Overview and Review by Jesse Weaver

Reviewing XSPARQL: Traveling between the XML and RDF worlds – and avoiding the XSLT pilgrimage

By Waseem Akhtar, Jacek Kopecky, Thomas Krennwallner, and Axel Polleres
Overview

• Lifting and Lowering
  – Using XSLT is too cumbersome for complex cases.
  – Lowering from RDF/XML is too difficult.

• GRDDL and SAWSDL

• XQuery and SPARQL

• XSPARQL

• Implementation
Lifting and Lowering

Fig. 2. From XML to RDF and back: “lifting” and “lowering”
GRDDL and SAWSDL

- GRDDL XSLT (for lifting data from XML data)
- SAWSDL XSLT (for lifting and lowering for WSDL)
Fig. 3. Lifting attempt by XSLT
Different RDF/XML Representations

(a) @prefix alice: <alice/> .
   @prefix foaf: <...foaf/0.1/> .

   alice:me a foaf:Person.
   alice:me foaf:knows :c.
   :c a foaf:Person.
   :c foaf:name "Charles".

(b) <rdf:RDF xmlns:foaf="...foaf/0.1/"
   xmlns:rdf="...rdf-syntax-ns#">
   <foaf:Person rdf:about="alice/me">
     <foaf:knows>
       <foaf:Person foaf:name="Charles"/>
     </foaf:knows>
   </foaf:Person>
 </rdf:RDF>

(c) <rdf:RDF xmlns:foaf="...foaf/0.1/"
   xmlns:rdf="...rdf-syntax-ns#">
   <rdf:Description rdf:nodeID="x">
     <rdf:type rdf:resource=".../Person"/>
     <foaf:name>Charles</foaf:name>
   </rdf:Description>
   <rdf:Description rdf:about="alice/me">
     <foaf:knows rdf:nodeID="x"/>
   </rdf:Description>
 </rdf:RDF>

(d) <rdf:RDF xmlns:foaf="...foaf/0.1/"
   xmlns:rdf="...rdf-syntax-ns#">
   <rdf:Description rdf:nodeID="x">
     <rdf:type rdf:resource=".../Person"/>
     <foaf:name>Charles</foaf:name>
   </rdf:Description>
   <rdf:Description rdf:about="alice/me">
     <foaf:knows rdf:nodeID="x"/>
   </rdf:Description>
 </rdf:RDF>

Fig. 1. Different representations of the same RDF graph
Lowering with XSLT
(for RDF/XML syntax like 1b)

```xml
<xsl:stylesheet version="1.0" xmlns:rdf="...rdf-syntax-ns#"
    xmlns:foaf="...foaf/0.1/" xmlns:xsl="...XSL/Transform">
  <xsl:template match="/rdf:RDF">
    <relations><xsl:apply-templates select="./foaf:Person"/></relations>
  </xsl:template>
  <xsl:template match="foaf:Person"><person name="/@foaf:name">
    <xsl:apply-templates select="/@foaf:knows"/>
  </person></xsl:template>
  <xsl:template match="foaf:knows[@rdf:nodeID]"> <knows>
    <xsl:value-of select="/foaf:Person[.@rdf:nodeID=./@rdf:nodeID]/@foaf:name"/>
  </knows></xsl:template>
</xsl:stylesheet>
```

Fig. 5. Lowering attempt by XSLT (mylowering.xsl)
XQuery and SPARQL
(a) XQuery

declare namespace foaf="http://xmlns.com/foaf/0.1/";
declare namespace rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#";
let $persons := //*[@name or ./.knows]
return <rdf:RDF>
  {
    for $p in $persons
      let $n := if( $p[@name] )
          then $p/@name else $p
      let $id := count($p/preceding::*)
          + count($p/ancestor::*);
      where 
          not(exists($p/following::*[
              @name=$n or data(.)=$n]))
    return <foaf:Person rdf:nodeID="b{$id}"/>
    <foaf:name>{data($n)}</foaf:name>
    {
      for $k in $persons
        let $kn := if( $k[@name] )
          then $k/@name else $k
        let $kid := count($k/preceding::*)
          + count($k/ancestor::*);
        where
          $kn = data(//*[[@name=$n]/knows) and
          not(exists($kn/..following::*[
              @name=$kn or data(.)=$kn]))
        return <foaf:knows>
          <foaf:Person rdf:nodeID="b{$kid}"/>
        </foaf:knows>
    }
  }
</foaf:Person>
</rdf:RDF>

(b) XSPARQL

declare namespace foaf="http://xmlns.com/foaf/0.1/";
declare namespace rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#";
let $persons := //*[@name or ./.knows]
return
for $p in $persons
  let $n := if( $p[@name] )
      then $p/@name else $p
  let $id := count($p/preceding::*)
          + count($p/ancestor::*);
  where 
      not(exists($p/following::*[
          @name=$n or data(.)=$n]))
  construct {
    .b{$id} a foaf:Person;
    foaf:name {data($n)};
    
    {
      for $k in $persons
        let $kn := if( $k[@name] )
          then $k/@name else $k
        let $kid := count($k/preceding::*)
          + count($k/ancestor::*);
        where
          $kn = data(//*[[@name=$n]/knows) and
          not(exists($kn/..following::*[
              @name=$kn or data(.)=$kn]))
        construct {
          .b{$id} foaf:knows .b{$kid}.
          .b{$kid} a foaf:Person.
        }
      }
  }
</foaf:Person>

Fig. 7. Lifting using XQuery and XSPARQL
<relations>
{
  for $Person $Name
  from <relations.rdf>
  where {
    $Person foaf:name $Name
  }
  order by $Name
  return
  <person name="{$Name}">
  {
    for $Fname
    from <relations.rdf>
    where {
      $Person foaf:knows $Friend .
      $Person foaf:name $Name .
      $Friend foaf:name $Fname
    }
    return
    <knows>{$Fname}</knows>
  }
  </person>
}
</relations>
Implementation

Fig. 10. XSPARQL architecture
Review

• Relevance: 8
• Significance: 9
• Technical Soundness: 7
• Novelty: 9
• Quality of Evaluation: 6
• Clarity: 9
• Overall: 8
• Reviewer Confidence: 6
Relevance
(to Semantic Web field)

• 8 – This paper focuses on lifting and lowering semantic web data.
  – While this is a subarea of SW, it does not seem specialized enough to constitute a 7 because there exist other major efforts to solve this problem (e.g., SAWSDL, GRDDL).
  – Because there are many broader subareas of SW (e.g., reasoning, querying, etc.), and because it does not seem relevant to any of these larger subareas, it does not constitute a 9.
Significance

• 9 – This approach advances the state of the art giving a pragmatic approach to lowering and lifting semantic web data, especially for lowering.

• Personally believed to be more flexible than XSLT, but depends upon user’s familiarity with XQuery and SPARQL.

• Can really contribute to the availability of semantic web data.
Technical Soundness

• 7 – The paper’s presented solution seems technically sound, but personal lack of familiarity with XQuery prevents me from certainty.

• Major oversight: handling typed literals or literals with language tags in Fig. 11.

• Not much given to assert the technical soundness of this approach in the form of proofs or empirical evidence. However, reference to demoable software is made.
Novelty

• 9 – Seems very novel compared to related work presented in paper.
• Combines standard query techniques of XQuery for XML data and SPARQL for RDF data.
• Lack of personal familiarity with literature in this field prevents me from giving a 10.
• Previous approaches depend heavily on more established XML technologies like XSLT and XPath.
Quality of Evaluation

• 6 – The overall goal is to make lifting and lowering easier. If users are familiar with XQuery and SPARQL, then they have obviously achieved this goal.

• It seems likely that there are more people familiar with XSLT than XQuery and SPARQL as XSLT has been around longer.

• No evaluation is given that shows that persons involved in lifting and lowering found this approach easier.
Clarity

• 9 – Mostly very clear with a few personal exceptions.
  – Discussion of XQuery as well as the algorithm presented for rewriting XSPARQL into XQuery were unclear; this is probably due to my lack of familiarity with XQuery.
  – However, this keeps me from rating 10.
Overall

• 8 – The average of other individual scores giving double weight to novelty.
• Seems to match reviewer’s intuition of overall quality.
• Lack of experience in reviewing makes determining “top X% of accepted papers” difficult to determine.
Reviewer Confidence

- 6 – I did not understand small parts specific to XQuery, and I certainly am not very familiar with research in this area.
- Presented algorithm for query rewriting was not checked in detail.
- Lack of reviewing papers decreases confidence in the overall score.
Questions?