Abstract. Schema mappings are high level specifications, usually given in some logical formalism, that describe the correspondence between schemas. In this paper we describe our research project about the development of a general framework for managing schema mappings in the context of data exchange both for the case of relational and XML databases. Our project includes the identification of operators between schema mappings that are relevant in practice, the development of formal models that allow us to compare these operators, and the study of their fundamental properties. In this paper we describe some of the results that we have obtained so far, and our plan for future work.

1 Introduction

A schema mapping is a specification that describes how data from a source schema is to be mapped to a target schema. Schema mappings have proved to be essential for data-interoperability tasks such as data exchange and data integration. The research on this area has mainly focused on performing these tasks. However, as Bernstein pointed out [3], many information-system problems involve not only the design and integration of complex application artifacts, but also their subsequent manipulation. Driven by this consideration, Bernstein proposed in [3] a conceptual framework for managing schema mappings. In this framework, schema mappings are specified in a logical language and high-level algebraic operators are used to manipulate them [3, 7, 12, 5, 4].

The creation of a mapping between two schemas usually means a considerable work by an expert who needs to know the semantics of the schema components. Only an expert can establish a meaningful high-level correspondence between those components. That is how schema mappings contain metadata that reflects, in some sense, the knowledge the expert has about the relationship between the schemas. This metadata could, at least in principle, be reused in other tasks beyond the interplay between the schemas for which the mapping was created.

An example of manipulation and reuse of schema mappings that is conceptually easy to understand, is the composition of mappings. Consider three independent schemas $A$, $B$, and $C$, and the schema mappings $M_{AB}$ and $M_{BC}$ that describe how data must be exchanged between $A$ and $B$, and between $B$ and $C$, respectively. Assume that a new application needs to exchange data, but this time between $A$ and $C$. To create a mapping between $A$ and $C$ from scratch could imply a considerable work since, among other requirements, the creator needs to know the meaning of every component

 Metadata Management in Data Exchange Systems

Jorge Pérez

Khipu South Andean Center for Database Research*
Pontificia Universidad Católica de Chile
Advisor: Marcelo Arenas

* http://www.khipu.cl
of both schemas and how these components are related. Then we would like to reuse the metadata in $M_{AB}$ and $M_{BC}$; we would like to use $M_{AB}$ and $M_{BC}$ in order to automatically build a mapping $M_{AC}$ between $A$ and $C$. Intuitively, $M_{AC}$ should be the result of composing $M_{AB}$ and $M_{BC}$. There are several questions that immediately arise. What is the semantics of composing schema mappings? What kind of schema mappings and what is the expressiveness needed to specify such composition? Is there an algorithm (ideally, an efficient algorithm) to compute the composition?

Besides the composition, several operations between schema mappings have been identified as important. Among them we have the inverse, the merge and the difference. For every one of these operations, the same questions arise [3, 5, 4]. Another important question that arises is whether there exist mapping specification formalisms closed under the application of some of these operators. For the composition of mappings between relational schemas, a big part of the above questions have been answered [10, 7]. Nevertheless, there are only initial studies about other operators, and there has been almost no progress about the study of operators beyond the relational model.

Nowadays there is a wide agreement that to bring data exchange to reality, it is imperative to provide a general framework to manipulate schema mappings and their metadata [9, 4]. The main general goal of this project is the development of a general framework for managing schema mapping in the context of data exchange both for the case of relational and XML databases. To this end, we plan to identify what operators between schema mappings are relevant in practice. We plan to develop formal mathematical models that allow us to compare these operators with respect to what kind of tasks they can solve, and to study in detail the fundamental properties of these operators developing algorithms to compute them for the relational and XML case.

In the rest of the paper we describe the results that we have achieved so far, the work in progress, and our plans for future work. Due to the lack of space and to favor a good exposition, most of the technical details (formal definitions, theorems and proofs) have been omitted. We refer the reader to [2, 1] for details about our achieved results.

2 Results

2.1 Inverse operator and data recovery in the relational model

The inverse of a schema mapping was one of the first operations detected as relevant in the data exchange context [11, 5]. Intuitively, if $M$ is a schema mapping that specifies how to exchange data from the source to the target schema, the inverse of $M$ should be a reverse schema mapping that specifies how to bring the exchanged data back to the source. In 2006, Fagin [5] introduces a first semantics for inverting schema mappings for a special case of relational mappings. In [5], Fagin focuses in schema mappings specified by source-to-target tuple generating dependencies (st-tgds), a fragment of first-order formulas of important practical interest [6]. The central idea in [5] is that a mapping $M'$ is considered an inverse of a mapping $M$, if the composition of $M$ with $M'$ results in the identity mapping. We call this notion Fagin-inverse from now on. Although intuitive, this definition has two important disadvantages: (1) it is too restrictive since it is rare that a schema mapping possesses a Fagin-inverse, and (2) it is not clearly applicable to other kind of mappings specified by formulas beyond st-tgds. In 2007, Fagin et al. [8] addressed the first of these issues introducing the notion of quasi-inverse of
a schema mapping as a relaxation of the Fagin-inverse. However, there still exist very simple mappings of practical interest that do not admit quasi-inverses. Among other results, in [8] the authors provide a necessary and sufficient condition for the existence of quasi-inverses, and they propose algorithms to compute quasi-inverses and inverses of schema mappings specified by st-tgds. Both algorithms run in exponential time and produce exponential-size outputs. Another important result in [8], is the identification of the language needed to specify inverses and quasi-inverses. They show that in order to specify inverses and quasi-inverses of mappings given by st-tgds, one needs a logical language that is strictly more expressive than st-tgds. Although [8] presents important results that sustained the quasi-inverse as a good notion, it is not clear whether this notion is applicable beyond mappings specified by st-tgds.

One of the main results that we have obtained so far, is the proposal of a new definition for the inverse operator of schema mappings and a thorough study of its fundamental properties. We have included these results in the paper “The Recovery of a Schema Mapping: Bringing the Exchanged Data Back” [2]. In [2] we introduce the notion of recovery of a schema mapping: given a mapping \( M \), a recovery of \( M \) is a mapping that, intuitively, allows one to recover sound data. Being a recovery is a sound but mild requirement, then we propose a criteria to compare alternative recoveries. Thus, in [2] we introduce an order relation on recoveries which allows us to choose the more informative among all the recoveries of a mapping. We call this mapping the maximum recovery. In [2], we study in detail the notion of maximum recovery as a natural semantics for the inverse operator, and we show that it outperforms the notions of Fagin-inverse and quasi-inverse in the above mentioned issues. In particular, we show that:

1. every mapping specified by st-tgds has a maximum recovery,
2. if a mapping \( M \) has a Fagin-inverse, then \( M' \) is a Fagin-inverse of \( M \) if and only if \( M' \) is a maximum recovery of \( M \), and
3. the notion of maximum recovery is applicable to any kind of mapping beyond st-tgds and also beyond the relational model. Furthermore, in [2] we provide a necessary and sufficient condition for the existence of a maximum recovery, we study in detail the language needed to specify maximum recoveries, and we provide an algorithm to compute the maximum recovery of mappings specified by an extension of st-tgds. In general, our algorithm runs in exponential time, but it is worth noticing that our algorithm works in polynomial time for a subclass of mappings of practical interest (for this class of mappings the algorithms for Fagin-inverses and quasi-inverses proposed in [8] work in exponential time).

2.2 Inverse and composition together

In the framework proposed by Bernstein [3] schema mappings are usually specified in a logical language, and high-level algebraic operators are used to manipulate them [3, 4]. In this algebraic context, a natural question is whether a logical language is closed under the application of some operator; given schema mappings specified in a language \( L \) and an algebraic operator, can the result of the operator be also specified in \( L \)? Furthermore, complex transformations of schema mappings can be obtained by combining several operators. Thus, one may wonder whether a closure property holds for a set of operators. Such a closure property would ensure that the output of some operator can be used as the input for subsequent operators. This has been raised as a “prominent issue” [9] in metadata management.
As we have mentioned, two of the most fundamental operators for metadata management are the **composition** and **inversion** of schema mappings. Fagin et al. [7] provide a comprehensive view of the composition, giving a natural semantics and studying several theoretical issues about this operator. In particular, they prove that there exist languages that are closed under composition. However, the study of closure properties for the inverse operator remains an almost unexplored field. To the best of our knowledge, there is no research program towards finding logical languages capable of expressing complex sequences of applications of both operators.

As part of our project we have obtained some preliminary results towards unifying inversion and composition of schema mappings, and in particular, towards finding schema-mappings specification languages that are closed under inversion and composition. Notice that this goal amounts to (1) first choose a particular semantics for the inverse and composition operator, and then (2) prove that under these semantics, there exists a mapping language $L$ that is closed under the application of both operators. As a desiderata, we would like to have natural and useful semantics, and a mapping-specification language expressive enough to contain the most common mappings used in practice, namely, mappings given by the class of st-tgds. It is important to notice that the notions of Fagin-inverse and quasi-inverse could not meet our requirements, as there exist mappings specified by st-tgds that admit neither Fagin-inverses nor quasi-inverses.

In [1] we propose a general framework that provides a range of natural semantics for the inverse operator. These new inverse notions are based on the idea of recovering information with respect to a query language. More specifically, let $C$ be a query language, we introduce the notion of $C$-maximum recovery that intuitively is a mapping that recovers the maximum amount of information that is accessible by using queries in $C$. This framework is powerful enough to capture the previous notions of inverse as mappings that recovers the maximum amount of information for specific choices of the query language $C$. This new parameter $C$ allows us to obtain different natural semantics. For example, if in some application the information is always accessed by using conjunctive queries (CQ), then we can choose the CQ-maximum recovery as our choice for the semantics of the inverse operator.

In [1] we show that there exists a mapping specification language that is closed under both CQ-maximum recovery and the notion of composition defined in [7]. Unfortunately this language does not meet our requirements since it does not contain the language of st-tgds. Thus, as a final contribution in [1] we explore the possibility of having a mapping language that contains the language of st-tgds and is closed under both inversion and composition. We formally prove in [1] that to reach this goal one cannot use the notion of composition introduced in [7]. Thus we use the notion of CQ-composition introduced in [10] and give formal evidence that it is the right notion to obtain a mapping language closed under both composition and inversion. In fact, we propose a mapping language with second order existential quantification based on the language of SO-tgds proposed in [7]. This new mapping language has the following good properties: (1) it contains the class of st-tgds, (2) it is closed under CQ-composition, and (3) every mapping specified in the language has a CQ-maximum recovery. Nevertheless, in the current state of our research we do not know whether this new language is closed under CQ-maximum recovery. We plan to attack this problem in the future.
3 Future work and concluding remarks

As part of our future work we would like to identify what other operators, apart from composition and inversion, are used in practice, starting from those identified in [12, 11, 4]. We would like to provide precise semantics for these operators, study their fundamental properties and give efficient algorithms for computing them. We also plan to develop some formal tools in order to determine the expressiveness of sets of schema-mapping operators. We would like to be able to answer questions like, can some operation between mappings be obtained by successively applying other operations? Special emphasis will be put in finding specification languages closed under some of these operations. In particular, we would like to answer question like, is there a specification language that is closed under a set of these operations? under which conditions a set of operators can be considered complete for certain tasks?

Another important line of research in our project is to extend our results for the case of XML databases. There is no work that systematically addresses the topic of defining and studying schema-mapping operators in the XML context. We plan to start studying inversion and composition for schema mappings for the XML case, giving a rigorous semantics and studying its main properties. In particular, we would like to develop efficient algorithms for computing these operators. We believe that, as for the relational case, the study of schema mapping operations will also play a fundamental role in the theory and practice of metadata management for XML data exchange.

Acknowledgements. Most of the results that we have mentioned in this paper are product of a fruitful collaboration with Cristian Riveros and Juan L. Reutter, both members of our Khipu group in Chile.

References