RDF Validation

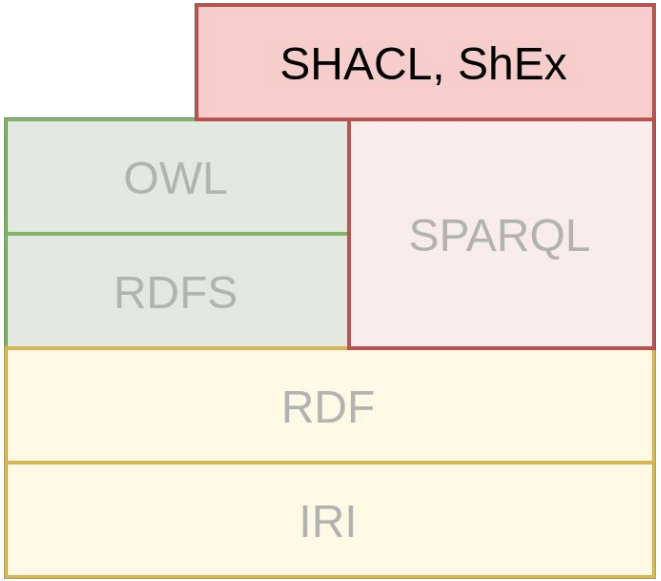
Petr Křemen

Ontologies and Semantic Web
Winter 2023

What we have got so far?

- how to identify linked data entities? IRI
- how to model a data schema? - RDFS, OWL
- how to describe data? - RDF
- how to query the data? - SPARQL

- **how to validate data? - SHACL, ShEx**



SHACL

- RDF vocabulary for validating RDF graphs
- Shapes Constraint Language (SHACL) - W3C Recommendation in 2017
- SHACL Advanced Features - W3C Working Group Note in 2017
- SHACL-Core
  - common types of shapes (constraints)
- SHACL-SPARQL
  - for constraints not expressible by SHACL-Core
- features
  - partial support for inference (`rdf:type`, `rdf:Class`, `rdfs:subClassOf`, `owl:imports`)
  - optional support for entailment (`sh:entailment`)
  - support for closed shapes
  - modular and reusable (`owl:import`, composition/inheritance of shapes)
  - support validation report, fixes for constraint violations
  - constraints extensions (SHACL-SPARQL, Javascript ...)

SHACL Example

```
@prefix ex: <http://example.org/ns#> .
@prefix schema: <http://schema.org/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
         sh:path schema:givenName ;
         sh:datatype xsd:string ;
    ] ;
    sh:property [
         sh:path schema:birthDate ;
         sh:lessThan schema:deathDate ;
         sh:maxCount 1 ;
    ]

.
```

node shape

a class, instances of which the shape applies to

property shape

path which is validated

only strings are allowed for givenName values

birthdate value must be lower than deathDate

everyone has at most one birthdate value

SHACL Shape Example

```
@prefix ex: <http://example.org/ns#> .
@prefix schema: <http://schema.org/> .
@prefix sh: <http://www.w3.org/ns/shacl#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .

schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
         sh:path schema:givenName ;
         sh:minCount 1 ;
    ] ;
.
```

node shape

a class, instances of which the shape applies to

property shape

path which is validated

everyone must have at least one given name

SHACL By Example

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:minCount 1 ;
    ]

.
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
.
```

Data graph

invalid
(givenName missing)

SHACL By Example

(added missing data)

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:minCount 1 ;
    ]

.
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
     schema:givenName "Alan"
.
```

Data graph

valid

SHACL By Example

(numeric facets)

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:minCount 1 ;
    ] ;
    sh:property [
        sh:path ex:age ;
        sh:minInclusive 0 ;
        sh:maxInclusive 150 ;
    ]
.
```

Shape graph

```
ex:AlanTuring
      a schema:Person ;
      schema:givenName "Alan" ;
      ex:age 10
.
```

Data graph

valid

SHACL string functions

- `sh:languageIn` - checking language of the literal to be from a predefined list
- `sh:uniqueLang` - checking that a property value does not have multiple values in one language
- `sh:minLength` - minimal string length
- `sh:maxLength` - maximal string length
- `sh:pattern` - regex matching

SHACL By Example

(regex)

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:minCount 1 ;
    ] ;
    sh:property [
        sh:path schema:email ;
        sh:pattern ".+@.+[.].+" ;
        sh:flags "i" ;
    ]
.
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
     schema:givenName "Alan" ;
     schema:email "alan@write.me"
.
```

Data graph

valid

SHACL By Example

(added datatype)

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
    ]

.
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
     schema:givenName "Alan"
.
```

Data graph

valid

SHACL By Example

(added family name)

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
    ]

.
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
     schema:givenName "Alan" ;
     schema:familyName "Turing"
.
```

Data graph

valid

SHACL By Example

(closed shapes)

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
    ] ;
    sh:closed true
.
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
     schema:givenName "Alan" ;
     schema:familyName "Turing"
.
```

Data graph

invalid
(closed shape)

SHACL By Example

(added extra given name)

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ]

.
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
     schema:givenName "Alan" ;
     schema:givenName "Al" ;
     schema:familyName "Turing"
.
```

Data graph

invalid

(multiple givenNames)

SHACL property pair functions

- `sh:equals` - values of two property pairs equal
- `sh:disjoint` - values of two properties differ
- `sh:lessThan` - value of one property is smaller than value of the other
- `sh:lessThanOrEquals` - value of one property is smaller than value of the other or equal to it

SHACL By Example

(added birthdate shape)

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path schema:birthDate ;
        sh:lessThan schema:deathDate ;
        sh:maxCount 1 ;
    ] ;
.
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
     schema:givenName "Alan" ;
     schema:familyName "Turing"
.
```

Data graph

valid

SHACL By Example

(added birth date)

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path schema:birthDate ;
        sh:lessThan schema:deathDate ;
        sh:maxCount 1 ;
    ] ;
.
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
     schema:givenName "Alan" ;
     schema:familyName "Turing" ;
     schema:birthDate
          "1912-06-23"^^xsd:date ;
.
```

Data graph

valid

SHACL By Example

(added death date)

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path schema:birthDate ;
        sh:lessThan schema:deathDate ;
        sh:maxCount 1 ;
    ] ;
.
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
     schema:givenName "Alan" ;
     schema:familyName "Turing" ;
     schema:birthDate
          "1912-06-23"^^xsd:date ;
     schema:deathDate
          "1904-06-07"^^xsd:date
.
```

Data graph

invalid

(deathDate before birthDate)

SHACL By Example

(added correct death date)

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path schema:birthDate ;
        sh:lessThan schema:deathDate ;
        sh:maxCount 1 ;
    ] ;
.
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
     schema:givenName "Alan" ;
     schema:familyName "Turing" ;
     schema:birthDate
          "1912-06-23"^^xsd:date ;
     schema:deathDate
          "1954-06-07"^^xsd:date
.
```

Data graph

valid

SHACL By Example

(added correct death date)

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:datatype xsd:string ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path schema:birthDate ;
        sh:lessThan schema:deathDate ;
        sh:maxCount 1 ;
    ] ;
.
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
     schema:givenName "Alan" ;
     schema:familyName "Turing" ;
     schema:birthDate
          "1912-06-23"^^xsd:date ;
     schema:deathDate
          "1954-06-07"^^xsd:date
.
```

Data graph

valid

SHACL By Example

(added parents)

```
schema:PersonShape
      a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:datatype xsd:string ;
    ] ;
    sh:property [
        sh:path schema:birthDate ;
        sh:lessThan schema:deathDate ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path schema:parent ;
        sh:class schema:Person ;
        sh:maxCount 2 ;
    ].
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
     schema:givenName "Alan" ;
     schema:familyName "Turing" ;
     schema:birthDate
          "1912-06-23"^^xsd:date ;
     schema:deathDate
          "1954-06-07"^^xsd:date ;
     schema:parent
          ex:JuliusMathisonTuring,
ex:EthelSaraStoney
.
```

Data graph

invalid

(IRIs are not typed as persons)

SHACL By Example

(added parent types)

```
schema:PersonShape
      a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:datatype xsd:string ;
    ] ;
    sh:property [
        sh:path schema:birthDate ;
        sh:lessThan schema:deathDate ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path schema:parent ;
        sh:class schema:Person ;
        sh:maxCount 2 ;
    ].
```

Shape graph

```
ex:AlanTuring
      a schema:Person ;
      schema:givenName "Alan" ;
      schema:familyName "Turing" ;
      schema:birthDate
            "1912-06-23"^^xsd:date ;
      schema:deathDate
            "1954-06-07"^^xsd:date ;
      schema:parent
            ex:JuliusMathisonTuring,
ex:EthelSaraStoney .

ex:JuliusMathisonTuring a schema:Person .
ex:EthelSaraStoney a schema:Person .
```

Data graph

valid

Other constraints

- logical constraints
  - `sh:not`, `sh:or`, `sh:and`
  - `sh:xone` – conforms to exactly one shape
- shape-based constraints
  - `sh:node` - shape to validate against
  - `sh:property` - property shape to validate against
  - `sh:qualifiedValueShape`, `sh:qualifiedMinCount`, `sh:qualifiedMaxCount`

SHACL By Example

(xone - exclusive constraints)

```
schema:PersonShape
      a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:xone (

      [sh:property [
            sh:path schema:name ;
            sh:minCount 1 ; ]]

      [sh:property [
            sh:path schema:givenName ;
            sh:minCount 1 ; ] ;
       sh:property [
            sh:path schema:familyName ;
            sh:minCount 1 ; ]]

    ) .
```

Shape graph

```
ex:AlanTuring
      a schema:Person ;
      schema:givenName "Alan" ;
      schema:familyName "Turing" ;
      schema:name "Alan Turing" .
```

Data graph

invalid

(both name and familyName+givenName provided)

SHACL By Example

(qualified constraints)

```
schema:PersonShape
      a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
      sh:path schema:parent ;
      sh:qualifiedValueShape [
            sh:path schema:gender ;
            sh:hasValue schema:Female ] ;
      sh:qualifiedMinCount 1 ] ;
    sh:property [
      sh:path schema:parent ;
      sh:qualifiedValueShape [
            sh:path schema:gender ;
            sh:hasValue schema:Male ] ;
      sh:qualifiedMinCount 1 ]
.
```

Shape graph

```
ex:AlanTuring
      a schema:Person ;
      schema:givenName "Alan" ;
      schema:familyName "Turing" ;
      schema:name "Alan Turing" ;
      schema:parent
            ex:JuliusMathisonTuring,
ex:EthelSaraStoney .

ex:JuliusMathisonTuring
      schema:gender schema:Male .
```

Data graph

invalid

(missing female parent)

SHACL By Example

(qualified constraints)

```
schema:PersonShape
      a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
      sh:path schema:parent ;
      sh:qualifiedValueShape [
            sh:path schema:gender ;
            sh:hasValue schema:Female ] ;
      sh:qualifiedMinCount 1 ] ;
    sh:property [
      sh:path schema:parent ;
      sh:qualifiedValueShape [
            sh:path schema:gender ;
            sh:hasValue schema:Male ] ;
      sh:qualifiedMinCount 1 ]
.
```

Shape graph

```
ex:AlanTuring
      a schema:Person ;
      schema:givenName "Alan" ;
      schema:familyName "Turing" ;
      schema:name "Alan Turing" ;
      schema:parent
            ex:JuliusMathisonTuring,
ex:EthelSaraStoney .

ex:JuliusMathisonTuring
      schema:gender schema:Male .

ex:EthelSaraStoney
      schema:gender schema:Female .
```

Data graph

valid

SHACL By Example

(SPARQL constraints)

```
schema:PersonShape
      a sh:NodeShape ;
  sh:targetClass schema:Person ;
  sh:sparql [
    a sh:SPARQLConstraint ;
    sh:message "Values are literals with Czech language tags." ;
    sh:select """
      SELECT $this (schema:givenName AS ?path) ?value
      WHERE {
        $this schema:givenName ?value .
        FILTER (!isLiteral(?value)
                || !langMatches(lang(?value), "cs"))
      }""" ;
  ] .
.
```

Shape graph

```
ex:AlanTuring
  a schema:Person ;
  schema:givenName "Alan" .
.
```

Data graph

invalid

(givenName does not have Czech language tag)

Defining targets

- shape applied to all instances of a class
  - `sh:targetClass schema:Person`
- shape applied to a given resource
  - `sh:targetNode ex:AlanTuring`
- shape applied to all subjects of the given predicate
  - `sh:targetSubjectsOf schema:parent`
- shape applied to all objects of the given predicate
  - `sh:targetObjectsOf schema:parent`

subClassOf resolution for class targets

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:minCount 1 ;
    ]

.
```

Shape graph

```
ex:AlanTuring
     a ex:Mathematician
.

ex:Mathematician rdfs:subClassOf
schema:Person .
```

Data graph

invalid
(IRIs are not typed as persons)

SPARQL-based targets

(Advanced feature [2])

```
schema:PersonShape
     a sh:NodeShape ;
      sh:target [
          a sh:SPARQLTarget ;
          sh:select """
              SELECT ?this
              WHERE {
                 ?this a schema:Person .
                 ?this schema:birthDate ?date .
                 FILTER (?date > "1980-01-01"^^xsd:date)
             }""" ;
      ] .
```

variable defining the object to be checked

Not part of the W3C recommendation.

SHACL Validation Results

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:minCount 1 ;
    ]
.
```

Shape graph

```
ex:AlanTuring
     a schema:Person ;
.
```

Data graph

```
[
      a sh:ValidationReport ;
      sh:conforms false ;
      sh:result
      [
            a sh:ValidationResult ;
            sh:resultSeverity sh:Violation ;
            sh:sourceConstraintComponent sh:MinCountConstraintComponent ;
            sh:sourceShape _:n1580 ;
            sh:focusNode <http://example.org/ns#AlanTuring> ;
            sh:resultPath schema:givenName ;
            sh:resultMessage "Less than 1 values" ;
      ]
] .
```

Validation report

SHACL Validation Results

(custom message)

```
schema:PersonShape
    a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:minCount 1 ;
        sh:severity sh:Warning ;
        sh:message "Missing value"@en ;
    ].
```

Shape graph

```
ex:AlanTuring
    a schema:Person ;
.
```

Data graph

Severity levels

- sh:Info
- sh:Warning
- sh:Violation

```
[
      a sh:ValidationReport ;
      sh:conforms false ;
      sh:result
      [
            a sh:ValidationResult ;
            sh:resultSeverity sh:Warning ;
            sh:sourceConstraintComponent sh:MinCountConstraintComponent ;
            sh:sourceShape _:n1580 ;
            sh:focusNode <http://example.org/ns#AlanTuring> ;
            sh:resultPath schema:givenName ;
            sh:resultMessage "Missing value"@en ;
      ]
] .
```

Validation results

SHACL functions

(Advanced feature[2])

```
ex:multiply
  a sh:SPARQLFunction ;
  rdfs:comment "Multiplies its arguments $p1 and $p2." ;
  sh:parameter [
    sh:path ex:p1 ;
    sh:datatype xsd:integer ;
    sh:description "The first operand" ; ] ;
  sh:parameter [
    sh:path ex:p2 ;
    sh:datatype xsd:integer ;
    sh:description "The second operand" ; ] ;
  sh:returnType xsd:integer ;
  sh:select """SELECT ($p1 * $p2 AS ?result) WHERE {}"""
.
```

SHACL function

```
SELECT ?subject ?area

WHERE {
  ?subject ex:width ?w.
  ?subject ex:height ?h.
  BIND (ex:multiply(?w, ?h) AS ?area) .
}
```

SPARQL

Not part of the W3C recommendation.

SHACL rules

(Advanced feature[2])

- triple rules
  - condition = triple pattern
- SPARQL rules
  - condition = SPARQL pattern
- other types possible

SHACL rules

(Advanced feature[2])

```
ex:RectangleShape
      a sh:NodeShape ;
      sh:targetClass ex:Rectangle ;
      sh:property [ sh:path ex:height ;
            sh:datatype xsd:integer ;
            sh:count 1 ;
            sh:name "height" ; ] ;

      sh:property [ sh:path ex:width ;
            sh:datatype xsd:integer ;
            sh:count 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 ; ] ; ] ; ] .
```

Shape graph

```
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 .
```

Data graph

Generates triple **`ex:SquareRectangle a ex:Square`**.

Correspondence between SHACL and SPARQL

```
schema:PersonShape
    a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:minCount 1 ;
    ]
```

Shape graph

```
ASK {
   ?this a schema:Person .
   FILTER NOT EXISTS {
       ?this schema:givenName ?givenName
   }
}
```

SPARQL

OWL vs. SHACL

checking any possible model

closed-world assumption

uses tableau reasoner

checking that data conform to data structure

deduce new knowledge from the existing one

open-world assumption

uses SPARQL

ČVUT
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

OWL vs. SHACL

- open-world assumption
- checking any possible model
- uses tableau reasoner
- deduce new knowledge from the existing one

- closed-world assumption
- checking that data conform to data structure
- uses SPARQL
- validates data

Semantic difference between SHACL and OWL

```
ex:AlanTuring a schema:Person .
```

SHACL

```
schema:PersonShape
     a sh:NodeShape ;
    sh:targetClass schema:Person ;
    sh:property [
        sh:path schema:givenName ;
        sh:minCount 1 ;
    ] ;
.
```

OWL (in DL syntax)

```
Class: schema:Person
  SubClassOf:
    :hasChild min 1 schema:givenName


(Person ⊑ (≥ 1 givenName)
```

SHACL Validation result

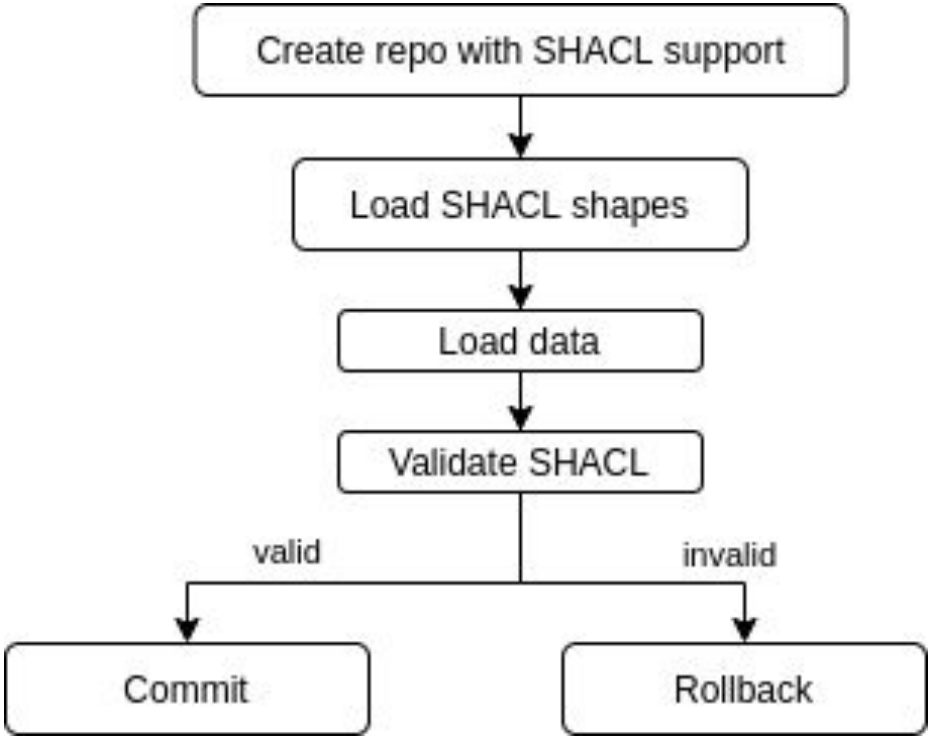
invalid
(ex:AlanTuring has no name)

OWL Consistency check

consistent
(the name is “generated” by the tableau - assured in every model)

SHACL in GraphDB

- partial SHACL support, see https://graphdb.ontotext.com/documentation/standard/shacl-validation.html
- support for SHACL and DASH constraints
- shapes not queryable using SPARQL
- validation happens upon repository update



Reference

1. Shapes Constraint Language (SHACL), H. Knublauch, D. Kontokostas, Editors, W3C Recommendation, July 20, 2017, https://www.w3.org/TR/2017/REC-shacl-20170720. Latest version available at https://www.w3.org/TR/shacl/.
2. SHACL Advanced Features, H. Knublauch, D. Allemang, S. Steyskal, Editors, W3C Working Group Note, June 8, 2017, https://www.w3.org/TR/2017/NOTE-shacl-af-20170608. Latest version available at https://www.w3.org/TR/shacl-af .
3. Form Generation using SHACL and DASH, H. Knublauch. Available at https://datashapes.org/forms.html .