Evaluation of XPath Fragments Using Lambda Calculi

Pavel Loupal, Karel Richta
Dept. of Computer Science and Engineering
Faculty of Electrical Engineering, Czech Technical University
Karlovo nám. 13, 121 35 Praha 2
Czech Republic
loupalp@fel.cvut.cz, richta@fel.cvut.cz

Abstract. XML Path Language (XPath) is the most important standard for navigation within XML documents nowadays. In this paper we present the state of our current research that is focused on using a functional framework based on simply typed lambda calculi and a general type system - XML-λ - for description of semantics of a query language. With such formalism we are able to describe the semantics of all language constructs and evaluate XPath queries using the XML-λ virtual machine.

1 Motivation and Problem Statement

The World Wide Web Consortium designed the XML language [2] for an exchange of data on the Web and elsewhere. Then, there emerged many subsequent proposals of query languages. One of the most important of them is the XML Path Language – XPath [4]. It is a language for addressing parts of an XML document, designed to be used by both XSLT and XPointer. Later on, its extended version (denoted as 2.0) became a crucial part of the XQuery 1.0 standard [1].

As all experimenters who try to develop an efficient implementation of the standard we have faced this problem during our work on a prototype of a native XML database management system ExDB [10] within our research group. While the XPath engine is usually heavily used it is worth to put extra effort into its optimal design.

With respect to the fact that we deal for a long time with a functional data model for XML and its properties our current research is focused on using such model for evaluating XPath queries. Proposed paper suggest a way how to transform XPath queries into their functional version and then evaluate it using a functional virtual machine. Actually, it is only an outline of the research (ergo “Work in Progress”) but so far the result is promising.

Our long-term goal is to extend the scope of this work to XPath 2.0 and further to XQuery 1.0 – here we plan to formalize so called Core XQuery 1.0 [9] that is formed mainly by the FLWOR (For–Let–Where–OrderBy–Return) expression and XPath node selection. Results of this research are also expected to be published as a consistent work in the doctoral thesis of the first author.

2 Related Work

The essential document – XML Path Language Version 1.0 [4] – defines the syntax of the language and its informal semantics. The most important part is the description of the data model used (XML Infoset [5]) and denotation of all location paths (i.e. axis, node tests and predicates). The standard also defines few basic functions for manipulation with XML documents.

Successive works propose various extended data models and discuss the efficiency and complexity of XPath evaluation algorithms. From the point of XPath’s semantics, Wadler discussed the denotational semantics, for example here [13] or [14], seen from an XSLT point of view.

Gottlob et al. [7, 8] propose their own denotational semantics and discuss mostly the time and space complexity of XPath evaluation algorithms using their proposals.

3 Prerequisite Specifications

We expect that the reader is familiar with XML, XPath and other W3C’s specifications. XML-λ is not so known and therefore, for convenience, we repeat basic facts in following sections.

3.1 XML Path 1.0

XML Path 1.0 (XPath) [4] is a W3C standard for locating nodes in an XML tree. It is a variable-free language based on so called location steps. By successive evaluation of these steps we obtain a result. The result can be either a node-set, a boolean, a number or a string value.

For purpose of this paper we consider only part of this standard sometimes referred as ”Core XPath”. Such subset is defined in various ways, see for example [6], [7] or [13]. Here, we use the version shown in Figure 1 written using Extended Backus-Naur Form.

For example, for the ”XMP” Use Case published in the XML Query Use Cases [3] we may write a query that selects all books with given title and price as
The cornerstones of the framework are three components related to its type system: *element types*, *element objects* and *abstract elements*. Figure 3 shows the relationships between basic terms of W3C standards and the XML-λ Framework. We can imagine these components as the data dictionary in relational database systems.

**Fig. 3.** Relationship Between W3C and XML-λ Models

*Element types* are derived from a particular DTD, i.e. for each element defined in the DTD there exists exactly one element type in the set of all available element types (called $T_E$). Consequently, we denote $E$ as a set of abstract elements. Set members are of element types.

*Element objects* are basically functions of type either $E \rightarrow \text{String}$ or $E \rightarrow (E \times \ldots \times E)$. Application of these functions to an abstract element allows access to element’s content. *Elements* are, informally, values of element objects, i.e. of functions. For each $t \in T_E$ there exists a corresponding $t$-object.

For convenience, we add a "nullary function" (also known as 0-ary function) into our model. This function returns a set of all abstract elements of a given element type from an XML document.

Finally, we can say that in XML-λ the instance of an XML document is represented by a subset of $E$ and a set of respective $t$-objects.

As an example, we can rewrite the XPath query from Section 3.1 using XML-λ as

```
xmldata("bib.xml");
lambda b (/bib/book(b) and
    b/title = "TCP/IP Illustrated" and
    b/price > 399);
```

Such query is then evaluated as a $\lambda$-term (see [11, 12]).
4 Solution Proposal

We split the problem into two parts: (1) Transformation of XPath query into a λ-term and (2) evaluation of acquired term in the XML-λ virtual machine (VM). Formally, the first step can be defined as the mapping Comp from the input query into a λ-term, which serves as a code for the XML-λ VM. The translation is driven by required structure – let us suppose, that D is the required DTD, and we get the following signature:

\[ \text{Comp}_D : \text{CoreXPath} \rightarrow \text{λ-term} \]

In the second step we evaluate resulting λ-term by XML-λ VM. In general, the XML-λ VM can be formally described as a pair of mappings (the expression \( \text{SET}(X) \) denotes a set of all sets containing elements of the type \( X \)):

\[ \begin{align*}
\text{Eval}_M &: \lambda\text{-term} \times \text{XML} - \text{Doc} \rightarrow \\
\text{Next}_M &: \lambda\text{-term} \times \text{XML} - \text{Doc} \rightarrow \\
\end{align*} \]

\[ \text{SET}(\text{XML} - \text{Node}) \]

\[ \text{SET}(\text{XML} - \text{Doc}) \]

The expression \( \text{Next}_M[t] \) denotes the meaning of λ-term \( t \) as a mapping from the input state \( s \) of XML-λ VM into a new state \( \text{Next}_M[t](s) \). The computation can produce a set of resulting XML-nodes \( \text{Eval}_M[t](s) \). The whole picture is a combination of both steps (translation and interpretation), so resulting mapping \( \text{Sem}_D \) is:

\[ \text{Sem}_D : \text{CoreXPath} \times \text{XML} - \text{Doc} \rightarrow \\
\text{SET}(\text{XML} - \text{Node}) \]

\[ \text{Sem}_D[\text{expr}] = \text{Eval}_M[\text{Comp}_D[\text{expr}]] \]

where \( \text{expr} \) is an arbitrary CoreXPath expression.

The expression \( \text{Sem}_D[\text{expr}] \) denotes the meaning of CoreXPath query \( \text{expr} \) on the well-formed document (well-formed according to the XML-schema \( D \)) as a mapping from an XML-document into resulting set of XML-nodes. E.g.:

\[ \text{Sem}_D["/bib/book[1]/title"](\text{bib.xml}) = \\
= \{"TCP/IP Illustrated"\} \]

The transformation \( \text{Comp}_D \) can be defined by particular evaluation steps accordingly to the CoreXPath grammar (abbreviated):

\[ \begin{align*}
\text{Comp}_D[\text{expr}] &= \lambda x.(\text{Eval}_M[\text{Comp}_D[\text{expr}]]) \\
\text{Comp}_D[\text{Step}/\text{Path}] &= \\
&= \lambda x(\text{Comp}_D[\text{Path}](\text{Comp}_D[\text{Step}])) \\
\text{Comp}_D[\text{Path}] &= \\
&= \lambda x(\text{Comp}_D[\text{Path}](\text{Comp}_D[\text{Path}])) \\
\text{Comp}_D[\text{Step}/\text{Path}] &= \\
&= \lambda x(\text{Comp}_D[\text{Path}](\text{Comp}_D[\text{Path}])), \\
\text{Comp}_D[\text{Step}/\text{Path}] &= \\
&= \lambda x(\text{Comp}_D[\text{Path}](\text{Comp}_D[\text{Path}])), \\
\text{Comp}_D[\text{Step}/\text{Path}] &= \\
&= \lambda x(\text{Comp}_D[\text{Path}](\text{Comp}_D[\text{Path}])), \\
\text{Comp}_D[\text{Step}/\text{Path}] &= \\
&= \lambda x(\text{Comp}_D[\text{Path}](\text{Comp}_D[\text{Path}])). \\
\end{align*} \]

The query "/bib/book[1]/title" will be compiled into \( \text{Comp}_D["/bib/book[1]/title"] \), which is the following λ-term:

\[ \begin{align*}
\text{Comp}_D["/bib/book[1]/title"] &= \\
&= \lambda x(\text{Comp}_D["/book[1]/title"](\text{Comp}_D["/bib"])) \\
&= \lambda x(\text{Comp}_D["/title"](\text{Comp}_D["/book[1]/bib"]) \\
&= \lambda x((/title)(/book[1]))(/bib) \\
&= \lambda x(/bib/book[1]/title) \\
\end{align*} \]

Its meaning on the input document \( \text{bib.xml} \) will be:

\[ \begin{align*}
\text{Sem}_D["/bib/book[1]/title"](\text{bib.xml}) &= \\
&= \text{Eval}_M[\text{Comp}_D["/bib/book[1]/title"]](\text{bib.xml}) \\
&= \text{Eval}_M[\lambda x(/bib/book[1]/title)](\text{bib.xml}) \\
&= \lambda x(/bib/book[1]/title)(\text{bib.xml}) \\
&= \{"TCP/IP Illustrated"\} \\
\end{align*} \]

5 Conclusion and Future Work

We have presented the status of our current research focused on using a functional framework for description of XPath 1.0 semantics. We are now in an inceptive phase where we define basic formalisms and check the suitability of the method for the purpose requested.

As outlined above, we expect to extend the scope of this work to XPath 2.0 and later to XQuery 1.0. In general, it means redefinition of the syntax input and...
formalization of the XML-λ virtual machine. This research should unambiguously lead to a complete work that will be part of the doctoral thesis of the first author of this paper.

6 Acknowledgement

This work has been supported by the Ministry of Education, Youth and Sports under Research program no. MSM 6840770014 and also by the grant project of the Czech Grant Agency no. GA201/06/0756.

References