XXXXX

XXXXX[1] and XXXXX[2]

[1] XXXXX
XXXXX,
[2] XXXXX
XXXXX

Abstract. Keywords: ..

1 Idea

"What I would consider the greatest success for us would be to create something that can be mapped to any reasonable validation method, but that makes it easy for people to define the constraints they want to use in their application. BIBFRAME has used a somewhat reduced DSP in this way: the AP is what drives the BIBFRAME instance data editor. (I would love for us to have a similar editor for the *creation* of APs.)"

That's interesting! Kai and I had a related idea a couple of months ago ;-)

There should be 1 constraint language-independent SPIN mapping for each RDF validation constraint (requirement) in order to validate this constraint automatically and independently from any specific constraint language.

Then, for each constraint language (ShEx, DSP, OWL2) - we want to use in order to express our constraints, we need for each constraint a mapping to the language-independent constraint. This way, we do not need a SPIN mapping for each constraint language in order to implement the validation of the language-specific constraint. Now, we have immediately an automatic validation of these constraints. This way, constraints expressed with different languages can be validated in the same manner without differences.

When we use a UI to define our constraints in a user-friendly way, these constraints can be either mapped to a constraint expressed in a specific constraint language and then to SPIN or directly to SPIN in the background. So, the user does not have to know how to express constraints using a constraint language such as OWL 2, DSP, or ShEx.

And if there is a new constraint not covered by any constraint language you can just define the SPIN mapping for this constraint.

2 XX

ontology describing constraint languages interoperability between constraint languages express constraint in CL A and translate constraint in CL B provide SPARQL validation (SPIN mapping) for each general C

3 constraint language ontology

4 examples

4.1 min cardinality

OWL 2:

```
foaf:Person
    a owl:Restriction ;
                owl:minQualifiedCardinality "2"^^xsd:nonNegativeInteger ;
                owl:onProperty :hasName ;
                owl:onClass xsd:string .
```

general constraint: min cardinality constraint (OWL2): data min cardinality translation to SPARQL:

```

```

4.2 object property range

DSCL constraint (OWL 2):

```
:hasDog rdfs:range :Dog .
```

DSCL constraint (DSP):

```
:hasDogRange
        a dsp:DescriptionTemplate ;
        dsp:resourceClass owl:Thing ;
        dsp:statementTemplate [
            a dsp:NonLiteralStatementTemplate ;
            dsp:property :hasDog ;
            dsp:nonLiteralConstraint [
                a dsp:NonLiteralConstraint ;
                dsp:valueClass :Dog ] ] .
```

general constraint:

```
clo:objectPropertRange
    a clo:GeneralConstraint ;
                clo:requirement clo:R-28-OBJECT-PROPERTY-RANGE ;
                clo:DSCLConstraints ( clo:objectPropertRange_OWL2 clo:objectPropertRange_DSP ) ;
                clo:objectProperty :hasDog ;
                clo:range ( :Dog ) ;
```

valid data:

```
:Peter
    :hasDog :Brian .
:Brian
    a :Dog ;
    a owl:Thing .
```

invalid data:

```
:Peter
    :hasDog :Brian .
:Brian
#   a :Dog ; # commented --> constraint violation
    a owl:Thing .
```

general constraint: object property range constraint (OWL2): object property range

general C is translated to SPARQL for automatic validation

for each DSCL constraint you have to define the binding to the general constraint.

binding (DSCL constraint (OWL2) - general constraint):

```
# OWL 2 constraint
# -----
?OPE rdfs:range ?CE .

clo:objectProperty => ?OPE
clo:range => ?CE
```

binding (DSCL constraint (OWL2) - general constraint):

```
# DSP constraint
# -----
:hasDogRange
        a dsp:DescriptionTemplate ;
        dsp:resourceClass owl:Thing ;
        dsp:statementTemplate [
            a dsp:NonLiteralStatementTemplate ;
            dsp:property :hasDog ;
            dsp:nonLiteralConstraint [
                a dsp:NonLiteralConstraint ;
                dsp:valueClass :Dog ] ] .

# SPIN
# -----
?descriptionTemplate a dsp:DescriptionTemplate .
?descriptionTemplate dsp:resourceClass ?resourceClass .
?descriptionTemplate dsp:statementTemplate ?statementTemplate .
?statementTemplate dsp:property ?property .
?statementTemplate dsp:nonLiteralConstraint ?nonLiteralConstraint .
?nonLiteralConstraint dsp:valueClass ?valueClass .
FILTER EXISTS { ?nonLiteralConstraint dsp:valueClass ?valueClass . }
FILTER NOT EXISTS { ?object a ?valueClass . ?nonLiteralConstraint dsp:valueClass ?valueClass . }

# binding (DSCL constraint (OWL2) - general constraint)
# -----
clo:objectProperty => ?OPE
clo:range => ?CE
```

binding using SPIN

SPARQL (validation of general constraint):

```
?x ?clo:objectProperty ?this .
?this rdf:type owl:Thing .
FILTER NOT EXISTS { ?this a ?clo:range } .
```

SPARQL (validation of DSCL constraint (OWL2)):

```
# constraint
# -----
?OPE rdfs:range ?CE .

# data
# -----
?x ?OPE ?this .
?this rdf:type owl:Thing .
FILTER NOT EXISTS { ?this rdf:type ?CE } .
```

SPARQL (validation of DSCL constraint (DSP)):

```
# data
# -----
?this a ?resourceClass .
?this ?property ?object .
FILTER ( ?property != rdf:type ) .

# constraint
# -----
?descriptionTemplate a dsp:DescriptionTemplate .
?descriptionTemplate dsp:resourceClass ?resourceClass .
?descriptionTemplate dsp:statementTemplate ?statementTemplate .
?statementTemplate dsp:property ?property .
?statementTemplate dsp:nonLiteralConstraint ?nonLiteralConstraint .
?nonLiteralConstraint dsp:valueClass ?valueClass .
FILTER EXISTS { ?nonLiteralConstraint dsp:valueClass ?valueClass . }
FILTER NOT EXISTS { ?object a ?valueClass . ?nonLiteralConstraint dsp:valueClass ?valueClass . }
```

ontology design pattern.org constraint grammar constraint elements are the same DSP uses other design pattern / constraint / constraint elements / constraint design patterns / constraint language define terminology related to requirements DB / relate requirements to constraints, constraint languages constraint element: e.g. min cardinality dependency between requirements / when this requirements is fulfilled then this is also fulfilled (e.g. min card and requ. car) min car more powerful than req. propery why ontology? automatic translation from DSP to OWL define general constraint language we do not invent a new constraint languages requirements in SPARQL umsetzten only one validation for each requirement constraints may be composed

References