Symbolic Modelling of Database Representations

Alina Andreica, Daniel Stuparu, Iulia Mantu
ICT Department, “Babeș-Bolyai” University
Cluj-Napoca, Romania
{alina, dstuparu, iuliam}@staff.ubbcluj.ro

Abstract

The paper proposes a symbolic model for fundamenting the transformations between the relational database form and its XML representation. The model we propose aims at proving the consistency of such transformations, which are often used in software applications that process databases.

On a more abstract level, our aim is to show that the categorial theory developed in symbolic computation can offer tools for systematically tackling the basis of fairly complex problems which rise in software design.

Keywords: category theory, symbolic modelling, databases, XML representations of databases, representation consistency

AMS Subject Classification: 68Q40, 68P05

1. Defining the working framework

We believe that the XML representation of a database is very important because it provides a portable hierarchical form for the database, which successfully complements the classical representation and provides, in certain situations, processing advantages: independence of the database engine – portability, explicit hierarchical form, simple access to database attributes, efficiently communicating results of database queries between application modules.

The XML representation of a database and the algorithms for obtaining it [4] already formulate a solution for this practical problem. But we consider that this solution can be enhanced with a theoretical basis, i.e to prove that this representation transformation is consistent. In this respect, we propose a model based on category theory in symbolic computation: we define the domain of relational databases and the domain of XML representation of databases in a functional manner in order to prove the isomorphism between the two domains.

We consider that the category theory in symbolic computation and the tools for implementing it, such as Theorema functorial facilities [7] are powerful tools for building systematic solutions in various areas of computer science and software design.

This statement is sustained by the fact that category theory in symbolic computation introduces techniques for implementing general working contexts of performing symbolic algorithms. The defined categories can later be particularized for various domains, by elegantly using the same definition. Definitions for categories and domains, from the symbolic point of view, can be found in [3].

In Theorema, B. Buchberger introduces a powerful tool for “carrying” this semantics - the functors, which map domains into other domains. The use of functors has supplemental advantages, since their syntax, condensing their essence, is very useful in proving [5].

The model we further propose shows how the principles of category theory can be applied in systematically describing the phenomenon of transforming database representations into xml representations.

2. Advantages of the XML Representation of a Database

The XML (eXtended Markup Language) [8] representation has considerable advantages in structuring data, which are very important both in processing and communicating data. The use of XML for structuring the messages that are communicated between application modules or in client-server design has proven to be very efficient and portable, since XML messages are structured on uniform principles, can be easily processed and rapidly transmitted.
Compared to a database structure [8], XML displays certain similarities (storage facilities, schemas, interrogation languages – like XQuery, XPath, XML, programming interfaces etc.). On the other hand, XML lacks certain powerful facilities offered by relational databases: efficient storage, indexes, security, transactions, data integrity, distributed access, triggers, etc.

The natural conclusion is that the relational database structure and the XML structure are to be used in parallel, depending on the aims of the processing.

The XML structure of a database was introduced in order to use the advantages of the XML representation when processing and communicating large amount of data structured in databases. There is already a number of references in the field; a clear description for an XML representation of a database and programming this transformation is given in [4].

Nevertheless, the theoretical foundation for proving the consistency of these data representation transformations in a systematic manner is less tackled. In this respect, we consider that the categorical model based on symbolic computation principles we further propose is a novelty in the field

3. Definitions for the Relational and the XML Representations of Databases

Within this paragraph we define the domain of databases and the domain of databases in XML representation in a categorical style, in order to further use it in symbolic computation processings.

3.1. Defining the domain of databases

The well-known relational model of a database was introduced in 1970 by Codd. Within this paragraph, we shall refer this relational model, even if we sometimes omit the attribute “relational”.

Taking into account the relational database definition given in [2], we give the following definition for the domain of databases:

**Definition 3.1.** We define the domain of databases

\[ \text{DB} := \bigcup \{ \text{db} \mid \text{is} \sim \text{database(db)} \} \]

where a database \( \text{db} \) is defined as:

\[ \text{is-database(db)} \iff \text{db} \in \bigcup \{ \text{Schema} \mid \text{is} \sim \text{schema(Schema)} \} \]

A database schema from a database \( \text{db} \) is defined as:

\[ \text{is-Schema} \iff \bigcup_{\text{Table} \in \text{db}} \{ \text{Table} \mid \text{is} \sim \text{Table(Table)} \} \]

Given a set of attributes \( \text{Attr} \) and a set containing sets of attribute values \( D \), we define a column as a function mapping an attribute into the set containing its corresponding values:

\[ \text{ValColumn} : \text{Attr} \rightarrow D, \text{ValColumn(a)} := \{ d \mid d \in D, d \text{ is a value for attribute 'a'} \} \]

Given \( n \) sets of attributes \( \text{Attr}_i, i=1,\ldots,n \) and \( n \) sets containing sets of attribute values \( D_i, i=1,\ldots,n \), we define a table \( T \) from a database scheme by:

\[ \text{is-Table}(T, \text{Attr}(i), i=1,\ldots,n, D(i), i=1,\ldots,n) \iff \]

\[ T \in \bigcup_{\text{Attr}} \{ \text{ValColumn(Attr)} \}, \text{where} \]

\[ \text{Card}(\text{ValColumn(Attr)}) = nrw = \text{NoRows}(T) \text{ the number of lines in table } T, i=1,\ldots,n, n= \text{NoCol}(T) \text{ the number of columns in table } T \]

We also introduce the notations:

\[ \text{SchemaDB} = \bigcup_{\text{db} \in \text{DB}} \{ \text{Schema} \mid \text{is} \sim \text{Schema(Schema)} \} \]

the domain of schemes from the databases’ domain \( \text{DB} \)

\[ \text{TableDB} = \bigcup_{\text{Schemadb} \in \text{SchemaDB}} \bigcup_{\text{Table} \in \text{Schemadb}} \{ \text{Table} \mid \text{is} \sim \text{Table(Table)} \} \]

the domain of tables from the domain of schemes included in the databases’ domain \( \text{DB} \).

3.2. Defining the domain of XML representations of databases

Based on the XML structure of a database given in [4], we propose the following framework for the XML structure of a database \( \text{DatabaseName} \), including the schemas \( \text{SchemaName}_i \) containing the tables \( \text{TableName}_i \) (with columns \( \text{ColumnName}_i \) (values \( \text{Val}_i \), \( \text{Val}_2 \), ...) \( \text{ColumnName}_j \) (values \( \text{Val}_j \), \( \text{Val}_k \), ...), \( \text{TableName}_2 \), ...; \( \text{SchemaName}_2 \), ...

\[ <\text{DATABASE}_{\text{DatabaseName}}> <\text{SCHEMA}_{\text{SchemaName}_i}> <\text{TABLE}_{\text{TableName}_i}> <\ROW> <\COLUMN_{\text{ColName}_1}> \text{Val}_1 </\COLUMN_{\text{ColName}_1}> <\COLUMN_{\text{ColName}_2}> \text{Val}_2 </\COLUMN_{\text{ColName}_2}> \ldots </\ROW> <\ROW> <\COLUMN_{\text{ColName}_1}> \text{Val}_1 </\COLUMN_{\text{ColName}_1}> <\COLUMN_{\text{ColName}_2}> \text{Val}_2 </\COLUMN_{\text{ColName}_2}> \ldots </\ROW> \ldots \]
Definition 3.2 We define the domain of XML representation of databases DBxml by means of a functor which transforms the domain of databases (definition 3.1) into DBxml:

\[
\text{TaggedBD} : DB \rightarrow DBxml \\
\text{Tagged}(bd) := <\text{DATABASE}_\text{DatabaseName}> \text{TaggedSchema}(bd)</\text{DATABASE}_\text{DatabaseName}>
\]

\[\text{DBxml} = \bigcup_{bd \in DB} \text{TaggedDB}(bd)\]

\[
\text{TaggedSchema} : DB \rightarrow DBxml \\
\text{TaggedSchema}(bd) := \\
\bigcup_{SName \in bd} <\text{SCHEMA}_\text{SchemaName}> \text{TaggedTable}(SName)</\text{SCHEMA}_\text{SchemaName}>
\]

\[
\text{TaggedTable} : \text{SchemaDB} \rightarrow \text{SchemaDBxml} \\
\text{TaggedTable}(st) := \\
<\text{TABLE}_\text{TableName}> \bigcup_{TabName \in st} (<ROW> \text{TaggedColumn}(TabName,j) <ROW>)</\text{TABLE}_\text{TableName}>
\]

SchemaDBxml is the database scheme domain in XML representation.

\[
\text{TaggedColumn} : \text{TableDB} \times \mathbb{N} \rightarrow \text{TableDBxml} \\
\text{TaggedColumn}(T,j) = \\
\bigcup_{i=1}^{\text{NoCol}(T)} <\text{COLUMN}_\text{Attr}_i > \text{ValColumn}(\text{Attr}_i) <\text{COLUMN}_\text{Attr}_i > \\
\text{where } n = \text{NoCol}(T), \text{Attr}_i \text{ and ValColumn(Attr}_i), i=1,\ldots,n \text{ defining the table } T \text{ are used in the sense of definition 3.1, ValColumn}(a) \text{ is the } j^{th} \text{ component of ValColumn(a)} \text{ viewed as an ordered set of values and TableDBxml denotes the table domain TableBD in XML representation.}
\]

4. Theorema Modelling for Database Representations

Within this paragraph we put the basis of a Theorema proof in order to show that the domain of databases DB and the domain of databases in XML representation DBxml are isomorphic. DBxml is constructed in a functional manner, based on the databases domain DB.

We extend the database domain DB (definition 3.1) with a relation order ≤ defined as the lexicographical order on the identifiers which appear in the domain. By abuse of notation, we denote this domain also with DB. We also extend the mapped DBxml domain in the same manner. This "extension" enables us to use in the definitions of DBxml (which implement definition 3.2) Theorema tuples instead of union.

Figures 1-3 give Theorema definitions for the databases' domain:

Definition "ValColumn", any
\[
\text{ValColumn}(a) := \{d | d \in D\}
\]

Definition "is–table", any
\[
\text{is–table}(T, n) := \{\text{Attr}_i | i = 1, \ldots, n\}, \text{ValColumn(Attr}_i) = \text{nrw}, i = 1, \ldots, n\}
\]

Definition "TableDB", any
\[
\text{TableDB} := \bigcup_{S \in \text{SchemaDB}} \bigcup_{T \in S} \{\text{Table} | \text{is–Table}(T)\}
\]

Figure 1: Theorema definitions for tables and the domain of tables

Definition "is–Schema", any
\[
\text{is–Schema}(S) := \{\text{Table} | \text{is–Table}(\text{Table})\}
\]

Definition "SchemaDB", any
\[
\text{SchemaDB} := \bigcup_{bd \in DB} \bigcup_{S \in \text{bd}} \{S | \text{is–Schema}(S)\}
\]

Figure 2: Theorema definitions for schemas and the domain of schemas
Figure 3: Theorema definitions for databases and the domain of databases

Figures 4-6 give Theorema definitions for the domain of databases in XML representation:

Figure 4: Theorema definitions for tagged tables

Figure 5: Theorema definitions for tagged schemas

Figure 6: Theorema definitions for the domain of databases in XML representation

Figure 7: Theorema definition for bijective mapping

Using the above described definitions, the goal is to prove:

is-bijective[TaggedDB, DB, DBxml]

5. Conclusions and Future Work

The paper proposes a model for proving the consistency of transforming the relational representation into a XML structured form. The model, which is still to be refined in our future work, is based on symbolic computation and category theory principles and reveals the power of such tools in tackling various computer science problems.

We consider that the category theory and its symbolic computation implementation are powerful tools for systematically dealing with complex problems that rise not only in symbolic computation but also in software design and that further applications in this field are to be found.

6. References

[8] http://www.s-concept.net/s-concept/ro/2,-1,-1,1,2,0.html