

Extended Abstract: Towards Verifying Semistructured Data

Gillian Dobbie¹

Jing Sun¹

Yuan Fang Li²

Scott UK-Jin Lee¹

¹ Department of Computer Science, The University of Auckland, New Zealand
Email: {gill,j.sun,scott}@cs.auckland.ac.nz

² School of Computing, National University of Singapore, Republic of Singapore
Email: liyf@comp.nus.edu.sg

1 Introduction

Semistructured data is now widely used in both web applications and database systems. There are many research challenges in this area, such as data integration, change management, view definition, and data normalization. Traditionally in these areas a formalism is defined for the database model, and properties of the algorithms can be reasoned about, such as the dependency preserving property of the normalization algorithm in the relational data model. Because research into semistructured data is still in its infancy, many algorithms have been defined in this area and a number of formalisms have been proposed but there is no widely accepted formalism that is generally accepted to reason about the properties of the algorithms. Such a formalism must capture all the necessary semantics required to model the algorithms, should not be too complex, and should be easy to use.

Another area that has been developing steadily is automatic verification. This involves formally specifying a model of a system, and running an automatic model checker or theorem prover that proves or disproves the consistency of the model. In this paper we describe research that we have undertaken in this direction. We have determined the semantics that we believe are required to reason about the properties of algorithms, we have modelled them using two different logics and used automatic verification to reason about the properties. This work is a first step towards establishing a widely accepted formalism, and it does highlight some important findings:

- the importance of the model containing enough semantics to express the algorithms over the data, but not excessive semantics,
- the model must be broken down into logical sections for understandability and extensibility,
- the importance of basing research into semistructured data on previous research in the database area rather than reinventing the wheel.

More specifically, we use a data modelling notation that extends the entity relationship (ER) data

model. The ORA-SS (Object Relationship Attribute data model for SemiStructured Data) data model models the schema and instance, so it is possible to model an XML Schema document in an ORA-SS schema diagram and an XML document in an ORA-SS instance diagram. However the semantics captured in an ORA-SS schema diagram are richer than those that are represented in XML Schema. We have modelled the semantics expressed in the ORA-SS diagrams in OWL (Web Ontology Language), which is based on a description logic, which itself can be translated into first order predicate logic. Because OWL was designed for sharing data using ontologies on the web, it was a natural starting point. OWL has an automatic reasoning tool called RACER (Renamed ABox and Concept Expression Reasoner). We also modelled the semantics expressed in the ORA-SS diagrams in PVS (Prototype Verification System), which is a typed higher order logic based verification system with a theorem prover. PVS can express more complex structures than OWL and in part, you get typing for free.

2 Related Work

Much of the work that has addressed challenges in the semistructured data area, such as (Li et al. 2006, Leonardi et al. 2006, Embley & Mok 2001), have proposed algorithms. However, there is little comparative analysis of algorithms that are designed for similar tasks. Because there is no widely accepted formalism, it is not possible to reason about the correctness, or show specific properties of the algorithms in a general way. Moreover it is difficult to compare properties of algorithms that are designed for similar purposes. For example it is difficult to compare the properties of the normalization algorithms that have been defined for semistructured data.

The area that our group is specifically interested in is normalization of semistructured data (or XML documents). There has been some very good and very practical work done in this area, such as (Embley & Mok 2001, Wu et al. 2001, Arenas & Libkin 2004, Wang & Topor 2005). If there was a widely accepted data model, with a set of defined operations it would be possible to show properties such as, whether data is lost during the transformations specified and it would also be possible to reason about what constraints are lost in the transformations. The kinds of formal definitions that we have seen to date in the normalization area include (Mani et al. 2001, Arenas & Libkin 2004, Vincent et al. 2004, Wang & Topor 2005).

Another area where there are similar transformations are view definitions. It would be helpful to be able to show that given a set of view transformations again no data is lost, and also show that particular operations are reversible. At one level, data integration can be thought of as creating a view over a set

This work is funded by a Marsden Grant (UOA317) from the Royal Society of New Zealand and also by the Singapore Millennium Foundation (SMF).

Copyright ©2007, Australian Computer Society, Inc. This paper appeared at Fourth Asia-Pacific Conference on Conceptual Modelling (APCCM 2007), Ballarat, Victoria, Australia, January 2007. Conferences in Research and Practice in Information Technology (CRPIT), Vol. 67. John F. Roddick and Annika Hinze, Ed. Reproduction for academic, not-for profit purposes permitted provided this text is included.

of schemas. The unified view that is formed can be defined by a set of operations and the unified view would be better understood if there were some way to state properties of the operations. The operations defined in change management, such as *insert*, *drop* and *move* can be defined formally. There should be operators for both changes to the data and changes to the schema. A formal definition would not only be useful when defining what changes to allow, but could also help in the definition of operations and perhaps in detecting the kinds of changes that have occurred between different versions of the data or schema.

There was an interesting workshop that highlights the need to bring together people who are working in foundations of semistructured data from different areas that are related to semistructured data (Neven et al. 2005). As you can see there have been a number of different approaches to defining foundations for semistructured data (e.g. (Milo et al. 2003, Jagadish et al. 2001), where most model the schema and data as a tree or graph and they are unable to model some of the constraints that can be specified in schema languages such as cardinality of children in relationships in the schema. These works consider limited semantics and do not provide automatic verification.

3 Background

The three key components in our work is the data modelling notation, ORA-SS, the ontology language OWL with the automatic reasoner RACER, and the formal verification language PVS. We briefly describe each of these components in this section.

3.1 ORA-SS (Object Relationship Attribute Model for Semistructured Data)

ORA-SS provides a notation for representing constraints on schemas and instances of semistructured data. The ORA-SS schema diagram extends ER diagrams with hierarchical parent-child relationships, and ordering. Schema diagrams represent relationship types between object classes, cardinality of relationship types, and distinguishes between attributes of object classes and attributes of relationship types. The ORA-SS instance diagram represents the same information as DOM diagrams, namely the relationships between objects and values of attributes. Objects match elements in XML documents, while attributes match leaf elements or attributes. A full description of the ORA-SS data modeling language can be found in (Dobbie et al. 2001, Ling et al. 2005).

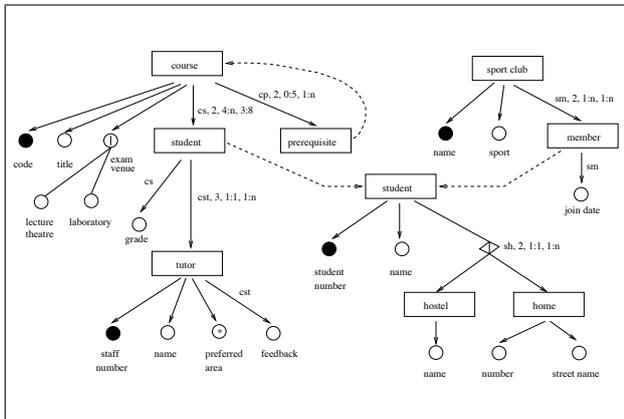


Figure 1: The ORA-SS schema diagram of a *Course-Student* data model

We will now highlight some of the salient points in the ORA-SS schema diagram in Figure 1. There

is a relationship type between object class *course* and object class *student*. It is a binary relationship type with name *cs*. Each course can have 4 to many students and a student can take 3 to 8 courses. Attribute *code* is an identifying attribute of course, and there is a reference between *prerequisite* and *course*, i.e. each prerequisite is in fact another course. The attribute *grade* belongs to the relationship type *cs*, i.e. it is the grade of a student in a course. Notice that relationship *cst* is a ternary relationship type, i.e. it relates object classes *course*, *student* and *tutor*.

3.2 OWL (Web Ontology Language)

OWL was designed to share data using ontologies over the web, and represents the meanings of terms in vocabularies and the relationships between those terms in a way that is suitable for processing by software.

Description logics (Nardi & Brachman 2003) are logical formalisms for representing information about knowledge in a particular domain. It is a subset of first-order predicate logic and is well-known for the trade-off between expressivity and decidability. Based on RDF Schema (Brickley & Guha 2004) and DAML+OIL (Connolly et al. 2001), the Web Ontology Language (OWL) (Horrocks et al. 2003) is the de facto ontology language for the Semantic Web. It consists of three increasingly expressive sub-languages: OWL Lite, DL and Full. OWL DL is very expressive yet decidable. As a result, core inference problems, namely concept subsumption, consistency and instantiation, can be performed automatically. RACER is an automatic reasoning tool for OWL ontologies, which supports min/max restrictions on integers, roles, role hierarchies, inverse and transitive roles.

3.3 PVS (Prototype Verification System)

PVS is a typed higher-order logic based verification system where a formal specification language is integrated with support tools and a theorem prover (Owre et al. 1992). It provides formal specification and verification through type checking and theorem proving. PVS has a number of language constructs including user-defined types, built-in types, functions, sets, tuples, records, enumerations, and recursively-defined data types such as lists and binary trees. With the language constructs provided, PVS specifications are represented in parameterized theories that contain assumptions, definitions, axioms, and theorems. Many applications have adopted PVS to provide formal verification support to their system properties (Lawford & Wu 2000, Srivas et al. 1997, Vitt & Hooman 1996).

4 Modelling ORA-SS Diagrams

The ORA-SS notation separates concerns naturally, separating the schema and the instance into individual diagrams. In our modelling we go a step further. We distinguish between:

- constraints that must hold on all schema diagrams,
- constraints that must hold on all instance diagrams,
- constraints that hold on the relationships between schemas and instances.

The kind of constraint that must hold on the schema is that a child object class must be either related to a parent object class to form a binary relationship or related to another subrelationship type to form an n-ary relationship. The schema in Figure 2(a)

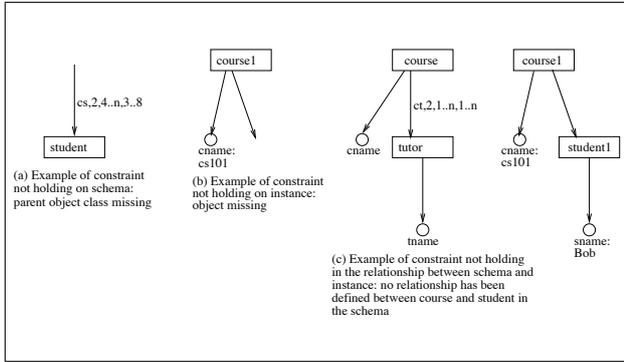


Figure 2: Examples of constraints not holding in ORA-SS diagrams

violates this constraint. An example of a constraint that must hold on all instance diagrams is that a relationship is between 2 or more objects. Figure 2(b) shows a relationship in an instance diagram, where there is no child object. An example of a constraint that must hold on the relationship between schemas and instances is that if there is a relationship between two objects in an instance, then there must be a relationship type between the related object classes at the schema level. Figure 2(c) shows an instance with a relationship between objects *course1* and *student1*. Assuming that *student1* is a student and not a tutor, then this relationship violates the relationship type in the corresponding schema diagram. Although the mistakes appear obvious in the simple examples in Figure 2, they are harder to see in more complex diagrams or when the data is in a different format, such as XML and XMLSchema.

For each of the models described above, we have a corresponding instance model. There are models for:

- the instance of the schema,
- the instance of the instance,
- the relationship between the instances of the schema and the instance.

In the instance of the schema, it is possible to state constraints such as *course* is an object class. In the instance of the instance, it is possible to state that *course1* is an object, and in the relationship between instances of the schema and the instance it is possible to state that *course1* is a *course*. Figure 3 summarizes the components that make up the ORA-SS model.

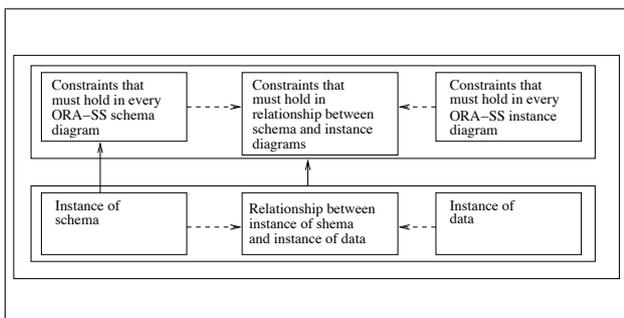


Figure 3: Components of the Model

5 Conclusion

Much of the research in the area of semistructured data involves algorithms that transform the data and

schema, such as data integration, change management, view definition, and data normalization. The work presented in this paper is a first step towards formally defining a data model that is rich enough to capture the semantics that are needed to model and reason about the properties of operations that are capable of describing the algorithms mentioned above. We have modelled the ORA-SS data model using OWL and PVS, and conclude that:

- Semantic Web languages, such as OWL, are extremely good at capturing semantic information about semistructured data.
- Reasoning tools, such as Racer, can be used to check the consistency of the ORA-SS schema and instance diagrams.
- PVS can be used to define a formal mathematical semantics for semistructured data.
- The automated verification provided by PVS empowers our definition of ORA-SS semantics, and if designed well it can be easy to extend.

There are a number of directions that we wish to take this work. Firstly we will study the algorithms that have been defined for the normalization of semistructured data, and derive the basic operations that are necessary to describe the algorithms. Secondly we will extend the OWL and PVS models of ORA-SS diagrams with a formal definition of the basic operations to compare how each model of ORA-SS performs. Thirdly, we will test the success of the models by modelling at least two of the normalization algorithms and comparing the properties of each. Finally we will investigate if the operators that are defined for normalization are general enough to express the general problem of view definitions.

References

- Arenas, M. & Libkin, L. (2004), 'A Normal Form for XML Documents', *ACM Transactions on Database Systems* **29**(1), 195–232.
- Brickley, D. & Guha, R. (2004), 'Resource description framework (rdf) schema specification 1.0'. <http://www.w3.org/TR/rdf-schema/>.
- Connolly, D., van Harmelen, F., Horrocks, I., McGuinness, D., Patel-Schneider, P. & Stein, L. (2001), 'Reference description of the DAML+OIL ontology markup language'. <http://www.w3.org/TR/daml+oil-reference>.
- Dobbie, G., Wu, X., Ling, T. & Lee, M. (2001), ORA-SS: Object-Relationship-Attribute Model for Semistructured Data, Technical Report TR 21/00, School of Computing, National University of Singapore.
- Embley, D. W. & Mok, W. Y. (2001), 'Developing XML Documents with Guaranteed "Good" Properties', *Proceedings of the 20th International Conference on Conceptual Modeling*.
- Horrocks, I., Patel-Schneider, P. & van Harmelen, F. (2003), 'From SHIQ and RDF to OWL: The making of a web ontology language', *J. of Web Semantics* **1**(1), 7–26.
- Jagadish, H., Lakshmanan, L., Srivastava, D. & Thompson, K. (2001), TAX: A tree algebra for XML, in 'Proceedings of the International Workshop on Database Programming Languages', pp. 149–164.

- Lawford, M. & Wu, H. (2000), Verification of real-time control software using PVS, *in* P. Ramadge and S. Verdu, ed., 'Proceedings of the 2000 Conference on Information Sciences and Systems', Vol. 2, Dept. of Electrical Engineering, Princeton University, Princeton, NJ, pp. TP1-13-TP1-17.
- Leonardi, E., Hoai, T. T., Bhomwick, S. S. & Madria, S. (2006), 'DTD-Diff : A Change Detection Algorithm for DTDs', *Proceedings of the International Conference on Database Systems for Advanced Applications*.
- Li, C., Ling, T. W. & Hu, M. (2006), 'Efficient processing of updates in dynamic XML data', *Proceedings of the International Conference on Data Engineering*.
- Ling, T. W., Lee, M. L. & Dobbie, G. (2005), *Semistructured Database Design*, Springer-Verlag.
- Mani, M., Lee, D. & Muntz, R. (2001), Semantic Data Modeling Using XML Schemas, *in* 'Proceedings of the 20th International Conference on Conceptual Modeling', LNCS 2224.
- Milo, T., Suciu, D. & Vianu, V. (2003), 'Typechecking for XML Transformers', *J. Comput. Syst. Sci.* **66**(1), 66-97.
- Nardi, D. & Brachman, R. (2003), An introduction to description logic, *in* F. Baader, D. Calvanese, D. McGuinness, D. Nardi & P. Patel-Schneider, eds, 'The description logic handbook: theory, implementation, and applications', Cambridge University Press, pp. 1-40.
- Neven, F., Schwentick, T. & Suciu, D., e. (2005), *Foundations of Semistructured Data.*, Vol. 05061, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany.
- Owre, S., Rushby, J. M., & Shankar, N. (1992), PVS: A Prototype Verification System, *in* D. Kapur, ed., '11th International Conference on Automated Deduction (CADE)', Vol. 607 of *Lecture Notes in Artificial Intelligence*, Springer-Verlag, Saratoga, NY, pp. 748-752.
- Srivias, M., Rueß, H. & Cyrluk, D. (1997), Hardware Verification Using PVS, *in* T. Kropf, ed., 'Formal Hardware Verification: Methods and Systems in Comparison', Vol. 1287 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 156-205.
- Vincent, M., Liu, J. & Liu, C. (2004), 'Strong Functional Dependencies and their Application to Normal Forms in XML', *ACM Transactions on Database Systems* **29**(3), 445-462.
- Vitt, J. & Hooman, J. (1996), Assertional Specification and Verification Using PVS of the Steam Boiler Control System, *in* J.-R. Abrial, E. Boerger & H. Langmaack, eds, 'Formal Methods for Industrial Applications: Specifying and Programming the Steam Boiler Control', Vol. 1165, Springer-Verlag, pp. 453-472.
- Wang, J. & Topor, R. (2005), Removing XML Data Redundancies Using Functional and Equality Generating Dependencies, *in* 'Proceedings of the Australasian Database Conference', CRPIT.
- Wu, X., Ling, T. W., Lee, M. L. & Dobbie, G. (2001), Designing Semistructured Databases Using the ORA-SS Model, *in* 'WISE '01: Proceedings of 2nd International Conference on Web Information Systems Engineering', IEEE Computer Society, Kyoto, Japan.