

# Representing and Reasoning about Mappings between Domain Models

**Jayant Madhavan**

University of Washington  
jayant@cs.washington.edu

**Philip A. Bernstein**

Microsoft Research  
philbe@microsoft.com

**Pedro Domingos**

University of Washington  
pedrod@cs.washington.edu

**Alon Y. Halevy**

University of Washington  
alon@cs.washington.edu

## Abstract

Mappings between disparate models are fundamental to any application that requires interoperability between heterogeneous data and applications. Generating mappings is a labor-intensive and error prone task. To build a system that helps users generate mappings, we need an explicit representation of mappings. This representation needs to have well-defined semantics to enable reasoning and comparison between mappings. This paper first presents a powerful framework for defining languages for specifying mappings and their associated semantics. We examine the use of mappings and identify the key inference problems associated with mappings. These properties can be used to determine whether a mapping is adequate in a particular context. Finally, we consider an instance of our framework for a language representing mappings between relational data. We present sound and complete algorithms for the corresponding inference problems.

## 1 Introduction

The emergence of the World-Wide Web (WWW) and the promise of the Semantic Web have refocused our attention on building systems in which knowledge can be shared in an ad hoc, distributed environment. For example, information integration systems allow queries to be answered using a set of data sources on the WWW or across several databases in an enterprise. The Semantic Web goes a step further, and envisions a less centralized architecture where agents can coordinate tasks using rich ontologies.

A crucial element of all these system architectures is the ability to map between different models of the same or related domains. It is rare that a global ontology or schema can be developed for such a system. In practice, multiple ontologies and schemas will be developed by independent entities, and coordination will require mapping between the different models. Such a mapping will be a set of formulae that provide the semantic relationships between the concepts in the models. There will always be more than one representation of any domain of discourse. Hence, if knowledge and data are to be shared, the problem of mapping between models is as fundamental as modeling itself.

In current systems, mappings between models are provided manually in a labor-intensive and error-prone process,

Copyright © 2002, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

which is a major bottleneck to scaling up systems to a large number of sources. Recently, several tools have been developed to provide support for constructing mappings. The approaches underlying these tools are usually based on heuristics that identify structural and naming similarities between models (Noy & Musen 2000; Rahm & Bernstein 2001) or on using machine learning to learn mappings (Doan, Domingos, & Halevy 2001; Lacher & Groh 2001; Doan *et al.* 2002; Berlin & Motro 2002). In both cases, the systems require feedback from a user to further refine a proposed mapping.

Our opening claim in this paper is that the study of mappings between models needs to be recognized as an important item on the research agenda. A robust tool for aiding users to generate mappings will need the ability to incorporate heuristics (domain dependent and independent), use machine learning techniques, and incorporate user feedback. To exploit all of these techniques in concert and in a principled fashion, we must have an *explicit* and well-defined representation of mappings that enables reasoning about mappings and evidence supporting these mappings, comparing between different mappings, and ultimately, learning mappings.

First, we must define semantics for such mappings. We therefore offer a framework for defining representations of mappings with associated semantics (Section 3). An instance of the framework is a particular mapping language, where the source and destination representation languages have been fixed. The framework makes three important contributions. First, it enables mapping between models in vastly different representation languages (e.g. relational data, XML, RDF, DAML+OIL (Horrocks, van Harmelen, & Patel-Schneider 2001)) *without* first translating the models into a common language. Second, the framework introduces a *helper model* in the mapping, which is needed in cases where it is not possible to map directly between a pair of models. Third, the framework enables representing mappings that are either incomplete or lose information.

While this paper focuses on the issue of representing mappings and reasoning about them, the problem of generating the mappings is still the ultimate goal. Our recent work on the problem of generating mappings (Doan, Domingos, & Halevy 2001; Madhavan, Bernstein, & Rahm 2001; Doan *et al.* 2002) has led us to the conclusion that in order to make further progress on the problem we need a well







lows. (The definition is written w.r.t. queries over  $T_1$ , but it can be defined similarly for queries over  $T_2$ ).

**Definition 3** Let  $M$  be a mapping between models  $T_1$ ,  $T_2$  and an optional helper model  $T_3$ , and let  $Q$  be a query over  $T_1$ . We say that the mapping  $M$  enables query answering of  $Q$  if the following holds.

Let  $T_2'$  be an extension of  $T_2$ . Let  $\mathcal{I}$  be the set of interpretations of  $T_1$  for which there exists an interpretation  $I_3$  of  $T_3$  and a logical model  $I_2$  of  $T_2'$  such that  $(I_1, I_2, I_3) \models M$ . Then, for every tuple of constants  $\bar{a}$ , either  $I_1 \models Q(\bar{a})$  for every  $I_1 \in \mathcal{I}$  or  $I_1 \not\models Q(\bar{a})$  for every  $I_1 \in \mathcal{I}$ .

Informally, the definition states that given an extension of  $T_2$ , it uniquely determines the answers for  $Q$  over  $T_1$ .

**Example 3** Consider the two models in example 1. For simplicity we ignore the constraints on the addresses, and hence, there is no need for a helper model. Instead, consider that the grades at the two universities are on different scales. Consider the mapping formula

YourUniv.student(std,x,y, $\alpha \times$  gpa,z) =  
MyUniv.STUDENT(std) $\wedge$ MyUniv.Grade(std,gpa)

where  $\alpha$  is an unknown conversion factor between the grades. Consider the following two queries over MyUniv,

- (1) max-grade :- Max{ gpa | MyUniv.Grade(std,gpa)}, and
- (2) high-grade(std,gpa) :- MyUniv.Grade(std,gpa)  $\wedge$  gpa  $\geq$  3.8

The mapping formula enables answering query (1) on YourUniv because the max does not depend on the particular scale, i.e., given any database of students for YourUniv, the answer to max-grade will be uniquely determined. However the mapping does not enable answering query (2) because the answers to the query will differ depending on the value of  $\alpha$ .

**Mapping Inference:** A formal semantics for mappings enables a precise definition of several reasoning tasks. One particularly important task is determining whether a given mapping formula is entailed by a mapping (in which case, it doesn't provide new information). This, in turn, enables other reasoning tasks such as (1) whether two mappings are equivalent, and (2) whether a mapping is minimal (i.e., removing any formula from the mapping loses information). We formalize the mapping inference problem as follows:

**Definition 4** Let  $M$  be a mapping between models  $T_1$  and  $T_2$ , and let  $e$  be a formula over  $T_1$  and  $T_2$ . The mapping inference problem is to determine whether  $M \models e$ , i.e., whether whenever  $(I_1, I_2, I_3) \models M$ , then  $(I_1, I_2, I_3) \models e$ , where  $(I_1, I_2, I_3)$  are interpretations for  $T_1$ ,  $T_2$  and a helper model  $T_3$ , respectively.

**Mapping Composition:** The need to compose mappings arises when we need to piece together a mapping given other mappings. In its simplest form, given two mappings from models  $T_0$  to  $T_1$  and  $T_1$  to  $T_2$ , we might have to obtain a mapping directly between  $T_0$  and  $T_2$ . Formally, the problem is defined as follows.

**Definition 5** Let  $M_{01}$  be a mapping between models  $T_0$  and  $T_1$ , and  $M_{12}$  be a mapping between models  $T_1$  and  $T_2$ . Let  $e$  be a formula over  $T_0$  and  $T_2$ . The mapping composition problem is to determine whether  $\{M_{01}, M_{12}\} \models e$ , i.e., whether whenever a triple of interpretations  $(I_0, I_1, I_2)$  for  $T_0, T_1, T_2$  respectively, satisfies  $(I_0, I_1) \models M_{01}$  and  $(I_1, I_2) \models M_{12}$  then  $(I_0, I_1, I_2) \models e$ .

In section 5, we solve the problems of query answerability, mapping inference and mapping composition for a mapping language over relational representations.

## 5 Mappings between Relational Models

In this section we consider a specific instance in our framework, show that the properties discussed in the previous section can be decided, and specify the complexity of the decision problems. The language we consider expresses mappings between a pair of relational-database models (i.e., schemas). Formally, each model includes a set of relations, and an extension of the model includes a set of ground facts for each relation. The expressions in our language are of the following form:

$$q(\bar{X}) \Leftrightarrow p_1(\bar{X}_1) \wedge \dots \wedge p_n(\bar{X}_n)$$

where the  $p_i$ 's are database relations,  $\bar{X}_i$  and  $\bar{X}$  are tuples of variables (and  $\bar{X} \subseteq \bar{X}_1 \cup \dots \cup \bar{X}_n$ ). The variables in  $\bar{X}$  are called *distinguished* and the other variables are called *existential*. The distinguished variables are assumed to be universally quantified, while the others are existentially quantified. We also use the same notation to describe queries, and the answer to the query is the set of tuples for  $\bar{X}$ . We note that although simple, this is a rather expressive language, since it enables defining relations in a very common subset of the SQL query language.

A formula in our language has the form  $q_1(\bar{X}) = q_2(\bar{X})$ , where  $q_1$  ( $q_2$ ) is defined by the left-hand side of a single expression over the relations in  $T_1$  ( $T_2$ ). We assume mappings that are conservative augmentations. Recall that this only means that a mapping cannot entail new conclusions about the *schema* but still allows to deduce new ground facts.

Our results assume the following restriction on mappings in our language: a mapping is said to be *variable-partitionable* if whenever a variable  $x$  appears in position  $i$  of relation  $R$  in one of the formulae in a mapping  $M$ , and is an existential variable in the formula, then there is no formula in  $M$  where a variable that appears in position  $i$  of  $R$  is distinguished. Note that this property holds trivially in the common case where the expressions in  $M$  do not contain existential variables.

Our first result concerns query answerability in our language. In the complexity analysis we are mostly concerned with the size of data in  $T_1$  and  $T_2$ .

**Theorem 1** Let  $M$  be a variable-partitionable mapping between models  $T_1$  and  $T_2$ , and let  $Q$  be a query in our language. The problem of deciding whether  $M$  enables answering  $Q$  is NP-complete.  $\square$

The proof is based on showing that  $M$  enables answering  $Q$  if and only if there exists an *equivalent* rewriting of the

query  $Q$  using the expressions over  $T_1$  used in  $M$ . The latter problem is NP-complete (Halevy 2001).  $\square$

Our next result shows that reasoning about mappings is decidable:

**Theorem 2** *Let  $M$  be a mapping between models  $T_1$  and  $T_2$ , and let  $u(\bar{X}) = v(\bar{X})$  be a mapping formula. If  $M$  is variable-partitionable, then the problem of deciding whether  $M \models u(\bar{X}) = v(\bar{X})$  is NP-complete.  $\square$*

The proof of Theorem 2 is quite lengthy and omitted because of space limitations. Below we sketch the algorithm underlying the proof.

Suppose  $M$  is of the form  $u_i(\bar{X}) = v_i(\bar{X})$  for  $1 \leq i \leq k$ , where the  $u_i$ 's and the  $v_i$ 's are defined using expressions in the language. The  $u_i$ 's are defined over the schema of  $T_1$  and the  $v_i$ 's over  $T_2$ . We denote by  $Q_v$  (resp.  $Q_u$ ) the rewriting of  $v$  (resp.  $u$ ) using  $v_1, \dots, v_k$  (resp.  $u_1, \dots, u_k$ ). The rewriting can be found using the techniques in (Halevy 2001).

We denote by  $Q_v^u$  the result of substituting  $u_i$  for  $v_i$  in  $Q_v$ . Note that  $Q_v^u$  is an expression over the schema  $T_1$ . The algorithm checks that (1)  $Q_v$  is equivalent to  $v$ , and (2)  $Q_v^u$  is equivalent to  $u$ , and outputs that  $M \models u(\bar{X}) = v(\bar{X})$  if and only if both (1) and (2) hold. Equivalence checking for these expressions is NP-complete (Sagiv & Yannakakis 1981).  $\square$

It should be noted that the above algorithm is sound even when  $M$  is not variable-partitionable.

**Example 4** *Consider two relational models identical to YourUniv (from example 1) - Univ<sub>1</sub> and Univ<sub>2</sub>. Model Univ<sub>i</sub> has tables student<sub>i</sub>, enrolled-in<sub>i</sub> and course<sub>i</sub> ( $i=1,2$ ). Consider the following view definitions,*

Major-Gender<sub>i</sub>(std,mjr,gnd) :- student<sub>i</sub>(std,x,mjr,y,gnd),  
Some-Class<sub>i</sub>(std) :- enrolled-in<sub>i</sub>(std,x),  
Some-Student<sub>i</sub>(crs) :- enrolled-in<sub>i</sub>(x,crs), and  
Major<sub>i</sub>(std,mjr) :- student<sub>i</sub>(std,x,mjr,y,z).

*Consider the mapping  $M = \{ \text{course}_1 = \text{course}_2, \text{Some-Class}_1 = \text{Some-Class}_2, \text{Major-Gender}_1 = \text{Major-Gender}_2 \}$ . Note that Major<sub>i</sub> is rewritable in terms of Major-Gender<sub>i</sub>, but Some-Student<sub>i</sub> is not rewritable in terms of course<sub>i</sub>, Some-Class<sub>i</sub> and Major-Gender<sub>i</sub>. It follows that,*

- *Given an extension of Univ<sub>2</sub>,  $M$  enables answering of Major<sub>1</sub>, but does not enable answering of Some-Student<sub>1</sub>.*
- *$M$  implies Major<sub>1</sub> = Major<sub>2</sub>, but does not imply Some-Student<sub>1</sub> = Some-Student<sub>2</sub>.*

Our final result shows that compositionality is also decidable for our language:

**Theorem 3** *Let  $T_0$ ,  $T_1$  and  $T_2$  be models, and let  $M_{01}$  (resp.  $M_{12}$ ) be a variable-partitionable mapping between  $T_0$  and  $T_1$  (resp.  $T_1$  and  $T_2$ ). Let  $e$  be a formula over  $T_0$  and  $T_2$ . Deciding whether  $M_{01}, M_{12} \models e$  is NP-complete.*

The proofs for each of the results in this section are available in the full version of our paper.

## 6 Discussion and Related Work

Thus far, our discussion has focused on manipulating mappings mostly as a form of logical reasoning. The next steps

are to incorporate *inaccurate* mappings and handle *uncertainty* about mappings. Inaccuracy arises because in many contexts there is no precise mapping. Reasoning about uncertainty of mappings is necessary because the generation of mappings often involves combining different heuristics and learned hypotheses. The two issues are highly intertwined, because heuristics are often used as a vehicle to choose the best mapping when no perfectly accurate one exists. We believe that some form of first-order probabilistic representation of mappings will be the appropriate tool for representing mappings, but none of the existing proposals (e.g. (Pfeffer *et al.* 1999)) provide sufficient expressive power for reasoning about formulas with variables and quantification.

In general, mappings may be inaccurate for multiple reasons: either the mapping language is too restricted to express more accurate mappings, or the concepts in the two models simply do not match up precisely. As an example of the former, in ontology mapping, it might not be possible to get an exact mapping when mapping formulae are restricted to be one-one correspondences between concepts.

When no accurate mapping exists, the issue becomes choosing the *best* mapping from the viable ones. A common heuristic is to restrict the language for expressing the mapping in order to prune the search space (of viable mappings). Mapping formulae or correspondences are produced in one of two ways: (1) applying a set of matching rules (Mitra, Wiederhold, & Jannink 1999; Noy & Musen 2000; Stumme & Maedche 2001), or (2) evaluating interesting similarity measures that compare and help choose from the set of all possible correspondences (Calvanese *et al.* 2001; Melnik, Garcia-Molina, & Rahm 2002; Lacher & Groh 2001; Doan *et al.* 2002). These heuristics often use syntactic information such as the names of the concepts or nesting relationships between concepts. They might also use semantic information such as the inter-relationship between concepts (slots of frames in (Noy & Musen 2000)), the types of the concepts, or the labeled-graph structure of the models (Calvanese *et al.* 2001; Melnik, Garcia-Molina, & Rahm 2002). Other techniques use data instances belonging to input models to estimate the likelihood of these correspondences (Doan, Domingos, & Halevy 2001; Lacher & Groh 2001; Stumme & Maedche 2001; Berlin & Motro 2002). Several systems also have powerful features for the efficient capture of user interaction (Noy & Musen 2000; McGuinness *et al.* 2000). In (Chalupsky 2000), an expressive rule language is proposed to support transformations between different symbolic representations. In (Rahm & Bernstein 2001) the authors survey the schema matching techniques in the database literature.

It should be noted that in some contexts there may be other factors than accuracy that affect the choice of mapping, such as the cost of applying the mapping to data. For example, in a data management application, mappings that result in more efficient query execution plans may be preferred.

The work of (Calvanese, Giuseppe, & Lenzerini 2001) describes a specialization of our framework to ontology integration. Their goal is to define the semantics of an ontology integration system whose sources are described by different

ontologies. Since they focus on data integration, they require a global mediated ontology, whereas in our framework a mediated ontology will exist only if it makes sense for the task at hand. They do not address any of the inference questions that we outlined in Section 5.

The results we presented in Section 5 make use of the theory of answering queries using views (Friedman, Levy, & Millstein 1999; Halevy 2001), but extend the theory in important ways. We defined the notion of mappings enabling query answering, and showed the exact role that equivalent rewritings, which have been studied in that literature, play in this context. In addition, we presented results on equivalence and compositionality of mappings, which were not studied in previous work.

## 7 Conclusions

Mappings are crucial components of many applications. To reason simultaneously about multiple models, we need well-defined semantics for mappings. In addition to representing mappings and making inferences across models, this enables us to combine multiple techniques to mapping generation. In this paper we proposed a framework and associated semantics for mappings. We demonstrated an instantiation of our framework to answer questions about query answerability, mapping inference and composition for relational models.

Our work lays the foundation for research towards a broader goal of building a system that combines multiple techniques in helping users create mappings between models. The system should be able to improve over time and even transfer knowledge from one mapping task to another when appropriate. The next step in this work is to develop an appropriate probabilistic representation of mappings that enables capturing both inaccuracy and uncertainty.

## Acknowledgments

We thank Corin Anderson, Rachel Pottinger, Pradeep Shenoy, and the anonymous reviewers for their invaluable comments. This work is supported by NSF Grants 9523649, 9983932, IIS-9978567, and IIS-9985114. The third author is also supported by an IBM Faculty Partnership Award. The fourth author is also supported by a Sloan Fellowship and gifts from Microsoft Research, NEC and NTT.

## References

Berlin, J., and Motro, A. 2002. Database Schema Matching Using Machine Learning with Feature Selection. In *Proc. of the 14th Int. Conf. on Advanced Information Systems Engg. (CAiSE02)*.

Calvanese, D.; Castano, S.; Guerra, F.; Lembo, D.; Melchiorri, M.; Terracina, G.; Ursino, D.; and Vincini, M. 2001. Towards a Comprehensive Framework for Semantic Integration of Highly Heterogeneous Data Sources. In *Proc. of the 8th Int. Workshop on Knowledge Representation meets Databases (KRDB2001)*.

Calvanese, D.; Giuseppe, D. G.; and Lenzerini, M. 2001. Ontology of Integration and Integration of Ontologies. In *Proc. of the Int. Workshop on Description Logics (DL2001)*.

Chalupsky, H. 2000. Ontomorph: A Translation system for symbolic knowledge. In *Proc. of the 7th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR2000)*.

Doan, A.; Madhavan, J.; Domingos, P.; and Halevy, A. 2002. Learning to Map between Ontologies on the Semantic Web. In *Proc. of the 11th Int. World Wide Web Conf. (WWW2002)*.

Doan, A.; Domingos, P.; and Halevy, A. Y. 2001. Reconciling Schemas of Disparate Data Sources: A Machine Learning Approach. In *Proc. of the ACM SIGMOD Conf.*

Duschka, O. M., and Genesereth, M. R. 1997. Answering recursive queries using views. In *Proc. of PODS*.

Friedman, M., and Weld, D. 1997. Efficient execution of information gathering plans. In *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence (IJCAI)*.

Friedman, M.; Levy, A.; and Millstein, T. 1999. Navigational Plans for Data Integration. In *Proc. of the 16th National Conf. on Artificial Intelligence (AAAI)*.

Gribble, S.; Halevy, A.; Ives, Z.; Rodrig, M.; and Suciu, D. 2001. What can databases do for peer-to-peer? In *ACM SIGMOD WebDB Workshop 2001*.

Halevy, A. Y. 2001. Answering queries using views: A survey. *VLDB Journal* 10(4).

Horrocks, I.; van Harmelen, F.; and Patel-Schneider, P. 2001. DAML+OIL. <http://www.daml.org/2001/03/daml+oil-index.html>.

Lacher, M. S., and Groh, G. 2001. Facilitating the exchange of explicit knowledge through ontology mappings. In *Proc. of the 14th Int. FLAIRS conference*.

Lambrech, E.; Kambhampati, S.; and Gnanaprakasam, S. 1999. Optimizing recursive information gathering plans. In *Proc. of the 16th Int. Joint Conf on Artificial Intelligence (IJCAI)*, 1204–1211.

Levy, A. Y.; Rajaraman, A.; and Ordille, J. J. 1996. Query answering algorithms for information agents. In *Proc. of the 13th National Conf. on Artificial Intelligence (AAAI)*.

Madhavan, J.; Bernstein, P. A.; and Rahm, E. 2001. Generic Schema Matching with Cupid. In *Proc. of the 27th Int. Conf. on Very Large Databases (VLDB)*.

McGuinness, D.; Fikes, R.; Rice, J.; and Wilder, S. 2000. The Chimaera Ontology Environment. In *Proc. of the 17th National Conf. on Artificial Intelligence (AAAI)*.

Melnik, S.; Garcia-Molina, H.; and Rahm, E. 2002. Similarity Flooding: A Versatile Graph Matching Algorithm. In *Proc. of the 18th Int. Conf. on Data Engg. (ICDE)*.

Mitra, P.; Wiederhold, G.; and Jannink, J. 1999. Semi-automatic Integration of Knowledge Sources. In *Proc. the 2nd Int. Conf. on Information FUSION*.

Noy, N., and Musen, M. 2000. PROMPT: Algorithm and Tool for Automated Ont. Merging and Alignment. In *Proc. of the 17th National Conf. on Artificial Intelligence (AAAI)*.

Pfeffer, A.; Koller, D.; Milch, B.; and Takusagawa, K. 1999. SPOOK: A system for probabilistic object-oriented knowledge representation. In *Proc. of the 15th Conf. on Uncertainty in Artificial Intelligence (UAI)*.

Rahm, E., and Bernstein, P. A. 2001. A survey on approaches to automatic schema matching. *VLDB Journal* 10(4).

Sagiv, Y., and Yannakakis, M. 1981. Equivalence among relational expressions with the union and difference operators. *Journal of the ACM* 27(4):633–655.

Stumme, G., and Maedche, A. 2001. FCA-MERGE: Bottom-Up Merging of Ontologies. In *Proc. of the 17th Int. Joint Conf. on Artificial Intelligence (IJCAI)*.