

TDDD43

Theme 1.2: XML query languages

Fang Wei-Kleiner

<http://www.ida.liu.se/~TDDD43>



Query languages for XML

- Xpath
 - Path expressions with conditions
 - Building block of other standards (XQuery, XSLT, XLink, XPointer, etc.)
- Xquery
 - XPath + full-fledged SQL-like query language
- XSLT
 - XPath + transformation templates

```

- <minimodel name="sugartransport" xsi:noNamespaceSchemaLocation="minimodel.xsd">
  - <listOfCompartments>
    <compartment id="blood" name="inblood"/>
    <compartment id="cell" name="musclecell"/>
  </listOfCompartments>
  - <listOfSpecies>
    <species id="sug1" name="sugarinblood" compartment="blood"/>
    <species id="ins" name="insulin" compartment="blood"/>
    <species id="sug2" name="sugarincell" compartment="cell"/>
    <species id="en" name="energy" compartment="cell"/>
  </listOfSpecies>
  - <listOfReactions>
    - <reaction id="tocell" name="sugartocell">
      - <listOfReactants>
        <speciesReference species="sug1"/>
        <speciesReference species="ins"/>
      </listOfReactants>
      - <listOfProducts>
        <speciesReference species="sug2"/>
      </listOfProducts>
    </reaction>
    - <reaction id="move" name="makemovement">
      - <listOfReactants>
        <speciesReference species="sug2"/>
      </listOfReactants>
      - <listOfProducts>
        <speciesReference species="en"/>
      </listOfProducts>
    </reaction>
  </listOfReactions>
</minimodel>

```

XPath

- XPath specifies path expressions that match XML data by navigating down (and occasionally up and across) the tree
- Example
 - Query: `/minimodel/listOfReactions/reaction`
 - Like a UNIX path
 - Result: all author elements reachable from root via the path `/minimodel/listOfReactions/reaction`

Basic Xpath constructs

/	separator between steps in a path
<i>name</i>	matches any child element with this tag name
*	matches any child element
@ <i>name</i>	matches the attribute with this name
@*	matches any attribute
//	matches any descendent element or the current element itself
.	matches the current element
..	matches the parent element

Simple XPath examples

- All reactions
`/minimodel/listOfReactions/reaction`
- All reaction names
`/minimodel/listOfReactions/reaction/@name`
- All reaction elements, anywhere in the document
`//reaction`
- All species id, anywhere in the document
`//species/@id`
- Species of the minimodel
`/minimodel/*/species`

Predicates in path expressions

[*condition*] matches the “current” element if *condition* evaluates to true on the current element

- Species in blood compartment
`/minimodel//species[@compartment='blood']`
- Reactions with listOfProducts as child element
`//reaction[listOfProducts]`
- Name of species in blood compartment
`/minimodel//species[@compartment='blood']/@name`

Predicates

- Predicates can have and 's and or 's
- Species with id as 'en' and compartment as 'cell'
`//species[@id='en' and @compartment='cell']`
- Species with id as 'en' or compartment as 'cell'
`//species[@id='en' or @compartment='cell']`

Predicates involving nodesets

- `/minimodel/listOfSpecies/[species/@compartment='cell']`
 - There may be multiple species element, so `species/@compartment` in general returns a node-set (in XPath terminology)
 - The predicate evaluates to true as long as it evaluates true for at least one node in the node-set, i.e., at least one species has a compartment with value “cell”
- Tricky query
`/minimodel/listOfSpecies/[species/@compartment='cell'`
and `species/@compartment != 'cell']`

XPath functions

- `contains( $x$ ,  $y$ )` true if string x contains string y
- `count(node-set)` counts the number nodes in **node-set**
- `position()` returns the context position (roughly, the position of the current node in the node-set containing it)
- `last()` returns the “context size” (roughly, the size of the node-set containing the current node)
- `name()` returns the tag name of the current element

More examples

- All elements whose tag names contain “species”
`//*[contains(name(), 'species')]`
- Name of the first species
`/minimodel/listOfSpecies/species[position()=1]/@name`
A shorthand: `/minimodel/listOfSpecies/species[1]/@name`
- Name of the last species
`/minimodel/listOfSpecies/species[position()=last()]/@name`
- Lists with fewer than 10 species
`/minimodel/listOfSpecies [count(species)<10]`
- All elements whose parent's tag name is not “reaction”
`//*[name()!='reaction']/*`

General XPath location steps

- Technically, each XPath query consists of a series of location steps separated by /
- Each location step consists of
 - An axis: one of self, attribute, parent, child, ancestor, ancestor-or-self, descendant, descendant-or-self, following, following-sibling, preceding, preceding-sibling, and namespace
- A node-test: either a name test (e.g., reaction, *) or a type test (e.g., text(), node(), comment()), separated from the axis by ::
- Zero or more predicates (or conditions) enclosed in square brackets

Example

- Verbose (axis, node test, predicate):
`/child::minimodel/descendant-or-self::node()/child::species[attribute::id='sug1']`
- Abbreviated:
`/minimodel//species[@id='sug1']`
- `child` is the default axis
- `//` stands for `/descendant-or-self::node()/`

Example

- Which of the following queries correctly find the third speciesReferences in the entire input document?
- `//speciesReference [position()=3]`
 - Finds all third speciesReferences (for each of its parent element)
- `/descendant-or-self::node()[name()='speciesReference' and position()=3]`
 - Returns the third element in the document if it is a speciesReferences
- `/descendant-or-self::node()[name()='speciesReference'] [position()=3]`
 - After the first condition is passed, the evaluation context changes
 - Context size: # of nodes that passed the first condition
 - Context position: position of the context node within the list of nodes

Technical details of evaluation

Given a context node, evaluate a location path as follows:

1. Start with node-set $N = \{\text{context node}\}$
2. For each location step, from left to right:
 - $U \leftarrow \emptyset$
 - For each node n in N :
 - Using n as the context node, compute a node-set N' from the axis and the node-test
 - Each predicate in turn filters N'
 - For each node n' in N' , evaluate predicate with the following context:
 - Context node is n'
 - Context size is the number of nodes in N'
 - Context position is the position of n' within N'
 - $U = U \cup N'$
 - $N \leftarrow U$
3. Return N

XQuery

- XPath + full-fledged SQL-like query language
- XQuery expressions can be
 - XPath expressions
 - FLWOR expressions
 - Quantified expressions
 - Aggregation, sorting, and more...
- An XQuery expression in general can return a new result XML document
- Compare with an XPath expression, which always returns a sequence of nodes from the input document or atomic values (boolean, number, string, etc.)

FLWR expression

```
<result>{  
  for $s in doc("minimodel.xml")/minimodel//species  
  let $n := $s/@name  
  where $s/@compartment = 'cell'  
  return
```

```
    <cellspecies>  
      { $s/@id }  
      { $n }  
    </cellspecies>
```

```
}</result>
```

for: loop

let: assignment

where: filter condition

return: result construction

FLWR expression

```
for $x in doc('minimodel.xml')//species
  where data($x/@compartment)="blood"
  return $x
```

Join in XQuery

```
for $x in doc('minimodel.xml')//species,  
    $y in doc('minimodel.xml')//compartment  
where data($x/@compartment)=data($y/@id)  
return  
    <pair> {data($x/@name)} is inside {data($y/@name)} </pair>
```

```
for $x in doc('minimodel.xml')//species,  
    $y in doc('minimodel.xml')//compartment  
    [@id=data($x/@compartment)]  
return  
    <pair> {data($x/@name)} is inside {data($y/@name)} </pair>
```

XQuery functions

```
declare function local:fak($n)
  {if ($n=1)
   then 1
   else ($n * local:fak($n - 1))
  };
<res>{local:fak(3)}</res>
```

Note: recursion

```

declare function local:next($r)
{ <next>
{data($r/listOfProducts/speciesReference/@species)}
{for $x in $r/listOfProducts/speciesReference,
    $y in doc('minimodel.xml')
        //listOfReactants/speciesReference
        [@species=data($x/@species)]
return local:next($y/../../..)}
</next>};

for $x in doc('minimodel.xml')//species,
    $y in doc('minimodel.xml')
        //listOfReactants/speciesReference
        [@species=data($x/@id)]
return <answer> {data($x/@id)} leads to
                {local:next($y/../../..)} </answer>

```