When Conceptual Model Meets Grammar: A Formal Approach to Semi-structured Data Modeling*

Martin Nečaský and Irena Mlýnková
Department of Software Engineering, Charles University in Prague, Czech Republic
{necasky,mlynkova}@ksi.mff.cuni.cz

Abstract. Currently, XML is a standard for information exchange. An important task in XML management is designing particular XML formats suitable for particular kinds of information exchange. There exist two kinds of approaches to this problem. Firstly, there exist XML schema languages and their formalization – regular tree grammars. Secondly, there are approaches based on conceptual modeling and automatic derivation of an XML schema from a conceptual schema.

In this paper, we provide a unified formalism for both kind of approaches. It is based on formal specification of XML schemas, conceptual schemas, and mappings between both kinds of schemas. The formalism gives necessary conditions on the mappings. The mapping may then be applied in practice not only for unified process of designing XML schemas on both levels, i.e. conceptual and grammatical, but also for integration and evolution of XML schemas.

1 Introduction

Currently, XML is a de-facto standard meta-format for information exchange on the Web. An important task in XML management is designing particular XML formats suitable for particular kinds of information exchange. This means creating of XML schemas describing the required structure of XML documents. For this reason, there have recently appeared various works on this topic.

Related work. In one direction, there exist various languages for expressing XML schemas (XML schema languages), e.g. XML Schema [14] or Relax NG [2]. These languages are practical but lack a formal background. This background was recently introduced by Murata et al. in [9] and is called regular tree grammars. The introduced formalism unifies a family of XML schema languages based on specification of grammar rules. It allows for a formal comparison of the languages and also enables one to formally describe algorithms for validation of XML documents against XML schemas.

In the other direction, there is a bunch of works which look at the problem of designing XML schemas from the conceptual point of view. These approaches usually apply the ER model (such as [3,8,7,10]) or UML class model (such as [4,13,14]). They suppose designing a conceptual diagram of the problem domain first. After that, a representation in an XML schema language is derived automatically from the conceptual diagram. We provide an extensive survey of these approaches in [10].

* Supported the Czech Science Foundation (GAČR), grants number P202/10/0573 and 201/09/P364.

© Springer-Verlag Berlin Heidelberg 2010
Contributions. The contribution of this paper is a formalism which unifies both directions. This is a first attempt in this area to our best knowledge. The introduced formalism is based on mappings between a regular tree grammar and conceptual schema. These mappings bring a lot of advantages when integrating XML schemas and managing their evolution as we described in [5][12][6]. We have implemented the results presented in this work in a case tool called XCase [11].

Outline. The outline of the paper is as follows. In Section 2 we introduce an extension to regular tree grammars as proposed in [9]. In Section 3 we introduce a formal conceptual model for XML data based on UML class diagrams. In Section 4 we introduce a mapping between both levels, i.e. a regular tree grammar and conceptual schema. We conclude in Section 5.

2 Describing Structure with Regular Tree Grammar

In this section, we formally introduce the model of XML documents and schemas. As it is usual in the recent literature, we will represent an XML document as an XML tree. An XML schema will be expressed in a form of a regular tree grammar.

For the purposes of the following text, we introduce some notation. Let \( C = \{ m..n \mapsto m \in \mathbb{N}_0 \land n \in (\mathbb{N}_0 \cup \ast) \land (m \leq n \lor n = \ast) \} \) denote an infinite set of cardinality constraints where \( \mathbb{N}_0 \) denotes the set of natural numbers including 0. Further, let \( L \) denote an infinite set of labels (words over a finite alphabet \( \Sigma \), i.e. \( L \subseteq \Sigma^* \)). Let \( D \) denote a finite set of supported data types. A data type \( d \in D \) is a possibly infinite set of data values. For example, a data type \( \text{integer} \) is an infinite set of all integers. Moreover, let \( D = \bigcup_{d \in D} d \), i.e. \( D \) denotes the set of all values of all considered data types. Finally, let \( \mathcal{X} \) be a set. We will use \( \text{SUBSETS}(\mathcal{X}) \) and \( \text{SUBLISTS}(\mathcal{X}) \) to denote the set of all subsets and ordered subsequences of \( \mathcal{X} \), respectively.

Definition 1. An XML tree \( \tau \) is an expression described by the following grammar:

\[
\tau ::= l [ \{ f_a \}, (f_e) ] \mid l[ v ]; \quad f_a ::= f_a , f_a \mid \@ l[ v ] \mid () ; \\
\quad f_e ::= f_e , f_e \mid \tau \mid () \]

where () denotes an empty expression, \( v \) stands for a value from \( D \) and \( l \) stands for a label from \( L \). An expression \( \@ l[ v ] \) is an XML attribute with a name \( l \) and value \( v \). An expression \( l[ \{ a_1, \ldots, a_m \}, (e_1, \ldots, e_n) ] \) is an XML element with a complex content with name \( l \), XML attributes \( a_1, \ldots, a_m \) and child XML elements \( e_1, \ldots, e_n \). The XML attributes must have distinct names from each other. An expression \( l[ v ] \) is an XML element with a simple content with a name \( l \) and simple content \( v \).

Example 1. A sample XML tree is depicted in Fig.1(b). Fig.1(a) shows the corresponding XML document.

Let us note that Definition 1 unifies the term XML tree and XML element – an XML tree \( \tau \) is in fact an expression which is an XML element at the same time. This XML element is called root XML element.