Rules for Efficient XPath Evaluation

Dan Olteanu, Holger Meuss, Tim Furche, François Bry

olteanu@pms.informatik.uni-muenchen.de
The issue of concern

Our goal
An efficient XPath evaluation in a stream-based context.

The problem
- XPath allows back and forth navigation in the XML tree using location paths.
- Backward navigation is not appropriate in a stream-based context.
- Can backward navigation be avoided in XPath?

Our current contribution
Rules for rewriting arbitrary XPath 1.0 location paths into ones without backward navigation:
- two sets of equivalences between location paths with and without backward navigation.
- syntactical characterization of non-rewritable location path class.
- rewriting of the above mentioned class using XPath 2.0 / XQuery 1.0 variables.
Motivation

Stream-based query processing gained interest due to its application in

- handling large documents that can not be processed in memory, e.g. corpora, astronomical or biological data.
- data integration, e.g. Tukwila.
- publish-subscribe architectures, e.g. XTrie, XFilter.

The existing stream-based query systems consider only a subset of XQuery capabilities for forward navigation. We aim at enabling the entire set of XPath navigation capabilities to be considered for stream-based processing.
Tree Navigation using XPath 1.0

- ancestor
- preceding
- parent
- following
- sibling
- descendant
- child
- self
The xPath location path language

Grammar

\[
\text{path ::= path | path | / path | path / path | step}.
\]

\[
\text{step ::= axis :: nodetest | axis :: nodetest [ qualifier ] | ⊥}.
\]

\[
\text{qualifier ::= qualifier and qualifier | qualifier or qualifier | ( qualifier ) | path = path | path == path | path}.
\]

\[
\text{axis ::= reverse_axis | forward_axis}.
\]

\[
\text{reverse_axis ::= parent | ancestor | ancestor-or-self | preceding | preceding-sibling}.
\]

\[
\text{forward_axis ::= self | child | descendant | descendant-or-self | following | following-sibling}.
\]

\[
\text{nodetest ::= tagname | * | text() | node()}.\]
Differences between xPath and XPath

What we leave out

- references
- functions and most operators
- attributes
- schema types and namespaces
- comments and processing instructions
- abbreviated syntax

What we add

- equality based on node identity (==), cf. XQuery, XPath 2.0 Datamodel.
  \[ p == q \text{ computed as } \text{count}(p \cup q) < \text{count}(p) + \text{count}(q) \text{ in XPath 1.0.} \]
- unions at every location step, cf. XQuery 1.0.
Denotational Semantics for XPath

Q : Qualifier → Node → Boolean

Q[\text{and } q_1 \text{ and } q_2]x = Q[q_1]x \land Q[q_2]x

Q[\text{or } q_1 \text{ or } q_2]x = Q[q_1]x \lor Q[q_2]x

Q[(q)]x = Q[q]x

Q[p]x = S[p]x \neq \emptyset

Q[p_1 = p_2]x = \{ x_1 | x_1 \in S[p_1]x \land x_2 \in S[p_2]x \land value(x_1) = value(x_2) \} \neq \emptyset

Q[p_1 \Rightarrow p_2]x = \{ x_1 | x_1 \in S[p_1]x \land x_1 \in S[p_2]x \} \neq \emptyset
\( S \) : Pattern → Node → Set(Node)

\[
S[p_1 \mid p_2] x = S[p_1] x \cup S[p_2] x
\]

\[
S[/p] x = S[p](\text{root}(x))
\]

\[
S[p_1/p_2] x = \{ x_2 \mid x_1 \in S[p_1] x \land x_2 \in S[p_2] x_1 \}
\]

\[
S[p[q]] x = \{ x_1 \mid x_1 \in S[p] x \land Q[q] x_1 \}
\]

\[
S[\text{self::}n] x = \{ x_1 \mid x_1 = x \land \text{nodetest}(x_1, n) \}
\]

\[
S[\text{child::}n] x = \{ x_1 \mid x_1 \in \text{children}(x) \land \text{nodetest}(x_1, n) \}
\]

\[
S[\text{parent::}n] x = \{ x_1 \mid x_1 \in \text{parent}(x) \land \text{nodetest}(x_1, n) \}
\]

\[
S[\text{descendant::}n] x = \{ x_1 \mid x_1 \in \text{children}^+(x) \land \text{nodetest}(x_1, n) \}
\]

\[
S[\text{descendant-or-self::}n] x = \{ x_1 \mid x_1 \in \text{children}^*(x) \land \text{nodetest}(x_1, n) \}
\]

\[
S[\text{ancestor::}n] x = \{ x_1 \mid x_1 \in \text{parent}^+(x) \land \text{nodetest}(x_1, n) \}
\]

\[
S[\text{ancestor-or-self::}n] x = \{ x_1 \mid x_1 \in \text{parent}^*(x) \land \text{nodetest}(x_1, n) \}
\]

\[
S[\text{preceding::}n] x = \{ x_1 \mid x_1 \in \text{preceding}(x) \land \text{nodetest}(x_1, n) \}
\]

\[
S[\text{preceding-sibling::}n] x = \{ x_1 \mid x_1 \in \text{preceding-sibling}(x) \land \text{nodetest}(x_1, n) \}
\]

\[
S[\text{following::}n] x = \{ x_1 \mid x_1 \in \text{following}(x) \land \text{nodetest}(x_1, n) \}
\]

\[
S[\text{following-sibling::}n] x = \{ x_1 \mid x_1 \in \text{following-sibling} \land \text{nodetest}(x_1, n) \}
\]

\[
S[\bot] x = \emptyset
\]
Location Path Equivalences

Path equivalence
Location paths $p_1$ and $p_2$ are equivalent, noted $p_1 \equiv p_2$, when, for every document and context node within, they select the same set of nodes.

$p_1 \equiv p_2$ if $S[p_1] = S[p_2]$, i.e. if $S[p_1]x = S[p_2]x$ for all nodes $x$ (from any document).

1. Equivalences involving paths with reverse steps
   General and Specific Equivalences

2. Basic equivalences

   Adjunctions
   - If $p_1 \equiv p_2$ and $p$ relative, then $p_1/p \equiv p_2/p$.
   - If $p_1 \equiv p_2$ and $p_1, p_2$ relative, then $p/p_1 \equiv p/p_2$.
   - If $p_1 \equiv p_2$, then $p_1[q] \equiv p_2[q]$.

   Qualifier flattening If $p_2$ relative, then $p[p_1/p_2] \equiv p[p_1[p_2]]$.

   Qualifiers with joins If $p_2$ relative, then $p[p_1 \theta /p_2] \equiv p[p_1[\text{self::node()} \theta /p_2]]$. 
General Equivalences

- provide interaction between a location path $p$ or $p/a_n::n$ and a reverse step $a_m::m$, outside or inside qualifiers:
  
  \[ /p/a_n::n/a_m::m \equiv p' \text{ or } p[a_m::m] \equiv p' \]

- $p'$ does no longer contain the reverse axis $a_m$, but its symmetrical forward axis, i.e. if $a_m$ is an ancestor axis, then the symmetrical forward axis is descendant.

- there are only two general equivalences, i.e. for reverse steps outside and inside qualifiers.
General Equivalences for Absolute Paths with Reverse Steps

Let $p$ be a relative path, $m$ and $n$ node tests, $a_n$ a forward axis, $a_m$ a reverse axis and $b_m$ the symmetrical axis of $a_m$. 

\[ / p / a_n :: n / a_m :: m \equiv / \text{descendant} :: m [ b_m :: n == / p / a_n :: n ] \]
General Equivalences for Qualifiers with Reverse Steps

Let $p$ be a relative path, $m$ a node test, $a_m$ a reverse axis and $b_m$ the symmetrical axis of $a_m$. 

\[ p \left[ a_m :: m \right] \equiv p \left[ \text{/ descendant :: } m \text{ / } b_m :: \text{node()} == \text{self :: node()} \right] \]
Specific Equivalences

Provide interaction between every reverse step $L_r$ and every forward step $L_f$:

$$p/L_f/L_r \equiv p' \quad \text{or} \quad p/L_f [L_r] \equiv p'$$

The equivalent location path $p'$ can be of three kinds:

- without the reverse step $L_r$ or
- with the reverse step $L_r$, but pushed leftwise or
- with the reverse step $L_r$, at the same position from left to right in the location path, but without the leading $L_f$.

There are (at least) 10 specific equivalences for each reverse step.
Specific Equivalences for Parent steps

Consider the interaction between a descendant and a parent step outside qualifiers:

\[
\text{descendant} :: n / \text{parent} :: m \equiv \text{descendant-or-self} :: m [ \text{child} :: n ]
\]
Specific Equivalences for Ancestor steps

Consider the interaction between a descendant and an ancestor step outside qualifiers:

\[ p / \text{descendant} :: n / \text{ancestor} :: m \equiv p \ [ \text{descendant} :: n ] / \text{ancestor} :: m \]

| p / descendant-or-self :: m [ descendant :: n ] |
Specific Equivalences for Preceding steps

Consider the interaction between a descendant and a preceding step outside qualifiers:

\[
p \text{ / descendant :: n / preceding :: m } \equiv \ p \ [ \text{ descendant :: n } ] / \text{ preceding :: m }
\]

\[
p \text{ / child :: * [ following-sibling :: * / descendant-or-self :: n ] / descendant-or-self :: m}
\]
Specific Equivalences for Preceding-Sibling steps

Consider the interaction between a descendant and a preceding-sibling step outside qualifiers:

\[
\text{descendant} \;::\; n / \text{preceding-sibling} \;::\; m \equiv \text{descendant} \;::\; m [ \text{following-sibling} \;::\; n ]
\]
Equivalences using Variables

General and Specific Equivalences cover all absolute location paths involving reverse steps, except those having RR joins within qualifiers.

RR join

\[ p_1 \theta p_2 \] where \( \theta \in \{==,=\} \), and \( p_1 \) and \( p_2 \) are Relative paths such that at least one of them contains a Reverse step.

Example

/\text{descendant::n}[\text{self::n} = \text{preceding::n}]

Solution

variables from XPath 2.0 or host languages, e.g. XQuery, XSLT.

Example

for \( m \) in /\text{descendant::n}
    return \( m/\text{following::n}[m = \text{self::n}] \)
Location Path Rewriting

Each equivalence $p_l \equiv p_r$ gives rise to a rewriting rule $p_l \rightarrow p_r$.

- RuleSet$_1$ contains the general rules, RuleSet$_2$ the specific rules.

Consider $p$ an absolute location path without qualifiers containing RR joins, $p_l \rightarrow p_r$ a rule either from RuleSet$_1$ or RuleSet$_2$ and $q$ the result of the application of this rule to $p$.

Rule application

- If $p = p_l/p'$, then $q = p_r/p'$.
- If $p = p'/p_l/p''$, then $q = p'/p_l/p''$.

Properties of rule application

1. If $p$ contains a reverse step, then a rule from RuleSet$_1$ and a rule from RuleSet$_2$ is applicable to $p$.
2. The result of a rule application to the first reverse step in $p$ is an absolute location path without qualifiers containing RR joins.
3. If $q$ is the result of a rule application to $p$, then $p \equiv q$. 
Location Path Rewriting (II)

Consider an absolute location path $p$ without qualifiers in which RR joins occur. There exists an absolute location path $p'$ with no reverse steps such that $p \equiv p'$.

Using RuleSet$_1$

- $p'$ has a length and can be computed in a time linear in the length of $p$.
- $p'$ contains as many identity-based joins as reverse steps in $p$.

Using RuleSet$_2$

- $p'$ has a length and can be computed in a time exponential in the length of $p$ in the worst case.
- $p'$ might contain unions.
Rewriting Examples

Consider a query asking for all sections appearing immediately under titles, that appear before a name and are inside journals:

/descendant::name/preceding::title[ancestor::journal]/child::section

• Using RuleSet$_1$, the equivalent rewritten path is

/descendant::title[following::name == /descendant::name]
[/descendant::journal/descendant::node() == self::node()]/child::section.

• Using RuleSet$_2$, the equivalent rewritten path is

/descendant::journal/descendant::title[following::name]/child::section.
Future Work

• adding to XPath the **position** functions

• simplifying the XPath location paths obtained as result of rewriting

• further comparison of the two rule sets

• design and implementation of a stream-based XPath processor
  [http://www.pms.informatik.uni-muenchen.de/lehre/projekt-diplom-arbeit/streamedxpath.html](http://www.pms.informatik.uni-muenchen.de/lehre/projekt-diplom-arbeit/streamedxpath.html)

• find more in