Reducing the Cost of Validating Mapping Compositions by Exploiting Semantic Relationships

Eduard Dragut[1] and Ramon Lawrence[2]

[1] Department of Computer Science, University of Illinois at Chicago
[2] Department of Computer Science, University of British Columbia Okanagan

Abstract. Defining and composing mappings are fundamental operations required in any data sharing architecture (e.g. data warehouse, data integration). Mapping composition is used to generate new mappings from existing ones and is useful when no direct mapping is available. The complexity of mapping composition depends on the amount of syntactic and semantic information in the mapping. The composition of mappings has proven to be inefficient to compute in many situations unless the mappings are simplified to binary relationships that represent "similarity" between concepts. Our contribution is an algorithm for composing metadata mappings that capture explicit semantics in terms of binary relationships. Our approach allows the hard cases of mapping composition to be detected and semi-automatically resolved, and thus reduces the manual effort required during composition. We demonstrate how the mapping composition algorithm is used to produce a direct mapping between schemas from independently produced schema-to-ontology mappings. An experimental evaluation shows that composing semantic mappings results in a more accurate composition result compared to composing mappings as morphisms.

Keywords: mapping, composition, integration, model management, semantics, metadata.

1 Introduction

The focus of this work is the study of mappings between models. A model is a schema, an ontology, or some other data representation construct. Mappings are designed to relate the information stored in models. There has been a wide variety of model and mapping representations [2,6,7,9,15,19] defined. Each representation has its own benefits and characteristics related to expressability, applicability, and operability. The problem of semi-automatically creating mappings can be divided into two steps. First, a *match* between two schemas specifies semantic correspondences between their elements [18]. These correspondences can be as simple as *inter-schema correspondences* [16] (or *morphisms* [15]) or they can convey certain semantic relationships (e.g. IsA, HasA) [8]. Second, these correspondences are elaborated to generate *instance (data) level mappings* (e.g. using a system such as Clio [16]). In this paper we are concerned with the former, which we refer to as *metadata level mappings*.

Our contribution is an efficient algorithm for composing semantic mappings at the metadata level that is able to focus human attention only on the potential problems areas in the composition result. Previous composition/reusing algorithms [2,4,5,12,15]

R. Meersman, Z. Tari et al. (Eds.): OTM 2006, LNCS 4275, pp. 882–890, 2006.

did not use semantic mappings, and consequently, require the user to validate the entire mapping. We propose a semantic mapping that can be easily expressed and constructed, preserves semantics under composition, complements data level mappings defined using views and it is supported by the state of art matching algorithms [3,4,8,14]. We show that by capturing explicit semantics in mappings, the composition result contains more semantic information, and it is possible to quickly identify problems in the computed mapping that require human intervention. We identify the composition issues both analytically and experimentally.

The organization of this paper is as follows. In Section 2, we overview existing metadata mapping representations and discuss the issues in capturing semantics and performing composition. Our main contribution is in Sections 3 and 4 where we define semantic metadata mappings between models and provide a set of rules for inverting and composing mappings. An experimental evaluation in Section 5 shows that composing semantic mappings, as opposed to morphisms, results in a better mapping composition result. The paper closes with future work and conclusions.

2 Background

Various forms of metadata level mappings have been used to define model management [2] operators and their semantics. In this setting, models are assumed to have the same expressive characteristics as an EER model. *We assume the same definition for models in this work.* Two different mapping representations have been proposed: morphisms [15] and helper models [2,13].

Morphisms. are simple binary relationships between model elements [15,16] that carry loose semantics about the actual relation of the concepts mapped. Formally, given two models, A and B, a mapping map between them is defined as a morphism that consists of a set of pairs $< a, b >$, where $a \in A$ and $b \in B$, which implies that a and b are *similar*. Composition of morphisms assumes similarity relation to be transitive. That is, given a morphism, m_1, between models A and O, and another morphism, m_2, between O and B, and if $< a, o > \in m_1$ and $< o, b > \in m_2$, then *a* is assumed to be similar to *b*.

The composition of morphims has several problems. First, any time an m:1 mapping is composed with an 1:n mapping, the composition result is a cross-product consisting of $m \times n$ correspondences [4,5]. Many of these correspondences incorrectly specify a relationship between the elements. Second, an even worse case occurs when there are no direct correspondences between the concepts. For example, there are no direct relationships between {Street1, Street2} and {City, Country, State, Street}. These problems arise due to the lack of an explicit semantic information in the mapping representation.

Mappings with helper models. [2,17] have also been defined for use in model management [2]. In this case, a mapping between two models A and B can be expressed by using a model, map, and two morphisms, one between A and map and the other between map and B. This mapping format allows a set of objects of A to be related to a set of objects in B in complex ways. It uses the elements and the relationships within the helper model to relate *sets of objects* in A and B (see [2] for a comprehensive description). A mapping composition algorithm is presented in [2,17] for use

in model management. This composition algorithm is imperfect as it does not exploit relationships in the helper model to yield a more accurate composition result. Consequently, the algorithm may miss some of the relationships and second it may suggest false relationships.

3 Mapping Representation

We aim to provide a mapping definition that subsumes most of the relationship kinds that the state of the art matching algorithms discover and investigate operations over this mapping definition. A mapping is a metadata level semantic correspondence between elements in different models.

Definition 1. *Given two models A and B, a mapping between them map consists of a set of mapping elements. Each mapping element is a directed, kinded binary relationship between a pair of elements not in the same model. That is, it is a set of triplets $< a, type, b >$, where $a \in A$, $b \in B$, and* type *is the semantic type of the relationship. The values of* type $\in$ {IsA, AKindOf, HasA, PartOf, =, Contains, ContainedBy, Unknown, Complex, NoRel}.

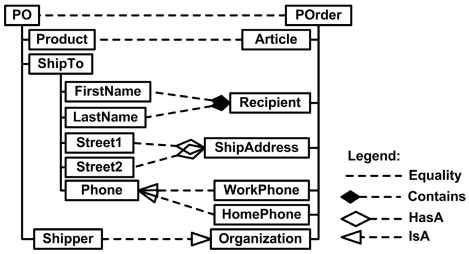

Fig. 1. Mapping example

The mapping is directed from A to B. The relationships used in our mapping representation are well-known except for *Unknown*, *Complex*, and *NoRel* which will be defined later. We assume a set-theoretic framework for defining the semantics of the relations as in [8,19,22]. For instance, a *IsA* b if the content of b contains the content of a, a *HasA* b if the content of b is *part of* the content of a. An example mapping in our representation is in Figure 1. The semantic relationship types are:

- **Equality:** An $Equality$ relationship between two elements a and b, $< a, =, b >$, means that the concept a is equivalent to b.
- **IsA:** An IsA relationship between two elements a and b, $< a, IsA, b >$, means that the concept a is a specialization of b. As this is a binary relation its inverse can be defined as in the algebra of relations [10]. That is, $< b, a > \in IsA^{-1}$ iff $< a, b > \in IsA$. We assign a more descriptive name to the inverse of IsA, i.e. *AKindOf*. Since all remaining relations are binary, a similar approach is assumed for the definition of their inverses.
- **HasA:** A $HasA$ relationship between two elements a and b, $< a, HasA, b >$, means that the concept a has a concept b as part of its representation. The inverse of $HasA$ relationship is $HasA^{-1}$ and its descriptive name is $PartOf$.
- **Contains:** A $Contains$ relationship between two elements a and b, $< a, Contains, b >$, means that the concept a fully encapsulates concept b which cannot independently exist without a. $Contains$ is a stronger relationship than

$HasA$. The inverse of $Contains$ relationship is $Contains^{-1}$ and its descriptive name is $ContainedBy$.

- **Complex:** A $Complex$ relationship between two elements a and b, $< a, Complex, b >$, means that the relationship between concept a and b may require a functional specification: $f(a) = b$. That is, two concepts related by $Complex$ do not have a direct equivalence relationship, but the equivalence relationship can be revealed through a function, whose purpose is to "equalize" the meaning of the two concepts. $Complex$ is considered a Level 2 relationship in [6] and it is the chief subject of the work in the iMap project [3].
- **NoRel:** A $NoRel$ relationship between two elements a and b, $< a, NoRel, b >$, means that there is no relationship between concept a and b. The inverse of the $NoRel$ relationship is still a $NoRel$ relationship (i.e. it is symmetric).
- **Unknown:** An $Unknown$ relationship between two elements a and b, $< a, Unknown, b >$, means that the relationship between concept a and b is not known. $Unknown$ relationships often arise after composition and can be considered the default relationship between all elements until a mapping is defined.

Our mapping relationships capture the semantics of the relationship between model elements like using helper models, but retain the simplicity of inversion and composition given by algebra of relations [20,21]. The usefulness of relationship types besides equality and similarity in specifying mappings is clear from the example in Figure 1. For instance, the IsA relationship clearly captures the semantic relationship between WorkPhone and HomePhone to Phone. It is evident that if morphisms were used then we would have lost the semantic information between these concepts.

4 Invert and Compose Operators

The two fundamental operators required in integration scenarios is the ability to $Invert$ a mapping and the ability to $Compose$ two mappings to produce another mapping. We discuss how these operators are defined in this section.

Invert Operator: The $Invert$ operator has been defined for morphisms [15] and is simply the swapping of the left and right elements of the morphism. Such a simple definition is possible since the similarity relationship is symmetric.

A mapping in our framework consists of a set of directed, kinded, binary relationships. Thus, the semantic type of the relationship must also be inverted when applying the $Invert$ operator. The approach is very similar in spirit with other works that manipulate relations, e.g. [1] in the field of temporal logic.

Definition 2. *Consider two models A and B and a mapping, map, between them. Let* m *be a mapping element whose expression is $< a, type, b >$, where type is one of the types defined above. Then its corresponding inverted mapping element, denoted m^{-1}, is given by the following expression: $< b, type^{-1}, a >$. Moreover, the invert of map, denoted by map^{-1}, is defined from B to A and its expression is given by: $map^{-1} = \{< b, type^{-1}, a > | < a, type, b > \in map\}$.*

For example, $< HomePhone, IsA, Phone >^{-1} = < Phone, IsA^{-1}, HomePhone >$.

Table 1. Composition rules

(a,b) (b,c)	=	IsA	IsA^{-1}	Contains	Contains^{-1}	HasA	HasA^{-1}
=	=	IsA	IsA^{-1}	Contains	Contains^{-1}	HasA	HasA^{-1}
IsA	IsA	IsA	Unknown	Unknown	Unknown	Unknown	Unknown
IsA^{-1}	IsA^{-1}	Unknown	IsA^{-1}	Contains	Contains^{-1}	HasA	HasA^{-1}
Contains	Contains	Contains	Unknown	Contains	Unknown	Contains	Unknown
Contains^{-1}	Contains^{-1}	Contains^{-1}	Unknown	Unknown	Contains^{-1}	Unknown	Contains^{-1}
HasA	HasA	HasA	Unknown	Contains	Unknown	HasA	Unknown
HasA^{-1}	HasA^{-1}	HasA^{-1}	Unknown	Unknown	Contains^{-1}	Unknown	HasA^{-1}

Compose Operator: Composing two mappings involves defining a composition operation between the elements of the mappings (i.e. triplets of form $< a, type, b >$). The relationship types that constitute the core of a mapping definition have well defined composition properties. Table 1 provides the composition rules that govern our composition technique. This table can be read where a cell that has a row header of type T, a column header of type U, and a cell value of V means that composing the mapping elements $< a, T, b >$ with $< b, U, c >$ results in a composed mapping element of $< a, V, c >$. For instance, composing $< a, =, b >$ with $< b, IsA, c >$ results in a composed mapping element of $< a, IsA, c >$.

Definition 3. *Given two mappings map_{AB} between models A and B and map_{BC} between models B and C the $Compose$ operator, denoted by $\circ$, produces a new mapping $map_{AC} = map_{BC} \circ map_{AB} = map_{BC}(map_{AB})$ between models A and C:*

$$map_{AC} =_{df} \{< a, t, c > | \exists b :< a, t_1, b > \in map_{AB} \wedge < b, t_2, c > \in map_{BC} \wedge$$

$$t = T(t_1, t_2)\}, \textit{ where } T(\cdot, \cdot) \textit{ is the entry in Table 1.}$$

Some of the entries in the table were presented in other works. For instance, the composition between IsA, IsA^{-1}, *Equality* and *NoRel* are presented in [6] and some of the composition outcomes between *IsA*, *Contains* and *HasA* are reported in [17].

There are no entries for the $Unknown$, $Complex$ and $NoRel$ relationships in the table because $Unknown$ composed with any other type gives $Unknown$, $Complex$ composed with $Equality$ preserves $Complex$ and composed with any other type is $Unknown$. We do not explicitly handle $NoRel$ relation because we prefer $Unknown$ over $NoRel$ since it helps us establish desirable indirect relationships, which would have been lost employing $NoRel$. For instance, if we followed the directions of other works that handle relations (e.g. [1,6]) we would have obtained *No Info*, which is not a desirable outcome. An example (in Figure 2) has convergent IsA relationships (<Shipper, IsA, Organization> and <Organization, IsA^{-1}, Supplier>). For this case it is desirable to retain the information that the two are IsA siblings. The motivation is two fold: (1) it is hard, if not impossible, to define a direct relationship between such concepts and (2) such information might prove valuable if a merge or an alignment [11] operation is performed between the models.

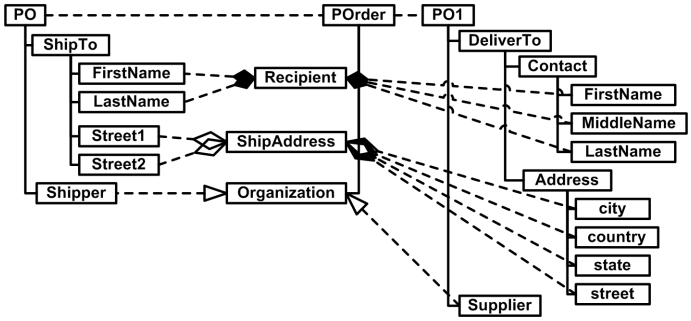

Fig. 2. An example of the problematic cases

Having relationship types that capture more semantics besides the generic *similarity* relationship, we are able to create a composition result with better semantics and avoid the false relationships that simple similarity can produce. Further, the cases where the result is $Unknown$ are especially important as they are direct indicators that the composition of these elements are not well-defined and should be investigated by the user.

The mappings in this system have the following desirable properties:

Proposition 1. *Consider two models A and B and a mapping, map, between them. We denote the invert of map by map^{-1}. The following properties hold:*

1. $b \in (map(map^{-1}))(b)$, $b \in B$,
2. $a \in (map^{-1}(map))(a)$, $a \in A$.

Proposition 2. *Mapping composition is symmetric in our framework, i.e.*

$$map_{AC}^{-1} = (map_{BC} \circ map_{AB})^{-1} = map_{AB}^{-1} \circ map_{BC}^{-1} = map_{CA}.$$

An important feature of the composition algorithm is that it only suggests correct directed, kinded relationships in the composition result. This can be seen by examining the table of composition rules. False results are not produced as transitivity is only applied when it is safe to do so, and the $Compose$ operator uses the $Unknown$ relationship to indicate when it is not possible (in general) to suggest a relationship type given only the information expressed in the two mappings. Thus, $Unknown$ relationship types are valuable as they are the ones that require semi-automatic resolution (e.g. human intervation). Note that we do not claim to retrieve the set of *all* true relationships, but state that the set of relationships we retrieve consists *only* of true relationships.

5 Experimental Evaluation

The experiment has two main goals: to show that our framework is robust when applied to real world application and that we are able to correctly identify problematic cases.

Experiment Setup: We considered five real-world XML schemas in the purchase order domain: CIDR, Excel, Noris, Paragon, and Apertum [14]. They are assigned

Table 2. Mapping distribution

relations	CIDR	Excel	Noris	Paragon	Apertum
equality	18	30	32	36	34
isa	6	11	24	2	13
hasa/part of	14	18	0	10	2
complex	0	0	2	2	2
overlap ratio	0.775	0.84	0.65	0.62	0.61

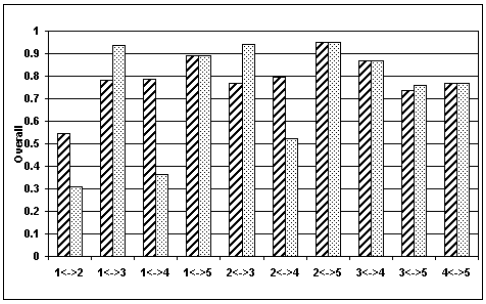

Fig. 3. Schema-to-Schema Mapping Statistics

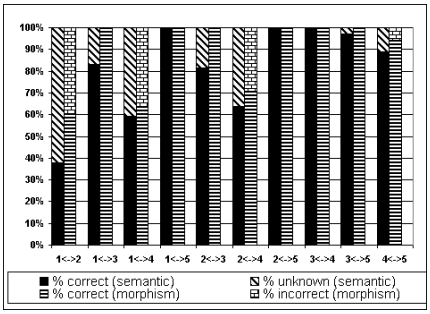

Fig. 4. Algorithm Output Breakdowns

numbers 1, 2, 3, 4, and 5 respectively in order to reference them in our graphs. We also considered a reference ontology to which each XML schema is manually mapped. The reference ontology (see [5]) was created such that it did not have all concepts in the five schemas, such as *unitOfMeasure*, *count*, and *VAT* information. Table 2 shows in the last row the ratio of schema elements that can be mapped to the ontology. These ratios are an useful indicator of the inherent limitation in the performance of the composition because the composition may miss some valid correspondences. Table 2 also shows the mapping broken down in terms of the relationship types defined in this paper for each of the schema-to-ontology mappings.

Measures for composition quality: We measure the performance of our system via three metrics: *Precision*, *Recall*, and *Overall* [4,5,14]. Overall is defined as $Overall = Recall * (2 - 1/Precision)$. We introduce a new metric, called *User Effort*, meant to characterize the post-processing user effort after the composition algorithm is applied.

The Experiment: After we have manually defined the mappings between the five schemas and the ontology we have applied the composition algorithm developed in this work to compute the direct mappings between the schemas. We have also applied composition when the mappings are morphisms (using a natural join). The results are given in terms of the Overall statistic in Figure 3. The first striped bar is our semantic approach, while the second bar is for morphisms. The results are shown for all 10 possible schema-to-schema mappings.

Observe that overall our composition algorithm outperforms composition with morphisms. In five cases, the performances of the two systems are comparable. These cases are an indication that transitivity plays by the rules and both systems are able to

correctly compute the composition. However, in the bad cases, where assuming transitivity is costly and produces many false matches, composing semantic mappings is much better and results in much higher overall performance. For instance, the composed mapping between schemas 1 and 4 produced using our approach has an overall value double that of morphisms. This supports our statement that our composition rules preserve the transitivity of composition as exhibited by morphisms only when it is safe to do so.

Let's consider the two cases when our composition is apparently outperformed, i.e. 1<–>3 and 2<–>3 in Figure 3. In order to understand the behavior of composition in these situations we provide in Figure 4 a detail breakdown of the output of the two composition algorithms. The result of our algorithm is divided into the percentage of the correct and $Unknown$ relationships suggested (as we do not produce incorrect relationships). The morphism output is divided into the percentage of correct and incorrect relationships suggested. All percentages are relative to the total number of output relationships. In these two cases our composition algorithm casts doubts on the set of relationships that later turn out to be true. They are mostly the effect of composing divergent $HasA$ relationships.

Other interesting cases are 1<–>2 and 1<–>4. For these two cases the percentage of the incorrect relationships among the set of all $Unknowns$ is 64% and 89%, respectively. Every time incorrect relationships are likely to be produced during composition our algorithm is clearly better than the composition over morphisms.

User Effort: Define the *User Effort* metric as the ratio of the number of $Unknown$ relationships to the number of all produced relationships. These statistics are represented by the percentage of $Unknown$'s in Figure 4. This measure does not consider the effort required to find the relationships missed by the composition operator. Within the mapping composition frameworks presented in Section 2 (e.g. morphisms) the user effort is at its peak since every proposed relationship needs to be investigated and validated.

In our experimental environment, on average the User Effort is only 19%, and even lower rates of user involvement occur when transitivity can be safely applied and thus no validation is required. The worst two cases that require a substantial user effort are the computation of a direct mapping between CIDR and Excel, and CIDR and Paragon.

6 Future Work and Conclusions

Our contribution is a mapping representation that captures explicit semantics that can be efficiently inverted and composed. Unlike morphisms, semantics are preserved during composition and only correct mapping correspondences are retained. $Unknown$ mappings formed during composition are highlighted for semi-automatic correction by the user. We have shown using experiments that the mappings are efficient to construct and the composition result is much improved as opposed to composing morphisms. This is important as the mappings are simple enough such that it is realistic that they could be semi-automatically generated, but also expressive enough to ensure that composition is well-defined and produces correct results. The composition algorithm is closed under the defined mapping relationships and composition rules and does not produce false

relationships. In addition, we have provided sound arguments that this representation system is able to isolate the hard composition cases, which significantly reduces user effort in finding bad matches and validating the entire mapping produced.

Our future efforts will strive to further confine the cases captured by the $Unknown$ relationship and to find a set of heuristics that would allow us to efficiently suggest relationships between model elements.

References

1. James F. Allen. Maintaining knowledge about temporal intervals. In *Communications of the ACM, v.26 n.11*, pages 832–843, 1983.
2. P. Bernstein. Applying Model Management to Classical Meta Data Problems. In *CIDR*, 2003.
3. R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: Discovering Complex Mappings between Database Schemas. In *SIGMOD Conference 2004*, pages 383–394, 2004.
4. Hong Hai Do and E. Rahm. COMA - A System for Flexible Combination of Schema Matching Approaches. In *VLDB*, pages 610–621, 2002.
5. E. Dragut and R. Lawrence. Composing mappings between schemas using a reference ontology. In *ODBASE*, pages 783–800, 2004.
6. J. Euzenat. An API for Ontology Alignment. In *International Semantic Web Conference*, pages 698–712, 2004.
7. R. Fagin, P. Kolaitis, L. Popa, and W. Tan. Composing schema mappings: Second-order dependencies to the rescue. In *PODS*, pages 83–94, 2004.
8. F. Giunchiglia, P. Shvaiko, and M. Yatskevich. S-Match: an Algorithm and an Implementation of Semantic Matching. *ESWS*, pages 61–75, 2004.
9. A. Halevy. Answering queries using views: A survey. *VLDB Journal*, 10(4):270–294, 2001.
10. Robin Hirsch and Ian Hodkinson. *Relation algebras by games*. North-Holland, Amsterdam, 2002.
11. K. Kotis and G. A. Vouros. The HCONE Approach to Ontology Merging. In *ESWS*, pages 137–151, 2004.
12. J. Madhavan, P. Bernstein, A. Doan, and A. Halvey. Corpus-based Schema Matching. In *ICDE*, 2005.
13. J. Madhavan, P. Bernstein, P. Domingos, and A. Halevy. Representing and reasoning about mappings between domain models. In *AAAI*, pages 80–86, 2002.
14. J. Madhavan, P. Bernstein, and E. Rahm. Generic Schema Matching with Cupid. In *VLDB*, pages 49–58, 2001.
15. S. Melnik, E. Rahm, and P. Bernstein. Rondo: A Programming Platform for Generic Model Management. In *SIGMOD Conference 2003*, pages 193–204, 2003.
16. L. Popa, Y. Velegrakis, R. Miller, M. Hernndez, and R. Fagin. Translating web data. In *VLDB*, pages 598–609, 2002.
17. R. Pottinger and P. Bernstein. Merging Models Based on Given Correspondences. In *VLDB*, pages 826–873, 2003.
18. E. Rahm and P. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
19. S. Spaccapietra and C. Parent. View integration: A step forward in solving structural conflicts. *IEEE Trans. Knowl. Data Eng.*, 6(2):258–274, 1994.
20. A. Tarski. On the calculus of relations. *J. Symbolic Logic.*, 6:73–89, 1941.
21. A. Tarski. Some methodological results concerning the calculus of relations. *J. Symbolic Logic.*, 18:188–189, 1953.
22. L. Xu and D. Embley. Discovering Direct and Indirect Matches for Schema Elements. In *DASFAA*, pages 39–46, 2003.