RDF Data Descriptions

Georg Lausen
Albert-Ludwigs-University of Freiburg, Germany

Michael Schmidt
Kaiserstraße 86, 66133 Scheidt, Germany
RDF W3C Recommendation, 10 February 2004:

The intention of RDF is to provide an open, minimally constraining way for representing information.

Does minimally constraining mean arbitrary?

What are the consequences for LOD?

A description of the RDF graph’s structure would be helpful!

Could be published as part of VoID.  
…(Vocabulary of Interlinked Datasets, http://www.w3.org/TR/void/)
RDF W3C Recommendation, 10 February 2004:

The intention of RDF is to provide an open, minimally constraining way for representing information.

Does minimally constraining mean arbitrary?

What are the consequences for LOD?

A description of the RDF graph’s structure would be helpful!

Could be published as part of VoID.

…(Vocabulary of Interlinked Datasets, http://www.w3.org/TR/void/)
RDF W3C Recommendation, 10 February 2004:

The intention of RDF is to provide an open, minimally constraining way for representing information.

Does minimally constraining mean arbitrary?

What are the consequences for LOD?

A description of the RDF graph’s structure would be helpful!

Could be published as part of VoID.

…(Vocabulary of Interlinked Datasets, http://www.w3.org/TR/void/)
RDF W3C Recommendation, 10 February 2004:

The intention of RDF is to provide an open, minimally constraining way for representing information.

Does minimally constraining mean arbitrary?

What are the consequences for LOD?

A description of the RDF graph’s structure would be helpful!

Could be published as part of VoID.  
...(Vocabulary of Interlinked Datasets, http://www.w3.org/TR/void/)
The intention of RDF is to provide an open, minimally constraining way for representing information.

Does *minimally constraining* mean *arbitrary*?

What are the consequences for LOD?

A description of the RDF graph’s structure would be helpful!

Could be published as part of VoID.

…(Vocabulary of Interlinked Datasets, http://www.w3.org/TR/void/)
1. RDF Validation

W3C Workshop on RDF Validation, 10-11 September 2013:

- Save data consumers from malformed input.
- Help debug generators/exporters of RDF data.
- Orient users as to what form of data would be considered "valid".
- ...

... semantic query optimization
1. RDF Validation

W3C Workshop on RDF Validation, 10-11 September 2013:

- Save data consumers from malformed input.
- Help debug generators/exporters of RDF data.
- Orient users as to what form of data would be considered "valid".
- ...

... semantic query optimization
Discussion: *Orient users as to what form of data would be considered ”valid”.*

**Task:** write a SPARQL query that extracts a one-row per person summary!

Assume the data resides at an Sparql-endpoint so we have to write the query without knowing the respective RDF graph. However we might know the predicates used to describe persons.
Discussion: *Orient users as to what form of data would be considered ”valid”.*

**Task: write a SPARQL query that extracts a one-row per person summary!**

Assume the data resides at an Sparql-endpoint so we have to write the query without knowing the respective RDF graph. However we might know the predicates used to describe persons.
Discussion: Orient users as to what form of data would be considered "valid".

Task: write a SPARQL query that extracts a one-row per person summary!

Assume the data resides at an Sparql-endpoint so we have to write the query without knowing the respective RDF graph. However we might know the predicates used to describe persons.

We have to make some assumptions, e.g.:
Task: write a SPARQL query that extracts a one-row per person summary!

```
SELECT ?person ?name ?age (GROUP_CONCAT(?mail; separator="", "") AS ?mail)
WHERE {
  OPTIONAL { ?person rdfs:label ?name }
  OPTIONAL { ?person foaf:age ?age }
  OPTIONAL { ?person foaf:mbox ?mail }
} GROUP BY ?person ?name ?age
```
Task: write a SPARQL query that extracts a one-row per person summary!

```sparql
SELECT ?person ?name ?age (GROUP_CONCAT(?mail; separator="", ") AS ?mail)
WHERE {
  OPTIONAL { ?person rdfs:label ?name }
  OPTIONAL { ?person foaf:age ?age }
  OPTIONAL { ?person foaf:mbox ?mail } }
GROUP BY ?person ?name ?age
```
Constrained RDF-graph

Definition of constraints:

CLASS foaf:Person {
    KEY rdfs:label : LITERAL
    TOTAL foaf:mbox : LITERAL
    PARTIAL foaf:age :
        LITERAL(xsd:integer)}

Task: write a SPARQL query that extracts a one-row per person summary!

    OPTIONAL { ?person foaf:age ?age } }
related work

- SPIN
- Resource Shape
- Stardog ICV
2. **RDF Data Description Constraint Language**

RDD semantics is based on FOL - for constraint checking SPARQL ASK-queries are generated. RDD prototype exists and is currently evaluated.

- **Class constraints:**
  - SINGLETON, SUBCLASS, KEY

- **Property constraints:**
  - MIN, MAX, DOMAIN, RANGE, PATH, SUBPROPERTY, TOTAL, PARTIAL
2. **RDF Data Description Constraint Language**

RDD semantics is based on FOL - for constraint **checking** SPARQL ASK-queries are generated. RDD prototype exists and is currently evaluated.

- **Class constraints:**
  - SINGLETON, SUBCLASS, KEY

- **Property constraints:**
  - MIN, MAX, DOMAIN, RANGE, PATH, SUBPROPERTY, TOTAL, PARTIAL
RDD distinguishes

- *qualified* constraints, i.e. assigned to classes, from *unqualified*, i.e. global constraints;
- *mandatory definition* of constraints, i.e. every property has to be covered by some constraint thereby enforcing a closed-world assumption (CWA), from *optional definition*, not every property has to be covered by a constraint thereby allowing an open-world assumption (OWA).

RDD distinguishes

- *qualified* constraints, i.e. assigned to classes, from *unqualified*, i.e. global constraints;
- *mandatory definition* of constraints, i.e. every property has to be covered by some constraint thereby enforcing a closed-world assumption (CWA), from *optional definition*, not every property has to be covered by a constraint thereby allowing an open-world assumption (OWA).

Formal semantics: *MS and GL, Pleasantly Consuming Linked Data with RDF Data Descriptions, CoRR abs/1307.3419v2, 2013 (COLD Workshop 2013)*
RDD in Datalog+/-

**Range-constraint** \( \text{range}(C, p, R) \):
\[
G(s, \text{type}, C), G(s, p, o) \rightarrow G(o, \text{type}, R)
\]

**Min-constraint** \( \text{min}(C, p, n), n \geq 1 \):
\[
G(s, \text{type}, C) \rightarrow \exists o_1, \ldots, o_n \ (G(s, p, o_1), \ldots, G(s, p, o_n), \text{allDist}(o_1, \ldots, o_n))
\]

**Max-constraint** \( \text{max}(C, p, n), n \geq 1 \):
\[
G(s, \text{type}, C), G(s, p, o_1), \ldots, G(s, p, o_{n+1}) \rightarrow \text{someEq}(o_1, \ldots, o_{n+1})
\]

**Partiality-constraint** \( \text{partial}(C, p) : \text{max}(C, p, 1) \)

**Totality-constraint** \( \text{total}(C, p) : \text{min}(C, p, 1), \text{max}(C, p, 1) \)

**Path-constraint** \( \text{path}(C, p, q_1, \ldots, q_n) \):
\[
G(s, \text{type}, C), G(s, p, o) \rightarrow \exists o_1, \ldots, o_{n-1} \ (G(s, q_1, o_1), \ldots, G(o_{n-1}, q_n, o))
\]

**Subproperty-constraint** \( \text{subprop}(C, p, p_s) \):
\[
G(s, \text{type}, C), G(s, p_s, o) \rightarrow G(s, p, o)
\]

**Key-constraint** \( \text{key}(C, p_1, \ldots, p_n) \)

**Singleton-constraint** \( \text{single}(C) \)

... checking can be done using SPARQL ASK-queries.
### RDD in Datalog+-/

<table>
<thead>
<tr>
<th>Constraint Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Range-constraint</strong></td>
<td>[ \text{range}(C, p, R) : \ G(s, \text{type}, C), G(s, p, o) \rightarrow G(o, \text{type}, R) ]</td>
</tr>
<tr>
<td><strong>Min-constraint</strong></td>
<td>[ \text{min}(C, p, n), n \geq 1 : \ G(s, \text{type}, C) \rightarrow \exists o_1, \ldots, o_n \ (G(s, p, o_1), \ldots, G(s, p, o_n), \text{allDist}(o_1, \ldots, o_n)) ]</td>
</tr>
<tr>
<td><strong>Max-constraint</strong></td>
<td>[ \text{max}(C, p, n), n \geq 1 : \ G(s, \text{type}, C), G(s, p, o_1), \ldots, G(s, p, o_{n+1}) \rightarrow \text{someEq}(o_1, \ldots, o_{n+1}) ]</td>
</tr>
<tr>
<td><strong>Partiality-constraint</strong></td>
<td>[ \text{partial}(C, p) : \text{max}(C, p, 1) ]</td>
</tr>
<tr>
<td><strong>Totality-constraint</strong></td>
<td>[ \text{total}(C, p) : \text{min}(C, p, 1), \text{max}(C, p, 1) ]</td>
</tr>
<tr>
<td><strong>Path-constraint</strong></td>
<td>[ \text{path}(C, p, q_1, \ldots, q_n) : \ G(s, \text{type}, C), G(s, p, o) \rightarrow \exists o_1, \ldots, o_{n-1} \ (G(s, q_1, o_1), \ldots, G(o_{n-1}, q_n, o)) ]</td>
</tr>
<tr>
<td><strong>Subproperty-constraint</strong></td>
<td>[ \text{subprop}(C, p, p_s) : \ G(s, \text{type}, C), G(s, p_s, o) \rightarrow G(s, p, o) ]</td>
</tr>
<tr>
<td><strong>Key-constraint</strong></td>
<td>[ \text{key}(C, p_1, \ldots, p_n) ]</td>
</tr>
<tr>
<td><strong>Singleton-constraint</strong></td>
<td>[ \text{single}(C) ]</td>
</tr>
</tbody>
</table>

... checking can be done using SPARQL ASK-queries.
RDD in Datalog+-/

**Range-constraint** \(\text{range}(C, p, R)\):

\[G(s, \text{type}, C), G(s, p, o) \rightarrow G(o, \text{type}, R)\]

**Min-constraint** \(\text{min}(C, p, n), n \geq 1\):

\[G(s, \text{type}, C) \rightarrow \exists o_1, \ldots, o_n (G(s, p, o_1), \ldots, G(s, p, o_n), \text{allDist}(o_1, \ldots, o_n))\]

**Max-constraint** \(\text{max}(C, p, n), n \geq 1\):

\[G(s, \text{type}, C), G(s, p, o_1), \ldots, G(s, p, o_{n+1}) \rightarrow \text{someEq}(o_1, \ldots, o_{n+1})\]

**Partiality-constraint** \(\text{partial}(C, p) : \text{max}(C, p, 1)\)

**Totality-constraint** \(\text{total}(C, p) : \text{min}(C, p, 1), \text{max}(C, p, 1)\)

**Path-constraint** \(\text{path}(C, p, q_1, \ldots, q_n)\):

\[G(s, \text{type}, C), G(s, p, o) \rightarrow \exists o_1, \ldots, o_{n-1} (G(s, q_1, o_1), \ldots, G(o_{n-1}, q_n, o))\]

**Subproperty-constraint** \(\text{subprop}(C, p, p_s)\):

\[G(s, \text{type}, C), G(s, p_s, o) \rightarrow G(s, p, o)\]

**Key-constraint** \(\text{key}(C, p_1, \ldots, p_n)\)

**Singleton-constraint** \(\text{single}(C)\)

... checking can be done using SPARQL ASK-queries.
3. More than one RDF graph, more than one RDD

RDF graph A

RDF graph B
3. More than one RDF graph, more than one RDD

RDF graph A

RDF graph B
... can be accessed and processed in advance:

**RDD A**

PREFIX ex: <http://www.exple.com#>
CWA CLASS ex:Employee {  
  KEY rdfs:label : LITERAL  
  PARTIAL ex:employedBy : RESOURCE  
  MAX(2) ex:prevEmployedBy : RESOURCE  
  TOTAL ex:worksFor,  
    RANGE (ex:Project)  
  PATH(ex:worksFor/ex:assignedTo)  
    ex:reportsTo,  
    RANGE(ex:Consortium) }

CWA CLASS ex:Project {  
  TOTAL ex:assignedTo,  
  RANGE(ex:Consortium) }

**RDD B**

PREFIX ex: <http://www.exple.com#>
CWA CLASS ex:Employee {  
  KEY rdfs:label : LITERAL  
  PARTIAL ex:employedBy : RESOURCE  
  ex:prevEmployedBy : RESOURCE,  
  SUBPROPERTY ex:employedBy  
    MIN(2), MAX(2) ex:reportsTo,  
    RANGE(ex:Association) }

...what constraints can be assumed for the merge of the RDF-graphs?
... can be accessed and processed in advance:

**RDD A**

PREFIX ex: <http://www.exple.com#>
CWA CLASS ex:Employee {
  KEY rdfs:label : LITERAL
  PARTIAL ex:employedBy : RESOURCE
  MAX(2) ex:prevEmployedBy : RESOURCE
  TOTAL ex:worksFor,
    RANGE (ex:Project)
  PATH(ex:worksFor/ex:assignedTo)
    ex:reportsTo,
    RANGE(ex:Consortium) }

CWA CLASS ex:Project {
  TOTAL ex:assignedTo,
    RANGE(ex:Consortium) }

**RDD B**

PREFIX ex: <http://www.exple.com#>
CWA CLASS ex:Employee {
  KEY rdfs:label : LITERAL
  PARTIAL ex:employedBy : RESOURCE
  MAX(2) ex:prevEmployedBy : RESOURCE
    SUBPROPERTY ex:employedBy
  ex:prevEmployedBy : RESOURCE,
    RANGE(ex:Consortium)
  MIN(2), MAX(2) ex:reportsTo,
    RANGE(ex:Association) }

...what constraints can be assumed for the merge of the RDF-graphs?
RDD A and RDD B are conflicting

A Whenever reportsTo is defined, because of the functionality of worksFor and assignedTo, it must be functional;

B by reportsTo, each employee reports to exactly 2 associations.

A By prevEmployedBy, for each employee at most 2 previous employers are recorded;

B as prevEmployedBy is a subproperty of employedBy, which is a partial function, for each employee at most 1 previous employer is recorded.
RDD A and RDD B are conflicting

A Whenever reportsTo is defined, because of the functionality of worksFor and assignedTo, it must be functional;

B by reportsTo, each employee reports to exactly 2 associations.

A By prevEmployedBy, for each employee at most 2 previous employers are recorded;

B as prevEmployedBy is a subproperty of employedBy, which is a partial function, for each employee at most 1 previous employer is recorded.
Problem statement

- Given RDDs $\Sigma_a$ and $\Sigma_b$, we are interested to construct an RDD $\Gamma(\Sigma_a, \Sigma_b)$,
- such that for any RDF graphs $G_a$ and $G_b$, where $G_a \models \Sigma_a$ and $G_b \models \Sigma_b$, we have $G_a \models \Gamma(\Sigma_a, \Sigma_b)$, $G_b \models \Gamma(\Sigma_a, \Sigma_b)$ and $G_a \cup G_b \models \Gamma(\Sigma_a, \Sigma_b)$.

Properties of $\Gamma(\Sigma_a, \Sigma_b)$?
Problem statement

- Given RDDs $\Sigma_a$ and $\Sigma_b$, we are interested to construct an RDD $\Gamma(\Sigma_a, \Sigma_b)$,
- such that for any RDF graphs $G_a$ and $G_b$, where $G_a \models \Sigma_a$ and $G_b \models \Sigma_b$,
- we have $G_a \models \Gamma(\Sigma_a, \Sigma_b)$, $G_b \models \Gamma(\Sigma_a, \Sigma_b)$ and $G_a \cup G_b \models \Gamma(\Sigma_a, \Sigma_b)$.

Properties of $\Gamma(\Sigma_a, \Sigma_b)$?
First step

Detect and resolve conflicts between cardinality constraints (min, max, total, partial).

- Conflict detection is based on the implication problem of cardinality constraints (with respect to the set of all constraints).
- Conflict resolution is based on aggregation of cardinality constraints.
Conflict resolution (1)

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Whenever <code>reportsTo</code> is defined, because of the functionality of <code>worksFor</code> and <code>assignedTo</code>, it must be functional;</td>
</tr>
<tr>
<td>B</td>
<td>by <code>reportsTo</code>, each employee reports to exactly 2 associations.</td>
</tr>
</tbody>
</table>

by aggregation: \( \min(\text{reportsTo}) = 0, \max(\text{reportsTo}) = 3. \)
Conflict resolution (1)

A Whenever \textit{reportsTo} is defined, because of the functionality of \textit{worksFor} and \textit{assignedTo}, it must be functional;

B by \textit{reportsTo}, each employee reports to exactly 2 associations.

by aggregation: $\min(\text{reportsTo}) = 0$, $\max(\text{reportsTo}) = 3$. 
Conflict resolution (2)

A By prevEmployedBy, for each employee at most 2 previous employers are recorded;

B as prevEmployedBy is a subproperty of employedBy, which is a partial function, for each employee at most 1 previous employer is recorded.

by aggregation: \( \min(\text{prevEmployedBy}) = 0, \max(\text{prevEmployedBy}) = 3 \).
### Conflict resolution (2)

- **A**: By `prevEmployedBy`, for each employee at most 2 previous employers are recorded;
- **B**: as `prevEmployedBy` is a subproperty of `employedBy`, which is a partial function, for each employee at most 1 previous employer is recorded.

By aggregation: $min(prevEmployedBy) = 0$, $max(prevEmployedBy) = 3$. 
RDF Data Descriptions

Heuristic *Cardinality Constraint Aggregation*

Algorithm `basicConstAgg`:
Input: RDDs $\Sigma_1, \Sigma_2$
Output: RDD $\Gamma(\Sigma_1, \Sigma_2)$ of cardinality constraints

$\Gamma := \emptyset$

for $\Sigma_i, i \in \{1, 2\}$
for each predicate $p$
    if $p$ does not have a min-constraint
        if $min(p, 1) \in \Sigma_i^*$ then add $min(p, 1)$ to $\Gamma$ else add $min(p, 0)$
    if $p$ does not have a max-constraint
        if $max(p, 1) \in \Sigma_i^*$ then add $max(p, 1)$ to $\Gamma$ else add $max(p, \infty)$

for each predicate $p$ in $\Sigma_a$ and $\Sigma_b$ add `aggregate(p)` to $\Gamma$
for each not considered predicate $p$ add the respective cardinality constraints to $\Gamma$

$\Gamma(\Sigma_1, \Sigma_2) := \Gamma$
inferring $\min(1)/\max(1)$

Chase the body of the constraint and check whether it is contained in the head.

Will the Chase terminate?

- The rules are bounded,
- however not innocuous.

Remember, RDD was defined for *checking*!
inferring \( \text{min}(1)/\text{max}(1) \)

Chase the body of the constraint and check whether it is contained in the head.

Will the Chase terminate?
- The rules are bounded,
- however not innocuous.

Remember, RDD was defined for checking!
inferring $min(1)/max(1)$

Chase the body of the constraint and check whether it is contained in the head.

Will the Chase terminate?
- The rules are bounded,
- however not innocuous.

Remember, RDD was defined for checking!
... not innocuous

constraints: $\min(C, p, 1)$, $\text{path}(C, p, q_1, q_2)$, $\text{path}(C, p, q_1, q_3)$, $\max(C, q_1, 1)$

$G(s, \text{type}, C),$

$G(s, p, o),$

$G(s, q_1, o_1), G(o_1, q_2, o)$

$G(s, q_1, o_2), G(o_2, q_3, o)$

$o_1 = o_2$

$G(o_1, q_3, o)$
... not innocuous

constraints: $min(C, p, 1)$, $path(C, p, q_1, q_2)$, $path(C, p, q_1, q_3)$, $max(C, q_1, 1)$

$$G(s, type, C),$$

$$G(s, p, o),$$

$$G(s, q_1, o_1), G(o_1, q_2, o)$$

$$G(s, q_1, o_2), G(o_2, q_3, o)$$

$$o_1 = o_2$$

$$G(o_1, q_3, o)$$
... not innocuous

constraints: $\min(C, p, 1)$, $\text{path}(C, p, q_1, q_2)$, $\text{path}(C, p, q_1, q_3)$, $\max(C, q_1, 1)$

$G(s, \text{type}, C)$,
$G(s, p, o)$,
$G(s, q_1, o_1)$, $G(o_1, q_2, o)$
$G(s, q_1, o_2)$, $G(o_2, q_3, o)$
$o_1 = o_2$
$G(o_1, q_3, o)$
4. Conclusion

Related to, e.g.
- Information integration
- Data exchange
- Implication of cardinality constraints
- RDF constraints
- ...

... in our case target constraints are initially unknown and have to be derived, respectively the constraint language is different.
4. Conclusion

Related to, e.g.
- Information integration
- Data exchange
- Implication of cardinality constraints
- RDF constraints
- ...

... in our case target constraints are initially unknown and have to be derived, respectively the constraint language is different.