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: [1]

- 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 [1].

5 RDF Validation Requirements Not Covered By OWL 2 QL

5.1 class-specific disjoint data properties

constraint (ShEx):

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

5.2 class-specific disjoint object properties

5.3 Class-Specific Disjoint Group of Properties

requirements DB:

- ID: R-13-DISJOINT-GROUP-OF-PROPERTIES-CLASS-SPECIFIC
- URL: http://lelystad.informatik.uni-mannheim.de/rdf-validation/?q=node/20

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" .
```

6 Related Work

7 Evaluation

evaluation using practical data set

8 Conclusion and Future Work

9 Appendix

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

– class
– ObjectSomeValuesFrom (where the class is limited to owl:Thing)
– DataSomeValuesFrom

Superclass Expressions

– class
– ObjectIntersectionOf
– ObjectComplementOf
– ObjectSomeValuesFrom
– 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)
– disjoint properties (DisjointObjectProperties and DisjointDataProperties)
– symmetric properties (SymmetricObjectProperty)
– reflexive properties (ReflexiveObjectProperty)
– irreflexive properties (IrreflexiveObjectProperty)
– asymmetric properties (AsymmetricObjectProperty)
– assertions other than individual equality assertions and negative property 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

9.4 RDF Validation Requirements Covered by OWL 2 QL

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
	R-4-EQUIVALENT-OBJECT-PROPERTIES
	R-5-EQUIVALENT-DATA-PROPERTIES
	R-7-DISJOINT-CLASSES
	R-9-DISJOINT-OBJECT-PROPERTIES
	R-10-DISJOINT-DATA-PROPERTIES
	R-14-DISJOINT-INDIVIDUALS

9.5 RDF Validation Requirements Not Covered by OWL 2 QL

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

Requirements	Covering Constraint Languages
R-6-EQUIVALENT-INDIVIDUALS	OWL 2 DL
R-8-DISJOINT-UNION-OF-CLASS-EXPRESSIONS	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

References

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