Ontology mediation, merging and aligning

Jos de Bruijn Marc Ehrig Cristina Feier
Francisco Martín-Recuerda François Scharffe
Moritz Weiten

May 20, 2006

Abstract

Ontology mediation is a broad field of research which is concerned with determining and overcoming differences between ontologies in order to allow the reuse of such ontologies, and the data annotated using these ontologies, throughout different heterogeneous applications.

Ontology mediation can be subdivided into three areas: *ontology mapping*, which is mostly concerned with the representation of correspondences between ontologies; *ontology alignment*, which is concerned with the (semi-)automatic discovery of correspondences between ontologies; and *ontology merging*, which is concerned with creating a single new ontology, based on a number of source ontologies.

This chapter reviews the work which has been done in the three mentioned areas and proposes an integrated approach to ontology mediation in the area of knowledge management. A language is developed for the representation of correspondences between ontologies. An algorithm, which generalizes current state-of-the-art alignment algorithms, is developed for the (semi-)automated discovery of such mappings. A tool is presented for browsing and editing ontology mappings. An ontology mapping can be used for a variety of different tasks, such as transforming data between different representations and querying different heterogeneous knowledge bases.

1 Introduction

On the Semantic Web, data is envisioned to be annotated using ontologies. Ontologies convey background information which enriches the description of the data and which makes the context of the information more explicit. Because ontologies are *shared* specifications, the same ontologies can be used for the annotation of multiple data sources, not only Web pages, but also collections of XML documents, relational databases, et cetera. The use of such shared terminologies enables a certain degree of inter-operation between these data sources. This, however, does not solve the integration problem completely, because it cannot be expected that all individuals and organizations on the Semantic Web will ever agree on using one common terminology or ontology (Visser

& Cui 1998, Uschold 2000). It can be expected that many different ontologies will appear and, in order to enable inter-operation, differences between these ontologies have to be reconciled. The reconciliation of these differences is called *ontology mediation.*

Ontology mediation enables reuse of data across applications on the Semantic Web and, in general, cooperation between different organizations. In the context of semantic knowledge management, ontology mediation is especially important to enable *sharing* of data between heterogeneous knowledge bases and to allow applications to *reuse* data from different knowledge bases. Another important application area for ontology mediation is that of Semantic Web Services. In general, it cannot be assumed that the requester and the provider of a service use the same terminology in their communication and thus mediation is required in order to enable communication between heterogeneous business partners.

We distinguish two principled kinds of ontology mediation: *ontology mapping* and *ontology merging.* With ontology mapping, the correspondences between two ontologies are stored separately from the ontologies and are thus not part of the ontologies themselves. The correspondences can be used for, for example, querying heterogeneous knowledge bases using a common interface or transforming data between different representations. The (semi-)automated discovery of such correspondences is called *ontology alignment.*

When performing *ontology merging*, a new ontology is created which is the union of the source ontologies. The merged ontology captures all the knowledge from the original ontologies. The challenge in ontology merging is to ensure that all correspondences and differences between the ontologies are reflected in the merged ontology.

Summarizing, *ontology mapping* is mostly concerned with the representation of correspondences between ontologies; *ontology alignment* is concerned with the discovery of these correspondences; and *ontology merging* is concerned with creating the union of ontologies, based on the correspondences between the ontologies. We provide an overview of the main approaches in ontology merging, ontology mapping and ontology alignment in Section 2.

After the survey in Section 2 we present a practical approach to ontology mediation where we describe a language to specify ontology mappings, an alignment method for semi-automatically discovering mappings, a graphical tool for browsing and creating mappings in a user friendly way, in Section 3.

We conclude with a summary in Section 4.

2 Approaches in ontology mediation

In this section we give an overview of some of the major approaches in ontology mediation, particularly focusing on ontology mapping, alignment and merging.

A major issue of all of these approaches is the location and specification of the overlap and the mismatches between concepts, relations, and instances

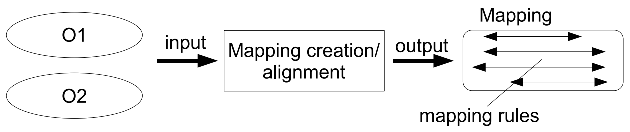


Figure 1: Ontology Mapping

in different ontologies. In order to achieve a better understanding of the mismatches which all these approaches are trying to overcome, we give an overview of the mismatches which might occur between different ontologies, based on the work by Klein (2001), in Section 2.1.

We survey number of representative approaches for ontology mapping, ontology alignment, and ontology merging in Sections 2.2, 2.3 and 2.4, respectively. For more elaborate and detailed surveys we refer the reader to (Shvaiko & Euzenat 2005, Noy 2004, Doan & Halevy 2005, Kalfoglou & Schorlemmer 2003).

2.1 Ontology mismatches

The two basic types of ontology mismatches are: (1) *Conceptualization mismatches*, which are mismatches of different conceptualizations of the same domain and (2) *Explication mismatches*, which are mismatches in the way a conceptualization is specified.

Conceptualization mismatches fall in two categories. A *scope mismatch* occurs when two classes have some overlap in their extensions (the set s of instances), but the extensions are not exactly the same (e.g., the concepts Student and TaxPayer). There is a mismatch in the *model coverage and granularity* if there is (a) a difference in the part of the domain that is covered by both ontologies (e.g., the ontologies of university employees and students) or (b) the level of detail with which the model is covered (e.g., one ontology might have one concept Person whereas another ontology distinguishes between YoungPerson, MiddleAgedPerson and OldPerson).

Explication mismatches fall in three categories. There is (1) a mismatch in the *style of modeling* if either (a) the *paradigm* used to specify a certain concept (e.g., time) is different (e.g., intervals versus points in time) or (b) the way the *concept is described* differs (e.g., using subclasses versus attributes to distinguish groups of instances). There is a (2) *terminological mismatch* when two concepts are equivalent, but they are represented using different names (*synonyms*) or when the same name is used for different concepts (*homonyms*). Finally, an (3) *encoding mismatch* occurs when values in different ontologies are encoded in a different way (e.g., using kilometers versus miles for a distance measure).

2.2 Ontology Mapping

An *ontology mapping* is a (declarative) specification of the semantic overlap between two ontologies; it is the output of the mapping process (see Figure 1).

The correspondences between different entities of the two ontologies are typically expressed using some axioms formulated in a specific mapping language. The three main phases for any mapping process are: (1) mapping discovery, (2) mapping representation, and (3) mapping exploitation/execution. In this section we survey a number existing approaches for ontology mapping, with a focus on the mapping representation aspect.

A common tendency among the ontology mapping approaches is the existence of an ontology of mappings (e.g., MAFRA (Maedche, Motik, no Silva & Volz 2002), RDFT (Omelayenko 2002)), which constitutes the vocabulary for the representation of mappings.

MAFRA (MApping FRAmework for distributed ontologies) (Maedche et al. 2002) supports the interactive, incremental and dynamic ontology mapping process, where the final purpose of such a process is to support instance transformation. It addresses all the phases of the mapping process: *lift & normalization* (lifting the content of the ontologies to RDF-S and normalization of their vocabularies by eliminating syntactical and lexical differences), *similarity* (computation of the similarities between ontology entities as a support for mapping discovery), *semantic bridging* (establishing correspondences between similar entities, in the form of so-called semantic bridges - defining the mapping), *execution* (exploiting the bridges/mapping for instance transformation), and *post-processing* (revisiting the mapping specification for improvements).

We will focus in the following on the representation of mappings using semantic bridges in MAFRA. The semantic bridges are captured in the Semantic Bridging Ontology (SBO). SBO is a taxonomy of generic bridges; instances of these generic bridges, called *concrete bridges*, constitute the actual concrete mappings. We give an overview of the dimensions along which a bridge can be described in MAFRA, followed by a shallow description of the classes of SBO which allow one to express such bridges.

A bridge can be described along five dimensions:

- *entity dimension*: pertains to the entities related by a bridge which may be concepts (modeling classes of objects in the real world), relations, attributes and extensional patterns (modeling the content of instances).
- *cardinality dimension*: pertains to the number of ontology entities at both sides of the semantic bridge (usually 1:n or m:1, m:n is seldom required and it can be usually decomposed into m:1:n)
- *structural dimension*: pertains to the way elementary bridges may be combined into a more complex bridge (relations that may hold between bridges: *specialization*, *alternatives*, *composition*, *abstraction*)
- *transformation dimension*: describes how instances are transformed by means of an associated transformation function.

- *constraint dimension*: allows one to express conditions upon whose fulfilment the bridge evaluation depends. The transformation rule associated with the bridge is not executed unless these conditions hold.

The abstract class `SemanticBridge` describes a generic bridge, upon which there are no restrictions regarding the entity types that the bridge connects or the cardinality. For supporting *composition*, this class has defined a relation `hasBridge`. The class `SemanticBridgeAlt` supports the *alternative* modeling primitive by grouping several mutual exclusive semantic bridges. The abstract class `SemanticBridge` is further specialized in the SBO according to the entity type, the ontology defining as subclasses of this class: `RelationBridge`, `ConceptBridge`, `AttributeBridge`. `Rule` is a class for describing generic rules. `Condition` and `Transformation` are its subclasses which are responsible for describing the condition necessary for the execution of a bridge, and the transformation function of a bridge, respectively. The `Service` class maps the bridge parameters with the transformation procedure arguments and call to procedures.

RDFT (Omelayenko 2002) is a mapping meta-ontology for mapping XML DTDs to/and RDF schemas specially targeted for business integration tasks. The business integration task in this context is seen as a service integration task, where each enterprise is represented as a Web service specified in WSDL. A conceptual model of WSDL was developed based on RDF Schema extended with the temporal ontology PSL. Service integration is reduced to concept integration, RDFT mapping specific concepts such as events, messages, vocabularies, and XML-specific parts of the conceptual model.

The most important class of the meta-ontology is `Bridge`, which enables one to specify correspondences between one entity and a set of entities or vice versa, depending on the type of the bridge: *one-to-many* or *many-to-one*. The relation between the source and target components of a bridge can be an `EquivalentRelation` (states the equivalence between the two components) or a `VersionRelation` (states that the target set of elements form a later version of the source set of elements, assuming identical domains for the two). This is specified via the bridge property `Relation`. Bridges can be categorized in:

- `RDFBridge`s, which are bridges between RDF Schema entities. These can be (`Class2Class` or `Property2Property`) bridges.
- `XMLBridge`s, which are bridges between XML tags of the source/target DTD and the target/source RDF Schema entities. These can be `Tag2Class`, `Tag2Property`, `Class2Tag`, or `Property2Tag` bridges.
- `Event2Event` bridges, which are bridges that connect two events pertaining to different services. They connect instances of the meta-class `mediator:Event`.

Collections of bridges which serve a common purpose are grouped in a `map`. When defined in a such a way, as a set of bridges, mappings are said to be *declarative*, while *procedural* mappings can be defined by means of an XPath expression transforming instance data.

C-OWL Another perspective on ontology mapping is given by Context OWL (C-OWL) (Bouquet, Giunchiglia, van Harmelen, Serafini & Stuckenschmidt 2004) which is a language that extends the ontology language OWL (Dean & Schreiber 2004) both syntactically and semantically in order to allow for the representation of contextual ontologies. In this vision, the term *contextual ontology* refers to the fact that the contents of the ontology are kept local and they can be mapped with the contents of other ontologies via explicit mappings (bridge rules) to allow for a controlled form of global visibility. This is opposed to the OWL importing mechanism where a set of local models is globalized in a unique shared model.

Bridge rules allow to connect entities (concepts, roles or individuals) from different ontologies that subsume one another, are equivalent, are disjoint or have some overlap. A C-OWL mapping is a set of bridges between two ontologies. A set of OWL ontologies together with mappings between each of them is called a context space.

The local models semantics defined for C-OWL, as opposed to the OWL global semantics, considers that each context uses a local set of models and a local domain of interpretation. Thus, it is possible to have ontologies with contradicting axioms, or unsatisfiable ontologies without the entire context space being unsatisfiable.

2.3 Ontology Alignment

Ontology alignment is the process of discovering similarities between two source ontologies. The result of a matching operation is a specification of similarities between two ontologies. Ontology alignment is generally described as the application of the so-called *Match* operator (cf. (Rahm & Bernstein 2001)). The input of the operator is a number of ontology and the output is a specification of the correspondences between the ontologies.

There are many different algorithms which implement the match operator. These algorithms can be generally classified along two dimensions. On the one hand there is the distinction between schema-based and instance-based matching. A schema-based matcher takes different aspects of the concepts and relations in the ontologies and uses some similarity measure to determine correspondence (e.g., (Noy & Musen 2000*b*)). An instance-based matcher takes the instances which belong to the concepts in the different ontologies and compares these to discover similarity between the concepts (e.g., (Doan, Madhaven, Domingos & Halevy 2004)). On the other hand there is the distinction between element-level and structure-level matching. An element-level matcher compares properties of the particular concept or relation, such as the name, and uses these to find similarities (e.g., (Noy & Musen 2000*b*)). A structure level matcher compares the structure (e.g., the concept hierarchy) of the ontologies to find similarities (e.g., (Noy & Musen 2000*a*, Giunchiglia & Shvaiko 2004)). These matchers can also be combined (e.g., (Ehrig & Staab 2004, Giunchiglia, Shvaiko & Yatskevich 2004)). For example, Anchor-PROMPT (Noy & Musen 2000*a*), a structure-level matcher, takes as input an initial list of similarities between concepts. The algorithm is then used to find additional similarities, based on the initial similarities and the structure of the ontologies. For a more detailed classification of alignment techniques we refer to (Shvaiko & Euzenat 2005). In the following, we give an overview of those approaches.

Anchor-PROMPT (Noy & Musen 2000*a*) is an algorithm which aims to augment the results of matching methods which only analyze local context in ontology structures, such as PROMPT (Noy & Musen 2000*b*), by finding additional possible points of similarity, based on the structure of the ontologies. The algorithm takes as input two pairs of related terms and analyzes the elements which are included in the path that connects the elements of the same ontology with the elements of the equivalent path of the other ontology. So, we have two paths (one for each ontology) and the terms that comprise these paths. The algorithm then looks for terms along the paths that might be similar to the terms of the other path, which belongs to the other ontology, assuming that the elements of those paths are often similar as well. These new potentially related terms are marked with a similarity score which can be modified during the evaluation of other paths in which these terms occur. Terms with high similar scores will be presented to the user to improve the set of possible suggestions in, for example, a merging process in PROMPT.

GLUE (Doan et al. 2004, Doan, Domingos & Halevy 2003) is a system which employs machine learning technologies to semi-automatically create mappings between heterogeneous ontologies based on instance data, where an ontology is seen as a taxonomy of concepts. GLUE focuses on finding 1-to-1 mappings between concepts in taxonomies, although the authors say that extending matching to relations and attributes and involving more complex mappings (such as 1-to-n and n-to-1 mappings) is the subject of ongoing research.

The similarity of two concepts A and B in the two taxonomies O_1 and O_2 is based on the sets of instances that overlap between the two concepts. In order to determine whether an instance of concept B is also an instance of concept A, first a classifier is built using the instances of concept A as the training set. This classifier is now used to classify the instances of concept B. The classifier then decides for each instance of B, whether it is also an instance of A or not.

Based on these classifications, four probabilities are computed, namely, $P(A,B)$, $P(\overline{A},B)$, $P(A,\overline{B})$ and $P(\overline{A},\overline{B})$, where, for example, $P(A,\overline{B})$ is the probability that an instance in the domain belongs to A, but not to B. These four probabilities can now be used to compute the *joint probability distribution* for the concepts A and B, which is a user supplied function with these four probabilities as parameters.

Semantic Matching (Giunchiglia & Shvaiko 2004) is an approach to matching classification hierarchies. The authors implement a *Match* operator that takes two graph-like structures (e.g. database schemas or ontologies) as input and produces a mapping between elements of the two graphs that semantically correspond to each other.

Giunchiglia & Shvaiko (2004) have argued that almost all earlier approaches to schema and ontology matching have been *syntactic* matching approaches, as opposed to *semantic* matching. In syntactic matching, the labels and sometimes the syntactical structure of the graph is matched and typically some similarity coefficient $[0, 1]$ is obtained, which indicates the similarity between the two nodes. Semantic Matching computes a set-based relation between the nodes, taking into account the meaning of each node; the semantics of a node is determined by the label of that node and the semantics of all the nodes which are higher in the hierarchy. The possible relations returned by the Semantic Matching algorithm are *equality* ($=$), *overlap* ($\cap$), *mismatch* ($\perp$), *more general* ($\subseteq$) or *more specific* ($\supseteq$). The correspondence of the symbols with set theory is not a coincidence, since each concept in the classification hierarchies represents a set of documents.

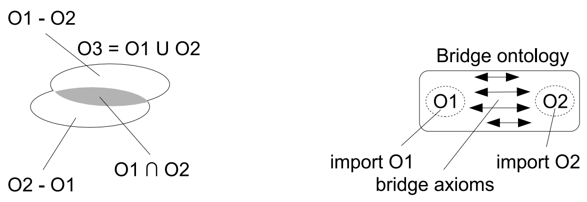
QOM - Quick Ontology Mapping (Ehrig & Staab 2004, Ehrig & Sure 2004) was designed to provide an efficient matching tool for on-the-fly creation of mappings between ontologies.

In order to speed up the identification of similarities between two ontologies, QOM does not compare all entities of the first ontology with all entities of the second ontology, but uses heuristics (e.g., similar labels) to lower the number of candidate mappings, i.e., the number of mappings to compare. The actual similarity computation is done by using a wide range of similarity functions, such as string similarity.

Several of such similarity measures are computed, which are all input to the similarity aggregation function, which combines the individual similarity measures. QOM applies a so-called sigmoid function, which emphasizes high individual similarities and de-emphasizes low individual similarities. The actual correspondences between the entities in the ontologies are extracted by applying a threshold to the aggregated similarity measure. The output of one iteration can be used as part of the input in a subsequent iteration of QOM in order to refine the result. After a number of iterations, the actual table of correspondences between the ontologies is obtained.

2.4 Ontology merging

Ontology merging is the creation of one ontology from two or more source ontologies. The new ontology will unify and in general replace the original source ontologies. We distinguish two distinct approaches in ontology merging. In the first approach the input of the merging process is a collection of ontologies and the outcome is one new, merged, ontology which captures the original ontologies (see Figure 2(a)). A prominent example of this approach is PROMPT (Noy &



(a) Complete Merge (b) Bridge Ontology

Figure 2: Output of the merging process

Musen 2000*b*), which is an algorithm and a tool for interactively merging ontologies. In the second approach the original ontologies are not replaced, but rather a 'view', called *bridge ontology*, is created which imports the original ontologies and specifies the correspondences using *bridge axioms*. OntoMerge (Dou, McDermott & Qi 2002) is a prominent example of this approach. OntoMerge facilitates the creation of a 'bridge' ontology which imports the original ontologies and relates the concepts in these ontologies using a number of *bridge axioms*. We describe the PROMPT and OntoMerge approaches in more detail below.

PROMPT (Noy & Musen 2000*b*) is an algorithm and an interactive tool for the merging two ontologies. The central element of PROMPT is the algorithm which defines a number of steps for the interactive merging process:

1. Identify merge candidates based on class-name similarities. The result is presented to the user as a list of potential merge operations.

2. The user chooses one of the suggested operations from the list or specifies a merge operation directly.

3. The system performs the requested action and automatically executes additional changes derived from the action.

4. The system creates a new list of suggested actions for the user, based on the new structure of the ontology, determines conflicts introduced by the last action, finds possible solutions to these conflicts and displays these to the user.

PROMPT identifies a number of ontology merging operations (merge classes, merge slots, merge bindings between a slot and a class, etc) and a number of possible conflicts introduced by the application of these operations (name conflicts, dangling references, redundancy in the class hierarchy and slot-value restrictions that violate class inheritance).

OntoMerge (Dou et al. 2002) is an on-line approach in which source ontologies are maintained after the merge operation, whereas in PROMPT the merged ontology replaces the source ontologies. The output of the merge operation in OntoMerge is not a complete merged ontology, as in PROMPT, but a bridge ontology which imports the source ontologies and which has a number of so-called Bridging Axioms (see Figure 2(b)), which are translation rules used to connect the overlapping part of the source ontologies. The two source ontologies, together with the bridging axioms, are then treated as a single theory by a theorem prover optimized for three main operations:

1. Dataset translation (cf. instance transformation in (de Bruijn & Polleres 2004)). Dataset translation is the problem of translating a set of data (instances) from one representation to the other.

2. Ontology extension generation. The problem of ontology extension generation is the problem of generating an extension (instance data) O2s, given two related ontologies O1 and O2 and an extension O1s of ontology O1. The example given by the authors is to generate a WSDL extension based on an OWL-S description of the corresponding Web Service.

3. Querying different ontologies. Query rewriting is a technique for solving the problem of querying different ontologies, whereas the authors of (Dou et al. 2002) merely stipulate the problem.

3 Mapping and Querying disparate knowledge bases

In the previous section we have seen an overview of a number of representative approaches for different aspects of ontology mediation in the areas of ontology mapping, alignment and merging. In this section we focus on an approach for ontology mapping and ontology alignment to query disparate knowledge bases in a knowledge management scenario. However, the techniques are largely applicable to any ontology mapping or alignment scenario.

In the area of knowledge management we assume there are two main tasks to be performed with ontology mappings: (a) transforming data between different representations, when transferring data from one knowledge base to another; and (b) querying of several heterogeneous knowledge bases, which have different ontologies. The ontologies in the area of knowledge management are large, but lightweight, i.e., there is a concept hierarchy with many concepts, but there are relatively few relations and axioms in the ontology. From this follows that the mappings between the ontologies will be large as well, and they will generally be lightweight; the mapping will consist mostly of simple correspondence between concepts. The mappings between ontologies are not required to be one hundred percent accurate, because of the nature of the application of knowledge management: if a search result is inaccurate it is simply discarded.

In order to achieve ontology mapping, one needs to specify the relationship between the ontologies using some language. A natural candidate to express these relationships would seem to be the ontology language which is used for the ontologies themselves. We see a number of disadvantages to this approach:

Ontology language There exist several different ontology languages for different purposes (e.g., RDFS (Brickley & Guha 2004), OWL (Dean & Schreiber 2004), WSML (de Bruijn, Fensel, Keller, Kifer, Lausen, Krummenacher, Polleres & Predoiu 2005)) and it is not immediately clear how to map between ontologies which are specified using different languages.

Independence of mapping Using an existing ontology language would typically require to import one ontology into the other and specify the relationships between the concepts and relations in the resulting ontology; this is actually a form of ontology merging. The general disadvantage of this approach is that the mapping is tightly coupled with the ontologies; one can essentially not separate the mapping from the ontologies.

Epistemological adequacy The constructs in an ontology language have not been defined for the purpose of specifying mappings between ontologies. For example, in order to specify the correspondence between two concepts Human and Person in two ontologies, one could use some equivalence or subclass construct in the ontology language, even though the intension of the concepts in both ontologies is different.

In Section 3.1 we describe a mapping language which is independent from the specific ontology language but which can be *grounded* in an ontology language for some specific tasks. The mapping language itself is based on a set of elementary *mapping patterns* which represent the elementary kinds of correspondences one can specify between two ontologies.

As we have seen in Section 2.3, there exist many different alignment algorithms for the discovery of correspondences between ontologies. In Section 3.2 we present an interactive process for ontology alignment which allows to plug-in any existing alignment algorithm. The input of this process consists of the ontologies which are to be mapped and the output is an ontology mapping.

Writing mapping statements directly in the mapping language is a tedious and error-prone process. The mapping tool OntoMap is a graphical tool for creating ontology mappings. This tool, described in Section 3.3 can be used to create a mapping between two ontologies from scratch or it can be used for the refinement of automatically discovered mappings, described in Section 3.2.

3.1 Mapping language

An important requirement for the mapping language which is presented in this section is the epistemological adequacy of the constructs in the language. In other words, the constructs in the language should correspond to the actual correspondences one needs to express in a natural way. More information about

Name: Class by Attribute Mapping
Problem: The extension of a class in one ontology corresponds to the extension of a class in another ontology, provided that all individuals in the extension have a particular attribute value.
Solution: *Solution description:* A mapping is established between a class/attribute/ attribute value combination in one ontology and a class in another ontology. *Mapping syntax:* **mapping** ::= classMapping(*direction* A B attributeValueCondition(P o))
Example: classMapping(Human Female attributeValueCondition(hasGender "female"))

Table 1: Class by Attribute Mapping pattern

the mapping language can be found in (Scharffe & de Bruijn 2005) and on the web site of the mapping language[1].

Now, what do we mean with 'natural way'? There are different patterns which one can follow when mapping ontologies. One can map a concept to a concept, a concept with a particular attribute value to another concept, a relation to a relation, etc. We have identified a number of such elementary mapping patterns which we have used as a basis for the mapping language.

Example 1. *As a simple example of possible mapping which can expressed between ontologies, assume we have two ontologies* O1 *and* O2 *which both describe humans and their gender. Ontology* O1 *has a concept* Human *with an attribute* hasGender*;* O2 *has two concepts* Woman *and* Man*.* O1 *and* O2 *use different ways to distinguish the gender of the human;* O1 *uses an attribute with two possible values* "male" *and* "female"*, whereas* O2 *has two concepts* Woman *and* Man *to distinguish the gender. Notice that these ontologies have a mismatch in the* style of modeling *(see Section 2.1). If we want to map these ontologies, we need to create two mapping rules: (1) 'all humans with the gender "female" are women' and (2) 'all humans with the gender "male" are men'.*

Example 1 illustrates one elementary kind of mapping, namely a mapping between two classes, with a condition on the value of an attribute. The elementary kinds of mappings can be captured in so-called *mapping patterns*. Table 1 describes the mapping pattern used in Example 1. The pattern is described in terms of its *name*, the *problem* addressed, the *solution* of the problem, both in natural-language description and in terms of the actual mapping language, and an *example* of the application of the pattern to ontology mapping, in this case a mapping between the class Human in ontology O1 and the class Woman in ontology O2, but only for all humans which have the gender "female".

The language contains basic constructs to express mappings between the different entities of two ontologies: from classes to classes, attributes to attributes, instances to instances, but also between any combination of entities like classes

[1]http://www.omwg.org/TR/d7/d7.2/

to instances, etc. The example in Table 1 illustrates the basic construct for mapping classes to classes, classMapping.

Mappings can be refined using a number of operators and mapping conditions. The operators in the language can be used to map between combinations of entities, such as the intersection or union (conjunction, disjunction, respectively) of classes or relations. For example, the mapping between Human and the union of Man and Woman can be expressed in the following way:

classMapping(Human or(Man Woman))

The example in Table 1 illustrates a mapping condition, namely the attribute value condition. Other mapping conditions include attribute type and attribute occurrence.

The mapping language itself is not bound to any particular ontology language. However, there needs to be a way for reasoners to actually use the mapping language for certain tasks, such as querying disparate knowledge bases and data transformation. For this, the mapping language can be grounded in a formal language. There exists, for example, a grounding of the mapping language to OWL DL and to WSML-Flight.

In a sense, the *grounding* of the mapping language to a particular language transforms the mapping language to a language which is specific for mapping ontologies in a specific language. All resulting mapping languages still have the same basic vocabulary for expressing ontology mappings, but have a different vocabulary for the more expressive expressions in the language. Unfortunately, it is not always the case that all constructs in the mapping language can be grounded to the logical language. For example, WSML-Flight does not allow disjunction or negation in the target of a mapping rule and OWL DL does not allow mapping between classes and instances. In order to allow the use of the full expressive power offered by the formal language to which the mapping language is grounded, there is an extension mechanism which allows to insert arbitrary logical expressions inside each mapping rule.

The language presented in this section is suitable for the specification and exchange of ontology mappings. In the next section we present a semi-automatic approach to the specification of ontology mappings.

3.2 A (Semi-)Automatic Process for Ontology Alignment

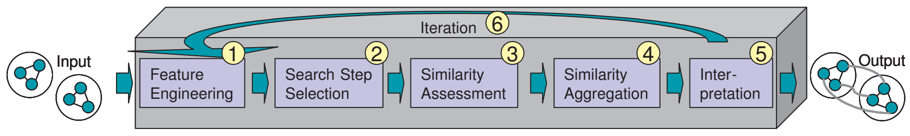


Figure 3: Alignment Process

Creating mappings between ontologies is a tedious process, especially if the ontologies are very large. We introduce a (semi-)automatic alignment process implemented in the FOAM-tool (Framework for Ontology Alignment and Mapping),[2] which relieves the user of some of the burdens in creating mappings. It subsumes all the alignment approaches we are aware of (e.g., PROMPT (Noy & Musen 2003), GLUE (Doan et al. 2003), QOM (Ehrig & Staab 2004, Ehrig & Sure 2004)). The input of the process consists of two ontologies which are to be aligned; the output is a set of correspondences between entities in the ontologies. Figure 3 illustrates its six main steps.

1. Feature engineering selects only parts of an ontology definition in order to describe a specific entity. For instance, alignment of entities may be based only on a subset of all RDFS primitives in the ontology. A feature may be as simple as the label of an entity, or it may include intensional structural descriptions such as super- or sub-concepts for concepts (a sports car being a subconcept of car), or domain and range for relations. Instance features may be instantiated attributes. Further, we use extensional descriptions. In an example we have fragments of two different ontologies, one describing the instance Daimler and one describing Mercedes. Both o1:Daimler and o2:Mercedes have a generic ontology feature called type. The values of this feature are automobile and luxury, and automobile, respectively.

2. Selection of Next Search Steps. Next, the derivation of ontology alignments takes place in a search space of candidate pairs. This step may choose to compute the similarity of a restricted subset of candidate concepts pairs of the two ontologies and to ignore others. For the running example we simply select every possible entity pair as an alignment candidate. In our example this means we will continue the comparison of o1:Daimler and o2:Mercedes. The QOM approach of Section 2.3 carries out a more efficient selection.

3. Similarity Assessment determines similarity values of candidate pairs. We need heuristic ways for comparing objects, i.e., similarity functions such as on strings, object sets, checks for inclusion or inequality, rather than exact logical identity. In our example we use a similarity function based on the instantiated results, i.e., we check whether the two concept sets, parent concepts of o1:Daimler (automobile and luxury) and parent concepts of o2:Mercedes (only automobile), are the same. In the given case this is true to a certain degree, effectively returning a similarity value of 0.5. The corresponding feature/similarity assessment (FS2) is represented in Table 2 together with a second feature/similarity assessment (FS1) based on the similarity of labels.

4. Similarity Aggregation. In general, there may be several similarity values for a candidate pair of entities from two ontologies, e.g., one for the similarity of their labels and one for the similarity of their relationship to other terms. These different similarity values for one candidate pair must be aggregated into a single aggregated similarity value. This may be achieved through

[2] `http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/`

Comparing	No.	Feature Q_F	Similarity Q_S
Entities	FS1	(label,X_1)	string similarity(X_1, X_2)
Instances	FS2	(parent,X_1)	set equality(X_1, X_2)

Table 2: Feature/Similarity Assessment

a simple averaging step, but also through complex aggregation functions using weighting schemes. For the example we only have to result of the parent concept comparison which leads to: simil(o1:Daimler,o2:Mercedes)=0.5.

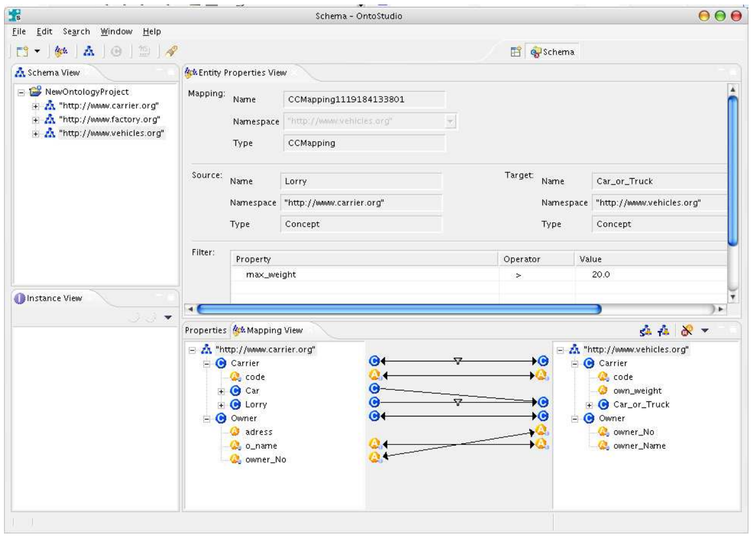
5. Interpretation uses the aggregated similarity values to align entities. Some mechanisms here are, e.g., to use thresholds for similarity (Noy & Musen 2003), to perform relaxation labeling (Doan et al. 2003), or to combine structural and similarity criteria. simil(o1:Daimler,o2:Mercedes)=0.5≥0.5 leads to align(o1:Daimler)=o2:Mercedes. This step is often also referred to as *matcher*. Semi-automatic approaches may present the entities and the alignment confidence to the user and let the user decide.

6. Iteration. Several algorithms perform an iteration (see also similarity flooding (Melnik, Garcia-Molina & Rahm 2002)) over the whole process in order to bootstrap the amount of structural knowledge. Iteration may stop when no new alignments are proposed, or if a predefined number of iterations has been reached. Note that in a subsequent iteration one or several of steps 1 through 5 may be skipped, because all features might already be available in the appropriate format or because some similarity computation might only be required in the first round. We use the intermediate results of step 5 and feed them again into the process and stop after a predefined number of iterations.

The output of the alignment process is a mapping between the two input ontologies. We cannot in general assume that all mappings between the ontologies are discovered, especially in the case of more complex mappings. Therefore, the mapping which is a result of the alignment procedure can be seen as the input of a manual refinement process. In the next Section we describe a graphical tool which can be used for manual editing of ontology mappings.

3.3 OntoMap: an Ontology Mapping tool

OntoMap® (Schnurr & Angele 2005) is a plugin for the ontology-management platform OntoStudio® that supports the creation and management of ontology mappings. Mappings can be specified based on graphical representation, using a schema-view of the respective ontologies. OntoMap encapsulates the formal statements for the declaration of mappings, users just need to understand the semantics of the graphical representation (e.g., an arrow connecting two concepts). Users of OntoMap are supported by drag-and-drop functionality and simple consistency checks on property-mappings (automatic suggestion of necessary class-mappings). For concept-to-concept mappings constraints can be specified on the available attributes.



Mapping: Name CCMapping1119184133801
Namespace "http://www.vehicles.org"
Type CCMapping

Source: Name Lorry
Namespace "http://www.carrier.org"
Type Concept

Target: Name Car_or_Truck
Namespace "http://www.vehicles.org"
Type Concept

Filter:

Property	Operator	Value
max_weight	>	20.0

Figure 4: Screenshot of OntoStudio® with the OntoMap® plugin

OntoMap supports a number of most elementary mapping patterns: concept to concept mappings, attribute to attribute mappings, relation to relation mapings, and attribute to concept mappings.

Additionally, OntoMap allows to specify additional conditions on concept-to-concept mappings using a form for mapping properties. A concept 'lorry' for example might map onto a concept 'Car or truck' only if the weight of the latter exceeds a certain limit (e.g. according to the legal definition within some countries).

Attribute to concept mappings enable users to specify one or more 'identifiers' for instances of a concept - similar to the primary keys in relational databases. This way the properties of different source concepts can be 'unified' in one target concept, e.g., in order to join the information of different database entries within a single instance on the ontology level. Different source instances having the same 'identifier values' are then joined within a single target instance.

The focus of OntoMap is on the intuitive creation and management of mappings. If complex mappings are needed, which are not within the scope of a graphical tool (possibly using complex logical expressions or built-ins), they have to be encoded manually. OntoStudio has its own grounding of mappings, based on F-Logic rules (Kifer & Lausen 1997). In addition to the internal storage format OntoMap supports the import and export of mappings in the mapping-language which we described in this section. An extension of OntoMap based on a library for ontology alignment[3] which was described in Section 3.2 provides

[3]http://www.aifb.uni-karlsruhe.de/WBS/meh/foam/

the functionality for the semi-automatic creation of mappings.

Some additional features of the OntoStudio environment support users in typical mediation tasks. Those include the import of schemas for relational databases (syntactic integration) and the possibility to create and execute queries instantly. The latter gives users the possibility to test the consequence of mappings they have created (or imported). Such usage of mappings to query disparate knowledge bases is described in the next section.

4 Summary

Overlap and mismatches between ontologies are likely to occur when the vision of a Semantic Web with a multitude of ontologies becomes a reality.

There exist different approaches to ontology mediation. These approaches can be broadly categorized as: (a) *Ontology Mapping* (Maedche et al. 2002, Scharffe & de Bruijn 2005), (b) *Ontology Alignment* (Ehrig & Staab 2004, Ehrig & Sure 2004, Rahm & Bernstein 2001, Doan et al. 2004) and (c) *Ontology Merging* ((Noy & Musen 2000*b*, Dou et al. 2002)). We have presented a survey of the most prominent approaches in these areas.

Additionally, we described a practical approach to representing mappings using a mapping language, discovering mappings using an alignment process which can be used in combination with any ontology alignment algorithm, and a graphical tool to edit such ontology mappings. These ontology mappings can now be used, for example, for transforming data between different representation, as well as querying different heterogeneous knowledge bases.

Although there is some experience with ontology mediation and most approaches to ontology mediation, especially in the field of ontology alignment, have been validated using some small test set of ontologies. The overall problem which the area of ontology mediation faces is that the number of ontologies which is out there on the Semantic Web is currently very limited and it is hard to really validate the approaches using real ontologies.

References

Bouquet, P., Giunchiglia, F., van Harmelen, F., Serafini, L. & Stuckenschmidt, H. (2004), 'Contextualizing ontologies', *Journal of Web Semantics* **1**(4), 325.

Brickley, D. & Guha, R. V. (2004), RDF vocabulary description language 1.0: RDF schema, W3c recommendation 10 february 2004, W3C.
URL: *http://www.w3.org/TR/rdf-schema/*

de Bruijn, J., Fensel, D., Keller, U., Kifer, M., Lausen, H., Krummenacher, R., Polleres, A. & Predoiu, L. (2005), The web service modeling language WSML, W3C member submission 3 june 2005.
URL: *http://www.w3.org/Submission/WSML/*

de Bruijn, J. & Polleres, A. (2004), Towards and ontology mapping specification language for the semantic web, Technical Report DERI-2004-06-30, DERI.
URL: *http://www.deri.at/publications/techpapers/documents/DERI-TR-2004-06-30.pdf*

Dean, M. & Schreiber, G. (2004), OWL web ontology language reference, W3C recommendation 10 february 2004.
URL: *http://www.w3.org/TR/owl-ref/*

Doan, A., Domingos, P. & Halevy, A. (2003), 'Learning to match the schemas of data sources: A multistrategy approach', *VLDB Journal* **50**, 279–301.

Doan, A. & Halevy, A. (2005), 'Semantic integration research in the database community: A brief survey', *AI Magazine, Special Issue on Semantic Integration* .

Doan, A., Madhaven, J., Domingos, P. & Halevy, A. (2004), Ontology matching: A machine learning approach, *in* S. Staab & R. Studer, eds, 'Handbook on Ontologies in Information Systems', Springer-Verlag, pp. 397–416.

Dou, D., McDermott, D. & Qi, P. (2002), Ontology translation by ontology merging and automated reasoning, *in* 'Proc. EKAW2002 Workshop on Ontologies for Multi-Agent Systems', pp. 3–18.

Ehrig, M. & Staab, S. (2004), QOM - quick ontology mapping, *in* F. van Harmelen, S. McIlraith & D. Plexousakis, eds, 'Proceedings of the Third International Semantic Web Conference (ISWC2004)', LNCS, Springer, Hiroshima, Japan, pp. 683–696.

Ehrig, M. & Sure, Y. (2004), Ontology mapping - an integrated approach, *in* 'Proceedings of the First European Semantic Web Symposium, ESWS 2004', Vol. 3053 of *Lecture Notes in Computer Science*, Springer Verlag, Heraklion, Greece, pp. 76–91.

Giunchiglia, F. & Shvaiko, P. (2004), 'Semantic matching', *The Knowledge Engineering Review* **18**(3), 265–280.

Giunchiglia, F., Shvaiko, P. & Yatskevich, M. (2004), S-match: an algorithm and an implementation of semantic matching, *in* 'Proceedings of ESWS'04', number 3053 *in* 'LNCS', Springer-Verlag, Heraklion, Greece, pp. 61–75.

Kalfoglou, Y. & Schorlemmer, M. (2003), 'Ontology mapping: the state of the art', *The Knowledge Engineering Review* **18**(1), 1–31.

Kifer, M. & Lausen, G. (1997), 'F-logic: A higher-order language for reasoning about objects', *SIGMOD Record* **18**(6), 134–146.

Klein, M. (2001), Combining and relating ontologies: an analysis of problems and solutions, *in* A. Gomez-Perez, M. Gruninger, H. Stuckenschmidt & M. Uschold, eds, 'Workshop on Ontologies and Information Sharing, IJCAI'01', Seattle, USA.

Maedche, A., Motik, B., no Silva, N. & Volz, R. (2002), Mafra a mapping framework for distributed ontologies, *in* 'Proceedings of the 13th European Conference on Knowledge Engineering and Knowledge Management EKAW-2002', Madrid, Spain.

Melnik, S., Garcia-Molina, H. & Rahm, E. (2002), Similarity flooding: A versatile graph matching algorithm and its application to schema matching, *in* 'Proceedings of the 18th International Conference on Data Engineering (ICDE'02)', IEEE Computer Society, p. 117.

Noy, N. F. (2004), 'Semantic integration: A survey of ontology-based approaches', *Sigmod Record, Special Issue on Semantic Integration* **33**(4), 65–70.

Noy, N. F. & Musen, M. A. (2000*a*), Anchor-PROMPT: Using non-local context for semantic matching, *in* 'Proceedings of the Workshop on Ontologies and Information Sharing at the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-2001)', Seattle, WA, USA.

Noy, N. F. & Musen, M. A. (2000*b*), PROMPT: Algorithm and tool for automated ontology merging and alignment, *in* 'Proc. 17th Natl. Conf. On Artificial Intelligence (AAAI2000)', Austin, Texas, USA.

Noy, N. F. & Musen, M. A. (2003), 'The PROMPT suite: interactive tools for ontology merging and mapping', *International Journal of Human-Computer Studies* **59**(6), 983–1024.

Omelayenko, B. (2002), RDFT: A mapping meta-ontology for business integration, *in* 'Proceedings of the Workshop on Knowledge Transformation for the Semantic Web (KTSW 2002) at the 15-th European Conference on Artificial Intelligence', Lyon, France, pp. 76–83.

Rahm, E. & Bernstein, P. A. (2001), 'A survey of approaches to automatic schema matching', *VLDB Journal: Very Large Data Bases* **10**(4), 334–350.

Scharffe, F. & de Bruijn, J. (2005), A language to specify mappings between ontologies, *in* 'Proceedings of the Internet Based Systems IEEE Conference (SITIS05)', Yandoué, Cameroon.

Schnurr, H.-P. & Angele, J. (2005), Do not use this gear with a switching lever! automotive industry experience with semantic guides, *in* '4th International Semantic Web Conference (ISWC2005)', pp. 1029–1040.

Shvaiko, P. & Euzenat, J. (2005), 'A survey of schema-based matching approaches', *Journal on Data Semantics* .

Uschold, M. (2000), Creating, integrating, and maintaining local and global ontologies, *in* 'Proceedings of the First Workshop on Ontology Learning (OL-2000) in conjunction with the 14th European Conference on Artificial Intelligence (ECAI-2000)', Berlin, Germany.

Visser, P. R. S. & Cui, Z. (1998), On accepting heterogeneous ontologies in distributed architectures, *in* ‘Proceedings of the ECAI98 workshop on applications of ontologies and problem-solving methods’, Brighton, UK.