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-de£ned semantics to enable reasoning and comparison between mappings. This paper £rst presents a powerful framework for de£ning 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, 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 re£ne 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-de£ned representation of mappings that enables reasoning about mappings and evidence supporting these mappings, comparing between different mappings, and ultimately, learning mappings.

First, we must de£ne semantics for such mappings. We therefore offer a framework for de£ning 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 £xed. 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* £rst 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

founded representation of mappings. With such a representation, we will be able to combine in a principled way different types of knowledge about mappings and make some inferences that will further help the process. Previous work has mostly avoided de£ning mapping semantics formally either because the focus was on how to obtain the mappings, or because £nding mappings was a part of other application dependent goals (e.g., ontology merging, data migration), and hence the properties of mappings themselves were not investigated.

A mapping between models rarely maps all the concepts in one model to all the concepts in another, because models don't usually cover precisely the same domains. As a result, there are usually several possible mappings, so we need a way of determining whether a mapping is acceptable for a particular task. In Section 4 we identify a core set of properties that can be used to determine whether a mapping suf£ces for a particular context: (1) the ability to answer queries over a model, (2) inference of mapping formulas, and (3) compositionality of mappings.

As our £nal contribution, we demonstrate the application of the framework to a particular mapping language that maps between relational models. The mapping language we consider is the one used in several data integration systems (Levy, Rajaraman, & Ordille 1996; Duschka & Genesereth 1997; Friedman & Weld 1997; Lambrecht, Kambhampati, & Gnanaprakasam 1999; Gribble *et al.* 2001). However, the point of this section is not to argue for this particular language, but to illustrate our framework on one concrete (and important) case. We describe sound and complete algorithms for deciding each of the above properties. These results are also of independent interest in answering queries using views (Halevy 2001) and information integration.

2 Problem de£nition

In this section we £rst demonstrate the pervasive role that mappings between models play in several applications, and then outline the desiderata for formalisms specifying and manipulating such mappings.

Mappings between models are the foundation behind the following classes of applications:

Information integration and the Semantic Web: In many contexts, data resides in a multitude of data sources (e.g., many databases within an enterprise, data sources on the WWW, etc). Data integration enables users to ask queries in a uniform fashion, without having to access each data source independently (or even know about the existence of multiple sources). In an information integration system, users ask queries over a *mediated schema*, which captures only the aspects of the domain that are salient to the application. The mediated schema is solely a logical one (no data is stored in it), and mappings are used to describe the relationship between the mediated schema and the schemas of the data sources. The Semantic Web goes one step further. Here, there is no central mediated schema, tasks involve actions in addition to queries, and coordination is achieved using ontologies. Here too, mappings between ontologies are necessary for agents to interoperate.

Data migration: The goal of data migration is to take data/knowledge from an external source and merge it with some existing data stored using a different schema or ontology. In many cases, some information may be lost in the transformation since the external source does not necessarily represent the same aspects of the domain as the local one. Here too, the migration needs to be guided by a mapping between the external and internal representations. In *data warehousing*, several data sources are used to populate a data warehouse, which is later used for analysis queries. Here it is usually assumed that some information is lost during the loading of the warehouse (e.g., the warehouse may have only summary data, rather than detailed data).

Ontology merging: Several applications require that we combine multiple ontologies into a single coherent ontology (Noy & Musen 2000; Stumme & Maedche 2001). In some cases, these are independently developed ontologies that model overlapping domains. In others, we merge two ontologies that evolved from a single base ontology. The £rst step in merging ontologies is to create a mapping between them. Once the mapping is given, the challenge of a merge algorithm is to create a *minimal* ontology that covers the given ones.

The creation of mappings will rarely be completely automated. However, automated tools can signi£cantly speed up the process by proposing plausible mappings. In large domains, while many mappings might be fairly obvious, some parts need expert intervention. There are several approaches to building such tools. The £rst (and more prevalent) uses a wide range of heuristics to generate mappings. The heuristics are often based on structure (e.g., if two classes match, then some of their sub-classes probably match), or based on naming (course and class may refer to the same concept). In some cases, domain-independent heuristics may be augmented by more speci£c heuristics for the particular representation language or application domain. A second approach is to learn mappings. In particular, manually provided mappings present examples for a learning algorithm that can generalize and suggest subsequent mappings.

Our ultimate goal is to build a system that uses a multitude of techniques to help users construct mappings. The system will combine different heuristics, hypotheses of learning modules and user feedback to guide the mapping process. Such a system needs a principled representation of mappings that it can manipulate in a well de£ned fashion. To £ll this need, in the next section we offer a framework in which mappings between models can be de£ned. The framework can be instantiated to a particular context by considering a speci£c language (or set of languages) relevant to that context. The following example illustrates the issues involved.

Example 1 *Figure 1 includes two different models of a domain of students. The £rst model,* MyUniv, *is in DAML+OIL (Horrocks, van Harmelen, & Patel-Schneider 2001) (i.e., a description logic). The second one,* YourUniv, *is a relational schema.*

The ontology MyUniv *includes the concepts* STUDENT *with subclasses* ARTS-STD *and* SCI-STD *and* COURSE *with*

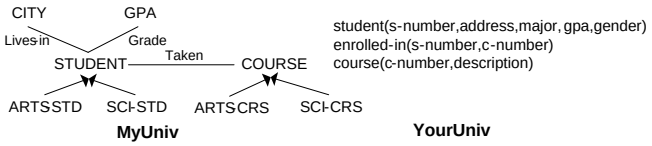

Figure 1: Models of the student domain

subclasses ARTS-CRS *and* SCI-CRS*. The binary relationship* Taken *represents the courses taken by students, and the relationships* Grade *and* Lives-in *represent properties of students.* Lives-in *is constrained to have the value "*myCity*".*

The schema YourUniv *includes the tables* student, course *and* enrolled-in*. In addition, the schema includes an integrity constraint specifying that the attribute* address *must contain the string "*yourCity*". Our goal is to specify a mapping between the two models.*

The £rst issue illustrated by this example is that we need to specify a mapping between models in different representation languages. The second issue is that not all concepts in one model exist directly in the other. For example, arts students in MyUniv are identi£ed by a concept, while in YourUniv they can only be identi£ed by a query (all students with major="arts"). A majority of works in the related literature have concentrated more on obtaining simple correspondences between concepts, and have ignored richer mappings such as this. Also note that some concepts in one model might not have corresponding ones in the other (gender of students in YourUniv).

A more subtle issue is that the mapping cannot simply equate MyUniv.STUDENT with YourUniv.student. The constraints on the addresses imply that these are disjoint sets (assuming that myCity and yourCity are distinct). The only way to relate the students in the different models is to de£ne a *third* model that represents a set of students from (at least) both cities, and then relate MyUniv and YourUniv to the third model. □

To summarize our discussion, we pose the following desiderata for representations of mappings:

Clear semantics: The meaning of mappings should be formally de£ned. The semantics will provide a basis for reasoning about mappings (e.g., determining whether two mappings are equivalent or if a certain mapping formula is entailed by a mapping), combining evidence to propose likely mappings, and learning mappings.

Accommodate incompleteness: Incompleteness can arise in two ways: (1) because of loss of information when the two models cover different domains, and (2) when a certain concept can be mapped between the models, but the mapping is unknown or hard to specify. An example of the latter would occur if the two universities had different grading systems, and there is no direct mapping between them. The key point here is that even if the mapping is incomplete, it still may suf£ce for the task at hand. Detecting this is one of the important issues of interest to us.

Allow heterogeneity: By nature, mappings between models will involve multiple representation languages. Therefore, a mapping language needs to be able to represent mappings between models in different languages. An alternative approach would be to £rst translate all the models into a common representation language, and then specify mappings as formulas in this language. However, it is often desirable to avoid the problems associated with designing a single representation language for all models (such as £nding a suitably rich language in which inference is still tractable).

3 Mappings: syntax and semantics

We now de£ne the basic terms we use for our framework. Unfortunately, the term "model" is overloaded. In our discussion we use the word *model* to denote a representation of a domain in a formal language, and use *logical model* to refer to an interpretation that satis£es all the formulas in a model. Our description considers mapping between a pair of models, but the framework can be extended in a straightforward way to multiple models.

Representation languages and models: We represent a domain of discourse using a set of expressions in some formal representation language. In this paper we consider languages with formal semantics (e.g., First-order logic, Description Logics, Horn rules, relational schemata, graph-structured data such as XML). We denote a language used in a representation by $\mathcal{L}$ possibly with a subscript. The set of expressions to represent a domain is called a *model*, and is denoted by T. We do not assume that the model is closed under logical deduction or even that reasoning in the representation language is decidable.

A model of a domain is often (but not always!) speci£ed in two parts, terminological and extensional. The £rst part identi£es the concepts in the domain and relations between them (or the set of relations in a relational schema). The second part populates the model with facts about speci£c individuals in the domain. The phrase *an extension of a model* T refers to a model in which we added an extensional component to the terminological component T. If T has a single component, then the phrase simply refers to T itself.

Expressions and formulas: A mapping between models consists of a set of relationships between expressions over the two models. An expression is built from the terms in the language and a set of operators. In order to be comparable, the expressions need to have compatible output types. For example, a SQL query produces a relation; a concept in a description logic represents a unary relation. An XML query language generates as output a nested XML document.

The expression language used in a mapping varies depending on the languages of the models being mapped. For our semantics to be well de£ned, we require the following: given a model T in a language $\mathcal{L}$ and an instance $\bar{a}$ of the data type of the output of an expression e, and an interpretation I of T, the question of whether I satis£es $e(\bar{a})$ (i.e., $I \models e(\bar{a})$) is well de£ned in $\mathcal{L}$ (and hence, $T \models e(\bar{a})$ is well de£ned as well). This essentially means that the set of outputs of an expression is well de£ned. Note that we don't

require the entailment to be decidable. Also note that the output of an expression can be a constant.

A formula relates expressions over two models. Given two models T_1 (in language $\mathcal{L}_1$) and T_2 (in $\mathcal{L}_2$), *a formula over T_1 and T_2* is of the form e_1 **op** e_2, where e_1 and e_2 are expressions, and the operator **op** is well de£ned w.r.t. the output types of e_1 and e_2. For example, if both expressions have relations as output types, then **op** can be $=, \subseteq$, and if they both output numeric constants, then **op** can be $\geq, \leq, =$. If e_1 outputs a constant and e_2 outputs a unary relation, then **op** can be $\in$. Note that given an interpretation I_1 (I_2) for T_1 (T_2), $\{I_1, I_2\} \models e_1$ **op** e_2 is well de£ned.

This ability of a formula to relate expressions of different languages implies a measure of compatibility between the languages, namely that some of their types are in common. This aspect of formulas is what enables us to de£ne mappings between models in different representation languages.

Syntax of mappings

We are now ready to de£ne the syntax of a mapping between models T_1 and T_2.

De£nition 1 *Let T_1 and T_2 be models in languages $\mathcal{L}_1$ and $\mathcal{L}_2$ respectively. A mapping between T_1 and T_2 may include a helper model T_3 (in language $\mathcal{L}_3$), and consists of a set of formulas each of which is over (T_1, T_2), (T_1, T_3) or (T_2, T_3).*

Example 2 *One possible mapping between* YourUniv *and* MyUniv *uses a helper model* Univ*, a relational schema with tables* Student, Course, Arts-Std, Sci-Std, Arts-Crs, *and* Sci-Crs. *The following are examples of the mapping formulae.*

Univ.Student(std,ad,gpa)$\supseteq$ MyUniv.STUDENT(std) $\wedge$
MyUniv.Lives-in(std,ad) $\wedge$ MyUniv.Grade(std,gpa)

Univ.Student(std,ad,gpa)$\supseteq$YourUniv.student(std,ad,x,gpa,y)

Univ.Arts-Std(std)$\supseteq$MyUniv.ARTS-STD(std)

Univ.Arts-Std(std)$\supseteq$YourUniv.student(std,x,"arts",y,z)

The £rst two formulae map students in the two universities to the single student concept in the helper model. The other two formulae map art students and art majors to a single table for arts students. Other mappings can be de£ned similarly.

Semantics of mappings

The semantics of a model constrains its possible interpretations by specifying which ones are logical models. The semantics of a mapping between two models is a constraint on the pairs of interpretations (or triplets, when a helper model is used), and therefore speci£es which pairs of interpretations can co-exist, given the mapping. Formally, satisfaction of a mapping is de£ned as follows:

De£nition 2 *Let M be a mapping between the models T_1 and T_2, and assume that M also involves a helper model T_3. Let I_i be an interpretation of an extension of T_i for $i = 1, 2, 3$. The triplet (I_1, I_2, I_3) satis£es the mapping M if the following hold:*

- *I_i is a logical model of T_i, for $i = 1, 2, 3$, and*
- *For each formula $e \in M$, $\{I_1, I_2, I_3\} \models e$.*

In practice, it is often useful to restrict mappings to be *conservative augmentation* of the models being mapped. Intuitively, a mapping is a conservative augmentation if it does not entail new facts *within* the model, given the mapping. Formally, M is a conservative extension of T_1 if whenever $I_1 \models T_1$, there exist I_2 and I_3 such that $(I_1, I_2, I_3) \models M$.

Note that in examples 1 and 2, if the mapping formulae were simple equalities between expressions in the two models, their semantics would not be well-de£ned. There can be no interpretations of the two models over a common domain of discourse (body of students) since they have con¤icting constraints on the city addresses of students. The use of the helper model enables a well-de£ned mapping.

There are a few points worth noting about our framework. First, the mapping formulae can be fairly expressive, and therefore it is possible to represent complex relationships between models. In particular, the expression language may even translate data elements in one model into schema elements in another (or vice versa), which is an important feature in practice. Second, though we have only emphasized mapping between concepts (or schema elements) here, the framework can also be used to map between data elements in the two models. Finally, our semantics are de£ned in terms of instances in the domain (interpretations map instances in the domain of discourse to elements in the models). However, that does not mean that we must have actual instances in the models; in fact, many ontologies do not have associated instances. Even in such cases, since these ontologies have well-de£ned semantics, there exists some implicit interpretation over their intended domains of discourse.

4 Important properties of mappings

In this section we identify several important properties of mappings that are worth studying. These properties are needed to decide whether a mapping is adequate for a particular task at hand. The £rst property is *query answerability*. A mapping between two models rarely maps all the concepts in one model to all concepts in the other. Instead, mappings typically lose some information and can be partial or incomplete. However, lossy mappings can be adequate for some tasks. Query answerability is a formalization of this property. The second property, *mapping inference*, provides a reasoning tool for determining whether mappings are equivalent. *Mapping composition* enables creating mappings between models that are related by intermediate models.

Query answerability: Mappings are typically required to enable a certain task. For example, in data integration a mapping is adequate if it enables the answering of queries over the mediated schema. For data migration, a mapping is adequate if we can populate the target schema, given an extension of the source schema. For ontology or schema merging, the merged ontology is adequate if we can use it to answer all the queries that were answerable in the original ontologies. Hence, it seems natural to consider a mapping adequate if it enables correct answering of a certain set of queries over the models at hand (note that the set of queries may be in£nite!). We de£ne the problem formally as fol-

lows. (The de£nition is written w.r.t. queries over T_1, but it can be de£ned similarly for queries over T_2).

De£nition 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 de£nition 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 α 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 α.*

Mapping Inference: A formal semantics for mappings enables a precise de£nition 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:

De£nition 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 de£ned as follows.

De£nition 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, satis£es $(I_0, I_1) \models M_{01}$ 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 speci£c 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:

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

where the p_i's are database relations, $\bar{X}_i$ and $\bar{X}$ are tuples of variables (and $\bar{X} \subseteq \bar{X}_1 \cup \ldots \cup \bar{X}_n$). The variables in $\bar{X}$ are called *distinguished* and the other variables are called *existential*. The distinguished variables are assumed to be universally quanti£ed, while the others are existentially quanti£ed. 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 de£ning 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 de£ned 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 £rst 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.* □

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). □

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.* □

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 de£ned using expressions in the language. The u_i's are de£ned 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 $v_1, \ldots, v_k$ (resp. $u_1, \ldots, u_k$). 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). □

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_1 *and* Univ_2*. Model* Univ_i *has tables* student_i, enrolled-in_i *and* course_i *(i=1,2). Consider the following view de£nitions,*

Major-Gender_i(std,mjr,gnd) :- student_i(std,x,mjr,y,gnd),
Some-Class_i(std) :- enrolled-in_i(std,x),
Some-Student_i(crs) :- enrolled-in_i(x,crs), *and*
Major_i(std,mjr) :- student_i(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_i *is rewritable in terms of* Major-Gender_i*, but* Some-Student_i *is not rewritable in terms of* course_i*,* Some-Class_i *and* Major-Gender_i*. It follows that,*

- *Given an extension of* Univ_2*, M enables answering of* Major_1*, but does not enable answering of* Some-Student_1.
- *M implies* $\text{Major}_1 = \text{Major}_2$*, but does not imply* $\text{Some-Student}_1 = \text{Some-Student}_2$.

Our £nal 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 £rst-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 suf£cient expressive power for reasoning about formulas with variables and quanti£cation.

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 ef£cient 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 ef£cient 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 de£ne 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 de£ned 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-de£ned 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. Ef£cient execution of information gathering plans. In *Proc. of the 15th Int. Joint Conf. on Arti£cial Intelligence (IJCAI)*.

Friedman, M.; Levy, A.; and Millstein, T. 1999. Navigational Plans for Data Integration. In *Proc. of the 16th National Conf. on Arti£cial 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 explixit knowledge through ontology mappings. In *Proc. of the 14th Int. FLAIRS conference*.

Lambrecht, E.; Kambhampati, S.; and Gnanaprakasam, S. 1999. Optimizing recursive information gathering plans. In *Proc. of the 16th Int. Joint Conf on Arti£cial 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 Arti£cial 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 Arti£cial 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 Arti£cial 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 Arti£cial 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 Arti£cial Intelligence (IJCAI)*.