RDF Validation: A Brief Survey

Dominik Tomaszuk(✉)

Institute of Informatics, University of Bialystok,
ul. Konstantego Ciołkowskiego 1M, 15-245 Białystok, Poland
d.tomaszuk@uwb.edu.pl

Abstract. The last few years have brought a lot of changes in the RDF validation and integrity constraints in the Semantic Web environment, offering more and more options. This paper analyses the current state of knowledge on RDF validation and proposes requirementsL for RDF validation languages. It overviews and compares the previous approaches and development directions in RDF validation. It also points at the pros and cons of particular implementation scenarios.

Keywords: Validation · Semantic Web · RDF · Integrity constraints

1 Introduction and Motivation

Schemes pay a key role in databases. They organize data and determine the way in which the database is constructed and what integrity constraints it is affected by. Databases allow for checking the conformity of their instances with the given scheme. Schemes in the RDF context describe integrity constraints imposed by the application on documents and/or RDF graph stores. Integrity constraints may account for the multitude of relations, restrictions in the allowed property values, the presence of properties, certain types of data values or default values. Current solutions [6,17,18,20,26] are trying to answer the needs of defining the graph's structure for validation and the way of matching the data with the description. The main purpose of this paper is to support discussions of those solutions and to help working groups in the development of suitable approaches.

In this paper, we perform an overview and comparison of current options for RDF validation. Section 2 describes the basic notions of RDF. Section 3 presents requirements for RDF validation languages. In Sect. 4 we perform an overview of solutions and we compare their characteristics and expressiveness. This section is also devoted to related work. Section 5 presents experiments performed to evaluate presented approaches. The last section is a conclusion.

2 RDF Background

An RDF constitutes a universal method of the description and information modeling accessible in Web resources. RDF is a very common data model for

S. Kozielski et al. (Eds.): BDAS 2017, CCIS 716, pp. 344–355, 2017.
DOI: 10.1007/978-3-319-58274-0_28

resources and relationship description between them. In other words, tt provides the crucial foundation and infrastructure to support the management and description of data.

An assumption in RDF [12] is to describe resources by means of the statement consisting of three elements (the so-called RDF *triple*): *subject*, *predicate* and *object*. RDF borrows often from natural languages. An RDF *triple* may then be seen as an expression with *subject* corresponding to the subject of a sentence, *predicate* corresponding to its verb and *object* corresponding to its object. The RDF data model depends on the concept of creating web-resource expressions in the form of subject-predicate-object statements, which in the RDF terminology, are referred to as *triples*.

An RDF triple comprises a subject, a predicate, and an object. In [12], the meaning of subject, predicate and object is explained. The *subject* denotes a resource, the *object* fills the value of the relation, the *predicate* refers to the aspects or features of resource and expresses a subject – object relationship. In other words, the predicate (also known as a property) denotes a binary relation.

The elemental constituents of the RDF data model are RDF terms that can be used in reference to resources: anything with identity. The set of RDF terms is divided into three disjoint subsets:

- IRIs,
- literals,
- blank nodes.

Definition 1 (IRIs). IRIs *serve as global identifiers that can be used to identify any resource. For example,*

Example 1 (IRIs). `<http://dbpedia.org/page/Car>` is used to identify the car in DBpedia [1].

Definition 2 (Literals). Literals *are a set of lexical values.*

Example 2 (Literals). Literals comprise a lexical string and a datatype, such as `"1"^^http://www.w3.org/2001/XMLSchema#int`. Datatypes are identified by IRIs, where RDF borrows many of the datatypes defined in XML Schema 1.1 [29].

Definition 3 (Blank nodes). Blank nodes *are defined as existential variables used to denote the existence of some resource for which an IRI or literal is not given. They are inconstant for blank nodes and are in all cases locally scoped to the RDF space.*

RDF triple is composed of the above terms. Following [12], we provide definitions of RDF triples below.

Definition 4 (RDF triple). *Assume that $\mathcal{I}$ is the set of all Internationalized Resource Identifier (IRI) references, $\mathcal{B}$ (an infinite) set of blank nodes, $\mathcal{L}$ the set of literals. An* RDF triple *is defined as a triple $t = \langle s, p, o \rangle$ where $s \in \mathcal{I} \cup \mathcal{B}$ is called the* subject, *$p \in \mathcal{I}$ is called the* predicate *and $o \in \mathcal{I} \cup \mathcal{B} \cup \mathcal{L}$ is called the* object.

Example 3 (RDF triple). The example presents an RDF triple consisting of subject, predicate and object.

```
<http://example.net/spec#d> rdfs:label "specification".
```

A set of RDF triples intrinsically represents a labeled directed multigraph. The nodes are the subjects and objects of their triples. RDF is often referred to as being *graph structured data* where each $\langle s, p, o \rangle$ triple can be interpreted as an edge $s \xrightarrow{p} o$.

Definition 5 (RDF graph). *Let* $\mathcal{O} = \mathcal{I} \cup \mathcal{B} \cup \mathcal{L}$ *and* $\mathcal{S} = \mathcal{I} \cup \mathcal{B}$. $G \subset \mathcal{S} \times \mathcal{I} \times \mathcal{O}$ *is a finite subset of* RDF triples *and called an* RDF graph.

Example 4 (RDF graph). The example presents an RDF graph of a FOAF [8] profile in Turtle [2] syntax. This graph includes two RDF triples:

```
<#d>    rdf:type    foaf:Document.
<#d>    rdfs:label  "specification".
```

3 RDF Validation Language Requirements

In this section, we propose requirements that an RDF validation language should fulfil. From presented approaches [6,17,18,20,26] and our experience in using RDF validation language we have composed a list of key requirements:

1. representatable in RDF and concise language,
2. expressive power,
3. shortcuts to recurring patterns,
4. self-describability,
5. provide a standard semantics.

3.1 Representatable in RDF

Any non-RDF syntax will lose a key advantage of a triple-based notation. RDF is well-implemented so RDF validation language shall use RDF as its syntax. On the other hand, RDF constraints should be specifiable in a compact form. Table 1 presents key approaches and their syntaxes. A language shall be designed to easily integrate into deployed systems that use RDF (i.e. graph stores), and provides a smooth upgrade. The use of RDF in a validation language makes constraints accessible to developers without the obligation to install additional parsers, software libraries or other programs. This requirement is satisfied by all RDF validation languages, but an RDF syntax is the primary format for SHACL, ReSh and SPIN.

Table 1. Summary of RDF validation approaches

Name	Syntax	Related work
ShEx	Compact, RDF, JSON	[4,20,24]
SHACL	RDF	[17]
ReSh	RDF	[25,26]
DSP	XML, RDF[a]	[6,11]
SPIN	RDF, SPARQL	[14,18]
OWL, RDFS	RDF (Turtle), XML, Manchester, functional	[7,21,22]

[a]The main syntax is XML.

3.2 Expressive Power

An RDF validation language should express:

- RDF terms restrictions:
  - value restrictions,
  - allowed values and not allowed values,
  - default values,
  - literal values,
  - datatypes comparison,
  - IRI pattern matching,
- cardinality constraints:
  - minimum cardinality,
  - maximum cardinality,
  - exact cardinality,
- predicate constraints:
  - class-specific property range,
  - OR operator (including groups),
  - required predicates,
  - optional predicates,
  - repeatable predicates,
  - negative predicate constraints.

Table 2 presents features of the below-mentioned approaches, namely: RDF terms constraints, cardinality and predicates constraints compared to RDFS and OWL. Note that OWL and RDFS are developed for inference and do not provide the features strictly for validation. This requirement is not satisfied by any languages, but SHACL supports most of the features.

3.3 Shortcuts to Recurring Patterns

Another important requirement is a macro supporting. An RDF validation language shall enable the definition of shortcuts to recurring patterns that improve overall readability and maintainability. Macros also can separate various parts of schema and enable users define rich constraints. It should provide a way to define high level reusable components in SPARQL [15], JavaScript or other languages. This requirement is satisfied by ShEx, SHACL and SPIN.

Table 2. Validation features

	ShEx	SHACL	ReSh	DSP	SPIN	RDFS, OWL
RDF terms						
Value restrictions	☑	☑	☑	☑	☑	☑
Allowed values	☑	☑	☑	☑	☑	☑
Not allowed values	☑	☑	☒	☒	☑	☑
Default values	☒	☑	☑	☒	☑	☒
Literal values	☑	☑	☒	☒	☑	☒
Datatypes comparison	☑	☑	☑	☑	☑	☑
IRI pattern matching	☑	☑	☒	☒	☑	☒
Cardinality						
Min. cardinality	☑	☑	□	☑	☑	☑
Max. cardinality	☑	☑	□	☑	☑	☑
Exact cardinality	☑	☑	□	☑	☑	☑
Predicates						
Class-specyfic property range	☑	☑	☑	☑	☑	☑
OR operator	☑	☑	☒	☒	☑	☑
OR operator (groups)	☑	□	☑	☒	☑	□
Required predicates	☑	☑	☑	☑	☑	☑
Optional predicates	☑	☑	☑	☑	☑	☑
Repeatable predicates	☑	☑	☑	☑	☑	☑
Negative predicate constraints	☑	☑	☒	☒	☑	☑

All features are marked as ☑ (yes), □ (partial) or ☒ (no).

3.4 Self-describability

An RDF validation language shall represent the schema specification using the schema itself being defined. This makes creating meta scheme easier than in a language without this property, since constraints can be treated as data. It means that examining the schema's entities depends on a homogeneous structure, and it does not have to handle several different structures that would appear in a complex syntax. Moreover, this meta schema can valuable in bootstrapping the implementation of the constraint language. This requirement is satisfied by all RDF validation languages.

3.5 Standard Semantics

An RDF validation language shall provide a standard semantics for describing RDF constraints. It should be defined in a commonly agreed-upon formal specification, which describe a model-theoretic semantics for RDF constraints, providing an exact formal specification of when truth is preserved by various validation operations. This requirement is satisfied only by SHACL.

4 RDF Validation

In this section we discuss main approaches, which are strictly designed for validation (Sect. 4.1) and other approaches, which can be used for some scenarios (Sect. 4.2).

4.1 Main Approaches

Shape Expressions. The first approach is Shape Expressions [4,20,24], which is a language for describing constraints on RDF triples. In addition to validating RDF, ShEx also allows to inform expected graph patterns for interfaces and create graphical user interface forms. It is a domain specific language used to define shapes, which describe conditions that handle a given node. It can be transformed into SPARQL queries. The most common syntax of ShEx compact, which is similar to RELAX NG Compact [9].

Example 5 (ShEx). The example presents a Shape Expressions schema in compact syntax according to Example 3.

```
<Shape1> {
  (rdfs:label xsd:string+)
}
```

Shapes Constraint Language. The next approach is the Shapes Constraint Language [17], which is a language constraining the contents of graphs. SHACL, similarly to ShEx, organises these constraints into shapes, which provide a high-level vocabulary to distinguish RDF predicates and their constraints. Moreover, constraints can be linked with shapes using SPARQL queries and JavaScript. In addition to validating RDF, it also allows to describe information about data structures, generate RDF data, and build GUIs.

Example 6 (SHACL). The example presents a Shapes Constraint Language schema in Turtle according to Example 3.

```
<Shape1> a sh:Shape;
  sh:property [
    sh:predicate rdfs:label;
    sh:datatype xsd:string;
    sh:minCount 1.
  ].
```

Resource Shapes. Another approach is Resource Shapes [25,26] which is a vocabulary for specifying the RDF shapes. ReSh authors assume that RDF terms come from many vocabularies. The ReSh shape is a description of the RDF graphs to integrity constraints those data are required to satisfy.

Example 7 (ReSh). The example presents a Resource Shapes schema in Turtle according to Example 3.

```
<Shape1> a oslc:ResourceShape;
  oslc:property [
    a oslc:Property;
    oslc:propertyDefinition rdfs:label;
    oslc:valueType xsd:string;
    oslc:occurs oslc:One-or-many
  ].
```

Description Set Profiles and SPARQL Inferencing Notation. ShEx, SHACL and ReSh are based on declarative and a high-level descriptions of RDF graph contents. These three approaches have similar features. Description Set Profiles [6,11] and SPARQL Inferencing Notation [18] are different approaches. The first is used to define structural constraints on data in a Dublin Core Application Profile. That approach allows to declare the metadata record contents in terms of validatable constraints. In addition to validating RDF, it can be used as configuration for databases and configuration for metadata editing software.

Example 8 (DSP). The example presents a Description Set Profiles schema in XML according to Example 3.

```
<dsp:StatementTemplate minOccurs="1"
 maxOccurs="infinity" type="literal">
  <dsp:Property>
http://www.w3.org/2000/01/rdf-schema#label
  </Property>
</dsp:StatementTemplate>
```

The latter one is a constraint and SPARQL-based rule language for RDF. It can link class with queries to capture constraints and rules which describe the behavior of those classes. SPIN is also a method to represent queries as templates. It can represent SPARQL statement as RDF triples. That proposal allows to declare new SPARQL functions.

Example 9 (SPIN). The example presents a SPARQL Inferencing Notation schema in Turtle according to Example 3.

```
<#c> spin:constraint [
  a spl:Attribute;
  spl:predicate rdfs:label;
  spl:minCount 1;
  spl:valueType xsd:string.
].
```

4.2 Other Approaches

In addition to the approaches presented in Sect. 4.1 there are OWL/RDFS [7,22] based approaches [10,21,23,28,30] and SPARQL [15] based approaches [13,19, 27]. The first group uses of RDF and OWL expressions with a CWA (Closed World Assumption) to express integrity constraints. The main drawback of those approaches is that inference engines cannot be used for checking the constraints.

Kontokostas *et al.* [19] focus on the data quality test patterns and data quality integrity constraints, which are represented in SPARQL query templates. Fischer *et al.* present RDD, language for expressing RDF constraints, which can be easily transformed into SPARQL queries. In the [27] authors propose approach based on SPARQL property paths. This proposal is similar to Schematron [16] in XML documents. The main drawback of those approaches is the need to know SPARQL. Another approach to the requirement is presented in [5].

5 Evaluation

In this Section we evaluate the most promising constraint languages. We evaluate all shape approaches (ShEx, SHACL and ReSh) that have community support and advanced features (see Table 2). In this Section we analyze whether languages are concise and how fast we can process them.

All experiments have been executed on a Intel Core i7-4770K CPU @ 3.50GHz (4 cores, 8 thread), 8GB of RAM, and a HDD with reading speed rated at ∼160 MB/s[1]. We have been used Linux Mint 17.3 Rosa (kernel version 3.13.0) and Python 3.4.3 with RDFLib 4.2.1 and Virtuoso Open-Source Edition 7.2.4.

We prepare constraint rules for RDF data that was generated by Berlin SPARQL Benchmark (BSBM) [3]. According to [3], the BSBM benchmark is settled in an e-commerce scenarios in which a set of products (denotated P) is offered by different vendors and consumers who have posted reviews on various sites. The benchmark defines an abstract data model for this scenarios together with data production rules that allow benchmark datasets to be scaled to arbitrary sizes using the number of products as a scale factor. We enrich BSBM to fake datatypes (denotated E). Our implementation is available at https://github.com/domel/bsbm_validation. The data description which was used in the experiment, is presented in Table 3.

All RDF validation languages were created in RDF syntax. We choose N-Triples[2] because this serialization is normalized. In ShEx, SHACL and ReSh we declared appropriate datatypes for predicates such as: `foaf:name`, `dc:publisher`, `dc:title`, `bsbm:price`, `bsbm:validFrom`, `bsbm:validTo` and `bsbm:deliveryDays`. Incorrect values refer to `dc:date` predicate. We choose that scenario, because al validation languages support tested features, such as value and datatype restrictions. We also prepare constraint rules for all BSBM datasets.

[1] We test it in `hdparm -t`.

[2] https://www.w3.org/TR/n-triples/#canonical-ntriples.

Table 3. Datasets description

$\|P\|$	$\|G\|$	$\|E\|$	$\|P\|$	$\|G\|$	$\|E\|$
10	4987	614	60	26143	2913
20	8458	928	70	29707	3230
30	11962	1244	80	33194	3544
40	16901	1845	90	36699	3859
50	22629	2598	100	40177	4174

$|P|$ - cardinality of BSBM products.
$|G|$ - cardinality of RDF triples.
$|E|$ - cardinality of fake datatypes.

To test our schemes for different datasets, we built an RDF data validation system. The prototype implementation of the testbed has been fully implemented in the Python programming language with RDFLib. For storage, a Virtuoso Open-Source Edition 7.2.4 RDF graph store has been used. Figure 1 shows architecture of our testbed. Our data validation system checks datatypes on-the-fly before loading data.

Definition 6 (RDF data validation system). *An* RDF data validation system *is a tuple* $\langle \mathcal{D}, C, \Sigma \rangle$ *where* $\mathcal{D}$ *is the source data,* C *is the constraints and* Σ *is the state of the system.*

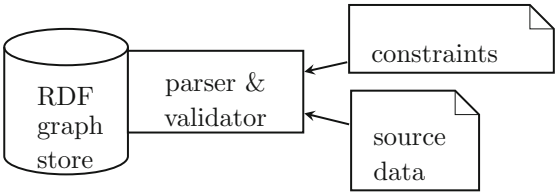

Fig. 1. Testbed architecture

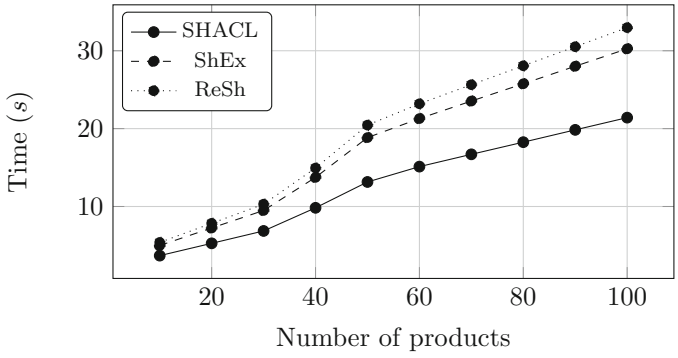

Fig. 2. Evaluation times of validation and loading valid data

Figure 2 shows that the number of predicates needed to describe validation rules has the greatest influence on the differences in the validation times. The fastest processing time belongs to SHACL and the slowest belongs to ReSh.

6 Conclusions

RDF validation is an important issue in the Semantic Web stack. There is no standard way to validate RDF data conforming to RDF constraints like DDL for relational databases or DTD for XML documents. Integrity constraints may be necessary in editing documents and actions performed on RDF graph stores. In this article, we conduct an overview of approaches to RDF validation and we show the differences in these approaches, drawing attention to the advantages and disadvantages of particular solutions.

We argue that RDF validation on the Semantic Web nowadays faces some of the challenges we were facing in the past, when databases were at their infancy. However, this area evolves very fast and attracts the attention of many researchers, the resulting in the vast scope of works we showed in this paper. We hope that this survey contributes to a better understanding of RDF validation. As part of future work, we will continuously analyze solutions within the RDF validation area.

Acknowledgment. We thank David Wood, co-chair of the RDF Working Group, for comments that greatly improved the paper.

A Used Prefixes

In Table 4 we enumerate the prefixes used throughout this paper to abbreviate IRIs.

Table 4. Used prefixes

Prefix	IRI
`rdf`	http://www.w3.org/1999/02/22-rdf-syntax-ns#
`rdfs`	http://www.w3.org/2000/01/rdf-schema#
`foaf`	http://xmlns.com/foaf/0.1/
`xsd`	http://www.w3.org/2001/XMLSchema#
`sh`	http://www.w3.org/ns/shacl#
`oslc`	http://open-services.net/ns/core#
`dsp`	http://purl.org/metainfo/terms/dsp#
`spin`	http://spinrdf.org/spin#
`spl`	http://spinrdf.org/spl#
`dc`	http://purl.org/dc/elements/1.1/
`bsbm`	http://www4.wiwiss.fu-berlin.de/bizer/bsbm/v01/vocabulary/

References

1. Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., Cyganiak, R., Ives, Z.: DBpedia: a nucleus for a web of open data. In: Aberer, K., et al. (eds.) ASWC/ISWC-2007. LNCS, vol. 4825, pp. 722–735. Springer, Heidelberg (2007). doi:10.1007/978-3-540-76298-0_52
2. Beckett, D., Berners-Lee, T., Prud'hommeaux, E., Carothers, G.: Turtle - Terse RDF Triple Language. W3C recommendation, World Wide Web Consortium. https://www.w3.org/TR/2014/REC-turtle-20140225/
3. Bizer, C., Schultz, A.: The Berlin SPARQL benchmark. Int. J. Semant. Web Inf. Syst. (IJSWIS) **5**(2), 1–24 (2009)
4. Boneva, I., Labra Gayo, J.E., Hym, S., Prud'hommeau, E.G., Solbrig, H.R., Staworko, S.: Validating RDF with shape expressions. CoRR abs/1404.1270 (2014)
5. Bosch, T., Eckert, K.: Requirements on RDF constraint formulation and validation. In: Proceedings of the 2014 International Conference on Dublin Core and Metadata Applications, pp. 95–108. Citeseer (2014)
6. Bosch, T., Eckert, K.: Towards description set profiles for RDF using SPARQL as intermediate language. In: Proceedings of the 2014 International Conference on Dublin Core and Metadata Applications (DCMI 2014), pp. 129–137. Dublin Core Metadata Initiative (2014)
7. Brickley, D., Guha, R.: RDF Schema 1.1. W3C recommendation, World Wide Web Consortium, February 2014. http://www.w3.org/TR/2014/REC-rdf-schema-20140225/
8. Brickley, D., Miller, L.: FOAF vocabulary specification 0.99. Technical report FOAF Project, January 2014. http://xmlns.com/foaf/spec/20140114.html
9. Clark, J.: RELAX NG compact syntax. Committee specification, The Organization for the Advancement of Structured Information Standards (2002). http://www.oasis-open.org/committees/relax-ng/compact-20021121.html
10. Clark, K., Sirin, E.: On RDF validation, stardog ICV, and assorted remarks. In: RDF Validation Workshop. Practical Assurances for Quality RDF Data, Cambridge, MA, Boston (2013)
11. Coyle, K., Baker, T.: Dublin core application profiles. Separating validation from semantics. In: RDF Validation Workshop. Practical Assurances for Quality RDF Data, Cambridge, MA, Boston (2013)
12. Cyganiak, R., Lanthaler, M., Wood, D.: RDF 1.1 concepts and abstract syntax. W3C recommendation, World Wide Web Consortium, February 2014. http://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/
13. Fischer, P.M., Lausen, G., Schätzle, A., Schmidt, M.: RDF constraint checking. In: EDBT/ICDT Workshops, pp. 205–212 (2015)
14. Fürber, C., Hepp, M.: Using SPARQL and SPIN for data quality management on the semantic web. In: Abramowicz, W., Tolksdorf, R. (eds.) BIS 2010. LNBIP, vol. 47, pp. 35–46. Springer, Heidelberg (2010). doi:10.1007/978-3-642-12814-1_4
15. Harris, S., Seaborne, A.: SPARQL 1.1 Query Language. W3C recommendation, World Wide Web Consortium. https://www.w3.org/TR/2013/REC-sparql11-query-20130321/
16. Jelliffe, R.: The schematron assertion language 1.5. Academia Sinica Computing Center (2000)
17. Knublauch, H.: Shapes constraint language (SHACL). W3C editor's draft, World Wide Web Consortium, September 2014. http://w3c.github.io/data-shapes/shacl/

18. Knublauch, H., Hendle, J.A., Idehen, K.: SPIN - overview and motivation. W3C member submission, World Wide Web Consortium, February 2011. http://www.w3.org/Submission/2011/SUBM-spin-overview-20110222/
19. Kontokostas, D., Westphal, P., Auer, S., Hellmann, S., Lehmann, J., Cornelissen, R., Zaveri, A.: Test-driven evaluation of linked data quality. In: Proceedings of the 23rd International Conference on World Wide Web, pp. 747–758. ACM (2014)
20. Labra Gayo, J., Prud'hommeaux, E., Solbrig, H., Rodríguez, J.: Validating and describing linked data portals using RDF shape expressions. In: Workshop on Linked Data Quality (2015)
21. Motik, B., Horrocks, I., Sattler, U.: Adding integrity constraints to OWL. In: Third International Workshop on OWL: Experiences and Directions 2007 (OWLED 2007), Innsbruck, Austria (2007)
22. Patel-Schneider, P., Hayes, P.: RDF 1.1 semantics. W3C recommendation, World Wide Web Consortium, February 2014. http://www.w3.org/TR/2014/REC-rdf11-mt-20140225/
23. Pérez-Urbina, H., Sirin, E., Clark, K.: Validating RDF with OWL integrity constraints (2012). http://docs.stardog.com/icv/icv-specification.html
24. Prud'hommeaux, E., Labra Gayo, J.E., Solbrig, H.: Shape expressions: an RDF validation and transformation language. In: Proceedings of the 10th International Conference on Semantic Systems (SEM 2014), NY, USA, pp. 32–40. ACM, New York (2014)
25. Ryman, A.: Resource shape 2.0. W3C member submission, World Wide Web Consortium, February 2014. http://www.w3.org/Submission/2014/SUBM-shapes-20140211/
26. Ryman, A.G., Hors, A.L., Speicher, S.: OSLC resource shape: a language for defining constraints on linked data. In: LDOW, CEUR Workshop Proceedings, vol. 996. CEUR-WS.org (2013)
27. Simister, S., Brickley, D.: Simple application-specific constraints for RDF models. In: RDF Validation Workshop. Practical Assurances for Quality RDF Data (2013)
28. Sirin, E., Tao, J.: Towards integrity constraints in OWL. In: OWLED, vol. 529 (2009)
29. Sperberg-McQueen, M., Thompson, H., Peterson, D., Malhotra, A., Biron, P.V., Gao, S.: W3C XML schema definition language (XSD) 1.1 Part 2: datatypes. W3C recommendation, World Wide Web Consortium, April 2012. http://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/
30. Tao, J., Sirin, E., Bao, J., McGuinness, D.L.: Integrity constraints in OWL. In: AAAI (2010)