Astrea: automatic generation of SHACL shapes from ontologies

Andrea Cimmino[0000−0002−1823−4484],
Alba Fernández-Izquierdo[0000−0003−2011−3654], and
Raúl García-Castro[0000−0002−0421−452X]

Ontology Engineering Group
Universidad Politécnica de Madrid
`cimmino,albafernandez,rgarcia@fi.upm.es`

Abstract. Knowledge Graphs (KGs) that publish RDF data modelled using ontologies in a wide range of domains have populated the Web. The SHACL language is a W3C recommendation that has been endowed to encode a set of either value or model data restrictions that aim at validating KG data, ensuring data quality. Developing shapes is a complex and time consuming task that is not feasible to achieve manually. This article presents two resources that aim at generating automatically SHACL shapes for a set of ontologies: (1) Astrea-KG, a KG that publishes a set of mappings that encode the equivalent conceptual restrictions among ontology constraint patterns and SHACL constraint patterns, and (2) Astrea, a tool that automatically generates SHACL shapes from a set of ontologies by executing the mappings from the Astrea-KG. These two resources are openly available at Zenodo, GitHub, and a web application. In contrast to other proposals, these resources cover a large number of SHACL restrictions producing both value and model data restrictions, whereas other proposals consider only a limited number of restrictions or focus only on value or model restrictions.

1 Introduction

Knowledge Graphs (KGs) are becoming pervasive on the Web [5]. Since 2014 there is a growing number of KGs from different domains that publish a quite large amount of data using RDF and modelled with ontologies [19]. As a result, in the last decade a considerable effort has been put in developing ontologies for specific domains [21]. Due to the growth of these public available KGs, the W3C has promoted a recommendation called SHACL (Shapes Contraint Language) to validate the RDF graphs [2]. In the last years Ks validation by means of SHACL shapes has gained momentum and has become a relevant research topic [14].

A shape defines a set of restrictions that data from a KG must fulfil. There are two kinds of restrictions [15], those that refer to the data model, e.g., cardinality, and those that apply to the data values, e.g., string patterns. Due to this reason developing shapes has become the cornerstone solution to validate KG data. Nevertheless, developing data shapes is a complex task due to the potential size of the data and all the available restrictions that require a deep domain knowledge (like the one encoded in ontologies); in addition, developing shapes manually is a dull-task and error-prone.

Different proposals to assist shapes generation have been proposed. Some focus on learning shapes from a set of data [16, 7, 22, 1]; these proposals cover a small amount of the restrictions, and most of the learnt restrictions refer to value restrictions. Nevertheless, since KGs are modelled by ontologies, when these proposals learn model restrictions from data they do not take such ontologies into account, leading to a potential discordance with the model. A lower number of proposals aim at aligning the restrictions encoded by OWL constructs with those of SHACL [17, 12]. Unfortunately, these proposals cover a small number of constructs, and do not generate any shapes.

In this paper two resources to generate automatically SHACL shapes [13] from a set of ontologies are introduced. The resources are: A) the Astrea-KG[1] that contains 158 mappings, each of which relates an ontology constraint pattern with an equivalent SHACL constraint pattern; and B) the Astrea[2] tool that automatically generates SHACL shapes for a set of input ontologies by using the mappings provided by the Astrea-KG. The mappings in the Astrea-KG are endowed from a theoretical point of view, presented as ontology and SHACL construct patterns; in addition, the Astrea-KG also contains an implementation as SPARQL CONSTRUCT queries for such mappings. These queries issued over a set of ontologies produce their SHACL shapes, which is the task performed by Astrea.

The shapes generated with the resources presented in this paper contain data and model restrictions, covering 60% of the SHACL available restrictions. Astrea has been evaluated by performing two experiments. The former consists in generating the SHACL shapes of 5 well-known ontologies, such as SAREF or SSN, and two ontologies developed in the context of two European projects. The latter consists in analysing the expressivity and richness of the generated shapes. For the sake of readability, Table 1 shows the prefixes and their associated namespaces that are used through the paper.

Table 1. Summary of the prefixes used through the paper

Prefix	Namespace
sh	http://www.w3.org/ns/shacl#
owl	http://www.w3.org/2002/07/owl#
rdfs	http://www.w3.org/2000/01/rdf-schema#
xsd	http://www.w3.org/2001/XMLSchema#

---

[1] https://astrea.helio.linkeddata.es

[2] https://astrea.linkeddata.es

The rest of this article is structured as follows. Section 2 reports an analysis of some proposals from the literature; Section 3 introduces and details the mappings published in the Astrea-KG; Section 4 reports the implementation of Astrea that produces shapes using the Astrea-KG; Section 5 explains the experiments carried out in this article; finally, Section 6 recaps our findings and conclusions.

2   Related work

The increasing uptake of SHACL shapes as a mechanism for validating RDF data has lead to the proposal of several approaches to assist practitioners in their generation. Approaches can be classified into two types: A) Automatic generation of shapes from data, which aim at learning shapes from a training data set; and B) Analysis of the equivalence between ontology and SHACL restrictions. Table 2 summarises these approaches indicating the source of the shapes and whether they support their automatic generation.

Table 2. Comparison of approaches that deal with shapes generation

Proposal	Extracted from data	Extracted from ontologies	Automatically generated
Mihindukulasooriya et al.	✓	✕	✓
Fernández-Alvarez et al.	✓	✕	✓
Spahiu et al.	✓	✕	✓
Boneva et al.	✓	✕	✓
Pandit et al.	✕	✓	✕
Knublauch	✕	✓	✕
Astrea	✕	✓	✓

✓ supported
✕ not supported

Regarding the approaches oriented to the generation of shapes through data, Mihindukulasooriya et al. [16] aim at using machine learning techniques to produce RDF Shapes. The authors propose a data-driven approach for inducing integrity constraints for RDF data using data profiling, which are then combined into RDF Shapes. Although the proposed approach is defined in a generic way, it is validated using only cardinality and range constraints.

Another work related to the generation of shapes from data is the one presented by Fernández-Alvarez et al. [7], which infers Shape expressions associated to the classes in an RDF graph. This approach consists in the following steps: (1) find all the instances of the target classes; (2) for each class, find all the triples whose subject is one of its instances and use them all to build a profile of the class; and (3) turn each profile into a shape written in ShEx language[3].

The work of Spahiu et al. [22] was also designed to generate shapes from RDF data. It uses semantic profiles, i.e., a summary that provides an abstract but complete description of the dataset content and statistics, of a given RDF graph and translates them into SHACL shapes.

Finally, Boneva et al. [1] presented Shape Designer, a graphical tool for building SHACL or ShEx constraints for an existing RDF graph. Shape Designer

---

[3] https://shex.io/

provides a set of queries and shape patterns that can be selected by the user to generate a shape constraint. Such shape constraint can then be added to the SHACL or ShEx schema under construction.

Concerning the analysis of the equivalence between ontology and SHACL restrictions, the position paper presented by Pandit et al. [17] encourages the reuse of Ontology Design Patterns (ODPs) [8] beyond the data modelling phase to generate SHACL shapes. The authors discuss the similarity that could be obtained between the axioms used to model ODPs and the constraints within SHACL shapes. However, this work does not identify such equivalences between ontologies and SHACL.

To conclude, Knublauch [12] proposes a comparison between OWL and SHACL. This work associates each OWL constraint with its similar SHACL constraint, claiming that a syntactic translation between OWL and SHACL is feasible. Although the author of this work identified similarities between OWL and SHACL, he relates an OWL construct with at most two SHACL constructs. Therefore, it is not taken into account the use of patterns, which hinders the translation.

The resources presented in this paper aim at assisting the automatic generation of SHACL shapes from ontologies, taking into account OWL 2, RDFS, and XSD restrictions. As it is illustrated in Table 2, although there are approaches that deal with shapes extracted from ontologies, only Astrea supports their automatic generation. However, this work is grounded on these previous works that discuss the similarity between OWL and SHACL constraints.

3   Astrea-KG Mappings

The cornerstone element to automatically generate shapes from a set of ontologies are the mappings within the Astrea-KG. These mappings relate one or more ontology construct patterns with the equivalent SHACL construct patterns that validate such ontology construct pattern. However, OWL and SHACL are not considered equivalent in their interpretation. There are differences in how OWL interprets restrictions (for inferencing), and how SHACL interprets constraints (for validation) [12].

The ontology construct patterns include constructs from the well-known OWL 2, RDFS, and XSD specifications. In addition, the mappings have been implemented as SPARQL CONSTRUCT queries in which the WHERE clause contains the ontology construct patterns and the CONSTRUCT clause contains the SHACL construct patterns.

Notice that from a conceptual point of view the mappings are bi-directional, since they relate construct patterns. Nevertheless, their implementation is not bi-directional: the current SPARQL queries identify ontology construct patterns and translate them into the equivalent SHACL construct pattern.

Previous works already stated the similarity between OWL and SHACL constructs. However, during the development of the mappings we noticed that the relation between single constructs was not enough to generate the shapes. The reason is that an ontology construct may be specified within different contexts

and the equivalent SHACL construct may change depending on such context. As a result, the mappings relate patterns of constructs rather than just constructs.

For instance, the RDFS construct *rdfs:range* can be considered equivalent to the SHACL construct *sh:class*. But in order for such equivalence to be correct the *rdfs:range* has to be defined in the context of an *owl:ObjectProperty*. Listing 1.1 shows the SHACL construct pattern of the *sh:class* that is related to the ontology contruct *rdfs:range*. Notice that this SHACL construct makes only sense in the context of a *sh:PropertyShape*.

```
?shapeUrl  a  sh:PropertyShape;
    sh:class  ?rangeURI  .
```

Listing 1.1. SHACL contruct pattern for *sh:class*

Listing 1.2 reports the ontology construct pattern for *rdfs:range* that is equivalent to the one of Listing 1.1. However, the statement *rdfs:range* could be expressed alternatively: instead of having an URL in its range it may have a blank node that has different properties (*owl:unionOf*, *owl:someValuesFrom*, or *owl:allValuesFrom* among others). For these cases other ontology construct patterns must be specified, like the one reported in Listing 1.3. Both ontology construct patterns are different yet they are equivalent to the same SHACL construct pattern.

```
?subject  a  owl:ObjectProperty;
    rdfs:range  ?rangeURL  .
```

Listing 1.2. Ontology construct pattern for *rdfs:range*

```
?subject  a  owl:ObjectProperty;
    rdfs:range  [
        owl:unionOf  ?rangeURL
    ]  .
```

Listing 1.3. Alternative ontology construct pattern for *rdfs:range*

As it can be noticed the constructs are not enough to automatically generate shapes, patterns of constructs are required in the mappings to generate them. Also, it is worth mentioning that different ontology construct patterns may generate the same SHACL construct pattern, or vice versa.

3.1   Methodology for mappings generation

The mappings were designed and written following a thorough process, which consist in the following steps:

1. **To manually list OWL 2, RDFS, XSD constructs**. Based on the OWL 2 specification [11], a list of OWL 2 constructs were gathered from the OWL 2 Web Ontology Language Primer [11]. It was decided to consider every OWL 2 construct except for the versioning constructs and the instance ones. Thus, this list includes 55 constructs out of the 88 total constructs defined in the OWL 2 specification. The same process was performed for the RDFS [3] and XSD [18] specifications obtaining 8 and 37 constructs, respectively.

2. **To manually list SHACL constructs**. Based on the SHACL specification [13], the list of SHACL restrictions was collected from the Shapes Constraint Language specification. It was decided to consider every SHACL restriction except for those related to Validation Report restrictions, since they do not add additional content to the shape. Thus, the list includes 58 constructs out of the 301 defined in the SHACL specification document.

3. **To review existing relations among ontologies and SHACL**. Some previous works have attempted to relate OWL 2 constructs with the SHACL constructs [12, 17]. Some authors hypothesised that construct patterns would be required to automatically generate shapes [17]. These proposals were taken as starting point to develop the Astrea-KG mappings.

4. **To generate mappings between ontology construct patterns and their equivalent SHACL construct patterns**. For each ontology construct pattern, the equivalent pattern in SHACL has been proposed. It should be mentioned that an ontology construct pattern can be equivalent to multiple SHACL construct patterns, and vice versa.

5. **To include data restrictions**. The only restrictions over data that ontologies encode is the *xsd:pattern* one. Nevertheless, the XSD schema specifies different datatypes that have specific restrictions over values [18], such as their maximum, minimum, or lexical pattern. We incorporated to the mappings the restrictions specified by the XSD datatype anytime in an ontology a XSD datatype was specified. For instance, when a *xsd:nonNegativeInteger* is defined as datatype for a literal it means that it follows the pattern "$[-\backslash+]?[0-9]+$", an has a minimum inclusive of 0.

6. **To implement executable-mappings**. Finally, all the identified equivalences between the ontology construct patterns (OWL, RDFS, and XSD) and the SHACL construct patterns were implemented as SPARQL queries.

3.2 Mappings ontology

The Astrea-KG contains and publishes the information of the 157 defined mappings. A vocabulary to model these mappings has been defined[4]. Figure 1 shows an overview of such vocabulary, which models the relation between the mappings between ontology and SHACL construct patterns, and the mapping implementations as SPARQL queries.

   As depicted in Figure 1, two *Patterns* can be mapped by each other, which indicates that they are equivalent. Any of these patterns could be an *OntologyPattern*, if the pattern includes statements that contain any construct from OWL, RDFS, or XSD. Alternatively, a pattern may be a *SHACLPattern* if it includes statements that contain SHACL constructs. A *Pattern* can be mapped by other *Pattern*, and this relation is symmetric. Furthermore, each *Pattern* is related to a *MappingImplementation* which contains the necessary information to translate this source pattern to the equivalent target pattern.

---

[4] https://w3id.org/def/astrea

An example of mapping is depicted in Figure 2. It can be observed that the *OntologyPattern* is capturing the *cardinality* pattern when it is specified as an *owl:Restriction* for an *owl:Class*. Similarly, the *SHACLPattern* refers to a *sh:PropertyShape* related to a *sh:NodeShape*, which has a *sh:minCount* and *sh:maxCount*. These two patterns are related by means of the *isMappedBy* relation, which indicates that the *OntologyPattern* is mapped by the *SHACLPattern* and vice versa.

Finally, in order to generate a shape the *OntologyPattern* is related to a *MappingImplementation* instance that implements such translation by means of a SPARQL query; notice that the WHERE clause encodes the ontology construct pattern and the CONSTRUCT the SHACL construct pattern.

4   Astrea

The tool Astrea aims at reading the Astrea-KG, fetching the mappings within, and executing their implementation, i.e., the SPARQL queries, over a set of ontologies. The architecture of Astrea is depicted in Figure 3; the depicted components and their performed tasks are the following:

OntologyManager: this component is fed with a set of ontology URLs provided as input (step 1 in Figure 3). These ontologies are downloaded and, then,
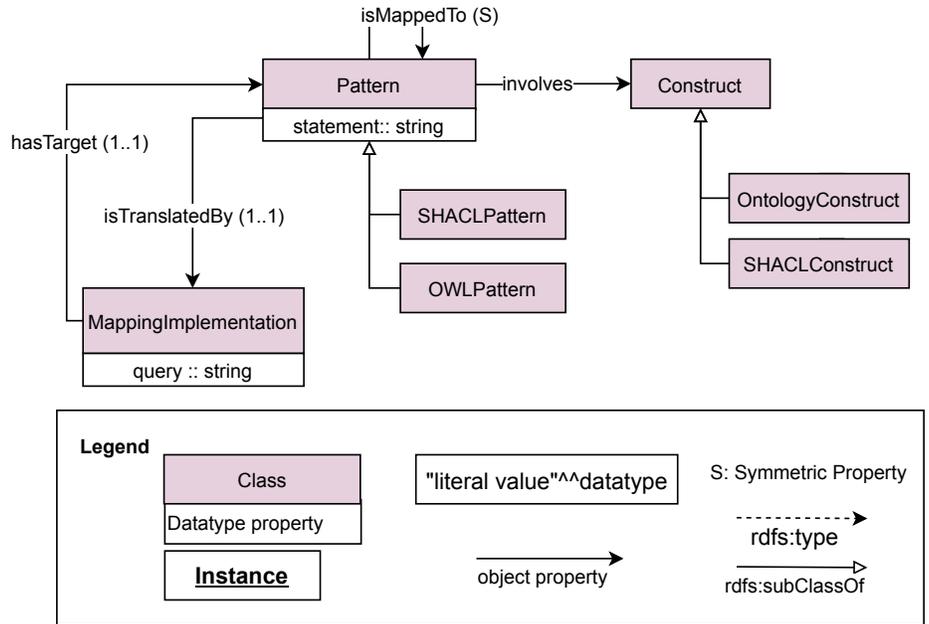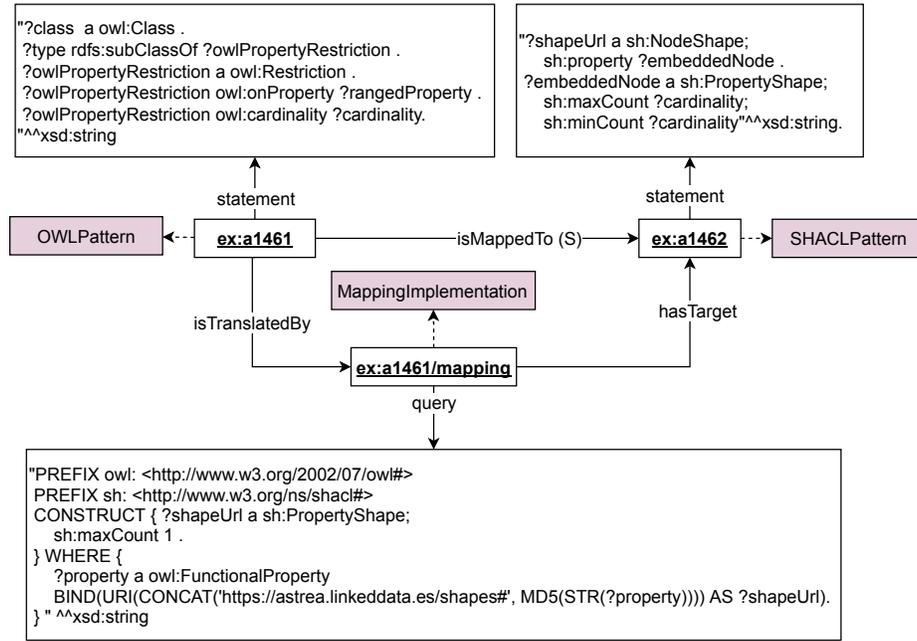

Fig. 1. Overview of the vocabulary.

```
"?class  a owl:Class .
 ?type rdfs:subClassOf ?owlPropertyRestriction .
 ?owlPropertyRestriction a owl:Restriction .
 ?owlPropertyRestriction owl:onProperty ?rangedProperty .
 ?owlPropertyRestriction owl:cardinality ?cardinality.
"^^xsd:string
```

```
"?shapeUrl a sh:NodeShape;
    sh:property ?embeddedNode .
 ?embeddedNode a sh:PropertyShape;
    sh:maxCount ?cardinality;
    sh:minCount ?cardinality"^^xsd:string.
```

statement   statement

OWLPattern ◁- - **ex:a1461** —isMappedTo (S)→ **ex:a1462** - -▷ SHACLPattern

MappingImplementation

isTranslatedBy   hasTarget

ex:a1461/mapping

query

```
"PREFIX owl: <http://www.w3.org/2002/07/owl#>
 PREFIX sh: <http://www.w3.org/ns/shacl#>
 CONSTRUCT { ?shapeUrl a sh:PropertyShape;
    sh:maxCount 1 .
} WHERE {
    ?property a owl:FunctionalProperty
    BIND(URI(CONCAT('https://astrea.linkeddata.es/shapes#', MD5(STR(?property))))) AS ?shapeUrl).
} " ^^xsd:string
```

Fig. 2. Example of SPARQL query using the vocabulary proposed in this work

for each ontollogy the *OntologyManager* checks if the statement *owl:import* is
present. When such statement is present, the ontology URL specified is down-
loaded by the *OntologyManager* (steps 2 and 3 in Figure 3). Finally, all the
ontologies retrieved are sent to the *KG-Manager* (step 4 in Figure 3).

Astrea-KG: the Astrea-KG is the KG previously described and available
online. The Astrea tool reads the latest version of this KG, entailing that any-
time a mapping is included, or modified, Astrea is aware of such update.

KG-Manager: Then, all the mapping implementations are retrieved from
the Astrea-KG (step 5 in Figure 3). Since the implementations of the mappings
are CONSTRUCT queries, they produce as result an RDF graph that contains
the SHACL shapes associated to the ontology construct mapping encoded in
the CONSTRUCT query. The *KG-Manager* issues all these queries over the
ontologies provided by the *OntologyManager*, during step 4. As a result, the
output of each query is stored in the same RDF graph. Finally the RDF graph
containing all the generated SHACL shapes is returned (step 6 in Figure 3).

4.1   Expressivity of the SHACL shapes generated by Astrea

Table 3 summarises the SHACL restrictions that are supported by Astrea using
the Astrea-KG mappings. The symbol "✓" indicates that Astrea supports the
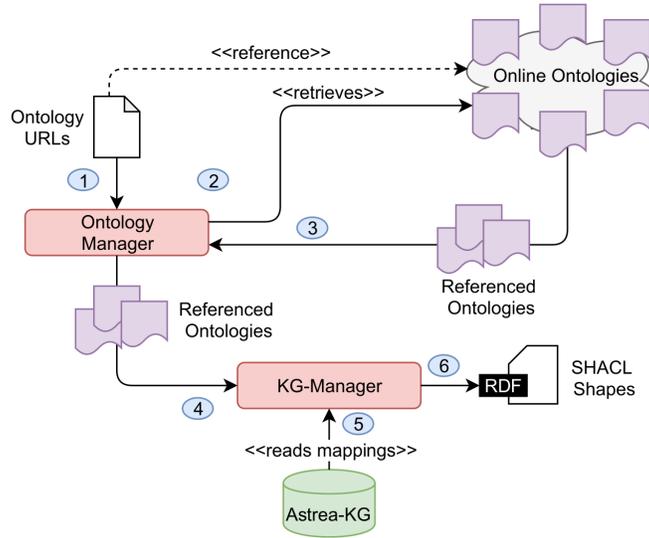
Fig. 3. Astrea architecture.

restriction, while "$\approx$" indicates that Astrea supports it partially, and "$\times$" that it is out of scope of Astrea.

From Table 3 it can be observed that Astrea covers 60% of the SHACL restrictions and 40% are not supported. Notice that the *sh:pattern* restriction is only partially covered. In addition, the unsupported restrictions can be classified as follows: data value restrictions, practitioner-required restrictions, and unfeasible restrictions. Next, we provide an insight for the partially covered restrictions and for the three unsupported types of restrictions.

Partially covered: ontologies infrequently contain patterns for data values, although they could be specified by means of the *xsd:pattern* statement. Nevertheless, it is common to assign a *XSD* datatype to data values; these datatypes have restrictions defined by the W3C [18]. The mappings are endowed to cover the patterns specified by the *xsd:pattern* statement. Additionally, when a data value has a *XSD* datatype with no pattern defined, the mappings automatically inject the restrictions defined by the W3C. As a result, Astrea covers *sh:pattern* restrictions under these two circumstances.

In addition, the *sh:qualifiedValueShape* restriction specifies the condition that a specified number of value nodes needs to conform to. Thus, the range of this construct could be any shape that refers to a constraint, e.g., the class of the node, the possible list of values of the node or the pattern that the node should follow. Astrea only covers the *sh:qualifiedValueShape* restriction when it defines the specific class of the node.

Data value restrictions: the majority of the restrictions from Table 3 are not supported because they refer to data instances; which are 20.7% of the unsupported restrictions. Due to the fact that Astrea only takes ontologies into account, and they are not expected to have data instances, supporting these restrictions is not feasible. These restrictions are: *sh:targetObjectsOf*, *sh:targetSubjectsOf*, *sh:alternativePath*, *sh:zeroOrMorePath*, *sh:oneOrMorePath*, *sh:zeroOrOnePath*, *sh:ignoredProperties*, *sh:lessThan*, *sh:lessThanOrEquals*, *sh:order*, *sh:BlankNode*, and *sh:BlankNodeOrLiteral*.

Practitioner-required restrictions: a smaller amount of restrictions, i.e., 13.8%, require a practitioner to establish them. These restrictions depend on a domain problem and the granularity of the desired validation using shapes. These restrictions are: *sh:targetNode*, *sh:value sh:flags*, *sh:defaultValue*, *sh:group*, *sh:qualifiedValueShape*, *sh:qualifiedValueShapeDisjoint*, and *sh:close*.

Unfeasible restrictions: 5.5% of the restrictions were not supported because their equivalent ontology construct patterns were not found. These restrictions are: *sh:languageIn*, *sh:uniqueLang*, and *sh:xone*.

SHACL restriction	Coverage	SHACL restriction	Coverage
sh:Shape	✓	sh:maxInclusive	✓
sh:NodeShape	✓	sh:maxLength	✓
sh:PropertyShape	✓	sh:minCount	✓
sh:nodeKind	✓	sh:minExclusive	✓
sh:targetClass	✓	sh:minInclusive	✓
sh:targetNode	×[b]	sh:minLength	✓
sh:targetObjectsOf	×[a]	sh:node	×[b]
sh:targetSubjectsOf	×[a]	sh:not	✓
sh:value	×[b]	sh:or	✓
sh:path	✓	sh:pattern	≈
sh:inversePath	✓	sh:flags	×[b]
sh:alternativePath	×[a]	sh:property	✓
sh:zeroOrMorePath	×[a]	sh:qualifiedMaxCount	✓
sh:oneOrMorePath	×[a]	sh:qualifiedValueShape	≈
sh:zeroOrOnePath	×[a]	sh:qualifiedValueShapesDisjoint	×[b]
sh:and	✓	sh:qualifiedMinCount	✓
sh:class	✓	sh:uniqueLang	×[c]
sh:closed	×[b]	sh:xone	×[c]
sh:datatype	✓	sh:defaultValue	×[b]
sh:ignoredProperties	×[a]	sh:description	✓
sh:maxCount	✓	sh:group	×[b]
sh:disjoint	✓	sh:name	✓
sh:equals	✓	sh:order	×[a]
sh:hasValue	✓	sh:BlankNode	×[a]
sh:in	✓	sh:BlankNodeOrIRI	✓
sh:languageIn	×[c]	sh:BlankNodeOrLiteral	×[a]
sh:lessThan	×[a]	sh:IRI	✓
sh:lessThanOrEquals	×[a]	sh:IRIOrLiteral	✓
sh:maxExclusive	✓	sh:Literal	✓

✓ covered
≈ partially covered
× not covered
[a] not covered due to data value restrictions
[b] not covered due to practitioner-required restrictions
[c] not covered due to unfeasible restrictions

Table 3. List of SHACL restrictions supported by Astrea

4.2 Availability, sustainability, extensibility and maintenance of the resources

The Astrea tool is available as an online web application[2]. This web application also provides a REST API and its documentation[5]. Furthermore, the code of the tool is also available in GitHub[6] under the Apache 2.0 licence[7]. The maintenance of the tool will be facilitated through the continuous update of the mappings, and the fact that the architecture is automatically aware of these changes.

The current version of the Astrea-KG is available as a Zenodo resource. It has a canonical citation using a DOI (https://doi.org/10.5281/zenodo.3571009(and is published under the Creative Commons Attribution 4.0 International[8]

---

[5] https://astrea.linkeddata.es/swagger-ui.html

[6] https://github.com/oeg-upm/Astrea

[7] https://www.apache.org/licenses/LICENSE-2.0

[8] https://creativecommons.org/licenses/by/4.0

(CC BY 4.0) license. In addition, the KG is publicly available for third-party contributions or reusability[1]. A further analysis will be performed in order to include, if possible, new mappings.

5 Experiments

In order to validate the two resources of this paper, i.e., Astrea-KG and Astrea, we performed two experiments. Both experiments rely on a set of well-known ontologies, most of which are standards, namely: *W3C Time* [4], *ETSI SAREF* [6], *ETSI SAREF* extension for environment[9] (S4ENVI), *ETSI SAREF* extension for buildings [20] (S4BLDG)[10], and *W3C SSN* [10] and *DBpedia* 2016-10[11]. In addition, experiments relied on two ontologies developed in the context of the European Projects VICINITY and DELTA, namely: the VICINITY ontology[12] and the DELTA ontology[13]. All the shapes generated during both experiments using Astrea were manually validated in term of syntax correctness using the SHACL playground[14].

Bear in mind that whether an ontology uses the term *owl:imports* the referenced ontology will be loaded and its shapes generated; for example, SAREF imports W3C Time or SSN imports SOSA [9], or VICINITY imports other well-known ontologies. Consider that the restrictions in the shapes depend directly on how the ontologies are expressed, using third-party ontologies provides a fair input for Astrea since they have not been biased to produce especially expressive and rich shapes.

In addition to these experiments the Astrea implementation has been tested by means of 108 JUnit tests, available on its GitHub repository[6]. For each test the expected SHACL shape was defined, and an ontology fragment was provided as input to Astrea. As a result, the expected output was compared to the actual result of Astrea.

5.1 SHACL shapes of public available ontologies

The first experiment consists in generating the SHACL shapes of all the ontologies previously mentioned, measuring the number of classes and properties (data and object properties) within and the number of *sh:NodeShape* and *sh:PropertyShape* generated. In addition, the total number of SHACL restrictions and the average time that took generating 10 times each of the shapes is reported. Table 4 recaps the results of this experiment.

The generation times for the ontologies are quite low. Figure 4 depicts how the generation times are related to the number of total SHACL restrictions

---

[9] https://saref.etsi.org/saref4envi
[10] https://saref.etsi.org/saref4bldg
[11] http://downloads.dbpedia.org/2016-10/dbpedia_2016-10.owl
[12] http://iot.linkeddata.es
[13] http://delta.iot.linkeddata.es
[14] https://shacl.org/playground/

Ontology	Classes	Properties	Node Shapes	Property Shapes	Total Restrictions	Generation Time (s)
DELTA	74	100	74	166	1,269	0.77
DBpedia	760	2,865	760	3,247	26,940	4.73
SAREF	112	97	112	238	2,076	2.05
S4BLDG	71	263	71	570	3,533	0.48
S4ENVI	30	37	30	86	699	0.35
SSN	22	38	22	87	628	1.23
Time	20	61	20	113	807	0.92
VICINITY	94	155	94	210	1,728	1.49

Table 4. SHACL shapes generated with Astrea from a set of ontologies

generated in the shapes. No correlation can be analysed since Astrea requests to remote servers the code of the ontologies when generating shapes on the fly. As a result, some servers can answer faster with large ontologies and others slower with smaller ones. This action introduces noise that prevents studying the scalability or the correlation between SHACL restrictions and generation time.
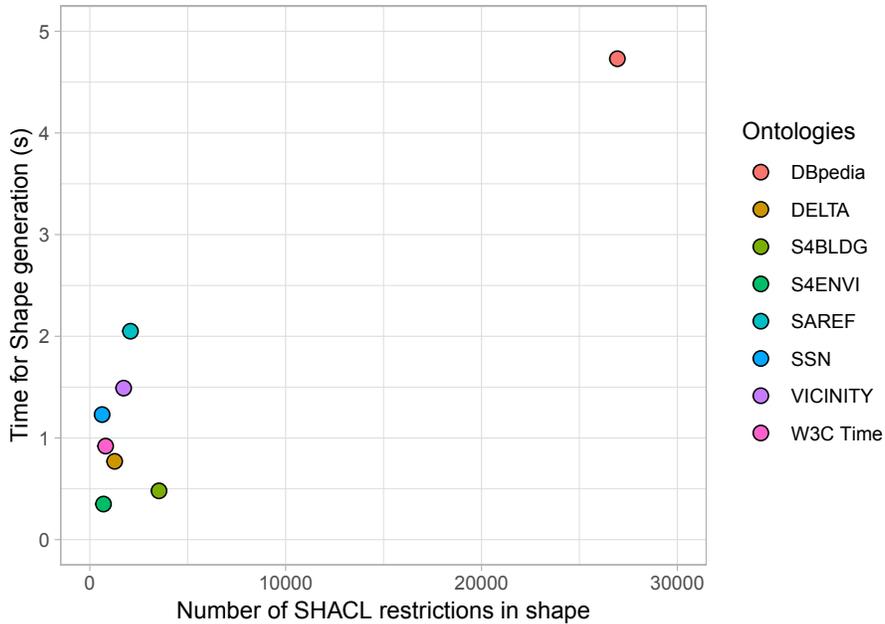
Fig. 4. Shape generation time and number of SHACL restrictions for given ontologies

Conclusions: the ratio of classes is held in the generated shapes. The ratio of properties is lower than the property shapes, the reason is the existence of restrictions for properties that apply only when they are used by a specific class producing more property shapes. Unfortunately, analysing the scalability is unfeasible in this scenario as previously explained. Nevertheless, generating shapes is not a critical task and the generation times for large ontologies such as DBpedia are rather low; less than 5 seconds for producing 26,940 SHACL restrictions.

5.2   SHACL restrictions produced by Astrea

In this experiment the expressiveness and richness of the generated shapes for the ontologies considered are analysed; the expressiveness and richness refer to the variety of SHACL constructs that a generated shape may contain. For this purpose, Table 5 reports all the restrictions that can be defined in SHACL and, for each, which were produced for the given ontologies. Notice that the last row reports the total number of SHACL restrictions that a shape has.

Conclusions: the results reported in Table 5 show how expressive and rich the SHACL shapes produced by Astrea are. The shape of the Time ontology contains 65% of all the supported constructs, SAREF 77%, SAREF for environment and for building contain both 68%, SSN 56%, DBpedia 38%, and VICINITY and DELTA contain both 65%. It is worth mentioning that some of the SHACL constructs where not generated although they are supported, the reason is that their associated ontologies lacked of the ontology construct patterns required to generate such SHACL constructs.

6   Discussion and Conclusions

Nowadays, data has been published as Knowledge Graphs in a wide number of environments and domains. The data of these KGs is expressed in RDF and modelled by means of ontologies. SHACL shapes have been endowed with the goal of providing a validation mechanism to guarantee data quality for these Knowledge Graphs. Producing SHACL shapes manually is an unfeasible task, due to the data size, to the need for expert knowledge, and to the need to use as much different restrictions as possible. Astrea generates SHACL shapes taking as input ontologies, which encode the expert knowledge of a domain, and produces either data model and data value restrictions.

In this article two main contributions were presented: A) The Astrea-KG, that contains a set of mappings that allow the generation of SHACL shapes from one or more ontologies; and B) Astrea, which is a tool that using the Astrea-KG produces the shapes. We carried out two experiments, the former aims at evaluating the generation time, whereas the second evaluates the expressiveness of the SHACL shapes produced.

Table 5. Expressivity of the generated SHACL shapes

SHACL Restriction	DBpedia	DELTA	SAREF	S4BLDG	S4ENVI	SSN	Time	VICINITY
sh:NodeShape	760	74	112	71	30	22	20	94
sh:PropertyShape	3,247	166	238	570	86	87	113	210
sh:nodeKind	4,037	240	350	641	116	109	133	304
sh:targetClass	760	74	112	71	30	22	20	94
sh:path	-	67	141	307	49	49	52	60
sh:inversePath	-	-	1	-	1	1	-	2
sh:and	-	-	-	-	-	-	-	-
sh:class	828	42	84	218	37	41	33	89
sh:datatype	1,760	33	37	172	24	1	32	28
sh:disjoint	25	1	5	2	11	23	1	6
sh:equals	634	1	-	-	-	-	-	3
sh:hasValue	-	-	15	-	3	-	5	-
sh:in	-	-	-	-	-	-	-	-
sh:maxCount	30	-	42	1	9	15	38	16
sh:maxExclusive	-	-	-	-	-	-	-	-
sh:maxInclusive	2	-	-	-	-	-	-	-
sh:maxLength	-	-	-	-	-	-	-	-
sh:minCount	-	-	19	-	-	-	19	-
sh:minExclusive	-	-	-	-	-	-	-	-
sh:minInclusive	285	-	4	1	-	-	4	-
sh:minLength	-	-	-	-	-	-	-	-
sh:not	-	30	49	42	15	17	9	29
sh:name	11,780	154	209	333	66	57	81	221
sh:BlankNodeOrIRI	-	62	63	179	24	36	33	121
sh:IRI	-	74	112	71	30	22	20	94
sh:Literal	-	38	34	84	13	2	28	29
sh:IRIOrLiteral	-	67	141	307	49	49	52	60
sh:pattern	1,560	22	28	82	9	1	24	24
sh:property	-	29	49	42	15	17	9	28
sh:qualifiedMaxCount	-	1	2	2	5	-	-	4
sh:qualifiedMinCount	-	1	10	2	5	-	-	-
sh:description	1,232	92	208	333	67	57	80	208
sh:or	-	-	1	-	-	-	1	-
sh:qualifiedValue\-Shape	-	1	10	2	5	-	-	4
Total restrictions	26,940	1,269	2,076	3,533	699	628	807	1,728

The mappings presented in this article are bi-directional; however, the implementation provided in Astrea-KG works only from ontology construct patterns to SHACL construct patterns. A further analysis will be performed in the future to design the mapping implementations that produce ontologies from SHACL shapes. Implementing the round-trip translation will enable an interesting research path that may settle the basis to establish potential new quality measurements for shapes; checking how aligned their restrictions are regarding a given ontology.

Finally, during the experiments carried out we realised that there is not an automatic way to compare shapes, which could report which one has richer expressiveness or which is more restrictive. Similarly, we noticed that there is no way to combine two or more shapes, which is a rather interesting issue; this experience led to endow the definition of operators that combine shapes. For instance, defining an operator *restrictive* whose input are two shapes, and which output will be a new shape with the hardest restrictions from both inputs.

Astrea is meant to offer different extension points: including new patterns, applying other shape-learning techniques that are not ontology-based to enhance output shapes, or derived research lines that use automatic-shape generation.

References

1. Boneva, I., Dusart, J., Fernández-Álvarez, D., Labra-Gayo, J.E.: Shape Designer for ShEx and SHACL Constraints (2019)
2. Boneva, I., Labra-Gayo, J.E., Hym, S., Prud'hommeau, E.G., Solbrig, H.R., Staworko, S.: Validating RDF with shape expressions. CoRR, abs/1404.1270 (2014)
3. Brickley, D., Guha, R.V., McBride, B.: RDF Schema 1.1. Tech. rep. (2014)
4. Cox, S., Little, C.: Time ontology in OWL. Tech. rep. (2017)
5. Debattista, J., Lange, C., Auer, S., Cortis, D.: Evaluating the quality of the LOD cloud: an empirical investigation. Semantic Web **9**(6), 859–901 (2018)
6. ETSI, T.: 264 411-v1. 1.1: Smartm2m; smart appliances; reference ontology and onem2m mapping. Tech. rep., Technical report, ETSI (2015)
7. Fernández-Álvarez, D., García-González, H., Frey, J., Hellmann, S., Labra-Gayo, J.E.: Inference of Latent Shape Expressions Associated to DBpedia Ontology. In: Proceedings of the ISWC 2018 Posters & Demonstrations, Industry and Blue Sky Ideas Tracks co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 8th-12th (2018)
8. Gangemi, A., Presutti, V.: Ontology design patterns. In: Handbook on ontologies, pp. 221–243. Springer (2009)
9. Haller, A., Janowicz, K., Cox, S., Le Phuoc, D., Taylor, K., Lefrançois, M.: Semantic sensor network ontology. W3C recommendation, W3C (2017)
10. Haller, A., Janowicz, K., Cox, S.J., Lefrançois, M., Taylor, K., Le Phuoc, D., Lieberman, J., García-Castro, R., Atkinson, R., Stadler, C.: The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. Semantic Web **10**(1), 9–32 (2019)
11. Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S., et al.: OWL 2 web ontology language primer. W3C recommendation (2009)
12. Knublauch, H.: SHACL and OWL Compared (2017), https://spinrdf.org/shacl-and-owl.html
13. Knublauch, H., Kontokostas, D.: Shapes Constraint Language (SHACL). W3C recommendation (2017)
14. Labra-Gayo, J.E., García-González, H., Fernández-Alvarez, D., Prud'hommeaux, E.: Challenges in RDF Validation. In: Current Trends in Semantic Web Technologies: Theory and Practice, pp. 121–151. Springer (2019)
15. Labra-Gayo, J.E., Prud'Hommeaux, E., Boneva, I., Kontokostas, D.: Validating rdf data. Synthesis Lectures on Semantic Web: Theory and Technology **7**(1), 1–328 (2017)

---

16. Mihindukulasooriya, N., Rashid, M.R.A., Rizzo, G., García-Castro, R., Corcho, O., Torchiano, M.: RDF shape induction using knowledge base profiling. In: Proceedings of the 33rd Annual ACM Symposium on Applied Computing. pp. 1952–1959. ACM (2018)
17. Pandit, H.J., O'Sullivan, D., Lewis, D.: Using Ontology Design Patterns To Define SHACL Shapes. In: Proceedings of the 9th Workshop on Ontology Design and Patterns (WOP 2018) co-located with 17th International Semantic Web Conference (ISWC 2018), Monterey, USA, October 9th, 2018. pp. 67–71 (2018)
18. Peterson, D., Gao, S., Malhotra, A., Sperberg-McQueen, C.M., Thompson, H.S., Biron, P.: W3C XML schema definition language (XSD) 1.1 part 2: Datatypes. World Wide Web Consortium, Working Draft (June 2008) (2012)
19. Schmachtenberg, M., Bizer, C., Paulheim, H.: State of the LOD Cloud 2014. University of Mannheim, Data and Web Science Group. August **30** (2014)
20. SmartM2M: SAREF extension investigation Technical Report (TR 103 411). Tech. rep., Technical report, ETSI
21. Spahiu, B., Maurino, A., Meusel, R.: Topic profiling benchmarks in the linked open data cloud: Issues and lessons learned. Semantic Web (Preprint), 1–20 (2019)
22. Spahiu, B., Maurino, A., Palmonari, M.: Towards Improving the Quality of Knowledge Graphs with Data-driven Ontology Patterns and SHACL. In: Proceedings of the 9th Workshop on Ontology Design and Patterns (WOP 2018) co-located with 17th International Semantic Web Conference (ISWC 2018). pp. 103–117 (2018)