Combining and relating ontologies: an analysis of problems and solutions

Michel Klein
Vrije Universiteit Amsterdam
Michel.Klein@cs.vu.nl

Abstract

With the grown availability of large and specialized online ontologies, the questions about the combined use of independently developed ontologies have become even more important. Although there is already a lot of research done in this area, there are still many open questions. In this paper we try to classify the problems that may arise into a common framework. We then use that framework to examine several projects that aim at some ontology combination task, thus sketching the state of the art. We conclude with an overview of the different approaches and some recommandations for future research.

1 Introduction

In the last few years, there has been put a lot of effort in the development of techniques that aim at the "Semantic Web". This next step in the evolution of the World Wide Web, will enable computers to partly "understand" the information on the internet. A lot of those newly developed techniques requires and enables the specification of ontologies (Gruber, 1993) on the web. Consequently, there will emerge a lot of freely accessible domain specific ontologies. The reuse of these ontologies may be very attractive.

However, there are several problems when one tries to use independently developed ontologies together, or when existing ontologies are adapted for new purposes. Although there is already a lot of research done in this area, there are still many open questions. In this paper, we investigate the problems that may arise. We will distinguish several types of mismatches that can occur between different ontologies, we will look at practical problems and we will look at some of the consequences of changes to ontologies. Altogether, this will give us a framework that can be used to compare approaches that aim at solving the problems. We will use this to examine several techniques and tools that has the purpose to solve these problems or to support users in performing ontology combining tasks.

The paper is organized as follows. We will first clarify the terminology that is used in the field of ontology combining (section 2). In section 3, we will investigate all problems that arise when ontologies are combined or related, which results in a framework of relevant issues. Next, in section 4, we will use the framework to examine existing approaches for ontology combining. In section 5, we summarize the techniques and give an overview of the different approaches that are used. Finally, in section 6 we will conclude the paper and we will make some remarks based on our observations.

2 Terminology

Before we can analyse the problems that play a role, we need to clarify the terminology and define the terms we will use. We will have to make some decisions about our understanding of the terminology, because there is not always an agreement on the exact meaning of the terms. We have tried to be consistent as far as possible with definitions and descriptions found elsewhere.

Reuse of existing ontologies is often not possible without considerable effort (Uschold *et al.*, 1998). When one wants to reuse different ontologies together, those ontologies have to be *combined* in some way. This can be done by *integrating* (Pinto *et al.*, 1999) the ontologies, which means that they are merged into one new ontology, or the ontologies can be kept separate. In both cases, the ontologies have to be *aligned*, which means that they have to be brought into mutual agreement.

Ontology integration consist of (the iteration of) the following steps (McGuinness *et al.*, 2000):

1. find the places in the ontologies where they overlap;
2. relate concepts that are semantically close via equivalence and subsumption relations (aligning);
3. check the consistency, coherency and non-redundancy of the result.

The alignment of concepts between ontologies is especially difficult, because this requires understanding of the meaning of concepts. Aligning two ontologies implies changes to at least one of them. Changes to an ontology will result in a new *version* of an ontology.

If the ontologies are not represented in the same language, a *translation* is often required.

Throughout this paper, we will use the following terms consistently according to their specified meaning:

combining: Using two or more different ontologies for a task in which their mutual relation is relevant.

merging, integrating: Creating a new ontology from two or more existing ontologies with overlapping parts, which can be either virtual or physical.

aligning: Bring two or more ontologies into mutual agreement, making them consistent and coherent.

mapping: Relating similar (according to some metric) concepts or relations from different sources to each other by an equivalence relation. A mapping result in a virtual integration.

articulation: The points of linkage between two aligned ontologies, ie. the specification of the alignment.

translating: Changing the representation formalism of an ontology while preserving the semantics.

transforming: Changing the semantics of an ontology slightly (possibly also changing the representation) to make it suitable for purposes other than the original one.

version: The result of a change that may exist next to the original.

versioning: A method to keep the relation between newly created ontologies, the existing ones, and the data that conforms to them consistent.

3 Problems with ontology combination

The combined use of multiple ontologies is hindered by several problems. In this section, we will investigate and describe them.

The problems that underlies the difficulties in merging and aligning are the mismatches that may exist between separate ontologies. In the next subsection, we will discuss these mismatches. We will then look at the different type of problems involved with versioning and revisioning. Finally, we will discuss some practical problems that come up when one tries to combine ontologies.

Thus doing, we will build a framework with the different types of problems that can occur when relating ontologies. This framework can be used when we compare the existing approaches and tools.

3.1 Mismatches between ontologies

Mismatches between ontologies are the key type of problems that hinder the combined use of independently developed ontologies. We will now explore *how* ontologies may differ. In the literature, there are a lot of possible mismatches mentioned, which are not always easy comparable. To make them more comparable, we try to classify the different types of mismatches and relate them to each other.

As a first step, we will distinguish between two levels at which mismatches may appear. The first level is the **language** or meta-model level. This is the level of the language primitives that are used to specify an ontology. Mismatches at this level are mismatches between the *mechanims* to define classes, relations and so on. The second level is the **ontology** or model level, at which the actual ontology of a domain lives. A mismatch at this level is a difference in the way the domain is modelled. The distinction between these two levels of differences is made very often. Kitakami *et al.* (1996) and Visser *et al.* (1997) call these kinds of differences respectively *non-semantic* and *semantic* differences. Others make this distinction implicitly, by only concentrating on one of the two levels. For example, Wiederhold (1994) analyses domain differences (i.e., ontology level), while Grosso *et al.* (1998) and Bowers and Delcambre (2000) look at langauge level differences. In the following, we will avoid the use of the words "semantic differences" for ontology level differences, because we reserve those words for a more specific type of difference (which will be described below).

Below, we will give an overview and characterization of different types of mismatches that can appear at each of those two levels.

Language level mismatches

Mismatches at the language level occur when ontologies written in different ontology languages are combined. Chalupsky (2000) defines mismatches in *syntax* and *expressivity*. In total, we distinguish four types of mismatches that can occur, although they often coincide.

- **Syntax** Obviously, different ontology languages often use different syntaxes. For example, to define the class of chairs in RDF Schema (Brickley and Guha, 2000), one uses `<rdfs:Class ID="Chair">`. In LOOM, the expression `(defconcept Chair)` is used to define the same class. This difference is probably the most simple kind of mismatch. However, this mismatch often doesn't come alone, but is coupled with other differences at the language level. A typical example of a "syntax only" mismatch is an ontology language that has several syntactical representations. In this simple case, a rewrite mechanims is sufficient to solve those problems.

- **Logical representation** An slightly more complicated mismatches at this level is the difference in representation of logical notions. For example, in some languages it is possible to state explicitly that two classes are disjoint (e.g., `disjoint A B`), whereas it is necessary to use negation in subclass statements (e.g., `A subclass-of (NOT B), B subclass-of (NOT A)` in other languages. The point here is not whether something can be expressed — the statements are logically equivalent — but which langauge constructs should be used to express something. Also, notice that this mismatch is not about the representation of *concepts*, but about the representation of *logical notions*.

  This type of mismatch is still relatively easy solvable, e.g. by giving translation rules from one logical representation to another.

- **Semantics of primitives** A more subtle possible difference at the metamodel level is the semantics of language constructs. Despite the fact that sometimes the same name is used for a language construct in two languages,

the semantics may differ; e.g., there are several interpretations of `A equalTo B`.

Note that even when two ontologies seem to use the same syntax, the semantics can differ. For example, the OIL RDF Schema syntax (Broekstra *et al.*, 2001) interprets multiple `<rdfs:domain>` statements as the intersection of the arguments, whereas RDF Schema itself uses union semantics[1].

- **Language expressivity** The mismatch at the metamodel level with the most impact is the difference in expressivity between two languages. This difference implies that some langauges are able to express things that are not expressible in other languages. For example, some languages have constructs to express negation, others have not. Other typical differences in expressivity are the support of lists and sets, default values, etc.

  This type of mismatch has probably the most impact, and is mentioned by several others. The "fundamental differences" between knowledge models that are described in (Grosso *et al.*, 1998) are also very close to our interpretation.

Our list of differences at the language level can be seen as more or less compatible with the broad term "language heterogeneity" of Visser *et al.* (1997).

Ontology level mismatches

Mismatches at the ontology — or model — level happen when two or more ontologies that describe (partly) overlapping domains are combined. These mismatches may occur when the ontologies are written in the same language, as well as when they use different languages. Based on the literature and on our own observations, we can distinguish several types of mismatches at the model level.

Visser *et al.* (1997) make a very useful distinction between mismatches in the *conceptualization* and *explication* of ontologies. A conceptualization mismatch is a difference in the way a domain is interpreted (conceptualized), which results in different ontological concepts or different relations between those concepts. An explication mismatch, on the other hand, is a difference in the way the conceptualization is *specified*. This can manifest itself in mismatches in definitions, mismatches in terms and combinations of both. Visser *et al.* list all the combinations. Four of them are related to hymonym terms and synonym terms.

Wiederhold (1994) also mentions the problems with synonym terms (called *naming differences*) and homonym terms (*subjective meaning*). Besides that, he describes possible differences in the *scope of concepts*, which is an example of a conceptual mismatch. Finally, he mentions *value encoding* differences, for example, differences in the currency of prices.

Chalupsky (2000) list four types of mismatches in ontologies. One of them, *inference system bias* is in our opinion not a real mismatch, but a reason for modeling style differences. The other three mismatches, *modeling conventions*, *coverage and granularity* and *paradigms* can be categorized as instances of the two main mismatch types of Visser *et al.*. We will describe them in a slightly altered way below.

We will now relate the different types of mismatches that are distinguished by the authors cited above. Thus, we will continue the build of our framework.

The first two mismatches at the model level that we distinguish are instances of the **conceptualization mismatches** of Visser *et al.*. This are semantic differences, i.e., not only the specification, but also the conceptualization of the domain (see the definition of Gruber, 1993) is different in the ontologies that are involved.

- **Scope** Two classes seem to represent the same concept, but do not have exactly the same instances, although these intersect. The standard example is the class "employee": several administrations use slightly different concepts of employee, as mentioned by Wiederhold (1994). In (Visser *et al.*, 1997), this is called a *class mismatch* and is worked out further into detailed descriptions at class- or relation-level.
- **Model coverage and granularity** This is a mismatch in the part of the domain that is covered by the ontology, or the level of detail to which that domain is modelled. Chalupsky (2000) gives the example of an ontology about cars: one ontology might model cars but not trucks. Another one might represent trucks but only classify them into a few categories, while a third one might make very fine-grained distrinctions between types of trucks based on their general physical structure, weight, purpose, etc.

Conceptualization differences as described above can not be solved automatically, but require knowledge and decisions of a domain expert. In the second case, the mismatch is often not a problem, but a motive to use different ontologies together. In that case, the remaining problem is to align the overlapping parts of the ontology.

The other ontology-level mismatches can be categorized as **explication mismatches**, in the terminology of Visser *et al.*. The first two of them result from explicit choices of the modeler about the **style of modeling**:

- **Paradigm** Different paradigms can be used to represent concepts such as time, action, plans, causality, propositional attitudes, etc. For example, one model might use temporal representations based on interval logic while another might use a representation based on point (Chalupsky, 2000). The use of different "top-level" ontology is also an example of this kind of mismatch.
- **Concept description** This type of differences are called *modeling conventions* in (Chalupsky, 2000). Several choices can be made for the modeling of concepts in the ontology. For example, a distinctions between two classes can be modeled using a qualifying attribute or by introducing a separate class. These choices are sometimes influenced by the intended inference system. Another choice in concept descriptions is the way in which is-a hierarchy is build; distinctions between features can be made higher or lower in the hierarchy. For example, consider the place where the distinction between

[1] Although this will probably change in the next revision of RDF Schema, according to a discussion on the RDF-interest mailinglist.

scientific and non-scientific publications is made: a dissertation can be modeled as `dissertation < book < scientific publication < publication`, or as `dissertation < scientific book < book < publication`, or even as subclass of both `book` and `scientific publication`.

Further, the next two types of differences can be classified as **terminological mismatches**.

- **Synonym terms** Concepts are represented by different names. A trivial example is the use of the term "car" in one ontology and the term "automobile" in another ontology. This type of problem is called *term mismatch (T or TD)* in (Visser *et al.*, 1997).

  A special type of this problem is the case that the natural language in which ontologies are described differ.

  Although the technical solution for this type of problems seems relatively simple (the use of thesauri), the integration of ontologies with synonyms or different languages requires usually a lot of human effort and comes with several semantic problems. Especially, one must be careful not to overlook a scope difference (see above).

- **Homonym terms** The meaning of a term is different in an other context. For example, the term "conductor" has a different meaning in a music domain than in an electric engineering domain. Visser *et al.* (1997) calls this a *concept mismatch (C or CD)*.

  This inconsistency is much harder to handle; (human) knowledge is required to solve this ambiguity.

Finally, there is a one trivial type of difference left.

- **Encoding** Values in the ontologies may be encoded in different formats. For example, a date may be represented as "dd/mm/yyyy" or as "mm-dd-yy", distance may be described in miles or kilometers, etc. There are many mismatches of this type, but these are all very easy to solve. In most cases, a transformation step or wrapper is sufficient to eliminate all those differences.

3.2 Ontology versioning

The problems listed above are mismatches between ontologies. Most projects and approaches focus on solving these mismatches. However, mismatches are not the only problems that have to be solved when one want to use several ontologies together for one task.

As changes to ontologies are inevitable in an open domain, it becomes very important to keep track of the changes and of its impact on the dependencies of that ontology. It is often not practically possible to synchronize the changes of an ontology with the revisions to the applications and datasources that use them. Therefore, a versioning method is needed to handle revisions of ontologies and the impact on existing sources. In some sense, the versioning problem can also be regarded as a derivation of ontology combination; it results from changes (possibly required by combination tasks) to individual ontologies.

Ontology versioning covers several aspects. Although the problem is introduced by subsequent changes to one specific ontology, the most important problems are caused by the dependencies on that ontology. Therefore, it is useful to distinguish the aspects of ontology versioning. A versioning scheme should take care of the following aspects:

1. the relation between succeeding revisions of one ontology;
2. the relation between the ontology and its dependencies:
   - instance data that conforms to the ontology;
   - other ontologies that are built from, or import the ontology;
   - applications that use the ontology.

The central question that a versioning scheme answers is: how to reuse existing ontologies in new situations, without invalidating the existing ones. A versioning scheme provides ways to disambiguate the interpretation of concepts for users of the ontology revisions, and it makes the compatibility of the revisions explicit. Consequently, we can impose the following requirements on a versioning scheme, in increasing level of difficulty:

- for every use of a concept or a relation, a versioning framework should provide an unambigious reference to the intended definition (**identification**);
- a versioning framework should make the relation of one version of a concept or relation to other versions of that construct explicit (**change tracking**);
- a versioning framework should — as far as possible — automatically perform conversions from one version to another, to enable transparant access (**transparant translating**).

We will examine the current approaches with respect to those requirements.

3.3 Practical problems

Besides the technical problems that we discussed in the previous sections, there are also practical problems that hinder the easy use of combined ontologies. Aligning and merging ontologies, the central aspect of ontology combining, is a complicated process and requires serious effort of the ontology designers. Until now, this task is mostly done by hand (Noy and Musen, 2000), which makes it difficult to overlook in two aspects:

- it is difficult to find the terms that need to be aligned;
- the consequences of a specific mapping (unforeseen implications) are difficult to see.

Because it is unrealistic to hope that merging or alignment at the semantic level could be performed completely automatically, we should take these practical aspects into consideration.

Another practical problem is that repeatability of merges. Most often, the sources that are used for the merging, continue to evolve. The alignments that are created for the merging, should be as much reusable as possible for the merging of the revised ontologies. This issue is very important in the context of ontology maintenance. The repeatability could for example be achieved by an executable

specification of the alignment.

Summarizing the previous sections, we can construct the framework of issues that is depicted in Figure 1.

4 Current approaches and techniques

In this section, we will use the framework to examine several tools and techniques that are aimed at ontology combining. We start at the top of the framework, looking at techniques for solving language mismatches. Then, we will look at techniques for solving ontology level mismatches and user support. Of course, it is not possible to make a strict distinction between the type of problems that a technique solves, because some tools or techniques provide support for several types of problems. The place where we mention them does therefore not imply a classification, but serves as a guideline only.

4.1 Solving language mismatches

There are several approaches for solving the problem of integrating ontologies that are written in different knowledge representation languages. Some of them are just techniques, others also provide some kind of automated support.

Superimposed metamodel

Bowers and Delcambre (2000) describes an approach to transforming information from one representation to another. Their focus is on model-based information where the information representation scheme provides structural modeling constructs (analogous to a data model in a database). For example, the XML model includes elements, attributes, and permits elements to be nested. Similarly, RDF models information through resources and properties. The goal of their work is to enable the user to apply tools of interest to the information at hand. Their approach is to represent information for a wide variety of model-based applications in a uniform way, and to provide a mapping formalism that can easily transform information from one representation to another.

To achieve this, the ontology languages (i.e., the specific constructs in the language that are used to describe an ontology) are each represented in a meta-model, a so called ”superimposed” model. These superimposed models are represented as RDF triples. Mappings are then specified using production rules. The rules are defined over triples of the RDF representation for superimposed information. Since a triple is a simple predicate, mapping rules are specified using a logic-based language such as Prolog, which allows to both specify and implement the mappings. There is no requirement that the mappings between superimposed layers should be complete, since only part of a model or schema may be needed while using a specific tool.

If superimposed information from a source language is being mapped to the target language, is is possible to convert *data* from the source layer into data that conforms to the target layer. Although the focus is on conversion, it is also possible to perform integration between superimposed layers. Integration goes a step further by combining the source and target data. The mapping rules can be used to provide integration at both the schema and instance levels.

The “superimposed model approach” provides mechanisms to solve language level mismatches of syntax, representation and semantics. The mappings between the language constructs have to be specified manually. The semantic resolution of mismatches at the ontology level is not covered by this approach.

Layered approach to interoperability

Melnik and Decker (2000) introduce some initial ideas targeted at facilitating data interoperation using a layered approach. Their approach resembles the layering principles used in internetworking. To harness the complexity of data interoperation, Melnik and Decker suggest viewing Web-enabled information models as a series of layers: the syntax layer, the object layer, and the semantic layer. The semantic layer, or knowledge representation layer, deals with conceptual modeling and knowledge engineering tasks. The basic function of the object layer, or frame layer, is to provide applications with an object-oriented view of their domain. The syntax layer is responsible for ”dumbing down” object-oriented information into document instances and byte streams. Each layers has a number of sublayers, which corresponds to a specific data modeling feature (e.g., aggregation or reification) that can be logically implemented in different ways.

It seems that a clean separation between different layers may ease the achievement of interoperability. The first two layers that the authors distinguish map nicely onto the first two types of mismatches that we described in Section 3.1. However, all other types of mismatches that we distinguish are comprised in the “semantic layer”. Therefore, the layering as described in its inital version only solves some of the language level mismatches.

As the authors also have noticed, considering data models in a layered fashion is a contemporary approach. For example, OIL is presented as an extension to RDF Schema (Broekstra *et al.*, 2001), and Euzenat (2001) investigates the characteristics of interoperability of knowledge representations at various levels.

OKBC

The Open Knowledge Base Connectivity (Chaudhri *et al.*, 1998) is a generic interface to knowledge representation systems (KRS). An “application programming interface” (API), specifies the operations that can be used to access a system by an application program. When specifying this API for knowledge representation systems, some assumptions are made about the representation used by that KRS. These assumptions are made explicit in the OKBC knowledge model.

A specific knowledge representation language — or ontology language — can be bound to OKBC by defining a mapping from OKBC knowledge model to the specific language. The users of the ontology are then isolated from the peculiarities of specific language and can use the OKBC model. The interoperability achieved by using OKBC is at the level of the OKBC knowledge model.

OKBC thus can solve some mismatches at the language level. However, semantic differences that are beyond representation can not be solved by providing mappings to the

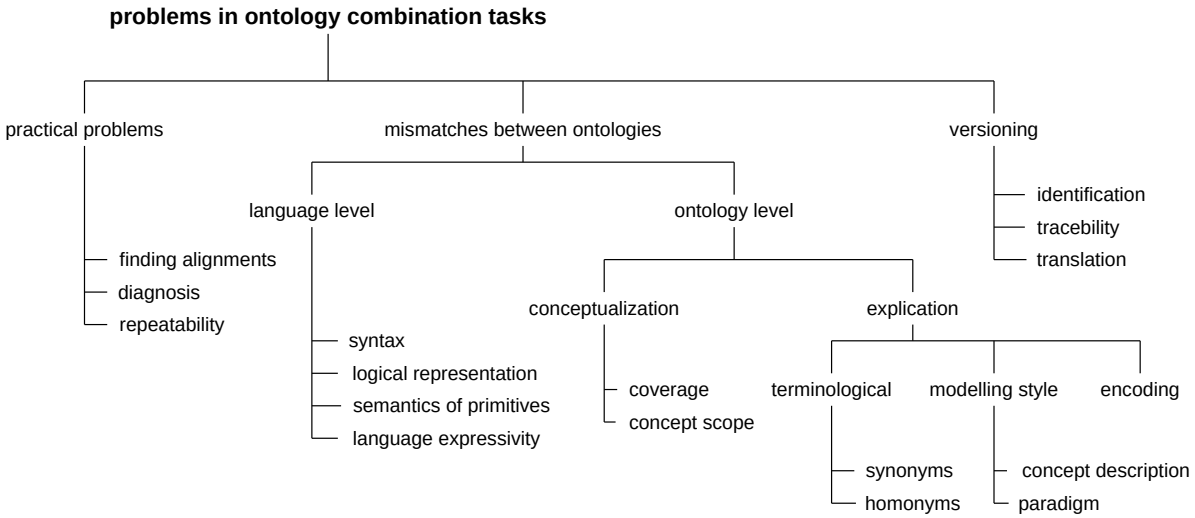

Figure 1: The resulting framework of issues that are involved in ontology combining

OKBC knowledge model. More general, when using a mapping to a common knowledge model, the notions that requires a higher level of expressivity than that model provides, will be lost.

OntoMorph

OntoMorph (Chalupsky, 2000) is a transformation system for symbolic knowledge. It facilitates ontology merging and the rapid generation of knowledge base translators. It combines two mechanisms to describe knowledge base transformations: (1) *syntactic rewriting* via pattern-directed rewrite rules that allow the concise specification of sentence-level transformations based on pattern matching, and (2) *semantic rewriting* which modulates sytactic rewriting via (partial) semantic models and logical inference via an integrated KR system. The integration of these mechanisms allows transformations to be based on any mixture of syntactic and semantic criteria. The OntoMorph architecture facilitates incremental development and scripted replay of transformations, which is particulary important during merging operations.

OntoMorph focuses at transformations to individual ontologies that are needed to align two or more ontologies. This is small but important step in the process of merging ontologies. In fact, step number 2 of the ontology merging process (see Section 2) is split into three:

2a. design transformations to bring the sources into mutual agreement;

2b. editing or *morphing* the sources to carry out the transformations;

2c. taking the union of the morphed sources;

OntoMorph facilitates step 2b, by transforming the ontologies into a common format with common names, common syntax, uniform modeling assumptions, etc.

Step 2a, the design of the transformations, involves understanding of the meaning of the representation, and is therefore a less automatable task. Additionally, this step often involves human negotiation to reconcile competing views on how a particular modeling problem should be solved.

OntoMorph is able to solve several problems at the language level of ontology mismatches framework. Of course, a difference in expressivity between two languages is not solvable, but some implies loss of knowledge. Solutions for ontology level problems can also be formulated in OntoMorph. Because OntoMorph requires a clear and executable specification of the transformation, the process can be repeated with modified versions of the original ontologies.

4.2 Ontology level integration and user support

In the previous section, we saw that OntoMorph provide mechanisms and support for some model level integration, too. We will now look at a transformation system, that allows the specification and execution of transformation of individual ontologies. We will then discuss two tools that assist the user in the complicated task of performing a merge.

Algebra for Scalable Knowledge Composition

The Scalable Knowledge Composition (SKC)[2] project developed an *algebra* for ontology composition. This algebra is used in the ONION system (ONtology compositION), described in (Mitra *et al.*, 2000). The current work in the SKC project is not solely in the area of ontology combination, but in the broader field of integrating heterogenous datasources.

The algebra operates on ontologes that are represented by nodes and arcs (terms and relationships) in a directed labelled graph. Each algebraic operator takes as input a graph of semistructured data and transforms it to another graph. This guarantees that the algebra is composable. The algebra operations are themselves knowledge driven, using articulation rules. The rules can be both logical rules (e.g., semantic

[2]See `http://www-db.stanford.edu/SKC/`.

implication between terms across ontologies) and functional rules (e.g., dealing with conversion between terms across ontologies). The composition rules are partly suggested by expert and lexical knowledge.

Intersection is the most crucial operation since it identifies the terms where linkage occurs among the domains, which is called "the articulation". An *articulation ontology* contains the terms from the source ontologies that are related and their relation, and can be seen as a specification of an alignment. This separate specification facilitates repeated executions of the composition.

When we relate this to our framework, we see that the algebra allows the specification of solutions to solve several conceptual and terminological mismatches. Via the functional rules, term synonyms and encoding problems can be eliminated. The logical articulation rules provides a mean to solve mismatches in scope, coverage and homonym terms. By using expert and lexical knowledge to suggest articulations, the system also meets the practical problems of finding alignments.

One of the main advantages of using an algebra for combination, is the reusability. The unified ontology is not an physical entity, but an term to denote the result of applying the articulation rules. This approach ensures minimal coupling between the sources, so that the sources can be developed and maintained independently.

Chimaera

Chimaera (McGuinness *et al.*, 2000) is an ontology merging and diagnosis tool developed by the Stanford University Knowledge Systems Laboratory (KSL). Its initial design goal was to provide substantial assistance with the task of merging KBs produced by multiple authors in multiple settings. Later, it took on another goal of supporting testing and diagnosing ontologies as well. Finally, inherent in the goals of supporting merging and diagnosis are requirements for ontology browsing and editing. It is mainly targeted at lightweight ontologies.

The two major tasks in merging ontologies that Chimaera support are (1) coalesce two semantically identical terms from different ontologies so that they are referred to by the same name in the resulting ontology, and (2) identify terms that should be related by subsumption, disjointness, or instance relationships and provide support for introducing those relationships. There are many auxiliary tasks inherent in these tasks, such as identifying the locations for editing, performing the edits, identifying when two terms could be identical if they had small modifications such as a further specialization on a value-type constraint, etc.

Chimaera generates name resolution lists that help the user in the merging task by suggesting terms each of which is from a different ontology that are candidates to be merged or to have taxonomic relationships not yet included in the merged ontology. It also generates a taxonomy resolution list where it suggests taxonomy areas that are candidates for reorganization. It uses a number of heuristic strategies for finding such edit points.

Finally, Chimaere also has diagnostic support for verifying, validating, and critiquing ontologies. Those functions only include domain independent tests that showed value in previous experiments.

We see that Chimaera can be used to solve mismatches at the terminological level. It is also able to find some similar concepts that have a different description at the model level. Further, Chimaera seems to do a great job in helping the user to find possible edit point. The diagnostic functions are difficult to evaluate, because their description is very brief.

PROMPT

PROMPT (formerly known as SMART) is an interactive ontology-merging tool (Noy and Musen, 2000). It guides the user through the merging process making suggestions, determining conflicts, and proposing conflict-resolution strategies. PROMPT starts with the linguistic-similarity matches of frame names for the initial comparison, but concentrates on finding clues based on the structure of the ontology and users actions. After the user selects an operation to perform, PROMPT determines the conflicts in the merged ontology that the operation have caused and proposes possible solutions to the conflict. It then considers the structure of the ontology around the arguments to the latest operations — relations among the arguments and other concepts in the ontology — and proposes other operations that the user should perform.

In the PROMPT project, a set of knowledge-base operations for ontology merging or alignment is identified. For each operation in this set is defined (1) the changes that PROMPT performs automatically, (2) the new suggestions that PROMPT presents to the user, and (3) the conflicts that the operation may introduce and that the user needs to resolve. When the user invokes an operation, PROMPT creates members of these three sets based on the arguments to the specific invocation of the operation.

The conflicts that may appear in the merged ontology as the result of these operations are: name conflicts, dangling references, redundancy in the class-hierarchy and inconsistencies. PROMPT not only points to the places where changes should be made, but also presents a list of actions to the user.

Summarizing, PROMPT gives iterative suggestions for concept merges and changes, based on linguistic and structural knowledge, and it points the user to possible effects of these changes.

Common top level model

A different approach for enabling model level interoperability, is the use of a common top level ontology. One of the project that takes this approach is ABC (Brickley *et al.*, 1999), a common conceptual model to facilitate interoperability among application metadata vocabularies.

ABC aims at the interoperability of multiple metadata packages that may be associated with and across resources. These packages are by nature not semantically distinct, but overlap and relate to each other in numerous ways. It exploits the fact that many entities and relationships - for example, people, places, creations, organizations, events, certain relationships and the like - are so frequently encountered that they do not fall clearly into the domain of any particular metadata vocabulary but apply across all of them. ABC is an attempt to:

- formally define these underlying common entities and relationships;
- describe them (and their inter-relationships) in a s/usr/bin/smbclient -M pelikaanimple logical model;
- provide the framework for extending these common semantics to domain and application-specific metadata vocabularies.

A comparable approach - although more general - is the IEEE Standard Upper Ontology (IEEE SUO Working Group). This standard will specify the semantics of a general-purpose upper level ontology. This will be limited to the upper level, which provides definition for general-purpose terms and provides a structure for compliant lower level domain ontologies. It is estimated to contain between 1000 and 2500 terms plus roughly ten definitional statements for each term. Is intended to provide the foundation for ontologies of much larger size and more specific scope.

These approaches can solve some interoperability, but requires a manual mapping of the ontologies to the common ontology.

4.3 Versioning

We only found one technique that provides support for the ontology versioning problems. Of course, there is a lot of experience with these kind of problems in the area of software engineering and databases, but it is not yet clear whether this can directly applied to web-based ontologies. This should be investigated further.

SHOE ontology integration

SHOE (Heflin and Hendler, 2000) is an is an HTML-based ontology language. The language includes techniques for combining and integrating different ontologies. SHOE provides a rule mechanism to align ontologies. Common items between ontologies can be mapped by inference rules. First, terminological differences can be mapped using simple if-and-only-if rules. Second, scope differences require mapping a category to the most specific category in the other domain that subsumes it. Third, some encoding differences can be handled by mapping individual values. Not all encodings can be mapped in SHOE, for example arithmetic functions would be needed to map meters to feet.

To solve versioning problems, the SHOE project gives version numbers to ontologies and suggest three ways to incorporate the results of an ontology integration effort. These revising schemes allows for the different effects of revisions on the compatibility.

- In the first approach, a new mapping ontology that extends all the existing ones is created; users of the integrated ontology should explicitly conform to the newly created ontology.
- Second, each ontology that is involved in the integration could be revised with the mutual relations to the other ontologies.
- Third, it is possible to create a new intersection ontology, that will be extended by the already existing ontologies. Any source can commit to the ontology of its choice, thus saying that it agrees with any conclusions that logically follow from its statements and the rules in the ontology. Agents are free to pick which ontologies they use to interpret a source, and depending on the differences between these two ontologies they may get the intended meaning or an alternate one.

The SHOE versioning facilities provides both identification of the revisions and an explicit specification of its relation to other versions.

5 Overview of approaches

We will now give an overview of the approaches that are used in the projects and techniques mentioned above and also list some papers that describe a similar approach.

Additionally, Table 1 relates the the tools and approaches that we have discussed to the framework. The table should read as follows: an ‘A’ means ”tool or technique solves this without user interaction (automatically)”, ‘U’ means ”tool or technique suggests solutions to the user, and ‘M’ means ”tool or technique provides a mechanism for specifying the solution to this problem”.

- We discovered four different approaches that aims at enabling **interoperability** between different ontologies at the *language level*.
  - aligning the metamodel: the constructs in the language are formally specified in a general model (Bowers and Delcambre, 2000), (MOF);
  - layered interoperability: aspects of the language are split up in clearly defined layers, as a result of what interoperability can be solved layer by layer (Melnik and Decker, 2000);
  - transformation rules: the relation between two specific constructs in different ontology languages is described with a rule that specifies the transformation from the one to the other (OntoMorph);
  - mapping onto a common knowledge model: the constructs of an ontology language are mapped onto a common knowledge model (OKBC).

  Note that the third approach can be used to implement the fourth.
- We want to recall that the alignment of concepts is a task that requires understanding of the meaning of concepts, and cannot be fully automated. Consequently, at the *model level*, we only found tools that **suggest alignments and mappings** based on heuristics matching algorithms and provide means to specify these mappings. Such tools support the user in finding the concepts in the separate ontologies that might be candidates for merging. Some tools go a little bit further by even suggesting the actions that should be performed. Roughly spoken, there are two types of heuristics:
  - linguistic based matches: terms with the same word-stem, nearby terms in WordNet, or even simpler heuristics, like omitting hyphens and capitalizing all terms, etc. Examples can be found in (Hovy, 1998; Knight and Luk, 1994);

Table 1: Table of problems and approaches for combined use of ontologies

Issues		SKC	Chim.	PROMPT	SHOE	OntoM.	Metamodel	OKBC	Layering
Language level mismatches	Syntax					M	M	M	M
	Representation					M	M	M	M
	Semantics					M	M		M
	Expressivity								
Ontology level mismatches	Paradigm					M			
	Concept description					M			
	Coverage of model								
	Scope of concepts	M	U	U	M	M			
	Synonyms	M	U	U	M	M			
	Homonyms	M				U			
	Encoding	M			M	M			
Practical problems	Finding alignments	U	U	U					
	Diagnosis of results		A	A		A			
	Repeatability	A		A		A			
Ontology versioning	Identification				M				
	Change tracking				M				
	Translation								

  - structural and model similarity: see for example the techniques described in Chimaera and Weinstein and Birmingham (1999).

- A slightly different approach for interoperability at the model is the use of a **common top level ontology**. This approach is only useful if there is a willingness to conform to a common standard.
- There are also different approaches for **diagnosing** or **checking** the results of the alignments. We have seen two types of checks:
  - domain independent verification and validation checks: name conflicts, dangling references, etc. (Chimaera, and others);
  - validation that requires some kind of reasoning: redundancy in the class hierarchy, value restrictions that violate class inheritance, etc. (OntoMorph, PROMPT).
- Several tools support an **executable specification** of the mappings and transformations (SKC, OntoMorph, PROMPT). This allows re-merging of revised ontologies. In this way, the intellectual effort that is invested in finding and formulation the alignments is preserved.
- Finally, most techniques and tools do not deal with **versioning**. Only SHOE elaborates on schemes that enables the combined use of different ontologies. They mention three ways to integrate separate (revisions of) ontologies without invalidating the existing ones.

6 Conclusion and remarks

In this paper, we analyzed the problems that hinder the combined use of ontologies. These problems are of several kinds and may occur at several levels. The analyse of the problems yielded a framework that is used to examine what solutions are provided by current tools and techniques. This examination is still very general, and will be worked out further in the future.

We have seen that there are several approaches that provide reasonable support for language level interoperability. Mismatches in expressiveness between languages are not solvable, and consequently, none of the approaches takes this into account.

The most difficult problems are those of conceptual integration. There are a lot of techniques and heuristics for suggesting alignments. We think that semantic mapping at the model level will remain a task that requires a certain level of human intervention.

Finally, in an open environment such as the Web, versioning methods will be very important. We have seen that this aspect is underdeveloped in most approaches. We think that more comprehensive schemes for interoperability of ontologies are required.

Acknowledgements

We would like to thank Dieter Fensel, Mike Uschold for helpful comments and remarks on previous versions of this paper.

References

Shawn Bowers and Lois Delcambre. Representing and transforming model-based information. In *First Workshop on the Semantic Web at the Fourth European Conference on Digital Libraries*, Lisbon, Portugal, September 18–20, 2000.

D. Brickley and R. V. Guha. Resource Description Framework (RDF) Schema Specification 1.0. Candidate recommendation, World Wide Web Consortium, March 2000.

Dan Brickley, Jane Hunter, and Carl Lagoze. ABC: A logical model for metadata interoperability, October 1999. Harmony discussion note, see `http://www.ilrt.bris.ac.uk/discovery/harmony/docs/abc/abc_draft.html`.

Jeen Broekstra, Michel Klein, Stefan Decker, Dieter Fensel, Frank van Harmelen, and Ian Horrocks. Enabling knowledge representation on the web by extending RDF schema. In *Proceedings of the 10th World Wide Web conference*, Hong Kong, China, May 1–5, 2001.

Hans Chalupsky. OntoMorph: A translation system for symbolic logic. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *KR2000: Principles of Knowledge Representation and Reasoning*, pages 471–482, San Francisco, CA, 2000. Morgan Kaufmann.

Vinay K. Chaudhri, Adam Farquhar, Richard Fikes, Peter D. Karp, and James P. Rice. OKBC: A programmatic foundation for knowledge base interoperability. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98) and of the 10th Conference on Innovative Applications of Artificial Intelligence (IAAI-98)*, pages 600–607, Menlo Park, July 26–30 1998. AAAI Press.

Jérôme Euzenat. Towards a principled approach to semantic interoperability. In Asuncion Gomez-Perez, Michael Gruninger, Heiner Stuckenschmidt, and Michael Uschold, editors, *Workshop on Ontologies and Information Sharing, IJCAI'01*, Seattle, USA, August 4–5, 2001.

Norman Foo. Ontology revision. In Gerard Ellis, Robert Levinson, William Rich, and John F. Sowa, editors, *Proceedings of the 3rd International Conference on Conceptual Structures (ICCS'95): Applications, Implementation and Theory*, volume 954 of *LNAI*, pages 16–31, Berlin, GER, August 1995. Springer.

William E. Grosso, John H. Gennari, Ray W. Fergerson, and Mark A. Musen. When knowledge models collide (how it happens and what to do). In *Proceedings of the 11th Workshop on Knowledge Acquisition, Modeling and Management (KAW '98)*, Banff, Canada, April 18–23 1998.

T. R. Gruber. A translation approach to portable ontology specifications. *Knowledge Acquisition*, 5(2), 1993.

Jeff Heflin and James Hendler. Dynamic ontologies on the web. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 443–449. AAAI/MIT Press, Menlo Park, CA, 2000.

E. H. Hovy. Combining and standardizing large-scale, practical ontologies for machine translation and other uses. In *Proceedings of the 1st International Conference on Language Resources and Evaluation (LREC)*, Granada, Spain, May 28–30 1998.

IEEE SUO Working Group. Standard upper ontology. See `http://suo.ieee.org/`.

H. Kitakami, Y. Mori, and M. Arikawa. An intelligent system for integrating autonomous nomenclature databases in semantic heterogeneity. In *Database and Expert System Applications, DEXA'96*, number 1134 in Lecture Notes in Computer Science, pages 187–196, Zürich, Switzerland, 1996.

Kevin Knight and Steve K. Luk. Building a large-scale knowledge base for machine translation. In *Proceedings of the 12th National Conference on Artificial Intelligence. Volume 1*, pages 773–778, Menlo Park, CA, USA, July 31–August 4 1994. AAAI Press.

Deborah L. McGuinness, Richard Fikes, James Rice, and Steve Wilder. An environment for merging and testing large ontologies. In Anthony G. Cohn, Fausto Giunchiglia, and Bart Selman, editors, *KR2000: Principles of Knowledge Representation and Reasoning*, pages 483–493, San Francisco, 2000. Morgan Kaufmann.

Sergey Melnik and Stefan Decker. A layered approach to information modeling and interoperability on the web. In *Electronic proceedings of the ECDL 2000 Workshop on the Semantic Web*, Lisbon, Portugal, September 21 2000.

Prasenjit Mitra, Gio Wiederhold, and Martin Kersten. A graph-oriented model for articulation of ontology interdependencies. In *Proceedings of Conference on Extending Database Technology, (EDBT 2000)*, Konstanz, Germany, March 2000. Also, Stanford University Technical Note, CSL-TN-99-411, August, 1999.

Natalya Fridman Noy and Mark A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, Austin, TX, 2000. AAAI/MIT Press.

D. E. Oliver, Y. Shahar, M. A. Musen, and E. H. Shortliffe. Representation of change in controlled medical terminologies. *Artificial Intelligence in Medicine*, 15(1):53–76, 1999.

H. Sofia Pinto, Asunción Gómez-Pérez, and João P. Martins. Some issues on ontology integration. In *Proceedings of the Workshop on Ontologies and Problem Solving Methods during IJCAI-99*, Stockholm, Sweden, August 1999.

Mike Uschold, Mike Healy, Keith Williamson, Peter Clark, and Steven Woods. Ontology reuse and application. In N. Guarino, editor, *Formal Ontology in Information Systems (FOIS'98)*, Treno, Italy, June 6-8, 1998. IOS Press, Amsterdam.

Pepijn R. S. Visser, Dean M. Jones, T. J. M. Bench-Capon, and M. J. R. Shave. An analysis of ontological mismatches: Heterogeneity versus interoperability. In *AAAI 1997 Spring Symposium on Ontological Engineering*, Stanford, USA, 1997.

Peter C. Weinstein and William P. Birmingham. Comparing concepts in differentiated ontologies. In *Proceedings of the 12th Workshop on Knowledge Acquisition, Modeling and Management (KAW '99)*, Banff, Canada, October 16–21, 1999.

Gio Wiederhold. An algebra for ontology composition. In *Proceedings of 1994 Monterey Workshop on Formal Methods*, pages 56–61, U.S. Naval Postgraduate School, Monterey CA, September 1994.