Mathematical Over XML Family of Technologies

Adriana Georgieva,
Technical University of Sofia, Faculty of Applied Mathematics and Informatics

Abstract: **This paper presents one theoretical ground and some simple applications of the mathematical tools applied to XML-based technologies, analyzed in the paper. The paper studies XML hierarchical constructions as structures of unranked trees, which lead to modern logical formalisms for querying and navigation over XML documents. By using some advanced algebraic tools, it is possible to work with ordinary linear operators for further data representation on the physical level instead of performing more difficult operations, which are specific for the widely used models. The realizations of the presented algebraic method implemented to real XML hierarchical data structures are connected with possibilities for high efficiency with respect to optimization of the time access to XML information. Some problems in the area of this specific XML terminology are discussed as well.**

Keywords: **XML tree structure, XML conceptual model, First and Monadic second order logic, conceptual level.**

I. Introduction

The XML standard, defined by [1], is widely used for internet content. XML is a text format, not a binary format. The advantage is that the data can be inspected with a text editor and that it is easy for scripts to process the data. XML is a modular standard. Similarly to LaTeX, which can use different style files in parallel, XML can use different document structures simultaneously. XML provides a syntax for the resource description framework (RDF), a language to express data in a relational way aiming for setting a "semantic web". Unlike the limited tag-set offered by HTML, XML allows users to define tags based upon the logical structure of their documents. With the purpose to provide a convenient access to the document, in the WEB-programming practice a programming tool, named XML processor, is created. An XML processor is a programming mechanism that parses XML source to build an internal representation of the document. This is a natural interface for the application, because XML processor incorporates lexical and syntax analyzers together and has become conceptually popular under the common name parser. A parser is a software component that sits between the application and the XML files. When Microsoft supplied Internet Explorer 4.0 as the first browser with the XML support, they bundled two XML parsers with it. According to the literature, the word parser comes from the fundamental theory of Parsing, Translation and Compiling [2] and covers only the well-known imagination of syntax analyzer. The scanner reads the text and classifies it as "tokens". A token is a category that is recognized by the parser. For example, a scanner for the Java programming language might return the tokens that include: identifier, integer, reserved words, symbols for arithmetic operation (+, –, *) and so on. In terms of the classical theory of Parsing, Translation and Compiling, the parser (or syntax analyzer) creates a parse tree, which is an in-memory representation of the source code. Another part of the compiler [2], known as the backend processor, uses parse tree to generate object files and to perform the necessary optimization of the object file. Nowadays, the parsing process (in WEB-terms) is usually done by two logical components: a parser and a scanner. As it was mentioned above, the specification of W3C about XML definitions presents the parser as an essential idea and combines lexical and syntax analyzers, interpretation of DTD-parameters, the links to WEB, etc. The parser has to convert internal encoding (ASCII or UTF) to Unicode, verify well-formed XML document and optionally validate against an XML Schema or DTD. Finally, XML parser is the programming tool that could construct a DOM tree, perform transformations, bind to Java or C# objects and launch the application program.

In Section 2 the existing two approaches to XML API-object-based interface (DOM) and event-based interface (SAX) are described. In Section 3 the main characteristics of several widely used XML tree structures are shown. The access to the information stored in XML hierarchy is presented in Section 4. Finally, in Section 5, general issues, conclusions, lines of further research and some open problems are discussed.

II. Object Based Interface.VS. Event-Based Interface

There exist two basic ways to interface a parser with an application: using object-based interfaces or using event-based interfaces. By means of object-based interface, the parser explicitly builds a tree of objects that contains all the elements in the XML document. This is probably the most natural interface for the application because it is handled as a tree in the memory that exactly matches with the hierarchical file definition. Furthermore, if using a validating parser, the tree may have been validated against the DTD or XML Schema. The standard for object-based interfaces is DOM, Document Object Model, published by W3C.

The second approach to interfacing the parser and the application is through events. With an event-based interface the parser does not explicitly build a tree of objects. Instead, it reads the file and generates events as it finds elements, attributes or text in the file. There are events for element starts, element ends, attributes, text content, entities, and so on. At first sight, an event-based API is less natural for the application, because here an explicit tree that matches the input XML file is not given. In this case the parsing process has to mark out the events and determinewhich tree is being described implicitly. Object-based interfaces are ideal for applications that manipulate XML documents such as browsers, editors, XSL processors, etc. Event-based interfaces are geared toward applications that maintain their

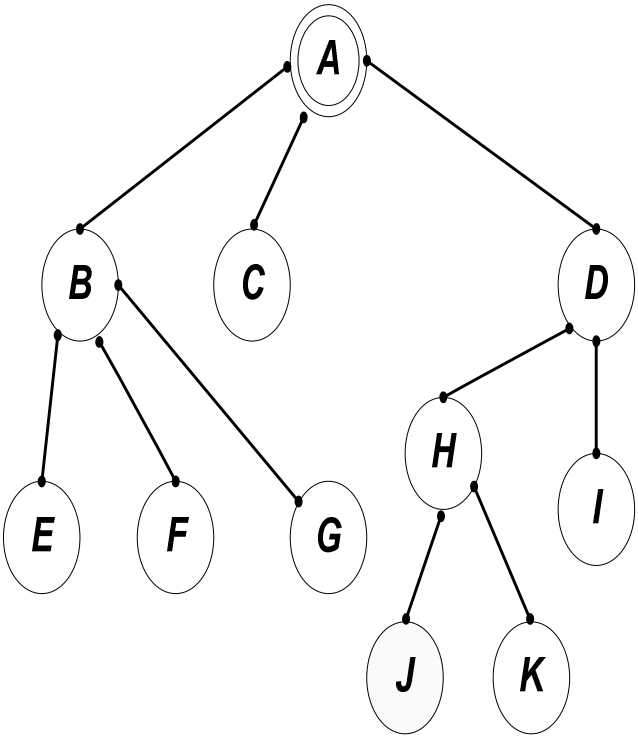
own data structure. If the format of corresponding application is database schema (not XML schema), these applications have their own data structure and they map from an XML structure to their internal structure. An event-based interface requires fewer objects and uses less memory. The standard for event-based interface is SAX (SIMPLE API for XML).

III. XML Document as a Tree Structure

Actually, every XML document can be represented by a tree. Here a XML fragment built from free elements is presented:

```
<a>
    <b>
        <e> </e>        or      <e/>
        <f> </f>        or      <f/>
        <g> </g>        or      <g/>
    </b>
        <c></c>         or      <c/>
    <d>
          <h>
        <j> </j>        or      <j/>
        <k> </k>        or      <k/>
          </h>
        <i> </i>        or      <i/>
    </d>
</a>
```

The hierarchical tree, corresponding to this XML code, is shown below in Fig.1.



(A (B (E) (F) (G)) (D (H (J) (K)) (I)))

Fig.1. Hierarchical tree presentation (unranked tree)

When the tree is not balanced, each random division into two sub-trees leads to unspecified number of nodes in sub-trees. The nodes in this kind of XML tree may have an arbitrary large number of children. In particular, labeled unranked trees are used as a model of XML document. When dealing with labeled unranked trees, we assume that each node has one label. The modern theory of trees proves the following assertion: each unranked tree can be presented by reciprocal and simple way through balanced binary trees. In other words, regularity preserving translations from unranked trees to binary trees exists [7]. There exist also well-known regularity-preserving translations between unranked and ranked (binary) trees [7] which are applied to a finite tree domain. Combined with regularity-preserving translations from unranked trees to binary trees these researches play a prominent role in the presented approach, especially in obtaining practical results. If we divide unbalanced tree into two sub-trees and try to navigate with two parsing procedures, both parsers handle the nodes (root, children and leaves) with different running times. In the parsing process both parsers execute handling of appropriate XML documents with time difference ΔT . We call this difference in times ΔT using the term “dead hug”. The so-called dead hug as time period depends on the number of nodes in both trees, the lengths of names of elements, etc.

Finally, we can apply these criteria for more detailed description of two parser parallel mechanisms by means of a concrete example. The presented below XML document with root element “branch “ is explicitly shown along with its unranked tree (Fig.2.):

<branch> <product> <vendor> <name> </name>

<price/> </vendor> </product> <product> <vendor/>

<name/></product></branch>

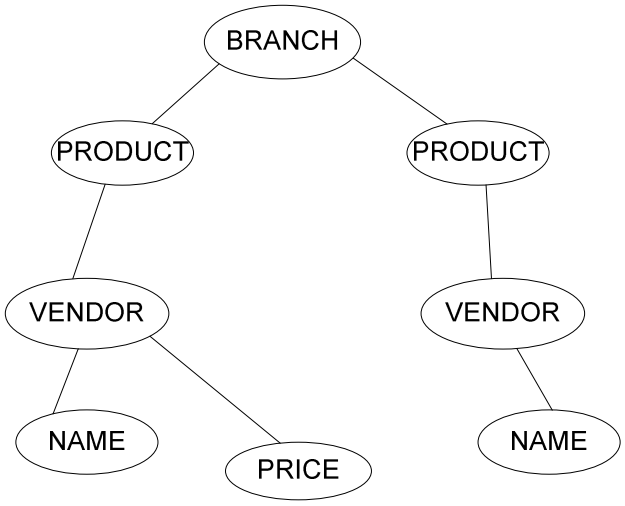


Fig.2. XML document and unranked tree equivalent

Obviously, here it is possible to build a two-parser architecture, which would work in parallel. It can be done using algebraic search and access to information, stored in XML document, explored below.

IV. Algebraic Search and Access to the Information Stored in XML Hierarchy

In order to present the data algebraically at the conceptual level, it is important to mention the following information:

- Any object with its attributes is considered as *n*-sequence of the numbers, which represent the code values of these attributes.
- This XML conceptual model permits to work with the natural numbers only and keeps the natural relationships in XML hierarchy.

This analysis is concentrated particularly on module-finite algebra over commutative rings. E. Noether [6] first demonstrated the importance and usefulness of the following conditions: collection A of subsets of a set A satisfies the ascending infinite chain (or ACC) if here doesn't exist a properly ascending infinite chain:
$A_1 \subset A_2 \subset \ldots \subset A_i \subset \ldots \subset A_n \ldots$ of subsets from A.

In particular, if A is a module, the notation $B < A$ means that B is a proper sub-module of A. In the theory of module-finite algebra [6] the following proposition has been proven: for a module A, the following conditions are equivalent:

- A has the ACC on sub-modules
- Every nonempty family of sub-modules of A has a maximal element
- Every sub-module of A is finitely generated.

For the purpose of this research on the conceptual model for data representation as an algebraic structure, the following propositions are proved [4]:

Proposition 3.1. There exists a bijective mapping f_1 : *M* $\rightarrow$ *K*, which correlates to every really existent relationship between the real data set (with appropriated structural schema) and XML data base conceptual model. This means that if *M* represents the data (objects) from the real world, this bijective mapping correlates with every element, which belongs to *M* in one-to-one manner as a unique element, described with the finite sequence (*n*-sequence). The elements of this finite sequence are the corresponding XML components – attributes ($a_{r1}, a_{r2}, a_{r3}, \ldots, a_{ri}, \ldots, a_{rn}$) from *K*, which define the object O_r from *M*.

Proposition 3.2. There exists a bijective mapping f_2 : *KR* $\rightarrow$ *KA*, which correlates to every existing relationship between *n* different XML components from *KR* in one-to-one manner as a unique finite sequence (of length *n*) from *KA*. At that *KR* is the set of relationships $R^{(n)}$ = (a_1, $a_2, \ldots, a_n$) and *KA* is the corresponding set of *n*- sequences ($a_{r1}, a_{r2}, a_{r3}, \ldots, a_{rn}$) at the conceptual level. To every relationship between XML components from object O_r (in *KR*) a unique *n*- sequence that includes numbers a_{ri} from *KA* is defined in bijective way. At that $a_{ri} \in D_i \subset Z_i$ (here Z is a set of integers), where $r = 1,\ldots,m;\ i = 1,..,n$.

In this way the set *KA* can be defined - the conceptual schema of really existing hierarchical XML-structure *M* as an algebraic structure denoted by $A = \{A_1, A_2, \ldots, A_n\}$ – a family of $Z_{\alpha i}$- modules A_i .Therefore, the formal description used for the interpretation of XML data hierarchy establishes a bijective map f_1 between object O_r and *n*- sequence ($a_{r1}, a_{r2}, a_{r3}, \ldots, a_{rn}$):

$$O_r \overset{f_1}{\rightleftarrows} (a_{r1}, a_{r2}, a_{r3}, \ldots, a_{rn})$$

By analogy, the one-to-one manner mappings can be determined between the object O_r and the relationships *KR*: $R^{(n)}$ = (a_1, a_2,.., a_n). Similar to the previous procedure, mappings are established between *n*-sequence (a_{r1} , a_{r2} , a_{r3} ,..., a_{rn}) and the relations $R^{(n)}$ = (a_1, a_2,.., a_n).

The XML hierarchical structure presented in this paper has the following main values:

- *n* - are the number of hierarchical levels.
- α_1, α_2, …, α_n - the size of every hierarchical level; informally, the corresponding number of the elements for each XML hierarchical level; $\alpha_i \in Z$; moreover $\alpha_1 < \alpha_2 < \ldots < \alpha_i < \ldots < \alpha_n$.
- c_0, c_1, c_2, ..., c_{n-1} - a number of children (subordinated elements) of any element from level *i* to level *i+1*; informally, the children of every element from one level into the next; $c_i \in Z$; ordinary $c_0 = 1$.

Every object from level *n* is presented by the finite sequence $(a_1, a_2, \ldots, a_n)$ belonging to A and can be transformed into the linear combination:

$$(a_1, a_2, \ldots, a_n) = \sum_{i=1}^{n} a_i \cdot e_i = a_1 \cdot e_1 + a_2 \cdot e_2 + \ldots + a_n \cdot e_n \qquad (4.1.)$$

where:

e_1 = (1,0, … ,0) ; e_2 = (0,1, … ,0) ; …; e_n = (0,0, … ,1)

are the generators (base elements) for the structure A. Obviously, here each e_i expresses (points at) the presence (or the absence) of an element from the appropriated level *i* in XML hierarchy. For the purpose of this research we assume that h_i = *(0, 0, ...,* $c_1 \cdot c_2 \cdot \ldots \cdot c_{i-1}$*, ..., 0),* and $h_1 = e_1$ are sequences of length *n,* which we call characteristic elements, because they depict the corresponding number of all subordinate elements for each hierarchical level. Actually, they show the existing children nodes connected with every corresponding XML node in the hierarchy. So in this algebraic modelling the transition "XML tree structure – physical presentation" is defined as the mapping

$\Phi : A \to P$, which correlates to every finite *n*-sequence $(a_1, a_2, \ldots, a_n)^r \in A$ in one-to-one manner as a fixed integer $p_n^r \in P$. In his way the integer p_n^r uniquely defines the place (address) of the object O_n^r from level *n* in the real XML hierarchical structure , i.e. its address in the physical database design. For the physical data representation it has been proved that this mapping $\Phi : A \to P$ is bijective and presents linear mapping (linear function) [5]. As the result of this theoretical model the unique determination of the physical address of the object O_k^r (XML node r from level *k)* in common structure, i.e. the number p_k^r by the following way, is given here:

$$p_k^r = \sum_{i=1}^{k-1} \alpha_i + a_k^I = \alpha_1 + \alpha_2 + \ldots + \alpha_{k-1} + a_k^I = \\ = \alpha_1 . \sum_{i=1}^{k-1} \Phi(h_i) + a_k^I \qquad (4.2.)$$

where:

$$\Phi(h_1) = c_0 = 1;\ \Phi(h_2) = c_0 . c_1 = c_1;\ \ldots;$$
$$\Phi(h_k) = c_0 . c_1 . c_2 . \ldots . c_{k-1} = c_1 . c_2 . \ldots . c_{k-1}$$

are the transforming characteristic elements from the XML tree hierarchy [5]. Definitely, according to the suggested formal algebraic description each of the physical addresses p_k^r of the objects in a real XML hierarchical data structure [6] can be simply calculated by means of formula 3.1. In the formulae 3.1 and 3.2 a_k^I is the code value a_k in the hierarchical level *k*:

$$a_k^I = \{\ldots\{(a_1 - 1) . c_1 + a_2 - 1)\} . c_2 + \ldots + (a_{k-1} - 1)\} c_{k-1} + a_k - 1 \qquad (4.3.)$$

Here a_k^I is the code value a_k in the hierarchical level *k.* The calculations in this formula are based on the formal description of the sets of code values of XML nodes components [5], [6].

By analogy (from formula 3.1.) there exists the opposite mapping $\Phi^{-1} : P \to A$, which presents the transition "physical presentation – XML tree structure". It permits to calculate every component of integers, which describe it in XML tree, from the address of the element at the physical level:

$$a_k^I = p_k^r - \sum_{i=1}^{k-1} \alpha_i$$

and $a_{k-1} = [p_k^r / c_k] + 1$ ------->

$$p_{k-1}^r = \alpha_1 . \sum_{i=1}^{k-2} \Phi(h_{i-1}) + a_{k-1}$$

……

$$a_2 = [p_3^r / c_3] + 1 \qquad \text{------->}$$

$$p_2^r = \alpha_1 . \sum_{i=1}^{1} \Phi(h_2) + a_2$$

$$a_1 = [p_2^r / c_2] + 1 \qquad (4.4.)$$

Here the notation [ ] is used for receiving the integer by division, that is necessary to get components $a_i \in Z$.

Finally, the proposed approach is different in comparison with many other well-known query and transformational languages (as XSL, XSLT, XPath) with respect to their definition, expressiveness, and search techniques.

The described formal presentation of the XML data structures as algebraic structure of modules over the rings of integers allows this model to be used as a common algebraic base in hierarchical data modelling [5]. The algebraic presentation of the binary relations in XML hierarchical structures permits to combine the formal description with the rules of natural XML hierarchy.

V. Conclusion

On the basis of the described XML tree structures, an approach was proposed in this article, which was inspired by the presented theoretical formalisms.The approach directly addresses XML hierarchical components. Furthermore, it will accelerate search paradigms in real environment. This approach will raise the rate of parsing and will accelerate the dialog between the browser and the server. It is especially useful in the parsing procedures for huge input XML-documents. Further development of the XML parallel parser representation foresees to extend these mathematical models with practical examples as soon as it is possible. Meanwhile, some practical works dedicated to two-parser architecture using algebraic methods are underway. The initial results give hopeful prospects about these new parsing technologies. That is why these new propositions in the traditional theory of parsing, translation and compiling will take effect on

WEB practice. This conclusion concerns the declarative languages, as, for example, the transformation) $(XML \Leftrightarrow HTML)$ and opposite transformation $(HTML \Leftrightarrow HML)$ [3]. The algebraic approach reveals the possibilities for reaching linear time functions in HML tree hierarchy handling, which could accelerate this critical process. n

REFERENCES

[1] World Wide Web Consortium, http://www.w3.org/TR/2004/. Extensible Markup Language (XML) 1.0.W3C Recommendation, third edition, February 2004.

[2] Alfred V. Aho, Jeffrey D. Ullman (1972): The Theory of Parsing, Translation and Compiling, Prentice - Hall, Inc. Englewood Cliffs, N.J.

[3] Jim Keogh, Ken Davidson (2005): XML DeMYSTiFied, McGraw-Hill, Emeryville, California, USA.

[4] A.Georgieva, B.Georgiev. Conceptual Method for Extension of Data Processing Possibilities in XML Hierarchy, Fourth International Bulgarian Greek Conference Computer Science. 2008, Kavala, Greece, 2008.

[5] A.Georgieva. One Algebraic Method of Database Design. Proceedings of the 1-st International Conference on Mathematics and Informatics for Industry (MII - 2003), Greece, April 2003. pp. 235-243.

[6] Garding L. and Tambour T. Algebra for Computer Science, Spring-Verlag, N.Y., 1988.

[7] J. Carme, J. Niehren, M. Tommasi, Querying unranked trees with stepwise tree automata. In Rewriting techniques and Applications, 2004, pp. 105-118.

Adriana Georgieva. In 1990, she received the Doctoral Degree in Informatics. The Doctoral Thesis was defended at the Technical University of Sofia (TU-Sofia), 8 Kliment Ohridski, 1000 Sofia, Bulgaria. From 1997, she is an Associate Professor of Mathematical Modeling and Applied Mathematics in Faculty of Applied Mathematics and Informatics of TU-Sofia.

Current Position: Vice Dean of Faculty of Applied Mathematics and Informatics of TU-Sofia and Associate Professor at the same faculty. Major areas of scientific activity are database and data structures, data models, mathematical modeling and applications in database design and database management systems, information systems, web technologies, algebra and algebraic structures, programming, software in education, correlation analysis.

She is a Member of the Union of Bulgarian Scientifics, the Union of Bulgarian Mathematicians, AMS, SIAM and fellow of IEEE, ICIW and others. She received grants for Specialization “Computer Science”-Helsinki, Finland (1989) and took part in numerous international scientific conferences and workshops (2008 – 2012). She is the author of over 50 scientific papers in the area of applied mathematics and informatics.

Address: 8 Kliment Ohridski, 1000 Sofia, Bulgaria

E-mail: adig@tu-sofia.bg