



Using Shape Expressions (ShEx) to Share RDF Data Models and to Guide Curation with Rigorous Validation

Katherine Thornton¹(✉), Harold Solbrig², Gregory S. Stupp³,
Jose Emilio Labra Gayo⁴, Daniel Mietchen⁵, Eric Prud'hommeaux⁶,
and Andra Waagmeester⁷

¹ Yale University, New Haven, CT, USA
katherine.thornton@yale.edu

² Johns Hopkins University, Baltimore, MD, USA
solbrig@jhu.edu

³ The Scripps Research Institute, San Diego, CA, USA
gstupp@scripps.edu

⁴ University of Oviedo, Oviedo, Spain
labra@uniovi.es

⁵ Data Science Institute, University of Virginia, Charlottesville, VA, USA
daniel.mietchen@virginia.edu

⁶ World Wide Web Consortium (W3C), MIT, Cambridge, MA, USA
eric@w3.org

⁷ Micelio, Antwerpen, Belgium
andra@micel.io

Abstract. We discuss Shape Expressions (ShEx), a concise, formal, modeling and validation language for RDF structures. For instance, a Shape Expression could prescribe that subjects in a given RDF graph that fall into the shape “Paper” are expected to have a section called “Abstract”, and any ShEx implementation can confirm whether that is indeed the case for all such subjects within a given graph or subgraph.

There are currently five actively maintained ShEx implementations. We discuss how we use the JavaScript, Scala and Python implementations in RDF data validation workflows in distinct, applied contexts. We present examples of how ShEx can be used to model and validate data from two different sources, the domain-specific Fast Healthcare Interoperability Resources (FHIR) and the domain-generic Wikidata knowledge base, which is the linked database built and maintained by the Wikimedia Foundation as a sister project to Wikipedia. Example projects that are using Wikidata as a data curation platform are presented as well, along with ways in which they are using ShEx for modeling and validation.

When reusing RDF graphs created by others, it is important to know how the data is represented. Current practices of using human-readable descriptions or ontologies to communicate data structures often lack sufficient precision for data consumers to quickly and easily understand data representation details. We provide concrete examples of how we

The original version of this chapter was revised: By mistake the chapter was originally published non open access. The correction to this chapter is available at <https://doi.org/10.1007/978-3-030-21348-0.40>

© The Author(s) 2019
P. Hitzler et al. (Eds.): ESWC 2019, LNCS 11503, pp. 606–620, 2019.
<https://doi.org/10.1007/978-3-030-21348-0.39>

use ShEx as a constraint and validation language that allows humans and machines to communicate unambiguously about data assets. We use ShEx to exchange and understand data models of different origins, and to express a shared model of a resource’s footprint in a Linked Data source. We also use ShEx to agilely develop data models, test them against sample data, and revise or refine them. The expressivity of ShEx allows us to catch disagreement, inconsistencies, or errors efficiently, both at the time of input, and through batch inspections.

ShEx addresses the need of the Semantic Web community to ensure data quality for RDF graphs. It is currently being used in the development of FHIR/RDF. The language is sufficiently expressive to capture constraints in FHIR, and the intuitive syntax helps people to quickly grasp the range of conformant documents. The publication workflow for FHIR tests all of these examples against the ShEx schemas, catching non-conformant data before they reach the public. ShEx is also currently used in Wikidata projects such as Gene Wiki and WikiCite to develop quality-control pipelines to maintain data integrity and incorporate or harmonize differences in data across different parts of the pipelines.

Keywords: RDF wd:Q54872 · ShEx wd:Q29377880 ·
FHIR wd:Q19597236 · Wikidata wd:Q2013 ·
Digital preservation wd:Q632897

1 Introduction

The RDF data model is a core technology of the Semantic Web. RDF is used to integrate data from heterogeneous sources, is extensible, flexible and can be manipulated with the SPARQL query language [9].

The need to describe the topologies, or shapes, of RDF graphs triggered the creation of an early version of Shape Expressions (ShEx 1) and the formation of a World Wide Web Consortium (W3C) Working Group—the Data Shapes Working Group—in 2014 [15]. Its task was to recommend a technology for describing and expressing structural constraints on RDF graphs. This has led to SHACL [8]—another shape-based data validation language for RDF—and further development of ShEx.

We provide an overview of ShEx, discuss implementations of the language, and then consider use cases for the validation of RDF data. The use cases we present consist of two types. For the first type, which is domain-specific, we provide an overview of how ShEx is being used for validation in medical informatics. For the second type, which is domain-generic, we provide examples that involve validation of entity data from the Wikidata knowledge base. We analyze workflows and highlight the affordances of multiple implementations of ShEx.

2 Shape Expressions

The Shape Expressions (ShEx) schema language can be consumed and produced by humans and machines [9] and is useful in multiple contexts. ShEx can be used

in model development, both for creating new models as well as for revising existing ones. ShEx is helpful for legacy review, where punch lists can be created for existing data issues that need to be fixed. ShEx is useful as documentation of models because it has a terse, human-readable representation that helps contributors and maintainers quickly grasp the model and its semantics. ShEx can be used for client pre-submission, when submitters test their data before submission to make sure they are saying what they want to say and that the receiving schema can accommodate all of their data. ShEx can also be used for server pre-ingestion, through a submission process that checks data as it comes in, and either rejects or warns of non-conformant data.

ShEx's semantics have undergone considerable peer review. [2] compares it with SHACL and discusses stratified negation and validation algorithms. [23] analyzes the complexity and expressive power of ShEx. With extensions like ShExMap [16], ShEx can generate an in-memory structure of the validated RDF, from which it is possible to operate, much like XSLT does for XML. Some experimental ShEx 1 extensions translated from RDF to XML¹ and JSON² [15]. To date, there are three serializations and five implementations that are actively maintained. We will discuss three of the implementations in this paper.

2.1 ShEx Implementations

shex.js for Javascript/N3.js. The `shex.js`³ JavaScript implementation of ShEx was used to develop the ShEx language and test suite⁴ and is generally used as a proving ground for language extensions. It was used to develop Gene Wiki⁵, WikiCite⁶ and FHIR/RDF schemas [21]. The online validator⁷ was used to develop and experiment with all of these schemas. In addition, the FHIR/RDF document production pipeline used its REST interface, and the Gene Wiki and WikiCite projects used its command line interface to invoke it in `node.js`. The development of the Gene Wiki schemas uses several branches of `shex.js` that are aggregated into a single “wikidata” branch⁸.

Shaclex. `Shaclex`⁹ is a Scala implementation of ShEx and SHACL. The library uses a purely functional approach where the validation is defined using monads and monad transformers [11]. The validator is defined in terms of a simple RDF interface (SRDF) that has several implementations. Two implementations are

¹ <http://w3.org/brief/NTAx>.

² <http://w3.org/brief/NTAy>.

³ <http://github.com/shexSpec/shex.js>.

⁴ <https://github.com/shexSpec/shexTest>.

⁵ <https://github.com/SuLab/Genewiki-ShEx>.

⁶ <https://github.com/shexSpec/schemas/tree/master/Wikidata/wikicite>.

⁷ <https://rawgit.com/shexSpec/shex.js/master/doc/shex-simple.html>.

⁸ <https://rawgit.com/shexSpec/shex.js/wikidata/doc/shex-simple.html>.

⁹ <http://labra.weso.es/shaclex/>.

based on RDF models that can be created using Apache Jena¹⁰ or RDF4J¹¹. Another implementation of the simple RDF is based on SPARQL endpoints, so the validator can be used to validate the RDF data that can be accessed through those endpoints. By leveraging Apache Jena or RDF4J libraries, the Shaclex library can take as input RDF defined in all the serialization syntaxes that they support, e.g. Turtle, RDF/XML, JSON-LD, or RDF/JSON. Shaclex also has an online demonstrator, available at <http://shaclex.validatingrdf.com/>.

PyShEx. PyShEx¹² is a Python 3 implementation of the ShEx¹³ specification. It uses the underlying model behind the ShEx JSON format (ShExJ)¹⁴ as the abstract syntax tree (AST), meaning that ShEx schemas in the JSON format can be directly loaded and processed. PyShEx uses the PyShExC parser¹⁵ to transform ShEx compact format (ShExC) schemas into the same target AST. PyShEx is based on the native Python RDF library – the rdflib¹⁶ package – meaning that it can support a wide variety of RDF formats. PyShEx can also use the sparql_slurper¹⁷ package to fetch sets of triples on demand from a SPARQL endpoint. An example of PyShEx can be found at <https://tinyurl.com/ycubhlog>.

2.2 Interoperability

The three implementations above offer a consistent command line and web invocation API. These same parameters can be embedded in “manifest” files, which store a list of objects that encapsulate an invocation. The shex.js and shaclex implementations offer a user interface allowing a user to select and execute elements in the manifest. In addition to agreement on the semantics of validation, this interface interoperability makes it trivial to swap between implementations, e.g. depending on immediate platform and user interface preferences.

3 Use Cases

We present use cases that encompass two distinct models for validation. In the first use case, validation is performed on clinical data in an institutional context. In the second group of use cases, validation is performed via the Wikidata Query Service, a public SPARQL endpoint maintained as part of the Wikidata infrastructure.

¹⁰ <https://jena.apache.org/>.

¹¹ <http://rdf4j.org/>.

¹² <https://github.com/hsolbrig/PyShEx>.

¹³ <http://shex.io/shex-semantic/>.

¹⁴ <https://github.com/hsolbrig/ShExJSG>.

¹⁵ <https://github.com/shexSpec/grammar/tree/master/parsers/python>.

¹⁶ <http://rdflib.readthedocs.io/en/stable/>.

¹⁷ https://github.com/hsolbrig/sparql_slurper.

3.1 Domain-Specific ShEx Validation in Medical Informatics

The Yosemite Project [29] started in 2013 as response to a 2010 report by the President’s Council of Advisors on Science and Technology [14] calling for a universal exchange language for healthcare. As part of its initial efforts, this project released the “Yosemite Manifesto”¹⁸, a position statement signed by over 100 thought leaders in healthcare informatics which recommended RDF as the “best available candidate for a universal healthcare exchange language” and stating that “electronic healthcare information should be exchanged in a format that either: (a) is in RDF format directly; or (b) has a standard mapping to RDF”.

Around the same time as the Yosemite Project meeting, a new collection of standards for the exchange of clinical data was beginning to gather momentum. “Fast Healthcare Interoperability Resources (FHIR)” [4] defined a modeling environment, framework, community and architecture for the REST oriented access to clinical resources. The FHIR specification defines some 130+ healthcare and modeling related “resources” and describes how they are represented in XML¹⁹ and JSON²⁰. One of the outcomes of the Yosemite project was the formation of the FHIR RDF/Health Care Life Sciences (FHIR/HCLS) working group²¹ tasked with defining an RDF representation format for FHIR resources.

ShEx played a critical role in the development of the FHIR RDF specification. Prior to its introduction to ShEx, the community tried to use a set of representative examples as the basis for discussion. This was a slow process, as the actual rules for the underlying transformation were implicit. There was no easy way to verify that the examples covered all possible use cases and that they were internally self-consistent. Newcomers to the project faced a steep learning curve. The introduction of ShEx helped to streamline and formalize the process [21]. Instead of talking in terms of examples, the group could address how instances of *entire* FHIR resource models would be represented as RDF. Edge cases that seldom appeared received the same scrutiny as did everyday usage examples. The proposed transformation rules could be implemented in software, with the entire FHIR specification being automatically transformed to its ShEx equivalent.

ShEx allowed the participants to finalize discussions and settle on a formal model and first specification draft in less than three months. A formal transformation was created to map the (then) 109 FHIR resource definitions into schemas for the RDF binding. This transformation uncovered several issues with the specification itself as well as providing a template for the bidirectional transformation between RDF and the abstract FHIR model instances. The documentation production pipeline was additionally extended to transform the 511 JSON and XML examples into RDF, which were then tested against the generated ShEx schemas. [21] These tests both caught multiple errors in the transformation software and

¹⁸ <http://yosemitemanifesto.org/>.

¹⁹ <http://hl7.org/fhir/xml.html>.

²⁰ <http://hl7.org/fhir/json.html>.

²¹ <https://www.w3.org/community/hclscg/>.

uncovered a number of additional issues in the specification itself, ensuring that the user-facing documentation was accurate and comprehensive. In early 2017, the FHIR documentation production framework, written in Java, switched from using the `shex.js` implementation to natively calling the Shaclex implementation. As a testament to the quality of the standard, both implementations agreed on the validity of all 511 examples. The first official version of the FHIR RDF specification was released in the FHIR Standard for Trial Use (STU3) release [5] in April of 2017.

3.2 Domain-Generic ShEx Validation in Wikidata

What Wikipedia is to text, Wikidata is to data: an open collaboratively curated resource that anyone can contribute to. In contrast to the language-specific Wikipedias, Wikidata is Semantic Web-compatible, and most of the edits are made using automated or semi-automated tools. This ‘data commons’ provides structured public data for Wikipedia articles [19] and other applications. For each Wikipedia article—in any language—there is an item in Wikidata, and if the same concept is described in more than one Wikipedia, then Wikidata maintains the links between them.

In contrast to language-specific Wikipedias, and to most other sites on the web, Wikidata does not assume that users who collaborate have a common natural language [7]. In fact, consecutive editors of a given Wikidata entry often do not share a language other than some basic knowledge about the Wikidata data model. Using ShEx to make those data models more explicit can improve such cross-linguistic collaboration.

Wikidata is hosted on Wikibase, a non-relational database maintained by the Wikimedia Foundation. The underlying infrastructure also contains a SPARQL engine <https://query.wikidata.org> that feeds on a triplestore which is continuously synchronized with Wikibase. This synchronization—which occurs in seconds—enables data in Wikidata to be available as Linked Data almost immediately and thus becoming part of the Semantic Web. Basically, Wikidata acts as an “edit button” to the Semantic Web and as an entry point for users who otherwise do not have the technical background to use Semantic Web infrastructure. While Wikidata and its RDF dump are technically separate, they can be perceived as one from a user perspective. Content negotiation presents either the Wikibase form or the RDF form, creating a sense of unity between the two. For instance, <https://wikidata.org/entity/Q54872> (which identifies RDF) points to the Wikibase entry at <https://www.wikidata.org/wiki/Q54872>, while <http://www.wikidata.org/entity/Q54872.ttl> will provides the Turtle representation and <http://www.wikidata.org/entity/Q54872.json> a JSON export.

The Wikidata data model [28] currently consists of two entity types: **items** and **properties** (a third one, for lexemes, is about to be introduced). All entities have persistent identifiers composed of single-letter prefixes (*Q* for items, *P* for properties, *L* for lexemes) plus a string of numbers and are allotted a page in Wikidata. For instance, the entity [Q1676669](https://www.wikidata.org/wiki/Q1676669) is the item for *JPEG File Interchange Format, version 1.02*. Properties like *instance of* ([P31](https://www.wikidata.org/wiki/P31)) and *part*

of (P361) are used to assert *claims* about an item. A claim, its references and qualifiers form a *statement*. Currently, Wikidata’s RDF graph comprises about 5 billion triples (with millions added per day), which reflects about 500 million statements involving about 50 million items and roughly 5000 properties.

Besides serving Wikipedia and its sister projects, Wikidata also acts as a data backend for a complex ecosystem of tools and services. Some of these are general-purpose semantic tools like search engines or personal assistants [1], while others are tailored for specific scientific communities, e.g. Wikigenomes [18] for curating microbial genomes, WikiDP for digital preservation of software [26], or Scholia [13] for exploring scholarly publications. Through such tools, communities that are not active on Wikidata can engage with the Wikidata RDF graph. ShEx can facilitate that.

Non-ShEx Validation Workflows for Wikidata. Wikidata uses constraints for validation in multiple ways. For instance, some edits are rejected by the user interface or the API, e.g. certain formats or values for dates cannot be saved. Some of the quality control also involves patrolling individual edits [20].

Most of the quality control, however, takes place on the data itself. Initially, the primary mechanism for this was a system of Mediawiki templates²², similar to the infobox templates on Wikipedia. These templates express a range of constraints like “items about movies should link to the items about the actors starring in it” or “this property should only be used on items that represent human settlements” or a regular expression specifying the format of allowed values for a given property. For more complex constraints, some SPARQL functionality is available through such templates. In addition, an automated tool goes through the data dumps on a daily basis, identifies cases where such template-based constraints have been validated, and posts notifications on dedicated wiki pages where Wikidata editors can review and act on them²³. This template-based validation infrastructure, while still largely functional, has been superseded by a parallel one that has been built later by having dedicated properties²⁴ for expressing constraints on individual properties or their values or on relationships involving several properties or specific classes of items. For instance, P1793 is for “format as a regular expression”, P2302 more generally for “property constraint”, and P2303 for “exception to constraint” (used as a qualifier to P2302). This way, the constraints themselves become part of the Wikidata RDF graph. This arrangement is further supported by dedicated Mediawiki extensions²⁵, one of which also contains a gadget that logged-in users can enable in their preferences in order to be notified through the user interface if a constraint violation has been detected on the item or statement they are viewing. Reading through the reports generated by constraint violation systems supports inspection on a per-property basis. This system of validation requires community members to

²² https://www.wikidata.org/wiki/Category:Constraint_templates.

²³ https://www.wikidata.org/wiki/Wikidata:Database_reports/Constraint_violations.

²⁴ https://www.wikidata.org/wiki/Help:Property_constraints_portal.

²⁵ https://www.mediawiki.org/wiki/Wikibase_Quality_Extensions.

create and apply constraint properties on each of the Wikidata properties, of which there are more than five thousand. Constraints have not yet been added to all properties.

ShEx is a context-sensitive grammar with algebraic operations while Property Validation is a context-free. Unlike in ShEx Validation, where properties have context-sensitive constraints, Property Validation constraints must be permissive enough to permit all current or expected uses of the property. For example, a ShEx constraint that every human gene use the common property P31 “instance of” to declare itself an instance of a human gene MUST NOT be expressed as a property constraint as P31 is used for 56,000 other classes. Of course Property Validation is additionally problematic because the author of a constraint may not be aware of its use in other classes. ShEx allows us to write schemas that describe multiple properties, their constraints, their permissible values in combinations for which there are not yet property constraints created in Wikidata’s infrastructure. This allows us to test conformance to schemas that describe features that may not yet be relevant for the Wikidata community, but may be necessary for an external application.

Generic ShEx Validation Workflow for Wikidata. One issue with the existing template-based constraint and validation mechanisms for Wikidata is that they are usually very specific to the Wikidata platform or to the tools used for interacting with it. ShEx provides a way to link Wikidata-based validation with validation mechanisms developed or used elsewhere. Getting there from the RDF representation of the Wikidata constraints is a relatively small step.

Efforts around the usage of ShEx on Wikidata are coordinated by WikiProject ShEx²⁶. The ShEx-based validation workflow for Wikidata consists of:

1. writing a schema for the data type in question, or choosing an existing one;
2. transferring that schema into the Wikidata model of items, statements, qualifiers and references;
3. writing a ShEx manifest for the Wikidata-based schema;
4. testing entity data from Wikidata for conformance to the ShEx manifest.

Initially, Wikidata may be missing some properties for adequately representing such a schema. Such missing properties can be proposed and, after a process involving community input, created. Once they appear in the Wikidata RDF graph, ShEx can be used to validate the corresponding RDF shapes.

At present, the ShEx manifests for Wikidata are hosted on GitHub, but they could be included into the Wikidata infrastructure, e.g. through a dedicated property similar to *format as a regular expression* (P1793).

²⁶ https://www.wikidata.org/wiki/Wikidata:WikiProject_ShEx.

4 ShEx Validation of Domain-Specific Wikidata Subgraphs

4.1 Molecular Biology

In 2008, the Gene Wiki project started to create and maintain infoboxes in English-language Wikipedia articles about human genes [6]. After the launch of Wikidata in 2012, the project shifted from curating infoboxes on Wikipedia pages towards curating the corresponding items on Wikidata [12]. Since then, Gene Wiki bots have been enriching and synchronizing Wikidata with knowledge from public sources about biomedical entities such as genes, proteins, and diseases, and are now regularly feeding Wikidata with life science data [3]. To date, there are items about ~24k human and 20k mouse genes from NCBI Gene²⁷, 8,700 disease concepts from the Disease Ontology²⁸, and 2,700 FDA-approved drugs.

The Gene Wiki bots are built using a Python framework called the Wikidata Integrator (WDI)²⁹. This platform is using the Wikidata API and does concept resolution based on external identifiers. The WDI is openly available.

Validation Workflows for Gene Wiki. In the Gene Wiki project, the focus is on synchronizing data between Wikidata and external databases. After the data models used by these external sources have been translated into Wikidata terms and the missing properties created, one or more exemplary entities from the sources in question are chosen and manually completed on Wikidata. Upon reaching consensus on the validity of these items and their data model, a bot is developed to reproduce these handmade Wikidata entries. Once the bot is able to replicate the items as they are, more items are added to Wikidata. This is done gradually to allow community input; first 10 items, then 100, then 1000 and finally all. During the development of a bot, it is run manually (at the developer's discretion). Upon completion of development, the bots are run from an automation platform where the sources are synchronized regularly³⁰.

ShEx has its value in both the development phase and the automation phase. During development, ShEx is used as a communication tool to express the data model being discussed. For instance, <https://github.com/SuLab/Genewiki-ShEx/blob/master/genes/wikidata-human-genes.shex> contains the data model of a human gene as depicted in Wikidata (note the many uses of the comment sign “#”). Currently, data-model design is done in parallel by writing ShEx and drawing graphical depictions of these models. We are currently working towards creating ShEx from a drawn diagram.

After completion of the bot, ShEx can be used to monitor for changes in the data of interest. This is either novel data, disagreement or vandalism. Regularly, all Wikidata items on a specific source/semantic type are collected and tested for inconsistencies.

²⁷ <https://www.ncbi.nlm.nih.gov/gene/>.

²⁸ <http://disease-ontology.org/>.

²⁹ <https://github.com/SuLab/WikidataIntegrator>.

³⁰ <http://jenkins.sulab.org>.

4.2 Software and File Formats

Metadata about software, file formats and computing environments is necessary for the identification and management of these entities. Creating machine-readable metadata about resources in the domain of computing allows digital preservation practitioners to automate programmatic interactions with these entities. People working in digital preservation have a shared need for accurate, reusable, technical and descriptive metadata about the domain of computing.

Wikidata’s WikiProject Informatics³¹ collaboratively models the domain of computing [25]. Until now, members of the Wikidata community have created items for more than 85,000 software titles³² and more than 3,500 file formats³³.

Schemas for software items³⁴ and file format items³⁵ in Wikidata have been created and entity data was tested using the ShEx2 Simple Online Validator (see Footnote 7). In order to use ShEx, we created manifests for software items³⁶ and file format items³⁷. These manifests contain a SPARQL query for the Wikidata Query Service Endpoint that gathers all of the Wikidata items one wishes to test for conformance. The online validator accepts the manifest and then tests the entity data pertaining to each item against the schema for conformance. It provides information about conformance status and error messages.

The ability to validate subgraphs of Wikidata pertaining to the domain of computing allows us to quickly get a sense of how other editors are modeling their data by identifying Wikidata items for which the entity data graph is not in conformance with our schema. This allows us to communicate detailed information about data quality metrics to other members of the digital preservation community. Outputs of validation from ShEx tools provide evidence we can incorporate into our data quality metrics. This allows us to communicate with precision and accuracy, which then allows us to build trust with members of the community who are unfamiliar with the work processes of the knowledge base that anyone can edit.

4.3 Bibliographic Metadata

WikiCite is an effort to collect bibliographic information in Wikidata [24]. Launched in 2016, it is concerned with developing Wikidata-based schemas for publications – such as monographs, scholarly articles, or conference proceedings – and with the application of such schemas to Wikidata items representing publications and related concepts (e.g. authors, journals, publishers, topics). While

³¹ https://www.wikidata.org/wiki/Wikidata:WikiProject_Informatics.

³² <https://github.com/emulatingkat/SPARQL/blob/master/software/softwareCount.rq>.

³³ <https://github.com/emulatingkat/SPARQL/blob/master/fileFormat/ffCount.rq>.

³⁴ <https://tinyurl.com/y46lotsy>.

³⁵ <https://tinyurl.com/y3spaw87>.

³⁶ <https://tinyurl.com/yxmugpvl>.

³⁷ <https://tinyurl.com/y6h5at4a>.

these schemas are mature enough to be encoded in a range of tools used for interacting with the WikiCite subgraph of Wikidata, they are still in flux, and using ShEx—especially with an interoperable set of implementations and graphic and multilingual layers on top if it—could help coordinate community engagement around further development. At present, the WikiCite community is curating around 15 million Wikidata items about ca. 700 types of publications³⁸, which are linked to each other through a dedicated property *cites* (P2860) as well as with other items, e.g. about authors, journals, publishers or the topics of the publications, and with external resources. Several hundred properties are in use in these contexts, the majority of which are for external identifiers.

The usage of ShEx in WikiCite is currently experimental, with tests being performed via the ShEx2 Simple Online Validator. Drafts of ShEx manifests exist for a small number of publication types like conference proceedings or journal articles as well as for specific use cases like defining a particular subset of the literature, e.g. on a specific topic. One such literature corpus is that about the Zika virus³⁹. In this context, a ShEx manifest has been drafted⁴⁰ that goes beyond the publications themselves and includes constraints about the way the authors and topics of those publications are represented. It is currently being tested, compared against the existing non-ShEx validation mechanisms and developed further. Other use cases include curating the literature by author (e.g. in the context of working on someone’s biography), by funder (e.g. for evaluating research outputs), or by journal or publisher (e.g. in the context of digital preservation).

5 Discussion

5.1 Novelty of Validation of RDF Data Using ShEx

RDF has been “on the radar” for the healthcare domain for a number of years, but always as a speculation: “If we could figure out how to build it, maybe they would come”. ShEx proved to be the key that enabled actual action, and it moved RDF from a topic of discussion to an active implementation. ShEx provided a formal, yet (relatively) easy to understand view of what the RDF associated with a particular model element would look like. It provided a mechanism for testing data for conformance, as well as a framework for *assembling* the elements of an RDF triple store into pre-defined structures. ShEx has the potential to define a unifying semantic for multiple modeling paradigms – in the case of FHIR, ShEx is able to represent the intent of the FHIR structure definitions model, constraint language and extension model in a single, easy to understand idiom. While it is yet to be fully explored, ShEx has exciting potential as a data mapping language, with early explorations showing real promise as an RDF transformation language [16]. The validation workflows introduced above for the

³⁸ <http://wikicite.org/statistics.html>.

³⁹ https://www.wikidata.org/wiki/Wikidata:WikiProject_Zika_Corpus.

⁴⁰ <https://tinyurl.com/yyf42sal>.

Wikidata cases are the first application of shapes to validate entity data from Wikidata. The impact of software frameworks that support validation of entity data is an important improvement in the feasibility of ensuring data quality for the Wikidata ecosystem and facilitating cross-linguistic collaboration. Wikidata data models are defined by the community and the knowledge base is designed to support multiple epistemological stances [27]. Wikidata contributors may model data differently from one another. ShEx makes it possible to validate entity data across the entire knowledge base, a powerful tool for data quality.

5.2 Uptake of ShEx Tooling

ShEx schemas are highly re-usable in that they can be shared and exchanged. The fact that ShEx schemas are human readable means that others can understand them and evaluate their suitability for reuse. ShEx schemas can also be extended. The ShEx Community Group of the W3C⁴¹ maintains a repository of ShEx schemas⁴² published under the MIT license that others are free to reuse, modify, or extend to fit novel use cases. We recommend that ShEx manifests be licensed as liberally as possible, so as to facilitate and encourage their usage. The Gene Wiki team led the way with workflows for the validation of entity data in Wikidata. An example of the uptake of ShEx tooling is that the Wikidata for Digital Preservation community modeled their validation workflow on that of the Gene Wiki team. We demonstrate the portability of these workflows for additional domains covered by the Wikidata knowledge base. Once a domain-based group has created ShEx schemas for the data models relevant for their area, others can follow this model to develop a validation workflow of their own.

5.3 Soundness and Quality

The ShEx test suite (see Footnote 4) consists of 1088 validation tests, 99 negative syntax tests, and 14 negative structure tests and 408 schema conversion tests between ShExC, ShExJ and ShExR. Work described in [2] provides efficient validation algorithms and verifies the soundness of recursion. [23] identifies the complexity and expressive power of ShEx. The comprehensive ShEx test suite (see Footnote 4) ensures compliance with these semantics. These projects used ShEx because it (1) has many implementations to choose from (2) has a well-engineered and tested, stable, human-readable syntax (3) is sound with respect to recursion. On the other hand, using ShEx poses new challenges about best practices to integrate the validation step into the data production pipeline, the performance of the validation for large RDF graphs and the interplay of ShEx with other Semantic Web tools like SPARQL, RDFS, or OWL.

⁴¹ <https://www.w3.org/community/shex/>.

⁴² <https://github.com/shexSpec/schemas>.

5.4 Availability

The ShEx Specification is available under the W3C Community Contributor License Agreement⁴³. In addition to the specification itself, the ShEx community also created a Primer⁴⁴ that provides additional explanation and illustrative examples of how to write schemas. All of the software tools we describe are available under an open source license which is either the MIT or the Apache license. The developers of these software frameworks have made them available for anyone to reuse [10, 17, 22]. Contributing to open specifications and releasing software tools under free and open licenses lowers barriers to entry for others who might like to explore, test or adopt ShEx. The use cases we present are evidence of how ShEx validation is applicable to different domains. Extending it to additional domains is the goal of a dedicated initiative in the Wikidata community, the aforementioned WikiProject ShEx.

6 Conclusion

The ability to test the conformance of RDF graph data shapes advances our ability to realize the vision of the Semantic Web. Validating RDF data through the use of ShEx allows for the integration of data from heterogeneous sources and provides a mechanism for testing data quality that has been adopted by communities in different domains. Using ShEx in data modeling phases allows communities to resolve ambiguity of interpretation that can arise when using diagrams or natural language. Through a data modeling process using ShEx, these differences are resolved earlier in a workflow, and reduce time spent fixing errors that could otherwise arise due to different understandings of model meaning. Using ShEx to validate RDF data allows communities to discover all places where data is not yet in conformance to their schema. From the validation phase, a community will generate a punch list of data needing attention. Not only does this allow us to improve data quality, it defines a practical workflow for addressing non-conformant data. Consumers of RDF data will benefit from the work of data publishers who create ShEx schemas to communicate the structure of the data. The use cases presented here demonstrate the viability of using ShEx in production workflows in several different domains. ShEx addresses the challenges of communicating about the structure of RDF data, and will facilitate wider adoption of RDF data in a broad range of data publishing contexts.

Acknowledgements. We would like to thank the members of the W3C Shape Expressions Community Group for insightful conversations and productive collaboration. We would also like to thank the members of the Wikidata community. This work was supported by the National Institutes of Health under grant GM089820. Portions of this work were also supported in part by NIH grant U01 HG009450.

⁴³ <https://www.w3.org/community/about/agreements/cla/>.

⁴⁴ <http://shex.io/shex-primer/>.

References

1. Bielefeldt, A., Gonsior, J., Krötzsch, M.: Practical linked data access via SPARQL: the case of Wikidata. In: Proceedings of the WWW 2018 Workshop on Linked Data on the Web (LDOW 2018). CEUR Workshop Proceedings. CEUR-WS.org (2018)
2. Boneva, I., Labra Gayo, J.E., Prud'hommeaux, E.: Semantics and validation of shapes schemas for RDF (2017)
3. Burgstaller-Muehlbacher, S., et al.: Wikidata as a semantic framework for the Gene Wiki initiative. Database (Oxford) 2016 (2016)
4. HL7: Welcome to FHIR. <https://hl7.org/fhir/>
5. HL7: WFHIR release 3 (STU). <https://hl7.org/fhir/STU3/index.html>
6. Huss, J.W., et al.: A gene wiki for community annotation of gene function. PLoS Biol. **6**(7), e175 (2008)
7. Kaffee, L.A., Piscopo, A., Vougiouklis, P., Simperl, E., Carr, L., Pintscher, L.: A glimpse into Babel: an analysis of multilinguality in Wikidata. In: Proceedings of the 13th International Symposium on Open Collaboration, OpenSym 2017, pp. 14:1–14:5. ACM, New York (2017). <https://doi.org/10.1145/3125433.3125465>
8. Knublauch, H., Kontokostas, D.: Shapes Constraint Language (SHACL). W3C Recommendation, June 2017. <https://www.w3.org/TR/shacl/>
9. Labra Gayo, J.E., Prud'Hommeaux, E., Boneva, I., Kontokostas, D.: Validating RDF Data. Morgan & Claypool Publishers, San Rafael (2017)
10. Labra Gayo, J.E.: SHACLex: Scala implementation of ShEx and SHACL, April 2018. <https://doi.org/10.5281/zenodo.1214239>
11. Liang, S., Hudak, P., Jones, M.: Monad transformers and modular interpreters. In: Proceedings of the 22nd ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL 1995, pp. 333–343. ACM, New York (1995). <http://doi.acm.org/10.1145/199448.199528>
12. Mitra, E., Waagmeester, A., Burgstaller-Muehlbacher, S., Schriml, L.M., Su, A.I., Good, B.M.: Wikidata: a platform for data integration and dissemination for the life sciences and beyond. bioRxiv (2015). <https://doi.org/10.1101/031971>
13. Nielsen, F.Å., Mietchen, D., Willighagen, E.: Scholia, Scientometrics and Wikidata. In: Blomqvist, E., Hose, K., Paulheim, H., Lawrynowicz, A., Ciravegna, F., Hartig, O. (eds.) ESWC 2017. LNCS, vol. 10577, pp. 237–259. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70407-4_36
14. President's Council of Advisors on Science and Technology (PCAST): Report to the President Realizing the Full Potential of Health Information Technology to Improve Healthcare for Americans: The Path Forward (2010). <https://obamawhitehouse.archives.gov/sites/default/files/microsites/ostp/pcast-health-it-report.pdf>
15. 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, pp. 32–40. ACM (2014)
16. Prud'hommeaux, E., Mayo, G.: ShExMap (2015). <http://shex.io/extensions/Map/>
17. Prud'hommeaux, E., et al.: shexSpec/shex.js: release for zenodo DOI (Version v0.9.2), April 2018. <https://doi.org/10.5281/zenodo.1213693>
18. Putman, T.E., et al.: Wikigenomes: an open web application for community consumption and curation of gene annotation data in Wikidata. Database 2017, bax025 (2017). <https://doi.org/10.1093/database/bax025>
19. Sáez, T., Hogan, A.: Automatically generating Wikipedia info-boxes from Wikidata. In: WWW 2018 Companion: The 2018 Web Conference Companion, Lyon, France, 23–27 April 2018. ACM (2018)

20. Sarabadani, A., Halfaker, A., Taraborelli, D.: Building automated vandalism detection tools for Wikidata. CoRR abs/1703.03861 (2017). <http://arxiv.org/abs/1703.03861>
21. Solbrig, H.R., et al.: Modeling and validating HL7 FHIR profiles using semantic web Shape Expressions (ShEx). *J. Biomed. Inform.* **67**, 90–100 (2017)
22. Solbrig, H.: PyShEx - Python implementation of Shape Expressions (Version v0.4.2), April 2018. <https://doi.org/10.5281/zenodo.1214189>
23. Staworko, S., Boneva, I., Labra Gayo, J.E., Hym, S., Prud'hommeaux, E.G., Solbrig, H.R.: Complexity and expressiveness of ShEx for RDF. In: 18th International Conference on Database Theory, ICDT 2015. LIPIcs, vol. 31, pp. 195–211. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2015)
24. Taraborelli, D., Dugan, J.M., Pintscher, L., Mietchen, D., Neylon, C.: WikiCite 2016 Report, November 2016. https://upload.wikimedia.org/wikipedia/commons/2/2b/WikiCite_2016_report.pdf
25. Thornton, K., Cochrane, E., Ledoux, T., Caron, B., Wilson, C.: Modeling the domain of digital preservation in Wikidata. In: iPRES 2017: 14th International Conference on Digital Preservation (2017)
26. Thornton, K., Seals-Nutt, K., Cochrane, E., Wilson, C.: Wikidata for digital preservation (2018). <https://doi.org/10.5281/zenodo.1214319>
27. Vrandečić, D.: Wikidata: a new platform for collaborative data collection. In: Proceedings of the 21st International Conference Companion on World Wide Web, pp. 1063–1064. ACM (2012)
28. Wikidata: Datamodel (2015). <https://www.mediawiki.org/wiki/Wikibase/DataModel>
29. Yosemite: About the Yosemite Project (2013). <http://yosemiteproject.org>

Open Access This chapter is licensed under the terms of the Creative Commons Attribution 4.0 International License (<http://creativecommons.org/licenses/by/4.0/>), which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons license and indicate if changes were made.

The images or other third party material in this chapter are included in the chapter's Creative Commons license, unless indicated otherwise in a credit line to the material. If material is not included in the chapter's Creative Commons license and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder.

