Available online at www.sciencedirect.com

ScienceDirect

Data & Knowledge Engineering 59 (2006) 576–602

DATA & KNOWLEDGE ENGINEERING

www.elsevier.com/locate/datak

Automaton meets algebra: A hybrid paradigm for XML stream processing ☆

Hong Su *, Elke A. Rundensteiner, Murali Mani

CS Department, Worcester Polytechnic Institute, Worcester, MA 01609-2280, USA

Received 4 October 2005; accepted 14 October 2005
Available online 28 November 2005

Abstract

XML stream applications bring the challenge of efficiently processing queries on sequentially accessible token-based data streams. The automata paradigm is naturally suited for pattern recognition on tokenized XML streams, but requires patches for fulfilling the filtering or restructuring functionalities in the XML query language. In contrast, the algebraic paradigm is a well-established technique for processing self-contained tuples. It however does not traditionally support token inputs. The *Raindrop* framework is the first to accommodate these two paradigms within one algebraic framework, taking advantage of both. This paper describes the overall framework, highlighting in particular three aspects. First, we describe how the tokens and tuples are modeled in one uniform query processing model. Second, we present the query rewriting that switches computations between these two data models. Third, we discuss strategies for the implementation and synchronization of the operators within the framework. We report experimental results that illustrate the unique optimization opportunities offered by this novel framework.
© 2005 Elsevier B.V. All rights reserved.

Keywords: XML stream; XQuery processing; Automata; Algebra

1. Motivation of XML stream processing

There is a growing interest in data stream applications such as monitoring systems for stock, traffic and network activities [6]. Various research projects have recently targeted stream applications, such as Aurora [3], Borealis [11], STREAM [7], Niagara [10], TelegraphCQ [9], Cougar [14] and CAPE [31]. Many of them focus on relational or object applications which assume a tuple-like data model (a tuple can contain flat values and objects as in a relational or object database).

Due to the proliferation of XML data in web services [27], there is also a surge in XML stream applications [8,13,17,18,20,26,30,37] such as XML packet routing [2], selective dissemination of information [4], and notification systems [28]. A challenge that XML stream applications pose is that the "tuple" processing unit no

☆ This research is partly supported by NSF under Grant No. 0414567. Hong Su would like to thank IBM for the IBM Cooperative Fellowship from 2001 to 2004.
* Corresponding author.
E-mail addresses: suhong@cs.wpi.edu (H. Su), rundenst@cs.wpi.edu (E.A. Rundensteiner), mmani@cs.wpi.edu (M. Mani).

0169-023X/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.datak.2005.10.008

longer completely fits. In the XML context, we specifically refer to a "tuple" as a set of cells, each cell containing a set of XML nodes (i.e. XML trees). This is because the XML query semantics [34] is defined as XML node outputs computed on the given XML node inputs.

However, XML streams are usually modeled as a sequence of primitive tokens, such as a start tag, an end tag or a PCDATA item. That is to say, a processing unit of XML streams has to be a token, which is at a lower granularity than an XML node. Such a processing style, i.e., a processing unit being at a lower granularity than the data model, has been little studied in the database community before. This granularity difference is a XML stream specific challenge that has to be addressed.

2. State-of-the-art

In this section, we describe two camps of solutions that have been proposed for XML stream processing. The first camp uses tokens as the processing unit throughout the whole evaluation process. In contrast, the second camp uses different processing units in different stages of evaluation. In the first stage, it consumes token inputs but generates tuple outputs. Tuple processing units are then used throughout the second stage.

2.1. Pure automata paradigm

Automata were originally designed to match patterns over strings, which is very similar to matching path expressions over tokens in XML. Several projects [20,26,30,37] are inspired to exclusively use automata for the complete query processing. Such a *pure automata paradigm* has to strike a balance between the expressive power of the query it can handle and the manageability of its constructs.

For example, XPush [20], using a push-down automaton, supports rather limited query capabilities. Since the push-down automaton has no output buffers, it cannot return the destination elements reachable via an XPath, not to mention restructure the destination elements. It can only give a boolean result indicating whether or not an XPath is contained in the input stream.

Some projects adopt more powerful automata in order to provide more query capabilities. Typical examples are XSM [26] and XSQ [30] supporting the XQuery and XPath languages respectively. However the increased expressive power of the queries is not gained without sacrifice. The Turing-machine-like model they adopt describes the computations at a rather low level. Such a query model is somewhat similar to a procedural language that presents all internal details of the computations. Fig. 1 gives an example of how a path expression "/a" is modeled in XSM. The automaton reads a token from the input buffer each time. The state transition indicates if a certain token has been read (expressed as the part before "|"), then the corresponding actions (expressed as the part after "|") will be taken. For instance, the transition from state 1 to state 2 indicates that if a token ⟨a⟩ is read, it should be copied to an output buffer.

Such a pure automata paradigm has not been thoroughly studied as a query processing paradigm before. Many problems that have been well studied in tuple-based algebraic frameworks remain unexplored in this new paradigm, such as how to optimize the queries in a modular fashion, how to rewrite the queries, how to cost alternative processing plans, etc.

2.2. Loosely coupled automata and algebra paradigm

The tuple-based algebraic paradigm has been widely adopted by the database community for query processing. It is thus not surprising that numerous tuple-based algebras for processing static XML have been

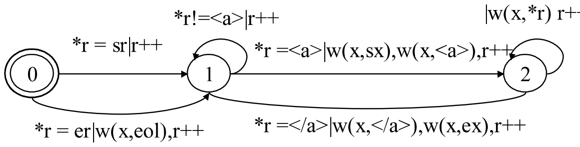

Fig. 1. Automata in XSM encoding an XPath expression "/a".