Validating RDF data

1 From the lecture

a) Why do we need a validation language for RDF?

b) Can you mention some approaches proposed for validation of RDF?

c) How is SHACL different from OWL?

d) What two main types of shapes used in SHACL and what do they describe?

Solution

a) We need to validate an RDF-graph if we want to make sure that some data is in the dataset or that it is on a certain form (not only that it exists).

b) Stardog ICV, Epistemic Description Logics, SPARQL, ShEx, SHACL

c) While OWL describes domain knowledge, vocabulary and properties. SHACL checks the actual data in the database.

d) We have node shapes and property shapes. Nodes hapes declare constraints directly on a node. Property shapes declare constraints on values associated with a node through a path.

2 Exercises: OWL and constraints

Consider this OWL statement $\mathsf{Student} \sqsubseteq \exists \mathsf{enrolledIn}.\mathsf{Course}$. It seems to express the same thing as this SHACL constraint:

```
:StudentShape a sh:NodeShape ;
  sh:targetClass :Student ;
  sh:property [
    sh:path :enrolledIn ;
    sh:minCount 1 ;
    sh:class :Course
              ] .
```

They do, however, express two quite different things.

2.1 Exercise

Give an interpretation $\mathcal{I}_1$ and a set of triples $\mathcal{A}_1$ such that:

1. $\mathcal{I}_1 \models \mathsf{Student} \sqsubseteq \exists \mathsf{enrolledIn}.\mathsf{Course}$
2. $\mathcal{I}_1 \models \mathcal{A}_1$
3. $\mathcal{A}_1$ does not satisfy the SHACL constraint.

Solution

There are infinitely many possible solutions, one possibility is to let the interpretation $\mathcal{I}_1$ be:

- $\Delta^{\mathcal{I}_1} = \{1, 2, 3\}$
- $1^{\mathcal{I}_1} = 1$, $2^{\mathcal{I}_1} = 2$, $3^{\mathcal{I}_1} = 3$
- $\text{Student}^{\mathcal{I}_1} = \{1\}$
- $\text{Course}^{\mathcal{I}_1} = \{3\}$
- $\text{enrolledIn}^{\mathcal{I}_1} = \{\langle 1, 3\rangle\}$

This interpretation satisfies the axiom $\mathcal{I}_1 \models \mathsf{Student} \sqsubseteq \exists \mathsf{enrolledIn}.\mathsf{Course}$. Now, let the set of triples A_1 to be only:

- $Student(1)$

This triple is valid in the interpretation, but since there is no triple in A_1 with information about $enrolledIn$ it does not satisfy the SHACL-constraint.

2.2 Exercises

Give an interpretation $\mathcal{I}_2$ and a set of triples $\mathcal{A}_2$ such that:

1. $\mathcal{I}_1 \not\models \mathsf{Student} \sqsubseteq \exists \mathsf{enrolledIn}.\mathsf{Course}$
2. $\mathcal{I}_1 \models \mathcal{A}_2$
3. $\mathcal{A}_2$ satisifies the SHACL constraint.

Solution

There are infinitely many possible solutions, one possibility is to let the interpretation $\mathcal{I}_2$ be:

- $\Delta^{\mathcal{I}_2} = \{1, 2, 3\}$
- $1^{\mathcal{I}_2} = 1$, $2^{\mathcal{I}_2} = 2$, $3^{\mathcal{I}_2} = 3$

- $\text{Student}^{\mathcal{I}_2} = \{1, 2\}$
- $\text{Course}^{\mathcal{I}_2} = \{3\}$
- $\text{enrolledIn}^{\mathcal{I}_2} = \{\langle 1, 3 \rangle\}$

This interpretation satisfies the axiom $\mathcal{I}_2 \not\models \mathsf{Student} \sqsubseteq \exists \mathsf{enrolledIn}.\mathsf{Course}$ because not all students in the interpretations are enrolled in a course (2 is not). More formally, $\text{Student}^{\mathcal{I}_2}$ is not a subset of $\{a \mid \text{there is a } b \text{ where } \langle a, b \rangle \in enrolledIn^{\mathcal{I}_2} \text{ and } b \in \text{Course}^{\mathcal{I}_2}\}$. Now, let the set of triples A_2 to be:

- $Student(1)$
- $enrolledIn(1, 3)$

A_2 is entailed by $\mathcal{I}_2$, and since all students in the triples are enrolled in a course, it satisfies the SHACL constraint.

3 SHACL constraints for the Simpsons family

Write the SHACL constaints in a turtle file. You can check the simpsons.ttl-file from oblig1 against these constraints using, for instance Shacl playground.

3.1 Exercises: Family shape

1. Create a shape `FamilyShape` that ensures that all instances of `fam:Family` have at least 2 members and the members are of type `foaf:Person`.
2. Run the test and check that the data does not violate the restriction.
3. Add a new instance to the family that is not of type `foaf:Person` and check that you get a violation (remove it afterwords)

Solution

```
:FamilyShape a sh:NodeShape ;
    sh:targetClass fam:Family ;
    sh:property [
        sh:path fam:hasFamilyMember ;
        sh:minCount 2;
        sh:class foaf:Person
    ] .
```

3.2 Exercises: name

1. Create a shape, `PersonShape` that ensures that all `foaf:Person`s have exactly one `foaf:name` and that it is of type `xsd:string`.
2. Run the test. What do you find?
3. Add the missing names:
   - Mona Simpson
   - Herbert Powell (Herb)
   - Abraham Simpson (Abraham)
   - Patricia Maleficent (Patty)
   - Selma Bouvier (Selma)
4. What do you find now?
5. Remove the blank-nodes with missing names from the graph and check that there are no violations.

Solution

```
:PersonShape a sh:NodeShape ;
    sh:targetClass foaf:Person ;
    sh:property [
        sh:path foaf:name ;
        sh:minCount 1;
        sh:maxCount 1;
        sh:dataType xsd:string
    ] .
```

3.3 Exercises: age

1. Extend the shape, `PersonShape` with add a property that checks that all `foaf:Persons` have exactly one `foaf:age` that is of type `xsd:int` and is a value between 0 and 120.
2. Run the test. What do you find?
3. Add missing age-values:
   - Abraham Simpson: 83
   - Mona Simpson: 66
   - Herb: 39
   - Patty: 41

- Selma: 41

4. Test again an check that the violations are gone.

Solution

```
:PersonShape a sh:NodeShape ;
    sh:targetClass foaf:Person ;
    sh:property [
        sh:path foaf:name ;
        sh:minCount 1;
        sh:maxCount 1;
        sh:dataType xsd:string
    ] ;
    sh:property [
        sh:path foaf:age ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:dataType xsd:int ;
        sh:minInclusive 0 ;
        sh:maxInclusive 120
    ] .
```

3.4 Exercises: different father and mother

In SHACL, create a property constraint, `DifferentFatherAndMother` checking that a person cannot have the same person as mother and father. Extend the `:PersonShape` with `DifferentFatherAndMother` and check if the simpsons-file violates this restriction.

Solution

```
:DifferentFatherAndMother sh:path fam:hasFather ;
  sh:disjoint fam:hasMother .


:PersonShape a sh:NodeShape ;
    sh:targetClass foaf:Person ;
    sh:property [
        sh:path foaf:name ;
        sh:minCount 1;
        sh:maxCount 1;
        sh:dataType xsd:string
    ] ;
```

```
    sh:property [
        sh:path foaf:age ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
        sh:dataType xsd:int ;
        sh:minInclusive 0 ;
        sh:maxInclusive 120
    ] ;
    sh:property :DifferentFatherAndMother.
```

It does not violate the restriction.