JSON syntax for input SHACL shapes

1 Example

The following JSON object represents the shape "`MovieShape`":

```
{
  "name": "MovieShape",
  "targetDef":{
    "query":"SELECT ?x WHERE {?x a dbo:Film}"
  },
  "constraintDef":{
    "conjunctions":[
      [
        { "path": "dbo:title", "min": 1, "max":1},
          { "path": "dbo:imdbId", "min": 1, datatype:"xsd:int"},
          { "path": "dbo:imdbId", "max":1},
          { "path": "dbo:director", "min": 1,
            "shape":"DirectorShape"},
          { "path": "dbo:starring", "max": 0,
            "shape":"ActorShape", "negated":true},
      ]
    ]
  }
}
```

Intuitively, this shape:

- has all instances of "`dbo:Film`" as targets,
- is verified by a node if:
  - it has exactly one title
  - it has exactly one imdb identifier, which is an integer
  - it has a director that verifies the shape "`DirectorShape`"
  - all its actors verify the shape "`ActorShape`"

2 Fields

The (mandatory) field "`name`" contains the name of the shape.

The (optional) field "`targetDef.query`" contains a monadic SPARQL query, in charge of retrieving the targets of the shape. The target must be bound to variable `?x`. A shape s without target definition is considered to have no target (i.e. $\text{targ}(s) = \bot$).

The JSON object "`constraintDef`" defines the constraint associated to the shape (i.e. $\text{def}(s)$). In addition to the constraint presented in the submitted article, we allow specifying the datatype of a node. To this end, the abstract syntax for shape constraints is extended with xsd datatypes as terminal symbols. The semantics is the one expected: e.g. a node v verifies the formula `xsd:int` iff v is a literal with datatype `xsd:int`. Otherwise, v violates `xsd:int`.

In the above example, the constraint for shape `MovieShape` is:

$$\begin{aligned}
&(=_1 \texttt{dbo:title}.\top) \wedge \\
&(\geq_1 \texttt{dbo:imdb.xsd:int}) \wedge \\
&(\leq_1 \texttt{dbo:imdb}.\top) \wedge \\
&(\geq_1 \texttt{dbo:director.DirectorShape}) \wedge \\
&(\leq_0 \texttt{dbo:starring}.\neg\texttt{ActorShape})
\end{aligned}$$

The constraint formula is assumed to be in disjunctive normal form, i.e. of the form $\phi_1 \vee .. \vee \phi_n$, where each ϕ_i is a conjunction of constraints. For instance, the formula in the above example contains only one disjunct.

Each disjunct ϕ_1 is a conjunction of *base expressions* of the form $\geq_m r.\phi'$, $\leq_n r.\phi'$ or $=_m r.\phi'$, where $m, n \in \mathbb{N}$, r is a SPARQL property path, and ϕ' is of the form ϕ'' or $\neg\phi''$, where ϕ'' is either $\top$, a datatype, a constant or a shape name.

The JSON field "`constraintDef.conjunctions`" is an array of arrays, one for each disjunct ϕ_i.

The array for ϕ_i contains JSON objects, one for each base expression (conjunct) of ϕ_i.

The JSON object for a base expression can contain the following attributes:

- "`path`": the property path r
- "`min`": the mininmal cardinality m if the expression is of the form $\geq_m r.\phi'$ or $=_m r.\phi'$
- "`max`": the maximal cardinality n if the expression is of the form $\leq_n r.\phi'$ or $=_n r.\phi'$
- "`negated`": `true` if ϕ' is of the form $\neg\phi''$, and `false` (default value) otherwise
- "`value`": the constant (i.e. an IRI) if ϕ'' is a constant
- "`shape`": the shape name if ϕ'' is a shape name

- "`datatype`": the xsd datatype if ϕ'' is a datatype

Attribute `path` is mandatory.
At least one of "`min`" or "`max`" must be present.
All other attributes are optional.

If attribute "`negated`" is absent, then it has value `false` by default.
If none of attributes "`value`", "`shape`" or "`datatype`" is present, then ϕ'' is considered to be $\top$ by default.