RULE EXECUTION WITH SHACL

HENRIETTE HARMSE

In my previous post, "Using Jena and SHACL to validate RDF Data", I have looked at how RDF data can be validated using SHACL. A closely related concern to that of constraints checking, is rule execution, for which SHACL also can be used.

1. A SHACL Rule Example

We will again use an example from the SHACL specification. Assume we have the a file `rectangles.ttl` that contains the following data:

```
@prefix ex: <http://example.com/ns#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .

ex:InvalidRectangle
        a ex:Rectangle .

ex:NonSquareRectangle
        a ex:Rectangle ;
        ex:height 2 ;
        ex:width 3 .

ex:SquareRectangle
        a ex:Rectangle ;
        ex:height 4 ;
        ex:width 4 .
```

Assuming we want to infer that when the height and width of a rectangle are equal, the rectangle represents a square, the following SHACL rule specification can be used (which we will store in `rectangleRules.ttl`):

```
@prefix ex: <http://example.com/ns#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix dash: <http://datashapes.org/dash#> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

ex:Rectangle
        a rdfs:Class, sh:NodeShape ;
        rdfs:label "Rectangle" ;
        sh:property [
                sh:path ex:height ;
```

Date: 14th March 2018.

```
                sh:datatype xsd:integer ;
                sh:maxCount 1 ;
                sh:minCount 1 ;
                sh:name "height" ;
        ] ;
        sh:property [
                sh:path ex:width ;
                sh:datatype xsd:integer ;
                sh:maxCount 1 ;
                sh:minCount 1 ;
                sh:name "width" ;
        ] ;
        sh:rule [
                a sh:TripleRule ;
                sh:subject sh:this ;
                sh:predicate rdf:type ;
                sh:object ex:Square ;
                sh:condition ex:Rectangle ;
                sh:condition [
                        sh:property [
                                sh:path ex:width ;
                                sh:equals ex:height ;
                        ] ;
                ] ;
        ] .
```

2. A Code Example using Jena

Naturally you will need to add SHACL to your Maven pom dependencies. Then the following code will execute your SHACL rules:

```
package org.shacl.tutorial;

import java.io.File;
import java.io.FileOutputStream;
import java.io.OutputStream;
import java.nio.file.Path;
import java.nio.file.Paths;

import org.apache.jena.rdf.model.Model;
import org.apache.jena.riot.RDFDataMgr;
import org.apache.jena.riot.RDFFormat;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.slf4j.Marker;
import org.slf4j.MarkerFactory;
import org.topbraid.shacl.rules.RuleUtil;
import org.topbraid.spin.util.JenaUtil;

public class ShaclRuleExecution {
  private static Logger logger = LoggerFactory.getLogger(ShaclValidation.class);
  // Why This Failure marker
  private static final Marker WTF_MARKER = MarkerFactory.getMarker("WTF");
  public static void main(String[] args) {
    try {
      Path path = Paths.get(".").toAbsolutePath().normalize();
      String data = "file:" + path.toFile().getAbsolutePath() +
          "/src/main/resources/rectangles.ttl";
      String shape = "file:" + path.toFile().getAbsolutePath() +
          "/src/main/resources/rectangleRules.ttl";


      Model dataModel = JenaUtil.createDefaultModel();
      dataModel.read(data);
      Model shapeModel = JenaUtil.createDefaultModel();
      shapeModel.read(shape);
      Model inferenceModel = JenaUtil.createDefaultModel();

      inferenceModel = RuleUtil.executeRules(dataModel, shapeModel,
          inferenceModel, null);

      String inferences = path.toFile().getAbsolutePath() +
          "/src/main/resources/inferences.ttl";
      File inferencesFile = new File(inferences);
      inferencesFile.createNewFile();
      OutputStream reportOutputStream = new FileOutputStream(inferencesFile);

      RDFDataMgr.write(reportOutputStream, inferenceModel, RDFFormat.TTL);
    } catch (Throwable t) {
      logger.error(WTF_MARKER, t.getMessage(), t);
    }
  }
}
```

3. Running the Code

Running the code will cause an `inferences.ttl` file to be written out to `$Project/src/main/resources/`. It contains the following output:

```
@prefix owl:   <http://www.w3.org/2002/07/owl#> .
@prefix rdf:   <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix xsd:   <http://www.w3.org/2001/XMLSchema#> .
@prefix rdfs:  <http://www.w3.org/2000/01/rdf-schema#> .

<http://example.com/ns#SquareRectangle>
        a       <http://example.com/ns#Square> .
```

Note that `ex:InvalidRectangle` has been ignore because it does not adhere to `sh:condition ex:Rectangle`, since it does not have `ex:height` and `ex:width` properties. Also, `ex:NonSquareRectangle` is a rectangle, not a square.

4. Conclusion

In this post I gave a brief overview of how SHACL can be used to implement rules on RDF data. This code example is available at shacl tutorial.

References