SCOOP-UI:
SHACL Shape Extraction in Just a Click!

Xuemin Duan[1], David Chaves-Fraga[2,1], and Anastasia Dimou[1]

[1] KU Leuven – Flanders Make@KULeuven – Leuven.AI, Belgium
{xuemin.duan,anastasia.dimou}@kuleuven.be
[2] Departamento de Electrónica e Computación, Universidade de Santiago de Compostela, Spain
david.chaves@usc.es

Abstract. The proliferation of knowledge graph validation using the Shapes Constraint Language (SHACL) has catalyzed significant efforts towards automating the extraction of SHACL shapes. These shapes may be derived from RDF graphs, or the various components which are involved in their creation, e.g., ontologies, raw data schemas, and mapping rules. In SCOOP, we integrate shapes extracted from these components, however, no system exists that enables the users to streamline the extraction and integration processes. In this work, we present SCOOP-UI, a web application built on top of SCOOP to provide an editor for the users to handle the different component's resources to extract and integrate their SHACL shapes. This work enables users to get directly involved in the translation of various resources into SHACL shapes, as well as in the integration of these shapes to produce a unified representation.

License: Apache-2.0
Demo URL: https://demos.citius.usc.es/scoop/
Source Code: https://github.com/dtai-kg/SCOOP-UI
Video: https://demos.citius.usc.es/scoop/#doc

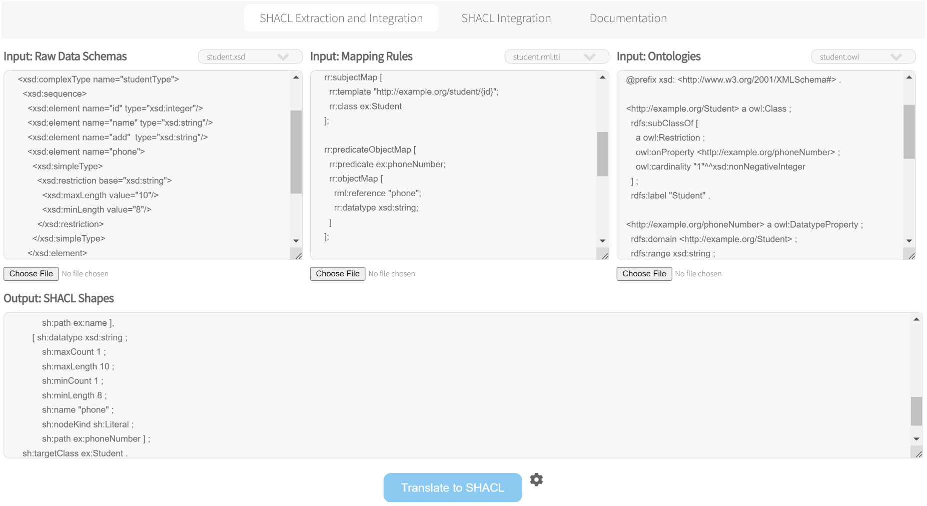
Keywords: SHACL · Shape integration · Web application

1 Introduction

Automatic extraction of shapes e.g., in the Shape Constraint Language[1] (SHACL), to validate RDF graphs is crucial to guarantee consistency in data transformation. Extracting SHACL shapes leverage either the RDF graph directly [5] or associated sources such as ontologies [1], mapping rules [2], and raw data schemas [4] utilized in constructing the RDF graph. The former approach entails constraint extraction via direct analysis of the RDF graph, whereas the latter involves the translation of pre-existing constraints into SHACL constraints.

While the majority of current research concentrates on extracting shapes from an individual source [3], e.g., from RDF graphs, ontologies, mapping rules, and raw data schemas, no attention has been given to the potential coexistence of

[1] https://www.w3.org/TR/shacl/



```
<xsd:complexType name="studentType">
 <xsd:sequence>
  <xsd:element name="id" type="xsd:integer"/>
  <xsd:element name="name" type="xsd:string"/>
  <xsd:element name="add"  type="xsd:string"/>
  <xsd:element name="phone">
   <xsd:simpleType>
    <xsd:restriction base="xsd:string">
     <xsd:maxLength value="10"/>
     <xsd:minLength value="8"/>
    </xsd:restriction>
   </xsd:simpleType>
  </xsd:element>
```

```
rr:subjectMap [
 rr:template "http://example.org/student/{id}";
 rr:class ex:Student
];

rr:predicateObjectMap [
 rr:predicate ex:phoneNumber;
 rr:objectMap [
  rml:reference "phone";
  rr:datatype xsd:string;
 ]
];
```

```
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

<http://example.org/Student> a owl:Class ;
 rdfs:subClassOf [
  a owl:Restriction ;
  owl:onProperty <http://example.org/phoneNumber> ;
  owl:cardinality "1"^^xsd:nonNegativeInteger
 ] ;
 rdfs:label "Student" .

<http://example.org/phoneNumber> a owl:DatatypeProperty ;
 rdfs:domain <http://example.org/Student> ;
 rdfs:range xsd:string ;
```

```
      sh:path ex:name ],
    [ sh:datatype xsd:string ;
      sh:maxCount 1 ;
      sh:maxLength 10 ;
      sh:minCount 1 ;
      sh:minLength 8 ;
      sh:name "phone" ;
      sh:nodeKind sh:Literal ;
      sh:path ex:phoneNumber ] ;
  sh:targetClass ex:Student .
```

Fig. 1: SHACL shape extraction and integration via SCOOP-UI

diverse sources related to RDF graph creation. SCOOP[2] [3], a shape integration framework, addressed this research gap. By leveraging SCOOP, SHACL shapes extracted from the data schema and ontology used to create an RDF graph according to a set of mapping rules, can be integrated into a unified SHACL shape, facilitating comprehensive validation of RDF graphs.

This work presents the SCOOP-UI[3], an open-source web-based application built on top of SCOOP[4], aiming to facilitate the automatic extraction and integration of SHACL shapes. SCOOP-UI supports (i) the extraction of shapes from a single source, such as XML Schema (XSD) via its XSD2SHACL component, RDF Mapping Language (RML) via its RML2SHACL component, and Web Ontology Language (OWL) via its OWL2SHACL component; (ii) the extraction and integration of unified shapes from multiple sources (integrate either shapes extracted or existing). Last, it offers various predefined integration strategies; and also allows user-defined integration source priority.

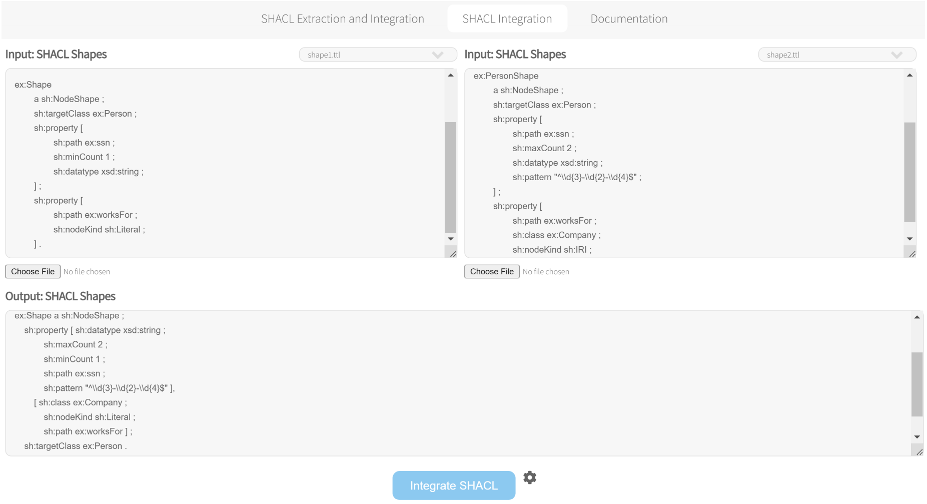
2 SCOOP-UI Features

This section presents the functions and configurations supported by SCOOP-UI.

Core functions SCOOP-UI may have a single source or multiple sources as inputs which can be data schemas, ontologies, mapping rules, or SHACL shapes.

Single source input. When a single source is given as input, e.g., a data schema, an ontology, or a set of mapping rules, the application automatically triggers the corresponding shapes extraction component to directly generate the

[2] Paper accepted by ESWC 2024 Resource Track
[3] https://demos.citius.usc.es/scoop/
[4] https://github.com/dtai-kg/SCOOP



```
ex:Shape
    a sh:NodeShape ;
    sh:targetClass ex:Person ;
    sh:property [
        sh:path ex:ssn ;
        sh:minCount 1 ;
        sh:datatype xsd:string ;
    ] ;
    sh:property [
        sh:path ex:worksFor ;
        sh:nodeKind sh:Literal ;
    ] .
```

```
ex:PersonShape
    a sh:NodeShape ;
    sh:targetClass ex:Person ;
    sh:property [
        sh:path ex:ssn ;
        sh:maxCount 2 ;
        sh:datatype xsd:string ;
        sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" ;
    ] ;
    sh:property [
        sh:path ex:worksFor ;
        sh:class ex:Company ;
        sh:nodeKind sh:IRI ;
```

```
ex:Shape a sh:NodeShape ;
  sh:property [ sh:datatype xsd:string ;
      sh:maxCount 2 ;
      sh:minCount 1 ;
      sh:path ex:ssn ;
      sh:pattern "^\\d{3}-\\d{2}-\\d{4}$" ],
    [ sh:class ex:Company ;
      sh:nodeKind sh:Literal ;
      sh:path ex:worksFor ] ;
  sh:targetClass ex:Person .
```

Fig. 2: SHACL shape integration via SCOOP-UI

corresponding SHACL shapes. As the layout illustrated in Figure 1, if only the Input: Mapping Rules box is filled while leaving other text boxes empty, a set of SHACL shapes will be extracted via the RML2SHACL component.

Multiple sources inputs. When multiple sources are given as inputs, the application invokes the corresponding shape extraction components to extract the shapes (e.g., the XSD2SHACL component when an XSD resource is given as the input or the RML2SHACL component if some mapping rules are given as the input). Subsequently, the shape integration module in SCOOP integrates the extracted shapes from various sources into a unified shapes graph. As depicted in Figure 1, upon populating two or three boxes within Raw Data Schemas, Mapping Rules, and Ontologies, the application automatically performs the extraction and integration, then presents the unified shape in the SHACL shapes box.

Multiple SHACL shapes inputs. While SCOOP is designed to accommodate various input sources, this application isolates the integration module of SCOOP to enable the support for integrating multiple inputs of SHACL shapes. As illustrated in Figure 2, the application also caters to users who seek to integrate multiple SHACL shapes without the need for extraction.

Configurations SCOOP-UI empowers users to select from three integration strategies: SCOOP-**All**, SCOOP-**Prior**, and SCOOP-**Prior-R**. The adoption of different strategies influences the resolution of conflicting constraints. The **All** aims to integrate all constraints from extracted or existing shapes, utilizing logical constraint components (e.g., sh:or) to resolve potential conflicts. The **Prior** integrates constraints from lower-priority sources that do not conflict with constraints from higher-priority sources, based on user-defined priorities. The **Prior-R** is designed to filter out redundant shapes from the ontology source relative to higher-priority sources. If the SCOOP-Prior or SCOOP-Prior-R strategy is chosen, users have the flexibility to customize the priority of sources.

3 Architecture and Demonstration

SCOOP-UI[3] is comprised of a front-end user interface, back-end service, and API, facilitating an end-to-end workflow. The front-end interface is developed using HTML, CSS, and JavaScript. It encompasses a text area for file input or upload and configuration options, along with an output area for displaying results. The back-end service is constructed using the FastAPI framework[5], which receives requests from the front-end interface and invokes the corresponding SCOOP modules. SCOOP currently incorporates Astrea [1] for handling OWL files, RML2SHACL [2] for RML files, and XSD2SHACL [4] for XSD files.

The SCOOP-UI offers test examples about students and real-world use cases from railway infrastructure register[6] (RINF) for each input source, conveniently accessible through direct clicks, as depicted in Figures 1 and 2. In the demo, we demonstrate the extraction of SHACL shapes from the individual source of RML, XSD, or OWL, and illustrate the impact of different configurations on the extracted constraints when multiple sources are provided. We showcase the direct integration process when multiple SHACL shapes are provided as inputs.

4 Conclusion

Through SCOOP-UI, users can seamlessly extract and integrate unified shapes directly from various sources without the need to familiarize themselves with multiple shape extraction tools. Presently, our application supports input from OWL, RML, XSD, and existing SHACL shapes. Its versatility allows for potential expansion to accommodate additional sources in the future, contingent upon the emergence of new shape extraction tools targeting other resources.

References

1. Cimmino, A., Fernández-Izquierdo, A., García-Castro, R.: Astrea: Automatic Generation of SHACL Shapes from Ontologies. In: ESWC. Springer (2020). https://doi.org/10.1007/978-3-030-49461-2_29
2. Delva, T., Smedt, B.D., Oo, S.M., Assche, D.V., Lieber, S., Dimou, A.: RML2shacl: RDF generation taking shape. In: Proceedings of the 11th on Knowledge Capture Conference. ACM (2021). https://doi.org/10.1145/3460210.3493562
3. Duan, X., Chaves-Fraga, D., Derom, O., Dimou, A.: SCOOP all the Constraints' Flavours for your Knowledge Graph. In: Proceedings of the 21th Extended Semantic Web Conference (ESWC) (2024)
4. Duan, X., Chaves-Fraga, D., Dimou, A.: XSD2SHACL: Capturing RDF Constraints from XML Schema. In: Proceedings of the 12th Knowledge Capture Conference 2023. K-CAP '23, ACM (2023). https://doi.org/10.1145/3587259.3627565
5. Rabbani, K., Lissandrini, M., Hose, K.: SHACTOR: Improving the Quality of Large-Scale Knowledge Graphs with Validating Shapes. In: SIGMOD-Companion '23. ACM (2023). https://doi.org/10.1145/3555041.3589723

[5] `https://fastapi.tiangolo.com/`

[6] `https://www.rinf-ch.ch/documentation?lang=EN`