Validating and Describing Linked Data Portals using RDF Shape Expressions

Jose Emilio Labra Gayo
University of Oviedo
Dept. of Computer Science
C/Calvo Sotelo, S/N
labra@uniovi.es

Eric Prud'hommeaux
World Wide Web Consortium (W3C) MIT,
Cambridge, MA, USA
eric@w3.org

Harold Solbrig
Mayo Clinic
College of Medicine, Rochester, MN, USA

Jose María Álvarez Rodríguez
Dept. Computer Science
Carlos III University
josemaria.alvarez@uc3m.es

ABSTRACT

In order to improve the quality of linked data portals, it is necessary to have a tool that can automatically describe and validate the RDF triples exposed.

RDF Shape Expressions have been proposed as a language based on Regular Expressions that can describe and validate the structure of RDF graphs.

In this paper we describe the WebIndex, a medium sized linked data portal, and how we have employed Shape Expressions to document its contents and to automatically validate the shapes of the resources.

Categories and Subject Descriptors

I.2.4 [**Knowledge Representation Formalisms and Methods**]: Representation languages; H.3.5 [**Online Information Services**]: Web-based services

General Terms

Theory

Keywords

RDF, Graphs, Validation, Transformation

1. INTRODUCTION

Linked Data portals have emerged as a way to publish data on the Web following a set of principles [1] which improve data reuse and integration. As indicated in [2], linked data relies on documents using RDF representations to make typed statements that link arbitrary things in the world. RDF appears as a data integration language and some linked data applications use RDF as a database technology or as an interoperability layer. However, there is a lack of an accepted practice to declare and constrain the shape of an RDF graph in a way that can be automatically validated to augment the quality of RDF based data portals.

Validation is standard practice in other conventional data languages. Industrial setting count on parsing grammars for domain-specific languages, DDL constraints in SQL databases, and W3C XML Schema or RelaxNG for XML documents.

In the case of RDF, although there are standards for inference like RDF Schema and OWL, these technologies employ Open World and Non-Unique Name Assumptions that create difficulties for validation purposes [18].

The RDF Shape Expressions language is intended to perform the same function for RDF graphs as Schema languages to XML. It can be used to validate documents, communicate expected graph patterns for interfaces, and generate user interface forms and code.

The syntax and semantics of Shape Expressions are designed to be familiar to users of regular expressions (specially RelaxNG). The main difference is that RDF data is a set (of triples) while regular expression data is a sequence (of characters). Regular expressions correlate an ordered pattern of atomic characters and logical operators against an ordered sequence of characters while Shape Expressions correlate an ordered pattern of pairs of predicate and object classes and logical operators against an *unordered* set of arcs in a graph.

In this paper we propose the use of RDF Shape Expressions to describe the contents of Linked Data portals in a way that can be automatically validated.

As a use case, we will describe the development of the 2013 WebIndex data portal[1], which is a linked data portal of medium size (around 3.5 million of triples) that contains information about the statistical computations that have been carried on to generate the WebIndex. We have selected this use case because it contains a data model with interrelated shapes and reuses several existing vocab-

[1] `http://data.webfoundation.org/webindex/v2013`

ularies like RDF Data Cube, Organization Ontology, Dublin Core, etc.

2. WEBINDEX DATA MODEL

The WebIndex is a multi-dimensional measure of the World Wide Web's contribution to development and human rights globally. It covers 81 countries and incorporates indicators that assess several areas like universal access; freedom and openness; relevant content; and empowerment[2].

The 2012 version offered a data portal where the data was obtained by transforming raw observations and precomputed values from Excel sheets to RDF. The 2013 version of the WebIndex data portal employs a new validation and computation approach that tries to obtain a verifiable linked data version of the Web Index data.

The WebIndex data model is based on the RDF Data Cube vocabulary. Figure 1 represents the main concepts of the data model[3].

As can be seen, the main concept are observations of type `qb:Observation` which have a float value `cex:value` and are related to a country, a year a dataset and an indicator.

A dataset contains a number of slices, each of which also contains a number of observations.

Indicators are provided by an organization of type `org:Organization` which employs the Organization ontology[15]. Datasets are also published by organizations.

As a sample of some data, an observation can be that Spain has value 23.78 in 2011 for the indicator ITU-B (*Broadband subscribers per 100 population*) in the dataset DITU provided by ITU (International Telecommunication Union). This information can be represented in RDF using Turtle syntax as[4]:

```
obs:obs8165 a qb:Observation ;
 rdfs:label "ITU B in ESP, 2009" ;
 dct:issued
   "2013-05-30T09:15:00"^^xsd:dateTime ;
 cex:indicator indicator:ITU_B ;
 qb:dataSet dataset:DITU ;
 cex:value 23.78^^xsd:float ;
 cex:ref-area country:Spain ;
 cex:ref-year 2011 ;
 ...other properties omitted for brevity
 .
```

Notice that the WebIndex data model contains data that is completely interrelated. Observations are linked to indicators and datasets. Datasets contain also links to slices and slices have links to indicators and observations again. Both datasets and indicators are linked to the organizations that publish or provide them.

The following example contains a sample of interrelated data for this domain.

[2]`http://thewebindex.org`
[3]In the paper we will employ common prefixes that can be found in `http://prefix.cc`
[4]The URI of the real observation is `http://data.webfoundation.org/webindex/v2013/observation/obs8165`

```
dataset:DITU a qb:DataSet ;
 rdfs:label "ITU Dataset" ;
 dct:publisher org:ITU ;
 qb:slice slice:ITU09B ,
          slice:ITU10B,
          ...;
 ...
slice:ITU09B a qb:Slice ;
 qb:sliceStructure wf:sliceByArea ;
 qb:observation obs:obs8165,
                obs:obs8166,
                ...
 ...
org:ITU a org:Organization ;
 rdfs:label "ITU" ;
 foaf:homepage <http://www.itu.int/>
 .
country:Spain a wf:Country ;
 wf:iso2 "ES" ; wf:iso3 "ESP" ;
 rdfs:label "Spain"
 .
```

A validator of this model can validate the simple structure of each type of resource but it would be better if it can declare and detect all these interrelationships.

The WebIndex data model also includes a linked data representation of computations so it is possible to declare the data from which a value has been computed so it can be checked. We omit those classes for brevity.

3. USING SHAPE EXPRESSIONS TO DESCRIBE THE WEBINDEX DATA MODEL

In this section we will describe the WebIndex data model using Shape Expressions.

The RDF Shape Expressions language is inspired by RelaxNG and also has two syntaxes: a compact one and an RDF serialization. The compact syntax is more oriented towards human readability while the RDF serialization can be employed to exchange and store shape expressions using standard semantic web tools. A primer to the Shape Expressions language can be found at `http://www.w3.org/2013/ShEx/Primer`.

A shape expression is a labelled pattern for a set of RDF Triples sharing a common subject. Syntactically, it is a pairing of a label, which can be an IRI or a blank node, and a rule enclosed in brackets (`{ }`). Typically, this rule is a conjunction of constraints separated by commas (`,`). For example, we can declare the shape of a country as:

```
<Country> {
  a (wf:Country)
, rdfs:label xsd:string
, wf:iso2 xsd:string
, wf:iso3 xsd:string
}
```

The above declaration indicates that a country must have `rdf:type` with value `wf:Country`. It must also have the properties

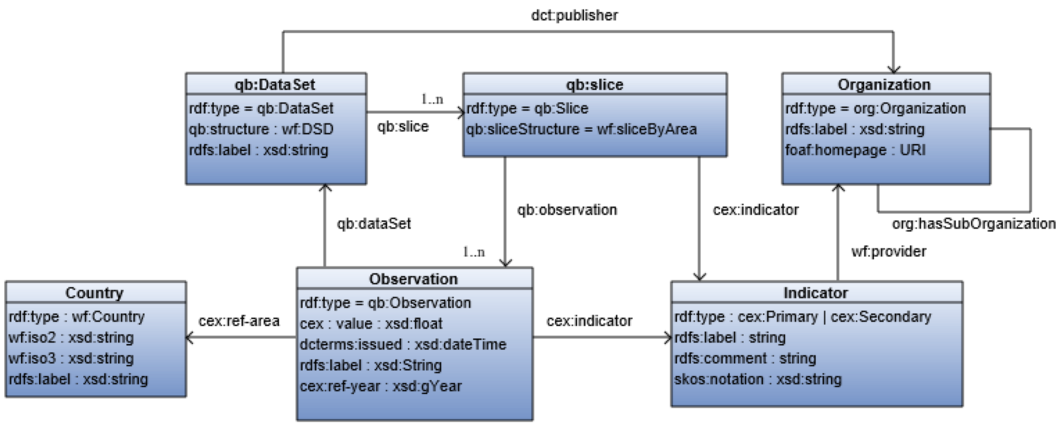


Figure 1: Simplified WebIndex data model

rdfs:label, wf:iso2 and wf:iso3 with a value of type xsd:string.

The semantics of Shape Expression validation acts as a type inference system which infers a type (shape) for a given node in an RDF graph.

With the previous declaration, a Shape Expressions validator would infer:

$$\texttt{country:Spain} \quad \rightsquigarrow \quad \texttt{<Country>}$$

The Shape Expressions language is inspired by Regular Expressions and the rules can contain cardinality constraints with the values + (one or more), $\star$ (zero of more), ? (zero or one) and even ranges {m,n} (between m and n repetitions).

It is also possible to declare that the value of some property has a given shape using the @ character.

For example, the shape of datasets can be described as:

```
<DataSet> {
   a (qb:DataSet)
 , qb:structure (wf:DSD)
 , rdfs:label xsd:string?
 , qb:slice @<Slice>+
 }
```

which declares that a dataset must have rdf:type with value qb:DataSet and qb:structure with value wf:DSD. It may have a rdfs:label with a value of type xsd:string and must have one or more slices with the shape Slice.

In a similar way, it is possible to declare slices as:

```
<Slice> {
   a (qb:Slice)
 , qb:sliceStructure ( wf:sliceByYear )
 , qb:observation @<Observation>+
 , cex:indicator @<Indicator>
 }
```

The declarations for observations and indicators are similar:

```
<Observation> {
   a (qb:Observation)
 , cex:value xsd:float
 , dct:issued xsd:dateTime
 , rdfs:label xsd:string?
 , qb:dataSet @<DataSet>
 , cex:ref-area @<Country>
 , cex:indicator @<Indicator>
 , cex:ref-year xsd:gYear
 }
```

```
<Indicator> {
   a ( wf:PrimaryIndicator
       wf:SecondaryIndicator
     )
 , rdfs:label xsd:string
 , rdfs:comment xsd:string ?
 , skos:notation xsd:string ?
 }
```

Finally, organizations can be declared as:

```
<Organization> {
   a ( org:Organization )
 , rdfs:label xsd:string
 , foaf:homepage IRI
 , org:hasSubOrganization
         @<Organization>
 }
```

As can be seen, Shape Expressions offer an intuitive way to describe the contents of linked data portals. In fact, we have employed Shape Expressions to document both the WebIndex[5] and Landbook[6] data portals. The documentation defines templates for the different shapes of resources and for the triples that can be retrieved when dereferencing those resources.

[5]http://weso.github.io/wiDoc
[6]http://weso.github.io/landportalDoc/data

These templates define the dataset structure in an intuitive way and can be used to act as a contract between developers of the data portal. We noted that having a good data model with its corresponding Shape Expressions specification facilitated the communication between the different teams involved in the development of the data portal.

4. IMPLEMENTATIONS OF SHAPE EXPRESSIONS

Currently, there are four implementations of Shape Expressions in progress:

- *FancyShExDemo*[7] was the first prototype implementation in Javascript. It handles semantic actions which can be used to extend the semantics of shape expressions and even to transform RDF to XML or JSON. It supports a form-based system with dynamic validation during the edition process and SPARQL queries generation.
- *JSShexTest*[8], developed by Jesse van Dam is another Javascript implementation. It both supports the SHEXc and SHEX/RDF syntax of Shape Expressions and contains a validation semantics for testing purposes based on truth tables.
- Shexcala[9]: an implementation developed in Scala with an efficient implementation based on derivatives of regular expressions. It supports validation against an RDF file and against a SPARQL endpoint. In the following section we describe an online validation service which is implemented on top of ShExcala.
- Haws[10]: a Haskell implementation based on type inference semantics and backtracking. This implementation can be seen as an executable monadic semantics of Shape Expressions [10].

5. RDFSHAPE: AN RDF SHAPE VALIDATION SERVICE

RDFShape[11] is an online RDF Shape validation web service that can be used to validate both the syntax and the shape of RDF data against some schema.

The online service has five types of inputs for RDF:

- By URI: The RDF data to be validated is downloaded from a given URI
- By File: The data is uploaded from a local file
- By Input: The data is inserted in a textarea
- By Endpoint: The RDF data triples are retrieved from a SPARQL endpoint on demand. The user has to provide the URI of the endpoint.
- By dereference: The RDF triples are obtained by dereferencing the URIs of the resources that will be validated and using content negotiation to ask for RDF/XML or Turtle representations.

[7]http://www.w3.org/2013/ShEx/FancyShExDemo
[8]https://github.com/jessevdam/shextest
[9]http://labra.github.io/ShExcala/
[10]http://labra.github.io/haws/
[11]http://rdfshape.weso.es

The RDFShape tool allows the user to specify whether to use a Shape Expression schema or not. If not, the tool just checks that the RDF can be parsed. Otherwise, the user can also enter a Schema by URI, by File or by Input.

Finally, it is possible to validate a specific IRI or just any IRI in the RDF graph. Specifying an IRI is recommended when validating by Endpoint to check the shape of a given IRI in the endpoint and it is mandatory when using by dereference, as it will be the IRI that will be dereferenced to validate its representation.

Figure 2 contains a screen capture of the RDFShape validation tool.

6. VALIDATING LINKED DATA PORTALS USING SHAPE EXPRESSIONS

RDF Shape Expressions can be used not only to describe the contents of linked data portals, but also to validate them.

We consider that one of the first steps in the development of a linked data portal should be the Shape Expression declarations of the different types of resources. Shape Expressions can play a similar role to Schema declarations in XML based developments. They can act as a contract for both the producers and consumers of linked data portals.

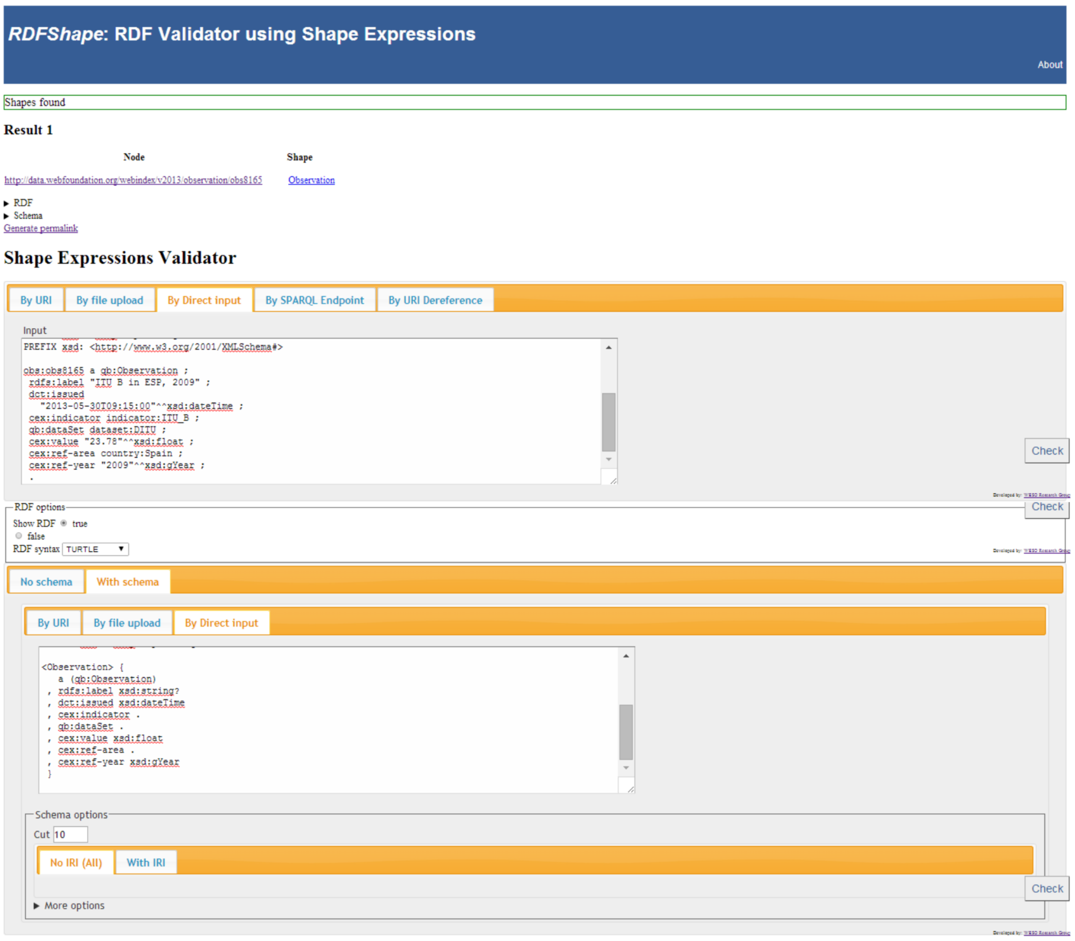
Notice, however, that this contract does not suppose an extra-limitation between the possible consumers a linked data portal can have. There is no impediment to have more than one shape expressions which enforce different constraints. As a naïve example, the declarations of the iso2 and iso3 code of Countries can be further constrained using regular expressions to indicate that they must be 2 or 3 alphabetical characters or could be more relaxed saying that it may be any value (not only strings). The advantage of Shape Expressions is that they offer a declarative and intuitive language to express and refer to those constraints.

Shape Expression declarations can also be employed to generate synthetic linked data in the development phase so one can perform stress tests. For example, during the development of the WebIndex data portal, we implemented the wiGenerator[12] tool which is a simple program that can generate random linked data that follows the WebIndex data model with any number of indicators, years of countries specified by the user. These fake RDF datasets can be employed to perform stress and usability tests of the data visualization software.

Shexcala offers the possibility to validate a URI in an endpoint or by dereferencing it (retrieving the RDF data behind that URI). The implementation performs a generic SPARQL query to obtain all the triples that have a given node as subject in the endpoint:

```
construct { $node ?p ?y } where {
 $node ?p ?y .
}
```

[12]http://labra.github.io/wiGenerator/


RDFShape: RDF Validator using Shape Expressions

Shapes found

Result 1

Node	Shape
http://data.webfoundation.org/webindex/v2013/observation/obs8165	Observation

▶ RDF
▶ Schema
Generate permalink

Shape Expressions Validator

Input

```
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>

obs:obs8165 a qb:Observation ;
 rdfs:label "ITU B in ESP, 2009" ;
 dct:issued
   "2013-05-30T09:15:00"^^xsd:dateTime ;
 cex:indicator indicator:ITU_B ;
 qb:dataSet dataset:DITU ;
 cex:value "23.78"^^xsd:float ;
 cex:ref-area country:Spain ;
 cex:ref-year "2009"^^xsd:gYear ;
 .
```

RDF options
Show RDF [✓ true] false
RDF syntax TURTLE

```
<Observation> {
   a (qb:Observation)
 , rdfs:label xsd:string?
 , dct:issued xsd:dateTime
 , cex:indicator .
 , qb:dataSet .
 , cex:value xsd:float
 , cex:ref-area .
 , cex:ref-year xsd:gYear
 }
```

Schema options
Cut 10

Figure 2: Screen capture of RDFShape tool

Once the triples are retrieved, the system validates the Shape Expressions declarations of that graph to check the shape of that node.

In this way, it is very easy to perform shape checking on the contents of linked data portals. For example, one can retrieve all the nodes of type `qb:Observation` and check that they have the shape `<Observation>`.

Notice that in general, this kind of validation is context sensitive to a given data portal. Shape Expressions deliberately separates types from shapes.

For example, LandPortal also expresses a shape for its use of `qb:Observation`. Both WebIndex and LandPortal respect the RDF data Cube definition of an Observation, but they can require or prohibit different properties (from that ontology or elsewhere) on those Observations. The observations in WebIndex have different shapes than the observations in LandPortal, but all of them have type `qb:Observation` without introducing any logical conflicts. This reflects a different usage pattern than generally seen for OWL or SPIN constraints (see 8). We consider that this difference between structural shapes and semantic types of resources improves the separation of concerns involved in linked data portal development.

Nevertheless, although some shapes can be specific to some linked data portals, nothing precludes to define templates and libraries of generic shapes that can be reused between different data portals.

7. EXTENSIONS AND CHALLENGES

At the moment of this writing, the W3C has just chartered a RDF Data Shapes Working Group with the mission to *produce a language for defining structural constraints on RDF graphs*. The Shape Expressions language is being used as part of the working group discussions so it is possible that some parts of the language will change in the future.

There are currently several topics and extension proposals for the Shape Expression language that may be interesting to mention:

- The Shape Expression language contains other common regular expression operators like alternatives (`|`), negations (`!`), groupings using parenthesis, etc. that can express more complex patterns. For example, we could declare that Countries can have either `wf:iso2` or `wf:iso3`, and that they must not have the property `dc:creator` as:

```
<Country> { a (wf:Country)
, rdfs:label xsd:string
, ( wf:iso2 xsd:string
  | wf:iso3 xsd:string
  )
, ! dc:creator .
}
```

- Open vs Closed shapes. An open shape is a shape expression that validates nodes that contain the triples specified in the shape but can also contain other triples. A closed shape only validates nodes with those triples and no more. For example, if we declare users shapes as:

```
<User> { a foaf:Person }
```

  and we have the following triples:

```
:john a foaf:Person,
  foaf:name "John" .
```

  Using closed shapes, the system would not assign `:john` the shape `<User>` because it contains an extra triple, while using open shapes it would assign it that shape.

  It is perceived that open shapes fit better in an Open World web, while closed shapes would be better for more controlled environments.

  In this line of work, there is also a proposal to reuse shape descriptions by including other shape declarations. For example, one may be interested to say that providers have the shape `<Organization>` but also contain the property `wf:sourceURI` as:

```
<Provider> & <Organization>
  { wf:sourceURI IRI }
```

- Incoming edges, relations and named graphs. The current Shape Expression language is based on describing the subjects of an RDF graph. It would be possible to extend the language to handle also objects and properties. For example, we can declare reverse arcs using the operator ^ to indicate incoming arcs. The Country declaration could be:

```
<Country> { a (wf:Country)
, rdfs:label xsd:string
, wf:iso2 xsd:string
, wf:iso3 xsd:string
, ^ cex:ref-area @<Observation> *
}
```

  with the meaning that a country can receive (zero or more) arcs with property `cex:ref-area` of shape `<Observation>`.

  In the same way, it may be interesting to declare the shape of RDF nodes that act as properties. Another extension proposal is to describe named graphs. These two proposals are not difficult to add, but it is not clear which syntax would be intuitive enough for them.

- More expressiveness. The Shape expression language can be extended with semantic actions to increase the expressiveness of the language. Semantic actions are marked by `%lang { actions %}` which means that the validator can invoke a processor of the language `lang` with the corresponding actions.

  The Javascript implementation supports semantic actions in Javascript and SPARQL which can add more expressiveness to the validation declarations. In fact, it also contains two simple languages (GenX and GenJ) which enable an easy way to transform RDF to both XML and JSon.

  Following the RelaxNG path, the Shape Expression language can be seen as a simple Domain Specific Language which is tailored to express the structure of RDF graphs. It is not intended to perform strong constraint checking or validation using computations. However, with semantic actions or shape expression validators embedded in other tool chains that possibility could be offered.

  In the same way, the interplay between Shape Expressions and reasoners is not established. Some applications could do inference between checking the shape of RDF graphs, while other applications could check the shapes before invoking a reasoner. Another possibility that could be explored is to have some built-in way that could invoke reasoning capabilities.

One of the challenges of the Shape Expressions language is the performance of Shape checking. A naïve implementation of Shape checking using backtracking can lead to exponential growth. We have found that using regular expression derivatives offers an efficient implementation and we are currently evaluating its performance.

8. RELATED WORK

Improving the quality of linked data has been of increasing interest in the last years. Sieve[11] proposed a framework for expressing quality assessment methods as well as fusion methods. RDFUnit[8] is a test-driven framework that can run test cases against an endpoint. In the case of RDF validation, the main approaches can be summarized as:

- Inference based approaches, which try to adapt RDF Schema or OWL to express validation semantics. The use of Open World and Non-unique name assumption limits the validation possibilities. In fact, what triggers constraint violations in closed world systems leads to new inferences in standard OWL systems. [4, 18, 12] propose the use of OWL expressions with a Closed World Assumption to express integrity constraints.

- SPARQL Inferencing Notation (SPIN)[7] constraints associate RDF types or nodes with validation rules. These rules are expressed as SPARQL ASK queries where `true` indicates an error or CONSTRUCT queries which produce `spin:ConstraintViolations`. SPIN constraints use the expressiveness of SPARQL plus the semantics of the `?this` variable standing for the current subject and the `spin:ConstraintViolation` class.

- SPARQL-based approaches use the SPARQL Query Language to express the validation constraints. SPARQL has much more expressiveness than Shape Expressions and can even be used to validate numerical and statistical computations [9]. However, we consider that the Shape Expressions language will be more usable by people familiar with validation languages like RelaxNG. Nevertheless, Shape Expressions can be translated to SPARQL queries. In fact, we have implemented a translator from Shape Expressions to SPARQL queries. This translator combined with semantic actions expressed in SPARQL can offer the same expressiveness as other SPARQL approaches with a more succinct and intuitive syntax.

There have been other proposals using SPARQL combined with other technologies. Fürber and Hepp[6] proposed a combination between SPARQL and SPIN as a semantic data quality framework, Simister and Brickley[17] propose a combination between SPARQL queries and property paths which is used in Google and Kontokostas et al [8] proposed *RDFUnit* a Test-driven framework which employs SPARQL query templates that are instantiated into concrete quality test queries. We consider that Shape Expressions can also be employed in the same scenarios as SPARQL while the specialized validation nature of Shape Expressions can lead to more efficient implementations.

- Grammar based approaches define a domain specific language to declare the validation rules. OSLC Resource Shapes [16] have been proposed as a high level and declarative description of the expected contents of an RDF graph expressing constraints on RDF terms. Shape Expressions have been inspired by OSLC although they offer more expressive power.

  Dublin Core Application Profiles [5] also define a set of validation constraints using Description Templates with less expressiveness than Shape Expressions.

The main inspiration for Shape Expressions has been RelaxNG [19], a Schema language for XML that offers a good trade-off between expressiveness and validation efficiency. The semantics of RelaxNG has also been expressed using inference rules in the specification document [14] and is based on tree grammars [13]. In the case of Shape Expressions the underlying semantics can be defined in terms of regular bag expressions [3].

Shape Expressions are also being employed in the development of more specialized validators. For example, the Vaskos project[13] is developing a SKOS validator using a combination between Shape Expressions and SPARQL queries.

9. CONCLUSIONS

Shape Expressions have been proposed as a Domain Specific Language that can describe and automatically validate RDF. They offer a more expressive way to define sets of RDF graph shapes than OSLC's Resource Shapes or Dublin Core's Application Profiles. There are trade-offs between expressiveness and implementability, but compared to schema languages in other data models, Shape Expressions represent a conservative point in that spectrum, emulating mostly the expressiveness of RelaxNG.

From a tooling perspective, shape expressions can be used standalone to validate RDF graphs and endpoints offering a dedicated language that can be implemented efficiently and generate specialized error messages for the concrete task of shape validation.

Given that Shape Expressions can be translated to SPARQL queries, they can also be combined with other widely deployed infrastructure.

[13]`http://vaskos.chemaar.cloudbees.net/`

The complexity of the validation algorithms for Shape Expressions offers some theoretical challenges related to regular bag expressions that have been tackled in [3]. The last implementation of Shexcala contained an algorithm based on derivatives of regular expressions which greatly improved the efficiency of the validation process.

Although the language is new and the syntax can seem strange at first sight, we noticed that people are able to learn the syntax and to declare shape expressions quickly.

In general we consider that the benefits of validation can help the adoption of RDF based solutions where the quality of data is an important issue.

10. REFERENCES

[1] T. Berners-Lee. Linked-data design issues. W3C design issue document, June 2006. http://www.w3.org/DesignIssue/LinkedData.html.

[2] C. Bizer, T. Heath, and T. Berners-Lee. Linked data - the story so far. *International Journal Semantic Web Information Systems*, 5(3):1–22, 2009.

[3] I. Boneva, J. E. Labra, S. Hym, E. G. Prud'hommeau, H. Solbrig, and S. Staworko. Validating RDF with Shape Expressions. *ArXiv e-prints*, Apr. 2014.

[4] K. Clark and E. Sirin. On RDF validation, stardog ICV, and assorted remarks. In *RDF Validation Workshop. Practical Assurances for Quality RDF Data*, Cambridge, Ma, Boston, September 2013. W3c, `http://www.w3.org/2012/12/rdf-val`.

[5] K. Coyle and T. Baker. Dublin core application profiles. separating validation from semantics. In *RDF Validation Workshop. Practical Assurances for Quality RDF Data*, Cambridge, Ma, Boston, September 2013. W3c, `http://www.w3.org/2012/12/rdf-val`.

[6] C. Fürber and M. Hepp. Using sparql and spin for data quality management on the semantic web. In W. Abramowicz and R. Tolksdorf, editors, *Business Information Systems*, volume 47 of *Lecture Notes in Business Information Processing*, pages 35–46. Springer, 2010.

[7] H. Knublauch. SPIN - Modeling Vocabulary. `http://www.w3.org/Submission/spin-modeling/`, 2011.

[8] D. Kontokostas, P. Westphal, S. Auer, S. Hellmann, J. Lehmann, R. Cornelissen, and A. 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.

[9] J. E. Labra and J. M. Alvarez Rodríguez. Validating statistical index data represented in RDF using SPARQL queries. In *RDF Validation Workshop. Practical Assurances for Quality RDF Data*, Cambridge, Ma, Boston, September 2013. W3c, `http://www.w3.org/2012/12/rdf-val`.

[10] J. E. Labra Gayo. Reusable semantic specifications of programming languages. In *6th Brazilian Symposium on Programming Languages*, 2002.

[11] P. N. Mendes, H. Mühleisen, and C. Bizer. Sieve: Linked Data Quality Assessment and Fusion. In *2nd International Workshop on Linked Web Data Management (LWDM 2012)*

at the 15th International Conference on Extending Database Technology, EDBT 2012, March 2012.

[12] B. Motik, I. Horrocks, and U. Sattler. Adding Integrity Constraints to OWL. In C. Golbreich, A. Kalyanpur, and B. Parsia, editors, *OWL: Experiences and Directions 2007 (OWLED 2007)*, Innsbruck, Austria, June 6–7 2007.

[13] M. Murata, D. Lee, M. Mani, and K. Kawaguchi. Taxonomy of xml schema languages using formal language theory. *ACM Trans. Internet Technol.*, 5(4):660–704, Nov. 2005.

[14] OASIS Committee Specification. RELAX NG Specification:. http://relaxng.org/spec-20011203.html, 2001.

[15] D. Reynolds. The Organization Ontology. `http://www.w3.org/TR/vocab-org/`, 2014.

[16] A. G. Ryman, A. L. Hors, and S. Speicher. OSLC resource shape: A language for defining constraints on linked data. In C. Bizer, T. Heath, T. Berners-Lee, M. Hausenblas, and S. Auer, editors, *Linked data on the Web*, volume 996 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2013.

[17] S. Simister and D. Brickley. Simple application-specific constraints for rdf models. In *RDF Validation Workshop. Practical Assurances for Quality RDF Data*, Cambridge, Ma, Boston, September 2013. W3c, `http://www.w3.org/2012/12/rdf-val`.

[18] J. Tao, E. Sirin, J. Bao, and D. L. McGuinness. Integrity constraints in OWL. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI-10)*. AAAI, 2010.

[19] E. van der Vlist. *Relax NG: A Simpler Schema Language for XML*. O'Reilly, Beijing, 2004.