A logic-based approach to cache answerability for XPath queries

M. Franceschet and E. Zimuel

Dipartimento di Scienze

Università “G. D’Annunzio” Chieti - Pescara (Italy)

Contributions

1. We present a new method for the evaluation of XPath queries with a logic-based approach with the aid of cache mechanisms.
2. We make a thorough experimental comparions of the following four evaluation techniques for XPath:
   - TopXPath
   - BottomXPath
   - CacheBottomXPath
   - Arb

TopXPath

- **TopXPath** (Gottlob et al., VLDB 2002) rewrites the original query into a Boolean combination of filter-free paths (sequences of steps without filters).
- Example:

$$\pi[\phi] = /child :: a[descendant :: b/following :: c]$$

- The query filter ϕ is rewritten by reading it from right to left and inverting each axis:

$$\rho = self :: c/preceding :: b/ancestor :: *$$

- Then the query $\pi[\phi]$ is evaluated as $\pi \cap \rho$.

BottomXPath

- **BottomXPath** (Franceschet et al., M4M 2005) rewrites the original query into a modal formula and then evaluates the formula *bottom-up*, that is, each formula is evaluated after the evaluation of its subformulas.
- Example: consider again the query

$$\pi[\phi] = /child :: a[descendant :: b/following :: c]$$

the corresponding modal formula is

$$a \wedge <parent> root \wedge <descendant> (b \wedge <following> c)$$

where tags are interpreted as atomic propositions ($root$ is a proposition that is true exactly at the tree root).

Arb

- **Arb** (Koch, VLDB 2003) is an automata-based method.
- The XML document is first converted into a binary tree representation, then two deterministic binary tree automata, one working *bottom-up* and the other one working *top-down*, are generated from the query.
- The evaluation is performed in two steps:
  1. the bottom-up query automaton runs on the XML binary tree;
  2. the top-down query automaton runs on the XML binary tree enriched with infos computed during the bottom-up run.

CacheBottomXPath

- **CacheBottomXPath** is a cache optimization of BottomXPath that we proposed in this work.
- The query is first converted into a modal formula and then chopped into a set of subformulas.
- Then, each subformulas, in bottom-up order, is searched in the cache. If the subformula is found, no evaluation is performed, since the result has been already computed.
- Otherwise, the subformula is evaluated and its result is possibly stored in the cache.
- The cache optimization is implemented using a hash table where the keys are the formula strings.

Computational complexity

- An analysis of the worst-case computational complexity of the above four methods does not help much to determine the most efficient evaluation strategy.
- Let k be the query complexity and n be the data complexity. On Core XPath, the worst-case complexity of TopXPath, BottomXPath, and CacheBottomXPath is $O(k \cdot n)$.
- Instead Arb terminates in $O(K + n)$, where K might be exponential in k.

Experimental evaluation

- To have a better understanding of the relative performance of the methods under testing, we conducted a probing experimental evaluation on synthetic and simulated real data.
- The main goals of these experiments are:
  1. to understand the effectiveness of the cache optimization in CacheBottomXPath;
  2. to compare the performance of the top-down and bottom-up approaches in TopXPath and BottomXPath;
  3. to test the scalability of the automata-based method encoded in Arb when the query length grows.

Experimental evaluation (2)

- We performed our experiments with XCheck, a benchmarking platform for XML query engines that we are demostrating in a demo session of VLDB.
- For *synthetic data* we used MemBeR (Afanasiev et al. XSym 2005) to generate XML documents and we implemented a random Core XPath query generator called XPathGen.
- For *simulated real data* we used the benchmarks XMark (Schmidt et al., VLDB 2002) and XPathMark (Franceschet, XSym 2005).

Conclusion

- The cache optimization is effective and should be integrated in an optimized full-fledged XPath/XQuery evaluator.
- The top-down approach of TopXPath is more efficient than the bottom-up approach of BottomXPath on queries with high selectivity, while the opposite is true on poorly selective queries.
- When the query is relative small, Arb is efficient and in fact the response times are independent on the query length. However, when the query size grows, Arb is no more scalable.

References

1. M.Franceschet, E.Zimuel *Modal logic and navigational XPath: an experimental comparison* M4M (2005) 156-172
2. C.Koch *Efficient processing of expressive node-selecting queries on XML data in secondary storage: A tree automata-based approach* VLDB (2003) 249-260
3. G.Gottlob, C.Koch, R.Pichler *Efficient algorithms for processing XPath queries* VLDB (2002) 95-106
4. L.Afanasiev, M.Franceschet, M.Marx, E.Zimuel *XCheck: A platform for benchmarking XQuery engines* VLDB (2006) `http://www.xcheck.org`
5. L.Afanasiev, I.Manolescu, P.Michiels *MemBeR: a micro-benchmark repository for XQuery* XSym (2005) 144-161
6. A.Schmidt et al. *XMark: A benchmark for XML data management* VLDB (2002) 974-985 `http://www.xml-benchmark.org`
7. M.Franceschet *XPathMark: an XPath benchmark for XMark generated data* XSym (2005) 129-143 `http://www.sci.unich.it/~francesc/xpathmark`