Validata: An online tool for testing RDF data conformance

Jacob Baungard Hansen, Andrew Beveridge, Roisin Farmer, Leif Gehrmann, Alasdair J G Gray, Sunil Khutan, Tomas Robertson, and Johnny Val

Department of Computer Science, Heriot-Watt University, Edinburgh, UK

Abstract. Validata is an online web application for validating an RDF document against a set of constraints. This is useful for data exchange applications or ensuring conformance of an RDF dataset against a community agreed standard. Constraints are expressed as a Shape Expression (ShEx) schema. Validata extends the ShEx functionality to support multiple requirement levels. Validata can be repurposed for different deployments by providing it with a new ShEx schema. The Validata code is available from `https://github.com/HW-SWeL/Validata`.

Keywords: Validation, Conformance, Metadata, Dataset Descriptions

1 Introduction

Validation – checking the conformance of a dataset against a schema – has been an integral part of XML and relational data generation, use, and exchange. It provides guarantees that the data conforms to some structure which enables tools to consume and process that data. This is essential to support data exchange. However there is no standard for expressing constraints for RDF data.

There has been a growing interest in validating RDF data beyond checking consistency with OWL ontologies [1]. For example, it may be desirable to constrain that a title for a resource is given using the `dcterm:title` property and that only one title is supplied. The W3C RDF Data Shapes Working Group[1] are defining a recommendation for capturing and testing such validation requirements. One existing language for defining such constraints for RDF data is the Shape Expression language (ShEx) [2,3] which provides a means to declaratively and concisely describe the shape of the graphs that are permitted. However applications are required that will validate an RDF dataset against a schema declared in ShEx.

The contribution of this paper is to present the Validata tool: a web application that enables users to easily check the conformance of an RDF document against a ShEx schema description. Validata provides client-side, browser-based validation of RDF documents, although it is also possible to use it through an API so that it can be included as part of a data publishing pipeline. The validation provides support for multiple RDF serialisations as well as different

[1] `http://www.w3.org/2014/data-shapes/` accessed November 2015

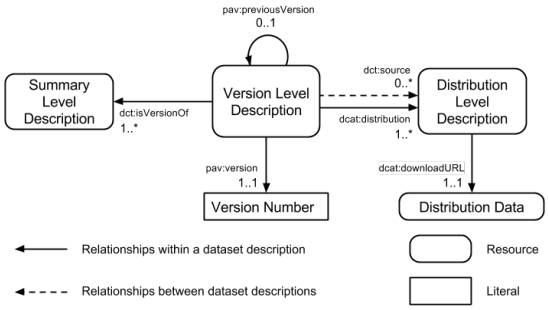

Fig. 1. The three levels of a dataset description.

requirement levels, i.e. stating whether a property is required, desirable or entirely optional. Validata was developed to check dataset descriptions (RDF documents) against the W3C Health Care and Life Sciences Community (HCLS) Profile for Dataset Descriptions [4]. Due to its use of ShEx, it is capable of being re-purposed for other metadata standards (e.g. DCAT [5]) or validation use cases that occur in data exchange scenarios.

2 Motivating Use Case

The W3C HCLS Community Profile [4] specifies a common format for dataset descriptions in the HCLS domain using RDF. The profile consists of three different types of description capturing different notions of a dataset (Figure 1): (i) the abstract idea of a dataset, (ii) specific versions of the dataset, and (iii) the file distributions of a specific version. That is, for a given dataset, say ChEMBL, there is an abstract idea of its existence that can be described providing properties that are unlikely to change over time (e.g. the name and description of the dataset); then there are the specific release versions of ChEMBL (20 at the time of writing) for which the properties specific to the release can be captured (e.g. version number and release date); and finally there are distribution files for the version, typically available in multiple file formats (e.g. database dump, RDF, or SD file), which can also be described.

For each description type, the metadata properties that need to be supplied are identified and provided with a requirement level drawn from [6], viz. MUST, SHOULD, or MAY. In total 61 metadata properties are identified in the HCLS Community Profile using terms drawn from 18 vocabularies. As such, it is difficult to verify when a dataset description conforms to the community profile. To encourage uptake, it was recognised that there would be a need to automatically validate the RDF descriptions of datasets against the HCLS Community Profile.

The HCLS Community Profile is not the only dataset description specification that exists. Others include BioDBCore [7] (a bioinformatics community initiative), Open PHACTS dataset descriptions [8] (a project specific profile),

and DCAT [5] (a cross-domain initiative). It is desirable that a validation tool is not made for a specific standard but could be employed for several of these standards through changing the configuration.

3 Requirements

We aimed to develop a tool that would meet the needs for validating RDF descriptions of datasets against different conformance schemas. The tool should be able to accept RDF in commonly used RDF serialisations, e.g. Turtle, TriG, and N-Triples, and the descriptions may be supplied by either copying and pasting into the web tool, or via file upload. The tool should also support common features of RDF such as string literals being provided with language tags.

It should be possible to specify different requirement levels such as those specified in [6]. The validator should consider the requirement level when returning conformance reports to the user.

The main motivating use case for our tool comes from the HCLS community profile. Thus the validation tool should be deployable on the W3C HCLS pages, i.e. it can be served from a standard web server without any server side code. The tool should support recent versions of the major browsers, i.e. Chrome, Firefox, Internet Explorer, and Safari. However, as validation is an important part of publishing data, it is desirable that the tool can also be used as part of a data publishing pipeline.

A typical interaction with the tool will be to refine a dataset description to ensure conformance with the chosen schema. To support this the tool should provide informative error and warning messages to the user. It should also be possible to edit the RDF data and easily recheck the validation.

4 Constraint Definitions

The Shape Expression Language (ShEx) was chosen for specifying the schema constraints on the RDF dataset descriptions [2,3], due to its concise notation (based on regular expressions), expressive coverage, and the existence of support libraries in Javascript. The concise notation enables a manual verification between the written ShEx schema for a conformance standard and the written documents which often include a checklist of properties, e.g. the table provided in Section 5 of [4] or the bullet lists in Section 4.1 of [8].

Figure 2 shows an excerpt of the ShEx schema for the HCLS community profile. The shape defines the properties and objects for a resource to match. In this case it defines a `<SummaryLevelShape>` that consists of four constraints. The shape must have a `rdf:type` declaration of `dctypes:Dataset`, and a title provided using `dct:title` with a language tagged string value. The shape may have an alternative title provided as a language tagged string using the `dct:alternative` property. The final constraint is that the `dct:created` property must not be used at all – the negation is stated with the '!' and the match

```
<SummaryLevelShape> {
    'MUST' rdf:type (dctypes:Dataset),
    'MUST' dct:title rdf:langString,
    'MAY'  dct:alternative rdf:langString+,
    'MUST' !dct:created .
}
```

Fig. 2. An excerpt of the HCLS Community Profile captured as a ShEx schema. Prefix definitions are omitted.

all with the '.'. Multiplicity of properties can be stated using standard regular expression syntax, e.g. '+' for one or more occurrences.

The full ShEx schema[2] contains a shape declaration for each of the description types: `SummaryLevelShape`, `VersionLevelShape`, `DistributionLevelShape`, and `RDFDistributionLevelShape`. Two distribution level shapes are defined since several of the properties only make sense when describing an RDF dataset, viz. the VoID properties [9].

5 Implementation

The Validata tool is a web application that is composed of two modules – a user interface over a standalone validator. Both the user interface and the validator are written in Javascript to enable the web application to be deployed on the W3C web server, and run entirely on the client side. The code for the Validata user interface is available from `https://github.com/HW-SWeL/Validata` while the code for the validation backend is available from `https://github.com/HW-SWeL/ShEx-validator`.

The user interface has been developed using various existing libraries. jQuery[3] is used for DOM manipulation with cross-browser functionality while Twitter Bootstrap[4] is used for cross-browser responsive pages. Browserify[5] was used to manage dependencies, and CodeMirror[6] was used to provide text editor panes which have line numbers. This enables informative error messages that reference the line in the original RDF document that was at fault.

The ShEx-Validator has been developed as a node.js module[7] which provides a standalone Javascript runtime environment. The core of the validator is the ShEx Javascript library[8]. The ShEx-Validator separates the tasks of validation

[2] `https://github.com/HW-SWeL/ShEx-validator/blob/043b396d9652deaf178628ba6f0b648464e4a8da/samples/HCLS%20Deployment/hcls-chembl17-example-description.ttl` accessed November 2015

[3] `https://jquery.com` accessed November 2015

[4] `http://getbootstrap.com/` accessed November 2015

[5] `http://browserify.org/` accessed November 2015

[6] `https://codemirror.net/` accessed November 2015

[7] `https://nodejs.org/` accessed November 2015

[8] Developed by Eric Prud'hommeaux `https://github.com/ericprud/ShExDemo/blob/96c7fb04827254af4618bdaeed49b3032fd002d3/RDF.js` accessed Nov 2015

and reporting errors; the latter being handled by the user interface. To enable support for different requirement levels, the ShEx language was extended to enable arbitrary tags prior to each path expression (shown in Figure 2). peg.js[9] was used to parse the extended grammar to generate the ShEx schema parser. For parsing RDF documents the N3.js library[10] was used which supports Turtle, TriG, N-Triples and N-Quads. Instead of using asynchronous callbacks in the code, the promises library[11] was used to enable an immediate response which would resolve to its actual value at some point in the future. Support for language tags has also been added to the ShEx validator. This means that multiplicity constraints are not violated if the property is given in multiple languages. A library of unit tests have been developed for the ShEx-Validator backend. These are run using the Mocha framework[12] and ensure that the new features added to the validator do not break existing features.

The separation of the UI and the ShEx-Validator means that the validator functionality can be deployed independently of the UI. For example, it could be incorporated into a data publishing pipeline meaning that dataset descriptions are validated as part of the data creation process. Alternatively it could be used as a library in other tools such as a dataset description editor tool [10]. Usage examples are available in the code documentation and in the Validata code itself.

6 Deployment

A screenshot of the W3C HCLS community profile deployment of the Validata tool is given in Figure 3. This is shown with the ChEMBL 2015 Demo deployed and the validation requirement level set to 'MAY' – meaning that the user would like error reports generated for properties declared at any of the conformance levels. Essentially the toggle enables the user to select the strictness of the validator which is desirable in different scenarios.

6.1 Information Panel

The left side of the UI is used to provide summary information to the user. At the top are the typical series of steps for using the tool, viz. (1) Select Schema, (2) Input Data, (3) Configure Options, and (4) Validation Results. These navigation aids guide the user through the main panel without breaking the information into multiple pages; thus allowing the user to scroll up and down and adjust things as they require. Immediately below these instructions is the option to load a preconfigured demo. In this case it corresponds to the example used in [4] that describes the ChEMBL dataset [11].

[9] `http://pegjs.org/` version 0.8.0; accessed November 2015
[10] Developed by Ruben Verborgh `https://github.com/RubenVerborgh/N3.js` version 0.4.1; accessed November 2015
[11] `https://www.promisejs.org/` version 6.1.0; accessed November 2015
[12] `http://mochajs.org` accessed November 2015

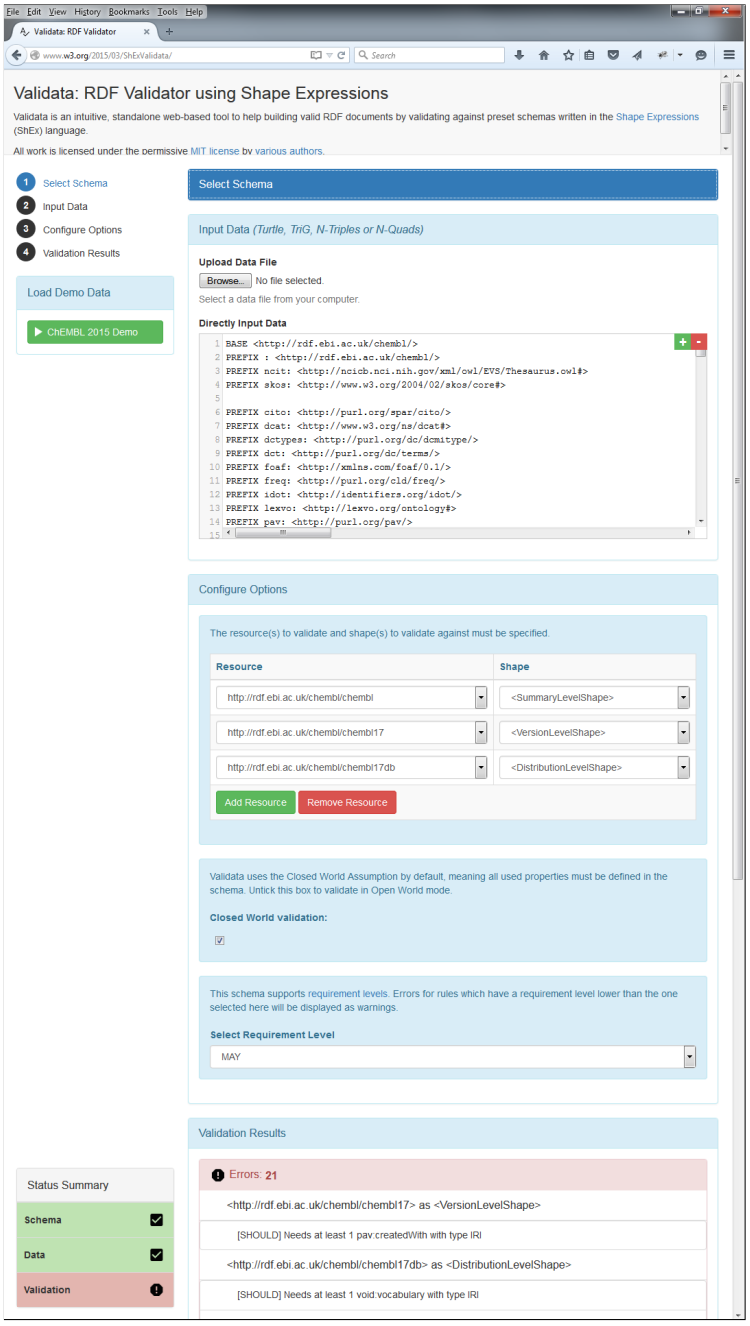

Fig. 3. Screenshot of the Validata tool deployed for the W3C Health Care and Life Sciences Community Profile `http://www.w3.org/2015/03/ShExValidata/`.

At the bottom of the left panel is a Status Summary box that indicates the validation progress. In this case it shows that the schema and the data are both syntactically valid but that the data does not conform to the constraints specified in the schema at the 'MAY' level.

6.2 Main Panel

The main part of the UI is used for providing the RDF data to validate, and displaying the results of the validation.

Select Schema Panel. The Select Schema panel (collapsed in the screenshot) is used for selecting a ShEx schema from a drop down menu; in the HCLS deployment there is only a single schema to choose from. The ShEx specification of the schema can be viewed by clicking on the Show Source button (not shown in the screenshot due to the collapsed panel). The motivation for the Select Schema panel is to allow users to validate their descriptions against multiple schemes in a single tool.

Input Data Panel. RDF data can be supplied for validation in one of several serialisations, with the validator automatically detecting the syntax. The user can either drag and drop an RDF file onto the panel, select the Browse button to upload a file, or copy and paste the RDF into the editor box. The RDF is parsed as it is entered to ensure that it is syntactically valid, resulting in the data row of the Status Summary turning green.

Configure Options Panel. The user may configure various options that are applied during validation. Initially the ShEx-Validator attempts to automatically identify the resources in the supplied RDF, and the shape that each resource should validate against. These are shown at the top of the Configure Options panel. These often need correcting, either resources are not identified or they are not being validated against the appropriate shape. The user can use the drop down menus to correct these. For example if a resource is identified but is being validated against the wrong shape then the correct shape can be selected. Additional resources from the supplied RDF can be added using the Add Resource button, and correspondingly removed using the Remove Resource button. The interaction between the user and the add and remove resource buttons needs additional work to improve the user experience as only the bottom resource can currently be removed. Instead the user should be able to directly select the rows that are to be removed.

The next configuration option allows the user to select whether to generate an error if additional properties are found, i.e. a resource matches a shape but has additional properties. Following the nomenclature of [3] the option is labelled Closed World Validation.

The final configuration option informs the ShEx-Validator at which requirement level the validation should take place. Any properties that are defined in

the ShEx schema with the selected requirement level and missing from the resource in the RDF will be reported as errors, those with a lower requirement level will be reported as warnings. This panel is only shown for schema that have requirement levels declared. The effect of choosing different requirement levels is that the UI will return validation results at the different levels as either errors or warnings. Users have found this desirable as a means of first developing a minimal description by ensuring that it matches the 'MUST' properties and then iteratively validating at the lower levels.

Validation Results Panel. The bottom panel of the tool provides the validation results. For each of the resources shown in the Configure Options panel, an entry is shown in the Validation Results panel. If the resource conforms to the shape description, then it is shown under the green Matches panel. This allows the user to see which parts of the RDF document match to a ShEx constraint. Clicking on an individual match will jump the focus of the UI up to the corresponding line of the supplied RDF data. If all the properties match then the validation row of the Status Summary will turn green.

Two severity levels of reports can be generated when properties are not present: (i) errors shown with red highlighting, and (ii) warnings shown with amber highlighting. The categorisation is controlled by the chosen requirement level. This is desirable as several properties defined at the SHOULD and MAY requirement level are seen as optional since they do not make sense for every situation. If ShEx optional properties were used, such reports would not be generated.

Additionally, error reports are generated for invalid syntax or when the incorrect object is supplied for a property, e.g. a string is provided instead of an IRI. Through the use of the codemirror library, when these errors are clicked-on the corresponding line in the RDF is highlighted in the editor pane allowing the user to correct the error. Due to the nature of parsing errors, these are not always accurate, but are generally in the correct region of the document.

6.3 Additional Deployments

Validata is a generic tool for validating RDF documents against a set of constraints captured as a ShEx schema. The tool can easily be deployed in different settings by editing the configuration file containing the ShEx schema. To demonstrate this flexibility, we have deployed Validata at two other locations for different dataset description schemas.

An instance capable of validating DCAT records [5] is available at `http://www.macs.hw.ac.uk/~ajg33/validata/`. This instance does not use requirement levels as these are not defined in the DCAT recommendation. Nor does the DCAT specification specify which properties are optional or mandatory, as such all properties have been specified as optional. The demonstration example uses the example in the DCAT specification. This example does not use closed world validation as it includes `rdfs:label` which is not specified anywhere else in the DCAT recommendation.

We have also deployed a version of Validata for validating dataset descriptions against the Open PHACTS specification [8]. This instance is available from `http://openphacts.cs.man.ac.uk/validata/` and uses a description for the DrugBank dataset for its example. Again requirement levels are used.

7 Related Work

Various approaches for describing constraints on an RDF document have been considered [1] and some tools have been developed for enabling users to validate their RDF documents against these constraints.

The Fancy ShEx Demo developed by Eric Prud'hommeaux[13] is a proof of concept system to showcase the capabilities of the ShEx language. It provides a Javascript implementation for validating RDF against a ShEx schema and was chosen as the basis for our own implementation. It is limited to only accepting RDF in the turtle serialisation and does not support the use of language tags. The focus of the tool is not on the user experience when validating an RDF document. Instead the purpose of the tool is on demonstrating the capabilities of the ShEx language. Thus it has several features that are not required for the validation use case, e.g. the ability to generate SPARQL queries to perform the validation. The error messages generated are terse and require knowledge of ShEx. We have aimed to provide more user oriented error messages.

ShExScala[14] developed by Jose Emilo Labra Gayo is a Scala implementation of a ShEx validator. It supports multiple RDF serialisations and can be used to validate an RDF dataset from a given URL or SPARQL endpoint. It was not possible to use our implementation as it requires a server-side deployment.

The most similar work to our own is the DCAT validator developed as part of the Google Summer of Code 2015[15]. This is a web application for validating DCAT dataset descriptions [5] written entirely in Javascript. Users can enter their description by file upload, by URI, or direct entry. As with our user interface, the reporting of validation results are split into errors and warnings. The validation constraints are captured in a JSON array as key value pairs, rather than reusing one of the approaches being developed in the W3C RDF Data Shapes Working Group. For the DCAT deployment, it is unclear where the choice of mandatory and optional constraints has come from. Entering the examples from the DCAT recommendation result in validation errors.

8 Conclusions

Validata provides a reconfigurable, online tool for validating RDF documents against a set of constraints defined in a ShEx schema. Informative error and

[13] `http://www.w3.org/2013/ShEx/FancyShExDemo.html` accessed November 2015
[14] `http://labra.github.io/ShExcala/` accessed November 2015
[15] System: `http://www.dcat.be:8080/validator/`
Code: `https://github.com/oSoc15/dcat-validator`
Blog: `http://open.summerofcode.be/2015/07/10/dcat/` all accessed Nov 2015

warning messages are presented to the user to enable them to refine their RDF to conform with the schema. The validation implementation extends the standard ShEx definition to support different requirement levels; with the levels being defined in the schema definition rather than prescribed by the ShEx language.

The tool has been deployed by the W3C HCLS Interest Group to validate dataset descriptions against their community profile. Additional deployments have been made for the DCAT specification and the Open PHACTS project.

Future work on the Validata tool includes improving the user experience, e.g. allowing users to save edited RDF files and improving the selection of resources to validate. Once these changes to the UI have been made a user evaluation should be conducted.

Acknowledgements

We thank Eric Prud'hommeaux for the support and advice received throughout the development of Validata.

References

1. Hors, A.L., Solbrig, H., Prud'hommeaux, E.: Rdf validation workshop report: Practical assurances for quality rdf data. Report, W3C (December 2012) `http://www.w3.org/2012/12/rdf-val/report`.
2. Prud'hommeaux, E., Labra Gayo, J.E., Solbrig, H.: Shape expressions: An RDF validation and transformation language. In: 10th International Conference on Semantic Systems. (2014) 32–40 doi:10.1145/2660517.2660523.
3. Gayo, J.L., Prudhommeaux, E., Solbrig, H., Rodríguez, J.A.: Validating and describing linked data portals using RDF Shape Expressions. In: Workshop on Linked Data Quality. (2014) `http://ceur-ws.org/Vol-1215/paper-06.pdf`.
4. Gray, A.J.G., Baran, J., Marshall, M.S., Dumontier, M.: Dataset descriptions: HCLS community profile. Interest group note, W3C (May 2015) `http://www.w3.org/TR/hcls-dataset/`.
5. Maali, F., Erickson, J.: Data catalog vocabulary (DCAT). Recommendation, W3C (January 2014) `http://www.w3.org/TR/vocab-dcat/`.
6. Bradner, S.: Key words for use in RFCs to indicate requirement levels. Best current practice, Network Working Group, Internet Engineering Task Force (March 1997) `http://www.ietf.org/rfc/rfc2119.txt`.
7. Gaudet, P., Bairoch, A., Field, D., et al.: Towards BioDBcore: a community-defined information specification for biological databases. Database: The Journal of Biological Databases and Curation **2011** (January 2011) doi:10.1093/database/baq027.
8. Gray, A.J.G.: Dataset descriptions for the open pharmacological space. Working draft, Open PHACTS (September 2012) `http://www.openphacts.org/specs/datadesc/`.
9. Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing Linked Datasets with the VoID Vocabulary. Note, W3C (2011) `http://www.w3.org/TR/void/`.
10. Goble, C., Gray, A.J.G., Tatakis, E.: Help me describe my data : A demonstration of the Open PHACTS VoID Editor. In: ISWC 2014 Poster and Demos. (2014)
11. Bento, A.P., Gaulton, A., Hersey, A., et al: The ChEMBL bioactivity database: an update. Nucleic acids research **42**(Database issue) (January 2014) D1083–1090