Datatypes for Lists and Maps in RDF Literals

Olaf Hartig[1, 2] Gregory Todd Williams[1] Michael Schmidt[1] Ora Lassila[1] Carlos Manuel Lopez Enriquez[1] Bryan Thompson[1]
[1]Amazon Neptune Team, Amazon Web Services, Seattle, WA, USA [2]Linköping University, Linköping, Sweden

What is the Problem?

In contrast to many other popular data representation forms and their query languages, RDF and SPARQL lack built-in support for generic types of composite values such as lists and maps. Instead, RDF introduces so-called containers and collections, which allow users to model composite values through a dedicated vocabulary *on top* of the core data model. Drawbacks:

- verbose representation, bloats up storage footprint
- cumbersome (even tricky) to query such containers & collections in SPARQL
- manipulation of such containers & collections in SPARQL even more complex

```
:srv71 :performance _:l1.
_:l1 rdf:type rdf:List .
_:l1 rdf:first  42.5 .
_:l1 rdf:rest _:l2 .
_:l2 rdf:first  41.9 .
_:l2 rdf:rest rdf:nil .
```

Listing: Example of a list of two values as an RDF collection.

Our Proposal

Dedicated datatypes for capturing lists and maps as literals; and corresponding extensions of SPARQL.

```
:srv71 :performance "[42.5, 41.9]"^^cdt:List .
```

Listing: The same list as above, captured as an RDF literal.

- Value space: sequences/functions over RDF terms and null (with some limitations)
- Lexical forms: Turtle based, including shorthands for literals; shorthands for nesting; superset of JSON

```
"['hello'@en, <http://liu.se>]"^^cdt:List
"{ 'id': 42, 'x': [4,null,7] }"^^cdt:Map
```

- Various functions to operate on these literals in SPARQL expressions

```
SELECT ?s (cdt:get(?list,1) AS ?v)
{ ?s :performance ?list .
  FILTER( cdt:size(?list) > 10 ) }
```

```
SELECT ?list WHERE {
  BIND( 1 AS ?x )
  BIND( cdt:List(?x, ?x+1, ?x+2) AS ?list ) }
```

- Aggregation function to produce such composite values in SPARQL

```
SELECT (FOLD(?name ORDER BY ?name) AS ?list)
WHERE { ?p rdf:type foaf:Person .
        ?p foaf:name ?name . }
```

```
SELECT (FOLD(?p,?name) AS ?map)
WHERE { ?p rdf:type foaf:Person .
        ?p foaf:name ?name . }
```

- New SPARQL operator to unfold composite values into their individual components

```
SELECT * { ?s :performance ?list .
           UNFOLD(?list AS ?elm, ?pos) }
ORDER BY ?s ?pos
```

Resources

- Formal **specification** of the approach
- Comprehensive **test suite** that covers all aspects of the specification
- ***Two* complete open source implementations** integrating support for the approach into the RDF programming frameworks Apache Jena (Java) and Attean (Perl)

ASSET AVAILABLE

spec repo

Broad Range of Use Cases

- Augmenting entities in a Knowledge Graph directly with corresponding **embedding vectors**
- Maintaining and operating over **lists** of all kinds; e.g., public transport timetables, series of measurements
- More direct interoperability between RDF graphs and Property Graphs that contain composite values
- Integration with other data ecosystems; e.g., **queries over JSON and CSV data** expressed directly within SPARQL, creation of JSON and CSV from SPARQL
- SPARQL as a ***bidirectional* mapping language** to describe mappings between RDF, JSON, and CSV

aws