A Unified Framework for Data Translation over the Web *

RICCARDO TORLONE and PAOLO ATZENI
DIA, Università Roma Tre
via della Vasca Navale, 79
00146 Roma, Italy
{torlone,atzeni}@dia.uniroma3.it

Abstract

In this paper we propose a comprehensive framework for the management and the exchange of (semi) structured Web data, described according to a variety of formats and models. We consider various schema definition languages for XML (DTD, XML Schema and XDR) a model for semi-structured data (OEM) and a model used to store Web data (the relational model) and show that the primitives adopted by all of them can be classified into a rather limited set of basic types. We then define, building on these basic types, a notion of "meta-formalism" that can be used to describe, in a uniform way, these heterogeneous representations of Web data.

In this framework, the translation of schemes and instances between different models are based on the translations of the involved primitives. Complex translations can be then obtained by simply combining a number of predefined operations, which implement standard translations between primitives. Moreover, we show that, for translating between any pair of models, it is sufficient to define a number of translations which is linear in the number of models.

These results can be used to support a number of involved Web-related activities like: information exchange between different organizations, integration of data coming from heterogeneous information sources, storage of native XML data in a DBMS and publishing of existing structured (relational) data in XML.

1. Introduction

XML is rapidly emerging as the new standard for data representation and exchange on the Web. Nevertheless, a consensus on the most convenient and effective way to represent the structure (in database terminology, the "scheme") of XML data is not emerging yet.

*This work was partially supported by *IASI-CNR* and by *MURST* within DataX project.

Actually, it is widely recognized that DTDs (Document Type Definition [10]), which summarize document contents by means of grammars, are often inadequate to represent structured information in form of XML documents. For this reason, several formalisms have been recently introduced to describe the structure of XML data in a more expressive and powerful way, most of them inspired by database modeling principles. However, in spite of the efforts of W3C to define standards in this context, a proliferation of models often varying for minor details has been witnessed (for a survey and a comparison see [21]). Among them, we cite (beyond DTD): XML Schema [8, 28] (W3C), XDR [16] (XML Data Reduced — Microsoft), RELAX (REgural LAnguage description for XML) [26], SOX [13] (Schemas for Object Oriented XML — Commerce One), DSD [20] (Document Structure Description — AT&T), DCD [11] (Document Content Description — W3C) and Schematron [19]. We have also to include in this collection several "semi-structured" models, which have been proposed to represent XML data [1].

It follows that the exchange of information between different and heterogeneous information sources can be difficult to achieve, even if we possess knowledge about their content, because of the difference in syntax and expressive power of the formats used to describe this content. To fulfill this need, a number of tools for the translation between different XML schema languages have been already proposed. For instance, XDRtoXSD [18], which translates XDR into XML schema definitions, and XMLSpy [29], which is able to translates between various (but specific) XML schema formalisms.

As a matter of fact, this problem fits into a larger and more relevant problem: that of the exchange of structured information between contexts (applications and/or tools) that adopt heterogeneous data models. One example (but relevant and largely studied in the context of XML) is the problem of the translation of XML data into a logical data model (e.g., the relational model or an object based model) with the purpose of permanently store such data [15]. The definition of a unified framework for the management and the translation of data represented according to heterogeneous data models is a topic largely investigated in literature. We refer, in particular, to the recent works on the "superimposed" information models [14, 22], on the "Model management systems" [5, 6, 7, 9] and on the techniques for data conversion [9, 12, 23, 24, 25, 27].

In this framework, the goal of our current research (inspired by our previous work on management of multiple models in database design tools [2, 3]) is the definition of methods and techniques for the management of a wide variety of model-based applications over the Web in a uniform way and to allow data translation from one representation to another. The first step in this trend is the definition of a "meta-formalism" that is able to capture both the main primitives adopted by different schema languages for structured data (XML in particular) over the Web and the basic constructs used by traditional database models to represent such information. The subsequent, and more concrete step is the definition of an effective (and possibly efficient) method for the translation between heterogeneous data representations that makes use of the meta-formalism as a level of reference. In the framework we have defined, the translation of schemes and, if needed, of the corresponding instances, can be effectively specified on the basis of translations of the involved types of meta-primitives. This makes a translation general and independent of the specific models on which it operates. Another interesting aspect is the fact that a number of general properties of translations can be analyzed on the basis of the properties of the model involved. These results can be used to support a number of involved Web-related activities like: information exchange between different organizations, integration of data coming from heterogeneous information sources, storage of native XML data in a DBMS and publishing of existing structured (relational) data in XML.

In this paper we report on the preliminary steps of this research. Specifically, we first present the general principles on which the work is based and then illustrate a first version of the metamodel. We also show, by means of a number of examples, that an even simple and relatively compact metamodel is able to capture the main features of different formalisms (DTD, XML Schema, XDR, OEM and the relational model). We then describe our approach to the problem of translation of heterogeneous data and discuss about important properties of such translations that need to be further investigated.

The rest of the paper is organized as follows. In Section 2 we provide a general overview of our approach. In Section 3 we present a first version of the metamodel, which is far to be complete but is however able to capture relevant features of several formalisms for XML schemes. Then, in Section 4 we discuss about properties and techniques for the translations between different formalisms and finally, in Section 5 we draw some conclusions.

2. An overview of the approach

The scenario of reference for our study is reported in Figure 1.

Four levels of abstractions can be identified. At the first level we have a number of *instances*, which contain the actual data. We assume here that data is organized according to some (semi) structured format used in the process of information exchange through the Web (some exemplars are indicated in the figure). At the second level we have *schemes*, which describe the structure of the instances. Then, we have different formalisms for the description of schemes, which we call *models* hereinafter. Finally, at the top level we have one *metamodel*, that is, a formalism for the definition of the various models. In the environment we have in mind, the number of models is not fixed: new models for semi-structured Web data can be dynamically added to the framework. Clearly, it may happen that the metamodel is not enough expressive to capture a specific feature of a new model. In this case, we assume that the metamodel can be extended accordingly.

In the following, we will use the term *element* to refer to components of a scheme, the term *primitive* to refer to components of models, and the term *metaprimitive* for components of the metamodel. As an example, given an XML document containing a description of persons according to a given XML Schema, there may be an element *person* composed by an unordered collection of sub elements (such as *name* and *age*). Then, the corresponding primitive is *all* (which is the tag used in XML Schema to denote an unordered tuple of sub elements) and the corresponding metaprimitive is *unordered sequence*. To refer to the primitive used to define an element we will use the phrase *type of the element*; similarly, we will use *type of the primitive* to indicate the metaprimitive corresponding to a primitive. In the example, the type of the element *person* is the primitive *all*, whose type is in turn the metaprimitive *unordered sequence*. We note that, since we consider in the same framework both traditional data models and schema languages for structured documents, we use the term "primitive" in a quite broader sense: a primitive can be a database construct, like set or aggregation, but it can also be a regular expression operator, like the operator $+$ or $*$ used in DTDs.

In this framework, translations between schemes and instances would occur as follows. For any two models M_1 and M_2 defined with the metamodel, and for each instance I_1 (the *source*) of a scheme S_1 (the *source scheme*) for M_1 (the *source model*), it should be possible to obtain an instance I_2 (the *target*) of a scheme S_2 (the *target scheme*) for M_2 (the *target model*) containing the same information

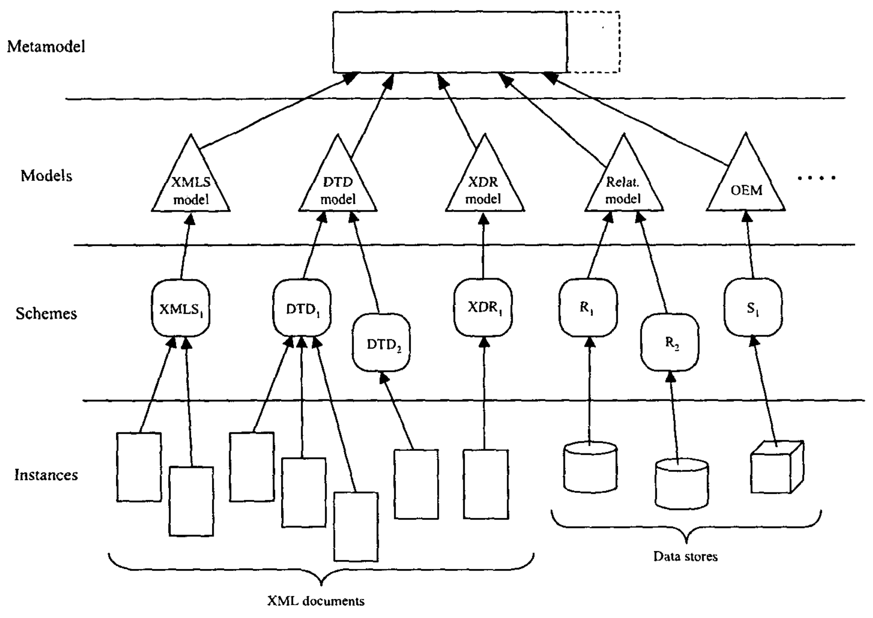

Figure 1. The reference scenario

as I_1. We say that I_2 (S_2) is the *translation* of I_1 (S_1) and into M_2.

The general approach to the definition of the environment we have illustrated follows from a former study on the management of heterogeneous conceptual data model [2, 3] and is based on the following principles.

- All the primitives used in most known formalisms for expressing Web data fall in a rather limited set of categories [21] (e.g., base types, ordered sequence, unordered sequence, choice, cardinalities and some others). At the same time, many of these primitives correspond to well known constructs used in conceptual data models [17] (e.g., lexical type, sequence, aggregation, disjoint union, set). Therefore, a metamodel able to capture the main features of such models can be defined by means of a basic set of meta-primitives, corresponding to the above categories. Then, a model can be described by classifying its primitives according to the metaprimitive in the metamodel. It can be argued that this approach is not "complete", as it does not cover all possible models, but it is however easily extensible: should a model with a completely new primitive be proposed, the corresponding type could be introduced in the metamodel.

- Since there is no clear notion of when a translation is correct we follow a pragmatic approach. We assume that the primitives that correspond to the same metaprimitive have the same semantics, and then we define translations that operate on individual primitives (or simple combinations thereof) as follows: for each primitive x used in the source scheme such that there is no primitive of the same type in a target model M, we replace x by other primitives of M. This work can be profitably supported by a predefined, "built-in" set of elementary transformations implementing standard translations between primitives, which we assume to be correct by definition. These translations operates both at scheme and at instance level. More specifically, they include a scheme restructuring and a corresponding data mapping. For instance, an ordered sequence of tuples s can be transformed into a relation r having the same tuples, each of which has a further attribute that codes the order. A complex translation can be then obtained as composition of elementary steps. Moreover, we will show that for translating between any pair of models in the framework, it is sufficient to define only one translation for each model.

3. The metamodel

3.1. Basic constructs

Let us consider the core of three schema formalisms for XML (DTD, XML Schema and XDR) and two data models often used in the context of Web data: the relational model, as a representative of a value-based logical model, and OEM, as a representative of a semi-structured model. It turns out that a metamodel for these representatives includes a quite limited set of metaprimitives, as follows.

1. *Base type*: it corresponds to a primitive whose instances are printable values; for example, *#PCDATA* and *CDATA* are the only base types available in DTD's.

2. *Object type*: it corresponds to a primitive whose instances are objects having a unique oid; the structure of an object is defined using other primitives; for example, this is the base primitive used in OEM to define instances.

3. *Ordered sequence*: it corresponds to a primitive whose instances are ordered sequences of instances of other primitives, called the *components* of the sequence. For example, the concatenation operator is the primitive used to define ordered sequences in DTD's.

4. *Unordered sequence*: it corresponds to a primitive whose instance are ordered sequences of instances of other primitives, called the *components* of the sequence. For example, the *all* operator is the primitive used to define unordered sequences in a XML Schema;

5. *Choice*: it corresponds to a primitive whose instances can be chosen among instances of other primitives. For example, the | operator is the primitive used to define choices in a DTD.

6. *Cardinality*: it is expressed as a pair of values (Min, Max) and is associated with a primitive of type 3, 4 or 5; it corresponds to a primitive whose instances are sets of instances of the primitive associated with it; these sets must have a cardinality included between *Min* and *Max*. For example, the $+$ operator is the primitive used to define a cardinality $(1, N)$ in a DTD.

7. *Key*: it is associated with a primitive of type 3 or 4; it corresponds to a primitive expressing the uniqueness for a (one or more) component the primitive associated with it. An well known exemplar of this metaprimitive is the key of the relational model.

8. *Foreign Key*: it is associated with primitives of type 3 or 4; it corresponds to primitives expressing referential constraints between a (one or more) component the primitives associated with it and the key of another primitive. For example, the *Keyref* tag is used in XML Schema to define a primitive of this type.

Note that metaprimitives can be combined. We will call a specific combination of metaprimitives a *pattern*.

3.2. Model definition

Once the metamodel has been fixed, a model can be specified by defining: (1) the primitives offered by the model in terms of the metaprimitive of the metamodel and the way in which these primitives can be combined, that is, the allowed patterns, and (2) the syntax used by the DDL of the model to specify the application of its primitives in the construction of schemes. We also assume that, in defining a model, a labeling functions can be freely used to name primitives and their components. As we have said, the extensibility of the approach needs always to be guaranteed: if a primitive of a model does not correspond to any of the metaprimitive at disposal, then we assume it is possible to include a new component in the metamodel.

Clearly, all of this can be specified using a specific syntax. However, in this preliminary report, we just show how this can be done by means of a number of informal and simplified examples and with reference to the metaprimitives above mentioned.

- *Relational model.* This model can be defined by means of a few primitives: the *domain*, which is a base type, the *tuple*, which corresponds to an unordered sequence of values taken from a domain, and the *relation*, which corresponds to the association of the cardinality primitive (with $Min = 0$ and $Max = N$) with the tuple primitive.

- *OEM.* This model can also be defined by means of a few primitives: it includes the *atomic type*, which is a base type of the metamodel, the *atomic object*, which corresponds to the application of the object type to the atomic type, and the *complex object*, which corresponds to the application of the object type to an unordered sequence of (atomic or complex) objects.

- *DTD.* A simplified version of DTD can be defined by means of the following primitives: two base type (#PCDATA for elements and CDATA for attributes), three forms of cardinality (*, ? and + with the obvious meaning) called *occurrences*, the *ID* and the *IDREF(s)*, which are used to define keys and foreign keys respectively, the *attribute*, which is just a labeled value of type ID, IDREF or CDATA, and the *element*. The element can have associated with it an unordered sequence of attributes and can be either *simple* or *composite*. A simple element is a labeled value of type #PCDATA. A composite element is either a choice (with possibly an occurrence associated with it) or an ordered sequence of (sub) elements (each of which with a possibly an occurrence associated with it).

- *XML Schema.* This is a quite complex formalism. A simplified version can be defined by means of the following primitives. There are 45 (!) base types (e.g., string, integer, float and so on) called *simple types*. There is a *cardinality* primitive that allows the specification of any value for *Max* and *Min* (here called *maxOccurs* and *minOccurs* respectively). An *attribute* is a labeled value from a base type. An *element* is a labeled (simple or complex) type. A *complex type* has an optional unordered sequence of attributes and a "group" of *elements* with possibly a cardinality associated with them. A group can be a *sequence* (ordered), a *all* (unordered sequence) or a *choice*. Finally, keys and foreign keys can be defined over both attributes and elements using the primitives *key* and *Keyref* respectively.

- *XDR*. It has more or less the same primitives of XML Schema but uses a different syntax. For instance, the ordered sequence primitive is called *seq* and the choice is called *one* in XDR.

4. Translation between models

4.1. A supermodel approach

As we have said in the Introduction, the final goal of our research is a system that, given two models defined by means of the metamodel, is able to translate schemes and possibly instances from one model to another.

As a preliminary tool, a subsumption relationship $\preceq$ between models is defined. We will not present here the technical details. Intuitively, it is a partial order relationship is based on: (1) set containment of patterns allowed in the models, and (2) a predefined partial order on patterns. The latter allows us to specify, for instance, that a pattern involving a cardinality primitive of a DTD (where only ? (0,1), $+$ (1,N) and $*$ (0,N) are allowed) is subsumed by the same pattern that involves the cardinality primitive of XML Schema (where *minOccurs* and *maxOccurs* can assume in principle any value). It follows that a scheme which is an instance of a model M, is a scheme of any model that subsumes M.

On the basis of this relationship, we then introduce a notion of *supermodel*, which is used as a reference in the translations. Intuitively, a supermodel is a model (that is, an instance of the metamodel) which includes a representative of each pattern used in the current models, in its most general version. Hence, by definition, the supermodel subsumes each other model according to the subsumption relationship.

The translations are then based on the following principles.

1. A translation between models is based on the translation of the basic primitives of involved models. More precisely, for each primitive P (or variation thereof) for a metaprimitive $\hat{P}$ of the source model M for which there is not a corresponding primitive P' for $\hat{P}$ in target model M', we have to specify how it is represented by means of the primitive available in M'.

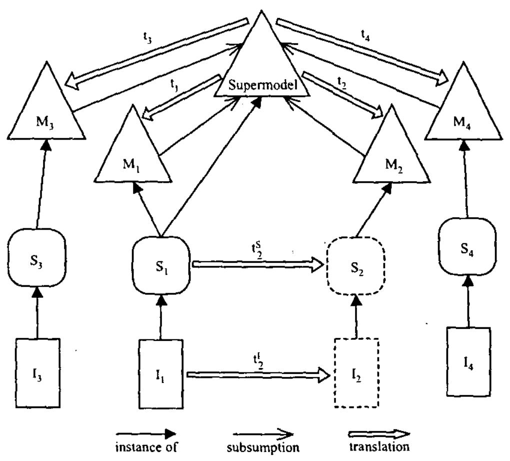

Figure 2. Relationships and translations between models

2. The translation process can be seen as composed of two steps: *(i)* from the source model to the supermodel, and *(ii)* from the supermodel to the target model. Step *(i)* is trivial, since, because of the subsumption relationship, every scheme of any model is a scheme of the supermodel. Step *(ii)* can be performed by means of suitable translations of the elements in the source scheme whose metatype has no counterpart in the target model into elements of allowed types. Note that, with this approach, it suffices to define translations from the supermodel to every other model in order to implement all the possible translations between models. It follows that the number of required translations is linear in terms of the number of models, instead of quadratic, as it would be if the process had to be specified for each pair of models.

3. As the number of constructs is limited, it is possible to predefine a number of basic translations, which can be composed to build more complex translations. It is interesting to note that in this way we can satisfactorily refer to the notion of "transformational equivalence" [4], under which two schemes are equivalent if one can be obtained from the other by applying a set of atomic transformations, which preserve equivalence *by definition*.

The overall picture is illustrated graphically in Figure 2. A translation t from the supermodel to every other model need to be specified. These translations are defined on individual primitives of the involved models and have two main sub-components: a translation t^S that operates on schemes and a corresponding translation t^I that operates on instances according to the transformations specified by t^S at scheme level. Thus, to translate the instance I_1 of the schema S_1 into the model M_2 we make use of the translation t_2 (note that S_1 is indeed a scheme of the supermodel). We then obtain the scheme S_2 using t_2^S and the instance I_2 using t_2^I.

4.2. An example of translation

In this section we present an example of translation. Specifically, we discuss the translation of the XML schema reported in Figure 3 into a DTD and into a relational schema.

In the context of the metamodel, this XML schema can be profitably represented by the graph reported in Figure 4 (where dot lines indicate attributes). In this figure we have also reported the notation used for the metaprimitives of the metamodel.

In our framework, the translation of this schema into a DTD can be obtained by applying the following transformations on base primitives:

1. All base types are translated into strings;
2. Unordered sequences are translated into ordered sequences;
3. Cardinalities are simplified, to get to the coarser alternatives allowed in DTDs;
4. Keys are translated into ID attributes;
5. Foreign keys are translated into IDREF attributes.

The representation of the schema we obtain by applying this sequence of operations is reported in Figure 5. Note that we can use here the same notation as above, since it actually refers to the metamodel. The corresponding DTD is shown in Figure 6.

At the instance level this translation just requires to transform elements involving foreign keys into attributes, since in DTDs only attributes can be of type IDREF.

Similarly, the translation of the same schema into a relational schema can be obtained by applying the following transformations on base primitives:

1. Elimination of choices;
2. Translation of ordered sequences into unordered sequences;
3. Unnest of nested unordered sequences;
4. Simplification of cardinalities;
5. Insertion of keys for sequences that do not have them;
6. Merge of attributes and elements.

```
<xsd:schema>
    <xsd:element name="MyCompany">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Department"
                             type="DepartmentType"
                             maxOccurs="unbounded"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:key name="EmpKey">
            <xsd:selector xpath="Department/Employee"/>
            <xsd:field xpath="@EmpID"/>
        </xsd:key>
        <xsd:keyref name="ManFK"
                    refer="Empkey">
            <xsd:selector xpath="Department"/>
            <xsd:field xpath="Manager"/>
        </xsd:keyref>
    </xsd:element>
    <xsd:complexType name="DepartmentType">
        <xsd:sequence>
            <xsd:element name="DepName"
                         type="xsd:string"/>
            <xsd:element name="URL"
                         type="xsd:string"/>
            <xsd:element name="Manager"
                         type="xsd:string"/>
            <xsd:choice minOccurs="0"
                        maxOccurs="15">
                <xsd:element name="Employee"
                             type="EmployeeType"/>
                <xsd:element name="Freelance"
                             type="FreelanceType"/>
            </xsd:choice>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="EmployeeType">
        <xsd:sequence>
            <xsd:element name="Name"
                         type="xsd:string"/>
            <xsd:element name="Title"
                         type="xsd:string"
                         minOccurs="0"/>
            <xsd:element name="PhoneExt"
                         type="xsd:integer"/>
        </xsd:sequence>
        <xsd:attribute name="EmpID"
                       type="xsd:string"
                       use="required"/>
        <xsd:attribute name="IsManager"
                       type="xsd:boolean"
                       use="optional"/>
    </xsd:complexType>
    <xsd:complexType name="FreelanceType">
        <xsd:all>
            <xsd:element name="Name"
                         type="xsd:string"/>
            <xsd:element name="Address"
                         type="xsd:string"/>
            <xsd:element name="EMail"
                         type="xsd:string"
                         minOccurs="0"/>
        </xsd:all>
    </xsd:complexType>
</xsd:schema>
```

Figure 3. An XML Schema

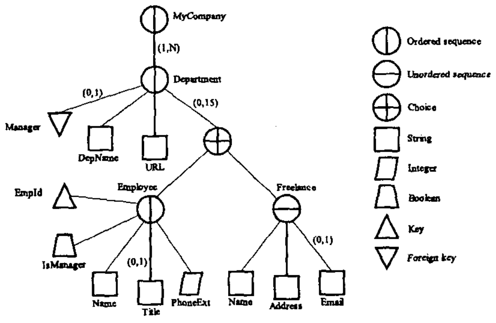

Figure 4. A graphical representation of the XML Schema in Figure 3

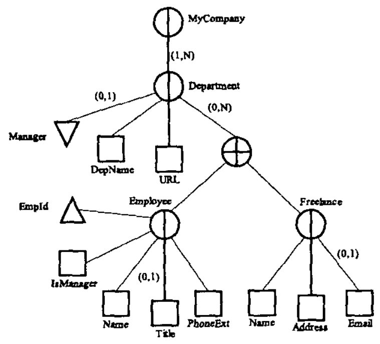

Figure 5. Translation of the schema in Figure 4 into a DTD

```
<!ELEMENT MyCompany (Department+)>
<!ELEMENT Department (
    DepName, URL, (Employee | Freelance)+
)>
<!ATTLIST Department
    Manager IDREF #IMPLIED>
<!ELEMENT DepName (#PCDATA)>
<!ELEMENT URL (#PCDATA)>
<!ELEMENT Employee (
    Name, Title?, PhoneExt
)>
<!ATTLIST Employee
    EmpID ID #REQUIRED
    IsManager CDATA>
<!ELEMENT Freelance (
    Name, Address, EMail?
)>
<!ELEMENT Name (#PCDATA)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT PhoneExt (#PCDATA)>
<!ELEMENT Address (#PCDATA)>
<!ELEMENT EMail (#PCDATA)>
```

Figure 6. A DTD obtained as the result of a translation

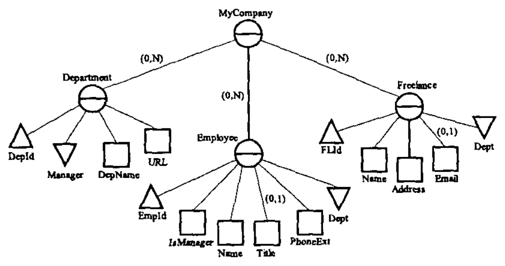

Figure 7. Translation of the schema in Figure 4 into a relational schema

The representation of the schema we obtain by applying this sequence of operations is reported in Figure 7. With traditional notation, the relational scheme would be as follows:

- Department(DeptID, Manager*, Name, URL)
- Employee(EmplID, IsManager, Name, Title*, PhoneExt, Dept)
- Freelance(FlID, Name, Address, Email*, Dept)

Clearly, this is only a possible way to do that. At the instance level this translation is more involved and requires the transformation of an XML document into a set of relations according to the translation at scheme level above described.

4.3. Properties of translations

Let us consider a number of interesting properties that can be studied for the translations we have defined.

The first requirement for a translation is to be *valid*, that is, the output scheme (instance) should be a valid scheme (instance) for the target model. Then, a notion of *equivalence* should be preserved. Many definition of equivalence have been proposed in the literature. In essentially all cases, a necessary (but not sufficient) condition for equivalence of schemes is that there is a one-to-one correspondence between their respective sets of instances.

Then, there are pairs of models for which it is possible to find an equivalent target scheme (under every definition) for each source scheme: for example, this holds if XDR and XML Schema, with suitable technical details, are considered. However, this is not true in general: it is clear that equivalence cannot be guaranteed when the source model allows finer representation of features than the target model. For example, if the source schema is an XML Schema that has associated cardinalities like $(1, 10)$ or $(5, N)$, and the target is a DTD (where the only allowed cardinalities are

$(0, 1)$, $(1, N)$, and $(0, N)$) then there cannot be an equivalent target scheme. In these cases we say that there is a *loss* of (meta) information in the translation process.

There are also cases where it is possible to find an equivalent target scheme for each source scheme, but the intuitive information represented by the target scheme is less rich than that of the source scheme. This is especially apparent when there is an "implementation" process: it is known that, given a source scheme in XML Schema, with rather general features, it is possible to find a relational scheme that is equivalent to the source scheme, but does not represent the same semantics as the source scheme. In these cases, it is usually impossible to invert the translation. Here we say that the information, though preserved, is *degraded*. Making a parallel with thermodynamics, we can say that in this case "energy" is preserved whereas "entropy" increases, without any chance of being decreased in the future.

Both in the case of loss of information and in the case of degradation, there are various possible translations of schemes, with incomparable properties (that is, each translation is better on some grounds and worse on others). Even when equivalent translations exist, if the models are *redundant* (that is, allow different primitives for the representation of individual features), there are different, reasonable translations.

5. Conclusion and future work

In this paper, we have illustrated some preliminary results of our research whose goal is the definition of an environment for the management and the exchange of (semi) structured Web data, described according to a variety of formats and models. The basic approach to this problem consists in the definition of a "metamodel" that generalizes the basic modeling primitives adopted by the various models. To this end, we started from considering the core of various schema definition languages for XML and a number of data models used for the description and the permanent storage of XML data. We have shown that the primitives adopted by all of them can be classified into a rather limited set of basic types. We have then proposed a metamodel having a metaprimitive for each of these class. In this way, the translation of schemes and of the corresponding instances, can be specified on the basis of translations of the involved types of primitives. This is effectively carried out by means of a number of predefined operations that implement standard translations between basic primitives. A number of issues need to be further investigated. First of all, several notions need to be better formalized and studied in a more systematic way. There are several important aspects about XML and schema languages for XML that have been disregarded and it has to be understood how they fit in this framework. Among them, we cite the namespaces, the content models and several form of constraints which can be defined in the context of XML.

References

[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 1999.

[2] P. Atzeni and R. Torlone. Management of Multiple Models in an Extensible Database Design Tool. In *Fifth International Conference on Extending Database Technology (EDBT '96), Lecture Notes in Computer Science 1057*, Springer–Verlag, pag. 79–95, 1996.

[3] P. Atzeni and R. Torlone. MDM: A Multiple-Data-Model Tool for the Management of Heterogeneous Database Schemes. In *ACM SIGMOD International Conference on Management of Data, Tucson*, pages 528–531, 1997.

[4] C. Batini, M. Lenzerini, and S.B. Navathe. A comparative analysis of methodologies for database scheme integration. *ACM Computing Surveys*, 18(4):323–364, 1986.

[5] P. A. Bernstein and T. Bergstraesser. Meta-data support for data transformations using Microsoft Repository. *IEEE Data Engineering Bulletin*, 22(1):9–14, March 1999.

[6] P. A. Bernstein, A. Y. Levy, and R. A. Pottinger. A Vision for Management of Complex Models. *SIGMOD Record*, 29(4):55–63, December 2000.

[7] P. A. Bernstein and E. Rahm. Data Warehouse Scenarios for Model Mangement. In *19th Int. Conference on Conceptual Modeling (ER2000), Salt Lake City, Lecture Notes in Computer Science 1920*, Springer-Verlag, pages 1–15, 2000.

[8] P. V. Biron and A. Malhotra (ed.). XML Schema Part 2: Datatypes. W3C Document, April 2000. http://www.w3.org/.

[9] S. Bowers and L. Delcambre. Representing and Transforming Model-Based Information. In *ECDL Work. on Semantic Web, Lisbon*, 2000. http://www.ics.forth.gr/proj/isst/SemWeb/.

[10] T. Bray, J.Paoli, C. M. Sperberg-McQueen (ed.). Extensible Markup Language (XML) 1.0. W3C Document, February 1998. http://www.w3.org/.

[11] T. Bray, C. Frankston, A. Malhotra (ed.). Document Content Description for XML. W3C Document, July 1998. http://www.w3.org/.

[12] S. Cluet, C. Delobel, J. Siméon, and K. Smaga. Your Mediators Need Data Conversion!. In *Proc. of ACM SIGMOD International Conference on Management of Data, Seattle, USA*, pages 177–188, 1998.

[13] A. Davidson, M. Fuchs, M. Hedin, et al. Schema for Object-Oriented XML 2.0. W3C Document, July 1999. http://www.w3.org/.

[14] L. Delcambre and D. Maier. Models for Superimposed Information. In *18th Int. Conference on Conceptual Modeling (ER99), Paris, Lecture Notes in Computer Science 1727*, Springer-Verlag, pages 264–280, 1999.

[15] D. Florescu and D. Kossmann. Storing and Querying XML Data using an RDMBS. *IEEE Data Engineering Bulletin*, 22(3): 27–34, 1999.

[16] C. Frankston and H. S. Thompson. XML-Data Reduced. Internet Document, July 1998. http://www.ltg.ed.ac.uk/~ht/XMLData-Reduced.htm

[17] R.B. Hull and R. King. Semantic database modelling: survey, applications and research issues. *ACM Computing Surveys*, 19(3):201–260, September 1987.

[18] IBM Corporation. XDRtoXSD. Internet Document, 2000. http://alphaworks.ibm.com/tech/xdrtoxsd.

[19] R. Jelliffe. The Schematron, an XML Structure Validation Language using Patterns in Trees. Internet Document, May 2000. http://www.ascc.net/xml/resource/schematron/.

[20] N. Klarlund, A. Moller, M. I. Schwatzbach. DSD: A Schema Language for XML. In *Proc. 3rd ACM Workshop on Formal Methods in Software Practice*, 2000.

[21] D. Lee and W. W. Chu. Comparative Analysis of Six XML Schema Languages. *SIGMOD Record*, 29(3):76–87, 2000.

[22] D. Maier and L. Delcambre. Superimposed Information for the Internet. In *ACM SIGMOD Workshop on The Web and Databases WebDB99, Philadelphia*, pages 1–9, 1999.

[23] P. McBrien and A. Poulovassilis. A Uniform Approach to Inter-model Transformations. In *11th Int. Conference on Advanced Information Systems Engineering (CAiSE99), Heidelberg, Lecture Notes in Computer Science 1626*, Springer-Verlag, pages 333–348, 1999.

[24] R.J. Miller, Y.E. Ioannidis, and R. Ramakrishnan. The use of information capacity in schema integration and translation. In *Eighteenth International Conf. on Very Large Data Bases, Dublin*, 1993.

[25] T. Milo and S. Zohar. Using Schema Matching to Simplify Heterogeneous Data Translation. In *Int. Conference on Very Large Data Bases (VLDB), New York*, 1998.

[26] M. Murata. RELAX (REgural LAnguage description for XML). Internet document, 2000. http://www.xml.gr.jp/relax/.

[27] Y. Papakonstantinou, H. Garcia-Molina, and J. Widom. Object Exchange Across Heterogeneous Information Sources. In *Proc. of the Eleventh International Conference on Data Engineering, Taipei, Taiwan*, pages 251–260, 1995.

[28] H. S. Thompson, D. Beech, M. Maloney, N. Mendelsohn (ed.). XML Schema Part 1: Structures. W3C Document, April 2000. http://www.w3.org/

[29] XML Spy. http://www.xmlspy.com/