Toward the
Web of Functions
Interoperable Higher-Order Functions
in SPARQL

Maurizio Atzori
Math/CS Department
University of Cagliari
Cagliari (Italy)
atzori@unica.it

International Semantic Web Conference (ISWC) - October 22, 2014
Outline

The talk is about **SPARQL functions**

- What we are currently missing
  - interoperability
  - expressivity

- How can we solve it
  - with the Web of Functions (wfn)
  - a function to rule them all *(as someone may know from the demo)*
  - various methods to achieve/implement it

- Conclusion
SPARQL functions

What we have:

● Standard SPARQL functions:
  ○ e.g., SUBSTR(), isURI(), STR(), functional IF(), bound()

● Operator extensibility
  ○ "allows implementations to return a result rather than raise an error"
  ○ User-defined functions (*Extensible Value Testing*)
    ■ they are represented as URIs
    ■ e.g., `afn:sqrt(num)`, `fn:contains()` (*Jena*)
    ■ e.g., `bif:contains()` (*Virtuoso*)
  ○ They are URIs
Limited Interoperability (1)

Functions are available only in some endpoints:

```PREFIX afn: <http://jena.hpl.hp.com/ARQ/function#>
...
BIND(afn:sqrt(9) as ?coolNumber )
```

<table>
<thead>
<tr>
<th>Apache Jena</th>
</tr>
</thead>
<tbody>
<tr>
<td>coolNumber</td>
</tr>
<tr>
<td>=============</td>
</tr>
<tr>
<td>3.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Virtuoso</th>
</tr>
</thead>
</table>
Limited Interoperability (2)

Even worst, the same URI can have different semantics!

PREFIX fn: <http://www.w3.org/2005/xpath-functions#>
SELECT * {
  # list of cool researchers :)
  VALUES (?s) {
    ("Tim Berners-Lee is cool")
    ("Nigel Shadbolt is cool")
    ("Maurizio Atzori mmm not sure")
  }

  BIND(fn:contains(?s, " is cool") as ?cool )
}
Limited Interoperability (2)

Apache Jena

<table>
<thead>
<tr>
<th>s</th>
<th>cool</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Tim Berners-Lee is cool&quot;</td>
<td>true</td>
</tr>
<tr>
<td>&quot;Nigel Shadbolt is cool&quot;</td>
<td>true</td>
</tr>
<tr>
<td>&quot;Maurizio Atzori mmm not sure&quot;</td>
<td>false</td>
</tr>
</tbody>
</table>

Virtuoso

<table>
<thead>
<tr>
<th>s</th>
<th>cool</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Tim Berners-Lee is cool&quot;</td>
<td>1</td>
</tr>
<tr>
<td>&quot;Nigel Shadbolt is cool&quot;</td>
<td>1</td>
</tr>
<tr>
<td>&quot;Maurizio Atzori mmm not sure&quot;</td>
<td></td>
</tr>
</tbody>
</table>
Important to stress that

1. These are not implementation bugs!
   - Both Jena and Virtuoso comply with SPARQL 1.1 in these examples

2. **Semantic (?) Web**
   - Data
     - There is a **Web of Data** where data is represented with URIs that have a unique semantics (*thanks Tim and all you SW guys*)
   - Functions (*Vision*)
     - There should be a **Web of Functions** where also functions' URIs have a unique semantics (*this is what this paper is about*)
Limited Expressivity (higher-order functions)

VALUES (?f) { (fn:contains)  
  (fn:concat) }

...  
BIND( ?f("Maurizio Atzori", " is cool") as ?cool )

<table>
<thead>
<tr>
<th>f</th>
<th>cool</th>
<th>desired result</th>
</tr>
</thead>
<tbody>
<tr>
<td>fn:contains</td>
<td>false</td>
<td></td>
</tr>
<tr>
<td>fn:concat</td>
<td>&quot;Maurizio Atzori is cool&quot;</td>
<td></td>
</tr>
</tbody>
</table>

Unfortunately, SPARQL does not allow to call a function whose URIs is in a variable.
Higher-Order Functions (examples)

Memoization

BIND( myf:memoize(myf:complex)(?arg) as ?res ) -> slow
BIND( myf:memoize(myf:complex)(?arg) as ?res ) -> fast

Map

BIND( myf:map(afn:max)(5,9,3) as ?res ) -> 9

Compose

BIND( compose(fn:abs, fn:round) as ?f )
BIND( ?f(-4.9) as ?res ) -> +5.0
Web of Functions

interoperable higher-order functions
call: a function to rule them all

We can implement a call function to:

- solve the interoperability problems
- solve the expressivity problem
- realize the Web of Functions!

PREFIX call: <http://webofcode.org/wfn/call>
call: a function to rule them all

VALUES (?f) { (fn:contains)
             (fn:concat) }
...
BIND( ?f("Maurizio Atzori", " is cool") as ?cool )
BIND( call:(?f, "Maurizio Atzori", " is cool") as ?res )

<table>
<thead>
<tr>
<th>f</th>
<th>cool</th>
</tr>
</thead>
<tbody>
<tr>
<td>fn:contains</td>
<td>false</td>
</tr>
<tr>
<td>fn:concat</td>
<td>&quot;Maurizio Atzori is cool&quot;</td>
</tr>
</tbody>
</table>

actual result
Semantics of the `call:` function

call:(function [, arg1, arg2, ...])

- `call` takes the given function (the first argument) and calls it with the given arguments
- calling a function $F$ means finding the right endpoint implementing $F$ and run it remotely
  - There are 3 alternatives in the paper
    - http://i-mozart.com/mywfn/answerToAllQuestions
    - http://i-mozart.com/mywfn/sparql
- we can also force the use of a different endpoint
  call:("myfunctionURI@endpointURI")
Architecture

user endpoint feat. call:
http://webofcode.org/wfn/sparql

"endpoint" of the function publisher
http://i-mozart.com/mywfn/sparql

PREFIX myf: <http://i-mozart.com/mywfn/>
BIND(
    call:(myf:answerToAllQuestions)
    as ?res )

Web of Function (happy) user
Publishing a function in the Web of Functions

1. deploy a SPARQL endpoint with your user defined functions

   OR

2. you just write a simple script in the language you prefer
   ○ 3 lines in PHP for an "hello world" function
Our implementation

● We implemented call: in Java, for Jena
  ○ it is opensource, see webofcode.org/wfn with
documentation and public git repository
  ○ we also deployed an Apache Jena endpoint at
    webofcode.org/wfn/sparql

● It's fast (introduces a delay of 25ms per call)
  ○ that is, the RPC approach is feasible
Alternative implementations of call: in the paper

- have user-defined function (Java for Jena, C for Virtuoso), OR

- take a pure-SPARQL approach: based on SPARQL Federated Queries
  - the solution is not trivial because arguments are not necessarily passed to the other endpoint according to Recommendations
Alternative implementations of call:

In the paper we provide two alternative:

1. having a proxy endpoint (endpoint-in-the-middle) featuring call:
   a. requiring a special proxy implementing call:
   b. require one non-normative parts to be implemented in the caller

2. a pure 100% SPARQL approach
   a. no proxy, no Java, no C, SPARQL-only
   b. require two non-normative parts to be implemented in the caller
   c. require more constraints on the publisher endpoint
Conclusions

through the Web of Functions we can

- as user: call any function in the Web of Functions (+10K functions, see demo paper)
- as publishers: implement a function quickly and in any language, without deploying a triple store server

**Web of Functions** can be realized without changing the current SPARQL specifications. Actually, it does exist already.
References

1. http://webofcode.org/wfn


