Chapter I
Extending XML Types Using Updates

Béatrice Bouchou
Université François Rabelais Tours, France

Denio Duarte
Universidade Comunitária Regional de Chapecó, Brazil

Mírian Halfeld Ferrari
Université François Rabelais Tours, France

Martin A. Musicante
Universidade Federal do Rio Grande do Norte, Brazil

ABSTRACT

The XML Messaging Protocol, a part of the Web service protocol stack, is responsible for encoding messages in a common XML format (or type), so that they can be understood at either end of a network connection. The evolution of an XML type may be required in order to reflect new communication needs, materialized by slightly different XML messages. For instance, due to a service evolution, it might be interesting to extend a type in order to allow the reception of more information, when it is available, instead of always disregarding it.

The authors’ proposal consists in a conservative XML schema evolution. The framework is as follows: administrators enter updates performed on a valid XML document in order to specify new documents expected to be valid, and the system computes new types accepting both such documents and previously valid ones. Changing the type is mainly changing regular expressions that define element content models. They present the algorithm that implements this approach, its properties and experimental results.
INTRODUCTION

The main contribution of the World Wide Web is data exchange. The advent of web services has compelled researchers to investigate different problems concerning XML data when encapsulated, to be used over the network. Several technologies have been proposed in order to support service interactions. However, these technologies do not consider the significant problem of how to reconcile structural differences between types of XML documents supported by two different web services. Indeed, given two web services, they can be required to communicate by exchanging some XML documents. The type (schema) of the documents should be known by both parties. The successful composition of the services relays on this condition.

Sometimes, changes in the requirements of a service (or simple convenience of programming), promote modifications on the schema of documents used by the next version of a service (w.r.t. the ones produced by previous versions). This can affect the behaviour of the composed service, since the new documents produced or expected by the modified service do not match the agreed type. If the modifications are unavoidable, then the type of the documents, as expected by the other services, needs to be changed. As the new type is a modified version of the old one, we say that the new type is an evolution of the original one.

It would be interesting to achieve the evolution of this type in a validity-preserving way. More precisely, we would like to change our XML type in order to accept documents built on a slightly different format, without modifying the XML documents valid w.r.t. our original type. In other words, we would like to perform a conservative schema evolution.

Non-conservative schema evolution is problematic: documents valid for the original schema are no more guaranteed to meet the structural constraints described by the evolved schema. These documents should be revalidated against the new schema and, if they are not valid, they should be adapted to it. When documents to be revalidated are stored in different sites, not only their transfer cost should be considered (in addition to the whole revalidation cost) but also problems due to the access control should be faced. Several proposals consist in offering schema update primitives and, subsequently, in performing a transformation (with or without user interference) to the XML documents. Although some methods propose to revalidate only the parts of the documents involved in the schema updates, revalidation can still be considered as an expensive step of this schema evolution mechanism.

To achieve our goal, we foresee two important steps. The second one is the kernel of our work:

1. Compare two different XML documents $D$ and $D'$ in order to determine the structural differences between them. These structural differences can be expressed by means of the necessary update operations on $D$ to obtain $D'$.
2. Given a set of updates on $D$, if the update is incompatible with the type of $D$, adapt the type in order to obtain a more general type that meets both the document structure produced by the update and the original document structure.

The first step has been treated in the area of tree matching (for instance, in (Wang, Zhang, Jeong, & Shasha, 1994; Zhang, Statman, & Shasha, 1992)). The second step is, to our knowledge, much more unexplored, since other schema evolution approaches are not consistency preserving, i.e., they impose changes on documents which were valid w.r.t. to the original schema.

Different situations may require changing XML type in order to accept documents built on a slightly different format, without modifying the XML documents valid w.r.t. an original type. The following three points summarise our motivation on this domain.