Exploiting Generalization for the Composition of Transformations of XML Schema Based Documents

Marek Lehmann

University of Klagenfurt
Dep. of Informatics-Systems
marek@isys.uni-klu.ac.at

Abstract. Open exchange of electronic documents in XML formats frequently requires the transformation to different XML-Schemas or DTDs (document type description). We propose a transformation system which eases the tedious production of such transformations, in particular XSLT transformations by composing (available) transformation programs. In this paper we show how the inheritance concepts of XML-Schema can be exploited to increase reuse of existing component transformations.

1 Introduction

The eXtensible Markup Language (XML) [2] has been widely accepted as a universal format for data interchange and publication, in particular for data published on the web or transmitted through web services. As XML is a meta-language, everyone can define in XML his or her own language using document type descriptions (DTD) or XML Schema [8]. Information systems dealing with information from other parties or sending information to others will have to manage this heterogeneity. In particular, it is frequently necessary to transform documents between different document types or schemas.

The most widely used and popular way to transform an XML document from one schema to another is to use XSLT transformations [4]. XSLT allows to write simple transformations for small documents quite easy, but complexity and error rates increase dramatically with the size of documents. For this reason writing of such transformations is very time consuming and tedious.

In [7] an approach was proposed to solve this problem for the DTD based documents by composing/decomposing XML transformations stored in a library and attaching them to components of XML documents. In [6] this approach was extended to XML Schema based documents. In particular the later proposal made use of the typing concept of XML Schema and linked XSLT transformations to XML Schema types.

In this paper we extend the approach presented in [6] by taking into account the concepts of type inheritance introduced by XML Schema. In [6] we proposed building of a library of well tested transformations between types. Before

a transformation is programmed, this library is searched for existing transformation, or at least transformations of components of the actual document. If such a transformation is not found, the IS_A hierarchy of types can be used to find transformations of closely related types.

2 XML Schema

Currently XML is widely used for documents exchange. Structure of these documents is mainly defined in document type declarations (DTD) [2]. But due to DTD's weaknesses, there are several other proposals of methods of defining schemas for XML documents [9]. One of the most promising is XML Schema, a new W3C recommendation [8] for defining the structure of XML documents. XML Schema specification defines few basic building blocks for designing schemas. Like in DTDs programmers can define elements and attributes. The most significant difference to DTDs is a possibility do derive new user defined types in the manner similar to the object oriented languages. We can distinguish two kinds of datatypes: simple datatypes and complex datatypes. The main difference between them is that simple types do not contain any child elements nor attributes and complex types have such possibility. Every element has to be of certain type – simple or complex. To simplify this idea we have introduced in an *AbstractType* class, which is not specified in [1].

New user defined types can be derived by restriction or extension of a base type. Simple types can be derived only by restriction of a value space of some existing simple type. Complex types can be derived both by restriction and extension of simple and complex types. In new complex type we can extend a structure of a base type by adding new attributes or subcomponents. We can also create a new complex type which is a subset of the base type. We can restrict a value space or a structure (e.g. restrict a number of occurrences of some elements or attributes).

XML Schema supports type substitutability. This means that any element of a base type can be substituted in a document instance by any element of a type derived from the base type. It is not important whether this type was derived by restriction or by extension. If we have a base complex type *PublicationType* and we derived from it another complex type called *BookType*, we can use instances of *BookType* instead of instances of *PublicationType*. But we have to inform the parser about this change by special a XML attribute `xsi:type` as on given example.

```
<Publication xsi:type="BookType">
  <Title>Quo Vadis?</Title>
  <Author>Henryk Sienkiewicz</Author>
  <Date>1997</Date>
  <ISBN>0781805503</ISBN>
  <Publisher>Hippocrene Books</Publisher>
</Publication>
```

3 Transformation Library

In [6] we presented the idea of decomposition of XSLT transformations and linking them to components described by XML Schema types. Every transformation was responsible for transforming elements of a given source type into elements of a given target type. We proposed to maintain a library of well tested transformations between types. Whenever a user wants to transform a document from one type into another type, our transforming system can look up a proper transformation in the library and apply it. The metamodel for our transformation library is presented on Fig. 1.

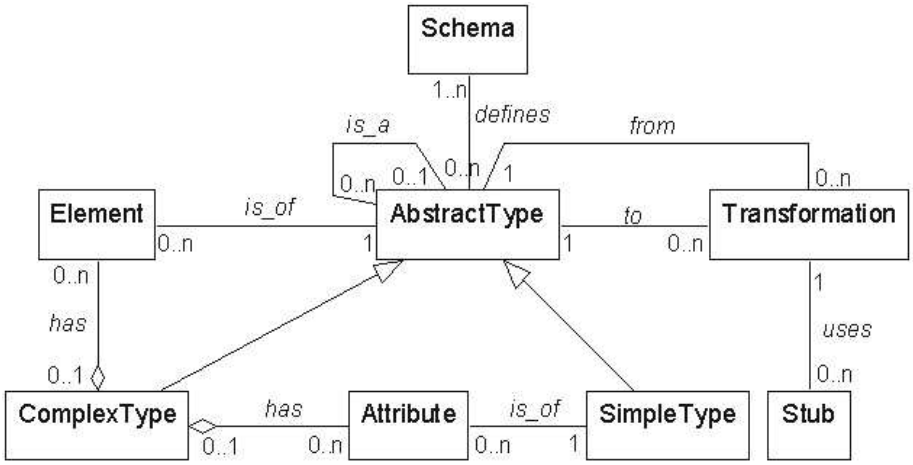

Fig. 1. Metamodel for a transformation library with hierarchies of types

The type substitutability in XML Schema gives us new possibilities in matching appropriate transformations. If any instance of a base type T can be replaced by an instance of a derived type $T´$, than any legal transformation of the type T should be valid for the type $T´$. We inherit transformations down the IS_A hierarchy. If we cannot find any transformation for the type $T´$ we can look for a transformation of its base type T. But type substitutability gives us also some possibilities with a target type. If we cannot find a correct transformation to the target type, we can try to transform our source to any type derived from the target type (including types derived transitively). Of course it is necessary to inform the parser about this change of the output document by adding a special XML attribute `xsi:type` to the result element. We must be aware that the type substitutability can also appear in the instance of the source document. The transformation system must check the real type of a source element being processed and look for the appropriate transformations.

4 Conclusions

We presented a new approach to transformations of XML Schema based documents. In particular we emphasized the following:

- Assigning source types and target types to transformations.
- A meta structure for storing XML Schema type's hierarchies and associated transformations in a library.
- Allowing to transform an element of a given type by transformations designed for types related to this given type in the IS_A hierarchy.

We shortly presented how to attach transformations to types and how to apply the knowledge about the type hierarchy to transform documents more efficiently. Our purpose was to minimize the effort in development process for creating transformations between XML documents. It is achieved by reusing existing transformations for given types and the types related to them through inheritance.

References

1. Paul V. Biron, Ashok Malhotra: *XML Schema Part 2: Datatypes.* W3C Recommendation 2 May 2001, http://www.w3.org/TR/xmlschema-2/
2. Tim Bray, Jean Paoli, C. M. Sperberg-McQueen, Eve Maler: *Extensible Markup Language (XML) 1.0 (Second Edition).* W3C Recommendation 6 October 2000, http://www.w3.org/TR/2000/REC-xml-20001006
3. K. Cagle, J. Ducket et al.: *Professional XML Schema.* Wrox Press, 2001
4. James Clark *XSL Transformations (XSLT) Version 1.0.* W3C Recommendation 16 November 1999, http://www.w3.org/TR/xslt
5. James Clark, Steve DeRose: *XML Path Language (XPath) Version 1.0.* W3C Recommendation 16 November 1999, http://www.w3.org/TR/xpath
6. J. Eder, M. Lehmann: *Composition of Transformations for XML Schema Based Documents.* Submitted for publication, 2003.
7. J. Eder, W. Strametz: *Composition of XML-Transformations.* In *Electronic Commerce and Web Technologies. Second International Conference (EC-Web 2001).* LNCS 2115, p. 71 ff., 2001
8. David C. Fallside: *XML Schema Part 0: Primer.* W3C Recommendation 2 May 2001, http://www.w3.org/TR/xmlschema-0/
9. Dongwon Lee, Wesley W. Chu: *Comparative Analysis of Six XML Schema Languages* ACM SIGMOD Record, Vol. 29, No. 3, September, 2000
10. Henry S. Thompson, David Beech, Murray Maloney, Noah Mendelsoh:. *XML Schema Part 1: Structures.* W3C Recommendation 2 May 2001, http://www.w3.org/TR/xmlschema-1/