Full Untyped XQuery Canonization

Nicolas Travers\textsuperscript{1}  Tuyêt Trâm Dang Ngoc\textsuperscript{2}  Tianxiao Liu\textsuperscript{2}

(1) PRiSM Laboratory-University of Versailles, France
(2) ETIS Laboratory - University of Cergy-Pontoise, France

16 June 2007

Webetrends - APWeb/WAIM workshop 2007
# Plan

1. Context
2. Existing works
3. Canonization rules
4. Conclusion
1 Context
   - XQuery

2 Existing works

3 Canonization rules

4 Conclusion
XQuery: A rich syntax

```
declare function local:f($doc as xs:string) as element()
{
    for $x in (doc("rev.xml")/review|doc("$doc")/catalog)
        [. contains("Robin Hobb")]/book/.[//price > 15]
    where
        some $y in $x/comments
        satisfies contains ($y, "Excellent")
    order by $x/@isbn
    return
        <book>
            {$x/@isbn}
            <price>{$x//price/text()}</price>
            {
                if (count($x/title) > 2)
                then
                {
                    for $z in doc("books.xml")/book
                        where $z/@isbn = $x/@isbn
                        return <title>{$z/title}[3]</title>
                }
                else<title/>
            }
        </book>
}
```

- XPath;
- Constraints;
- Filters;
- Quantifiers;
- Document construction;
- Nesting;
- Aggregates;
- Conditional operators;
- Set operators;
- Sorts;
- Sequences;
- Functions;
Equivalent forms (W3C specifications):

```
for $i in doc("cat.xml")/catalog/book
where $i//author = "Robin Hobb"
   and $i/title = "King's spy"
return
   <price>
       {$i/price/text()}
   </price>
```

```
for $i in doc("cat.xml")/catalog/book[
    ./title = "King's spy"]
where $i//author = "Robin Hobb"
return
   <price>
       {$i/price/text()}
   </price>
```
Equivalent forms (W3C specifications):

for $i$ in doc("cat.xml")/catalog/book
where $i$/author = "Robin Hobb"
and $i/title = "King's spy"
return
<price>
    {$i/price/text()}
</price>

for $i$ in doc("cat.xml")/catalog/book[./title = "King's spy"]
where $i$/author = "Robin Hobb"
return
<price>
    {$i/price/text()}
</price>

Need to define a unique form for all XQuery specifications.
1. Context

2. Existing works
   - XPath
   - NEXT
   - Galax
   - Canonization rules

3. Canonization rules

4. Conclusion
XPath axes canonization [Olteanu et al. 2002]

XPath canonization rules:

- parent::n ;
- ancestor::n ;
- ancestor-or-self::n ;
- descendant-or-self::n ;

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
XPath axes canonization [Olteanu et al. 2002]

XPath canonization rules:

- `parent::*`;
- `ancestor::*`;
- `ancestor-or-self::*`;
- `descendant-or-self::*`;

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
</table>
XPath axes canonization [Olteanu et al. 2002]

XPath canonization rules:
- `parent::*n`;
- `ancestor::*n`;
- `ancestor-or-self::*n`;
- `descendant-or-self::*n`;

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
</table>
XPath axes canonization [Olteanu et al. 2002]

XPath canonization rules:

- parent::n;
- ancestor::n;
- ancestor-or-self::n;
- descendant-or-self::n;

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
</table>
XPath axes canonization [Olteanu et al. 2002]

XPath canonization rules:

- parent::n;
- ancestor::n;
- ancestor-or-self::n;
- descendant-or-self::n;

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
<tbody>
<tr>
<td>/catalog/book/descendant-or-self::title</td>
<td>/catalog/book/(.</td>
</tr>
</tbody>
</table>
NEXT queries [Deustch et al. 2004]

- Transformations for XQuery;
- For strong nested oriented queries;
- New types of clause: "groupby":

```xml
for $a in distinct-values($doc//book//title/author)
return
    <bibentry>
        {$a,
        for $b in $doc//book,
            $a1 in $b/author,
            $t in $b/title
            where $a1 eq $a
            groupby [$b], [$t]
        return $t
    }
</bibentry>
```
The Galax experience [Fernández et al. 2003]

- Navigational based XQuery processing system;
- Fully support by rewriting XQuery expressions;
- Series of nested loops for normalization;

```
snap {
  element results {
    for $b in
      fs:distinct-docorder(
        for $fs:dot in
          fs:distinct-docorder(for $fs:dot in $bib return child::bib)
        return child::book
      )
    return
      element result {
        fs:distinct-docorder(let $fs:dot := $b return child::title),
        fs:distinct-docorder(let $fs:dot := $b return child::author)
      }  
  }
}
```
Canonization rules [Chen 2004]

Existing rules:
- Filters;
- Nesting;
- Aggregates;
- Quantifiers.

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
# Canonization rules

## Existing rules:
- Filters;
- Nesting;
- Aggregates;
- Quantifiers.

### XQuery query

```xml
for $i$ in doc("cat.xml")/catalog/book[@isbn="12351234"]
return {$i}
```

### Canonical XQuery query

```xml
for $j$ in doc("cat.xml")/catalog/book where $j/@isbn = "12351234"
return {$j}
```
Canonization rules [Chen 2004]

Existing rules:
- Filters;
- Nesting;
- Aggregates;
- Quantifiers.

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $i$ in doc(&quot;cat.xml&quot;)/catalog/book[@isbn=&quot;12351234&quot;]/title return {$i}</td>
<td>for $j$ in doc(&quot;cat.xml&quot;)/catalog/book for $i$ in $j/title where $j/@isbn = &quot;12351234&quot; return {$i}</td>
</tr>
</tbody>
</table>
Canonization rules [Chen 2004]

Existing rules:
- Filters;
- Nesting;
- Aggregates;
- Quantifiers.

```
XQuery query       Canonical XQuery query
for $i in doc("cat.xml")/catalog/book
  return
  <book>
    {for $j in $i/title return { $j } }
  </book>
```

```
for $i in doc("cat.xml")/catalog/book
let $l :=
  (for $j in $i/title return { $j } )
return <book>{$l}</book>
```
**Canonization rules [Chen 2004]**

Existing rules:
- Filters;
- Nesting;
- Aggregates;
- Quantifiers.

### XQuery query vs Canonical XQuery query

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
<tbody>
<tr>
<td>return $&lt;count&gt;$ count($i/author) $&lt;/count&gt;$</td>
<td>let $l := $count($i/author)</td>
</tr>
<tr>
<td></td>
<td>return $&lt;count&gt;$ $l$ $&lt;/count&gt;$</td>
</tr>
</tbody>
</table>

N. Travers, T. T. Dang-Ngoc, T. Liu

Full Untyped XQuery Canonization
Existing rules:

- Filters;
- Nesting;
- Aggregates;
- Quantifiers.

### XQuery query

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $i$ in doc(&quot;cat.xml&quot;):catalog/book where some $s$ in $i$/price satisfies $s &gt; 15$ return {$i$}</td>
<td>for $i$ in doc(&quot;cat.xml&quot;):catalog/book let $l$ := (for $s$ in $i$/price where $s &gt; 15$ return {$s$}) where count($l$) &gt; 0 return {$i$}</td>
</tr>
</tbody>
</table>
Canonization rules [Chen 2004]

Existing rules:
- Filters;
- Nesting;
- Aggregates;
- Quantifiers.

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $i$ in doc(&quot;cat.xml&quot;)/catalog/book where every $s$ in $i$/price satisfies $s &gt; 15 return {$i}</td>
<td>for $i$ in doc(&quot;cat.xml&quot;)/catalog/book let $l :=$</td>
</tr>
<tr>
<td></td>
<td>(for $s$ in $i$/price where $s &lt;= 15 return {$s}) where count($l) = 0 return {$i}</td>
</tr>
</tbody>
</table>
1 Context

2 Existing works

3 Canonization rules
   - New canonization rules
   - Example

4 Conclusion
Untyped XQuery queries

Canonical XQuery [Chen 04]:

- XPath;
- Constraints;
- Filters;
- Quantifiers;
- Document construction;
- Nesting;
- Aggregates.

Need rules for:

- Sorts;
- Set operators;
- Conditional operators;
- Sequences;
- Functions.
Untyped XQuery queries

Canonical XQuery [Chen 04]:

- XPath;
- Constraints;
- Filters;
- Quantifiers;
- Document construction;
- Nesting;
- Aggregates.

Need rules for:

- Sorts;
- Set operators;
- Conditional operators;
- Sequences;
- Functions.
New canonization rules

Canonization rules

New canonization rules:
- Sorts;
- Set operators (intersect, union, except);
- Conditional operators;
- Sequences;
- Functions;

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
New canonization rules

Canonization rules

New canonization rules:

- **Sorts**;
- Set operators (intersect, union, except);
- Conditional operators;
- Sequences;
- Functions;

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $i in /catalog/book order by $i/title return $i/title</td>
<td>for $i in /catalog/book let $j := orderby ($i, $i/title) for $k in $j return $k/title</td>
</tr>
</tbody>
</table>
New canonization rules:

- Sorts;
- Set operators (intersect, union, except);
- Conditional operators;
- Sequences;
- Functions;

**XQuery query**

```xquery
for $i$ in (/catalog|/review)/book
return $i/title
```

**Canonical XQuery query**

```xquery
let $i_3 :=
  for $i_1$ in /catalog
  for $i_2$ in /review
  return ($i_1 | $i_2)$
for $i$ in $i_3/book$
return $i/title$
```
Canonization rules

New canonization rules:
- Sorts;
- Set operators (intersect, union, except);
- Conditional operators;
- Sequences;
- Functions;

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $i$ in /catalog/book return</td>
<td>for $i$ in /catalog/book</td>
</tr>
<tr>
<td>{if contains ($i/author, &quot;Hobb&quot;) then ( for $j$ in $i//title return $j ) }</td>
<td>let $l := for $j$ in $i//title return $j</td>
</tr>
<tr>
<td>else ( $i/author )}</td>
<td>return</td>
</tr>
<tr>
<td></td>
<td>{if contains ($i/author, &quot;Hobb&quot;)</td>
</tr>
<tr>
<td></td>
<td>then ( $l )</td>
</tr>
<tr>
<td></td>
<td>else ( $i/author )}</td>
</tr>
</tbody>
</table>
New canonization rules:

- Sorts;
- Set operators (intersect, union, except);
- Conditional operators;
- Sequences;
- Functions;

<table>
<thead>
<tr>
<th>XQuery query</th>
<th>Canonical XQuery query</th>
</tr>
</thead>
<tbody>
<tr>
<td>for $i$ in (/catalog/book)[2] return $i/title</td>
<td>let $i_1 := for $x$ in /catalog/book return $x$</td>
</tr>
<tr>
<td></td>
<td>for $i$ in $i_1$</td>
</tr>
<tr>
<td></td>
<td>where $i/position() == 2 return $i/title</td>
</tr>
</tbody>
</table>
New canonization rules

New canonization rules:
- Sorts;
- Set operators (intersect, union, except);
- Conditional operators;
- Sequences;
- Functions;

### XQuery query
```xquery
declare function local:section ($i as element() ) as element ()*
{
    for $j in $i/book
    return
        <book>
            {$j/title}
        for $s in $i/section/title
        return <section>{$s/text()}</section>
    </book>
}
for $f in doc("catalog.xml")/catalog
return local:section($f)
```

### Canonical XQuery query
```xquery
declare function local:section ($i as element() ) as element ()*
{
    for $j in $i/book
    return
        <book>
            {$j/title}
        let $l := (for $s in $i/section/title
        return <section>{$s/text()}</section>)
        return <section>{$l}</section>
    </book>
}
for $f in doc("catalog.xml")/catalog
return local:section($f)
```
Illustrating example

for $x$ in (doc("rev.xml")/review | doc("$doc")/catalog)
    [. contains("Robin Hobb")]/book[.//price > 15]
where
    some $y$ in $x$/comments
        satisfies contains ($y", "Excellent")
order by $x/@isbn
return
<book>
  {$x/@isbn}
  <price>{$x//price/text()} </price>
  {
    if (count($x/title) > 2)
    then
    {
      for $z$ in doc("books.xml")/book
          where $z/@isbn = $x/@isbn
              return <title>{($z/title)[3]} </title>
    }
    else <title/>
  }
</book>

1. Set operator ($l_1$);
2. Bind filtered xpathes ($f_3$);
3. Transform filters (constraints);
4. Transform quantifiers ($l_2$);
5. Transform sorts ($l_3$ & $f_4$);
6. Prepare aggregate ($l_4$);
7. Prepare nested queries ($l_5$);
8. Transform sequences ($l_6$).
let $l1 := (for $f1 in doc("rev.xml")/review
    for $f2 in doc("$doc")/catalog
    return ($f1 | $f2))
for $x in $l1[. contains("Robin Hobb")]/book[./price > 15]
where
    some $y in $x/comments
    satisfies contains ($y, "Excellent")
order by $x/@isbn
return <book>
    {$x/@isbn}
    <price>{$x//price/text()}
    </price>
    {if (count($x/title) > 2)
        then
        {
            for $z in doc("books.xml")/book
                where $z/@isbn = $x/@isbn
                return <title>{$z/title}[3]
        }
        else <title/>
    }
</book>

1. Set operator ($l1);
2. Bind filtered xpaths ($f3);
3. Transform filters (constraints);
4. Transform quantifiers ($l2);
5. Transform sorts ($l3 & $f4);
6. Prepare aggregate ($l4);
7. Prepare nested queries ($l5);
8. Transform sequences ($l6).
Illustrating example

let $l1 := (for $f1 in doc("rev.xml")/review
       for $f2 in doc("$doc")/catalog
       return ($f1 | $f2))

for $f3 in $l1[. contains("Robin Hobb")], $x in $f3/book[./price > 15]
where
    some $y in $x/comments
    satisfies contains ($y, "Excellent")
order by $x/@isbn
return
  <book>
    {$x/@isbn}
    <price>{$x//price/text()}</price>
    {if (count($x/title) > 2)
      then
        {for $z in doc("books.xml")/book
          where $z/@isbn = $x/@isbn
          return <title>{{$z/title}[3]}</title>
        }
      else <title/>
    }
  </book>

1 Set operator ($l1) ;
2 Bind filtered xpathes ($f3) ;
3 Transform filters (constraints) ;
4 Transform quantifiers ($l2) ;
5 Transform sorts ($l3 & $f4) ;
6 Prepare aggregate ($l4) ;
7 Prepare nested queries ($l5) ;
8 Transform sequences ($l6) .
Illustrating example

let $l1 := (for $f1 in doc("rev.xml")/review
   for $f2 in doc("$doc")/catalog
   return ($f1 | $f2))
for $f3 in $l1, $x in $f3/book
where contains($f3, "Robin Hobb") and $x//price > 15 and
   some $y in $x/comments
   satisfies contains ($y, "Excellent")
order by $x/@isbn
return
   <book>
   {$x/@isbn}
   <price>{$x//price/text()} </price>
   {
     if (count($x/title) > 2)
     then
     {
       for $z in doc("books.xml")/book
         where $z/@isbn = $x/@isbn
         return <title>{($z/title)[3]} </title>
     }
     else <title/>
   }
</book>

1. Set operator ($l1);
2. Bind filtered xpathes ($f3);
3. Transform filters (constraints);
4. Transform quantifiers ($l2);
5. Transform sorts ($l3 & $f4);
6. Prepare aggregate ($l4);
7. Prepare nested queries ($l5);
8. Transform sequences ($l6).
Illustrating example

let $l1 := (for $f1 in doc("rev.xml")/review
    for $f2 in doc("$doc")/catalog
    return ($f1 | $f2))

let $l2 := (for $y in $x/comments
    where contains ($y, "Excellent")
    return $y)

for $f3 in $l1, $x in $f3/book
where contains($f3, "Robin Hobb") and $x//price > 15 and count ($l2) > 0
order by $x/@isbn
return

  <book>
    {$x/@isbn}
    <price>{$x//price/text()}</price>
    {if (count($x/title) > 2)
     then
     {
        for $z in doc("books.xml")/book
        where $z/@isbn = $x/@isbn
        return <title>{$($z/title)[3]}</title>
     }
     else <title/>
    }
  </book>
Illustrating example

let $l_1 := (\text{for } f_1 \text{ in doc("rev.xml")/review}
\text{for } f_2 \text{ in doc("$doc")/catalog}
\text{return ($f_1 \mid f_2))/}
let $l_2 := (\text{for } y \text{ in } f_4/\text{comments}
\text{where contains (y, "Excellent")}
\text{return y)})
for $f_3 \text{ in } l_1, f_4 \text{ in } f_3/\text{book}
\text{where contains($f_3, "Robin Hobb") and } f_4/\text{price} > 15 \text{ and count ($l_2)} > 0
let $l_3 := \text{orderby ($f_4, $f_4/@isbn)}$
for $x \text{ in } l_3$
return
\text{<book>}
\{x/@isbn\}
\text{<price>}\{x/price/text()\} \text{</price>}
\{if (count(x/title) > 2)
\text{then}
\{for $z \text{ in } \text{doc("books.xml")/book}
\text{where } z/@isbn = x/@isbn
\text{return } \text{<title>}\{z/title}[3]\text{</title>}
\}
\text{else } \text{<title/>}
\}
\text{</book>}

1. Set operator ($l_1$);
2. Bind filtered xpathes ($f_3$);
3. Transform filters (constraints);
4. Transform quantifiers ($l_2$);
5. Transform sorts ($l_3$ & $f_4$);
6. Prepare aggregate ($l_4$);
7. Prepare nested queries ($l_5$);
8. Transform sequences ($l_6$).
Illustrating example

let $l1 := (for $f1 in doc("rev.xml")/review
     for $f2 in doc("$doc")/catalog
       return ($f1 | $f2))
let $l2 := (for $y in $f4/comments
       where contains ($y, "Excellent")
       return $y)
for $f3 in $l1, $f4 in $f3/book
  where contains($f3, "Robin Hobb") and $f4//price > 15 and count ($l2) > 0
let $l3 := orderby ($f4, $f4/@isbn)
for $x in $l3
  let $l4 := count ($x/title)
  return
    <book>
      {$x/@isbn}
      <price>{$x//price/text()} </price>
    </book>
  {if ($l4 > 2)
   then
     {for $z in doc("books.xml")/book
      where $z/@isbn = $x/@isbn
        return <title>{$(z/title)[3]}</title>}
   else <title/></title>
  }
</book>

1. Set operator ($l1);
2. Bind filtered xpathes ($f3);
3. Transform filters (constraints);
4. Transform quantifiers ($l2);
5. Transform sorts ($l3 & $f4);
6. Prepare aggregate ($l4);
7. Prepare nested queries ($l5);
8. Transform sequences ($l6).
Illustrating example

let $l1 := (for $f1 in doc("rev.xml")/review
    for $f2 in doc("$doc")/catalog
    return ($f1 | $f2))

let $l2 := (for $y in $f4/comments
    where contains ($y, "Excellent")
    return $y)

for $f3 in $l1, $f4 in $f3/book
    where contains($f3, "Robin Hobb") and $f4//price > 15 and count ($l2) > 0
    let $l3 := orderby ($f4, $f4/@isbn)
    for $x in $l3
        let $l4 := count ($x/title)
        let $l5 := (for $z in doc("books.xml")/book
            where $z/@isbn = $x/@isbn
            return <title>{($z/title)[3]}</title>)
    return
        <book>
            {$x/@isbn}
            <price>{$x//price/text()}
            {if ($l4 > 2)
                then {$l5}
                else <title/>
            }
        </book>

1 Set operator ($l1);
2 Bind filtered xpathes ($f3);
3 Transform filters (constraints);
4 Transform quantifiers ($l2);
5 Transform sorts ($l3 & $f4);
6 Prepare aggregate ($l4);
7 Prepare nested queries ($l5);
8 Transform sequences ($l6).
Illustrating example

let $l1 := (for $f1 in doc("rev.xml")/review
    for $f2 in doc("$doc")/catalog
    return ($f1 | $f2))
let $l2 := (for $y in $f4/comments
    where contains ($y, "Excellent")
    return $y)
for $f3 in $l1, $f4 in $f3/book
where contains($f3, "Robin Hobb") and $f4//price > 15 and count ($l2) > 0
let $l3 := orderby ($f4, $f4/@isbn)
for $x in $l3
let $l4 := count ($x/title)
let $l5 := (let
    $l6 := (for $z in doc("books.xml")/book
        where $z/@isbn = $x/@isbn
        return $z)
    for $f5 in $l6/title
        where $f5/position () = 3
        return <title>{$f5}</title>)
return
    <book>
    {$x/@isbn}
    <price>{$x//price/text()}<price>
    {if ($l4 > 2)
        then {$l5}
        else <title/>
    }
</book>

1. Set operator ($l1) ;
2. Bind filtered xpaths ($f3) ;
3. Transform filters (constraints) ;
4. Transform quantifiers ($l2) ;
5. Transform sorts ($l3 & $f4) ;
6. Prepare aggregate ($l4) ;
7. Prepare nested queries ($l5) ;
8. Transform sequences ($l6).
1 Context

2 Existing works

3 Canonization rules

4 Conclusion
Conclusion

Thanks to this canonization rules:

- A full untyped XQuery queries are bound to a unique form;
- Simplify treatments identification:
  - Operations orders;
  - Modeling XQuery (TGV [Travers et al. 2007]);
  - Distributing sub-queries (the XLive mediator);
- Validation with W3C use-cases [8/9] (except STRONG);
- Future works: typing;
Chen 2004 "From Tree Patterns to Generalized Tree Patterns : On Efficient Evaluation of XQuery”, University of British Columbia, MSc Thesis, 2004
Deustch et al. 2004 "The NEXT Framework for Logical XQuery Optimization”, VLDB, 2004
Fernández et al. 2003 "Implementing XQuery 1.0: The Galax Experience”, VLDB, 2003
Olteanu et al. 2002 "XPath : Looking Forward”, EDBT Workshop on XML Data Management (XMLDM), 2002


N.Travers,T.T.Dang-Ngoc,T.Liu  Full Untyped XQuery Canonization