Composing Mappings between Schemas using a Reference Ontology

Eduard Dragut, Ramon Lawrence

IDEA Lab, Department of Computer Science, University of Iowa
Iowa City, IA, USA, 52242
{eduard-dragut, ramon-lawrence}@uiowa.edu
http://www.cs.uiowa.edu/∼rlawrenc/

Abstract. Large-scale database integration requires a significant cost in developing a global schema and finding mappings between the global and local schemas. Developing the global schema requires matching and merging the concepts in the data sources and is a bottleneck in the process. In this paper we propose a strategy for computing the mapping between schemas by performing a composition of the mappings between individual schemas and a reference ontology. Our premise is that many organizations have standard ontologies that, although they may not be suitable as a global schema, are useful in providing standard terminology and naming conventions for concepts and relationships. It is valuable to leverage these existing ontological resources to help automate the construction of a global schema and mappings between schemas. Our system semi-automates the matching between local schemas and a reference ontology then automatically composes the matchings to build mappings between schemas. Using these mappings, we use model management techniques to compute a global schema. A major advantage of this approach is that human intervention in validating matchings mostly occurs during the matching between schema and ontology. A problem is that matching schemas to ontologies is challenging because the ontology may only contain a subset of the concepts in the schema or may be more general than the schema. Further, the more complicated ontological graph structure limits the effectiveness of some matchers. Our contribution is showing how schema-to-ontology matchings can be used to compose mappings between schemas with high accuracy by adapting the COMA schema matching system to work with ontologies.

1 Introduction

Database integration is a challenging problem that has been extensively studied [1, 2] for many years. Automating integration has proven difficult because schemas do not always capture the necessary semantics to identify related concepts. Schema matching systems [3] are used to build mappings between schemas that are then used to construct an integrated view. Although good accuracy has been achieved by schema matching techniques, validating matches is difficult because a user must understand the semantics of both schemas. Further, if many schemas are matched together, this validation must be performed for each pair-wise matching. In an integration scenario, it is hard for a global integrator to understand the semantics of each schema to be integrated in order to validate matches. It is also difficult to define and maintain a global schema as it requires identifying all concepts in all databases.

Many organizations, especially biomedical organizations such as the National Cancer Institute (NCI) and National Institutes of Health (NIH), have been developing standard ontologies for their domains. These ontologies are not suitable as global schemas because they are more general than the domain being modeled or do not contain all the concepts required. However, they are useful in the matching process as they can be used as a reference ontology. The idea is to match each source to the domain ontology, and each schema-to-ontology match is validated by the database administrator. The advantage of this approach is that the administrator only needs to understand the semantics of their schema when validating matches. Schema-to-ontology matches can be used to build mappings to any schema that is also matched to the ontology by composing the schema-to-ontology matchings. The goal of this work is to use these pre-existing ontologies to automate schema matching and global view construction.

The challenge is that an existing ontology may not cover the domain exactly. Schema concepts that are not in the ontology will not be discovered during matching. The ontology may be more general and have many more concepts which reduces the matching accuracy. The more complicated ontological structure reduces the effectiveness of some matchers, specifically those that use names and paths. Our overall contribution is demonstrating how existing schema matching systems can be adapted for discovering schema-to-ontology matchings useful for ontology-based integration. The contributions of this work are:

– An algorithm for mapping ontologies into schema graphs for use with automatic schema matching systems such as COMA [4].
– A method for composing schema-to-ontology matchings to produce mappings between schemas.
– A model management [5] methodology for producing an integrated view using schema-to-ontology matchings. The integrated view is a federated view in the sense that it can be dynamically constructed from any number of schemas and may be site specific.
– An experimental evaluation demonstrating that schema-to-ontology matching can be achieved with good accuracy and that schema-to-schema mappings derived from these matchings can have similar accuracy to direct, pair-wise schema matching.

This work is different than other ontology-based integration approaches [6–8] as the schema-to-ontology matchings are generated semi-automatically. Generation of these matchings is a bottleneck to integration using ontologies. The schema matching on ontologies is different than other schema matching systems [3, 4, 9] that either perform schema-to-schema matching or ontology-to-ontology matching. Ontological matching has distinctive features that have received less attention in schema matching systems such as IS-A relationships, complex hierarchies, limited or no data instances, and no explicit keys and identifiers. Thus, schema-to-ontology matching deserves special attention as many existing matchers have poor performance in this environment.

The organization of this paper is as follows. Section 2 provides a brief discussion on related work on database integration and schema matching. The problem domain and overall approach is covered in Section 3. In Section 4, we describe how the COMA matching system [4] is used to match ontologies with schemas. Once schemas are individually matched to an ontology, composition is used to build mappings between schemas. Composing mappings from schema-to-ontology matchings is discussed in Section 5. Constructing a global view using model management techniques and schema-to-ontology matchings is covered in Section 6. The approach allows each client to produce its own "global view" by composing only the schema-to-ontology matchings for the sources required. Detailed performance experiments on the accuracy of schema-to-ontology matchings and mapping composition are discussed in Section 7. The paper closes with future work and conclusions.

2 Related Work

Ontologies have been used in various roles for database integration [1, 2]. An ontology may be used instead of a global schema such as in the Carnot project [7] that used the Cyc ontology [10]. The Carnot system required administrators to manually map their schema into the global ontology. Global queries were then posed on the ontology. The MOMIS system [6] semi-automates the construction of the global view by extracting and manually annotating schema using WordNet [11] as a shared lexical database. Using WordNet allows the system to detect lexicon relationships as well as structural relationships in the schemas. The challenge with using a large ontology like WordNet is that it is not specific to the domain and does not model relationships between entities. For example, although the concepts *Order* and *Date* will be in WordNet, the complex concept *OrderDate* (representing that an order has a date) will not. This forces the designer to map a schema element to many WordNet terms. There are other systems that use ontologies for integration [12] including ONTOBROKER [13] and OBSERVER [8]. OBSERVER performs integration using multiple existing ontologies by translating vocabulary that conflicts in different ontologies. The common challenge in these approaches is that the mappings must be *manually* determined between ontology and schema. The deployment of ontology-based integration approaches would be greatly aided by more automated mapping discovery techniques as discussed in this paper.

Ontologies are also used to improve the accuracy of schema matching. Several systems [4, 14] use WordNet or thesauri to detect synonym relationships and related concepts. Xu and Embley [15] used custom constructed ontologies to detect concepts by matching their data values to expected data values using regular expressions. In these systems, the ontology serves a supporting role in the matching, but is not directly involved in the process. There has also been work on matching ontologies [9] using algorithms similar to matching schemas [16]. Methods for merging ontologies given manual matchings [17] have also been performed. The PROMPT system [18] semi-automatically guides a user in merging ontologies. OntoBuilder [19] can merge ontologies extracted from web search interfaces. Ontologies have been used in the SCROL project [20] to detect semantic conflicts given manually specified mappings between the schemas and federated schema. There has been limited work on matching schemas to ontologies, where the reference ontology is an intermediary in the integration process (without being the entire global view).

Model management [5, 21] and schema matching [3] have been proposed to semi-automate database integration. The idea is to semi-automatically match schemas and then use these mappings to manipulate schemas using higher-level operators. A *match* is a correspondence between schema elements. A *mapping* between two schema elements is an expression that relates the two elements. A schema matching system will detect matchings between elements, but may not determine the mapping expression between them. Schema matching systems [4, 16, 14] use schema and instance level matchers to determine when elements in different schemas represent the same concept. The matchers may use linguistic information such as names and comments, schema information such as paths and constraints, and data instances. Most systems combine matchers into hybrid or composite matchers to improve the accuracy compared to individual matchers. Schema matchers that use data instances are not applicable for schema-to-ontology matching as discussed in this paper as the reference schemas are assumed to have no data instances. COMA [4] is a schema matching system that contains many matchers and is a flexible system for adding and combining matchers. COMA supports re-use of matches to improve matching accuracy.

Corpus-based matching [22] also re-uses matches and is similar in spirit to our proposed approach. In corpus-based matching, previous matches are archived into a Mapping Knowledge Base (MKB) that functions as a universal schema. When a concept is matched, it is matched to an existing concept in the universal schema or added to the schema. When two schemas are to be matched, each schema element is matched to the concepts in the MKB. If two elements from different schemas, match to the same MKB concept, they are predicted to match to each other. The MKB functions as an intermediary for the matching and learns classifiers for each universal schema element. Our approach using a reference ontology is similar as the ontology acts as a given (incomplete) universal schema of the domain. The difference is that the ontology is an accepted reference ontology available to the user during matching. The MKB is a hidden construct used by the system for matching. Our system allows the user control over the schema-to-ontology matching process and is more suitable to environments where users map to shared ontologies.

Overall, ontology-based integration systems can benefit from semi-automatic schema-to-ontology mapping algorithms and from an approach to build a global view using a reference ontology that may not model the domain perfectly.

3 System Architecture

The goal is to use a *pre-existing* reference ontology to semi-automate the construction of mappings between schemas. The reference ontology contains some of the concepts in the schemas, but may be incomplete or more general than the schemas. We assume that the ontology is more than a taxonomy as it should contain containment and general relationships between concepts. The ontology does not have instances, and thus should not be considered as a schema. Source schemas have an overlapping set of concepts, but are not identical in their modeling of the domain. Constructing a global view of these schemas is accomplished in a three step process:

- Independent matching of each schema to the ontology.
- Composing schema-to-ontology matches to produce schema-to-schema mappings.
- Merging the schemas to build a global view using the mappings.

It is useful to consider how the approach would be performed manually before examining how to automate it. First, a database administrator would match schema elements to ontological concepts. A schema element may not be in the ontology or may not match perfectly if it is more general or more specific than an ontological concept. Each schema element matches to zero or more ontological concepts. Since the administrator understands the schema, producing and validating matchings to the ontology is reasonable, although it does require the administrator to understand the pre-existing reference ontology. Composing the schema-to-ontology matchings produced independently by two administrators is straightforward. It is assumed that two schema elements match if they both map to the same ontological concept. Finally, given the schema mappings, applying a *Merge* operator as defined in the model management approach can be used to build the global view. Even with an entirely manual approach, matching to a reference ontology has the benefit that administrators only have to understand the semantics of their own schema and must only perform and validate one matching. The manual matching approach has been used in previous ontology-based integration systems where it is assumed that the ontology serves as an all-encompassing global view. It is not common for an ontology to model all domain concepts in a form suitable for use as a global view, but it is very common for pre-existing, shared, standard ontologies to be available for many domains.

Several complexities arise when automating the global view construction. It is valuable to re-use existing schema matching algorithms (such as COMA [4]), but this requires converting an ontology into a suitable form (Section 4). The matching algorithms will be less accurate as the ontology does not completely cover the domain and will model it in a different form. The composition to produce schema-to-schema mappings (Section 5) may create false matches or may miss matches when elements map to different ontological concepts or do not have any correspondences within the ontology. Merging schemas, even with mappings, is not fully automatic as the mappings are often imperfect and require user intervention [21]. We discuss these issues in the following sections.

4 Ontological Matching

Given the reference ontology, the first step is to convert it into a form suitable for schema matching. We use the COMA [4] schema matching system that models a schema as a rooted directed acyclic graph. A schema consists of a set of elements, such as relational tables and columns or XML elements and attributes. In COMA schema elements are represented by graph nodes connected by directed links of different types, such as containment and referential relationships.

We map ontologies that consist of collections of taxonomies and properties. The native format of the ontologies are ASCII files containing the concept definitions in OWL or DAML format. The translation into graphs is performed using an import filter that understands the definitions of concepts and properties in OWL or DAML specification. The ontology is converted to COMA graph format using an import tool developed using the *JENA* ontology parser[1]. Schemas and an ontology in the order domain are used as examples. The reference ontology is in Figure 1.

During the import, each ontology concept (class) becomes a node in the graph. For the properties (attributes) of each class, add a node to the graph and connect it to its class. Each class property has associated information such as a data type and cardinality that is stored as additional information with the node. This information is used by many schema matching algorithms. Properties that have both domain and range as concepts in the ontology (i.e. *shipTo*) are represented as nodes in the graph. Each of these nodes has a parent node that represents the class that is its domain and a child node that represents the class that is its range. A directed edge from the parent (domain) node to the new node is added to the graph as well as a directed edge from the new node to its child (range) node. In the current implementation we do not support properties that have a domain or range specified as intersection or union of concepts. IS_A relationships in the ontology are inserted as directed edges from the subclass node to the superclass node. In Figure 2 is an example of converting the ontological relationship *shipTo* between *PurchaseOrder* and *Organization* into a *shipTo* node and two directed edges.

After all the relationships (edges) are in the graph, a graph traversal is performed along IS_A links to make IS_A relationships explicit. COMA does not handle IS_A relationships, so these relationships are made explicit by having each subclass contain the properties of its superclasses. In the final step, top nodes are identified that are not contained in any other node. If there are multiple top nodes, then a new root node is added and all top nodes become its children. In Figure 3 is an example of making superclass properties (*Phone*, *Email*, *Fax*) explicit in the subclasses *Person* and *Organization*.

Once the ontology is converted into a schema graph, the COMA system will *automatically* match the schema to the ontology. The result is a schema-to-ontology matching. We define two approaches to generating these schema-to-ontology matchings and extend the COMA system to support them. The first

[1] http://jena.sourceforge.net/

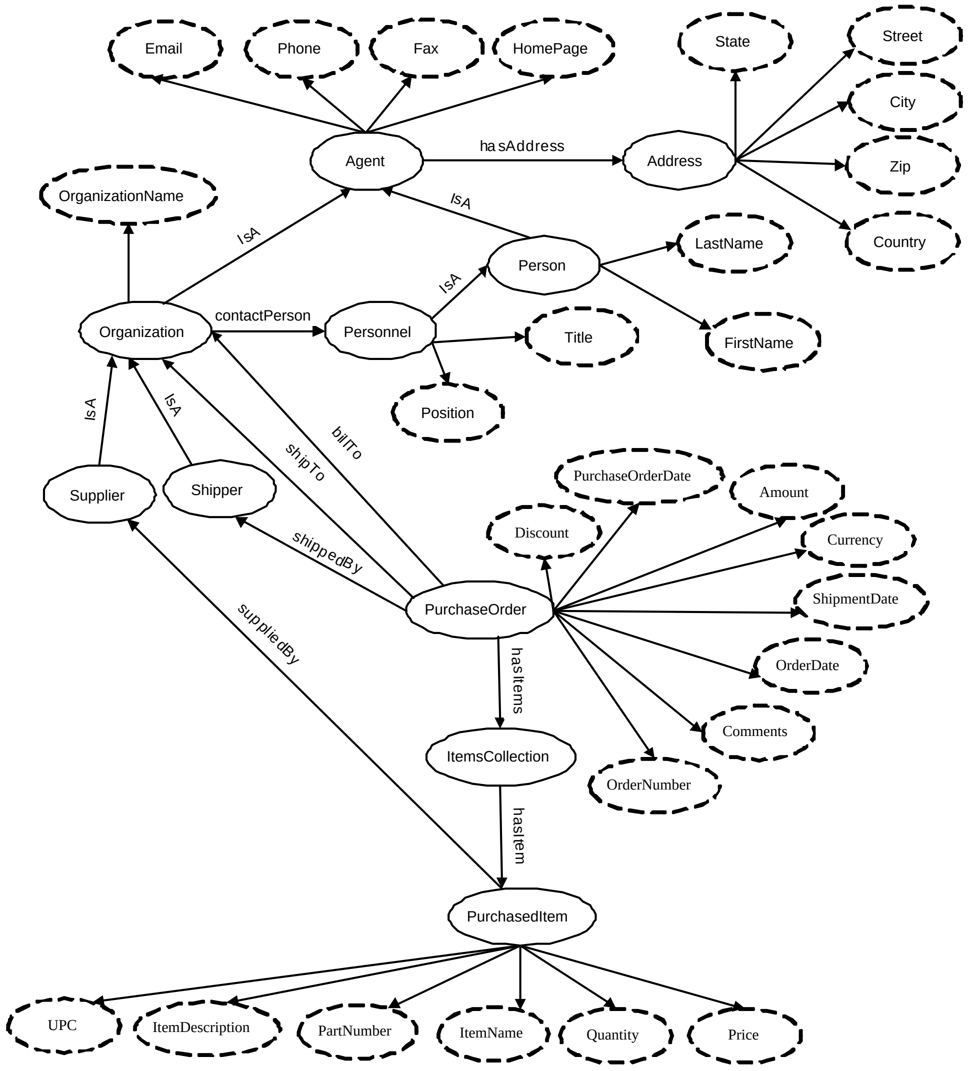

Fig. 1. Order Ontology

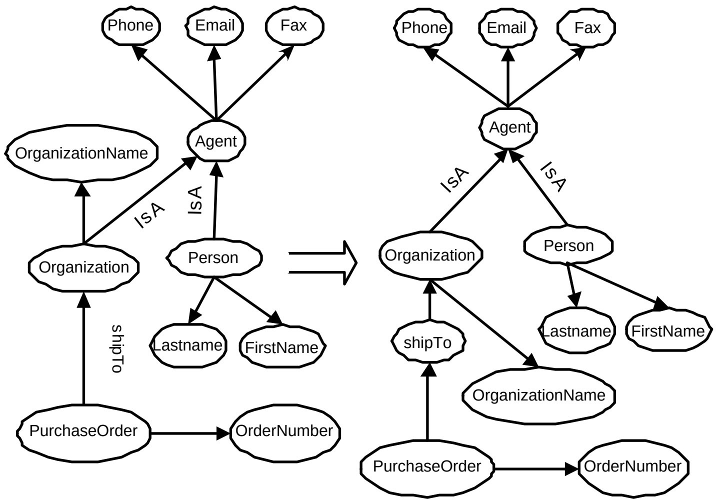

Fig. 2. Converting Relationships to Graph Format

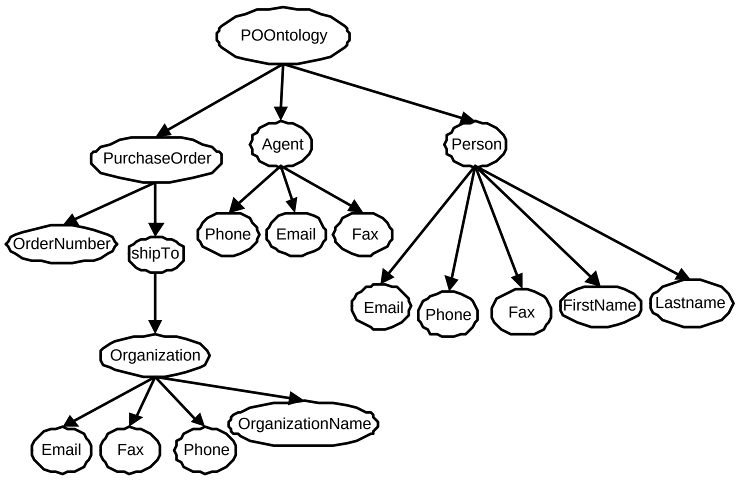

Fig. 3. Making IS-A Relationships Explicit

approach called `Max` generates up to one match between a schema element and its best matching ontological concept. A schema element may not have a match if the similarity is below a threshold. This may occur if the schema element concept is not in the ontology. The `Max` approach is good if the user will validate and improve matchings as it will generate only one match per schema element. Unfortunately, it will often miss matches where a schema element should map to two or more ontological concepts (such as *full name* matching to *first name* and *last name*) and may not always select the correct concept.

The second approach, called `noMax`, generates a variable number of matchings for each schema element. The advantage of `noMax` is that it allows a schema element to map to multiple ontological concepts and may allow mappings to be discovered that would have been discarded using `Max`. The problem with `noMax` is that it generates many incorrect mappings. An administrator seeking to create a "perfect" schema-to-ontology mapping would then spend a fair amount of time removing these invalid matches. If these invalid matches are left in the schema-to-ontology matching, the composition must then filter them out for the schema-to-schema mapping to be accurate.

The result after this stage is *automatically* constructed schema-to-ontology matchings. This is an improvement over previous ontology-based integration systems that required manual matching with the ontology. Except for the instance level matchers, we have used all the matchers included with COMA (e.g. Name, DataType, and NamePath). More details on how COMA automatically constructs matchings can be found in [4]. Our modifications include the algorithm to convert an ontology into a directed acyclic graph supported by COMA, and the `Max` and `noMax` approaches to filter ontological matches.

5 Composing Mappings

Mapping composition has been used in schema matching systems to reuse previous match results. In COMA [4], the *Compose* operation is used to build matchers that reuse previous match results. Re-using previous match results was shown to significantly improve the matching accuracy. In our system, the *Compose* operation is used to construct mappings between schemas by composing schema-to-ontology matchings. Two schema elements are assumed to be identical if they match the same ontological concept. Therefore we assume a transitive nature of the similarity relation between elements of schemas and the referenced ontology, i.e. if an element a of one schema is similar to an element o of the ontology and o is similar to an element b of the other schema, then a is also similar to b. If the schema-to-ontology matching is "perfect", then the schema-to-schema mapping will be very accurate. However, the schema-to-schema mapping will always miss matchings where an element in a schema does not have a matching ontological concept. The composition may also generate false matches if two or more schema elements map to the same ontological concept, but are not identical concepts.

In this paper, mappings are binary relations over the sets of elements of schemas and ontology, i.e. if $map : S \rightarrow O$ then map is a set of pairs $< l, r >$, where $l \in S$ and $r \in O$. This representation of mappings does not convey any semantics. Given two mappings map_1 that relates schema S_1 and the referenced ontology O and map_2 between schema S_2 and O, the *Compose* operation, denoted by $*$, produces a mapping *map* between the two schemas, as follows: $map = map_1 * map_2^{-1}$. That is given an element x of S_1, $(map_1 * map_2^{-1})(x) = (map_1(map_2^{-1}))(x)$ is an element in S_2, where map_2^{-1} denotes the inverse of map_2. The operation also computes the transitive similarity of schema elements. We adopt the COMA strategy of computing transitive similarity by taking the average of the two similarity values. For example, if <postalCode, Zip, 0.8> and <Zip, postCode, 0.7> *Compose* will produce <postalCode, postCode, 0.75>.

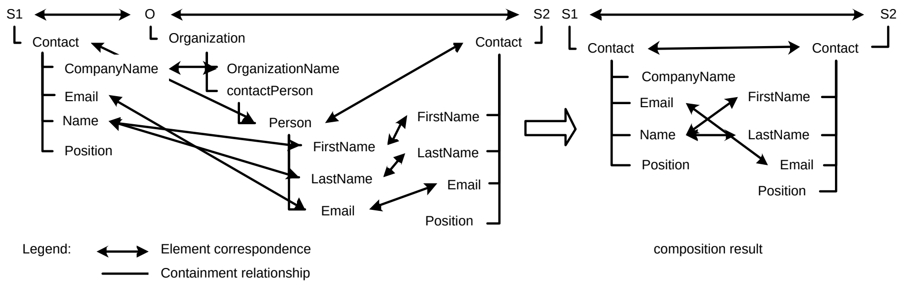

Fig. 4. Composition example

Figure 4 shows the general approach of deriving the match $S1 \leftrightarrow S2$ from composing the two match results $map_1 : S1 \leftrightarrow O$ and $map_2 : S2 \leftrightarrow O$. Since match results are binary relationships with similarity values, the *Compose* operator is defined as the natural join of two match results, yielding another match. The composition inherently filters out some of the bad schema-to-ontology matches. If the transitive similarity is below a threshold, the mapping produced is discarded. Thus, the difference between `Max` and `noMax` schema-to-ontology matching approaches is that the composition will discard fewer matches in the `Max` approach.

The example in Figure 4 illustrates some of the common problems of the *Compose* operation. Match composition may miss some correspondences, such as between *Position* of $S1$ and $S2$, due to the absence of a match counterpart in the ontology. In addition, composition may introduce unwanted correspondences when elements of the referenced ontology are related to several elements of the schemas. For example, in Figure 5, several contacts of schema $S1$ and $S2$ are matched to a generic contact person in the ontology. The composition result is six matches when only two are correct: S1.Billto.Contact=S2.InvoiceTo.Contact and S1.DeliverTo.Contact=S2.ShipTo.Contact.

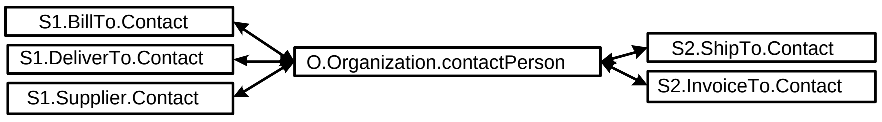

Fig. 5. Composition example with undesirable m:n matches

6 Global View Construction

In this section is an algorithm, called *GlobalView*, for computing the global view. The goal is to create a schema that represents all of the information expressed in n database schemas, $S_i, i = 1..n$. The algorithm is formulated using model management primitives and is initially described for two schemas and then generalized for n schemas. Model management is an approach to metadata-intensive applications that proposes a higher level of abstraction than current techniques [5]. Its main abstractions are models (e.g. schemas, interface definitions) and mappings between models. It offers such operators as Match, Merge, Extract, Delete, and Compose.

Consider a reference ontology O, two schemas $S1$ and $S2$, a mapping $S1\_O$ between $S1$ and O, and a mapping $S2\_O$ between $S2$ and O. The global view can be computed by:

1. Detecting similar objects in $S1$ and $S2$ using the *Compose* operator to compute a mapping between $S1$ and $S2$, called $S1\_S2$.
2. Given the mapping $S1\_S2$ computed in the previous step, using *Merge* operator to produce the integrated schema M and the mappings $S1\_M$ and $S2\_M$.
3. Using the *Compose* operator to compute a mapping between the newly created schema M and reference ontology O.

On the left-hand side of Figure 6 is a schematic representation of the process, where the rectangles denote schemas (e.g. the rectangles labeled S1, S2, M, and O) and the arcs between rectangles represent mappings between the schemas (e.g. the mapping between S1 and S2 is depicted as the labeled arc $S1\_S2$). The sequence of model management operations applied are:

```
operator GlobalView2($S1$, $S2$, $O$, $S1\_O$, $S2\_O$)
1. $S1\_S2 = S1\_O * \text{Invert}(S2\_O)$;
2. $< M, S1\_M, S2\_M > = \text{Merge}(S1, S2, S1\_S2)$;
3. $M\_O = \text{Invert}(S1\_M) * S1\_O + \text{Invert}(S2\_M) * S2\_O$;
4. return $< M, M\_O >$;
```

The merging of two schemas is driven by the mapping $S1\_S2$ computed using composition in Line 1. Observe that for the composition to be correct,

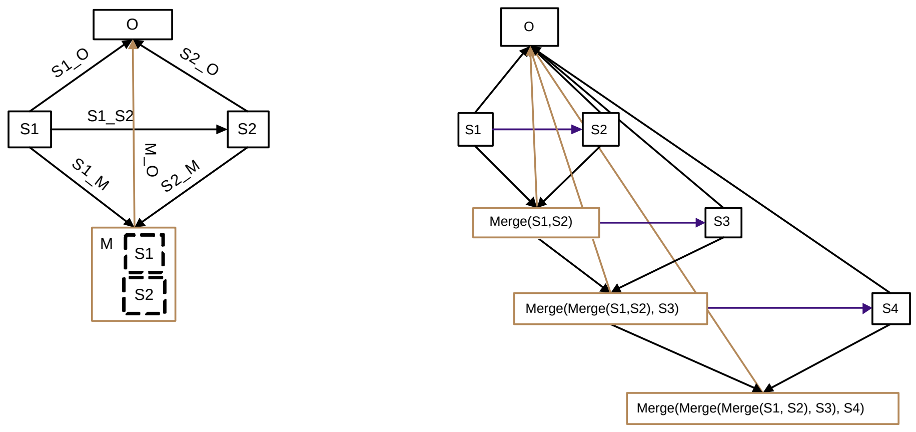

Fig. 6. Constructing a Global View using Model Management Operations

$S2\_O$ needs to be inverted (i.e. the domain and range of the mapping has to be swapped.) The global schema M is computed using the *Merge* operator that also produces two mappings $S1\_M$ and $S2\_M$ that relate M to the two original schemas. In Line 3, the mapping M_O is computed so that *GlobalView2* can be used in further merge operations. The output of the algorithm consists of pair $< M, M\_O >$, where M is the global schema over $S1$ and $S2$, and M_O is the mapping between new schema M and the referenced ontology O. The steps above are encapsulated as a new operator, called *GlobalView2*, that is re-used to compute the global view for N schemas. The general global schema composition algorithm for N sources is given in Figure 7. The iterative process of the computation of the global view using a reference ontology is depicted on the right-hand side of Figure 6 for $n = 4$.

Note that the integrated view construction algorithm is not a fully automated solution to the problem. Designer intervention may be required, especially when the intermediate output of the operations is only an approximate one. For example, with the current implementations of *Compose* operator it is very probable that false matches are suggested and that not all the correct matches are outputted. Merging is a semi-automatic process that requires human intervention and validation.

7 Experimental Study

We performed an experimental study to demonstrate the effectiveness of the approach. The five sample XML order schemas: CIDR, Excel, Noris, Paragon, and Apertum from www.biztalk.org used to evaluate COMA [4] were tested. These schemas are assigned numbers 1, 2, 3, 4, and 5 respectively. We constructed a reference order ontology (Figure 1) that models the order domain. This ontology

```
GlobalView($ArraySchemas$, $ArrayMappings$ ,$O$, $n$)
// ArraySchemas = source schemas, ArrayMappings = schema-to-ontology mappings
// O = reference ontology, n = number of source schemas
1. if n $\leq$ 0 then return empty schema;
2. if n = 1 then return ArraySchemas[0];
3. $S1$ = ArraySchemas[0];
4. $S2$ = ArraySchemas[1];
5. $map1$ = ArrayMappings[0];
6. $map2$ = ArrayMappings[1];
7. $< S, map >$ = GlobalView2($S1$, $S2$, $map1$, $map2$, $O$);
8. for(i = 2; i $\leq n - 1$; i++ )
9.        $S1 = S$;
10.       $map1 = map$;
11.       $S2$ = ArraySchemas[i];
12.       $map2$ = ArrayMappings[i];
13.       $< S, map >$ = GlobalView2($S1$, $S2$, $map1$, $map2$, $O$);
14. end for;
15. return $< S$, $map >$;
```

Fig. 7. Global View Construction Algorithm

has different structure than the schemas. For instance, the ontology uses IS-A whereas none of the sample schemas have IS-A relationships. The ontology does not have all concepts used in the schemas such as *unitOfMeasure*, *count*, and *VAT* information. Further, the ontology contains no ids or keys and does not model the order amounts, tax issues, and street addresses in as much detail as some schemas. We have used the correct mappings as given by COMA as ground-truth. As always there are some mappings that are open to interpretation which affect the results.

The first experiment is to determine the accuracy of the schema-to-ontology matching using both `Max` and `noMax`. The results are in Figure 8. The accuracy of schema-to-ontology matching is quite good. `Max` has precision of 75-80% and recall around 60%. Recall is lower as it misses some matchings that are not evaluated as the best. `noMax` has slightly better recall than `Max` but loses some precision as it generates many matchings where only one is correct. The overall is always positive for `Max` indicating that it saves effort over manual matching. For `noMax` the matching with schema 5 results in a negative overall because the schema contains the concept *Buyer* which is not in the ontology and gets incorrectly matched to several higher-level concepts in the ontology such as *Agent* and *Person*. Without the improvements such as expanding IS-A, the accuracy is very bad. Fortunately, we are willing to accept less accuracy in this case as the matching process will only be performed once and the administrator has full understanding of the semantics of their schema to detect and resolve mismatches. It is also important to note that perfect matching is not possible since the ontology may not cover the schema concepts exactly. The fraction of

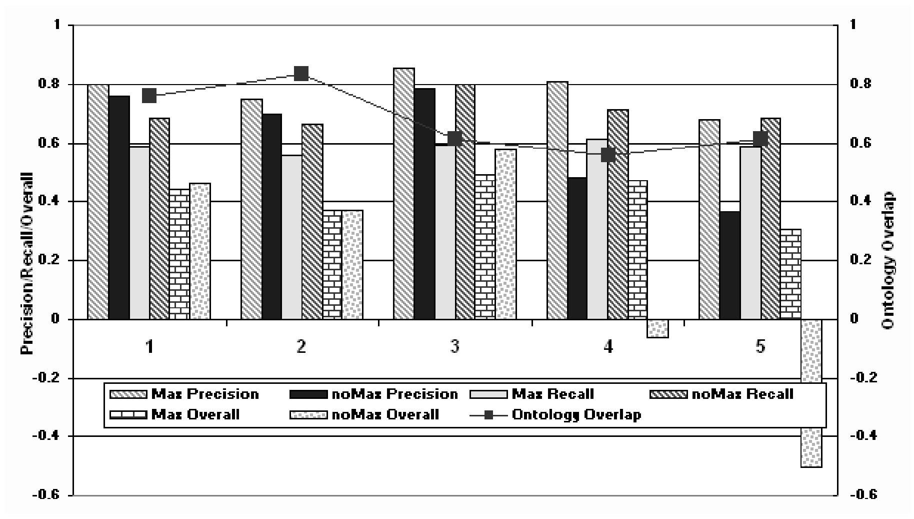

Fig. 8. Schema-to-Ontology Matching

schema elements that can be manually matched to the ontology is also shown in Figure 8. This fraction represents the schema overlap with the ontology and is the best possible match performance that can be achieved. A schema element is considered to match to the ontology even if it is not a perfect match. The last three schemas have relatively poor matching with the ontology (about 60% of the concepts are present in the ontology in some form). For example, approximately 60% of the elements in schema 4 can be matched to the ontology. `noMax` has a recall of 70%, so it finds ontological matches for 42% of all elements in schema 4. The statistic `Overall` is defined as $Recall * (2 - 1/Precision)$, and is a common measure used to evaluate schema matching systems.

The schema-to-ontology matchings are composed to produce schema-to-schema mappings and compared to the results generated by COMA. Even with average accuracy of schema-to-ontology matchings, the results (Figure 9) are in many cases comparable to direct schema matchings using COMA. The precision is high for both approaches. The weakness, especially for the `Max` approach, is recall as it only selects the best matching and discards all others. For the `noMax` case, the composition correctly filters out many mismatches. The overall statistics are very good, and are often close to direct schema matchings performed with COMA. The results are very good even when compared to perfect manual schema-to-ontology matchings, which themselves do not result in perfect schema-to-schema mappings as the ontology does not cover all concepts.

Many of the inaccuracies result from very simple modeling issues. For example, when one database has 4 fields: *Street1, Street2, Street3, Street4*, do all these fields map to an ontological concept of *Street*? If they all map to *Street* in the ontology, then the composition will generate one correct and numerous in-

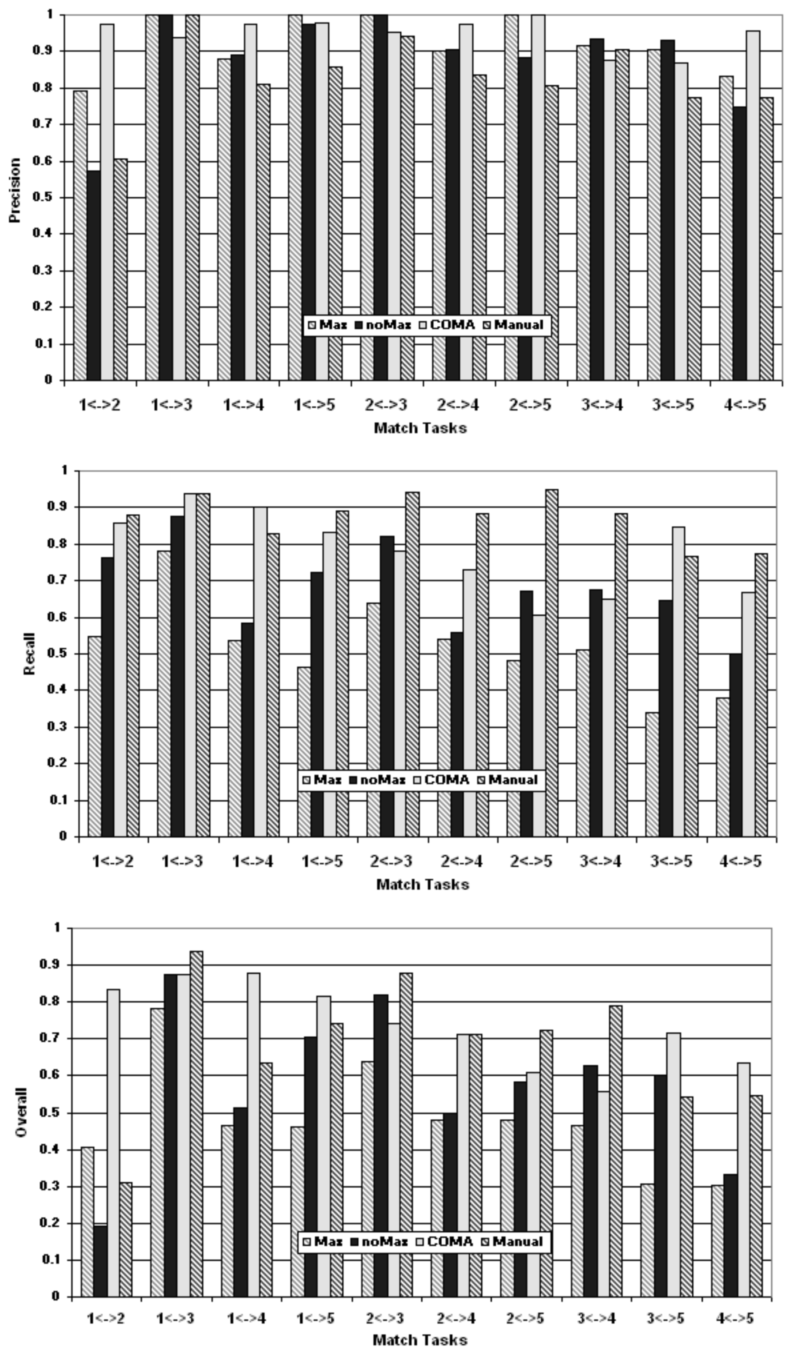

Fig. 9. Schema-to-Schema Mapping Statistics

correct matches with two schemas that represent street in this way as discussed in Section 5. This is the reason for the poor performance between schemas 1 and 2. Matchings involving schemas 3, 4, and 5 have lower accuracy due to their relative poor overlap with the ontology. However, the performance is still very good and sometimes is as good or better than COMA. Mapping schemas 4 to 5 is poor because the concept of *Buyer* in schema 5 is not in the ontology and gets incorrectly mapped to concepts in schema 4. This results in many false matchings after composition.

The matching accuracy can be further improved by using the schema-to-schema mappings generated as existing matches that are re-used when directly matching the schemas. The matches missed during composition because the concepts were not in the ontology can be correctly matched when the schemas are matched directly. An experiment is performed that allows COMA to re-use the schema-to-schema mappings found by composing schema-to-ontology matchings when directly computing pair-wise schema matches. Results (Figure 10) were determined when the schema-to-ontology matchings were manually specified, and when they were generated automatically using the `Max` and `noMax` algorithms. In almost all cases, the overall performance is near or better than using COMA alone to directly match schemas. This shows that there is benefit to building these schema-to-ontology matchings for use in integration as they are relatively easy to construct and validate and can be re-used across matching tasks. Although manual mappings are better, automatically generated mappings also add value. Re-using automatically generated mappings is not perfect because false matches introduced through the composition (as in matching schemas 1 and 2) negatively affect the result.

Overall, these experiments demonstrate that schema-to-ontology matching has additional challenges over schema-to-schema matching. Ontologies have more complex structure that confuses matchers like `NamePath`, and existing match algorithms are very sensitive to the degree of overlap and similar structure of schemas. In all cases, the overall measure was positive indicating that manual match effort is saved by using the approach. The good mapping accuracy allows the global view construction algorithm to construct quality global schemas with limited user input. This results in significant savings in designer effort in building the global schema for integrated systems.

8 Future Work and Conclusions

In this work we have provided algorithms for automatically constructing global views for integrated systems using schema-to-ontology matchings. These algorithms are useful for previous ontology-based integration approaches that had to manually generate such matchings and required the global ontology to completely model the entire domain. The experimental results demonstrate that the ontology does not have to perfectly overlap the integration domain for it to be useful in schema matching and global view construction. This allows pre-existing ontologies to be used for integration. By using semi-automatic matching tech-

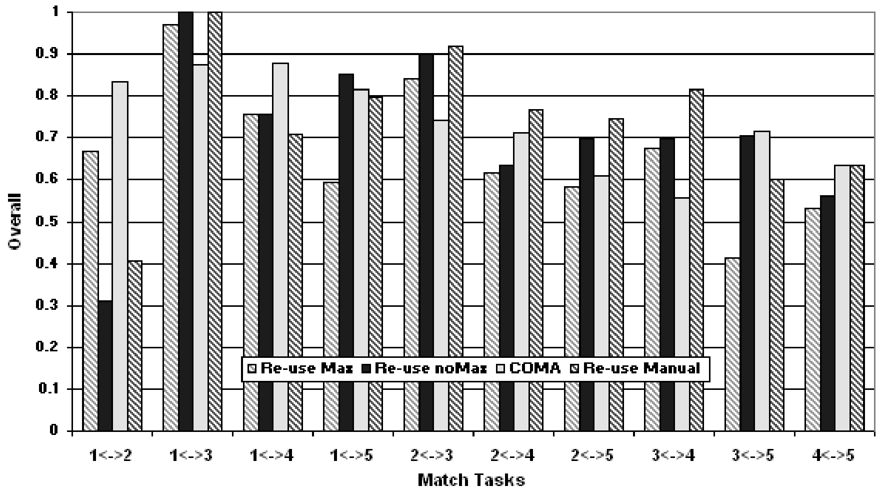

Fig. 10. Direct Schema-to-Schema Matching with Matching Re-use

niques developed for relational schemas, the overhead of manual matching to the ontology is avoided. We have shown how ontologies can be converted into a form suitable for use with existing relational matchers and demonstrated how the approach achieves high accuracy in finding schema-to-schema mappings.

Future work involves improving the composition to handle mismatches due to multiple matches to the same ontological concept or to different concepts in a IS-A hierarchy. This may involve using more sophisticated matches such as sub-concept and super-concept matches.

References

1. Batini, C., Lenzerini, M., Navathe, S.: A Comparative Analysis of Methodologies for Database Schema Integration. ACM Computing Surveys **18** (1986) 323–364
2. Sheth, A., Larson, J.: Federated Database Systems for Managing Distributed, Heterogenous and Autonomous Databases. ACM Computing Surveys **22** (1990) 183–236
3. Rahm, E., Bernstein, P.: A survey of approaches to automatic schema matching. VLDB Journal **10** (2001) 334–350
4. Do, H.H., Rahm, E.: COMA - A System for Flexible Combination of Schema Matching Approaches. In: VLDB. (2002) 610–621
5. Bernstein, P.: Applying Model Management to Classical Meta Data Problems. In: CIDR. (2003)
6. Beneventano, D., Bergamaschi, S., Guerra, F., Vincini, M.: Synthesizing an Integrated Ontology. IEEE Internet Computing **7** (2003) 42–51
7. Collet, C., Huhns, M., Shen, W.M.: Resource Integration Using a Large Knowledge Base in Carnot. IEEE Computer **24** (1991) 55–62

8. Mena, E., Illarramendi, A., Kashyap, V., Sheth, A.: OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies. Distributed and Parallel Databases **8** (2000) 223–271
9. Doan, A., Madhavan, J., Domingos, P., Halevy, A.: Learning to Map between Ontologies on the Semantic Web. In: Proceedings of the 11th International Conference on the World Wide Web. (2002) 662–673
10. Lenat, D., Guha, R., Pittman, K., Pratt, D., Shepherd, M.: Cyc: Towards programs with common sense. Communications of the ACM **33** (1990) 30–49
11. Miller, G., Beckwith, R., Fellbaum, C., Gross, D., Miller, K.: Five Papers on WordNet. Technical Report CSL Report 43, Cognitive Systems Laboratory, Princeton University (1990)
12. Tzitzikas, Y., Constantopoulos, P., Spyratos, N.: Mediators over Ontology-Based Information Sources. In: WISE. (2001) 31–40
13. Decker, S., Erdmann, M., Studer, R.: ONTOBROKER: Ontology based access to distributed and semi-structured information. In: Database Semantics - Semantic Issues in Multimedia Systems. Volume 138 of IFIP Conference Proceedings., Kluwer (1998)
14. Madhavan, J., Bernstein, P., Rahm, E.: Generic Schema Matching with Cupid. In: VLDB. (2001) 49–58
15. Xu, L., Embley, D.: Discovering Direct and Indirect Matches for Schema Elements. In: DASFAA. (2003) 39–46
16. Doan, A., Domingos, P., Halevy, A.: Reconciling schemas of disparate data sources: a machine-learning approach . In: Proceedings of the ACM SIGMOD Conference on Management of Data. (2001) 509–520
17. Pottinger, R., Bernstein, P.: Merging Models Based on Given Correspondences. In: VLDB. (2003) 826–873
18. Noy, N., Musen, M.: PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In: AAAI/IAAI. (2000) 450–455
19. Gal, A., Modica, G., Jamil, H.: OntoBuilder: Fully Automatic Extraction and Consolidation of Ontologies from Web Sources. In: ICDE. (2004) 853
20. Ram, S., Park, J.: Semantic Conflict Resolution Ontology (SCROL): An Ontology for Detecting and Resolving Data and Schema-Level Semantic Conflicts. IEEE Trans. Knowl. Data Eng. **16** (2004) 189–202
21. Melnik, S., Rahm, E., Bernstein, P.: Rondo: A Programming Platform for Generic Model Management. In: SIGMOD. (2003) 193–204
22. Madhavan, J., Bernstein, P., Chen, K., Halvey, A., Shenoy, P.: Corpus-based Schema Matching. In: Workshop on Information Integration on the Web (IJCAI03). (2003)