ShEx vs SHACL

RDF Validation tutorial

Jose Emilio Labra Gayo
WESO Research group
University of Oviedo, Spain

Eric Prud'hommeaux
World Wide Web Consortium
MIT, Cambridge, MA, USA

Harold Solbrig
Mayo Clinic, USA

Iovka Boneva
LINKS, INRIA & CNRS
University of Lille, France

ShEx vs SHACL

Simple example

```
<User> {
 schema:givenName xsd:string {1,3}
}
```

```
:User a sh:Shape ;
  sh:property [
   sh:predicate schema:name ;
   sh:minCount 1;
   sh:maxCount 3;
   sh:datatype xsd:string ;
 ] .
```

Try it with ShEx: http://goo.gl/GdQuaS

Try it with SHACL: http://goo.gl/UCSvmA

Both ShEx and SHACL behave similarly with simple examples

Some differences

Underlying philosophy

Notion of a shape

Abstract syntax

Default cardinalities

Shapes and Classes

Recursion

Repeated properties

Extension mechanism

Underlying philosophy

- ShEx is more Grammar oriented
  - Shapes look like grammar rules
  - More focus on validation results
- SHACL is more Constraint oriented
  - Shapes = collections of constraints
  - More focus on validation errors

Notion of shape

- Shapes in ShEx are defined as a label and a set of rules
  - The rules define the grammar that must be satisfied by a focus node
- Shapes in SHACL contain Scopes, Filters and constraints
  - Scopes define which nodes are selected for validation
  - Filters allow more fine-grain selection of those nodes
  - Constraints are collections of constraints on a focus node
    - Those constraints are conjunctive by default

Abstract syntax

ShEx defines an abstract syntax

- Its easy to have different serialization formats (ShExC, JSON, RDF, ...)
- ..and to check what is a well formed Schema

SHACL is defined as an RDF vocabulary

- It supports the serialization formats from RDF
- Difficult to check what is a well formed Shapes graph

See Issue 52: https://www.w3.org/2014/data-shapes/track/issues/52

Default cardinalities

ShEx: default = (1,1)

```
<User> {
 schema:givenName xsd:string
 schema:lastName  xsd:string
}
```

SHACL: default = (0,unbounded)

```
:User a sh:Shape ;
 sh:property [
  sh:predicate schema:givenName ;
  sh:datatype xsd:string ;
 ];
 sh:property [
  sh:predicate schema:lastName ;
  sh:datatype xsd:string ;
 ] .
```

```
:alice schema:givenName "Alice" ;
       schema:lastName  "Cooper" .

:bob   schema:givenName "Bob", "Robert" ;
       schema:lastName  "Smith", "Dylan" ;
```

Shapes and Classes

- ShEx is only concerned with RDF nodes
  - No interaction between validation and inference
  - Classes and just nodes with some rdf:type arc
  - ShEx can be used pre-/post-validation
- SHACL offers several mechanism that may interact with inference
  - Implicit scope Class: identifies a Shape with a Class
    - Triggers validation on all nodes that belong to that class (or its subclasses)
  - sh:class. Checks the rdf:type arc of a node
    - It also checks rdfs:subClassOf* relationships

Repeated properties

ShEx supports constraints on repeated properties

SHACL need qualifiedValueShape or partitions

- Those features are still under development

Recursion

- ShEx supports recursion
  - It is possible to define and validate cyclic data structures
- SHACL doesn't support recursion
  - Validation of cyclic data structures may require that every node has discriminating rdf:type arc
  - Allowing recursion in SHACL is still under discussion

Extension mechanism

- Extend the core language with more expressive features
  - Example: validate that "area" in a rectangle is effectively the product of "base" by "height"
- ShEx defines Semantic Actions which are language agnostic

```
%{language ...commands %}
```

- SHACL predefines an extension mechanism in SPARQL
  - In principle, it is intended that other languages could be used