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/HeriotWattMEng2015/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.

In recent years, there has been a growing interest in having a validation approach for RDF data beyond checking consistency with the OWL ontologies used in the data [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. There is an ongoing W3C working group defining a recommendation for capturing and testing such validation requirements[1]. 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: an online 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

[1] RDF Data Shapes Working Group `http://www.w3.org/2014/data-shapes/` accessed August 2015

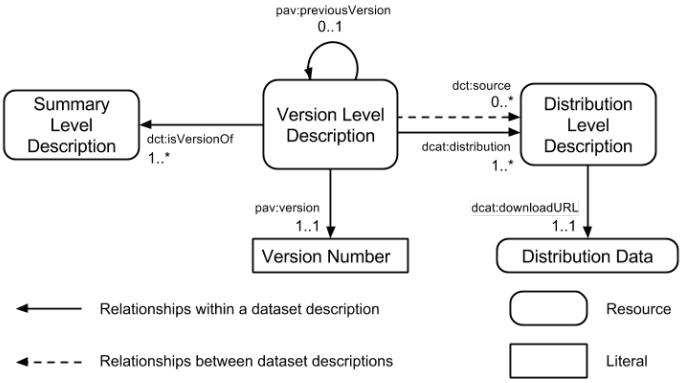

Fig. 1. The three levels of a dataset description.

validation provides support for multiple RDF serialisations as well as different requirement levels – properties that must, should, or may be provided. Validata was developed to provide a tool to check dataset descriptions (RDF documents) against the W3C Health Care and Life Sciences Community Profile for Dataset Descriptions [4] but due to its use of ShEx is capable of being re-purposed for other metadata standards (e.g. DCAT [5]) or validation use cases.

2 Motivating Use Case

The W3C Health Care and Life Sciences (HCLS) Community Profile[4] specifies a common format for dataset descriptions in the Health Care and the Life Sciences domain using the Resource Description Framework (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 proving 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[2] 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 using terms drawn from 18 vocabularies. As such, it is difficult to verify when a dataset

[2] 20 at the time of writing

description conforms to the community profile. To encourage uptake of the community profile, it was recognised that there would be a need to automatically validate the RDF descriptions of datasets against the HCLS Community Profile.

However, 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), or 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 these 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.

As several of the existing conformance schemas make use of different requirement levels, e.g. MUST, SHOULD and MAY as defined in [6], it should be possible to validate at these different requirement levels. However, we do not require that [6] is used for defining the requirement level.

The main motivating use case for our tool comes from the W3C HCLS community profile. Thus the validation tool should be deployable on the W3C HCLS pages. This meant that we needed to develop a tool that 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, i.e. it can be accessed through an API.

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]. This was 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].

FiXme Note: Need to say something about ShEx using shape declarations and a single schema can contain multiple shapes.

```
PREFIX dct: <http://purl.org/dc/terms/>
PREFIX dctypes: <http://purl.org/dc/dcmitype/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

<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.

Figure 2 shows an excerpt of the ShEx schema for the HCLS community profile. The shape defines the properties and objects that a resource that matches the shape should have. 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 all with the '.'.

FiXme Note: Merge issue33 branch into master

The full ShEx schema[3] contains a shape declaration for each of the description types: SummaryLevelShape (shown in Figure 2), 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.

5 Implementation

The Validata tool is a web application that is composed of two modules – a user interface over a standalone ShEx-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 both modules are available from `https://github.com/HeriotWattMEng2015`.

The user interface has been developed using various existing libraries. jQuery[4] is used for DOM manipulation with cross-browser functionality while Twitter Bootstrap[5] is used for cross-browser responsive pages, and Browserify[6] was used to manage dependencies. Finally, CodeMirror[7] was used to provide text editor

[3] `https://github.com/HeriotWattMEng2015/ShEx-validator/blob/master/samples/hcls_may_2015.shex` accessed September 2015
[4] `https://jquery.com` accessed August 2015
[5] `http://getbootstrap.com/` accessed August 2015
[6] `http://browserify.org/` accessed August 2015
[7] `https://codemirror.net/` accessed August 2015

panes which have line numbers. This was essential for providing informative error messages that could reference the line in the original RDF document that was at fault.

The ShEx-Validator has been developed as a node.js module[8] which provides a standalone Javascript runtime environment. The core of the validator is the ShEx Javascript library[9]. The ShEx-Validator separates the tasks of validation 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. peg.js[10] was used to parse the extended grammar to generate the ShEx schema parser. For parsing RDF documents the N3.js library[11] was used which supports Turtle, TriG, N-Triples and N-Quads RDF serializations. Instead of using asynchronous callbacks in the code, the promises library[12] 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 been added to the ShEx validator. This means that if the description of a dataset is given in multiple languages with the same property then it is not an error, but if different properties are used then an error is generated.

A library of unit tests have been developed for the ShEx-Validator backend. These are run using the Mocha framework[13] and ensure that the new features added to the validator do not break existing features.

The advantage of the separation between 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 the dataset descriptions are validated as part of the data are generated. Alternatively it could be used as a library in other tools such as a dataset description editor tool.

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'.

[8] `https://nodejs.org/` accessed August 2015
[9] Developed by Eric Prud'hommeaux `https://github.com/ericprud/ShExDemo/blob/master/RDF.js` initial import hash 96c7fb0; accessed August 2015
[10] `http://pegjs.org/` version 0.8.0; accessed August 2015
[11] Developed by Ruben Verborgh `https://github.com/RubenVerborgh/N3.js` version 0.4.1; accessed August 2015
[12] `https://www.promisejs.org/` version 6.1.0; accessed August 2015
[13] `http://mochajs.org` accessed September 2015

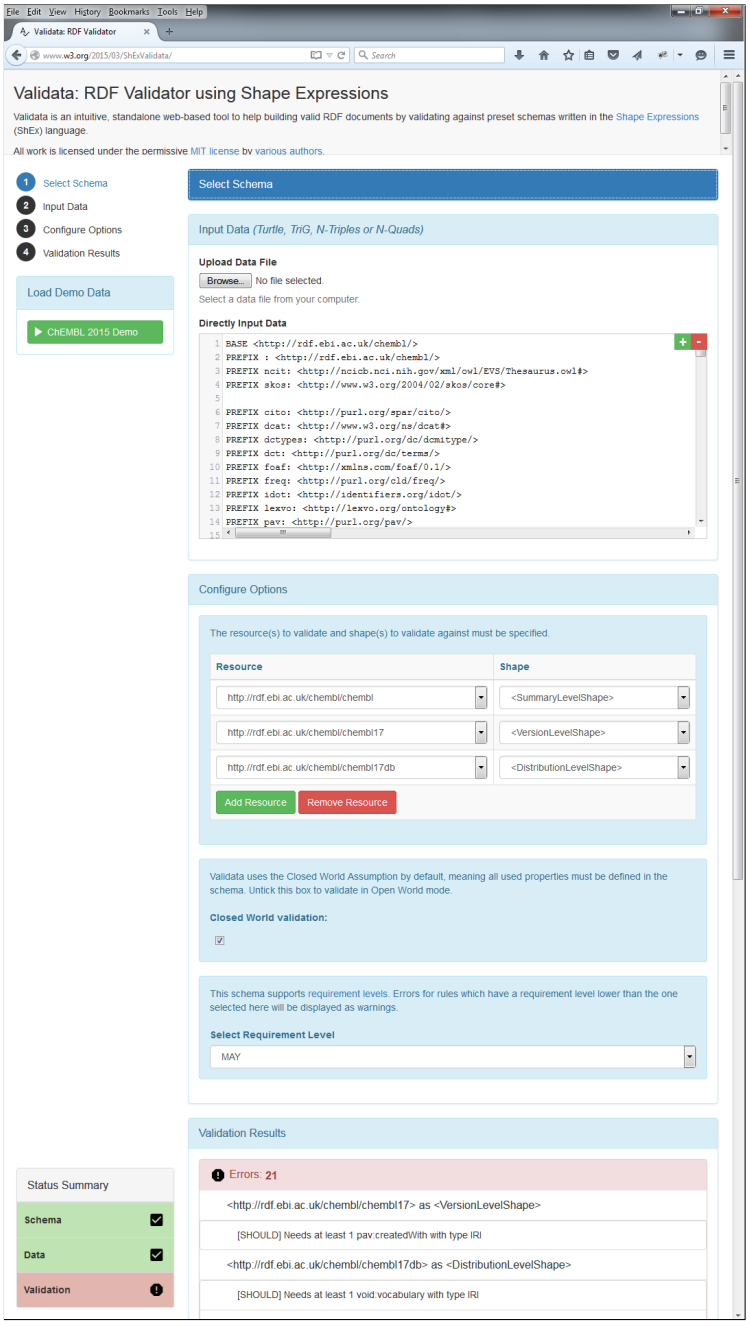

Fig. 3. Screenshot of the Validata tool deployed for the W3C Health Care and Life Sciences Community Profile `http://www.w3.org/2015/03/ShExValidata/`.

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 that a user would follow when using the tool. These are:

1. Select Schema
2. Input Data
3. Configure Options
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 [9].

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.

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).

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. The RDF is then validated against the chosen ShEx schema.

Configure Options Panel. The user may then configure various options that are applied during validation. Initially the ShEx-Validator attempts to 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. The user can use the drop down menus to override these initial settings. Additional

resources from the supplied RDF can be added using the Add Resource button and correspondingly removed using the Remove Resource button. This aspect of the user interface 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 between open and closed world semantics when validating against the ShEx schema [3]. This option controls whether the validator will generate an error if additional properties are discovered in the supplied RDF for a given shape description (closed world) or simply ignore them (open world).

The final configuration option informs the ShEx-Validator at which requirement level the validation should take place. 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 as either errors or warnings. 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 conformance level will be reported as warnings.

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 validation row of the Status Summary will turn green.

Two types 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 of the report 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. However, the requirement level reporting enables a message to be generated, which would not be the case if the ShEx optional properties were used alone.

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 a URI. 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.

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[14] 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 possibilities of ShEx. 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 eliminate to provide more user oriented error messages.

ShExScala[15] 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. However, it was not possible to use this for our implementation due to the requirement for the W3C deployment.

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 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 include ...

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.

FiXme Note: Give details of the OPS test deployment and the DCAT experimental deployment.

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.

[14] http://www.w3.org/2013/ShEx/FancyShExDemo.html accessed September 2015
[15] http://labra.github.io/ShExcala/ accessed September 2015

2. 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 '14, ACM Press (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., Sansone, S.A., Taylor, C., Attwood, T.K., Bateman, A., Blake, J.A., Bult, C.J., Cherry, J.M., Chisholm, R.L., Cochrane, G., Cook, C.E., Eppig, J.T., Galperin, M.Y., Gentleman, R., Goble, C.A., Gojobori, T., Hancock, J.M., Howe, D.G., Imanishi, T., Kelso, J., Landsman, D., Lewis, S.E., Karsch Mizrachi, I., Orchard, S., Ouellette, B.F.F., Ranganathan, S., Richardson, L., Rocca-Serra, P., Schofield, P.N., Smedley, D., Southan, C., Tan, T.W., Tatusova, T., Whetzel, P.L., White, O., Yamasaki, C.: Towards BioDBcore: a community-defined information specification for biological databases. Database: The Journal of Biological Databases and Curation **2011** (January 2011) baq027–baq027 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. Bento, A.P., Gaulton, A., Hersey, A., Bellis, L.J., Chambers, J., Davies, M., Krüger, F.A., Light, Y., Mak, L., McGlinchey, S., Nowotka, M., Papadatos, G., Santos, R., Overington, J.P.: The ChEMBL bioactivity database: an update. Nucleic acids research **42**(Database issue) (January 2014) D1083–1090