Discovering Semantically Similar Associations (SeSA) for Complex Mapping between Conceptual Models

Yuan An and Il-Yeol Song

College of Information Science and Technology, Drexel University, USA
{yan,isong}@ischool.drexel.edu

Abstract. There is an increasing demand for discovering *meaningful* relationships, i.e., *mappings*, between conceptual models for interoperability. Current solutions have been focusing on the discovery of correspondences between elements in different conceptual models. However, a *complex* mapping associating a structure connecting a set of elements in one conceptual model with a structure connecting a set of elements in another conceptual model is required in many cases. In this paper, we propose a novel technique for discovering semantically similar associations (SeSA) for constructing complex mappings. Given a pair of conceptual models, we create a *mapping graph* by taking the cross product of the two conceptual model graphs. Each edge in the mapping graph is assigned a weight based on the semantic similarity of the two elements encoded by the edge. We then turn the problem of discovering semantically similar associations (SeSA) into the problem of finding shortest paths in the mapping graph. We experiment different combinations of values for element similarities according to the semantic types of the elements. By choosing the set of values that have the best performance on controlled mapping cases, we apply the algorithm on test conceptual models drawn from a variety of applications. The experimental results show that the proposed technique is effective in discovering semantically similar associations (SeSA).

1 Introduction

A mapping between two conceptual models specifies a *meaningful* relationship between the two conceptual models. Semantic mappings have been used increasingly in achieving interoperability [11], capturing data semantics [3], and enabling various operations in the generic model management framework [4]. A mapping can be a simple correspondence between two elements in different conceptual models. For example, if a concept C_1 in one ontology is "equivalent" to a concept C_2 in another ontology, then we could specify the mapping between C_1 and C_2 as $C_1 \leftrightsquigarrow C_2$, where we use the symbol "$\leftrightsquigarrow$" to indicate the correspondence. Moreover, a mapping can be a complex relationship between a structure/association connecting multiple elements in one model and a structure/association connecting multiple elements in another model. For example, the born_in association between the concept Person and the concept Country in one ontology somehow is "equivalent" to the composition of the association born_in between the concept Person and the concept City and the association located_in between the City and the concept Country in another ontology. A complex mapping relationship is often expressed in a declarative formula with precise semantics.

Discovering mappings between models is a very difficult problem in both the database community [17] and the artificial intelligence community [1]. Nevertheless, great effort has been put into the problem of discovering correspondences between model elements, e.g., solutions for schema matching [16] and ontology mapping [10]. A few attempts take database schemas as their subjects and propose solutions for inferring *complex mappings* between database schemas [14, 5]. There is little effort, however, for deriving *complex mappings* between conceptual models in the literature.

A conceptual model (abbreviated as CM) uses modeling constructs such as concepts, relationships, attributes, and constraints to describe a subject matter based on well-defined abstraction mechanisms. Example CMs include Entity-Relationship diagrams, UML class diagrams, and OWL ontologies. Many data-centric applications require solutions to the problem of discovering complex CM mappings to fulfill their goals, for example, data translation over the semantic web [8], data management in peer-to-peer systems [9], and deriving schema mappings using CMs [2]. Current solutions rely on humans to specify the complex CM mapping formulas when they are required - a time-consuming and error-prone task. With the increasing complexity of various CMs in many applications, it is desirable to automate the process. In this paper, we deal with the above discovery problem. We propose a solution to the problem of *discovering semantically similar associations (SeSA) between pairs of corresponding concepts.* Our method takes as input two CMs and a set of correspondences between concepts in the CMs, and generates pairs of semantically similar associations (SeSA). Each association is between a pair of concepts in a CM. The following example illustrates the need for a complex mapping and describes the input and output of our solution.

Example 1. Figure 1 shows two different CMs, CM_1 and CM_2, describing Person, City, and Country, where we use rectangles for concepts, circles for attributes, and lines for relationships. The CM_1 contains two relationships between two concepts Person and Country: a many-to-one (functional) relationship born_in and a many-to-many relationship worked_in, both indicated by the cardinality constraints. The concept Person has an attribute pName, and Country has an attribute countryName. On the other hand, CM_2 describes three relationships born_in, located_in, and hasBeenTo as well as three concepts Person, City, and Country. Some concepts have attributes.

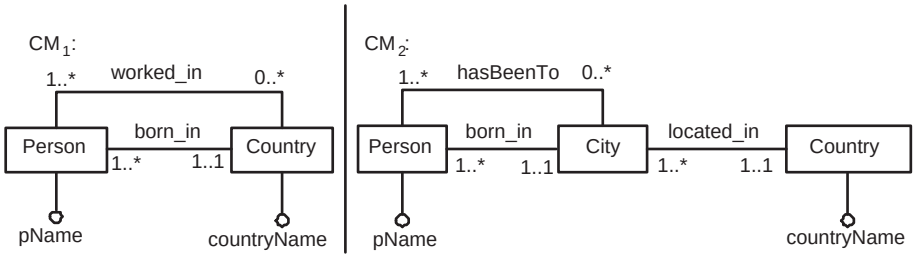

Fig. 1: Two Different CMs

Suppose that mappings between CM_1 and CM_2 are sought for information exchange (imagine that CM_1 is an ontology used by an information system wanting to load data

from another system using ontology CM_2 or vice versa). The creation of complex mappings between two CMs is inherently difficult to automate. To alleviate the problem, we take a two-step approach: (1) specifying simple correspondences between elements in the two CMs; (2) inferring complex mappings between semantically similar structures.

In this paper, we assume that a user can specify correspondences between elements in different CMs manually or using existing schema matching and ontology mapping tools [16, 10]. In particular, we consider the correspondences specified between concepts (which can be inferred from the correspondences between attributes.) For instance, we assume that the following correspondences have been specified: CM_1:Person $\leftrightsquigarrow$$\mathsf{CM}_2$:Person, CM_1:Country $\leftrightsquigarrow$$\mathsf{CM}_2$:Country, where we use prefixes CM_1 and CM_2 to distinguish terms in different CMs. Given two pairs of corresponding concepts, our solution infers a list of pairs of associations. Each pair consists of an association between the pair of concepts in the first CM and an association between the corresponding pair of concepts in the second CM. For example, given the above correspondences, our solution is expected to produce the pair of associations

$\langle$
$\boxed{\mathsf{CM}_1\text{:Person}}$ `-- born_In ->-` $\boxed{\mathsf{CM}_1\text{:Country}}$,
$\boxed{\mathsf{CM}_2\text{:Person}}$ `-- born_In ->-` $\boxed{\mathsf{CM}_2\text{:City}}$ `-- located_in ->-` $\boxed{\mathsf{CM}_2\text{:Country}}$
$\rangle$,
and the pair of associations
$\langle$
$\boxed{\mathsf{CM}_1\text{:Person}}$ `-- worked_in --` $\boxed{\mathsf{CM}_1\text{:Country}}$,
$\boxed{\mathsf{CM}_2\text{:Person}}$ `-- hasBeenTo --` $\boxed{\mathsf{CM}_2\text{:City}}$ `--located_in->-` $\boxed{\mathsf{CM}_2\text{:Country}}$
$\rangle$,
where we use a notation "`-- born_In ->-` " to indicate that `born_in` is a many-to-one (functional) relationship, and a notation `-- hasBeenTo --` to indicate that `hasBeenTo` is a many-to-many relationship, and so on. In each pair, the two associations are "semantically similar" in terms of their cardinality constraints.

Furthermore, we can express a pair of SeSA as a mapping statement in a declarative language (see [3]) or an executable query in a particular query language (e.g., SPARQL [15]). ■

In this paper, we aim to discover an association δ_1 between a pair of concepts $\langle C_1, C_2\rangle$ in a CM and an association δ_2 between a pair of concepts $\langle D_1, D_2\rangle$ in another CM when given correspondences $\mathsf{C}_1 \leftrightsquigarrow \mathsf{D}_1$ and $\mathsf{C}_2 \leftrightsquigarrow \mathsf{D}_2$. We expect that δ_1 and δ_2 are "semantically similar". More complex associations connecting more than two concepts can be constructed by using the pair-wise associations so we leave it for the future work. In addition, we **do not** take the linguistic information encoded in the names of elements into consideration, which will be incorporated in the future work as well. Essentially, we seek for associations that are "semantically similar" in terms of the semantic types of relationships between concepts. For example, an ISA relationship is semantically similar to an ISA relationship, a partOf relationship is semantically similar to a partOf relationship, and so on. To effectively discover "semantically similar" associations from complex CMs, we create a mapping graph by taking the cross prod-

uct of two CM graphs. We then turn the mapping discovery problem into a problem of finding some optimal structures in a graph, which can be solved by applying efficient graph-theoretic algorithms.

Our major contributions are: (1) we propose an innovative approach for discovering SeSA between CMs by using efficient graph-theoretic algorithms; and (2) we demonstrate the effectiveness of the proposed solutions through real world CMs. The rest of the paper is organized as follows. We contrast our approach with related work in Section 2. In Section 3 we present formal notations used later on. We describe the principles in Section 4 and the mapping discovery algorithm in Section 5. In Section 6 we report on experimental studies. Finally, we summarize the results of this work and conclude the paper in Section 7.

2 Related Work

A schema mapping tool infers meaningful relationships between a source and a target database schema from element correspondences. Typical schema mapping tools rely on integrity constraints, especially referential integrity constraints, to assemble "logically connected elements". These logical elements, together with the element correspondences, then give rise to mappings between the schemas. A representative schema mapping tool is Clio [14]. It is natural to ask whether we could utilize the mapping techniques developed in schema mapping tools by viewing the CMs as (relational) database schemas. Unfortunately, this approach does not work as illustrated below.

Let us view the CMs (consider CMs with only binary relationships for now) as relational schemas consisting of unary tables for concepts, binary tables for relationships and attributes. For example, in Example 1, the CM CM_1 could be viewed as a schema consisting of unary table CM_1:Person(x_1), binary tables such as CM_1:born_in(x_1,x_2) and CM_1:Country(x_2), and the obvious foreign key constraints from binary to unary tables; and the same view applies to the CM CM_2 thus creating various tables including CM_2:Person(y_1), CM_2:born_in(y_1, y_2) CM_2:City(y_2), CM_2:located_in(y_2, y_3), CM_2:Country(y_3) and again the obvious foreign key constraints. Suppose that element correspondences were given between the columns of unary tables Person and Country. Then one could in fact try to apply directly the schema mapping techniques to the problem. A desired mapping expressed in the formula M in Example 1 would not be produced due to the following reasons: (i) The schema mapping techniques (e.g., [14]) work by taking each table and using a chase-like algorithm to repeatedly extend it with columns that appear as foreign keys referencing other tables. Such "logical associations" in the source and target are then connected by queries. Specifically, for the CM CM_2 this would lead to logical relations such as CM_2:Person $\wedge$ CM_2:born_in $\wedge$ CM_2:City and CM_2:City $\wedge$ CM_2:located_in $\wedge$ CM_2:Country, but not the entire formula on the right-hand side of "$\Leftrightarrow$" in M. (ii) The semantics that CM_1:born_in is many-to-one relationship leads us to prefer a many-to-one relationship/association between CM_2:Person and CM_2:Country in CM_2. The schema mapping techniques (e.g., [14]) do not use such semantics to pair up "logical associations".

The previous work [2] proposes a semantic approach for deriving schema mapping expressions by using the semantics of the modeling constructs in a CM. That work an-

alyzes the graphical structures and the semantics of relationships (cardinalities, ISA, partOf, etc.,) of the CMs associated with input schemas to eliminate/downgrade *unreasonable* options that arise in mappings between database schemas. In this paper, we focus on the problem of discovering complex mappings between CMs and propose a novel technique which is different from the previous work.

Schema/ontology matching (e.g., [16, 7, 12, 10]) identifies semantic relations between model elements based on their names, data types, constraints, and model structures. The primary goal is to find the one-to-one correspondences between model elements. We aim at the discovery of complex relationships between sets of model elements.

3 Conceptual Models (CMs) and Mappings between CMs

Conceptual Models We consider in this paper the type of CMs, e.g., UML class diagrams, that are often used to describe static aspects of an application. However, we do not restrict ourselves to any particular language for describing CMs. Instead, we use a generic conceptual modeling language (CML), which contains many *common* aspects of most semantic data models (e.g., ER diagrams), UML class diagrams, ontology languages such as OWL, and description logics. Specifically, the language allows the representation of *entities/classes/concepts* (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). Concepts are organized in the familiar ISA hierarchy. Relationships, and their inverses, are annotated with types such as partOf and subject to cardinality constraints, which here allow 1 as lower bounds (called *total* relationships), and 1 as upper bounds (called *functional* relationships). For n-ary relationships connecting more than two entities, and relationships with attributes, we represent them by *"reified relationships"* [6] concepts whose instances represent tuples, connected by so-called "roles" to the tuple elements.

A CM can be represented in a labeled graph called *CM graph*. We construct the CM graph from a CM as follows: We create a concept node labeled with C for each concept C, and an edge labeled with p from the concept node C_1 to the concept node C_2 for each binary relationship p linking C_1 to C_2; for each such p, we annotate it with the type information such as partOf or reified role. For each subclass C_1 of a class C_2, create an edge labeled with ISA connecting C_1 to C_2 with cardinality 1..1 (a C_1 must be a C_2), and 0..1 on the inverse. Graphically, we use rectangles to represent concepts/classes and a line to represent relationships. Textually, a many-to-many relationship p between concepts C and D is written as C ---p--- D, while a many-to-one (functional) relationship p is written as C ---p->-- D.

In this paper, we assume that attributes are globally unique, simple, and single-valued (complex and multi-valued attributes can be transformed into concepts with simple and single-valued attributes.) We use circles to represent attributes. Each attribute is connected to the concept where the attribute belongs to. Since in this paper we focus on discovering associations between concepts, we will strip off attribute nodes in our illustrations in later sections.

CM Mappings A declarative mapping statement over a pair of CMs $\langle \mathsf{CM}_1, \mathsf{CM}_2 \rangle$ is of the form $\mathsf{CM}_1{:}E_1 \Leftrightarrow \mathsf{CM}_2{:}E_2$, where E_1 and E_2 are expressions representing associations over CM_1 and CM_2, respectively. Since the symbol "$\Leftrightarrow$" can be interpreted as subset, superset, or equivalent operator according to the particular application, a more generic mapping statement is written as a two-tuple $\langle E_1, E_2 \rangle$. In the sequel, we will use associations directly in a mapping statement as $\langle \delta_1, \delta_2 \rangle$. The algorithm for translating an association into a conjunctive formula is provided in [3].

4 Principles for Mapping Discovery

We now turn to the task for discovering "semantically similar" associations between CMs. First, we present the principles underlying our approach.

The problem we are addressing is formulated as follows. Given two simple correspondences $v_1{:}\mathsf{C}_1 \leftrightsquigarrow \mathsf{D}_1$ and $v_2{:}\mathsf{C}_2 \leftrightsquigarrow \mathsf{D}_2$ linking two pairs of concepts $\langle C_1, C_2 \rangle$ and $\langle D_1, D_2 \rangle$ in conceptual models CM_1 and CM_2, respectively, find an association δ_1 between C_1 and C_2 and an association δ_2 between D_1 and D_2 such that δ_1 and δ_2 are "semantically similar." The problem is graphically described in Figure 2

A simple case is that both associations δ_1 and δ_2 are direct relationships, i.e., δ_1 is a relationship between $\langle C_1, C_2 \rangle$ and δ_2 is a relationship between $\langle D_1, D_2 \rangle$. To determine whether two relationships are semantically similar, we analyze the types of the relationships, e.g., partOf, and the cardinality constraints imposed on the corresponding concepts participating in the relationships. Our *first principle* is to use the semantic information encoded in the types of relationships and the cardinality constraints imposed on the relationships to discover semantically similar relationships.

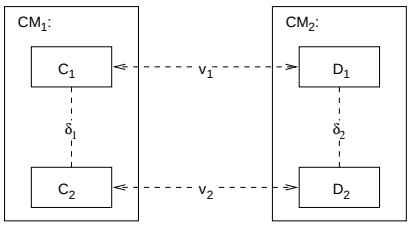

Fig. 2: The Mapping Discovery Problem

However, for a complex CM, an association between two concepts may consist of a sequence of relationships through a set of intermediary concepts. Our goal is to discover SeSAs that are not just single relationships. For instance, each of the two pairs of associations discovered in Example 1 contains an association consisting two relationships that connect Person to Country in CM_2.

Our *second principle* is to analyze the semantic information encoded in the types of the relationships as well as the cardinality constraints imposed on the relationships to discover pairs of SeSAs. Given a CM graph $G_1 = (V_1, E_1)$, an association δ_1 in G_1 is an alternating sequence of different nodes and edges $\delta_1 = \langle v_1, \ell_1, v_2, \ell_2, v_3..., v_m, \ell_m, v_{m+1} \rangle$, where $v_i \in V_1$ and $\ell_i = (v_i, v_{i+1}) \in E_1$ for $i \in \{1, ..., m\}$. Likewise, for a CM graph $G_2 = (V_2, E_2)$, we can represent an association δ_2 as an alternating sequence of different nodes and edges as $\delta_2 = \langle u_1, \gamma_1, u_2, \gamma_2, u_3..., u_n, \gamma_n, u_{n+1} \rangle$, where $u_i \in V_2$ and $\gamma_i = (u_i, u_{i+1}) \in E_2$ for $i \in \{1, ..., n\}$. Intuitively, the following associations δ_1 and δ_2 are semantically similar:

1. $\delta_1 = \langle v_1 \rangle$, $\delta_2 = \langle u_1 \rangle$, and $v_1 \leftrightsquigarrow u_1$;

2. δ_1=$\langle v_1\rangle$, δ_2=$\langle u_1, \gamma_1, u_2\rangle$, $v_1 \hookleftarrow\!\rightsquigarrow u_1$, $v_1 \hookleftarrow\!\rightsquigarrow u_2$, and γ_1 is a functional or ISA relationships; or δ_1=$\langle v_1, \ell_1, v_2\rangle$, δ_2=$\langle u_1\rangle$, $v_1 \hookleftarrow\!\rightsquigarrow u_1$, $v_2 \hookleftarrow\!\rightsquigarrow u_1$, and ℓ_1 is a functional or ISA relationships;
3. δ_1=$\langle v_1, \ell_1, v_2\rangle$, δ_2=$\langle u_1, \gamma_1, u_2\rangle$, $v_1 \hookleftarrow\!\rightsquigarrow u_1$, $v_2 \hookleftarrow\!\rightsquigarrow u_2$, and ℓ_1 and γ_1 are two relationships that both are (i) the type of partOf relationships; (ii) ISA; (iii) many-to-one; or (iv) many-to-many;
4. δ_1=$\langle v_1, \ell_1, v_2\rangle$, δ_2=$\langle u_1, \gamma_1, u_2, ..., u_n, \gamma_n, u_{n+1}\rangle$, $v_1 \hookleftarrow\!\rightsquigarrow u_1$, $v_2 \hookleftarrow\!\rightsquigarrow u_{n+1}$, and $\gamma_i, i = \{1, .., n\}$ have the same semantic type as ℓ_1, e.g., $\gamma_i, i = \{1, .., n\}$ are all many-to-many relationships if ℓ_1 is many-to-many; or the symmetric case when δ_1 and δ_2 get exchanged.
5. δ_1=$\langle v_1,\ \ell_1,\ v_2$,..., $v_m,\ \ell_m,\ v_{m+1}\rangle$, δ_2=$\langle u_1,\ \gamma_1,\ u_2,\ ...,\ u_n,\ \gamma_n,\ u_{n+1}\rangle$, $v_1 \hookleftarrow\!\rightsquigarrow u_1$, $v_{m+1} \hookleftarrow\!\rightsquigarrow u_{n+1}$, and there is a partition of δ_1=$\langle \delta_1^1,\ \delta_2^1,\ ...,\ \delta_k^1\rangle$ and a partition of δ_2=$\langle \delta_1^2, \delta_2^2, ..., \delta_k^2\rangle$ such that δ_j^1 and δ_j^2, $j = \{1, .., k\}$ are semantically similar.

The above conditions 1-4 describe several base cases for semantically similar associations (SeSA). Condition 5 states that two associations are considered semantically similar if they can be divided recursively into partitions in the same size, and the corresponding components of the partitions are semantically similar. The description provides guidelines for designing an algorithm; however, challenges are involved. First, what is the degree of similarity between two associations? Intuitively, the similarity between two "compatible" relationships should be greater than that between two paths with more than one relationships. Second, there are too many ways to enumerate associations between two concepts in a single CM graph. Which associations are the most likely ones in terms of mapping? Third, there are too many ways to enumerate the partitions of a single associations. How to divide an association into partitions? Can an edge/node be divided? How to efficiently decide whether two associations are semantically similar according to the condition 5?

To address these challenges, we turn to efficient graph-theoretic algorithms. The first step is to encode our mapping problem in terms of a single graph structure. We utilize the notion of cross product for two graphs which encodes certain relationships between the two graphs. We need to extend the notion of cross product to encode mapping relationships between two CM graphs.

5 Mapping Discovery Algorithm

In graph theory, the cross product $G = G_1 \times G_2$ of two graph $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the graph $G = (V, E)$, where $V = V_1 \times V_2$ and $t = (v_i u_i, v_j u_j) \in E$ for $v_i, v_j \in V_1$ and $u_i, u_j \in V_2$ if only if $e = (v_i, v_j) \in E_1$ and $r = (u_i, u_j) \in E_2$. We extend the definition of the cross product of two graphs to *the notion of mapping graph* by allowing $t = (v_i u_i, v_j u_j) \in E$ if $v_i = v_j$ and $r = (u_i, u_j) \in E_2$, or $e = (v_i, v_j) \in E_1$ and $u_i = u_j$.

Definition 1 (Mapping Graph). *The mapping graph $M = G_1 \Leftrightarrow G_2$ of two graph $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ is the graph $M = (V, E)$, where $V = V_1 \times V_2$ and $t = (v_i u_i, v_j u_j) \in E$ for $v_i, v_j \in V_1$ and $u_i, u_j \in V_2$ if only if one of the following conditions is satisified: (1) $e = (v_i, v_j) \in E_1$ and $r = (u_i, u_j) \in E_2$; (2) $v_i = v_j$ and $r = (u_i, u_j) \in E_2$; or (3) $e = (v_i, v_j) \in E_1$ and $u_i = u_j$.*

Example 2. Figure 3 (a) shows two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ that both are simple paths with three nodes. Figure 3 (b) shows the cross product of the two

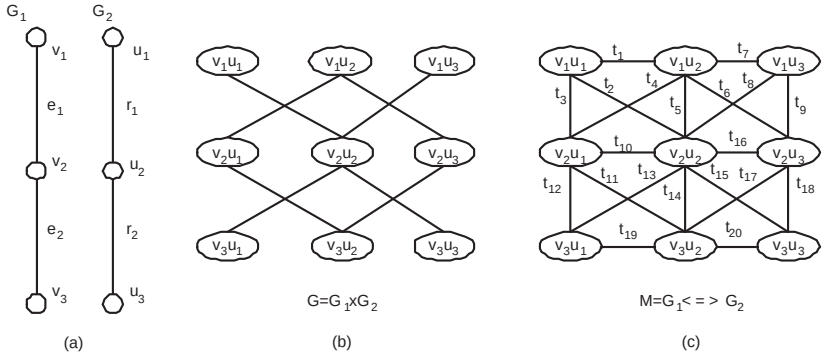

Fig. 3: Cross Product and Mapping Graph

graphs $G = G_1 \times G_2$, while Figure 3 (c) shows the mapping graph of the two graph $M = G_1 \Leftrightarrow G_2$. ■

An edge in the mapping graph $M = (V, E)$ encodes either a pair of edges or a node and an edge in the original graphs. For example, the edge $t_2 = (v_1u_1, v_2u_2) \in M$ in Figure 3 (c) encodes the edge $e_1 = (v_1, v_2) \in E_1$ and the edge $r_1 = (u_1, u_2) \in E_2$; the edge $t_1 = (v_1u_1, v_1u_2) \in M$ in Figure 3 (c) encodes the node $v_1 \in V_1$ and the edge $r_1 = (u_1, u_2) \in E_2$. Moreover, a path in the mapping graph encodes a way to map the source graph to the target graph. For example, the path $\langle (v_1u_1, t_2, v_2u_2, t_{15}, v_3u_3) \rangle \in M$ in Figure 3 (c) maps in a "one-to-one" fashion the elements of the original path G_1 to the elements of the path G_2.

For two conceptual models CM_1 and CM_2, if we are given two pairs of concepts $\langle C_1, C_2 \rangle$ and $\langle D_1, D_2 \rangle$, then a path between the two nodes $\boxed{C_1D_1}$ and $\boxed{C_2D_2}$ in the mapping graph M=$\mathsf{CM}_1 \Leftrightarrow \mathsf{CM}_2$ gives rise to an association δ_1 between C_1 and C_2 and an association δ_2 between D_1 and D_2.

However, the mapping graph encodes all pairing ups between all possible associations connecting $\langle C_1, C_2 \rangle$ in CM_1 and all possible associations connecting $\langle D_1, D_2 \rangle$ in CM_2. In addition, the mapping graph also encodes pairing ups between possible partitions of an association and possible partitions of another association. For a very dense mapping graph, the number of paths between any pair of nodes is quite huge. Therefore, we need to address the problem of discovering the paths in the mapping graph which probably encode those SeSA that are desirable.

The solution is to assign weights to the edges of the mapping graph and discover an optimal structure such as shortest/longest/heaviest paths in the mapping graph. The weight of an edge in the mapping graph denotes the semantic similarity of the two elements encoded by the edge. We assign the similarity as a real number between 0 and 1. We use letter α for highest similarity, e.g., the similarity between two ISA edges, letter β for the similarity between an ISA edge and a functional relationship, letter λ

for compatible similarity, e.g., the similarity between a node and a functional edge, and letter μ for the least similarity. Table 1 shows the categorization of pairs of elements and the similarity values that are assigned to the pairs.

$\langle e_1, e_2 \rangle$, $e_1 \in \mathsf{CM}_1$ $e_2 \in \mathsf{CM}_2$	**Similarity**	$\langle e_1, e_2 \rangle$, $e_1 \in \mathsf{CM}_1$ $e_2 \in \mathsf{CM}_2$	**Similarity**
e_1=ISA edge e_2=ISA edge	$0 \leq \alpha \leq 1$	e_1=ISA edge e_2=functional edge	$0 \leq \beta \leq 1$
e_1=many-to-many edge e_2=many-to-many edge	$0 \leq \alpha \leq 1$	e_1=a node e_2=functional edge	$0 \leq \lambda \leq 1$
e_1=many-to-one edge e_2=many-to-one edge	$0 \leq \alpha \leq 1$	e_1=reified role edge e_2=reified role edge	$0 \leq \alpha \leq 1$
e_1=partOf edge e_2=partOf edge	$0 \leq \alpha \leq 1$	other	$0 \leq \mu \leq 1$

Table 1: Assigning Similarity to Pair of Elements $\langle e_1, e_2 \rangle$ of two Conceptual Models CM_1 and CM_2

Example 3. Figure 4 (b) shows the mapping graph of the two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ in Figure 4 (a). Both G_1 and G_2 contain a functional relationship

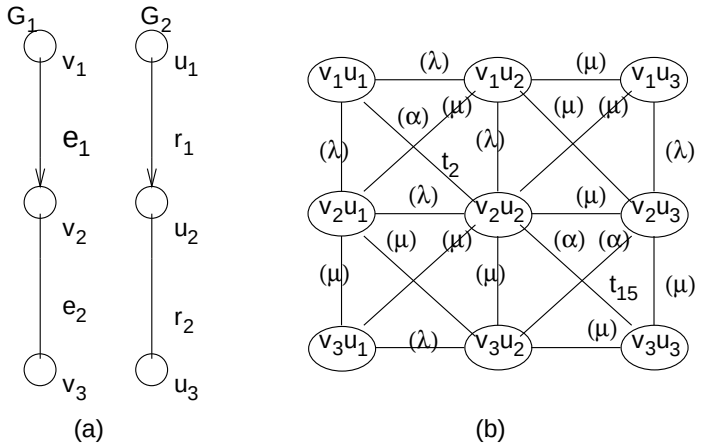

Fig. 4: Weight Assignments to a Mapping Graph

edge which is indicated by an arrow, e.g., $e_1 = (v_1, v_2) \in E_1$, and a many-to-may relationship edge. Weights enclosed by parentheses are assigned to the edges of the mapping graph in Figure 4 (b). To reduce clumsiness, we only show two edge labels: t_2 and t_{15} in the Figure.

We compute the similarity between the two original paths by computing the weights of the paths that encode the two original paths. The weight of a path is the product of

the weights of the edges along the path. The path $\langle(v_1u_1, t_2, v_2u_2, t_{15}, v_3u_3)\rangle \in M$ has the heaviest weight α^2. By taking the weight of the heaviest paths, we obtain the similarity between the two original paths as α^2. ■

Equipped with the weighted mapping graph, we design the mapping discovery algorithm as to discover the heaviest paths between two given nodes, where the weight of a path is the product of the weights of the edges along the path. This is justified by our preference to the paths with fewer edges. Each edge has a greater weight/similary value. To compute the heaviest paths, we take the logarithm of the edge weights and negate the results. After this, the traditional algorithms for computing shortest paths in a graph, e.g., Dijkstra's algorithm, will produce the expected results. Figure 5 presents the procedure discSeSA(), which takes as input two CMs and two simple correspondences linking a pair of concepts in the first CM to a pair of concepts in the second CM. The results of the discSeSA() are pairs of desired associations. A resulted pair of associations are considered as "semantically similar" because they have the highest similarity based on the appropriate similarities assigned to the edges in the mapping graph.

Procedure: discSeSA(G_1, G_2, L)
Input: conceptual model graphs $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$, and simple correspondences L=$\{C_1 \rightsquigarrow D_1, C_2 \rightsquigarrow D_2 \mid C_1, C_2 \in V_1, D_1, D_2 \in V_2\}$
Output: $\{\langle\delta_1, \delta_2\rangle \mid \delta_1$ is an association between C_1 and C_2, δ_2 is an association between D_1 and D_2, and δ_1 and δ_2 have the greatest similarity$\}$
Steps:

1. Create the mapping graph $M = (V, E) = G_1 \Leftrightarrow G_2$ of the input CM graphs, where $V = V_1 \times V_2$ and E is the set of edges of the mapping graph;
2. **For** each edge $e = (v_iu_i, v_ju_j) \in E$, $v_i, v_j \in V_1$, $u_i, u_j \in V_2$
   (a) Assign a weight to e according to Table 1; the two elements encoded by the edge e are either two edges $(v_i, v_j) \in E_1$, $(u_i, u_j) \in E_2$, or a node and an edge, e.g., $v_i = v_j$.
3. **End for**
4. **For** each edge $e \in E$, let w be the weight of e
   (a) **Let** $w = -\lg w$
5. **End for**
6. **Let** $P =$ shorestPath$(M)^a$.
7. **Let** $A =$ makeAssociates$(P)^b$.
8. return A.

[a] shortestPath() computes the shortest paths of a weighted graph.
[b] makeAssociation(P) splits each path in P into two associations in the original graphs.

Fig. 5: discSeSA Procedure

6 Experimental Results

We now report our experimental results. The purpose of the experiments is three-fold: (1) selecting the values for the parameters α, β, λ, and μ which are presented in Table 1, (2) applying the proposed technique to various CMs in different applications, and (3) testing the efficiency and effectiveness of the proposed algorithm. The algorithm is implementated in JAVA and the experiments were conducted on a PC with an Intel Core 2 Duo processor and 2G memory.

Data Set. The test data sets (see Table 2) in our experiments were collected from a variety of applications. The CMs Sdb0, Sdb1, Sdb2, and Sdb3 are four versions of the conceptual model for describing a biological sample database extracted from the industrial GeneExpress Data Management (GXDM) project described in [13]. In this paper, we used controlled mapping cases based on these four CMs to empirically determine the values of the similarity parameters that would have the best performance. We experimented the mappings between Sdb0 and Sdb1, betwen Sdb1 and Sdb2, and between Sdb2 and Sdb3. The remaining three pairs of test CMs in our experiments were collected from our previous work in [2].

First CM	# Nodes	# Edges	Second CM	# Nodes	# Edges	Time for Creating Mapping Graph (sec)	Avg. Time for Discovering SeSA (sec)
Sdb0	68	73	Sdb1	54	58	7.8	1.9
Sdb1	54	58	Sdb2	74	80	9.5	2.2
Sdb2	74	80	Sdb3	49	56	8.0	1.8
Bibliographic	75	80	DBLP	7	10	0.32	0.068
Amalgam1	7	14	Amalgam2	26	27	0.13	0.028
Factbook	52	112	Mondial	26	55	4.6	1.7

Table 2: Characteristics of Test Data

Table 2 shows the characteristics of the test CMs. For each pair, the table lists the numbers of nodes and the numbers of edges of the first and second CMs.

Selecting Values for the Similarity Parameters. The key to the proposed technique is to assign weights to the edges in the mapping graph. The value of a weight is based on the semantic similarity between two elements encoded by an edge. Table 1 presents the categorization of pairs of elements and the similarity values that are assigned. The values denoted by the letters α, β, λ, and μ are real numbers between 0 and 1. We hypothesized that different values assigned as element similarity might have different performance in terms of discovering SeSA. We conducted experiments to verify the hypothesis and hopefully to select the set of values that had the best performance on our controlled experiments. First, we assigned each parameter an array of possible values as follows. Let $\alpha = \{0.9, 0.8, 0.7, 0.6, 0.5\}$, $\beta = \{0.8, 0.6, 0.5, 0.4, 0.2\}$, $\lambda = \{0.8, 0.6, 0.5, 0.4, 0.2\}$, and $\mu = \{0.01\}$. Second, we chose a number of pairs of concepts in each of the following CMs: Sdb0, Sdb1, Sdb2, and Sdb3, and tested mappings between Sdb0 and Sdb1, between Sdb1 and Sdb2, and between Sdb2 and

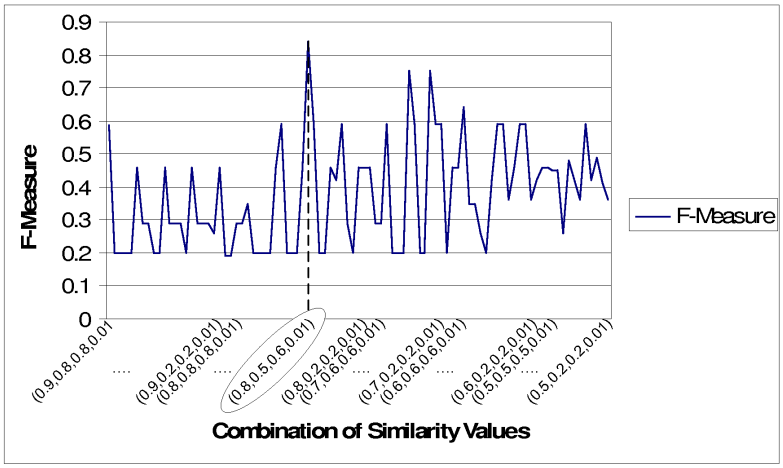

Fig. 6: Experimental Results for Selecting Similarity Values

Sdb3. Third, we combined the values from the four arrays under the constraints $\alpha \geq \beta$ and $\alpha \geq \lambda$, which indicate that the highest similarity value should not be less than other similarity values. Finally, for each combination of the values, we measure the performance of the algorithm using this set of values for assigning weights as described in the following.

To measure the performance of the algorithm, for each mapping case, we manually chose pairs of associations based on our understanding and expectation on the CMs. These selected pairs of associations acted as the "gold standard" when we compared the results generated by the algorithm using different similarity values. We were concerned with the following two questions: Did the algorithm generate all the expected pairs of associations? Did the algorithm generate pairs of associations that were not manually selected. The first concern is related to the traditional *recall* measure, while the second concern is about the *precision* measure. Specifically, let R be the set of "gold standard" pairs and let P be the set of pairs generated by the algorithm. The precision and recall measures are computed as: $precision = \frac{|P \cap R|}{|P|}$ and $recall = \frac{|P \cap R|}{|R|}$. To measure the overall performance, we take the harmonic mean of precision and recall which is called F-Measure, calculated as follows:

$$F\ measure = \frac{2}{\frac{1}{precision} + \frac{1}{recall}}.$$

Using the controlled mapping cases with expected results, we measure the performance of the algorithm with different similarity values in terms of the F-measures. Figure 6 shows the average F-meaures for all combinations of values over the controlled experiments. The x-axis lists the combinations of the values in the form of (α, β, λ, μ). There were 91 combinations of values tested (the total number of combinations is 125 but some combinations were not considered due to the given constraints.) The highest peak of the average F-measure curve appears at the point on the x-axis which corresponds to the combination $\{\alpha = 0.8, \beta = 0.5, \lambda = 0.6, \mu = 0.01\}$. The ups and downs

of the curve indicate that different values assigned as element similarity indeed had different performance.

Results of Applying the Algorithm. With the set of selected values for similarity, we applied the algorithm to the mapping pairs in our test data sets including SDB0,.., SDB3 again. The last two columns of Table 2 contain the times for creating mapping graphs and discovering SeSA for the test pairs. In terms of time complexity, it took several seconds to create a mapping graph for some pairs of CMs in our test set. However, the mapping graph of a certain pair only needs to be created once and can be reused many times. The process of discovering SeSA spent a couple of seconds to produce the final results. It employed the standard shortest path algorithm, e.g., Dijkstra's algorithm.

To evaluate the effectiveness of the algorithm, we continue to use the notion of recall and precision. This time, we conducted a post-inspection to measure the recall and precision. Specifically, for a mapping case, we inspected the pairs of associations generated by the algorithm against the CMs. For precision, we checked whether each pair in the result set indeed contained two "semantically similar" associations. For recall, we checked whether there were other "semantically similar" associations that were not returned by the algorithm. The inspection results showed that the algorithm is effective in discovering SeSA. In particular, precisions for all mapping cases were 100%, while the average recall over all mapping cases is about 90%. The imperfect recall is due to the algorithm's preference to heaviest (shortest) paths. For example, in the pair of CMs, CIA factbook and Mondial, both CMs contain a relationship City `-- capital->-` Country and a path City `-- capital ->-` Province `--located_in->-` Country. The algorithm generates the two relationships as a pair of SeSA excluding the two paths. A solution would be to set the similarity $\alpha = 1$; however, this setting would disable α as a damper factor for longer paths. We plan to extend the algorithm by using an additional parameter for controlling the content of the result set in our future work.

7 Conclusions

In this paper, we studied the problem of discovering semantically similar associations (SeSA) in two different conceptual models. Our method finds an association between a pair of concepts in one conceptual model and a "semantically similar" association between a pair of concepts in another conceptual model. We are motivated by the need of specifying complex semantic mappings between CMs for many applications that require interoperability such as data management over the semantic web and peer-to-peer systems. We proposed a novel technique for discovering desirable SeSA by using efficient graph-theoretic algorithms. Our solution is unique in that we turn the problem of discovering SeSA into a problem of finding shortest paths in a special graph called *mapping graph*. We create a mapping graph by taking the cross product of the two input CM graphs. Our contributions include experiments for evaluating the efficiency and effectiveness of the proposed algorithm. Experimental results showed that the technique was effective in discovering SeSA in our problem setting. We plan to incorporate the

linguistic information in the names of elements into the mapping discovery approach in the future work.

References

[1] AAAI. *AI Magazine 26(1): Special Issue on Semantic Integration*. 2005.

[2] Y. An, A. Borgida, R. J. Miller, and J. Mylopoulos. A Semantic Approach to Discovering Schema Mapping Expressions. In *Proceedings of International Conference on Data Engineering (ICDE)*, pages 206–215, 2007.

[3] Y. An, A. Borgida, and J. Mylopoulos. Discovering the Semantics of Relational Tables through Mappings. *Journal on Data Semantics*, VII:1–32, 2006.

[4] P. Bernstein. Applying Model Management to Classical Meta Data Problems. In *CIDR*, 2003.

[5] A. Bonifati, E. Q. Chang, T. Ho, V. S. Lakshmanan, and R. Pottinger. HePToX: Marring XML and Heterogeneity in Your P2P Databases. In *Proceedings of International Conference on Very Large Data Bases (VLDB)*, pages 1267–1270, 2005.

[6] M. Dahchour and A. Pirotte. The Semantics of Reifying n-ary Relationships as Classes. In *Information Systems Analysis and Specification*, pages 580–586, 2002.

[7] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. imap: discovering complex semantic matches between database schemas. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 383–394, New York, NY, USA, 2004. ACM Press.

[8] A. Halevy, Z. G. Ives, P. Mork, and I. Tatarinov. Piazza: data management infrastructure for semantic web application. In *Proceedings of International Conference on World Wide Web (WWW)*, pages 556–567, 2003.

[9] A. Y. Halevy, Z. G. Ives, D. Suciu, and I. Tatarinov. Schema Mediation in Peer Data Management Systems. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 505–516, 2003.

[10] Y. Kalfoglou and M. Scholemmer. Ontology Mapping: The State of the Art. *The Knowledge Engineering Review*, 18(1):1–31, 2003.

[11] M. Lenzerini. Data Integration: A Theoretical Perspective. In *Proceedings of the ACM Symposium on Principles of Database Systems (PODS)*, pages 233–246, 2002.

[12] J. Madhavan, P. Bernstein, A. Doan, and A. Halevy. Corpus-Based Schema Matching. In *Proceedings of the International Conference on Data Engineering (ICDE)*, pages 57–68, 2005.

[13] V. Markowitz and T. Topaloglou. Applying Data Warehousing Concepts to Gene Expression Data Management. In *BIBE'01*, pages 65–72.

[14] L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernández, and R. Fagin. Translating web data. In *VLDB*, pages 598–609, 2002.

[15] E. Prud'hommeaux and A. Seaborne. *SPARQL Query Language for RDF*. W3C Working Draft 4, http://www.w3.org/TR/rdf-sparql-query, 2006.

[16] E. Rahm and P. A. Bernstein. A Survey of Approaches to Automatic Schema Matching. *VLDB Journal*, 10:334–350, 2001.

[17] SIGMOD. *SIGMOD Record 33(4): Special Issue on Semantic Integration*. 2004.