OSLC Resource Shape: A Linked Data Constraint Language

Achille Fokoue
Research Staff Member
IBM Research

Arthur Ryman
Distinguished Engineer
IBM

Introduction

Linked Data has emerged as a principled, flexible, open standard based approach to share, integrate and link heterogenous and disparate data across tools, data silos within an organization, and across organizations. It leverages well-established web architecture concepts (e.g., resources identified by URI, and HTTP communication protocol), and introduces RDF as its fundamental data model and SPARQL as its query language. While much attention has been devoted to the use of Linked Data principles to share and access information in the growing Web of Data[1], at IBM, we have successfully relied on the Linked Data approach as an architectural style for addressing the issue of integrating a suite of applications [1].

In the context of Linked Data for Application Integration, a key challenge that we had to address was the lack of any standard mechanism to validate RDF data. Standard web ontology languages (e.g., RDF Schema and OWL) have been well documented [2, 3, 4, 5, 6] as inadequate as validation and integrity constraint languages for RDF. They were primarily designed for distributed environments where any single information provider or consumer is only assumed to have partial information. For example, their Open World Assumption (OWA[2]) semantics and the absence of Unique Name Assumption (UNA[3]) trigger the inference of new facts where the Closed World Assumption (CWA) of traditional constraint languages (e.g., XML Schema or RelaxNG for XML, Data Definition Language for relational data, etc) would directly trigger constraint violations.

In this paper, we start by motivating the need for a constraint language for RDF data. We then present some important requirements for such a language. Finally, we briefly introduce OSLC Resource Shape, our solution for addressing RDF data validation needs that was developed in context of using Linked Data for application integration.

Motivation and Requirements for a Linked Data Constraint Language

A Linked Data constraint language serves two important functions. First, it provides a contract language needed to define the structure and constraints on the input and output of REST service operations. Second, it can be used as metadata by other tools (e.g. query builders, form builders, test case generators, etc).

Defining REST Linked Data Interfaces

Linked Data fuses REST and RDF by requiring that resources be identified with dereferenceable HTTP

1 http://linkeddata.org/
2 In OWA, as opposed to Closed World Assumption (CWA), a statement that cannot be proven to hold based solely on partial information available to an agent is never assumed, by this agent, to be false.
3 The absence of UNA implies that two different URIs could be used to refer to the same resource.

URIs and that HTTP clients are able to get RDF representations of resources.

From a REST perspective, the definition of the content and structure of RDF payloads (HTTP request or response) is a key part of the definition of the REST service interface. It is sound engineering practice to define interfaces between components in a system. The interface definition defines the contract between the provider and consumer of a component. For software systems, the main part of the interface definition is a precise specification of the inputs and outputs. Type definition languages are used for this purpose. Such constraint language is needed in Linked Data to formally describe the structure of RDF graphs produced by a provider so that a consumer can automatically check whether RDF graphs it received conform to the interface contract.

Metadata for Tools Handling RDF Graphs

As for more traditional type definition languages (e.g., XML Schema, Relational Data Definition Language), constraints on RDF graphs can serve as metadata that can be used by various tools to achieve different goals. Some of the tools include: query and form builders and test case generators. They all need to exploit RDF graph constraints to understand, for example, the expected properties and their allowed values in order to achieve their goal: e.g.,

1. assist users in the formulation of their queries
2. generate appropriate forms in an editor, and
3. automatically generate test cases based on the expected structure of the input/ouput of an application, tool, or REST service operation.

Requirements for a Linked Data Constraint Language

Using a Linked Data constraint language for defining REST Linked Interfaces and metadata dictates two important requirements on the constraint language:

- It must be amenable to efficient implementation
- It must be expressed at a high level of abstraction to be understandable by both humans and machines

First, since the validation of RDF graphs is a key and frequent operation performed by both data consumers and producers, a RDF constraint language should be amenable to very efficient implementation, preferably on top of existing technologies. In particular, in the proposed constraint language described in the next section, this requirement for efficient implementation was fulfilled by a translation of all constraints into simple SPARQL ASK queries that can be efficiently answered by existing SPARQL engines.

Second, the constraint language has to be expressed at a high enough level of abstraction so that constraints can be successfully used as metadata easily understandable by both humans and machines. On the one hand, developers need to understand the REST interface constraints in order to develop clients and providers of a Linked Data REST API. On the other hand, tools (e.g., query and form builders, test case generators) need to understand, without complex static analysis, the expected structure of RDF graphs. This requirement explains why, although SPARQL ASK queries can be used to check whether RDF graphs conform to specific constraints, SPARQL language itself is not appropriate as the constraint language since it is too low level.

OSLC Resource shape

The Open Services for Lifecycle Collaboration (OSLC) has proposed the Resource Shape specification [7] for specifying constraints on RDF data. This proposal, which we advocate in this paper as a viable solution for a constraint language for RDF, fulfills the requirements highlighted in the previous section.

A resource shape specification lists the properties that are expected or required in a graph, their occurrence, range, allowed values, and so forth. A shape specification determines whether a given graph is valid or invalid. A shape checker could be implemented as a set of SPARQL ASK queries on the graph. The SPARQL ASK query corresponding to a shape constraint captures the pattern(s) that is (are) forbidden in valid RDF graphs. If all queries return false, the graph is valid; otherwise, it is invalid because the constraints corresponding to the ASK queries returning true are not satisfied.

We illustrate some important aspect of the Resource Shape language on a simple example (the formal specification of the language is available at [7]). Consider a simple web application that hosts resources about change requests. We will use the OSLC oslc_cm:ChangeRequest class to define the class of change requests. Assume that there is a REST service where we can POST HTTP requests to create new oslc_cm:ChangeRequest resources. The REST service looks at the HTTP request, and if it contains an oslc_cm:ChangeRequest resource, it will create a new resource and copy the properties from the HTTP request to it. The following HTTP POST request body should succeed:

HTTP POST changeRequest.ttl

```
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix oslc_cm: <http://open-services.net/ns/cm#> .

<http://example.com/resource> a oslc_cm:ChangeRequest ;
    dcterms:title "Null pointer exception in web ui" ;
    oslc_cm:status "Submitted" .
```

In the example, it is required that when a new resource is created, it must have exactly one dcterms:title property and zero or one oslc_cm:status property. These constraints are expressed in the simplified shape resource shown below (changeRequest-shape.ttl):

OSLC Resource Shape changeRequest-shape.ttl

```
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix oslc: <http://open-services.net/ns/core#> .
@prefix oslc_cm: <http://open-services.net/ns/cm#> .

@base <http://example.com/shape/oslc-change-request> .

<> a oslc:ResourceShape ;
    dcterms:title "Creation shape of OSLC Change Request" ;
    oslc:describes oslc_cm:ChangeRequest ;
    oslc:property <#dcterms-title>, <#oslc_cm-status> .

<#dcterms-title> a oslc:Property ;
    oslc:propertyDefinition dcterms:title ;
    oslc:occurs oslc:Exactly-one .

<#oslc_cm-status> a oslc:Property ;
    oslc:propertyDefinition oslc_cm:status ;
```

```
    oslc:occurs oslc:Zero-or-one .
```

This shape document specifies constraints to be applied as preconditions to creating oslc_cm:ChangeRequest resource through HTTP POST. It uses the oslc:occurs property to specify the occurrence constraints of the dcterms:title and oslc_cm:status properties. Specifying the occurrence of a property as either oslc:Exactly-one or oslc:Zero-or-one constrains the property to be functional.

As mentioned above, each constraint can be expressed as a SPARQL ASK query that captures patterns forbidden in the RDF data. For example, the following query, ask-oslc_cm-status-occurs.rq, checks the occurrence of the oslc_cm:status property:

SPARQL Query ask-oslc_cm-status-occurs.rq

```
prefix oslc_cm: <http://open-services.net/ns/cm#>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

ASK
{
    ?resource rdf:type oslc_cm:ChangeRequest.
    ?resource oslc_cm:status ?status1.
    ?resource oslc_cm:status ?status2.
    FILTER (?status1 != ?status2)
}
```

The above SPARQL ASK query returns true (meaning an invalid pattern has been detected in the RDF data) if and only if there is at least one oslc_cm:ChangeRequest that specifies two or more distinct values for the property oslc_cm:status.

Likewise, the following query, ask-dcterms-title.rq, checks the occurrence of the dcterms-title property, and it returns true if and only if there is a oslc_cm:ChangRequest that does not specify exactly one dcterms:title:

SPARQL Query ask-dcterms-occurs.rq

```
prefix oslc_cm: <http://open-services.net/ns/cm#>
prefix dcterms: <http://purl.org/dc/terms/>
prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

ASK
{
     ?resource rdf:type oslc_cm:ChangeRequest.
     {
        ?resource dcterms:title ?title1.
        ?resource dcterms:title ?title2.
        FILTER (?title1 != ?title2)
     }
     UNION
     {
        FILTER NOT EXISTS { ?resource dcterms:title ?title}

     }

}
```

The OSLC Resource Shape specification enables the expression of common constraints in addition to occurrence constraints (see [7] for more details). Although the semantic of each constraint can be expressed in terms of a suitable SPARQL ASK query, implementations of the specification are not required to use SPARQL to check constraints.

Conclusion

In this paper, we have motived the need for a RDF constraint language for defining REST Linked Data interfaces and metadata that can be leveraged by tools handling RDF data. We have also presented key requirements on such language; namely, its amenability to efficient implementation (e.g., via translation to SPARQL ASK queries), and its ability to express constraints at a high level of abstraction so that they are easily understandable by both humans and machines. Finally, we advocate the use of the Open Services for Lifecycle Collaboration (OSLC) Resource Shape specification [7] as a viable solution for a RDF constraint language.

References:

[1] Arnaud Le Hors, Martin Nally, Steve Speicher: Using read/write Linked Data for Application Integration – Towards a Linked Data Basic Profile. LDOW.
http://events.linkeddata.org/ldow2012/papers/ldow2012-paper-04.pdf
[2] Arthur Ryman: Linked Data Interfaces. DeveloperWorks.
http://www.ibm.com/developerworks/rational/library/linked-data-oslc-resource-shapes/
[3] Arthur Ryman, Arnaud Le Hors, Steve Speicher: OSLC Resource Shape
A language for defining constraints on Linked Data. LDOW.
http://events.linkeddata.org/ldow2013/papers/ldow2013-paper-02.pdf
[4] Jiao Tao, Evren Sirin, Jie Bao, Deborah L. McGuinness: Extending OWL with Integrity Constraints. Description Logics 2010. http://ceur-ws.org/Vol-573/paper_21.pdf
[5] Jiao Tao: Adding Integrity Constraints to the Semantic Web for Instance Data Evaluation. International Semantic Web Conference (2) 2010: 330-337.
http://rd.springer.com/chapter/10.1007%2F978-3-642-17749-1_24
[6] Jiao Tao, Evren Sirin, Jie Bao, Deborah L. McGuinness: Integrity Constraints in OWL. AAAI 2010.
http://www.aaai.org/ocs/index.php/AAAI/AAAI10/paper/view/1931
[7] OSLC Resource Shape: http://open-services.net/bin/view/Main/OSLCCoreSpecAppendixA?sortcol=table;up=#oslc_ResourceShape_Resource