SHACL and ShEx in the Wild: A Community Survey on Validating Shapes Generation and Adoption

Kashif Rabbani
Aalborg University
Denmark
kashifrabbani@cs.aau.dk

Matteo Lissandrini
Aalborg University
Denmark
matteo@cs.aau.dk

Katja Hose
Aalborg University
Denmark
khose@cs.aau.dk

ABSTRACT

Knowledge Graphs (KGs) are widely used to represent heterogeneous domain knowledge on the Web and within organizations. Various methods exist to manage KGs and ensure the quality of their data. Among these, the Shapes Constraint Language (SHACL) and the Shapes Expression Language (ShEx) are the two state-of-the-art languages to define validating shapes for KGs. Since the usage of these constraint languages has recently increased, new needs arose. One such need is to enable the efficient generation of these shapes. Yet, since these languages are relatively new, we witness a lack of understanding of how they are effectively employed for existing KGs. Therefore, in this work, we answer *How validating shapes are being generated and adopted?* Our contribution is threefold. First, we conducted a community survey to analyze the needs of users (both from industry and academia) generating validating shapes. Then, we cross-referenced our results with an extensive survey of the existing tools and their features. Finally, we investigated how existing automatic shape extraction approaches work in practice on real, large KGs. *Our analysis shows the need for developing semi-automatic methods that can help users generate shapes from large KGs.*

CCS CONCEPTS

• **Information systems → Graph-based database models**.

KEYWORDS

Knowledge Graphs, SHACL, ShEx, Shapes Extraction

ACM Reference Format:
Kashif Rabbani, Matteo Lissandrini, and Katja Hose. 2022. SHACL and ShEx in the Wild: A Community Survey on Validating Shapes Generation and Adoption . In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3487553.3524253

1 INTRODUCTION

The popularity of Knowledge Graphs (KGs) has consistently grown in the last decade due to the advent of public KGs, such as DBpedia [4], YAGO [26], and WikiData [28] and their adoption among companies [16]. These KGs (represented in RDF) are massive, diverse, and most importantly incomplete due to the evolving nature of human knowledge. Given the importance of KGs, it is paramount to ensure the quality of the information they represent. Validating languages, i.e., the shapes constraints languages (SHACL [13] and ShEx [19]) are used to ensure the data quality in KGs by defining integrity constraints in the form of shapes, i.e., high-level structural constraints for entities within a KG. While SHACL and ShEx differ at a syntactic level, they both allow to specify a set of characterizing properties and attributes for classes of entities that are expected to be hold within a KG [21]. For instance, they both allow to specify that all entities of type "Student" need to have "name" and "registration number" as properties, and be linked to at least one "Course". Shapes also specify data types for these properties, e.g., *String*, *Integer*, or *IRI*. Shapes can also be used for purposes other than validation, such as to design user interfaces [9, 29], or optimize query processing [21].

There exist various tools to help define validating shapes, either manually or semi-automatically, such as Astrea [6], TopBraid Composer [3], the SHACLGEN [2] python library, ShapeDesigner [5] and SheXer [8] to semi-automatically generate SHACL and ShEx from ontologies and KGs data, as well as approaches to define shapes using profiling [15] and ontology design patterns [18]. *Naturally, when a KG contains hundreds of classes, each having many attributes, manually specifying (post-hoc) the necessary shapes becomes a tedious and unmanageable task.* On the other hand, the approaches helping automatic generation of validating shapes produce a large number of shape constraints (for nodes and properties) such that it becomes non-trivial to verify the validity of the generated constraints.

To better understand the needs of users, both in industry and academia, to generate shapes for large KGs, we conducted an online community survey with a set of questions to learn how (and up to what extent) they generate and use shapes. We used Google Forms to create an online survey[1] and shared it with researcher and practitioners across different communities, e.g., within members of the W3C mailing list[2], the Solid project[3], the BlueBrainNexus[4] project, the Bayer-Group COLID team[5], and Slack channels[6] for Knowledge Graph and ISWC conferences. Additionally, we have contacted other related Semantic Web communities developing various tools and approaches to deal with validating shapes. When we asked users to participate in our survey, they all manifested great interest in the results of this study. Hence, we promised to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France.
© 2022 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-1-4503-9130-6/22/04…$15.00
https://doi.org/10.1145/3487553.3524253

[1] https://forms.gle/93KFZH5vcGy7Et27A
[2] https://lists.w3.org/Archives/Public/semantic-web/
[3] https://gitter.im/solid/
[4] https://bluebrainnexus.io
[5] https://github.com/Bayer-Group/COLID-Documentation
[6] https://knowledgegraphconf.slack.com, https://iswc-conf.slack.com

re-share our findings with the community. This work is also a way to keep up with that promise. Therefore, in the following *we first report our findings by analyzing the results obtained from our online community survey*, then *we discuss the state of the art in validating shapes*. Finally, *we report on some experimental results in actually using such tools*. Our analysis highlights a number of *gaps when it comes to generating shapes automatically* to validate existing (possibly noisy or erroneous) KGs, thus we discuss a number of future research directions based on our findings.

2 COMMUNITY SURVEY

This section summarizes the questions and answers to our survey. The survey contained nine questions and received in total 30 answers. The survey was conducted online, and the results were collected anonymously between November 2021 and January 2022. Among the questions, we surveyed also the area of occupation of the respondents, i.e., if they belong to Academia, Industry, or both. Answers to this question showed that 53% of the respondents are from Industry, 27% from Academia, and 20% from both (see Figure 2a). In the following, where relevant, we will report for each statistics, in parenthesis, the split of answers by respondents from academia, industry, or both respectively.

Our main question asked *"how the validating shapes were being generated"* in general. We provided three answer options and allowed the respondents to also add a free text answer. Answers to this question are presented in Figure 1, where numbers in circles represent the number of participants. The results show that most of the respondents (i.e., 26 - split 6/14/6) generate shapes manually, while 13 respondents (split 5/5/3) generate shapes from existing ontologies and 7 respondents (split 1/4/2) generate shapes from RDF graphs instance data, while only 1 respondent declared to use "RDF forms". The results overlap as it was a multi-choice question where respondents had chosen more than one option.

We then asked *which tools or methods the respondents used in practice* and provided a list of state-of-the-art tools and approaches that can help in generating shapes (see Table 1) along with a free text answer option as well. To our surprise, respondents use a very heterogeneous set of tools and methods, with the most used being TopBraid Composer [3] used by 33% (10) of respondents (split 1/7/2), Protégé was used by 20% (6) respondents (split 3/3/0), RDFShape [1] by another 16.7% (5) respondents (split 2/1/2), SheXer [8] was used by 10% (3) respondents (split 0/1/2), and text editors are used by 16.7% (5) respondents (split 2/4/0). The rest of the tools are used by 1 or 2 respondents on average. In addition to that, one of the respondents mentioned that they generate shapes by applying custom rules via Python scripts, another by using tabular formats (like Excel) to curate shapes manually. Lastly, one respondent commented: "I am looking for a suitable tool".

We further asked a number of questions related to the characteristics of the graphs for which the respondents were generating shapes. Results showed that 38% of respondents generate shapes from RDF graphs containing up to 100K triples (split 4/5/1), 17% use graphs having 100K to 1 million triples (split 0/5/0), and 45% use graphs containing more than 1 million triples (split 4/6/3), see Figure 2a and 2b. The great majority 48% (14) of KGs are described by ontologies containing between 10 and 1000 classes and 10-50

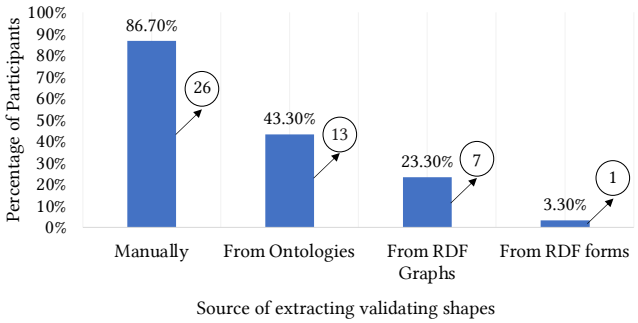

Figure 1: Analysis on extraction of validating shapes

distinct properties (see Figure 2c and 2d). While 32% (9) of the respondents generate up to 10 shapes, 47% (13) generate between 10-100 shapes, and 14% (4) generate more than 100 shapes and in some cases even more than 1000 shapes (see Figure 2e).

In addition to that, 55% (16) of the respondents generate shapes for the entire graph, and 43% (13) of them generate shapes for specific portions of the graph (see Figure 2f) due to various reasons (answers to our last question). These reasons include (i) use case specific requirements to focus on shapes targeting only the important classes of the graph (10%), (ii) interest or requirement to get a specific maintained view of the data graph or subset (30%), (ii) limitation of the known business rules to apply only to part of the graph (10%), (iv) the scalability of existing methods to generate shapes is insufficient for very large graphs (10%), (v) the decision to generate shapes progressively (such as for WikiData) to accommodate evolution through time and change in requirements (30%), and (vi) the requirement of validating only non-logical statements that cannot be validated using (already existing) OWL constraints (15%).

The results show that *most users work with fairly large graphs with tens or hundreds of classes and thousands or millions of triples*. Yet, *most users still generate shapes manually*; this way, they can generate only a handful of shapes (not enough to cover the entire graph). Thus, there is also the need for tools in supporting the scalable generation of shapes for very large graphs, which should allow to extract shapes for only a specific portion of a KG. Therefore, we conducted an extensive analysis of tools used to generate validating shapes and present our findings in the next section to better understand the capabilities offered by existing approaches when it comes to helping users generate shapes more efficiently.

3 STATE OF THE ART

Integrity constraints over KGs were initially defined using OWL[27]. Later on, the SPARQL Inferencing Notation (SPIN) [12], a SPARQL-based rule and constraint language was introduced to enforce constraints over KGs using SPARQL queries. SHACL [13] is known as the next generation of SPIN and has become a W3C recommended language in 2017. Similar to SHACL, ShEx [20] is a constraint language built on regular bag expressions inspired by schema languages for XML. Validating schemas have been adopted for type and compliance checking [14, 22, 23], as well as purposes other than validation as well. Specifically, for optimizing query processing in KGs [21], building and automatically generating forms to populate RDF datasets [9, 29], intelligent geoprocessing [10], and automatic detection of metadata errors in clinical studies [11].

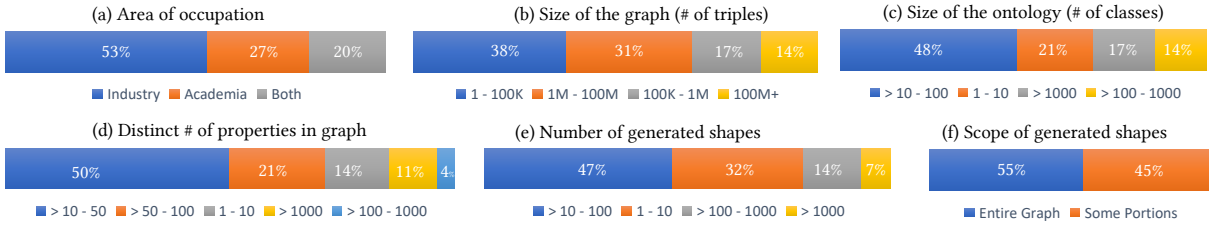

Figure 2: Survey Analysis: Answers to statistical questions

Shape Extraction: There has been recent development to derive validating shapes from existing KG instance data; various applications are created to assist the process of shapes extraction, which can potentially be divided into two phases, i.e., preprocessing and shapes construction. The *preprocessing* phase involves collecting information from the input dataset about classes and their properties, which are then used in the *shapes construction* phase. The preprocessing phase can be conducted in two alternative ways: query-based and non-query-based. The query-based solutions involve loading the KG into a triplestore able to answer SPARQL queries. The triplestore is then used to access the information required for the shapes construction phase. SHACLGEN [2], RDFShape [1], and ShapeDesigner [5] are query-based solutions to generate SHACL or ShEx shapes from existing RDF data. Non-query-based solutions, instead, parse RDF data stored as files on disk. SheXer [8] is the only tool to generate ShEx shapes that supports both a triplestore and RDF files as input. Spahiu et al. [24] proposed a solution based on a data profiling tool called ABSTAT [25] to generate semantic profiles and transform them into SHACL to improve the quality of the KGs. Mihindukulasooriya et al. [15] proposed a KG data profiling-based RDF shape induction approach by using predictive modeling to generate shapes. SHACLearner [17] is a method to learn SHACL constraints based on Inverse Open Path rules (IOP) rules. Astrea [6] is an ontology-based approach to extract SHACL from ontologies. Finally, RML2SHACL [7] also generates SHACL shapes but it requires RML mappings. We have classified existing approaches in Table 1 based on their features (expanded from a previous short survey [6]), i.e., support for shapes extraction from data or ontologies, support for automatic extraction of shapes, support for shapes extraction from a SPARQL triplestore, and whether they extract SHACL, ShEx, or both types of validating shapes. *Overall, as we show later, we note that shapes extracted by these methods are often incomplete, i.e., they do not implement functionalities to generate all types of shape constraints*, e.g., they often do not produce sh:class and cardinality constraints.

Table 1: State-of-the-art to extract validating shapes

Approach	*Extracted from*		*Auto-matic*	*Triple-store*	*Type*
	data	*ontology*			
Shape Induction [15]	✔	✗	✔	✔	SHACL,ShEx
SheXer [8]	✔	✗	✔	✔	SHACL,ShEx
Spahiu et al. [24]	✔	✗	✔	✔	SHACL
ShapeDesigner. [5]	✔	✗	✔	✔	SHACL,ShEx
SHACLGEN [2]	✔	✔	✔	✔	SHACL
TopBraid [3]	✔	✔	✔	✔	SHACL
Pandit et al. [18]	✗	✔	✗	✔	SHACL
Astrea [6]	✗	✔	✔	✗	SHACL
SHACLearner [17]	✔	✗	✔	✗	SHACL

4 LIMITATIONS AND OPPORTUNITIES

In light of the results presented above, this section highlights the existing gaps between the capabilities of current tools and the real user needs. After this analysis, we present a set of important research directions for automatic shape extraction.

Among the existing approaches, some tools support extraction of validating shapes from ontologies only (such as Astrea [6]), others support extraction from instance data, and very few support automatic generation of validating shapes from both ontology and instance data (see Table 1). Methods that generate shapes from ontologies assume the provided ontology to be complete (i.e., providing complete coverage of the instances in the KG and their properties) and of small size (they do not expect more than a few hundred classes). Approaches that extract shapes from RDF instance data assume that the KG is particularly small (in terms of the number of instances and classes) or already available in a triplestore. When this is not the case, they load it into an in-memory triplestore, which is problematic for large graphs, as further discussed below. We believe these assumptions clash with a number of real use cases. As a matter of fact, in the results of our survey, we see that, despite the existence of tools and approaches to extract validating shapes automatically, *most users are still generating shapes manually*. To understand this better, we ran some experiments to find out the real capabilities *of existing tools for automatically extracting shapes from RDF graphs*. The experiments are performed using state-of-the-art shapes extraction tools and approaches such as SheXer [8], ShapeDesigner [5], and SHACLGEN [2]. All experiments are performed on a machine with 24 cores and 256GB of RAM using a 2021 dump of DBpedia [4], YAGO-4 [26], and a version of LUBM scaled to 91M triples. More details on our experiments and datasets are available online[7]. We tried similar experiments for WikiData [28] as well, but the initial results show that existing methods are not able to handle its scale (they exhausted all memory or did not manage to produce an output in several days). *Therefore, we highlight that more research is needed to design scalable methods to extract validating shapes from large KGs.*

Our first finding from these experiments is that some tools are capable of handling only relatively small KGs. In particular ShapeDesigner [5] crashed with datasets with a few millions triples, while SHACLGEN [2] is not suited to extract shapes of datasets having hundreds of classes as it required days to generate shapes for YAGO-4 (8,897 classes). Furthermore, we note that both ShapeDesigner [5] and SHACLGEN [2] load the whole graph into a triplestore to generate shapes. Yet, extracting shapes by querying a

[7] https://relweb.cs.aau.dk/validatingshapes/
Zenodo: https://doi.org/10.5281/zenodo.5958985

triplestore is particularly inefficient when extracting shapes for the entire graph. Compared to parsing and analyzing a file, which can exploit optimized batch processing over multiple scans of the data, extracting shapes when the KG is in a triplestore requires to run queries for each target class to extract all necessary information. When processing RDF files directly, SheXer [8] provides better performance and was able to extract shapes from DBpedia consuming a maximum of 18GB of RAM in 26 minutes (LUBM: 33GB RAM in 58 minutes, YAGO-4: 24GB RAM in 117 minutes).

Our second finding relates to the usefulness and reliability of the shapes extracted from the instance data. Shapes are useful only if they are complete (containing all the required constraints to validate the input graph) and reliable if they do not contain constraints representing spurious data. Our analysis (via manual inspection) of the shapes extracted by these tools revealed that *none of the approaches extracts all the required constraints* for property shapes; for instance, we did not find any constraint for non-literal predicate objects (e.g., to indicate that objects for "takes course" should be of type "Course"). Furthermore, we saw that some property shapes were extracted because of the presence of some nodes with spurious or erroneous predicates (e.g., a city mistaken as a member of a musical band). To further investigate these issues, we implemented a Java application[7] to extract also the missing shape constraints and report their support in the graph, e.g., how many entities were satisfying a specific constraint. *We found that generating automatically the complete set of shapes would produce hundreds or thousands of node and property shapes* each having multiple constraints. Yet, since the source KGs are naturally noisy and incomplete, *those shapes and constraints are often unreliable.* In particular, for DBpedia, we extracted 426 node shapes and 11,916 property shapes, which have 38,454 non-literal and 5,335 literal constraints. Similarly, for YAGO-4, we extracted 8,897 node shapes and 76,765 property shapes, with a total of 315,413 non-literal and 50,708 literal constraints. We have published the extracted SHACL shapes on Zenodo[7]. Given the above numbers of extracted constraints for DBpedia and YAGO-4, it is non-trivial to manually establish the validity and the usefulness of thousands of automatically generated shape constraints. The only existing approach in this direction is SheXer [8], which supports filtering of shapes based on a "trustworthiness" score (even though we found this score does not directly translate into how frequently a shape is satisfied in a dataset, and thus it is hard to tune).

In conclusion, referring to the results of our community survey (Figure 1), we see a connection between the limitations of existing shapes extraction approaches in effectively supporting the real user needs and the (for now) common practice of generating shapes manually. Since validating shapes have become an essential tool to ensure the quality and completeness of large KGs in many organizations, these results suggest that *more research is needed to help users generate useful validating shapes for existing large KGs*, considering both the scalability of the approach and the informativeness and utility of the extracted shapes. This will help users curate and maintain high-quality large-scale KGs.

ACKNOWLEDGMENTS

This research was partially funded by the Danish Council for Independent Research (DFF) under grant agreement no. DFF-8048-00051B, the EU's H2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 838216, and the Poul Due Jensen Foundation.

REFERENCES

[1] 2022. RDFShape. http://rdfshape.weso.es. Accessed 20th January.
[2] 2022. SHACLGEN. https://pypi.org/project/shaclgen/. Accessed 20th January.
[3] 2022. TopBraid. https://www.topquadrant.com/products/topbraid-composer/. Accessed 20th January.
[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*. 722–735.
[5] Iovka Boneva, Jérémie Dusart, Daniel Fernández Alvarez, and Jose Emilio Labra Gayo. 2019. Shape designer for ShEx and SHACL constraints. In *ISWC 2019*.
[6] Andrea Cimmino, Alba Fernández-Izquierdo, and Raúl García-Castro. 2020. Astrea: Automatic Generation of SHACL Shapes from Ontologies. In *ESWC*. 497–513.
[7] Thomas Delva, Birte De Smedt, Sitt Min Oo, Dylan Van Assche, Sven Lieber, and Anastasia Dimou. 2021. RML2SHACL: RDF Generation Taking Shape. In *Proceedings of the 11th on Knowledge Capture Conference*. 153–160.
[8] Daniel Fernandez-Álvarez, Jose Emilio Labra-Gayo, and Daniel Gayo-Avello. 2022. Automatic extraction of shapes using sheXer. *Knowledge-Based Systems* 238 (2022), 107975.
[9] Aidan Hogan. 2020. Book. In *The Web of Data*. Springer, Cham.
[10] Zhi-Wei Hou, Cheng-Zhi Qin, A Zhu, Yi-Jie Wang, Peng Liang, Yu-Jing Wang, Yun-Qiang Zhu, et al. 2021. Formalizing Parameter Constraints to Support Intelligent Geoprocessing: A SHACL-Based Method. *ISPRS International Journal of Geo-Information* 10, 9 (2021), 605.
[11] Daniel Keuchel and Nicolai Spicher. 2021. Automatic Detection of Metadata Errors in a Registry of Clinical Studies Using Shapes Constraint Language (SHACL) Graphs. *Studies in health technology and informatics* 281 (2021), 372–376.
[12] Holger Knublauch, James A Hendler, and Kingsley Idehen. 2011. SPIN-overview and motivation. *W3C Member Submission* 22 (2011), W3C.
[13] Holger Knublauch and Dimitris Kontokostas. 2017. Shapes constraint language (SHACL). *W3C Candidate Recommendation* 11, 8 (2017).
[14] Martin Leinberger, Philipp Seifer, Claudia Schon, Ralf Lämmel, and Steffen Staab. 2019. Type checking program code using SHACL. In *ISWC*. 399–417.
[15] Nandana Mihindukulasooriya, Mohammad Rifat Ahmmad Rashid, Giuseppe Rizzo, Raúl García-Castro, Oscar Corcho, and Marco Torchiano. 2018. RDF shape induction using knowledge base profiling. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. 1952–1959.
[16] Natasha Noy, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. 2019. Industry-scale knowledge graphs: lessons and challenges. *Commun. ACM* 62, 8 (2019), 36–43.
[17] Pouya Ghiasnezhad Omran, Kerry Taylor, Sergio José Rodríguez Méndez, and Armin Haller. 2020. Towards SHACL Learning from Knowledge Graphs.. In *ISWC (Demos/Industry)*. 94–99.
[18] Harshvardhan J Pandit, Declan O'Sullivan, and Dave Lewis. 2018. Using Ontology Design Patterns To Define SHACL Shapes.. In *WOP@ ISWC*. 67–71.
[19] E Prud'hommeaux, J. Emilio L. Gayo, and H. Solbrig. 2014. Shape expressions: an RDF validation and transformation language. In *ICSS*. 32–40.
[20] Eric Prud'hommeaux, Jose Emilio Labra Gayo, and Harold Solbrig. 2014. Shape expressions: an RDF validation and transformation language. In *Proceedings of the 10th International Conference on Semantic Systems*. 32–40.
[21] Kashif Rabbani, Matteo Lissandrini, and Katja Hose. 2021. Optimizing SPARQL Queries using Shape Statistics. In *Proceedings of the 24th International Conference on Extending Database Technology, EDBT*.
[22] Livio Robaldo. 2021. Towards compliance checking in reified I/O logic via SHACL. In *International Conference on Artificial Intelligence and Law*. 215–219.
[23] Umutcan Şimşek, Kevin Angele, Elias Kärle, Oleksandra Panasiuk, and Dieter Fensel. 2020. Domain-specific customization of schema. org based on shacl. In *ISWC*. 585–600.
[24] Blerina Spahiu, Andrea Maurino, and Matteo Palmonari. 2018. Towards Improving the Quality of Knowledge Graphs with Data-driven Ontology Patterns and SHACL.. In *ISWC (Best Workshop Papers)*. 103–117.
[25] Blerina Spahiu, Riccardo Porrini, Matteo Palmonari, Anisa Rula, and Andrea Maurino. 2016. ABSTAT: ontology-driven linked data summaries with pattern minimalization. In *European Semantic Web Conference*. Springer, 381–395.
[26] Thomas Pellissier Tanon, Gerhard Weikum, and Fabian Suchanek. 2020. Yago 4: A reason-able knowledge base. In *ESWC*. 583–596.
[27] Jiao Tao, Evren Sirin, Jie Bao, and Deborah McGuinness. 2010. Integrity constraints in OWL. In *AAAI*, Vol. 24.
[28] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: a free collaborative knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85.
[29] Jesse Wright, Sergio José Rodríguez Méndez, Armin Haller, Kerry Taylor, and Pouya G Omran. 2020. Schimatos: a SHACL-based web-form generator for knowledge graph editing. In *ISWC*. 65–80.