Building Semantic Mappings from Databases to Ontologies

Yuan An and **John Mylopoulos**
Department of Computer Science
University of Toronto
Toronto, ON M5S 3G4, Canada
{yuana,jm}@cs.toronto.edu

Alex Borgida
Department of Computer Science
Rutgers University
Piscataway, NJ 08855, USA
borgida@cs.rutgers.edu

Abstract

A recent special issue of AI Magazine (AAAI 2005) was dedicated to the topic of *semantic integration* — the problem of sharing data across disparate sources. At the core of the solution lies the discovery the "semantics" of different data sources. Ideally, the semantics of data are captured by a *formal ontology* of the domain *together with* a *semantic mapping* connecting the schema describing the data to the ontology. However, establishing the semantic mapping from a database schema to a formal ontology in terms of formal logic expressions is inherently difficult to automate, so the task was left to humans. In this paper, we report on our study (An, Borgida, & Mylopoulos 2005a; 2005b) of a *semi-automatic* tool, called MAPONTO, that assists users to discover plausible semantic relationships between a database schema (relational or XML) and an ontology, expressing them as logical formulas/rules.

Introduction and Motivation

Semantic integration, the problem of sharing data across disparate sources, has been a long-standing challenge for the database community. A recent special issue of AI Magazine (AAAI 2005) was dedicated to the same topic. It is thus becoming clear that semantic integration also lies at the heart of many AI problems and that addressing it will require solutions that blend database and AI techniques. The core of semantic integration is the problem of discovering the semantics of different data sources. The solutions proposed for a number of important database challenges rely on an ontology to provide the *precise semantics* of the database schema. These include federated databases, data warehousing (Calvanese *et al.* 2001), information integration through mediated schemas (Levy, Srivastava, & Kirk 1996; Goasdoue & Rousset 2004), and peer-to-peer data management systems (Halevy *et al.* 2003). Since much information on the Web is generated from databases, the Semantic Web also requires associating semantics with database-resident data using ontologies (e.g., (Handschuh, Staab, & Volz 2003)).

In almost all of these cases semantics of the data is captured by some kind of *semantic mapping* between the database schema and the ontology. Although sometimes the mapping is just a *simple* association from terms to terms, in other cases what is required is a *complex* formula, often expressed in logic or a query language (Levy 2000; Amann *et al.* 2002).

For example, in both the Information Manifold data integration system presented in (Levy, Srivastava, & Kirk 1996), and the DWQ data warehousing system (Calvanese *et al.* 2001), rules of the form $T(\overline{X})$:- $\Phi(\overline{X}, \overline{Y})$ are used to connect a relational data source to an ontology described in a Description Logic, where $T(\overline{X})$ is a single predicate representing a table in the relational data source, and $\Phi(\overline{X}, \overline{Y})$ is a conjunctive formula over the predicates representing the concepts and relationships in the ontology.

In all previous work it has been assumed that *humans* specify the mapping rules – a difficult, time-consuming and error-prone task, especially since the specifier must be familiar with both the semantics of the database schema and the contents of the ontology. As the size and complexity of ontologies increase, it becomes desirable to have some kind of computer tool to assist people in the task. Note that the problem of semantic mapping discovery is superficially similar to that of database schema reverse engineering, whose goal is to find a conceptual model (often an Entity Relationship diagram) from which the database schema might have been derived; the significant difference is that in our case the ontology is already given, and we need to find a detailed connection between its conceptualizations of the world and that of the schema.

In this paper, we report on our studies (An, Borgida, & Mylopoulos 2005a; 2005b) in the context of the development of the MAPONTO tool, which assists users in discovering mapping rules between database schemas and ontologies. The discovery of data semantics is a necessarily heuristic task, but one which in our case is underpinned by a careful study of the standard design process relating the constructs of database schemas with those of conceptual modeling languages, such as the Entity Relationship model. In order to improve the effectiveness of MAPONTO, we assume some user input in addition to the database schema and the ontology. Specifically, inspired by the Clio project (Miller, Haas, & Hernandez 2000), we expect the MAPONTO

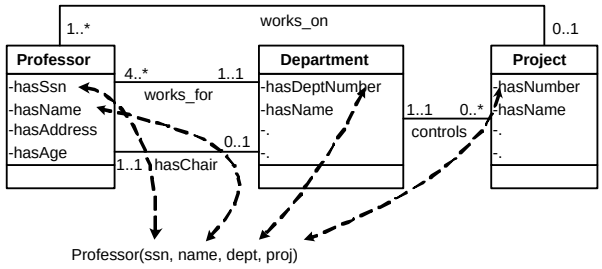

Figure 1: Relational table, Ontology, and Correspondences.

user to provide *simple correspondences* between atomic elements used in the database schema (e.g., column names in tables) and those in the ontology (e.g., attribute/"data type property" names on concepts). Given the set of correspondences, MAPONTO is expected to reason about the database schema and the ontology, and to generate a list of candidate rules for each individual component (e.g., a table) in the database schema. Ideally, one of the rules is the correct one — capturing the user's intention underlying the given correspondences. The hypothesis underlying this work is then that, compared to composing logical rules representing semantic mappings, it is much easier for users (i) to draw simple correspondences/arrows from atomic elements in the database schema to datatype properties of classes in the ontology[1] and then (ii) to evaluate proposed formulas returned by MAPONTO.

We illustrate the input/output behavior of MAPONTO by the following example.

Example Figure 1 shows an academic ontology containing some basic concepts (classes), attributes of concepts (datatype properties of classes), relationships between concepts (object properties of classes), and cardinality constraints on relationships. We actually construct an *ontology graph*, with concepts corresponding to nodes, and object properties to edges. (Attributes are treated as special edges.) In our diagrams, we represent the information using the UML notations. (In UML, cardinality constraints are written at the opposite end of the association: a Department has at least 4 Professors working for it, and a Professor works in one Department.)

Suppose we wish to discover the semantics of a relational table *Professor(ssn, name, dept, proj)* with key[2] ssn in terms of the academic ontology. And suppose that by looking at column names and the ontology graph, the user draws the simple correspondences shown as dashed lines in Figure 1. This indicates, for example, that the values in the ssn column correspond values of to the $hasSsn$ property/function of $Professor$ concept instances. Using prefixes $\mathcal{T}$ and $\mathcal{O}$ to distinguish tables in the relational schema and concepts in the ontology (both of which will be thought of as predicates), we represent the correspondences as follows:

$\mathcal{T} : Professor.ssn \leftrightsquigarrow \mathcal{O} : Professor.hasSsn$
$\mathcal{T} : Professor.name \leftrightsquigarrow \mathcal{O} : Professor.hasName$
$\mathcal{T} : Professor.dept \leftrightsquigarrow \mathcal{O} : Department.hasDeptNumber$
$\mathcal{T} : Professor.proj \leftrightsquigarrow \mathcal{O} : Project.hasNumber$

Given the above inputs, MAPONTO is expected to produce a list of plausible mapping rules, which should include the following rule, expressing the most plausible semantics for the table:

$\mathcal{T}$:Professor(ssn, name, dept, proj) :-
$\mathcal{O}$:Professor(x_1), $\mathcal{O}$:hasSsn(x_1,ssn), $\mathcal{O}$:hasName(x_1,name),
$\mathcal{O}$:Department(x_2), $\mathcal{O}$:works_for(x_1,x_2),
$\mathcal{O}$:hasDeptNumber(x_2,dept), $\mathcal{O}$:Project(x_3),
$\mathcal{O}$:works_on(x_1,x_3), $\mathcal{O}$:hasNumber(x_3,proj).

Note that, as explained in (Levy 2000), the above, admittedly confusing notation in the literature, should really be interpreted as the First Order Logic formula

$(\forall ssn, name, dept, proj)$ $\mathcal{T}$:Professor(ssn, name, dept, proj)
$\Rightarrow (\exists x_1, x_2, x_3)$ $\mathcal{O}$:Professor(x_1) $\wedge$...

because the ontology *explains* what is in the table (i.e., every tuple corresponds to a $\mathcal{O}$:Professor), rather than guaranteeing that the table satisfies the closed world assumption (i.e., for every $\mathcal{O}$:Professor there is a tuple in the table). ■

An intuitive (but somewhat naive) solution, inspired by early work of Quillian (Quillian 1968), is based on finding the *shortest* paths between concepts in the ontology. Technically, this involves (i) finding the minimum spanning tree(s) (actually Steiner trees [3]). connecting the concepts having datatype properties corresponding to table columns, and then (ii) encoding the tree(s) into rules. However, in some cases the spanning/Steiner tree may not provide the desired semantics for a table because of known relational schema design rules. For example, consider the relational table $Project$(<u>num</u>, $supervisor$), with key num corresponding to $\mathcal{O}$:$Project.hasNumber$, and column $supervisor$ corresponding to $\mathcal{O}$:$Professor.hasSsn$ in Figure 1. The minimum spanning tree consisting of $Project$, $Professor$, and the edge $works\_on$ probably does not match the semantics of table $Project$ because there are multiple $Professor$s working on a $Project$ according to the ontology cardinality, yet the table allows only one to be recorded, since $supervisor$ is functionally dependent on num, the key. Therefore we must seek a functional connection from $Project$ to $Professor$, and the connection will be the chair of the department controlling the project.

In MAPONTO, we use ideas of standard schema design from conceptual models in order to craft heuristics that systematically uncover the connections between the constructs of database schemas and those of ontologies. We propose to generate "reasonable" trees connecting the set of corresponded concepts in an ontology. For the sake of space limitation, in the sequel we will focus on reporting MAPONTO's

[1] In fact, there exist already tools used in schema matching which help perform such tasks using linguistic, structural, and statistical information (e.g., (Dhamankar *et al.* 2004; Rahm & Bernstein 2001)).

[2] The key of a table is a unique (set of) columns that is designated to uniquely identify each tuple in the table.

[3] A Steiner tree for a set M of nodes in graph G is a minimum spanning tree of M that may contain nodes of G which are not in M.

solution and results on building semantic mappings from *relational schemas* to ontologies appearing in (An, Borgida, & Mylopoulos 2005b). For XML schemas, the solution is in the same spirit except that it seeks to map a tree structure of an XML schema, instead of a relational table, to an ontology. Details can be found in (An, Borgida, & Mylopoulos 2005a).

Outline of Mapping Inference

For relational schemas, we aim at generating a formula of the form $T(X)$:- $\Phi(X, Y)$ to represent *semantic mappings*, where T is a table with columns X (which become arguments to its predicate), and Φ is a conjunctive formula over predicates representing the ontology, with Y existentially quantified, as usual. As implied earlier, we construct an ontology graph, where concept nodes are connected by directed edges – one for the property and one for its inverse. It will be important for our algorithm to distinguish *functional edges* — ones with upper bound cardinality of 1, and their composition: *functional paths*.

Given a table T, and correspondences to an ontology provided by a person or a tool, let the set $\mathcal{C}_T$ consist of those concept nodes which have at least one attribute corresponding to some column of T. Our task is to find semantic connections between concepts in $\mathcal{C}_T$, because attributes can then be connected to the result using the correspondence relation. The primary principle of our mapping inference algorithm is to look for *smallest* "reasonable" trees connecting nodes in $\mathcal{C}_T$. We will call such a tree a *semantic tree*.

Consider the case when $T(\underline{c}, b)$ is a table with key c, corresponding to an attribute f on concept C, and b is a foreign key corresponding to an attribute e on concept B.[4] Then for each value of c (and hence instance of C), T associates at most one value of b (instance of B). Hence the semantic mapping for T should be some formula that acts as a function from its first to its second argument. The semantic trees for such formulas look like functional edges in the ontology, and hence are preferable.

Conversely, for table $T'(\underline{c, b})$, where c and b are as above, values in column c are intended to be associated with multiple values of in column b, and conversely – otherwise the table would have been specified to have a smaller key: either c or b alone. Therefore a tree with a functional edge from C to B, or from B to C, is likely not to reflect a proper semantics.

To deal with such problems, our algorithm works in two stages: first connect the concepts corresponding to key columns into a *skeleton tree*, then connect the rest of the nodes in $\mathcal{C}_T$ to the skeleton using functional edges (whenever possible).

We must however also deal with the assumption that the relational schema and the ontology were developed independently, which implies that not all parts of the ontology are reflected in the database schema. This complicates things, since in building the semantic tree we may need to go through additional nodes, which end up not corresponding to columns of the relational table. An example of such a situation arises when the ontology contains detailed aggregation hierarchies (e.g., *city* part-of *township* part-of *county* part-of *state*), which are abstracted in the database (e.g., a table with columns for *city* and *state* only).

We have chosen to flesh out the above principles in a systematic manner by considering the behavior of our proposed algorithm on relational schemas designed from Extended Entity Relationship (EER) diagrams — a technique widely covered in undergraduate database courses. (We refer to this er2rel *schema design*.) One benefit of this approach is that it allows us to prove that our algorithm, though heuristic in general, is in some sense "correct" for a certain class of schemas. Of course, in practice such schemas may be "denormalized" in order to improve efficiency, and, as we mentioned, only parts of the ontology may be realized in the database. Our algorithm uses the general principles enunciated above even in such cases, with relatively good results in practice. Also note that the assumption that a given relational schema was designed from some EER conceptual model does not mean that given ontology is this EER model, or is even expressed in the EER notation. In fact, our heuristics have to cope with the fact that it is missing essential information, such as keys for weak entities.

In particular, we have specified a mapping $\tau(C)$ of the er2rel design returning a relational table for every EER component C. Furthermore, we have observed that the er2rel design, for our purposes, associates with a relational table T a number of additional notions. An important one is *anchor*. An anchor is the central object in the ontology from which T is derived. (for details see (An, Borgida, & Mylopoulos 2005b)).

Consequently, the algorithm consists of two major functions getSkeleton and getTree. The function getSkeleton returns a set of (skeleton, anchor)-pairs, when given a table T and a set of correspondences L from key(T) (the key column(s) of T). The function is essentially a recursive algorithm attempting to reverse the mapping $\tau(C)$. The function getTree finds the entire semantic tree by connecting the concepts corresponding to the rest of the columns, i.e., nonkey(T), to the anchor.

Summary of the Results

We now summarize our results. Theoretically, if the schema was derived from an EER diagram following the er2rel design and the ontology encodes the EER diagram, then our algorithm in most cases satisfies the following two properties: (1) A sense of "completeness": the algorithm lists, among others the correct semantics; (2) A sense of "soundness": if for such a table there are multiple semantic trees returned by the algorithm, then each of the trees would produce an indistinguishable relational table according to the er2rel design.

Nonetheless, it is crucial to test the algorithm on real-world schemas and ontologies. The algorithm has been implemented as a third-party plugin of the well-known knowledge base management system Protégé[5], which is an open

[4] In a relational table T, a column f is specified to be a *foreign key* to table S (which may be identical to T), if every value in f must appear in the key column of table S.

[5] http://protege.stanford.edu

platform for ontology modeling and knowledge acquisition. Ontologies are described in OWL ontology language in MAPONTO.

Our test data were obtained from various sources, and we have ensured that the databases and ontologies were developed independently. We have tested 5 schemas and 5 ontologies. The schemas and ontologies have a variety of sizes and complexities. Their characteristic features can be found in (An, Borgida, & Mylopoulos 2005b).

To evaluate MAPONTO, we sought to understand whether MAPONTO could produce the intended mapping formula if the simple correspondences were given. We were especially concerned with the number of formulas presented by MAPONTO for users to sift through. Further, we wanted to know whether MAPONTO was still useful if the correct formula was not generated. In this case, we expected that a user could more easily debug a generated formula to reach the correct one instead of creating it from scratch. The detailed experimental results are available in the original publication (An, Borgida, & Mylopoulos 2005b).

In summary, our results indicate that MAPONTO only presents a few mapping formulas (≤ 4) for users to examine. To measure the overall performance, we manually created the mapping formulas for all the 36 tables and compared them to the formulas generated by MAPONTO. We observed that MAPONTO produced correct formulas for 31 tables. This demonstrates that MAPONTO is able to infer the semantics of many relational tables occurring in practice in terms of an independently developed ontology.

We were also interested in the usefulness of MAPONTO in those cases where the formulas generated were not the intended ones. For each such formula, we compared it to the manually generated correct one, and we used a very coarse measurement to record how much effort it would take to "debug" the generated formula: the number of changes of predicate names in a formula. Our experience working with the data sets shows that at average only about 30% predicates in a single incorrect formula returned by MAPONTO needed to be modified to reach the correct formula. This is a significant saving in terms of human labors.

Our results also indicate that execution times were not significant (see (An, Borgida, & Mylopoulos 2005b)), since, as predicted, the search for subtrees and paths took place in a relatively small neighborhood.

Conclusion and Future Work

Numerous additional sources of knowledge, including richer ontologies, actual data stored in the databases, linguistic and semantic relationships between schema elements and the ontology, can be used to refine the suggestions of MAPONTO. In addition, more complex correspondences (e.g., from columns to sets of attribute names or class names), should also be investigated in order to generate the full range of mappings encountered in practice.

From our experience in developing and evaluating MAPONTO, we venture to generalize several observations that might be applicable to AI research in general:

- In solving complex problems, there is considerable benefit to be gained from finding hints that can be provided as additional inputs by (naive) users. In our case, as inspired by Clio (Miller, Haas, & Hernandez 2000), *correspondences* played this role.
- Finding an important core class of problems, which can be thoroughly studied, provides the opportunity to actually *prove* that the otherwise heuristic algorithm is correct for a subclass of inputs.
- For tools producing complex artifacts, it may be reasonable to measure success not just by the number of cases where the tool produced the exact answer desired, but also by the ease with which incorrect answers can be modified to produce correct ones. Ultimately, tools will be successful if they can reduce the amount of human effort required to perform a given task.

References

AAAI. 2005. *AI Magazine: Special Issue on Semantic Integration*. Volume 26, Number 1.

Amann, B.; Beeri, C.; Fundulaki, I.; and Scholl, M. 2002. Ontology-Based Integration of XML Web Resources. In *ISWC'02*.

An, Y.; Borgida, A.; and Mylopoulos, J. 2005a. Constructing complex semantic mappings between xml data and ontologies. In *ISWC'05*.

An, Y.; Borgida, A.; and Mylopoulos, J. 2005b. Inferring complex semantic mappings between relational tables and ontologies from simple correspondences. In *ODBASE'05*.

Calvanese, D.; Giacomo, G. D.; Lenzerini, M.; Nardi, D.; and Rosati, R. 2001. Data Integration in Data Warehousing. *J. of Coop. Info. Sys.* 10(3):237–271.

Dhamankar, R.; Lee, Y.; Doan, A.; Halevy, A.; and Domingos, P. 2004. iMAP: Discovering Complex Semantic Matches between Database Schemas. In *SIGMOD'04*.

Goasdoue, F., and Rousset, M.-C. 2004. Answering Queries using Views: a KRDB Perspective for the Semantic Web. *ACM TOIT* 4(3):255 – 288.

Halevy, A.; Ives, Z.; Mork, P.; and Tatarinov, I. 2003. Piazza: Data Management Infrastructure for Semantic Web Applications. In *WWW'03*.

Handschuh, S.; Staab, S.; and Volz, R. 2003. On Deep Annotation. In *Proc. WWW'03*.

Levy, A. Y.; Srivastava, D.; and Kirk, T. 1996. Data Model and Query Evaluation in Global Information Systems. *J. of Intelligent Info. Sys.* 5(2):121–143.

Levy, A. Y. 2000. Logic-Based Techniques in Data Integration. *Jack Minker (ed), Logic Based Artificial Intelligence, Kluwer Publishers*.

Miller, R.; Haas, L. M.; and Hernandez, M. A. 2000. Schema Mapping as Query Discovery. In *VLDB'00*.

Quillian, M. R. 1968. Semantic Memory. In *Semantic Information Processing*, 227–270. The MIT Press.

Rahm, E., and Bernstein, P. A. 2001. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal* 10.