
| | R-184-COMPACT-CONCRETE-SYNTAX |
| | R-187-DEFINE-SEMANTICS-OF-CONSTRAINTS-IN-TERMS-OF-SPARQL |
| | R-190-SPECIFY-EXPECTED-BEHAVIOR-UNDER-ALL-POSSIBLE-ENTAILMENT-RI |
| | R-192-DEFINE-ANNOTATIONS-FOR-CONSTRAINTS |
| | R-195-CONSTRAINT-LANGUAGE-EASY-TO-CONSUME-BY-TOOLS |
| | R-197-ATTACH-CONSTRAINTS-TO-CLASSES |
| | R-198-RDF-VALIDATION-AFTER-INFERENCING |
| | R-199-RDF-VALIDATION-MUST-COMPILE-DOWN-TO-SPARQL |

**Table 3.** RDF Validation Requirements Not Covered by OWL 2 QL

| Requirements | Covering Constraint La |
|---|---|
| R-6-EQUIVALENT-INDIVIDUALS (SameIndividual) | OWL 2 DL |
| R-8-DISJOINT-UNION-OF-CLASS-EXPRESSIONS (DisjointUnion) | OWL 2 DL |
| R-11-DISJOINT-DATA-PROPERTIES-CLASS-SPECIFIC | ShEx |
| R-12-DISJOINT-OBJECT-PROPERTIES-CLASS-SPECIFIC | ShEx |
| R-13-DISJOINT-GROUP-OF-PROPERTIES-CLASS-SPECIFIC | ShEx |
| R-15-CONJUNCTION-OF-CLASS-EXPRESSIONS (ObjectIntersectionOf) | OWL 2 DL |
| R-16-CONJUNCTION-OF-DATA-RANGES (DataIntersectionOf) | OWL 2 DL |
| R-17-DISJUNCTION-OF-CLASS-EXPRESSIONS (DisjointUnionOf) | OWL 2 DL |
| R-18-DISJUNCTION-OF-DATA-RANGES (DataUnionOf) | OWL 2 DL |
| R-19-NEGATION-OF-CLASS-EXPRESSIONS (ObjectComplementOf) | OWL 2 DL |
| R-20-NEGATION-OF-DATA-RANGES (DataComplementOf) | OWL 2 DL |
| R-21-IRI-PATTERN-MATCHING-ON-RDF-SUBJECTS | SPIN |
| R-22-IRI-PATTERN-MATCHING-ON-RDF-OBJECTS | SPIN, ShEx |
| R-23-IRI-PATTERN-MATCHING-ON-RDF-PROPERTIES | SPIN, ShEx |
| R-24-PROVENANCE-CONSTRAINTS | |
| R-29-CLASS-SPECIFIC-RANGE-OF-RDF-OBJECTS | SPIN, RS |
| R-30-ALLOWED-VALUES-FOR-RDF-OBJECTS (ObjectOneOf) | OWL 2 DL |
| R-31-DEFAULT-VALUES-OF-RDF-OBJECTS | SPIN, BF, RS |
| R-32-MEMBERSHIP-OF-RDF-OBJECTS-IN-CONTROLLED-VOCABULARIES | DSP, SPIN |
| R-33-NEGATIVE-OBJECT-CONSTRAINTS | |
| R-34-AVAILABLE-CLASS-DEFINITION | ShEx, SPIN |
| R-36-CLASS-SPECIFIC-RANGE-OF-RDF-LITERALS | ShEx, BF, SPIN |
| R-37-ALLOWED-VALUES-FOR-RDF-LITERALS (DataOneOf) | OWL 2 DL |
| R-38-DEFAULT-VALUES-OF-RDF-LITERALS | SPIN, BF, RS |
| R-39-MEMBERSHIP-OF-RDF-LITERALS-IN-CONTROLLED-VOCABULARIES | DSP, SPIN |
| R-44-PATTERN-MATCHING-ON-RDF-LITERALS | DQTP, OWL 2 DL, Sh |
| R-41-STATISTICAL-COMPUTATIONS | SPIN |
| R-42-COMPUTATIONS-BASED-ON-DATATYPE | |
| R-43-COMPARISONS-BASED-ON-DATATYPE | DQTP, ShEx, SPIN |
| R-45-RANGES-OF-RDF-LITERAL-VALUES | DQTP, SPIN |
| R-46-CONSTRAINING-FACETS | Stardog, SPIN |
| R-47-LANGUAGE-TAG-MATCHING | SPIN |
| R-48-MISSING-LANGUAGE-TAGS | SPIN |
| R-49-RDF-LITERALS-HAVING-AT-MOST-ONE-LANGUAGE-TAG | DQTP, SPIN |
| R-50-WHITESPACE-HANDLING-OF-RDF-LITERALS | SPIN |
| R-51-HTML-HANDLING-OF-RDF-LITERALS | SPIN |
| R-52-NEGATIVE-OBJECT-PROPERTY-CONSTRAINTS | ShEx, SPIN |
| R-53-NEGATIVE-DATA-PROPERTY-CONSTRAINTS | ShEx, SPIN |
| R-55-OBJECT-PROPERTY-PATHS (SubObjectPropertyOf) | OWL 2 DL |
| R-57-FUNCTIONAL-OBJECT-PROPERTIES (FunctionalObjectProperty) | OWL 2 DL |
| R-58-INVERSE-FUNCTIONAL-OBJECT-PROPERTIES (InverseFunctionalObjectProperty) | OWL 2 DL |
| R-63-TRANSITIVE-OBJECT-PROPERTIES (TransitiveObjectProperty) | OWL 2 DL |
| R-65-FUNCTIONAL-DATA-PROPERTIES (FunctionalDataProperty) | OWL 2 DL |
| R-66-PROPERTY-GROUPS | ShEx, SPIN |
| R-67-CLASSIFY-PROPERTIES-ACCORDING-TO-OCCURRENCE | |
| R-68-REQUIRED-PROPERTIES (ObjectMinCardinality, DataMinCardinality) | OWL 2 DL |
| R-69-OPTIONAL-PROPERTIES (ObjectMinCardinality, DataMinCardinality) | OWL 2 DL |
| R-70-REPEATABLE-PROPERTIES (ObjectMinCardinality, DataMinCardinality) | OWL 2 DL |
| R-71-CONDITIONAL-PROPERTIES | |
| R-72-RECOMMENDED-PROPERTIES | |

**Table 4.** RDF Validation Requirements Not Covered by OWL 2 QL

| Requirements |
|---|
| R-74-EXACT-QUALIFIED-CARDINALITY-RESTRICTIONS-ON-OBJECT-PROPERTIES (ObjectExactCardinality |
| R-75-MINIMUM-QUALIFIED-CARDINALITY-RESTRICTIONS-ON-OBJECT-PROPERTIES (ObjectMinCardinali |
| R-76-MAXIMUM-QUALIFIED-CARDINALITY-RESTRICTIONS-ON-OBJECT-PROPERTIES (ObjectMaxCardina |
| R-77-EXACT-QUALIFIED-CARDINALITY-RESTRICTIONS-ON-DATA-PROPERTIES (DataExactCardinality) |
| R-78-MINIMUM-QUALIFIED-CARDINALITY-RESTRICTIONS-ON-DATA-PROPERTIES (DataMinCardinality) |
| R-79-MAXIMUM-QUALIFIED-CARDINALITY-RESTRICTIONS-ON-DATA-PROPERTIES (DataMaxCardinality) |
| R-80-EXACT-UNQUALIFIED-CARDINALITY-RESTRICTIONS-ON-OBJECT-PROPERTIES (ObjectExactCardina |
| R-81-MINIMUM-UNQUALIFIED-CARDINALITY-RESTRICTIONS-ON-OBJECT-PROPERTIES (ObjectMinCardi |
| R-82-MAXIMUM-UNQUALIFIED-CARDINALITY-RESTRICTIONS-ON-OBJECT-PROPERTIES (ObjectMaxCarc |
| R-83-EXACT-UNQUALIFIED-CARDINALITY-RESTRICTIONS-ON-DATA-PROPERTIES (DataExactCardinality) |
| R-84-MINIMUM-UNQUALIFIED-CARDINALITY-RESTRICTIONS-ON-DATA-PROPERTIES (DataMinCardinality |
| R-85-MAXIMUM-UNQUALIFIED-CARDINALITY-RESTRICTIONS-ON-DATA-PROPERTIES (DataMaxCardinali |

- disjoint properties (DisjointObjectProperties and DisjointDataProperties)
- symmetric properties (SymmetricObjectProperty)
- reflexive properties (ReflexiveObjectProperty)
- irreflexive properties (IrreflexiveObjectProperty)
- asymmetric properties (AsymmetricObjectProperty)
- assertions: DifferentIndividuals, ClassAssertion, ObjectPropertyAssertion, and DataPropertyAssertion

### 9.3 Not Supported Constructs in OWL 2 QL

- existential quantification to a class expression or a data range (ObjectSomeValuesFrom and DataSomeValuesFrom) in the subclass position
- self-restriction (ObjectHasSelf)
- existential quantification to an individual or a literal (ObjectHasValue, DataHasValue)
- enumeration of individuals and literals (ObjectOneOf, DataOneOf)
- universal quantification to a class expression or a data range (ObjectAllValuesFrom, DataAllValuesFrom)
- cardinality restrictions (ObjectMaxCardinality, ObjectMinCardinality, ObjectExactCardinality, DataMaxCardinality, DataMinCardinality, DataExactCardinality)
- disjunction (ObjectUnionOf, DisjointUnion, and DataUnionOf)
- property inclusions (SubObjectPropertyOf) involving property chains
- functional and inverse-functional properties (FunctionalObjectProperty, InverseFunctionalObjectProperty, and FunctionalDataProperty)
- transitive properties (TransitiveObjectProperty)
- keys (HasKey)
- individual equality assertions and negative property assertions (SameIndividual, NegativeObjectPropertyAssertion, NegativeDataPropertyAssertion)

**Table 5.** RDF Validation Requirements Not Covered by OWL 2 QL

| Requirements | Coveri |
|---|---|
| R-86-EXISTENTIAL-QUANTIFICATION-ON-OBJECT-PROPERTIES (ObjectSomeValuesFrom) | OWL 2 |
| R-87-UNIVERSAL-QUANTIFICATION-ON-OBJECT-PROPERTIES (ObjectAllValuesFrom) | OWL 2 |
| R-88-INDIVIDUAL-VALUE-RESTRICTION-ON-OBJECT-PROPERTIES (ObjectHasValue) | OWL 2 |
| R-89-SELF-RESTRICTION (ObjectHasSelf) | OWL 2 |
| R-90-EXISTENTIAL-QUANTIFICATION-ON-DATA-PROPERTIES (DataSomeValuesFrom) | OWL 2 |
| R-91-UNIVERSAL-QUANTIFICATION-ON-DATA-PROPERTIES (DataAllValuesFrom) | OWL 2 |
| R-92-LITERAL-VALUE-RESTRICTION (DataHasValue) | OWL 2 |
| R-96-NEGATIVE-OBJECT-PROPERTY-ASSERTIONS (NegativeObjectPropertyAssertion) | OWL 2 |
| R-97-NEGATIVE-DATA-PROPERTY-ASSERTIONS (NegativeDataPropertyAssertion) | OWL 2 |
| R-98-CHECK-VALIDITY-OF-URIS | |
| R-106-EXTENSIBLE-CONSTRAINT-LANGUAGE | |
| R-112-EXTENSIBLE-CONSTRAINTS | |
| R-114-PROVIDE-RDF-REST-SERVICES-FOR-RDF-VALIDATION | |
| R-119-VALIDATION-ON-NAMED-GRAPHS | |
| R-120-HANDLE-RDF-COLLECTIONS | ShEx, |
| R-121-SPECIFY-ORDER-OF-RDF-RESOURCES | |
| R-123-STATE | |
| R-125-RDF-SHAPE-CHECKING | ShEx |
| R-126-CUSTOMIZABLE-VALIDATION-PROCESS | |
| R-134-SPECIFY-USAGE-OF-TERMS | |
| R-135-CONSTRAINT-LEVELS | |
| R-141-NEGATIVE-PATTERN-MATCHING-ON-RDF-LITERALS | DQTP |
| R-142-NEGATIVE-RANGES-OF-RDF-LITERAL-VALUES | DQTP |
| R-146-CONSTRAINT-VALIDATION-OF-RDF-INPUT-WITH-RESPECT-TO-EXISTING-RDF | |
| R-150-RDF-REPRESENTATION-OF-VALIDATION-RESULTS | SPIN, |
| R-151-USEFUL-MESSAGE-VALIDATION-RESULTS | SPIN, |
| R-152-FIND-NOT-VALIDATED-TRIPLES | ShEx |
| R-153-RDF-REPRESENTATION-OF-CONSTRAINT-VIOLATIONS | SPIN |
| R-154-HANDLE-CONSTRAINT-VIOLATIONS | ShEx, |
| R-155-GUIDANCE-HOW-TO-BECOME-VALID-DATA | ShEx, |
| R-156-REFERENCES-TO-TRIPLES-CAUSING-THE-CONSTRAINT-VIOLATIONS | SPIN |
| R-157-REFERENCES-TO-VALIDATION-RULES-CAUSING-CONSTRAINT-VIOLATIONS | ShEx, |
| R-158-SEVERITY-LEVELS-OF-CONSTRAINT-VIOLATIONS | SPIN |
| R-159-EXPLAIN-REASONS-OF-CONSTRAINT-VIOLATIONS | SPIN, |
| R-166-RDF-STREAMING-VALIDATION | |
| R-167-VALIDATE-RDF-IN-AN-HTML-DOCUMENT-CONSTAINING-RDFA-MARKUP | |
| R-170-VALIDATION-OF-SPARQL-ENDPOINTS | |
| R-171-VALIDATION-OF-URIS-BY-DEREFERENCING | |
| R-172-GENERATE-HUMAN-READABLE-DOCUMENTATION | |
| R-176-PROVIDE-HIGH-LEVEL-VOCABULARY-FOR-THE-MOST-COMMON-TYPES-OF-CONSTRAINTS | |
| R-179-ASSOCIATE-CONSTRAINTS-WITH-RDF-DOCUMENTS | |
| R-180-ASSOCIATE-CONSTRAINTS-WITH-RDF-DATASETS | |
| R-181-ASSOCIATE-CONSTRAINTS-WITH-RDF-REST-APIS | |
| R-183-CONSTRAINTS-ABOUT-CONSTRAINTS | |
| R-185-FEDERALIZED-RDF-VALIDATION | |
| R-186-EXTEND-EXPRESSIVITY-WITH-SPARQL | |
| R-188-EXPRESSIVITY-OF-CONSTRAINT-LANGUAGE-EQUIVALENT-TO-SPARQL | |
| R-189-ADD-ANNOTATIONS-TO-CONSTRAINT-VIOLATION-OBJECTS | |
| R-191-SHAPES-RELATED-TO-TYPES | |
| R-193-MULTIPLE-CONSTRAINT-VALIDATION-EXECUTION-LEVELS | |
| R-194-PROVIDE-STRING-FUNCTIONS-FOR-RDF-LITERALS | SPIN |

## References


1. 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.
2. Boris Motik, Bernardo Cuenca Grau, Ian Horrocks, Zhe Wu, Achille Fokoue, and Carsten Lutz. Owl 2 web ontology language: Profiles. Technical report, December 2008.