Negation & recursion validating RDF data

Jose Emilio Labra Gayo

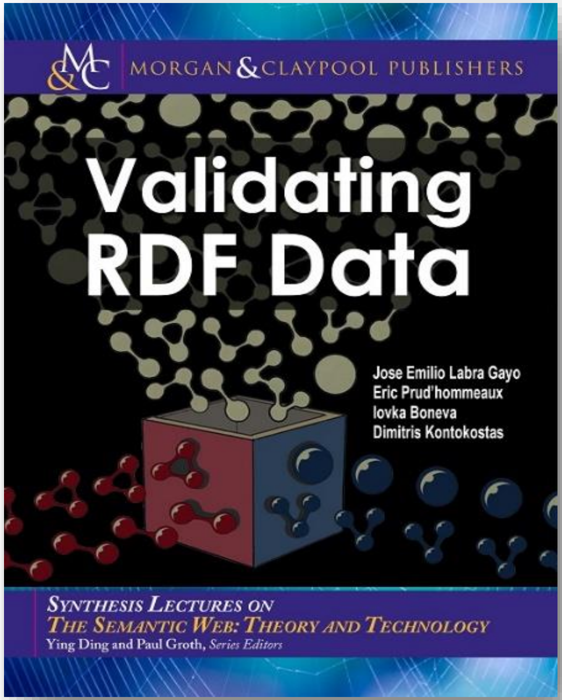
Departamento de Informática
Universidad de Oviedo

Background book

Validating RDF Data

Jose E. Labra, E. Prud'hommeaux, I. Boneva, D. Kontokostas

Morgan & Claypool, 2017



http://book.validatingrdf.com/

Validating RDF data

RDF = *lingua franca* of semantic web

- Simple Graph model
- Query language (SPARQL)
- Basis of knowledge representation (RDFS, OWL)
- Lots of applications based on RDF

But...

- Too much flexibility?
- Most RDF applications have latent schemas
- Tools to describe and validate those schemas?

RDF data validation history

Early initiatives

- 2000 – RDF Schema: lacks validation (only inference)
- 2004 – SPARQL: Queries can be used to validate
- 2007 – ICV: Modify OWL semantics with CWA
- 2010 – SPIN (by TopQuadrant): SPARQL templates

2013 – RDF Validation workshop (MIT, Boston)

- ShEx (Shape Expressions) proposed

2014 – W3C Data Shapes working group created

- Several inputs: ShEx, SPIN, …
- Decision: create new language called SHACL (Shapes Constraint Language)
- Difficulties to combine recursion, negation and other features

2017 – SHACL accepted as recommendation

- Decision: Leave recursion undefined in SHACL

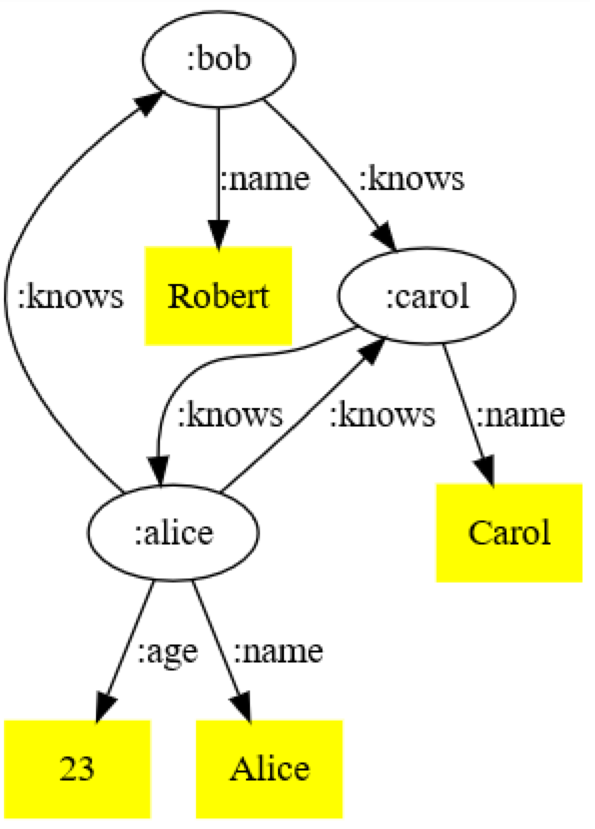
RDF data model

RDF data model is based on triples (statements)

- Subject – predicate – object

A set of triples forms an RDF graph

- Predicates are IRIs
- Subjects can be IRIs or blank nodes
- Objects can be IRIs, blank nodes or literals



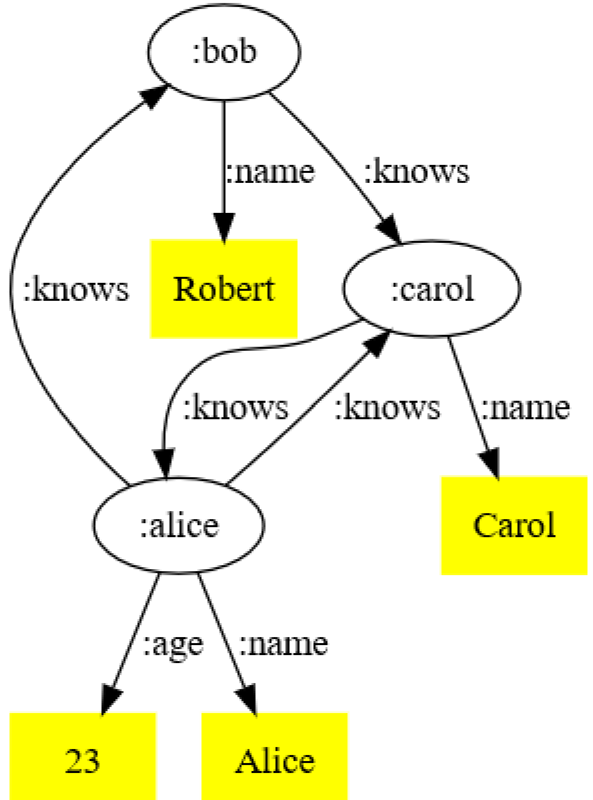
RDF playground: http://rdfshape.weso.es

ShEx (Shape Expressions)

Compact syntax

Regular expressions declare shapes of nodes

```
<Person> IRI {
 :name    xsd:string ;
 :age     xsd:integer ? ;
 :knows   @<Person>* ;
}
```



RDF playground: http://rdfshape.weso.es

Some ShEx features...

Recursion: shapes can refer to themselves

- Cyclic data models

Node constraints (constraints about a node)

- XML Schema facets

Triple constraints (neighbourhood of nodes)

Logical operations on shapes:

- AND, OR and NOT

```
<Teacher> @<Person> AND {
 :age MinInclusive 18
 :teaches @<Course>+
}

<Course> IRI {
 :code     xsd:string ;
 ^:teaches @<Teacher>
}

<Student> @<Person> AND {
 :enrolledIn  @<Course>+
} AND NOT {
 :knows @<Teacher>
}
```

Recursion and negation in ShEx

ShEx approach [Boneva et al, ISWC17]

- Stratified negation
  - Avoid schemas with negative cyclic dependencies
- According to the ShEx spec:
- "A schema must not contain any shapeExpr SE which has a shape which contains negated references, either directly or transitively, to SE."
- Note: Not all implementations force that constraint

[Boneva et al, 17] *Semantics and Validation of Shapes Schemas for RDF*, I. Boneva, Jose E. Labra, E. Prud'Hommeaux, International Semantic Web Conference, ISWC17

SHACL (Shapes Constraint Language)

Syntax based on RDF

Shapes = conjunctions of constraints

Common features between ShEx/SHACL

Several differences:

- Validation triggering mechanism
- Repeated properties
- Built-in support for inferencing
- Paths
- ....

```
<Person> IRI {
 :name    xsd:string ;
 :age     xsd:integer ? ;
 :knows   @<Person>* ;
}
```

```
:Person a sh:NodeShape ;
   sh:nodeKind sh:IRI ;
   sh:property [
    sh:path :name ;
    sh:minCount 1;
    sh:maxCount 1;
    sh:datatype xsd:string ;
  ] ;
  sh:property [
   sh:path :age  ;
   sh:maxCount 1;
   sh:datatype xsd:integer ;
  ] ;
  sh:property [
   sh:path :knows  ;
   sh:node :Person ;
  ] .
```

Recursion and negation in SHACL

- Recursion left out of SHACL recommendation
  - Recursive shapes have no standard semantics
  - Implementers can choose
    - Either ignore or use your own semantics
    - Undesired incompatibilities
    - Good as a research challenge
- Recent work [Corman et al'18] proposes partial assignments
  - 3-valued logics

[Corman et al, 18] *Semantics and validation of recursive SHACL*, J. Corman, J. L. Reutter, O. Savkovic, International Semantic Web Conference, Accepted in ISWC18

Current status of ShEx/SHACL

- ShEx/SHACL are increasingly being adopted
  - Some examples: EU multi-stakeholder, Wikidata, Clinical records,…
- Driven by practical applications & use cases
  - Although more theoretical work is needed
  - Negation/recursion may be defined in SHACL 2.0
  - Relating ShEx and SHACL languages
- Other challenges
  - Inheritance (a shape extends another shape)
  - Scalability and RDF streams validation
  - Visualization
  - …

Some intuitions…

Recursion/negation has been thoroughly studied in the ASP community

Maybe current ShEx/SHACL proposals could be improved

- Can ASP be applied to define ShEx/SHACL semantics?
- Converting ShEx/SHACL validators to ASP, is it feasible?

I started an exercise validating ShEx/SHACL in ASP (clingo)
More details?

Representing ShEx/SHACL in ASP

Possible ASP encoding of a simple shape

```
<User> {
 :name   xsd:string {1,1};
}
```

```
hasShape(X,user) :- node(X), string(X,name,1,1) .
not hasShape(X,user):- node(X), not string(X,name,1,1) .

string(X,P,MIN,MAX):- integer(MIN), integer(MAX),
  countStringProperty(X,P,C), C >= MIN , C <= MAX .
not string(X,P,MIN,MAX):- integer(MIN), integer(MAX),
  countProperty(X,P,C), countNoStringProperty(X,P,NS),
  C - NS < MIN .

countStringProperty(X,P,C):- node(X), property(P),
  C = #count { V: arc(X,P,V), string(V) }  .
countNoStringProperty(X,P,C):- node(X), property(P),
  C = #count { V: arc(X,P,V), not string(V) }  .
countProperty(X,P,C):- node(X), property(P),
  C = #count { V: arc(X,P,V) }  .

hasShape(X,A) | not hasShape(X,A) :- node(X), shape(A) .
```

```
arc(alice, name, "Alice").
arc(alice, knows, carol).
arc(bob,   name, "Robert").
arc(bob,   name, "Bob").
arc(carol, name, "Carol").
arc(carol, knows, alice).
arc(dave,  name, 2).
arc(emily, knows,dave).
```

https://goo.gl/sYcLqL
https://goo.gl/54kwZZ

```
clingo version 5.3.0
Solving...
Answer: 1
hasShape(alice,user) hasShape(carol,user)
```

Representing ShEx/SHACL in ASP

Adding recursion and conjunctions

```
<User> {
 :name    xsd:string {1,1};
} AND {
 :knows  @<User>     {1,*};
}
```

```
hasShape(X,user) :- string(X,name,1,1),
                    nodeShape(X,knows,user,1,star) .

not hasShape(X,user):- node(X), not string(X,name,1,1) .
not hasShape(X,user):- node(X),
                       not nodeShape(X,knows,user,1,star) .

nodeShape(X,P,S,MIN,MAX):- integer(MIN), integer(MAX), shape(S),
  countShapeProperty(X,P,S,C),
  C >= MIN, C <= MAX .
not nodeShape(X,P,S,MIN,MAX):-
  integer(MIN), integer(MAX), shape(S),
  countProperty(X,P,C), countNoShapeProperty(X,P,S,NS),
  C - NS < MIN .

countShapeProperty(X,P,S,C):- node(X), property(P), shape(S),
  C = #count { V: arc(X,P,V), hasShape(V,S) }  .
countNoShapeProperty(X,P,S,C):-
  node(X), property(P), shape(S),
  C = #count { V: arc(X,P,V), not hasShape(V,S) }  .
```

```
arc(alice, name, "Alice").
arc(alice, knows, carol).
arc(bob,   name, "Robert").
arc(bob,   name, "Bob").
arc(carol, name, "Carol").
arc(carol, knows, alice).
arc(dave,  name, 2).
arc(emily, knows,dave).
```

```
clingo version 5.3.0
Solving...
Answer: 1

Answer: 2
hasShape(alice,user) hasShape(carol,user)
SATISFIABLE

Models       : 2
```

Representing ShEx/SHACL in ASP

Adding negation

```
<User> {
  :name   xsd:string {1,1};
} AND {
  :knows  @<User>    {1,*};
}
<Teacher> {
  :name   xsd:string {1,1};
} AND NOT {
  :knows  @<User>    {1,*};
}
```

```
hasShape(X,user)  :- node(X),
                     string(X,name,1,1),
                     nodeShape(X,knows,user,1,star) .

hasShape(X,teacher)  :- node(X),
                     string(X,name,1,1),
                     not nodeShape(X,knows,user,1,star) .

not hasShape(X,user):- node(X),
                     not string(X,name,1,1) .

not hasShape(X,user):- node(X),
                     not nodeShape(X,knows,user,1,star) .

not hasShape(X,teacher):-node(X),
                     not string(X,name,1,1) .

not hasShape(X,teacher):- node(X),
                     nodeShape(X,knows,user,1,star) .
```

```
arc(alice, name, "Alice").
arc(alice, knows, carol).
arc(bob,   name, "Robert").
arc(bob,   name, "Bob").
arc(carol, name, "Carol").
arc(carol, knows, alice).
arc(dave,  name, 2).
arc(emily, knows,dave).
```

```
clingo version 5.3.0
Reading from userNameKnows.pl
Solving...
Answer: 1
hasShape(bob,teacher) hasShape(emily,teacher) hasShape(alice,teacher)
hasShape(carol,teacher)
Answer: 2
hasShape(bob,teacher) hasShape(emily,teacher) hasShape(alice,user)
hasShape(carol,user)
SATISFIABLE
Models       : 2
```

Language S

Minimal ShEx/SHACL

```
S ::= true
      | @S
      | datatype
      | IRI
      | S1 AND S2
      | NOT S
      | p S { min,max}
```

ϕ	$::=$	$\top$	true
	$\mid$	$@s$	reference to shape s
	$\mid$	Δ	node has datatype Δ
	$\mid$	IRI	node is an IRI
	$\mid$	$\phi_1 \wedge \phi_2$	conjunction
	$\mid$	$\neg\phi$	negation
	$\mid$	$\llcorner \xrightarrow{p} \phi\{min, max\}$	between min and max triples with predicate p whose values conform to ϕ

Example encoding ShEx in S

```
<User> {
 :name   xsd:string {1,1};
} AND {
 :knows  @<User>    {1,*};
}
<Teacher> {
 :name   xsd:string {1,1};
} AND NOT {
 :knows  @<User>    {0,*};
}
```

$$
\begin{aligned}
user &: \llcorner \xrightarrow{name} string\{1,1\} \quad \wedge \\
& \quad \llcorner \xrightarrow{knows} @user\{0,*\} \\
teacher &: \llcorner \xrightarrow{name} \top\{1,1\} \quad \wedge \\
& \quad \neg(\llcorner \xrightarrow{knows} @user\{0,*\})
\end{aligned}
$$

3-valued semantics for language S

The semantics is inspired by [Corman et al, 18] paper

The paper uses partial assignments

Which assign values $\mathcal{S}$ or $\neg\mathcal{S}$ to nodes during validation

$$
\begin{aligned}
[\![\top]\!]^{n,g,\sigma} &= 2 \\
[\![@s]\!]^{n,g,\sigma} &= [\![\sigma(s)]\!]^{n,g,\sigma} \\
[\![\Delta]\!]^{n,g,\sigma} &= \begin{cases} 2 & \text{if } n \in \Delta \\ 0 & \text{otherwise} \end{cases} \\
[\![IRI]\!]^{n,g,\sigma} &= \begin{cases} 2 & \text{if } n \text{ is an IRI} \\ 0 & \text{otherwise} \end{cases} \\
[\![\phi_1 \wedge \phi_2]\!]^{n,g,\sigma} &= min([\![\phi_1]\!]^{n,g,\sigma}, [\![\phi_2]\!]^{n,g,\sigma}) \\
[\![\neg\phi]\!]^{n,g,\sigma} &= 2 - [\![\phi]\!]^{n,g,\sigma} \\
[\![\llcorner \xrightarrow{p} \phi\{min, max\}]\!]^{n,g,\sigma} &= \begin{cases} 2 & \text{if } min \leq |(n,p,v) \in g \wedge [\![\phi]\!]^{v,g,schema} = 2| \leq max \\ 0 & \text{if } |(n,p,v) \in g| - |(n,p,v) \in g \wedge [\![\phi]\!]^{v,g,schema} = 0| < min \\ 1 & \text{otherwise} \end{cases}
\end{aligned}
$$

Exercise: Encoded that semantics in ASP...

Prototype implementation

Converts language S expressions to Clingo

Available at: http://labra.weso.es/shaclex/

Option `--clingoFile`

Example generated:

https://github.com/labra/shaclex/blob/master/examples/clingo/simple.pl