Maintaining Semantic Mappings between Database Schemas and Ontologies

Yuan An[1] and Thodoros Topaloglou[2]

[1] College of Information Science and Technology
Drexel University, USA
yan@ischool.drexel.edu
[2] Department of Mechanical and Industrial Engineering
University of Toronto, Canada
thodoros@mie.utoronto.ca

Abstract. There is a growing need to define a semantic mapping from a database schema to an ontology. Such a mapping is an integral part of the data integration systems that use an ontology as a unified global view. However, both ontologies and database schemas evolve over time in order to accommodate updated information needs. Once the ontology and the database schema associated with a semantic mapping evolved, it is necessary and important to maintain the validity of the semantic mapping to reflect the new semantics in the ontology and the schema. In this paper, we propose a formulation of the mapping maintenance problem and outline a possible solution using illustrative examples. The main points of this paper are: (1) to differentiate the semantic mapping maintenance problem from the schema mapping adaptation problem which only adapts mappings when schemas change; (2) to develop an approach for specifying the validity of a semantic mapping in terms of two-way legal instances translation between two models; (3) to explore the approach of using simple correspondences to capture changes to ontologies/schemas; and (4) to sketch a solution using examples.

1 Introduction

A semantic mapping from a database schema to an ontology defines a semantic relationship between the schema and the ontology. For example, a many-to-many relationship between a concept C_1 and a concept C_2 in an ontology may be mapped to relational tables storing attributes of C_1 and C_2 and a linking table that maintains the association of the identifiers[3] of C_1 and C_2. Such a semantic mapping can be expressed in a declarative language that encodes the formal semantics of the schemas. In recent years, we are witnessing a growing demand for defining semantic mappings from database schemas to ontologies. For example, semantic mappings are integral part of ontology-based information integration systems [8, 13], and data integration efforts in the context of the

[3] We assume that a subset of attributes of a concept in an ontology acts as identifier of the concept

Semantic Web. Furthermore, a recent work [1] suggests that the semantics of database schemas expressed in terms of semantic mappings from schemas to conceptual models/ontologies provide opportunities to improve the capabilities of traditional schema mapping tools, even when different database schemas are associated with different conceptual models or ontologies.

However, both ontologies and schemas change over time in order to accommodate new information needs. Such change may cause an existing semantic mapping *invalid*. Therefore, once a semantic mapping from a schema to an ontology has been created, it is important and necessary to automatically, at least to some extent, maintain the validity of the semantic relationship when the schema and ontology evolve. We call this process *maintaining semantic mappings under evolution* or *mapping maintenance* for short. A typical solution to the mapping maintenance problem is to regenerate the semantic mapping between the evolved ontology and schema. The problem of the mapping regeneration solution is that the solution can be costly in terms of human effort and expertise. The reason is that semantic mapping creation is a demanding task which requires huge amount of human effort, because both the schema and the ontology that are related by a semantic mapping are complex artifacts which may contain hundreds and thousands modeling constructs. There are existing methods and tools, e.g., [3, 2], for creating semantic mappings from database schemas to ontologies. But almost all the current tools are semi-automatic and interactive, requiring humans involved in the process. A better solution to the mapping maintenance problem is to incrementally update the existing semantic mapping to reflect *changes* in the ontology or schema. In this paper, we report on our preliminary study on the problem of incrementally maintaining a special type of semantic mapping, which, in a local-as-view fashion, relates a single atom (e.g., a table) in a schema with a conjunctive formula encoding a substructure in an ontology. The formalism is presented in Section 3.

The aims of the maintenance are two-fold: first, to preserve the semantic relationship between the schema and the ontology when the schema and ontology are modified; second, to reuse the existing semantic mapping as much as possible. A similar problem has been studied for adapting schema mappings under schema evolution. Two possible approaches are proposed in the literature: a schema change approach (SCA) [15] and a mapping composition approach (MCA) [16]. Both solutions focus on reusing the semantics encoded in previous mappings for merely adapting the mappings. Schemas are not updated accordingly. In our situation, adapting the ontology/schema associated with a semantic mapping along with the mapping will be essential for achieving desired goals. Consider a very simple case. Suppose the semantics of a relational database schema is expressed in terms of an ontology. If the database engineer wants to modify the schema by adding a new column to a table representing a concept in the ontology, it may be desirable to add a new attribute to the concept in the ontology in order to maintain the semantic relationship that covers the new element of the schema. Maximizing the coverage over schema will be one of the desired goals for maintaining a semantic mapping.

Although mapping maintenance is important and necessary for many applications, solutions to the problem are rare. This is due to many challenges involved, including: how to define validity/consistency of mapping and detect inconsistency of a mapping; what is a right mapping language; how to capture changes to ontologies and database schemas; how to devise a plan for updating mappings according to the intent and expectation of the user; and what are the principles for a systematic maintenance solution.

In this paper, we formulate the maintenance problem. We propose a specification for the validity of a semantic mapping. Subsequently, we describe the desired goals for maintaining semantic mappings between database schemas and ontologies, and we outline our solution for addressing the problem using a comprehensive set of examples.

The rest of the paper is organized as follows. Section 2 discusses related work. Section 3 introduces our formalism for a semantic mapping from a schema to an ontology. Section 4 characterizes schema and ontology evolution. Section 5 outlines a solution to the problem of semantic mapping maintenance. Finally, Section 6 concludes this paper.

2 Related Work

The directly related work is the study on schema mapping adaptation [15, 16]. The goal of schema mapping adaptation is to automatically update a schema mapping by reusing the semantics of the original mapping when the associated schemas change. Yu & Popa [16] explore the schema mapping composition approach. Schema evolutions are captured by formal and accurate schema mappings, and schema adaptation is achieved by composing the evolution mapping with the original mapping. On the other hand, the schema change approach in [15] proposed by Velegrakis et al. incrementally changes mappings each time a primitive change occurs in the source or target schemas. Both solutions focus on reusing the semantics encoded in existing mappings for merely adapting the mappings without considering the synchronization between schemas. This is due to the nature of their problems where schema mappings are primarily used for *data exchange* [10], i.e., translating a data instance under a source schema to a data instance under a target schema. If a schema mapping connecting two schemas which are semantically inconsistent, then the data exchange process simply does not always produce a target instance. Our approach is different from these solutions in that we aim to maintain the *semantic validity* of semantic mappings through incremental updates on the mappings as well as associated ontologies/schemas.

Other related work includes schema evolution in object-oriented databases (OODB). The problem of schema evolution in OODB is to maintain the consistency of an OODB when its schema is modified. The challenges are to update the database efficiently and minimize information loss. A variety of solutions, e.g., [6, 5, 9, 12], have been proposed in the literature. Our problem is different from the schema evolution problem in OODB in that we aim at the semantic consistency

between a schema and an ontology. However, we can draw some insights from the extensive study of the schema evolution problem in OODB. In AutoMed [7, 11], schema evolution and integration are combined in one unified framework. Source schemas are integrated into a global schema by applying a sequence of primitive transformations to them. The same set of primitive transformations can be used to specify the evolution of a source schema into a new schema. In our approach, we do not ask users to specify a sequence of transformations.

Another mapping maintenance problem studied in [14] mainly focus on detecting inconsistency of simple correspondences between schema elements when schemas evolve. This problem is complementary to the problem we consider here.

3 Semantic Mappings between Ontologies and Schemas

3.1 Relational Schemas and Ontologies

Here we focus on relational schemas described in the relational model. The basic data representation construct of the relational model is relation, which consists of a set of tuples. The schema of a relation or a *table* specifies the name of the relation, the name of each column (or attribute or field), and the type of each column. Furthermore, we can make the description of the collection of data more precise by specifying *integrity constraints*, which are conditions that the tuples in a table must satisfy. Here, we consider *key* and *foreign key* (abbreviated as *f.k.* henceforth) constraints. A key in a table is a subset of the columns of the table that uniquely identifies a tuple. A f.k. in a table T is a set of columns F that *references* the key of another table T' and imposes a constraint that the projection of T on F is a subset of the projection of T' on the key of T'. A relational schema thus consists of a set of relational tables and a set of key and f.k. constraints. Formally, we use the notation $T(\underline{k_1, k_2, ..., k_n}, y_1, y_2, ..., y_m)$ to represent a relational table T with key $K = (k_1, k_2, ..., k_n)$.

An ontology describes a subject matter in terms of concepts, relationships, and attributes. In this study, we do not restrict ourselves to any particular language for describing ontologies. Instead, we use a generic conceptual modeling language (CML) which has the following features. The language allows the representation of *classes/concepts/entities* (unary predicates over individuals), *object properties/relationships* (binary predicates relating individuals), and *datatype properties/attributes* (binary predicates relating individuals with values such as integers and strings); attributes are single valued in this paper. Concepts are organized in the familiar ISA hierarchy. Relationships and their inverses (which are always present) are subject to constraints such as specification of domain and range, plus cardinality constraints of the form $k..l$; if the lower bound, $k = 1$, the relationship is called , *total*, if the upper bound, $l = 1$, the relationship is called *functional*. In addition, a subset of attributes of a concept is specified as the identifier of the concept. As in the Entity-Relationship model, a strong entity has a global identifier, while a weak entity is identified by an identifying relationship plus a local identifier. An ontology thus contains a set of concepts,

relationships, and attributes as well as a set of identification and cardinality constraints.

We can represent a given ontology using a labeled directed graph, called an *ontology graph*. We construct the ontology graph from an ontology by considering concepts as nodes and relationships as edges. A many-to-many relationship p between concepts C and D will be written in text as C ---p--- D. It will be important for our approach to distinguish *functional edges* – ones with upper bound cardinality of 1, and their composition: *functional paths*. If the relationship p is functional from C to D, we write C ---p->-- D.

3.2 Semantic Mappings between Ontologies and Schemas

In this study, we use the semantic mapping notion that is proposed in [4] which relates tables in a schema with formulas over an ontology. The formula over an ontology is in a subset of conjunctive formulas and encodes a subtree in the ontology graph. In particular, we assume that the semantics of a table is represented by a subtree (subgraphs can be transformed into subtrees by duplicating nodes in cycles). We call such a subtree a *semantic tree (or s-tree)*, where columns of the table associate uniquely with attribute of the concepts in the s-tree. This assumption also corresponds to the standard database design practice where each table is derived from a structure, usually, a subtree, in a conceptual model. After encoding s-trees in conjunctive formulas by using unary predicates for concepts, binary predicates for attributes, and binary predicates for binary relationships (see [4]), we can represent a semantic mapping between a relational schema and an ontology using a set of formula of the form $T(X) \leftrightarrow \Phi(X, Y)$, where T is a table with columns X and Φ is a conjunctive formula over predicates representing an s-tree. X and Y are quantified variables as specified later.

Example 1 Gene expression databases maintain information on genes, biological samples and measurements on genes over samples. Biological sample is a central concept being modeled in a gene expression database. To record information about a sample which can be a tissue, cell, or RNA material that originates from a donor of a given species, one needs to create a sub-schema that we will refer to as the sample database (SDB). Suppose that a SDB contains a table

sample(sample_ID, species, organ, pathology,..., donor_ID),

where the underlined column sample_ID is the key of the table and donor_ID is a foreign key to a table called donor.

The semantics of the sample table can be expressed in terms of an s-tree in an ontology as shown in Figure 1 which is described in the UML notation, where identifier of a concept is indicated by the keyword key. The s-tree contains two concepts, SAMPLE and PERSON, and a relationship, originates, between the two concepts.

Graphically, we use dashed double-arrows to indicate the correspondences between columns of the relational table and attributes of concepts in the ontology. The correspondences plus the s-tree gives rise to a semantics of the table. Furthermore, the semantics of the table is expressed in the following formula

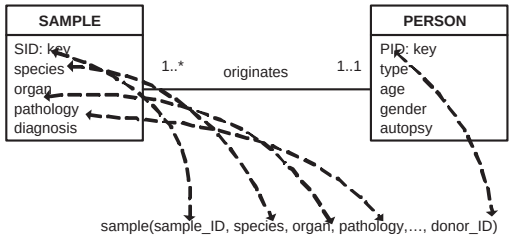

Fig. 1. The sample table and Its Semantics

$\mathsf{sample}(sample\_ID,\ species,...,\ donor\_ID) \leftrightarrow$
$\mathsf{SAMPLE}(x_1),\ \mathsf{SID}(x_1,\ sample\_ID),\ \mathsf{species}(x_1,\ species),$
$...,\ \mathsf{PERSON}(x_2),\ \mathsf{originates}(x_1,\ x_2),\ \mathsf{PID}(x_2,\ donor\_ID).$ ■

Valid Semantic Mappings Given a semantic mapping formula $T(X) \leftrightarrow \Phi(X,Y)$ which relates a table $T(X)$ in a schema with a conjunctive formula $\Phi(X,Y)$ encoding an s-tree $\mathcal{G}$ in an ontology. We say that $T(X) \leftrightarrow \Phi(X,Y)$ is valid if the table and the s-tree $\mathcal{G}$ are "semantically compatible". More specifically, we define the validity by using two logical formulas $\forall X(T(X) \rightarrow \exists Y.\Phi(X,Y))$ and $\forall X,Y(\Phi(X,Y) \rightarrow T(X))$, plus the key and f.k. constraints of the schema and the identification and cardinality constraints of the ontology.

The formula $\forall X(T(X) \rightarrow \exists Y.\Phi(X,Y))$ can be considered as the formal specification for translating instances from the table to the s-tree, and the formula $\forall X,Y(\Phi(X,Y) \rightarrow T(X))$ can be considered as the formal specification for translating instances from the s-tree to the table. Let Σ_T be the set of key and f.k. constraints of T. Let Σ_S be the set of identification and cardinality constraints of S. An instance I of T is a legal instance if I satisfies all constraints in Σ_T. An instance J of S is a legal instance if J satisfies all constraints in Σ_S. For each legal instance I of T, we can generate an instance J' of S through $\forall X(T(X) \rightarrow \exists Y.\Phi(X,Y))$ by instantiating Y. For each legal instance J of S, we can generate an instance I' of T through $\forall X,Y(\Phi(X,Y) \rightarrow T(X))$. If both J' and I' are legal instances of S and T, respectively, then we say that T and S are "semantically compatible."

We now define a valid semantic mapping using semantically compatible instances. Specifically, a formula $T(X) \leftrightarrow \Phi(X,Y)$ relating a table $T(X)$ in a schema with a conjunctive formula $\Phi(X,Y)$ encoding an s-tree $\mathcal{G}$ in an ontology is a **valid** semantic mapping formula, if and only if for each legal instance of T, we can generate a *legal* instance of $\mathcal{G}$ through $\forall X(T(X) \rightarrow \exists Y.\Phi(X,Y))$, and for each legal instance of $\mathcal{G}$, we can generate a *legal* instance of T through $\forall X.Y(\Phi(X,Y) \rightarrow T(X))$.

Having the definition about a valid semantic mapping formula, we attempt to (semi-)automatically maintain the validity of each formula when the schema and the ontology related by the formula evolve. In the next section, we begin with a characterization of possible changes in schemas and ontologies.

4 Evolution of Schemas and Ontologies

Changes to schemas and ontologies can be characterized by mappings [16] or by sequences of evolution primitives [15, 5]. Consider a mapping $\mathcal{M}$ between two schema $\mathcal{S}_1$ and $\mathcal{S}_2$. If one of the schemas, e.g., $\mathcal{S}_1$, evolves to a new schema $\mathcal{S'}_1$, the mapping composition approach (MCA) for schema mapping adaptation will compose the mapping $\mathcal{M}$ with an evolution mapping $\mathcal{M'}$ between $\mathcal{S'}_1$ and $\mathcal{S}_1$ to derive a new mapping between $\mathcal{S'}_1$ and $\mathcal{S}_2$, while the schema change approach (SCA) will look at a sequence of primitive changes for adapting $\mathcal{M}$.

Both MCA and SCA approaches are inadequate in dealing with the problem of maintaining a semantic mapping $\mathcal{M}$ between a schema $\mathcal{S}$ and an ontology $\mathcal{O}$. First of all, neither MCA nor SCA approach attempts to maintain the validity of the semantic mapping. For example, if the key information of a table in $\mathcal{S}$ changes, the mapping $\mathcal{M}$ may not change, but the ontology $\mathcal{O}$ may need to be modified in order to keep $\mathcal{M}$ as a valid semantic mapping. However, current MCA and SCA approaches only consider mapping adaptation. Second, the MCA does not capture the changes of adding elements to schemas. If an element is added, it will leave the existing mapping unchanged. Third, it is not guaranteed that the current SCA approach would maintain the semantics of the existing mapping by using a sequence of primitive changes, as the set of primitive changes for schema evolution may not cover some changes encountered in ontology evolution. For example, one of primitive changes that may happen in ontology evolution but are not captured by the set of primitive changes for schema evolution is adding/deleting an ISA relationship between two concepts.

In this study, we use a set of correspondences to link elements of the previous schema/ontology to elements of the new schema/ontology when a schema/ontology changes. We then analyze the existing semantic mapping and the semantics in the new schema/ontology. Through the set of correspondences, we will then (semi-)automatically adapt both the semantic mapping and the schema/ontology to maintain the validity of the semantic mapping.

Example 2 Figure 2 depicts on the left an old ontology $\mathcal{O}_1$ consisting of a single concept BIOSAMPLE. On the right is the new ontology $\mathcal{O}'_1$ which was evolved from $\mathcal{O}_1$ by adding a new concept TISSUE. The dashed double-arrows from attributes of the BIOSAMPLE concept in $\mathcal{O}_1$ to attributes of the BIOSAMPLE and TISSUE concepts in $\mathcal{O}'_1$ capture the relationship between the old ontology and the new ontology. ■

Changes to schemas and ontologies can be classified along two orthogonal axes. First, on the *action* axis, changes can be classified into (1) changes for adding/deleting elements; (2) changes for merging/splitting elements; (3) changes for moving/copying elements; (4) changes for renaming elements; and (5) changes for modifying constraints. Second, on the *effect* axis, changes can be classified into (i) changes that cause mapping modification; (ii) changes that cause the related schema (or ontology) modification; and (iii) changes that cause both mapping and the related schema (or ontology) modification. The classification along the *effect* axis is mainly concerned with maintaining the validity of a semantic mapping as specified in Section 3. In the next section, we discuss our

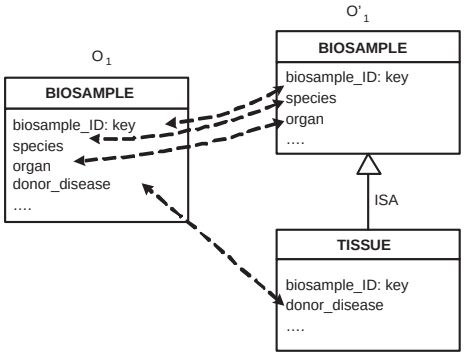

Fig. 2. The Correspondences between Old and New Ontologies

solution to the maintenance problem by associating changes classified along the *action* axis with changes classified along the *effect* axis.

5 Maintaining the Semantic Mappings

We outline an algorithm for maintaining semantic mappings between relational schemas and ontologies. The input to the algorithm consists of a relational schema $\mathcal{S}$, an ontology $\mathcal{O}$, an existing valid semantic mapping $\mathcal{M}$ between $\mathcal{S}$ and $\mathcal{O}$, a new schema $\mathcal{S}'$ (or ontology $\mathcal{O}'$) evolved from $\mathcal{S}$ (or $\mathcal{O}$), and a set of element correspondences $\mathcal{M}'$ between $\mathcal{S}$ and $\mathcal{S}'$ (or between $\mathcal{O}$ and $\mathcal{O}'$). The output of the algorithm is an ontology $\mathcal{O}''$ (or a schema $\mathcal{S}''$) and the semantic mapping $\mathcal{M}''$ between $\mathcal{S}'$ and $\mathcal{O}''$ (or between $\mathcal{O}'$ and $\mathcal{S}''$). The ontology $\mathcal{O}''$ may be just the original ontology $\mathcal{O}$, if the schema $\mathcal{S}$ evolved and only the semantic mapping gets adapted without any changes in the original ontology. Similarly, the schema $\mathcal{S}''$ may be just the original schema $\mathcal{S}$, if what evolved was the ontology $\mathcal{O}$ and there are no needs to change the original schema in order to adapt the semantic mapping.

Figure 3 graphically describes the semantic mapping maintenance settings. Figure 3 (a) shows the situation where the schema $\mathcal{S}$ evolved to a new schema $\mathcal{S}'$. $\mathcal{M}$ is the existing semantic mapping; $\mathcal{M}'$ is the set of correspondences from elements of $\mathcal{S}'$ to elements of $\mathcal{S}$. The aim of the mapping maintenance is to adapt $\mathcal{M}$ to a new semantic mapping $\mathcal{M}''$ between $\mathcal{S}'$ and $\mathcal{O}$ (or $\mathcal{O}''$ if the original ontology needs to be modified.) Likewise, Figure 3 (b) show the situation when the ontology $\mathcal{O}$ evolved to $\mathcal{O}'$, $\mathcal{M}$ needs to be adapted to $\mathcal{M}''$ between $\mathcal{S}$ (or $\mathcal{S}''$) and $\mathcal{O}'$.

The maintenance algorithm is based on the knowledge about the existing semantic mapping and the analysis of the semantics in the changes to the schema/ontology. We first explore the knowledge encoded in a semantic mapping as studied in the previous work for discovering semantic mappings from schemas

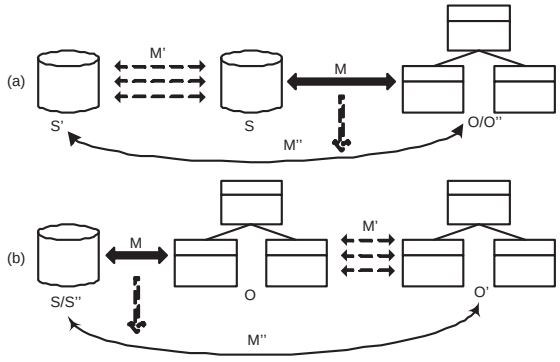

Fig. 3. Maintenance of Semantic Mapping

to ontologies [4]. Then we illustrate the algorithm by analyzing the semantics in changes using means of examples.

A semantic mapping formula $T(X) \leftrightarrow \Phi(X, Y)$ associates a table $T(X)$ with an s-tree in an ontology. There is additional knowledge about the association/relationship [4]. Specifically, an s-tree can be decomposed into several skeleton trees: a skeleton tree corresponding to the key of the table, skeleton trees corresponding to f.k.s of the table, and skeleton trees corresponding to the rest of the columns of the table. Each skeleton tree has an anchor concept which is the root of the skeleton tree. To satisfy the semantics of the key in a table, the s-tree is connected by functional paths from the anchor of the key skeleton tree to the anchors of f.k. skeleton trees and other skeleton trees.

Example 3 Figure 4 shows a table sample(sid,tid,donor) storing the information about a sample, where sid is the sample identifier, tid is the identifier of the test that screens the sample, and donor is the identifier of the person donating the sample. The concept SAMPLE is modeled as a weak entity owned by the TEST concept. Therefore, the key of the sample table is the combination of the key of

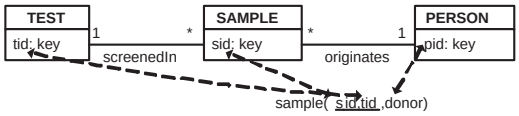

Fig. 4. Skeleton Trees in a Semantic Mapping

a table for the TEST concept and the local identifier sid. In addition, the donor column is a f.k. referencing the key of a table for the PERSON concept.

The semantics of the sample table is represented in terms of the s-tree above it in Figure 4. This s-tree consists of the skeleton tree SAMPLE ---screenedIn->--

TEST for the key of the sample table and the skeleton tree PERSON for the f.k. of the sample table. The anchor of the key skeleton tree is the concept SAMPLE, while the anchor of the f.k. skeleton tree is the concept PERSON. The s-tree is connected by a functional edge originates from the anchor SAMPLE to the anchor PERSON. ■

Each s-tree in a semantic mapping that consists of a key skeleton tree corresponding to the key of the table, skeleton trees corresponding to f.k.s of the table, skeleton trees corresponding to other columns of the table, and functional paths from the anchor of the key skeleton tree to anchors of other skeleton trees. To maintain a semantic mapping when the schema/ontology changes, we aim at maintaining the s-tree associated with the table. Our goals for maintaining semantic mappings are as follows.

Goal 1 *For a valid semantic mapping $\mathcal{M}$ between a schema $\mathcal{S}$ and an ontology $\mathcal{O}$, if $\mathcal{M}$ has been adapted to $\mathcal{M}''$ after some changes to the schema/ontology, then each mapping formula $m \in \mathcal{M}''$ must be a valid semantic mapping formula as specified in Section 3.*

Goal 2 *For a valid semantic mapping $\mathcal{M}$ between a schema $\mathcal{S}$ and an ontology $\mathcal{O}$, if $\mathcal{M}$ has been adapted to $\mathcal{M}''$ after some changes to the schema/ontology, then for each element $e \in \mathcal{S}$ that was covered by $\mathcal{M}$ (i.e., e was referred to by some mapping formulas in $\mathcal{M}$), e is covered by $\mathcal{M}''$ if e was not deleted from $\mathcal{S}$, and for each new element e' added to $\mathcal{S}$, e' is also covered by $\mathcal{M}''$.*

The first goal specifies the fundamental requirement for semantic mapping maintenance, that is, to maintain the validity of a semantic mapping according to the definition. The second goal requires that the semantic mapping after adapted should cover as much the remaining schema as covered by the existing semantic mapping and cover any newly added elements. The second goal comes from our intention of using semantic mappings for expressing semantics for database schemas. That is, for a database schema, we do not want to lose semantic information expressed in terms of the semantic mapping from the schema to an ontology after the semantic mapping is adapted due to changes to the schema/ontology.

The following examples outline the mapping maintenance algorithm in an intuitive way. The complete algorithm will be available in a full paper. At the present, the maintenance algorithm focuses on a pair of a schema and an ontology that are related by a semantic mapping.

Example 4 The following semantic mapping formula relates a relational table sample(sid,donor) with an s-tree in an ontology as shown in Figure 5 (a):

sample(sid, $donor$) $\leftrightarrow$
SAMPLE(x_1), sid(x_1, sid), PERSON(x_2),
originates(x_1, x_2), pid(x_2, $donor$).

First, we consider changes that add new column(s) to the relational table. (1) Add a column that is neither part of the key nor a f.k.. For example, a new column species was added to the sample table. In this case, the algorithm will suggest to add a new attribute to the anchor of the skeleton tree corresponding

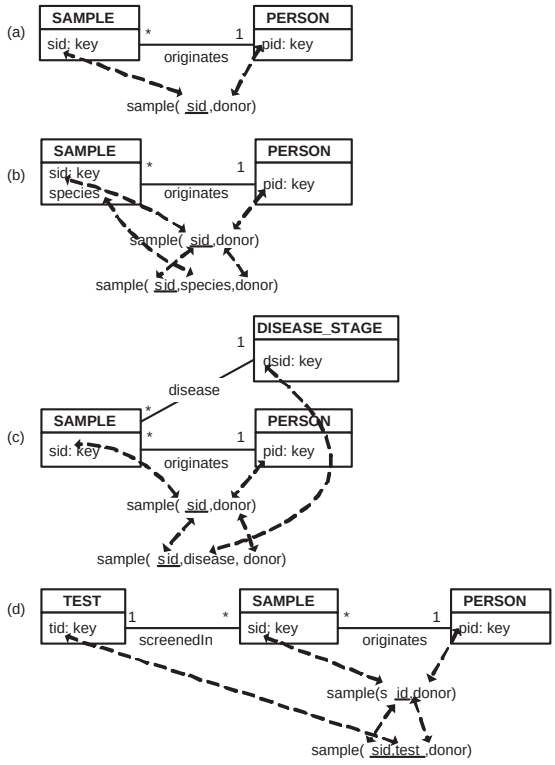

Fig. 5. Add Element to Schema

to the key as shown in Figure 5 (b) and update the semantic mapping formula to:

sample(sid, $species$, $donor$) $\leftrightarrow$
SAMPLE(x_1), sid(x_1, sid), PERSON(x_2),
species(x_1, $species$), originates(x_1, x_2), pid(x_2, $donor$).

(2) Add a column that is a f.k.. For example, a new column disease was added to the table sample where disease is a f.k. referencing to the key of a table T' for the concept DISEASE_STAGE. In this case, the algorithm finds a functional path from the anchor of the key skeleton tree for the key of the table sample to the anchor of the skeleton tree for the key of table T' as shown in Figure 5 (c), and updates the semantic mapping as the following candidate formula:

sample(sid, $disease$, $donor$) $\leftrightarrow$
SAMPLE(x_1), sid(x_1, sid), PERSON(x_2),
DISEASE_STAGE(x_3), disease(x_1, x_3), dsid(x_3, $disease$),
originates(x_1, x_2), pid(x_2, $donor$).

Note that there may be multiple functional paths connecting the anchor SAMPLE to the anchor DISEASE_STAGE, so the user will examine the candidate formulas

to choose the expected one.
(3) Add a column that becomes part of the key of the table. For example, a new column test was added to the table sample. If test is not a f.k., then the algorithm suggests to add an attribute as part of the identifier of the anchor of the skeleton tree for the key of the table. If test is a f.k., then the algorithms recomputes the skeleton tree for the key of the table as shown in Figure 5 (d), and suggests to update the semantic mapping as the following candidate formula:

$\mathsf{sample}(sid, test, donor) \leftrightarrow$
$\mathsf{SAMPLE}(x_1)$, $\mathsf{sid}(x_1, sid)$, $\mathsf{PERSON}(x_2)$,
$\mathsf{TEST}(x_3)$, $\mathsf{screenedIn}(x_1, x_3)$, $\mathsf{tid}(x_3, test)$,
$\mathsf{originates}(x_1, x_2)$, $\mathsf{pid}(x_2, donor)$.

As in case (2), the user needs to examine all candidate formulas.

Let us now consider changes that add new element(s) to the ontology. The following changes do not affect the semantic mapping: adding a new attribute which does not become part of the identifier of concepts in the s-tree, adding a new concept, and adding a new ordinary relationship. If an attribute is added to a concept in the s-tree such that the concept is an anchor of a skeleton tree, then the algorithm suggests to update the table by adding a column as part of the key or to update a f.k. that corresponds to the new identifier of the concept. Of course, the update of the f.k. must be carried out in a cascade fashion starting with the key referenced by the f.k.. If a new identifying relationship is added for changing the anchor corresponding to the key of the table from a strong entity to a weak entity, then the algorithm suggests to update the key of the table by combining the identifier of the owner entity and the local identifier of the weak entity. The semantic mapping formula is updated accordingly. ■

For a semantic mapping, changes that delete elements from the schema can be classified into: deleting a table, deleting an attribute that is not part of the key nor a f.k. of a table, deleting a f.k. of a table, and deleting part of the key of a table. The first three deletions result in updating a semantic mapping formula that references the deleted elements without updating the ontology. The last deletion would require updating the identifier of the associated concepts in the ontology. Changes that delete elements from the ontology would require updating the associated schema in order to maintain the validity of the semantic mapping. In general, if some changes in the ontology (or schema) cause updates in the associated schema (or ontology) in order to maintain the validity of the semantic mapping, the updates will not be carried out automatically; instead, the system will prompt the suggested updates and ask the user what next action should be: executing the update or prohibiting the changes in the ontology (or schema).

The next kinds of changes are merging/splitting elements and changing constraints in schema/ontology. For the sake of space, we omit discussion about merging/splitting, and we use the following example to illustrate how to maintain a semantic mapping when some constraints are changed in the associated schema and ontology.

Example 5 Suppose the following existing semantic mapping formula relate a relational table treat(tid,sgid) with an s-tree TREATMENT ---appliesTo--- SAMPLE_GROUP in an ontology:

treat(tid, $sgid$) ↔
TREATMENT(x_1), tid(x_1, tid), SAMPLE_GROUP(x_2),
appliesTo(x_1, x_2), sgid(x_3, $sgid$).

where the key of the table is the combination of both columns tid and sgid which are identifiers of concepts TREATMENT and SAMPLE_GROUP, respectively, and the relationship
appliesTo is many-to-many.

Later, the data modeler obtained a better understanding of the application by realizing that each treatment only applies to one sample group. Consequently, s/he changed the key of the treat table from the combination of columns tid and sgid to the single column tid. Having this change in the schema, the maintenance algorithm will suggest to change the relationship appliesTo from a many-to-many relationship to a functional relationship TREATMENT ---appliesTo->-- SAMPLE_GROUP.

Conversely, if the database designer obtained a better understanding of the application and changed the appliesTo relationship from many-to-many to functional, then the algorithm will suggest to update the key of the table treat from the combination of tid and sgid to the single column tid.

In both cases, the semantic mapping formula does not change. ■

In summary, the basic principle of maintaining semantic mappings under schema/ontology evolution is to repair the semantic relationship between a table and an s-tree according to knowledge in existing mappings and changes. Specifically, the algorithm attempts to align the key and foreign key constraints in the table with integrity constrains in the ontology by suggesting necessary updates.

6 Conclusions

A semantic mapping between a database schema and an ontology specifies a semantic relationship between the schema and the ontology. For relational schemas, we represented the semantic mapping as a set of relationships between relational tables and s-trees in an ontology. Such a relationship can be represented in terms of a formula with precisely defined semantics. Once such a semantic mapping is established, it is important to maintain the validity of the semantic mapping when the schema or ontology evolves. Mapping maintenance is a challenging problem and it will benefit from a principled and systematic solution. Here we reported on a preliminary effort to define such a solution which will empower database designers, administrators, and integrators.

Unlike the traditional solutions to the problem of schema mapping adaptation, our solution attempts to adapt both the semantic mapping and the associated schema and ontology in order to maintain the validity of the semantic mapping. Based on the previous study on discovering semantic mappings from

database schemas to ontologies, we aim at repairing the semantic relationship between a table and an s-tree by analyzing the semantics in changes to align integrity constraints in schemas and ontologies.

Future work includes developing the complete algorithm and conducting experiments for testing the performance of the solution using both synthetic and real-world semantic mapping evolution scenarios. In addition, we are interested in developing solutions to the problem of maintaining general semantic mappings.

References

1. Y. An, A. Borgida, R. J. Miller, and J. Mylopoulos. A Semantic Approach to Discovering Schema Mapping Expression. In *ICDE'07*.
2. Y. An, A. Borgida, and J. Mylopoulos. Constructing Complex Semantic Mappings between XML Data and Ontologies. In *ISWC'05*.
3. Y. An, A. Borgida, and J. Mylopoulos. Inferring Complex Semantic Mappings between Relational Tables and Ontologies from Simple Correspondences. In *ODBASE'05*.
4. Y. An, A. Borgida, and J. Mylopoulos. Discovering the Semantics of Relational Tables through Mappings. *JoDS VII*, pages 1–32, 2006.
5. J. Banerjee et al. Semantics and Implementation of Schema Evolution in Object-Oriented Databases. In *SIGMOD'87*.
6. B. Benatallah. A Unified Framework for Supporting Dynamic Schema Evolution in Object Databases. In *ER'99*.
7. P. M. Brien and A. Poulovassilis. Schema evolution in heterogeneous database architectures, a schema transformation approach. In *CAiSE'02*, 2002.
8. D. Calvanese, G. D. Giacomo, M. Lenzerini, D. Nardi, and R.Rosati. Data Integration in Data Warehousing. *J. of Coop. Info. Sys.*, 10(3):237–271, 2001.
9. K. T. Claypool, J. Jin, and E. Rundensteiner. SERF: Schema Evolution through an Extensible, Re-usable, and Flexible Framework. In *CIKM'98*.
10. R. Fagin, P. Kolaitis, R. J. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. In *Proceedings of International Conference on Database Theory (ICDT)*, 2003.
11. H. Fan and A. Poulovassilis. Schema evolution in data warehousing environments — a schema transformation-based approach. In *ER'04*, 2004.
12. F. Ferrandina, G. Ferran, T. Meyer, J. Madec, and R. Zicari. Schema and Database Evolution in the O2 Object Database System. In *VLDB'95*.
13. A. Y. Levy, D. Srivastava, and T. Kirk. Data Model and Query Evaluation in Global Information Systems. *J. of Intelligent Info. Sys.*, 5(2):121–143, 1996.
14. R. McCann et al. Maveric: Mapping Maintenance for Data Integration Systems. In *VLDB'05*.
15. Y. Velegrakis, R. J. Miller, and L. Popa. Mapping Adapdation under Evolving Schemas. In *VLDB'03*.
16. C. Yu and L. Popa. Semantic Adaptation of Schema Mappings when Schema Evolve. In *VLDB'05*.