Introduction to SHACL

Robin Keskisärkkä
robin.keskisarkka@gmail.com
Takeaways

• How SHACL fits into the Semantic Web technology stack
• Basic understanding of the core features in SHACL
• Hands-on experience in using SHACL to validate RDF graphs
About RDF

• The standard language for describing data on the Semantic Web
• Data is represented as directed graphs
• Many serialization formats
  • XML, JSON-LD, NT, N-Triples, Turtle, ...
• Allows for integration of different data sources
What about a schema language for RDF?

- RDF Schema
  - A misnomer and should really be “RDF Vocabulary Definition Language”
  - Limited expressivity

- OWL
  - Targets logic modeling and inferencing, not validation
  - Open-world assumption
  - No unique name assumption

- OWL under the closed-world assumption?

- Semi-official specifications (W3C submissions)
  - SPIN, Resource Shapes, ShEx, ...
What is SHACL?

- Shapes Constraint Language
- W3C recommendation since July 2017
- Allows RDF data to be validated against shapes
  - Data graph
  - Shapes graph
- ... and more (user-interface generation, inference, SPARQL extensions, etc.)
Overview

SHACL Shapes

- Targets
- Constraints
- Rules
Validation process

1. Data Graph
2. Apply targets to select focus nodes
3. Filter some focus nodes
4. Generate validation reports based on constraints
Turtle

- Shorthand expressions
  - Semi-colon denotes repeated use of preceding subject
  - Comma denotes repeated use of the preceding subject and property
  - Parentheses denote lists or collections
  - Brackets denote blank nodes (anonymous resources)
ex:PonyShape
  a sh:NodeShape ;
  sh:targetClass ex:RainbowPony ;
  sh:property ex:PonyPropertyShape .

ex:PonyPropertyShape
  a sh:PropertyShape ;
  sh:path ex:color ;
  sh:in ( ex:Pink ex:Purple ) .
ex:PonyShape
   a sh:NodeShape ;
   sh:targetClass ex:RainbowPony ;
   sh:property [ 
      sh:path ex:color ;
      sh:in ( ex:Pink ex:Purple )
   ] .
Targets

- Targets are applied to Node Shapes and specify the nodes that are to be validated

- Targets can be specified in several ways
  - targetClass – All instances of a class
  - targetNode – Specific nodes
  - targetObjectsOf – All object of a specific property
  - targetSubjectsOf – All subjects of a specific property
Targets

ex:PonyShape1
  a sh:NodeShape ;
  sh:targetClass ex:Pony .

ex:PonyShape2
  a sh:NodeShape ;

ex:PonyShape3
  a sh:NodeShape ;
  sh:targetSubjectsOf ex:hasFriend .
<table>
<thead>
<tr>
<th>Type</th>
<th>Constraints</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardinality</td>
<td>minCount, maxCount</td>
</tr>
<tr>
<td>Types of values</td>
<td>class, datatype, nodeKind</td>
</tr>
<tr>
<td>Values</td>
<td>node, in, hasValue</td>
</tr>
<tr>
<td>Range of values</td>
<td>minInclusive, maxInclusive</td>
</tr>
<tr>
<td></td>
<td>minExclusive, maxExclusive</td>
</tr>
<tr>
<td>String based</td>
<td>minLength, maxLength, pattern, stem, uniqueLang</td>
</tr>
<tr>
<td>Logical constraints</td>
<td>not, and, or, xor</td>
</tr>
<tr>
<td>Closed shapes</td>
<td>closed, ignoredProperties</td>
</tr>
<tr>
<td>Property pair constraints</td>
<td>equals, disjoint, lessThan, lessThanOrEquals</td>
</tr>
<tr>
<td>Non-validating constraints</td>
<td>name, value, defaultValue</td>
</tr>
<tr>
<td>Qualified shapes</td>
<td>qualifiedValueShape, qualifiedMinCount, qualifiedMaxCount</td>
</tr>
</tbody>
</table>
SHACL by example

https://www.slideshare.net/jelabra/shacl-by-example
Hands-on exercises
Tasks

- Tasks of increasing in difficulty
- More than a single correct solution
- Experimentation is encouraged!
- A subset of the Nobel Price dataset is available. Feel free to create your own shapes to gain more confidence in SHACL.