Foundational Data Modeling and Schema Transformations for XML Data Engineering

Reema Al-Kamha¹, David W. Embley², and Stephen W. Liddle³

¹ Informatics Department, Damascus University, Syria
² Department of Computer Science, Brigham Young University, USA
³ Information Systems Department, Brigham Young University, USA

Abstract. As XML data storage and interchange become ubiquitous, analysts and data engineers increasingly need tools to model their data and map it to XML schemas and to reverse engineer XML documents and schemas in support of evolution and integration activities. For effective data management, model transformations require guarantees of properties of interest including guarantees of information and constraint-preservation, redundancy-free and compactness guarantees, and assurances about readability and maintainability. In this paper, we make foundational observations about XML data management, including conceptual modeling for XML data, transformations to and from XML Schema and XML data models, and transformation guarantees concerning properties of interest, and we provide resolutions for conceptual mismatches between XML data management and more traditional data management. Our implemented prototype tools show that these observations and insights can provide a strong foundation for XML data engineering.

1 Introduction

Because XML has become a standard for data representation, there is a need for a simple conceptual model for XML-based data engineering. But this is not enough—engineers also need a suite of design and development tools to map conceptual designs into implementable designs and to reverse-engineer legacy implementations to conceptual designs. In addition to facilitating these activities, the tools should guarantee certain desirable properties about generated implementations and should warn developers if such properties do not hold.

In building a suite of XML design and development tools, we face several interesting challenges. (1) Creating a conceptual model is a delicate balance between providing enough but not too many high-level conceptualizations without introducing low-level, implementation detail; making the model formal but easily understandable; and having a notation that is easily understood by developers and customers alike. (2) Once a conceptual model exists, the challenge becomes defining equivalence transformations to and from XML Schema—a nontrivial task because of the large conceptual mismatch. (3) Beyond just having transformations, XML data engineering demands certain guarantees. As a minimum,
the translations must preserve information content and, to the extent possible, preserve constraints. These guarantees should also enable developers to guarantee that forward translations yield storage structures that are redundancy-free and thus free of update anomalies, and that reverse translations yield faithful and understandable conceptual models.

In a keynote address [4] Carey challenged the conceptual-modeling community to develop conceptual models usable in XML data engineering. Several researchers have contributed to making this challenge a reality. Many have attempted to create or define characteristics for conceptual models for XML (e.g., [5,6,10,11,12,14,15,16]), but all have fallen short of capturing some interesting features of XML Schema. Some have attempted to provide transformations or design guidelines based on standard conceptual models or on XML-augmented conceptual models (e.g., [5,10,11]), but few address design properties such as being redundancy free or address them in a way to provide mapping guarantees, and none provide transformations for XML reverse-engineering.

In this paper, we build on our earlier work [2,3,8] and describe our implemented algorithms for conversions between a generic conceptual model and XML Schema. Based on these implementations, we explain how to meet the challenges of creating a suite of design and development tools for XML. In particular, we make the following contributions: (1) We argue for a few augmentations to traditional conceptual models to accommodate XML (Section 2). (2) We provide implemented equivalence transformations between XML-augmented conceptual models and XML Schema (Sections 3 and 4). (3) We show how to guarantee properties of interest in XML design and development (Sections 3 and 4).

2 C-XML: Conceptual XML

We show here how to extend traditional conceptual models for XML-based data engineering. In our prototype implementation, we have extended the conceptual modeling language OSM [7] for use with XML, resulting in C-XML (Conceptual XML). Figure 1 shows an example of a graphical rendition of a particular C-XML model instance for a small part of a student database.

All who have addressed the issue of creating a conceptual model for XML Schema argue for augmenting conceptual models with XML-like sequence and choice features. Although we disagree with previously-suggested ways of including sequence and choice features, we agree that both should be included. Some may argue that sequence and choice constitute low-level, implementation detail. We, and most others who have faced this problem, disagree. Order is a natural, high-level conceptualization among some entities (e.g. to represent component parts of addresses and start-end values for chromosome sequences), and alternatives are natural for others (e.g. to represent various forms of international addresses). We do, however, recommend only appropriate and conceptual use of sequence and choice—not the inappropriate and artificial use often seen in XML Schema because of its lack of modeling alternatives.

Fundamentally, a C-XML model instance is a hypergraph whose nodes are object sets and whose edges are relationship sets, which are often binary, but