how to formulate and validate constraints?

R-28-OBJECT-PROPERTY-RANGE

DSP, OWL 2, SPIN, SPARQL

valid data

```
:myAddress
    a :PostalAddress ;
    :addressCountry :netherlands .
:netherlands a :Country .
```

constraint (DSP)

```
:postalAddress
    a dsp:DescriptionTemplate ;
    dsp:minOccur 1 ;
    dsp:maxOccur "infinity” ;
    dsp:resourceClass :PostalAddress ;
    dsp:statementTemplate [
        a dsp:NonLiteralStatementTemplate ;
        dsp:minOccur 1 ;
        dsp:maxOccur "infinity" ;
        dsp:property :addressCountry ;
        dsp:nonLiteralConstraint [
            a dsp:NonLiteralConstraint ;
            dsp:valueClass :Country ] ] .
```

invalid data

```
:myAddress
    a :PostalAddress ;
    :addressCountry :amsterdam .
:amsterdam a :Locality .
```

```
:myAddress
    a :PostalAddress ;
    :addressCountry :amsterdam .
```

constraint (OWL2)

```
:addressCountry
    a owl:ObjectProperty ;
    rdfs:range :Country .
```

validation

validator

purl.org/net/rdfval-demo

executable examples

R-28-OBJECT-PROPERTY-RANGE

R-68-REQUIRED-PROPERTIES

Bibframe, DQTP, DSP, OWL 2, ReSh, ShEx, SPIN, SPARQL

valid data

```
:dcmi
    a :Organization ;
    :name "Dublin Core Metadata Initiative" .
```

constraint (DSP)

```
:organizationDescriptionTemplate
    a dsp:DescriptionTemplate ;
    dsp:minOccur 1 ;
    dsp:maxOccur "infinity" ;
    dsp:resourceClass :Organization ;
    dsp:statementTemplate [
        a dsp:NonLiteralStatementTemplate;
        dsp:minOccur 1 ;
        dsp:maxOccur "infinity" ;
        dsp:property :name ] .
```

invalid data

```
:dcmi
    a :Organization .
```

constraint (OWL2)

```
:Organization
    rdfs:subClassOf [
        a owl:Restriction ;
        owl:minCardinality 1 ;
        owl:onProperty :name ] .
```

validation

validator

purl.org/net/rdfval-demo

executable examples

R-68-REQUIRED-PROPERTIES

R-38-DEFAULT-VALUES-OF-RDF-LITERALS

SPIN, SPARQL

constraint

```
owl:Thing
  spin:rule [
    a sp:Construct ;
      sp:text """
        CONSTRUCT {
          ?this :laserSwordColor "blue" ;
          ?this :numberLaserSwords 1 . }
        WHERE {
          ?this a :Jedi . } """ ; ] .
```

data

```
:Joda
    a :Jedi , owl:Thing .
```

inferred triples

```
:Joda
    :laserSwordColor "blue" ;
    :numberLaserSwords 1.
```

constraint

```
owl:Thing
  spin:rule [
    a sp:Construct ;
      sp:text """
        CONSTRUCT {
          ?this :laserSwordColor “red" ;
          ?this :numberLaserSwords 2 . }
        WHERE {
          ?this a :Sith . } """ ; ] .
```

data

```
:DarthSidious
    a :Sith , owl:Thing .
```

inferred triples

```
:DarthSidious
    :laserSwordColor “red" ;
    :numberLaserSwords 2.
```

validation

examples

R-38-DEFAULT-VALUES-OF-RDF-LITERALS

R-52-NEGATIVE-OBJECT-PROPERTY-CONSTRAINTS

ShEx, SPIN, SPARQL

constraint

```
<FeelingForce> {
    :feelingForce (true) ,
    :attitute xsd:string }
<JediMentor> {
    :feelingForce (true) ,
    :attitute ('good') ,
    :laserSwordColor xsd:string ,
    :numberLaserSwords xsd:nonNegativeInteger ,
    :mentorOf @<JediStudent> ,
   !:studentOf @<JediMentor> }
```

matching 'JediMentor' shape

```
:Obi-Wan
    :feelingForce true ;
    :attitute 'good' ;
    :laserSwordColor 'blue' ;
    :numberLaserSwords 1 ;
    :mentorOf :Anakin .
```

constraint

```
<JediStudent> {
    :feelingForce (true) ,
    :attitute ('good') ,
    :laserSwordColor xsd:string ,
    :numberLaserSwords xsd:nonNegativeInteger ,
   !:mentorOf @<JediStudent> ,
    :studentOf @<JediMentor> }
```

matching 'JediStudent' shape

```
:Anakin
    :feelingForce true ;
    :attitute 'good' ;
    :laserSwordColor 'blue' ;
    :numberLaserSwords 1 ;
    :studentOf :Obi-Wan .
```

validation

validator

www.w3.org/2013/ShEx/FancyShExDemo

executable examples

R-52-NEGATIVE-OBJECT-PROPERTY-CONSTRAINTS

R-200-NEGATIVE-LITERAL-CONSTRAINTS

ShEx, SPIN, SPARQL

constraint

```
<Jedi> {
    :feelingForce (true) ,
    :attitute ('good') ,
    :laserSwordColor ('blue') ,
    :numberLaserSwords (1) }
```

matching 'Jedi' shape

```
:Joda
    :feelingForce true ;
    :attitute 'good' ;
    :laserSwordColor 'blue' ;
    :numberLaserSwords 1 .
```

constraint

```
<Sith> {
     :feelingForce (true) ,
    !:attitute ('good') ,
    !:laserSwordColor ('blue') ,
     :numberLaserSwords (2) }
```

matching 'Sith' shape

```
:DarthSidious
    :feelingForce true ;
    :attitute 'evil' ;
    :laserSwordColor 'red' ;
    :numberLaserSwords 2 .
```

validation

validator

www.w3.org/2013/ShEx/FancyShExDemo

executable examples

R-200-NEGATIVE-LITERAL-CONSTRAINTS

validation and inferencing

R-113-INTERACTION-OF-VALIDATION-WITH-REASONING

R-198-RDF-VALIDATION-AFTER-INFERENCING

OWL 2

R-63-TRANSITIVE-OBJECT-PROPERTIES (constraint)

```
:ancestorOf a owl:TransitiveProperty .
```

R-63-TRANSITIVE-OBJECT-PROPERTIES (data)

```
:Carter
    :ancestorOf :Lois .
:Lois
    :ancestorOf :Meg .

# :Carter
#     :ancestorOf :Meg .
```

validation without inferencing

→ constraint violation

R-63-TRANSITIVE-OBJECT-PROPERTIES (data)

```
:Carter
    :ancestorOf :Lois ;
    a owl2spin:ToInfer .
:Lois
    :ancestorOf :Meg .

:Carter
    :ancestorOf :Meg .
```

validation with inferencing

→ NO constraint violation

validation

validator

purl.org/net/rdfval-demo

executable examples

R-63-TRANSITIVE-OBJECT-PROPERTIES

R-44-PATTERN-MATCHING-ON-RDF-LITERALS

DQTP, OWL 2 DL, ReSh, ShEx, SPARQL, SPIN

valid data

```
:TimBernersLee
    :hasSSN "123-45-6789"^^:SSN .
```

invalid data

```
:TimBernersLee
    :hasSSN "123456789"^^:SSN .
```

constraint

```
:SSN
  a rdfs:Datatype ;
  owl:equivalentClass [
    a rdfs:Datatype ;
    owl:onDatatype xsd:string ;
    owl:withRestrictions (
      [ xsd:pattern
        "[0-9]{3}-[0-9]{2}-[0-9]{4}" ] ) ] .

:hasSSN rdfs:range :SSN .
```

validation

validator

purl.org/net/rdfval-demo

executable examples

R-44-PATTERN-MATCHING-ON-RDF-LITERALS

R-43-COMPARISONS-BASED-ON-DATATYPE

DQTP, ShEx, SPARQL, SPIN

constraint

```
SELECT ?s WHERE {
     ?s %%P1%% ?v1 .
     ?s %%P2%% ?v2 .
     FILTER ( ?v1 %%OP%% ?v2 ) }
```

test binding

```
dbo:deathDate < dbo:birthDate

P1 => dbo:deathDate
P2 => dbo:birthDate
OP => <
```

valid data

```
:AlbertEinstein
    dbo:birthDate '1879-03-14'^^xsd:date ;
    dbo:deathDate '1955-04-18'^^xsd:date .
```

invalid data

```
:NeilArmstrong
    dbo:birthDate '2012-08-25'^^xsd:date ;
    dbo:deathDate '1930-08-05'^^xsd:date .
```

validation

examples

R-43-COMPARISONS-BASED-ON-DATATYPE

R-45-RANGES-OF-RDF-LITERAL-VALUES

DQTP, OWL 2 DL, SPARQL, SPIN

constraint

```
:NumberPlayersPerWorldCupTeam
  a rdfs:Datatype ;
  owl:equivalentClass [
    a rdfs:Datatype ;
    owl:onDatatype xsd:nonNegativeInteger ;
    owl:withRestrictions (
      [ xsd:minInclusive 1 ]
      [ xsd:maxInclusive 23 ] ) ] .

:position
  rdfs:range :NumberPlayersPerWorldCupTeam .
```

valid data

```
:MarioGoetze
    :position "19"^^:NumberPlayersPerWorldCupTeam .
```

invalid data

```
:MarioGoetze
    :position "99"^^:NumberPlayersPerWorldCupTeam .
```

validation

validator

purl.org/net/rdfval-demo

examples

R-45-RANGES-OF-RDF-LITERAL-VALUES

R-13-DISJOINT-GROUP-OF-PROPERTIES-CLASS-SPECIFIC

ShEx, SPIN, SPARQL

constraint

```
<Human> {
    (
            foaf:name xsd:string
        |
            foaf:givenName xsd:string+ ,
            foaf:familyName xsd:string
    )
}
```

matching 'Human' shape

```
:Luke
    foaf:givenName "Luke" ;
    foaf:familyName "Skywalker" .

-----

:Leia
    foaf:name "Leia Skywalker" .
```

NOT matching 'Human' shape

```
:Anakin
    foaf:givenName "Anakin" ;
    foaf:familyName "Skywalker" ;
    foaf:name "Anakin Skywalker" .
```

validation

validator

www.w3.org/2013/ShEx/FancyShExDemo

executable examples

R-13-DISJOINT-GROUP-OF-PROPERTIES-CLASS-SPECIFIC