Verification Theories for XML Schema

Suad Alagić, Mark Royer, and David Briggs

Department of Computer Science
University of Southern Maine
Portland, ME 04104-9300
{alagic, mroyer, briggs}@cs.usm.maine.edu

Abstract. XML Schema types and structures are represented as theories of a verification system, PVS, for proving properties related to XML schemas. Type derivations by restriction and extension as defined in XML Schema are represented in the PVS type system using predicate subtyping. Availability of parametric polymorphism in PVS makes it possible to represent XML sequences and sets via PVS theories. Transaction verification methodology is based on declarative, logic-based specification of frame constraints and the actual transaction updates. XML applications, including constraints typical for XML schemas, such as keys and referential integrity, have been verified.

1 XML Schema Types as PVS Theories

We describe a theorem prover technology for verifying properties related to XML schemas. We chose the PVS (Prototype Verification System [11]) theorem prover for our work because its type system includes predicate subtyping and bounded parametric polymorphism along with very general, and even higher order, logic capabilities. This makes PVS a suitable tool for expressing the complexity of XML Schema.

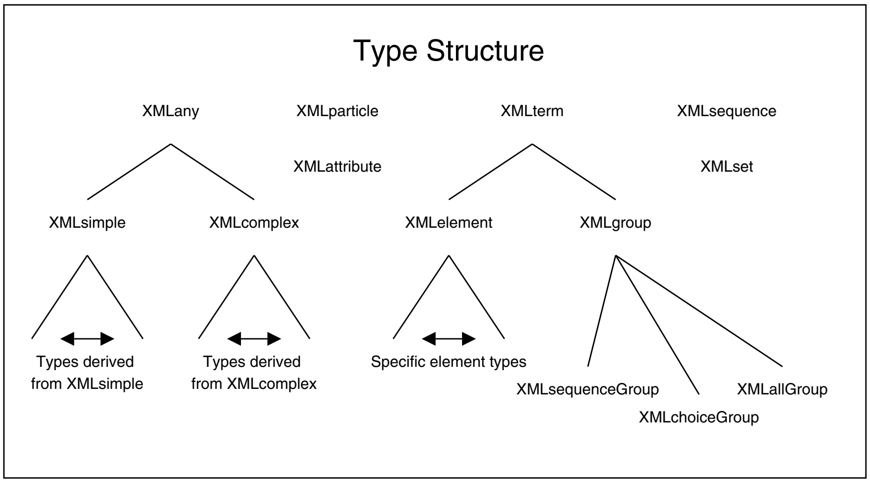
A PVS specification consists of a collection of theories. A theory is a specification of type signatures (of functions in particular) along with constraints applicable to instances of the theory expressed in the chosen logic. Hence in our approach, types and structures of the XML Schema have been represented as a collection of PVS theories.

We claim several advantages for this approach. First, structural properties are expressed in a type system that conforms to well-established type systems of programming languages with subtyping and parametric polymorphism. Second, complex rules specified in the XML Schema documents in semi-formal English are now specified in PVS theories much more precisely and formally in a suitable logic. Likewise, specification of a variety of constraints in an application schema is now both required and possible in a more general formal framework. The most important advantage is that PVS allows automated reasoning about properties expressed in its theories, even application properties that are not expressible in XML Schema. Thus, reasoning and verification are supported in situations when XML data is processed by a transaction or a general purpose programming language.

One particularly important application of a prover technology is verification that a transaction respects the integrity of a schema equipped with constraints. Our methodology for addressing this application requires explicit specification of frame constraints that the transaction does not affect, thus focusing the prover's attention on constraints that are at risk. The actual transaction update is specified in a declarative style as a binary predicate over pairs of database states, and the prover verifies that the update cannot violate the constraints. Although we use PVS to specify transaction updates, the methodology could be used with a variety of transaction languages.

2 Type Derivations in XML Schema

XML type anyType is the root of the XML type hierarchy [16, 17]. All other XML types are directly or indirectly derived from anyType by restriction or extension, and XML Schema has specific requirements on what derivations are valid. Two types that are derived directly from the type `XMLany` are `XMLsimple` and `XMLcomplex`. The subtyping relationships among XML types and our types specifying XML structures in PVS are represented in the following diagram:



A complex type is always derived from some other type, which may be either simple or complex. A complex XML type is equipped with a set of attributes and a content, which may be simple or complex. Complex content is specified via XML notions of elements, particles, groups, and group operators. Briefly, a complex content model determines a regular language of acceptable element instances, where the element tags are the symbols of the alphabet. A content derived by restriction will have a language that is a subset of the language of the base type, and a content derived by extension will have a language that is the concatenation of the base type's language with another language.

We define PVS theories for the XML constructs used in declaring XML complex types and predicates `extends` and `restricts` to formally capture the rules of XML Schema. A core idea behind type derivations in XML Schema is that an instance of a derived type may be viewed as a valid instance of its base type. This implies that all constraints associated with the base type are still valid when applied to an instance of a derived type. Our construction of PVS theories for XML Schema types and structures is governed by this basic requirement. This requirement corresponds to the notion of behavioral compatibility as presented in [4].

3 Related Research

The types as theories view is the basis of our previous results on generic data model management [1], semantics of objectified XML [3], and semantic compatibility problems for the object-oriented model [4].

A classical result on the application of theorem prover technology based on computational logic to the verification of transaction safety is [13]. Other results include usage of Isabelle/HOL [14] and PVS [2].

Results that address the problems of integration of a type system for XML with standard type systems [8, 9, 15] are confined to the problems of an integrated type system. These results do not address the issue of logic-based constraints, which is a distinctive feature of our work.

A variety of results are available on constraints for XML such as [7, 6, 10]. We consider XML constraints associated with a type system, and provide a prover technology to reason about constraints. This is probably the most distinctive feature of our work with respect to other related results.

4 Conclusions

Our experience with PVS had several lessons. First, intuitive techniques for verifying properties of transactions are inadequate. The PVS prover frequently exposed implicit assumptions we were making that were not logical consequences of the specifications. One advantage of using a prover tool is that it forces the developer to think more precisely and carefully about the application. Even when the goal theorem fails to prove, the prover gives valuable feedback to the developer. However, this feedback provided by PVS is not easy for a typical programmer to understand.

PVS does not check the consistency of a collection of axioms, and when possible such collections should not be used in writing specifications. We instead employ a definitional style which describes constraints as formulas, and then ask the PVS prover to show that the desired properties follow from the definitions.

A further conclusion is that tools such as PVS are not easy to use and require expertise and experience. A valid research goal is to develop proof strategies for particular tasks following the guidelines in [12, 5]. For a transaction verification proof strategy, a critical issue was separation of frame constraints from the logic-based specification of the actual updates. This strategy avoids expanding and rewriting the frame constraints and makes it possible to focus on the details of the proof of the active part of a transaction. In order to make these tools usable by typical programmers, a high-level user friendly interface based on suitable proof strategies is really required.

A major future research issue is extending this approach with reflective capabilities to allow the expression of XML features beyond conventional typing notions extended with constraints. In a separate piece of research we make use of a temporal logic specified by a suitable PVS theory in order to prove properties of object-oriented programs.

References

1. S. Alagić and P. A. Bernstein, A model theory for generic schema management, Proceedings of DBPL 2001, *Lecture Notes in Computer Science*, *2397*, pp. 228 - 246, 2002.
2. S. Alagić and J. Logan, Consistency of Java transactions, Proceedings of DBPL 2003, *Lecture Notes in Computer Science 2921*, pp. 71-89, Springer, 2004.
3. S. Alagić and D. Briggs, Semantics of Objectified XML, Proceedings of DBPL 2003, *Lecture Notes in Computer Science 2921*, pp. 147-165, Springer, 2004.
4. S. Alagić, S. Kouznetsova, Behavioral compatibility of self-typed theories, Proceedings of ECOOP 2002, *Lecture Notes in Computer Science 2374*, pp. 585-608, Springer, 2002.
5. M. Archer, B. Di Vito, and C. Munoz, Developing user strategies in PVS: A tutorial, Proceedings of STRATA 2003.
6. P. Buneman, S. Davidson, W. Fan, C. Hara and W-C. Tan, Reasoning about keys for XML, Proceedings of DBPL 2001, *Lecture Notes in Computer Science*, *2397*, pp.133 -148, 2002.
7. W. Fan and J. Simeon, Integrity constraints for XML, *Journal of Computer and System Sciences* 66, pp. 254-291, 2003.
8. H. Hosoya and B. Pierce, XDuce: A typed XML processing language, *ACM Transactions on Internet Technology*, 3(2), pp. 117-148, 2003.
9. H. Hosoya, A. Frisch, and G. Castagna, Parametric polymorphism for XML, Proceedings of POPL 2005, ACM, pp. 50-62.
10. G. M. Kuper and J. Simeon, Subsumption for XML types, Proceedings of ICDT, *Lecture Notes in Computer Science 1973*, pp. 331-345, Springer, 2001.
11. S. Owre, N. Shankar, J. M. Rushby, and D. W. J. Stringer-Clavert: PVS Language Reference, SRI International, Computer Science Laboratory, Menlo Park, California.
12. S. Owre and N. Shankar, Writing PVS proof strategies, Computer Science Laboratory, SRI International, http://www.csl.sri.com.
13. T. Sheard and D. Stemple, Automatic verification of database transaction safety, *ACM Transactions on Database Systems 14*, pp. 322-368, 1989.
14. D. Spelt and S. Even, A theorem prover-based analysis tool for object-oriented databases, *Lecture Notes in Computer Science 1579*, pp 375 - 389, Springer, 1999.
15. J. Simeon and P. Wadler, The Essence of XML, Proceedings of POPL 2003, ACM, pp. 1-13, 2003.
16. W3C: XML Schema Part 0: Primer, Second Edition, `http://www.w3.org/TR/xmlschema-0/`.
17. W3C: XML Schema Part 1: Structures, Second Edition, `http://www.w3.org/TR/xmlschema-1/`.