RDF Constraint Checking

Using RDF Data Descriptions (RDD)

Peter M. Fischer, Georg Lausen, Alexander Schätzle
University of Freiburg, Germany

Michael Schmidt
metaphacts GmbH, Germany

LWDM 2015 - Brussels, Belgium
Motivation

- **Linked Open Data (LOD) is becoming more and more popular (not only academia & research)**
  - RDF is the *lingua franca* of Semantic Web and LOD

- **However, quality of RDF datasets varies heavily**
  - What we would need:
    1. Mechanisms to describe the *schema* of a dataset
      - RDFS and OWL are made for inference but not for validation
    2. Tools to validate if a dataset adheres to a given schema

- **W3C RDF Data Shapes Working Group Charter**
  - Definition and publication of topology and value constraints
  - Verification of data integrity

---

1 [http://www.w3.org/2014/data-shapes/charter]
RDF Data Descriptions (RDD) [1]
- Human readable, easy to grasp, expressive constraint language for RDF
- Qualified constraints (assigned to a class)
- Unqualified constraints (global constraints on properties)
Example

```owl
OWA CLASS foaf:Person {
    KEY rdfs:label : LITERAL
    MAX(2) foaf:mbox : LITERAL
    TOTAL foaf:age : LITERAL(xsd:int)
    RANGE (foaf:Person) foaf:knows : IRI
}
```
Example

OWL CLASS foaf:Person {
  KEY rdfs:label : LITERAL
  MAX(2) foaf:mbox : LITERAL
  TOTAL foaf:age : LITERAL(xsd:int)
  RANGE (foaf:Person) foaf:knows : IRI
}

age is missing
RDD vs. Shape Expressions [2]

RDD

```owl
OWA CLASS foaf:Person {
  KEY rdfs:label : LITERAL
  MAX(2) foaf:mbox : LITERAL
  TOTAL foaf:age : LITERAL(xsd:int)
  RANGE(foaf:Person) foaf:knows : IRI
}
```

- More focus on verification
- Inspired by relational constraints
- Validation of typed datasets
- **Meaning:** Are there instances of type `person` that do not adhere to the schema?

Shape Expressions (ShEx)

```xml
<Person> {
  rdfs:label xsd:string
  , foaf:mbox xsd:string{0,2}
  , foaf:age xsd:int
  , foaf:knows @<Person>*
}
```

- More focus on type inference
- Inspired by XML RelaxNG
- **Meaning:** Which instances have the shape of a `person`
Semantics of RDD defined in first-order logic (FOL)

- Can be mapped to existing work on integrity constraints, e.g. relational constraints
- FOL is well understood (in terms of complexity and expressivity)

Example:

```owl
    OWL CLASS foaf:Person {
        KEY rdfs:label : LITERAL
        MAX(2) foaf:mbox : LITERAL
        TOTAL foaf:age : LITERAL(xsd:int)
        RANGE(foaf:Person) foaf:knows : IRI
    }
```

\[
total(p) : min(p, 1) \land max(p, 1)
\]

\[
min(foaf:age, foaf:Person, 1) :
G($s, rdf:type, foaf:Person) \rightarrow \exists o_1(G($s, foaf:age, o_1))
\]

\[
max(foaf:age, foaf:Person, 1) :
G($s, rdf:type, foaf:Person) \land G($s, foaf:age, o_1) \land G($s, foaf:age, o_2) \rightarrow o_1 = o_2
\]
RDD Checker: SPARQL-based implementation of RDD

- RDD is decomposed into corresponding FOL constraints
- FOL constraints are mapped to corresponding SPARQL 1.0 queries (some constraints impose more than one query)

Example:

```sparql
OWA CLASS foaf:Person {
    KEY rdfs:label : LITERAL
    MAX(2) foaf:mbox : LITERAL
    TOTAL foaf:age : LITERAL(xsd:int)
    RANGE (foaf:Person) foaf:knows : IRI
}
```

```
MIN(1):
SELECT ?s {
    ?s rdf:type foaf:Person
    OPTIONAL { ?s foaf:age ?o1 }
    FILTER (!BOUND(?o1))
} LIMIT 3
```

```
MAX(1):
SELECT ?s {
    ?s rdf:type foaf:Person .
    ?s foaf:age ?o1 . ?s foaf:age ?o2
    FILTER (!(?o1=?o2))
} LIMIT 3
```
**RDD Checker - Evaluation**

- **Setup:**
  - RDD Checker v2.11 using Sesame 2.7.12 and Virtuoso 7.1.0
  - 4 cores @ 3.06 GHz, 32 GB RAM, 12 TB disk, Ubuntu 12.04 LTS
  - **Data:** SP2 Benchmark Data → scalings 10K to 100M triples
  - **Constraints:** 215 in total → 251 SPARQL queries

- **Validation vs. Loading:**

![Graphs comparing loading and validation times for Sesame and Virtuoso](image-url)
1:1 mapping of FOL to SPARQL 1.0 is not optimal
- A lot of leeway for optimizations in various directions
- We cannot pass it on to the SPARQL optimizer as many optimizations rely on deep understanding of the constraint semantics

3 directions of optimization:
- Intra-Query Optimization
  - optimize individual queries (using SPARQL 1.1)
- Intra-Constraint Optimization
  - reduce queries for one constraint
- Inter-Constraint Optimization
  - combine constraints in one query
Optimizations

- Intra-Query Optimization

```sparql
OWA CLASS foaf:Person {
    KEY rdfs:label : LITERAL
    MAX(2) foaf:mbox : LITERAL
    TOTAL foaf:age : LITERAL(xsd:int)
    RANGE (foaf:Person) foaf:knows : IRI
}
```

**SPARQL 1.0:**
```
SELECT ?s {
    ?s rdf:type foaf:Person .
    FILTER (!(?o1=?o2 || ?o1=?o3 || ?o2=?o3))
} LIMIT 3
```

**RDD Checker v2.11**

- `max(p,C,n)`:  
  - $\rightarrow n+1$ joins  
  - `((n+1)/2)` filter conditions

- `max(p,C,n)`:  
  - $\rightarrow 1$ join  
  - independent of max value
Optimizations

- Intra-Query Optimization – Experiments
  - Varying \( max(n) \) between 1 and 9 (invalid for \( n < 4 \))
  - Qualified (for a specific class) vs. Unqualified (for all classes)
  - Limit (only a few violations) vs. No Limit (all violations)

- Grouping always faster without Limit
- Join with Limit fast for small \( n \)

- Grouping faster for \( n > 3 \)
- Join faster for small \( n \) (even without Limit)
Intra-Constrain Optimization

```owl
CLASS foaf:Person {
    KEY rdfs:label : LITERAL
    MAX(2) foaf:mbox : LITERAL
    TOTAL foaf:age : LITERAL(xsd:int)
    RANGE (foaf:Person) foaf:knows : IRI
}
```

**MIN(1):**
```
SELECT ?s {
    ?s rdf:type foaf:Person
    OPTIONAL { ?s foaf:age ?o1 }
    FILTER (!BOUND(?o1))
} LIMIT 3
```

**MAX(1):**
```
SELECT ?s {
    ?s rdf:type foaf:Person
    ?s foaf:age ?o1 . ?s foaf:age ?o2
    FILTER (!(?o1=?o2))
} LIMIT 3
```

Combine **MIN(1)** and **MAX(1)** using SPARQL 1.1:
```
SELECT ?s {
    ?s rdf:type foaf:Person
    OPTIONAL { ?s foaf:age ?o1 }
} GROUP BY ?s HAVING (COUNT(?o1) != 1)
LIMIT 3
```
### Intra-Constraint Optimization – Experiments

- $\min(p,1) + \max(p,1)$ using `joins` (SPARQL 1.0) as baseline
- $\min(p,1) + \max(p,1)$ using `grouping` (SPARQL 1.1)
- $\text{total}(p)$ in a single query using grouping (SPARQL 1.1)

- **Sesame**
  - Join for $n=1$ cheaper than grouping
  - But combination is superior to join

- **Virtuoso**
  - Join always cheaper for small datasets
  - Grouping combination superior for $\geq 5M$
Summary

- **RDF Data Descriptions (RDD)**
  - Expressive RDF constraint language
  - Inspired by relational constraints
  - Easy to grasp, human readable syntax
  - Semantics defined in FOL

- **RDD Checker (v2.11)**
  - Validation tool for RDD
  - Maps constraints into SPARQL 1.0 compliant queries
  - Queries can be run with any existing RDF triple store

- **Future Work**
  - Current implementation has many optimization potentials
  - Reduce number of SPARQL queries using Multi-Query Optimization
  - Adding more constraint definitions to RDD
Thank you for your attention!

Questions?
References
