Parsing and Querying XML Documents in SML

Dissertation

Andreas Neumann
Fachbereich IV — Informatik
Universität Trier

Abstract

The Extensible Markup Language (XML) is a language for storing and exchanging structured data in a sequential format. Though originally designed for the use in document processing, it is by now employed for representation of data in virtually all areas of computing, especially on the Internet.

The basis of all XML processing software is an XML parser, which parses a document in XML syntax and exposes it as a document tree to the application. Document processing then basically reduces to tree manipulation. Modern functional programming languages like SML and HASKELL with their tree-structured data-types are therefore well-suited for implementing such applications. Nonetheless, the area of XML processing software is dominated by JAVA. Functional programming languages play a minor role, partly due to the lack of a complete implementation of an XML parser.

One of the most important tasks in document processing is querying, that is the extraction of parts of a document that match a structural condition and are in a specific context. Due to the tree-like view of documents, querying XML can be implemented with techniques from tree language and automata theory. These techniques must, however, be adapted to the needs of XML. One specific requirement is that even extremely large documents must be processed. It must therefore be possible to perform the querying algorithm in a single pass through the document, without the need of constructing a copy of the document tree in memory.

This work is divided into two parts: The first part presents *fxp*, an XML parser written completely in SML. We describe the implementation of *fxp* and discuss our experiences with SML. We analyze the run-time behavior of *fxp* and compare it to XML parsers written in imperative and object-oriented programming languages.

The second part presents an XML querying algorithm based on forest automata theory. The algorithm locates all matches of a query in at most two passes through the document. For an important subclass of queries even a single run suffices. Moreover, we discuss the implementation of the algorithm based on *fxp*. For each of the two parts a separate, more detailed introduction is given.