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

Katherine Thornton[1], Harold Solbrig[2], Gregory S. Stupp[3], Jose Emilio Labra Gayo[4], Daniel Mietchen[5], Eric Prud'hommeaux[6], and Andra Waagmeester[7]

[1] Yale University, New Haven, CT, USA
katherine.thornton@yale.edu
[2] Mayo Clinic, College of Medicine, Rochester, MN, USA
Solbrig.Harold@mayo.edu
[3] The Scripps Research Institute, San Diego, CA, USA
gstupp@scripps.edu
[4] University of 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. Three 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

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 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. We end by discussing how ShEx validation of Wikidata subgraphs about software and file formats enables new approaches to digital preservation of software and data, including Emulation as a Service.

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, it 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 contributers 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. [24] 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 XML[8] and JSON[9][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 Javascrpt/N3.js The shex.js [10] Javascript implementation of ShEx was used to develop the ShEx language and test suite [11] and is generally used as a proving ground for language extensions. It was used to develop GeneWiki [12], WikiCite [13] and FHIR/RDF schemas[21]. The online validator[14] was used to develop and experiment with all of these schemas. In addition, the FHIR/RDF document production pipeline used a REST interface, and the GeneWiki and WikiCite projects used its command line interface to invoke it in node.js. The development of the GeneWiki schemas uses several branches of shex.js that are aggregated into a single "wikidata" branch [15].

Shaclex Shaclex[16] is a Scala implementation of ShEx and SHACL. The library uses a purely functional approach where the validation is defined using monads

[8] http://w3.org/brief/NTAx
[9] http://w3.org/brief/NTAy
[10] http://github.com/shexSpec/shex.js
[11] https://github.com/shexSpec/shexTest
[12] `https://github.com/SuLab/Genewiki-ShEx`
[13] `https://github.com/shexSpec/schemas/tree/master/Wikidata/wikicite`
[14] https://rawgit.com/shexSpec/shex.js/master/doc/shex-simple.html
[15] https://rawgit.com/shexSpec/shex.js/wikidata/doc/shex-simple.html
[16] http://labra.weso.es/shaclex/

and monad transformers[11]. The validator is defined in terms of a simple RDF interface (SRDF) that has several implementations. Two implementations are based on RDF models that can be created using Apache Jena[17] or RDF4J[18]. 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 PyShEx[19] is a Python 3 implementation of the ShEx[20] specification. It uses the underlying model behind the ShEx JSON format (ShExJ)[21] as the abstract syntax tree (AST), meaning that ShEx schemas in the JSON format can be directly loaded and processed. PyShEx uses the PyShExC parser[22] to transform ShEx compact format (ShExC) schemas into the same target AST. PyShEx is based on the native Python RDF library – the rdflib[23] package – meaning that it can support a wide variety of RDF formats. PyShEx can also use the sparql_slurper[24] package to fetch sets of triples on demand from a SPARQL endpoint. An example of PyShEx can be found at https://tinyurl.com/ycuhblog.

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.

[17] https://jena.apache.org/
[18] http://rdf4j.org/
[19] https://github.com/hsolbrig/PyShEx
[20] http://shex.io/shex-semantics/
[21] https://github.com/hsolbrig/ShExJSG
[22] https://github.com/shexSpec/grammar/tree/master/parsers/python
[23] http://rdflib.readthedocs.io/en/stable/
[24] https://github.com/hsolbrig/sparql_slurper

3.1 Domain-Specific ShEx Validation in Medical Informatics

The Yosemite Project[31] 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"[25], 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 were to be represented in XML[26] and JSON[27]. One of the outcomes of the Yosemite project was the formation of the FHIR RDF / Health Care Life Sciences (FHIR/HCLS) working group[28] 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 of 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 the 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 caught multiple errors in both the transformation software and uncovered a number of additional issues in the specification

[25] http://yosemitemanifesto.org/
[26] http://hl7.org/fhir/xml.html
[27] http://hl7.org/fhir/json.html
[28] https://www.w3.org/community/hclscg/

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 Wikipedia, 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 individual 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 facilitate 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://www.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 [29]. 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 is the item for *JPEG File Interchange Format, version 1.02*. Properties like *instance of* (*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 to specific scientific communities, e.g. Wikigenomes [18] for curating microbial genomes, WikiDP for digital preservation of software[27], or Scholia [13] for exploring scholarly publications. Through such tools, communities can engage with the Wikidata RDF graph that are not active on Wikidata itself. ShEx can facilitate that.

Non-ShEx Validation Workflows for Wikidata Wikidata uses constraints and 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 templates [29], 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 them[30]. This template-based validation infrastructure, while still largely functional, has been superseded by a parallel one that has been built later by having dedicated properties[31] 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 extensions[32], 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.

Many of the tools that are used to interact with Wikidata have elaborate mechanisms to validate the data, and users can of course query the data in multiple ways to perform specific quality checks. Here, it is helpful that Wikidata's SPARQL endpoint provides a range of visualizations. Problems with geolocation

[29] https://www.wikidata.org/wiki/Category:Constraint_templates
[30] https://www.wikidata.org/wiki/Wikidata:Database_reports/Constraint_violations
[31] https://www.wikidata.org/wiki/Help:Property_constraints_portal
[32] https://www.mediawiki.org/wiki/Wikibase_Quality_Extensions

data of train stations, for instance, can then sometimes simply be inferred from some of them being plotted in the sea[33].

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 ShEx[34]. 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).

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 $\tilde{2}$4k human and 20k mouse genes from NCBI Gene [35], 8,700 disease concepts from the Disease Ontology [36], and 2,700 FDA-approved drugs.

[33] https://twitter.com/piecesofuk/status/979292229159317504
[34] https://www.wikidata.org/wiki/Wikidata:WikiProject_ShEx
[35] `https://www.ncbi.nlm.nih.gov/gene/`
[36] `http://disease-ontology.org/`

The Gene Wiki bots are built using a Python framework called the Wikidata Integrator (WDI) [37]. 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 regularly [38] .

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.

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.

[37] https://github.com/SuLab/WikidataIntegrator
[38] http://jenkins.sulab.org

Wikidata's WikiProject Informatics[39] collaboratively models the domain of computing [26]. Until now, members of the Wikidata community have created items for more than 60,000 software titles[40] and more than 2,000 file formats[41].

Schemas for software items[42] and file format items[43] in Wikidata have been created and entity data was tested using the ShEx2 Simple Online Validator[44]. In order to use ShEx, we created manifests for software items[45] and file format items[46]. 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.

4.3 Bibliographic Metadata

WikiCite is an effort to collect bibliographic information in Wikidata[25]. 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 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 publications[47], 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

[39] https://www.wikidata.org/wiki/Wikidata:WikiProject_Informatics
[40] `https://github.com/emulatingkat/SPARQL/blob/master/software/softwareCount.rq`
[41] `https://github.com/emulatingkat/SPARQL/blob/master/fileFormat/ffCount.rq`
[42] https://github.com/shexSpec/schemas/blob/master/Wikidata/DigitalPreservation/wikidataSoftware.shex
[43] https://github.com/shexSpec/schemas/blob/master/Wikidata/DigitalPreservation/wikidataFileFormat.shex
[44] https://rawgit.com/shexSpec/shex.js/master/doc/shex-simple.html
[45] `https://github.com/shexSpec/schemas/blob/master/Wikidata/DigitalPreservation/manifest_all_software.json`
[46] `https://github.com/shexSpec/schemas/blob/master/Wikidata/DigitalPreservation/manifest_all.json`
[47] http://wikicite.org/statistics.html

of the literature, e.g. on a specific topic. One such literature corpus is that about the Zika virus[48]. In this context, a ShEx manifest has been drafted[49] 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 Using ShEx-Validated Wikidata Subgraphs in Digital Preservation

Digital Preservation is the set of practices that we undertake to ensure continued access to digital objects. Within digital preservation, we need to be able to unambiguously refer to resources in the domain of computing in order to characterize digital objects, plan preservation actions, and automate workflows. The Wikidata QIDs for software and file formats described above are unique URIs and can be used to identify these resources in digital preservation workflows.

Digital Preservation is an expensive activity for many institutions. Irrespective of their budgets for digital preservation, they can reuse content from Wikidata at no cost. The boundary infrastructure [23] of Wikidata provides digital preservation professionals from around the world, working in their own languages, with the means to collaborate by creating structured data in the knowledge base. This reduces the risk of redundant efforts to describe the same file format in numerous local format registries. The boundary infrastructure of the knowledge base also provides visibility and supports contributions from the crowd, i.e. people who have an interest in, and information about the domain of computing. This allows for collaborations that might not happen without the boundary infrastructure that facilitates communication in a community of practice.

Members of the general public will also have access to this information. Having this information in an accessible, structured repository will allow more people to consult it, which could lead to people making different computing choices in their lives, for example choosing an open format, which could impact the work of future generations of digital preservation professionals.

Wikidata's CC0 license ensures that this data will have an equalizing force, as it will not be controlled by any single institution, or even any consortium of institutions. Anyone with access to the internet will be able to inspect and reuse this data for their own systems.

[48] https://www.wikidata.org/wiki/Wikidata:WikiProject_Zika_Corpus
[49] https://github.com/shexSpec/schemas/blob/master/Wikidata/wikicite/Zika%20Corpus/zika_corpus.shex

5.1 Emulation as a Service

Some legacy software titles are no longer available for our inspection or use. Other pieces of legacy software may still be available, but they can be challenging to install and configure, and rapidly become unusable when the appropriate operating system is no longer supported on contemporary hardware. Software emulation technologies simulate older computer hardware. This allows us to provide emulated computing environments in which to run the legacy software.

The structured data in the Wikidata knowledge base plays a key role in the EaaS platform. The EaaS platform hosts pre-configured emulated environments that consist of legacy operating systems and combinations of software applications. EaaS also hosts an object library of software titles or software bundles that can be used within one or more of the base environments. The EaaS platform uses the relationships encoded by Wikidata property *readable file format* (P1072) to provide a list of options to present to the user of base environments that are compatible with the object they have selected.

We syndicate data from Wikidata in our Emulation as a Service platform. Wikidata editors contribute data using the properties available. Not all editors align their data models with one another before contributing data. In some cases, it is possible to express the same or similar statements using different sets of properties and qualifiers. Validating entity data from Wikidata using a ShEx schema allows to easily discover different modeling practices within the knowledge base. The reports generated within the validation workflows allow to quickly identify areas for additional curation work. This information supports our articulation work, meaning we can provision work tasks among members of our distributed project team.

One potential application of Emulation as a Service is in providing access to legacy data. In addition to data models of relevant software and file formats, this routinely requires handling of legacy physical storage media and associated legacy equipment. In the case of a recent analysis of data from the Apollo Space Program of the late 1960s and early 1970s, this involved "magnetic tape, microfilm, microfiche, or hard-copy document"[30]. Wikidata has some basic coverage of hardware associated with such legacy storage methods, as well as of datasets, but the data models are inconsistent across similar items. ShEx could help establish an infrastructure for sharing data models, building on the efforts that have gone into the curation of software and file formats.

6 Discussion

6.1 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 – it moved RDF from a topic of discussion to 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 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 [28]. 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.

6.2 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 W3C[50] maintains a repository of ShEx schemas[51] 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.

6.3 Soundness and Quality

[2] provides efficient validation algorithms and verifies the soundness of recursion. [24] identifies the complexity and expressive power of ShEx. The comprehensive ShEx test suite[52] 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

[50] https://www.w3.org/community/shex/
[51] https://github.com/shexSpec/schemas
[52] `https://github.com/shexSpec/shexTest`

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.

6.4 Impact

The ShEx Specification is available under the W3C Community Contributor License Agreement[53]. In addition to the specification itself, the ShEx community also created a Primer[54] 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.

7 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.

[53] https://www.w3.org/community/about/agreements/cla/
[54] http://shex.io/shex-primer/

8 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.

References

1. Bielefeldt, A., Gonsior, J., Krötzsch, M.: Practical linked data access via SPARQL: the case of wikidata. In: Proceedings of the WWW2018 Workshop on Linked Data on the Web (LDOW-18). 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., Waagmeester, A., Mitraka, E., Turner, J., Putman, T., Leong, J., Naik, C., Pavlidis, P., Schriml, L., Good, B.M., Su, A.I.: 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., Orozco, C., Goodale, J., Wu, C., Batalov, S., Vickers, T.J., Valafar, F., Su, A.I.: A gene wiki for community annotation of gene function. PLoS Biol. 6(7), e175 (Jul 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. pp. 14:1–14:5. OpenSym '17, ACM, New York, NY, USA (2017), `https://doi.org/10.1145/3125433.3125465`
8. Knublauch, H., Kontokostas, D.: Shapes Constraint Language (SHACL). W3C Recommendation (Jun 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 (2017)
10. Labra Gayo, Jose Emilio : SHACLex: Scala implementation of ShEx and SHACL (Apr 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. pp. 333–343. POPL '95, ACM, New York, NY, USA (1995), `http://doi.acm.org/10.1145/199448.199528`
12. Mitraka, 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: The Semantic Web: ESWC 2017 Satellite Events. pp. 237–259. Springer International Publishing, Cham (2017)
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., tombaker, Glenna, Labra Gayo, J.E., mrolympia, Waagmeester, A., Werkmeister, L., Booth, D.: shexSpec/shex.js: Release for zenodo DOI (Version v0.9.2) (Apr 2018), http://doi.org/10.5281/zenodo.1213693
18. Putman, T.E., Lelong, S., Burgstaller-Muehlbacher, S., Waagmeester, A., Diesh, C., Dunn, N., Munoz-Torres, M., Stupp, G.S., Wu, C., Su, A.I., Good, B.M.: Wikigenomes: an open web application for community consumption and curation of gene annotation data in wikidata. Database 2017, bax025 (2017), http://dx.doi.org/10.1093/database/bax025
19. Sáez, T., Hogan, A.: Automatically generating wikipedia info-boxes from wikidata. In: WWW '18 Companion: The 2018 Web Conference Companion, April 23–27, 2018, Lyon, France. 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., Prud'hommeaux, E., Grieve, G., McKenzie, L., Mandel, J.C., Sharma, D.K., Jiang, G.: Modeling and validating HL7 FHIR profiles using semantic web Shape Expressions (ShEx). J Biomed Inform 67, 90–100 (03 2017)
22. Solbrig, H.: PyShEx - Python implementation of Shape Expressions (Version v0.4.2) (Apr 2018), http://doi.org/10.5281/zenodo.1214189
23. Star, S.L.: This is not a boundary object: Reflections on the origin of a concept. Science, Technology, & Human Values 35(5), 601–617 (2010)
24. 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)
25. 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
26. Thornton, K., Cochrane, E., Ledoux, T., Caron, B., Wilson, C.: Modeling the Domain of Digital Preservation in Wikidata. iPRES 2017: 14th International Conference on Digital Preservation (2017)
27. Thornton, K., Seals-Nutt, K., Cochrane, E., Wilson, C.: Wikidata for digital preservation (2018), http://doi.org/10.5281/zenodo.1214319
28. 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)
29. Wikidata: Datamodel (2015), https://www.mediawiki.org/wiki/Wikibase/DataModel
30. Williams, D.R., Hills, H.K., Taylor, P.T., Grayzeck, E.J., Guinness, E.A.: Restoration of Apollo Data by the Lunar Data Project / PDS Lunar Data Node: An Update. In: Lunar and Planetary Science Conference. Lunar and Planetary Science Conference, vol. 47, p. 2385 (Mar 2016)
31. Yosemite: About the yosemite project (2013), http://yosemiteproject.org