Combining DAML+OIL, XSLT and probabilistic logics for uncertain schema mappings in MIND

Henrik Nottelmann and Norbert Fuhr

Institute of Informatics and Interactive Systems, University of Duisburg-Essen,
47048 Duisburg, Germany, {nottelmann,fuhr}@uni-duisburg.de

Abstract. When distributed, heterogeneous digital libraries have to be integrated, one of the crucial tasks is to map between different schemas. As schemas may have different granularities, and as schema attributes do not always match precisely, a general-purpose schema mapping approach requires support for uncertain mappings. In this paper we present one of the very few approaches for defining and using uncertain schema mappings. We combine different technologies like DAML+OIL, probabilistic Datalog (since DAML+OIL—as similar ontology languages—lacks rules) and XSLT for actually transforming queries and documents. This declarative approach is fully implemented in the project MIND (which develops methods for retrieval in networked multimedia digital libraries). However, as DAML+OIL lacks some important features, the proposed approach is only a stepping stone for an integrated solution.

1 Introduction

Federated digital libraries (DLs) integrate a large number of legacy DLs; so they give users the impression of one coherent, homogeneous library. MIND [15] is such a system for heterogeneous and non-co-operative multimedia libraries. Heterogeneity appears in different forms: In MIND, differences in query languages, communication protocols and document models are solved by DL-specific wrappers.

Libraries also differ in the document structure (schemas). Users cannot deal efficiently with this semantic heterogeneity and therefore should only know one system-wide or personalised standard schema. In MIND, the standard schema is defined ontologically and independent from the sources. Then, queries are transformed from the standard schema into the DL schemas, and documents vice versa.

In contrast to most of the approaches available so far, MIND explicitly supports uncertain schema mappings. Schemas may have different granularity, and schema attributes do not always match precisely (e.g. authors and editors vs. the more general attribute creators). So creators cannot be mapped onto authors precisely but only with a specific probability. Systems with purely deterministic mappings fail in such settings.

In MIND, we model MIND documents and queries in DAML+OIL (the forthcoming standard ontology language) so that we can extend our approach to nested structures fairly easy. As DAML+OIL lacks rules, we specify schema mappings in probabilistic Datalog. The rules are then converted into XSLT stylesheets for transforming the DAML+OIL models. XSLT implementations are available for all major programming languages, so our approach can be used easily in other projects. DAML+OIL schema definitions, pDatalog rules and the resulting XSLT stylesheets can be stored in textual

files, so it is very easy to update the mappings when a schema is changed (without recompiling code). We are aware that there are many open questions, so this work should be considered a stepping stone.

The rest of this paper is structured as follows. The next section gives a survey of other approaches for handling heterogeneous schemas. Section 3 introduces the MIND document model. In section 4 we model MIND documents and queries in DAML+OIL. Section 5 describes how schema mappings are expressed in probabilistic logics. Section 6 summarises the major ideas presented in this paper and gives an outlook on future research.

2 Related work

Mappings between heterogeneous schemas have been studied for quite a while, but only one of the existing approaches allows for uncertain mappings.

In the field of federated databases, two approaches are distinguished: In "local as view" (LaV), the source schemas are defined as views (mappings) over a fixed global schema. This makes it easy to add a new source, but query transformation has exponential time complexity. In contrast, the global schema is defined as a view over local schemas in the "global as view" (GaV) approach. Here, query transformation can be reduced to rule unfolding, but the global view has to be modified whenever a new source is added.

The BGLaV approach [20] combines the advantages of both worlds. The global schema is specified ontologically and independent from the sources, the source schema models the documents returned by the source, and mappings are defined by relational algebra expressions. This approach has polynomial time complexity for query transformation (like GaV) while adding new sources is fairly simple (like LaV), and is used in an extended form in MIND.

A framework for dealing with heterogeneous OSM schemas is presented in [1]. OSM models contain objects, their relationships and a predicate calculus for expressing constraints. As in BGLaV, the global schema is defined ontologically and independent from the source schemas. Interaction with an administrator is assumed (however not required) for setting up deterministic mappings between objects (and relations, respectively). These mappings can model specialisation/generalisation relationships, and string processing operators are provided.

The GaV system Demetrios [6] follows a different approach with SQL as query language for both the user and the sources. The SQL extension FRAQL is used for defining mappings from the source relations onto global relations. FRAQL allows e.g. for accessing metadata, restructuring tables and converting values.

TSIMMIS [3] is one of the early systems integrating heterogeneous digital libraries. Schema mappings are defined in a textual format with actions which are executed when a corresponding template matches a query.

MARIAN [10] shares some common aspects with our approach. Attributes can be mapped onto others with simple rules, specified with the declarative 5S language. Uncertain mappings are possible by weighting the mapping rules. In contrast to the wrapper based system MIND, MARIAN uses a harvesting approach: It periodically down-

loads all documents from an information source and stores them in a local index. Thus, MARIAN has to transform documents into a standard schema only once, and does not require query transformations at all.

With the growing popularity of XML, mappings between different DTDs are also investigated. Due to the deterministic nature of XML, uncertainty is not supported by any of these approaches.

A tree-grammar-based approach for inducing integrated views (XML-QL templates which can be used for stating user queries) for XML data with heterogeneous DTDs is presented in [11]. Type trees derived from the source DTDs are converted into a tree automaton. States belonging to similar types are merged to obtain a minimised integrated view.

LSD [5] uses a machine learning approach for finding matching elements. Different classifiers (e.g. naive Bayes, kNN, county name recogniser) are trained on pairs of XML documents (one corresponding to the global schema, the other one corresponding to one of the local schemas), where an administrator has to identify matching elements. After this training phase, the classifiers are combined and applied to different source schemas.

Cupid [13] is a generic schema matching algorithm for detecting matching schema elements (not restricted to DTDs or XML Schemas). Cupid discovers matchings based on names (by using linguistic techniques, e.g. a thesaurus), data types, constraints and schema structure.

None of the approaches described in this section aside from MARIAN (whose harvesting approach is incompatible with MIND) addresses the problem of uncertain mappings: Either a mapping between attributes of different schemas is deterministic, or it is ignored. However, since information retrieval has to deal with uncertainty and vagueness anyway, we think that uncertain mappings should be supported for retrieval in digital libraries, and thus was integrated within MIND.

3 MIND document model

MIND adopts the document model presented in [8] with only slight modifications.

Like in database systems, data types with comparison operators are explicitly modelled. However, vagueness of query formulations is one of the key concepts of Information Retrieval. Thus, it is crucial that comparison operators have a probabilistic interpretation (as proposed in [7]). Vagueness is required e.g. when a user is uncertain about the exact publication year of a document or the spelling of an author name. These comparison operators are called “vague predicates”. For a specific attribute value the vague predicate yields an estimate of the probability that the condition is fulfilled from the user’s point of view — instead of a Boolean value as in DB systems.

Definition 1 (Data types). *A data type D is a pair $(dom(D), pred(D))$, where $dom(D)$ is the domain (all possible values) and $pred(D) = \{p_1, \ldots, p_n\}$ is the set of (vague) predicates. Each predicate is a function p_i:$dom(D) \times dom'(D, p_i) \rightarrow [0,1]$, and $dom'(D, p_i)$ is the domain of all possible comparison values (values in a query) for that predicate p_i. The set of all data types is denoted by $\mathcal{D}$.*

In most cases we have $dom'(D, p_i) = dom(D)$. On the other hand, the predicate *isIn* of the data type *Year* has a pair of year numbers (a time period) as comparison value:

$dom'(Year, isIn) = dom(Year) \times dom(Year)$. In addition, some of the predicates may be Boolean (like =), i.e. the mapping is restricted to the set $\{0,1\}$.

MIND employes a linear document model. Different parts of documents are specified by named and typed attributes:

Definition 2 (Schema). *A schema S is a set of schema attributes $\{A_1,\ldots,A_n\}$. A schema attribute A is a pair $(dt(A), pred(A))$ of a data type $dt(A) \in \mathcal{D}$ and a subset $pred(A) \subseteq pred(dt(A))$ of supported predicates. For brevity, we write $dom(A) = dom(dt(A))$.*

Example 1. In the remainder of this paper, we will use the schemas

$$S_1 := \{au = (Name, \{equals\}), da = (DateISO8601, \{=\}), ti = (Text, \{contains\})\},$$
$$S_2 := \{author = (Name, \{equals, soundslike\}), editor = (Name, \{equals, soundslike\}),$$
$$date = (DateEN, \{same, vague-same\}), title = (Text, \{contains\})\}.$$

Here, valid values of *DateISO8601* are "2003-03-10" or "2003-12-24"; values of *DateEN* are "03/10/03" or "12/24/03".

A document is an instance of a specific schema, assigning values to attributes:

Definition 3 (Document). *A document d is a pair (S, att) of a schema $S = schema(d)$ and a set of document attributes $att = \{a_1,\ldots,a_n\}$. A document attribute a is a triple (w, A, v) of a schema attribute $A \in S$, a value $v \in dom(A)$ and a probabilistic weight $w \in [0,1]$ specifying the probability that v is a correct value of A.*

In most cases, the weight of a document attribute equals 1 (which can then be left out). It can be less than 1 for expressing uncertainty about the correct value (e.g. if the year of creation for an artefact is only approximately known).

A schema attribute can be used by more than one document attribute (e.g. for multiple authors), and schema attributes can be left out completely.

Example 2. These are the representations of the same document according to the two schemas defined above:

$$d_1 := (S_1, \{(au, \text{"N. Fuhr"}), (au, \text{"J. Callan"}), (au, \text{"B. Croft"}), (au, \text{"J. Lafferty"}),$$
$$(da, \text{"2001-01-01"}), (ti, \text{"Language Models and ..."})\}),$$
$$d_2 := (S_2, \{(author, \text{"N. Fuhr"}), (editor, \text{"J. Callan"}), (editor, \text{"B. Croft"}),$$
$$(editor, \text{"J. Lafferty"}), (date, \text{"01/01/01"}), (title, \text{"Language Models and ..."})\}).$$

These examples show that S_2 is more specific than S_1 as it distinguishes between authors and editors but S_1 puts both in the attribute *au*.

Definition 4 (Query). *A query q is a pair $(S, cond)$ of a schema $S = schema(q)$ and a set of query conditions $cond = \{c_1,\ldots,c_n\}$. A query condition c is a tuple (w, A, p, v) of a probabilistic weight $w \in [0,1]$ specifying the importance of this condition, a schema attribute $A \in S$, a predicate $p \in pred(A)$ and a comparison value $v \in dom'(S, p)$.*

Example 3. These queries corresponding to the two schemas should return documents d_1 and d_2, respectively:

$$q_1 := (S_1, \{(0.8, au, equals, \text{"Fuhr"}), (0.2, da, =, \text{"2001"})\}),$$
$$q_2 := (S_2, \{(0.8, author, soundslike, \text{"Fuhr"}), (0.2, date, vague-same, \text{"2001"})\}).$$

4 Modelling documents and queries in DAML+OIL

In the previous section we described the linear, flat document model used within the MIND project. We plan, however, to extend our model to nested structures. The most promising candidate for expressing nested schemas and documents is the RDF Schema extension DAML+OIL [4] which has the power to become the standard ontology language

RDF [12] is developed in the context of the Semantic Web as a specification language for objects and their properties via statements of the form "subject predicate object" (also called triples). The subject of the statement is an RDF object, the predicate is a property, and the object of a statement is another RDF object or a literal (a value, e.g. a string or a number). The vocabulary (object classes, properties) can be defined with the schema specification language RDF Schema (RDFS for short) [2]. DAML+OIL enriches RDFS with more advanced primitives: E.g. the range of a property can be specified w.r.t. the domain (in RDFS, it can only be specified globally). As there will never be one single standard ontology, schema mapping remains a crucial issue in the DAML+OIL world.

Thus, we decided to map the MIND document model onto DAML+OIL, i.e. that MIND schemas[1]) are modelled as DAML+OIL schemas. However, the DAML+OIL language has two restrictions we have to face:

1. DAML+OIL employs XML Schema for modelling data types. XML Schema is designed to define documents, not queries. The major difference is that the former simply store values (unary data type predicates) whereas queries compare two values (binary data type predicates). In contrast to approaches like $\mathcal{SHOQ}(\mathbf{D_n})$ [16], XML Schema (and thus, DAML+OIL), only support unary data type predicates.
2. DAML+OIL does not support the concept of uncertainty. However, in our document model document attribute values and query conditions can be weighted probabilistically.

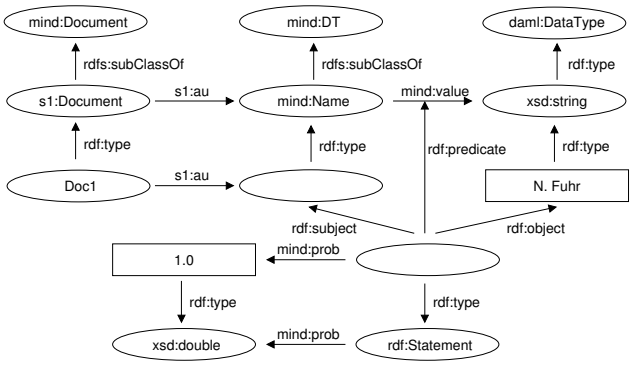
These problems can be solved in two different ways: We can extend the language by allowing n-ary data type predicates (i.e., allowing properties for `daml:DataType` subclasses and their instances), and by attaching a probabilistic weight to every statement. We disregarded this alternative, as it contradicts the efforts of establishing a standard ontology language, and existing DAML+OIL and RDF tools would not work any more.

So we followed solutions which remain within the RDF world: As DAML+OIL data types only have a restricted expressiveness, we model MIND data types as classes in DAML+OIL (subclasses of `mind:DT`) instead of DAML+OIL data types directly. Every instance of a MIND data type class has a property `mind:value`; the range is a DAML+OIL data type. Search predicates are modelled by a property `mind:pred` with the domain `mind:DT` and the range `mind:Predicate`.

Furthermore, reification (statements about statements) is used for supporting uncertainty. Statements can have a property `mind:prob` which specifies the probability that this statement is correct. Reification is supported by RDFS, but DAML+OIL tools cannot give any semantics to such triples. However, we will later see that we don't use

[1] With "schema", we mean MIND schemas in this paper, not DAML+OIL schemas.

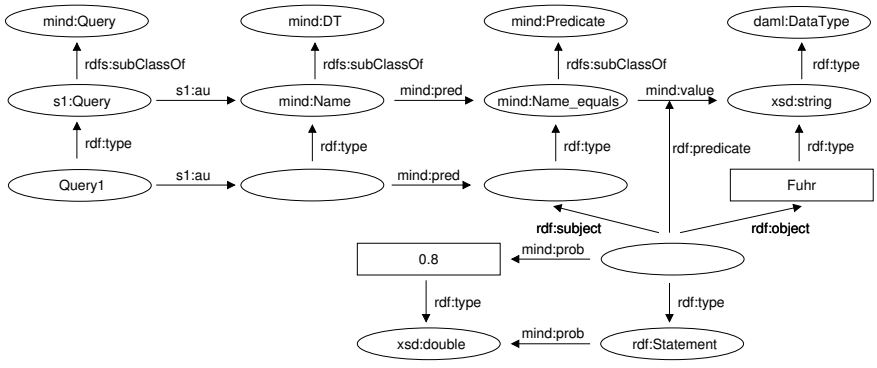
Fig. 1. Documents in DAML+OIL



DAML+OIL tools directly, so we think that this extension does not introduce additional problems.

Examples for documents and queries are shown in figures 1 and 2. The first "level" introduces some common concepts used in the MIND document model, the second row defines the MIND schema S_1 and a concrete data type. The lower part models one specific document. The document attribute *au* is modelled by anonymous instances of `mind:Name` and `rdf:Statement` (reification). Queries are modelled in a similar way.

Fig. 2. Queries in DAML+OIL



5 Heterogeneous schemas

As pointed out above, in heterogeneous environments queries have to be transformed from the standard schema to the DL schema, and documents from the DL schema to

the standard schema. In MIND, the standard schema is defined ontologically and independently from the libraries, and the DL schemas represent the structure of library documents (similar to BGLaV, but in contrast to LaV or GaV).

For these transformations, rules (stating the relationships between attributes of the two different schemas) are required. DAML+OIL does not support rules (besides properties like `rdfs:subClassOf` which could be viewed as rules). DAML-L is intended to add rules to DAML+OIL, but this language is not yet completed.

In MIND we decided to map parts of DAML+OIL models onto probabilistic Datalog. This probabilistic Horn clause logic provides a powerful and flexible rule mechanism. The rules are then automatically converted into XSLT stylesheets which operate on XML serialisations of DAML+OIL models (documents and queries).

5.1 Probabilistic Datalog

Datalog [18] is a variant of predicate logic based on function-free Horn clauses. Negation is allowed, but its use is limited. Rules have the form

```
father(X,Y) :- parent(X,Y) & sex(X,male).
```

where `father(X,Y)` is the head literal of the rule (with predicate `father` and variables `X` and `Y`). The right part forms the body, a conjunction of literals ("subgoals") `parent(X,Y)` and `sex(X,male)`. Negated literals start with an exclamation mark. Variables like `X` start with an uppercase character, constants like `male` with a lowercase character. Thus the rule states that `X` is the father of `Y` if it is a male parent.

Facts are rules with empty body and only constants in the head literal:

```
sex(peter,male). sex(mary,female). 0.9 parent(peter,mary).
```

In probabilistic Datalog (pDatalog for short) [9], every fact can have a probabilistic weight (if it is left out, the weight equals one). In the same way, rules also can be weighted. If more than one rule is formulated for the same head predicate, we have to know $Pr(p|\neg r_1 \wedge \neg r_2)$, $Pr(p|r_1 \wedge \neg r_2)$, $Pr(p|\neg r_1 \wedge r_2)$ and $Pr(p|r_1 \wedge r_2)$. This is equivalent to the situation in probabilistic inference networks where a link matrix with the same conditional probabilities has to be given.

Example 4. The following rules state that two documents are semantically related iff they have the same author or are linked together:

```
sameauthor(D1,D2) :- author(D1,A) & author(D2,A).
0.5 related(D1,D2) :- link(D1,D2) & !link(D1,D2).
0.2 related(D1,D2) :- !link(D1,D2) & link(D1,D2).
0.7 related(D1,D2) :- link(D1,D2) & link(D1,D2).
```

This rule set with 2^{n-1} rules for n original rules cannot be evaluated in general, but the 2^{n-1} probabilities and the n original rules can be transformed into a new set of n certain rules with 2^{n-1} additional uncertain weights [19, 14].

5.2 Mapping DAML+OIL onto pDatalog

For our purpose, a simple, straight-forward mapping of parts of a DAML+OIL model onto probabilistic Datalog is sufficient:

Definition 5 (Mapping DAML+OIL onto pDatalog). *Every DAML+OIL class is mapped onto a unary pDatalog predicate with the same name (where the namespace prefix and the local name are concatenated with a "#"). Instances of a DAML+OIL class are facts referring to that predicate.*

Every property is mapped onto a binary pDatalog predicate with the same name. Statements "s p o" with subject s, predicate p (the property) and object o are mapped onto facts `p(s,o)`*. If the statement is weighted, this is regarded as the fact's probabilistic weight.*

As we deal with text (including brackets, punctuations and whitespace), we allow strings in quotation marks as constants as well.

The statements (triples) depicted in figures 1 and 2 are converted into the following pDatalog facts:

```
s1#au(doc1,x1). mind#name(x1). mind#value(x1,"N. Fuhr").

s1#au(query1,xx1). mind#name(xx1). mind#pred(xx1,xx2).
mind#name_equals(xx2). 0.8 mind#value(xx2,"Fuhr").
```

For every document attribute we have three facts with an auxiliary constant (`x1`), for every query condition even five facts with two auxiliary constants (`xx1` and `xx2`). This is due to the problem that DAML+OIL does not support binary data type predicates in DAML+OIL for modelling query. However, pDatalog supports n-ary predicates, so we can define shortcuts:

```
s1#au(doc1,"N. Fuhr").

0.8 s1#au_equals(query1,"Fuhr").
```

The first fact can be obtained by

```
s1#au(D,V) :- s1#au(D,X) & mind#name(X) & mind#value(X,V).
```

This model can be used for retrieval in pDatalog. We need one rule for every predicate which combines corresponding query conditions, document attribute values and a pDatalog "implementation" of the search predicate:

```
s1#result(Q,D) :- s1#au_equals(Q,V1) & s1#au(D,V2) &
                  mind-dt#name_equals(V2,V1).
s1#result(Q,D) :- s1#da_eq(Q,V1) & s1#da(D,V2) &
                  =(V2,V1).
s1#result(Q,D) :- s1#ti_contains(Q,V1) & s1#ti(D,V2) &
                  mind-dt#text_contains(V2,V1).

?- s1#result(query1,D).
```

The last line retrieves the result of query `query1`; with the document base used so far, this is document `doc1` (with probability $0.8 \cdot \mathit{equals}(\text{"}\mathit{N.Fuhr}\text{"},\text{"}\mathit{Fuhr}\text{"}) = 0.8$).

5.3 Schema mapping rules

In this subsection, schema S_2 is the standard schema and S_1 is the DL schema. Then, we have to map document attribute `S1#au` onto document attributes `S2#author` and `S2#editor` with probabilities 0.7 and 0.3, respectively (the probabilities are fictious). These "mapping rules" can be expressed by these pDatalog rules:

```
s2#title(D,V) :- s1#title(D,V).
0.7 s2#author(D,V) :- s1#au(D,V).
0.3 s2#editor(D,V) :- s1#au(D,V).

s1#title_contains(D,V) :- s2#title_contains(D,V).
0.7 s1#au_equals(Q,V) :- s2#author_equals(Q,V) & !s2#editor_equals(Q,V).
0.3 s1#au_equals(Q,V) :- !s2#author_equals(Q,V) & s2#editor_equals(Q,V).
1.0 s1#au_equals(Q,V) :- s2#author_equals(Q,V) & s2#editor_equals(Q,V).
```

The first block defines schema mapping rules for documents, the rules of the second block can be used for query transformation. As we want to transform queries from the standard schema into the DL schema, we have DL schema attributes in the head and standard schema attributes in the bodies for query rules. The probabilities decrease the weights of the created S_2 document attributes and the created DL schema conditions, respectively.

In some cases, string processing is required, too, e.g. for transforming *DateISO8601* date values into *DateEN* value when we map `S1#da` onto `S2#date`. For this we introduce some new (predefined) string processing predicates:

Definition 6 (Built-in predicates). *For string processing, the deterministic built-in string processing predicates* `starts-with(B,C)`, `concat(A,B,C,D)` *(also possible with more arguments to be concatenated),* `substring-after(A,B,C)`, `substring-before(A,B,C)` *and* `substring(A,B,C,D)` *can be used with the obvious meanings, e.g.:*

$$\texttt{concat(A,B,C)} \Leftrightarrow \textit{A is concatenation of B and C.}$$

In addition, pDatalog provides the binary predicates = and <> (for all constants, e.g. `=("A","B")`*) and >, >=, < and <= (only for numbers).*

In some cases comparison values also have to be transformed. Due to the inverted direction for query rules (with the DL schema in the head), the transformed comparison value can be computed easily.

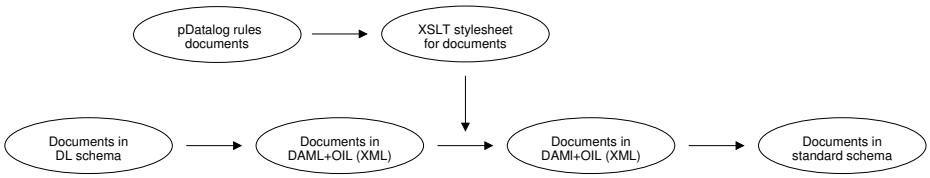
For other application areas, however, this approach is not feasible. If we want to perform retrieval in heterogeneous libraries, we need query rules with the same direction as for document rules. In our example, this means that S_2 attributes have to be used in the rule heads, and S_1 attributes must appear in the body. For retrieval involving schema mapping, we can formulate the following rules for our example schemas:

```
0.7 mapping(author,au).
s2#author(D,V) :- s1#au(D,V) & mapping(author,au).
s2#author_equals(Q,V) :- s1#au_equals(Q,V) & mapping(author,au).
s2#author_soundslike(Q,V) :- s1#au_equals(Q,V) & mapping(author,au).

s2#result(Q,D) :- s2#author_equals(Q,V1) & s2#author(D,V2) &
                  mind-dt#name_equals(V2,V1).
s2#result(Q,D) :- s2#author_soundslike(Q,V1) & s2#author(D,V2) &
                  mind-dt#name_soundslike(V2,V1).
```

The additional fact `0.7 mapping(author,au).` is required; if we weight the document and query mapping rules directly by 0.7, the weight is considered twice in the results.

Fig. 3. Document transformation process



5.4 Transforming queries and documents with XSLT

In federated DL systems, wrappers are used for querying the underlying libraries. Thus, we cannot directly retrieve DL documents with probabilistic Datalog. Of course the pDatalog rules could be evaluated for transforming queries and documents. We use a more general approach: DAML+OIL models are serialised in XML. XSLT is a well-established technology for transforming an XML document into other textual formats, e.g. another XML document (mapping between different DTDs or transforming the content), an HTML document (for creating web pages) or simple text. XSLT processors are freely available for a large number of programming languages.

The XSLT stylesheets are created based on the pDatalog rules. The mapping is straight-forward, but space precludes us from explaining it in detail. The transformation of pDatalog rules into XSLT is done once after the mapping rules are set up, and can be performed completely automatically. We obtain one stylesheet for transforming documents and another one for transforming queries.

Then, documents are transformed in three steps (see figure 3):

1. The documents (internal objects, referring to the DL schema) are converted into DAML+OIL and serialised in XML.
2. The resulting XML document is transformed into another XML document by the document transformation XSLT stylesheet. The resulting XML document is also a serialisation of a DAML+OIL model, corresponding to the standard schema.
3. The resulting XML document is parsed and converted to new internal objects.

Queries are transformed in a similar way.

6 Conclusion

In this paper we proposed a declarative approach for defining and handling uncertain schema mappings implemented in MIND. We start from a linear document model and convert it into DAML+OIL (the forthcoming ontology standard). Later this will allow us to extend our approach to documents with a nested structure (see below). Weights are modelled with reification (for schema mappings, we need weights only for specific cases, we do not weight the `rdfs:subClassOf` property). As rules cannot be expressed directly in DAML+OIL yet, we applied probabilistic Datalog for specifying schema mapping rules. As these rules and the stylesheets derived from them can be kept in text files, adding or modifying the schema mapping rules is very easy and does not require any new program code.

However, our approach raises a number of questions: DAML+OIL data types (which use XML Schema data types) do not support n-ary data type predicates, so we cannot model MIND data types and their search predicates directly in DAML+OIL. In addition, uncertainty is not supported by DAML+OIL. We overcome this problem by using reification, which is available in RDFS but not in DAML+OIL (so DAML+OIL tools are not able to assign semantics to these triples). The third problem is the lack of rules (the DAML-L language which adds rules to DAML+OIL is still not released). Thus, we mapped DAML+OIL onto pDatalog and used its rule and inference mechanism. However, it would be more pleasant to have an integrated framework which allows for stating rules and uncertain facts directly in DAML+OIL. Although this would lead to further incompatibilities, it might be worth extending the language itself by at least binary data type predicates, uncertainty and probabilistic rules.

In addition, we plan to learn the schema mapping rules. Rule candidates can be found by investigating the schema definitions, e.g. two attributes having the same name are good matching candidates, and attributes with the same data type might also match if different names are used. However these rules are not always correct, e.g. if the attributes *title* and *location* both have the data type *Text*. Thus, the candidates have to be verified with documents in both schemas. Although in general we do not have DL documents in the standard schema at this stage, we often have schema mapping rules for other libraries which have some overlap with the DL under consideration. Documents occurring in both DLs then can be used for improving the schema mapping rules.

Finally, we plan to extend schema mappings towards schemas with deeper structure. The ultimate goal would be to specify mappings between arbitrary DAML+OIL models, but this poses problems as we cannot specify properties for data types (see section 4). One solution would be to extend the DAML+OIL language; another one would be to restrict the approach to DAML+OIL descriptions with a deep structure but a form similar to the one we used in this paper.

7 Acknowledgements

This work is supported by the EU commission under grant IST-2000-26061 (project MIND).

References

[1] J. Biskup and D. W. Embley. Extracting information from heterogeneous information sources using ontologically specified target views. *Information Systems*, 28(3):169–212, 2003.

[2] D. Brickley and R. Guha. RDF vocabulary description language 1.0: RDF Schema, w3c working draft. Technical report, World Wide Web Consortium, Apr. 2002.

[3] S. Chawathe, H. Garcia-Molina, J. Hammer, K. Ireland, Y. Papakonstantinou, J. Ullman, and J. Widom. The TSIMMIS project: Integration of heterogeneous information sources. In *16th Meeting of the Information Processing Society of Japan*, pages 7–18. Tokyo, Japan, 1994.

[4] D. Connolly, F. v. Harmelen, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, and L. A. Stein. DAML+OIL (march 2001) reference description. Technical report, World Wide Web Consortium, 2001. http://www.w3.org/TR/daml+oil-reference.

[5] A. Doan, P. Domingos, and A. Y. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *SIGMOD Conference*, 2001.

[6] M. Endig, M. Hoding, G. Saake, K.-U. Sattler, and E. Schallehn. Federation services for heterogeneous digital libraries accessing cooperative and non-cooperative sources. In *Kyoto International Conference on Digital Libraries*, pages 314–321, 2000.

[7] N. Fuhr. A probabilistic framework for vague queries and imprecise information in databases. In *Proceedings of the 16th International Conference on Very Large Databases*, pages 696–707, Los Altos, California, 1990. Morgan Kaufman.

[8] N. Fuhr. Towards data abstraction in networked information retrieval systems. *Information Processing and Management*, 35(2):101–119, 1999.

[9] N. Fuhr. Probabilistic Datalog: Implementing logical information retrieval for advanced applications. *Journal of the American Society for Information Science*, 51(2):95–110, 2000.

[10] M. A. Goncalves, R. K. France, and E. A. Fox. MARIAN: Flexible interoperability for federated digital libraries. In P. Constantopoulos and I. T. Soelvberg, editors, *Research and Advanced Technology for Digital Libraries*, volume 2163 of *Lecture Notes in Computer Science*, pages 173–186, Berlin et al., 2001. Springer.

[11] E. Jeong and C.-N. Hsu. Induction of integrated view for XML data with heterogeneous DTDs. In Paques et al. [17], pages 151–158.

[12] O. Lassila and R. R. Swick. Resource description framework (RDF) model and syntax specification. W3C recommendation, World Wide Web Consortium, Feb. 1999. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[13] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proc. 27th VLDB Conference*, pages 49–58, 2001. http://www.research.microsoft.com/~philbe/CupidVLDB01.pdf.

[14] H. Nottelmann and N. Fuhr. Learning probabilistic Datalog rules for information classification and transformation. In Paques et al. [17], pages 387–394.

[15] H. Nottelmann and N. Fuhr. MIND: An architecture for multimedia information retrieval in federated digital libraries. In *Proceedings of the DELOS-Workshop on Interoperability in Digital Libraries*. DELOS-Network of Excellence on Digital Libraries, 2001.

[16] J. Z. Pan and I. Horrocks. Semantic web ontology reasoning in the $\mathcal{SHOQ}(\mathbf{D_n})$ description logic. In *Proc. of the 2002 Description Logic Workshop (DL 2002)*, 2002.

[17] H. Paques, L. Liu, and D. Grossman, editors. *Proceedings of the 10th International Conference on Information and Knowledge Management*, New York, 2001. ACM.

[18] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume I. Computer Science Press, Rockville (Md.), 1988.

[19] B. Wüthrich. On the learning of rule uncertainties and their integration into probabilistic knowledge bases. *Journal of Intelligent Information Systems*, 2:245–264, 1993.

[20] L. Xu and D. Embley. Combining the best of global-as-view and local-as-view for data integration. submitted for publication, http://www.deg.byu.edu/papers/PODS.integration.pdf.