Alignment of Heterogeneous Ontologies: A Practical Approach to Testing for Similarities and Discrepancies

Neli P. Zlatareva
Department of Computer Science, Central Connecticut State University
1615 Stanley Street, New Britain, CT 06050, USA
zlatareva@ccsu.edu

Maria Nisheva
Faculty of Mathematics and Informatics, Sofia University "St. Kliment Ohridski"
5 James Bourchier Blvd., Sofia, Bulgaria
marian@fmi.uni-sofia.bg

Abstract

Ontology alignment is regarded as one of the core tasks in many Web services. It is concerned with finding the correspondences between separate ontologies by identifying concepts with the same or similar semantics in order to resolve semantic heterogeneity between them. Existing ontology alignment techniques are tailored towards today's ontology languages, which are not capable of representing and reasoning with uncertain or incomplete information. It is expected, however, that future Semantic Web services will rely on the development and use of proper domain ontologies. Alignment of such ontologies goes beyond standard concept matching, and requires non-standard logic processing. In this paper, we present an alignment technique utilizing an alternative, rule-based representation, which provides a uniform framework for representing and mapping heterogeneous ontologies. To justify and illustrate our research, we describe an example application scenario.

Introduction

The World Wide Web (WWW) holds an enormous amount of information and provides an astonishing number of information services, which nowadays are primarily intended to be used by people. The next generation of the WWW, the Semantic Web, will make this information accessible to computers by annotating and explicating it using ontologies "that are composed of concepts that are to some extent valid in a domain, relations that hold to some degree of certainty, and rules that apply only in some cases" (Davis, Studer, and Warren 2006). It is expected that a huge number of Web ontologies will be created, some of them intended to be used by the same Semantic Web services. This can only be possible if the consistency and interoperability between cooperating ontologies is assured. The process, referred to as *ontology alignment* (de Bruijn et al. 2006), is intended to carry out this task. It is commonly implemented by the so-called *Match* operator, which takes two or more ontologies as an input, and returns a specification of the correspondences between them. Depending on the way ontologies are compared, we distinguish between *schema-based* and *instance-based* matching. The former accounts for different properties of the concepts (such as name, for example) and uses similarity measures to evaluate the correspondence between them (Noy and Musen 2000), while the latter compares instances of the concepts (Doan et al. 2004). Some alignment algorithms, such as the one presented in (Giunchiglia and Shvaiko 2003), compare the structure of ontologies based on the mapping between their elements. All of these alignment algorithms are tailored towards current ontology languages, which were shown to have limited representational and computational power to fulfill the needs of a broad range of Web services dealing with incomplete and/or uncertain information, or requiring more sophisticated reasoning capabilities (beyond those offered by Description Logics upon which they are built on). Such capabilities, ranging from conventional monotonic backward and forward chaining to various types of default reasoning, are inherent to rule-based languages, traditionally used for building knowledge-based systems and intelligent agents. Rules have already been shown to be very effective in various Web applications such as identifying matches between different data sets, e-learning, multimedia collection indexing, skill finding, device interoperability, etc. (Dean 2004; Antoniou and van Harmelen 2004). It is widely admitted that rules will be a major part of the Semantic Web, and critical for adapting AI techniques in a broad range of Web applications.

In this paper, we describe a hypothetical application scenario which illustrates the need for a rule-based representation capable of expressing heterogeneous (mixed) ontologies in a uniform fashion, and supporting non-monotonic reasoning. The problem addressed here is a special case of ontology alignment, where we are interested not only in establishing the correspondences

between two ontologies, but also in identifying the discrepancies between them and explicating the culprits for those discrepancies. Because ontologies can be encoded in different formats, we advocate that finding an alternative representation in which cooperating ontologies can be adequately mapped, will greatly simplify their alignment. We show how the Contradiction-tolerant Truth Maintenance System (CTMS) (Zlatareva 1992) can be utilized as a common representation for cooperating ontologies, and how its inference engine can be adapted to carry out the alignment procedure.

Motivation Example

Consider a university domain, where programs are described in terms of their basic curriculum as taxonomies of courses, and a course catalog provides course descriptions. Figures 1 and 2 illustrate computer science curriculum at universities A and B.

Assume John is interested to transfer out of university A, but he wants to make the most of the credits acquired there. In addition to the CS courses taken at university A, John has taken some non-CS courses, which are common prerequisites for CS courses. Since those are not part of the basic CS-course taxonomy, they are not shown on Figures 1 and 2. John is looking for a CS program which offers a specialization track in AI, and he is especially interested in a course on Semantic Web. University B is identified by him (or by his helper Web agent) as a possible choice. Can John transfer the two senior courses (Computer Architecture and Networking) that he has already taken at university A, and can he continue with AI specialization track without taking extra prerequisites at university B?

The first step towards processing John's query is to set up a common semantic framework for the two ontologies. This is typically done by establishing the so-called *semantic bridges* (Maedche et al. 2002; Ghidini and Serafini 2006) in a form of rules, which allow entities (concepts, relations, etc.) from one ontology to be connected to the entities of the other. This process is not always trivial. Consider, for example, university A course called "Data Structures" and university B course called "JAVA Programming 2". Note also that there is another course at university B called "Data Structures". The mapping procedure must be able to bridge the university A "Data Structures" course to university B "JAVA Programming 2" course rather than to university B "Data Structures" course. Finding a mapping between relations of the course ontologies should allow queries like "Is the networking course at university A eligible for transfer to university B?" to be answered.

As pointed out in (Ghidini and Serafini 2006), current ontology languages make it possible to express mappings between homogeneous components of different ontologies (that is, concepts to concepts, relations to relations, etc.). However, in many Semantic Web applications a need may arise to establish semantic relations between heterogeneous components. In our example, some of the relations between the courses must be acquired from course catalogs, which are not part of the course taxonomy.

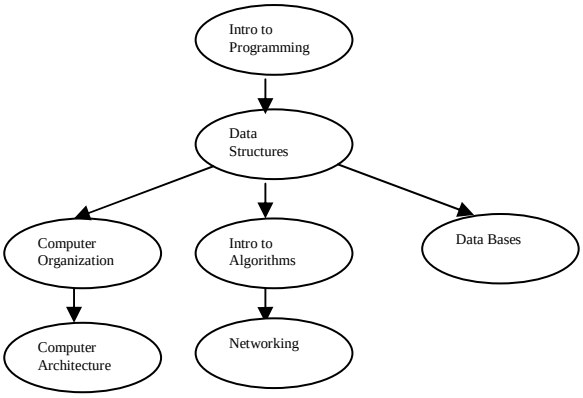

Figure 1: A taxonomy of courses at university A

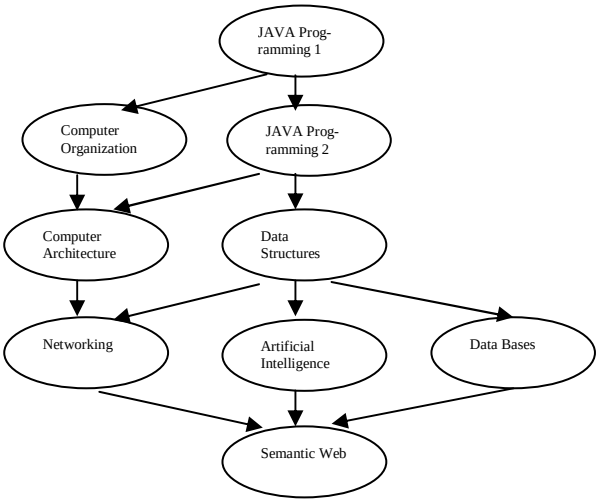

Figure 2: A taxonomy of courses at university B

Next, we define a specification format for representing heterogeneous ontologies and describe a simple mapping procedure to establish correspondences between their concepts and relations.

Specification of Heterogeneous Ontologies

In heterogeneous ontologies, data is not specified in a common format. Assume some data is defined declaratively, and other data is defined procedurally. Here is our working definition for an ontology, which accounts for both data specifications.

Definition 1: Let ontology O = {SchemaSet ∪ RuleSet}, where:

a) SchemaSet is a set of concepts describing classes of entities in a domain $\{C_1, C_2, \ldots, C_k\}$, such that $C_i = <N_i, D_i, S_i>$, where:
   - N_i is a term (the name of the concept);
   - D_i is a list of "property – value" pairs providing the syntactic definition of the concept;
   - S_i is a list of semantically equivalent to N_i terms.

b) RuleSet is a set of implications representing relations between concepts.

In our example domain, formal concept definitions can be acquired from informal course descriptions using keywords, and can be represented in the following format:

C_i = <CS-designator,
<CS-prereqs : {CS-designator},
non-CS-prereqs : {non-CS-designator},
credits : number>, {CS-designator}>

In XML, this format is defined as follows:

```
<!ELEMENT course (course_name, course_attributes,
                  equiv_names)>
<!ELEMENT course_name (#PCDATA)>
<!ELEMENT course_attributes (CS_prereqs,
                  non_CS_prereqs, credits)
<!ELEMENT CS_prereqs (prereq_item*)>
<!ELEMENT prereq_item (#PCDATA)>
<!ELEMENT non_CS_prereqs (prereq_item*)>
<!ELEMENT credits (#PCDATA)>
<!ELEMENT equiv_names (equiv_item*)>
<!ELEMENTS equiv_item (#PCDATA)>
```

Example descriptions of CS courses at universities A and B, respectively, are given below:

```
<course>
  <course_name>Data Structures</course_name>
  <course_attributes>
     <CS_prereqs>
        <prereq_item>Introduction to
                     Programming</prereq_item>
     </CS_prereqs>
     <non_CS_prereqs></non_CS_prereqs>
     <credits></credits>
  </course_attributes>
  <equiv_names>
     <equiv_item>CS-2</equiv_item>
     <equiv_item>Object-Oriented-
                 Programming</equiv_item>
  </equiv_names>
</course>
 <course>
  <course_name>JAVA-Programming 2</course_name>
  <course_attributes>
     <CS_prereqs>
        <prereq_item>JAVA-Programming-1
        </prereq_item>
     </CS_prereqs>
     <non_CS_prereqs></non_CS_prereqs>
     <credits></credits>
  </course_attributes>
  <equiv_names>
     <equiv_item>CS-2</equiv_item>
     <equiv_item>Functional-Programming</equiv_item>
  </equiv_names>
</course>
```

A simple matching procedure will be sufficient in this case to establish name equivalences between concepts. In our example ontologies, the following list of name equivalences will be returned:

{Intro-to-Programming ≅ JAVA-Programming-1 ≅ CS-1,
Data Structures ≅ JAVA-Programming-2 ≅
≅ Functional- Programming ≅ CS-2,
Intro-to-Algorithms ≅ Data-Structures ≅ CS-3}

The second component of our domain specification, namely the relations between concepts, is shown as course taxonomies on Figures 1 and 2. Rules are natural way to represent and process such taxonomies. It is interesting to note, however, that an adequate representation of domain semantics requires that we distinguish between two types of rule (course) prerequisites:

1. Prerequisites explicitly shown in course taxonomies (that is, reflecting relations between CS courses). We can interpret those as "required" (John cannot take an AI course at university B without a Data Structures course; recall, however, that the later is not the same as the Data Structures course at university A).
2. Prerequisites not shown in course taxonomies, but spelled out in course descriptions (that is, reflecting "CS – non-CS" relations). We can interpret those as "desirable" (John is expected to have a Discrete Math course in order to take AI at university B; however, he may still be allowed to take AI without Discrete Math with AI professor's permission, but John may not get it, in which case he cannot take the course).

Alignment of Web Ontologies: Basic Definitions and Notation

Let $O_1 = \{SchemaSet_1 \cup RuleSet_1\}$ and $O_2 = \{SchemaSet_2 \cup RuleSet_2\}$ be two propositional ontologies. Then, the degree of correspondence between O_1 and O_2 is defined as follows.

Definition 2: O_1 and O_2 are ***fully compatible*** iff:

a) The syntactic definitions of the concepts comprising their schema sets match, i.e.
   i. $\forall\ C_i(1) \in SchemaSet_1 \rightarrow \exists\ C_j(2) \in SchemaSet_2$, such that $N_i(1) = N_j(2)$ or $N_i(1) \in \{S_{j2}, \ldots, S_{jk}\}(2)$.
   ii. $\forall\ C_j(2) \in SchemaSet_2 \rightarrow \exists\ C_i(1) \in SchemaSet_1$, such that $N_j(2) = N_i(1)$ or $N_j(2) \in \{S_{i1}, \ldots, S_{il}\}(2)$.
b) Transitive closures of O_1 and O_2 contain only semantically equivalent sets of concepts, i.e. concepts which derivation paths are exactly the same. We shall say that such concepts **strongly agree**.

Definition 3: O_1 and O_2 are ***partially compatible*** iff:

a) A subset of concepts comprising $SchemaSet_1$ and $SchemaSet_2$ match.
b) Transitive closures of O_1 and O_2 contain subsets of concepts that strongly agree.

Definition 4: O_1 and O_2 are ***incompatible*** if there exists a concept from $SchemaSet_1$ which semantically contradicts a concept from $SchemaSet_2$, and all other concepts depend on them.

If two ontologies are fully or partially compatible, their complete or partial alignment is possible; incompatible ontologies can not be aligned.

The rest of the paper presents a effective procedure intended to test if two ontologies are fully or partially compatible. It returns not only the correspondences, but also the discrepancies between participating ontologies, and provides an explanation to justify the detected semantic similarity between them. The underlying idea is to translate the aligned heterogeneous ontologies into an alternative homogeneous representation where implicit relations between concepts are explicated and processed. We use the Contradiction-tolerant Truth Maintenance System (Zlatareva 1992) as a common representation framework, and show how its inference engine can be utilized to carry out the alignment task.

Representing Web Ontologies as CTMS Rules

The Contradiction-tolerant TMS (CTMS) was originally introduced as an alternative to other non-monotonic formalisms to allow for: (i) efficient processing of some types of non-monotonic theories, (ii) maintaining statements with different degrees of belief, not just *true*, *false*, and *unknown*; and (iii) reasoning in the presence of a logical contradiction, instead of halting and waiting until the contradiction is resolved. Implementation of these features is largely due to the fact that CTMS employs two types of inference rules, T-rules and P-rules. T-rules are regular monotonic rules, while P-rules are non-monotonic rules of the form:

(Premise-1, …, Premise-n) (Assumption-1, …, Assumption-m) ➔ Conclusion

Here Premise-1, …, Premise-n are monotonic supporters comprising the minimal evidence for Conclusion, and Assumption-1, …, Assumption-m provide additional evidence intended to strengthen the truth of Conclusion. Such rules will fire if the minimal evidence for Conclusion has been established. In turn, Conclusion will be derived with different degrees of belief depending on the accumulated evidence in its favor. For example, given the rule (BirdTweety) (¬PenguinTweety, ¬OstrichTweety) ➔ FliesTweety, if BirdTweety is true then we can derive FliesTweety, even though we do not know if Tweety is a penguin or an ostrich. The rule (PositiveResultOfTheBodyScanner) (Headache, Neurosis, MentalDisturbances) ➔ DiagnoseBrainTumor, implies a brain tumor as a possible diagnosis if PositiveResultOfTheBodyScanner is true, even though additional symptoms such as headache, neurosis and mental disturbances, which usually accompany the disease, are not observed in a particular case. If one, or more, or all in the extreme case, of the assumptions of such rules become true, then the degree of belief in Conclusion should increase. In such cases, the so-called *duplicate rules* are used instead of original P-rules. Duplicate rules are variants of the corresponding P-rules to account for the additional evidence accumulated for Conclusion and revise its belief status accordingly. For more on CTMS syntax and semantics, see (Zlatareva 1992).

Relative to our example domain, CTMS-rules have the following format:
(CS-1, …,CS-n) (non-CS-1, …, non-CS-m) → CS-i

Here CS-1, …, CS-n are the required prerequisites for CS-i acquired from course taxonomies shown on Figures 1 and 2, and non-CS-1, …, non-CS-m are "desired" or "assumed" prerequisites acquired from concept definitions. If such rule fires, conclusion CS-i will be recorded together with its justification as follows:
CS-i: (CS-1, …, CS-n) (non-CS-1, …, non-CS-m).

The resulting sets of CTMS rules describing example ontologies are shown below.

University A rules:

Rule 1A: (CS-2) (Web-Technologies) ➔ Data-Bases
Rule 2A: (CS-2) ( ) ➔ CS-3
Rule 3A: (CS-3) (Web-Technologies) ➔ Networking

Rule 4A: (Computer-Organization) ( ) ➔
➔ Computer-Architecture
Rule 5A: (CS-1) ( ) ➔ CS-2
Rule 6A: ( ) (Calculus) ➔ Intro-to-Programming
Rule 7A: (CS-2) ( ) ➔ Computer-Organization

University B rules:

Rule 1B: (CS-3) (Statistics) ➔ Data-Bases
Rule 2B: (CS-2) (Discrete-Math) ➔ CS-3
Rule 3B: (CS-3, Computer-Architecture) ( ) ➔
➔ Networking
Rule 4B: (Computer-Organization, CS-2) ( ) ➔
➔ Computer-Architecture
Rule 5B: (CS-1) (Calculus) ➔ CS-2
Rule 6B: ( ) ( ) ➔ CS-1
Rule 7B: (CS-1) ( ) ➔ Computer-Organization
Rule 8B: (CS-3) (Statistics) ➔ Artificial-Intelligence
Rule 9B: (Networking, Data-Bases, Artificial-Intelligence)
( ) ➔ Semantic-Web

Next, by running the CTMS inference engine on the two rule sets, we can explicate the immediate as well as all other course predecessors. Assuming that all required conditions for rule firing hold, we can first compute the stable extensions of the two rule sets, and then compute their so-called *grounded stable extensions* (GSEs). As described in the next section, GSEs contain useful information for explicating the correspondences and discrepancies between the two source ontologies.

Testing CTMS Rule Sets for Similarities and Discrepancies

The stable extension of a CTMS rule set shows how derived formulas depend on their immediate predecessors. To explicate all of the predecessors of a given formula, we compute the transitive closure of its immediate predecessors. The resulting set of formulas comprises the GSE of the CTMS rule set (Zlatareva 1992). Processing of our example ontologies converted into CTMS representation results in the following GSEs:

GSE(A) = {CS-1 : ( ) (Calculus), CS-2 : (CS-1) (Calculus), CS-3 : (CS-2, CS-1) (Calculus),
Computer-Organization : (CS-2, CS-1) (Calculus),
Data-Bases : (CS-2, CS-1) (Calculus, Web-Technologies),
Networking : (CS-3, CS-2, CS-1) (Calculus, Web-Technologies), Computer-Architecture : (Computer-Organization, CS-2, CS-1) (Calculus)}.

GSE(B) = {CS-1: ( ) ( ),
Computer-Organization:(CS-1)( ), CS-2:(CS-1) (Calculus), CS-3: (CS-2, CS-1) (Discrete-Math, Calculus),
Computer-Architecture: (Computer-Organization, CS-2, CS-1) (Calculus), Data-Bases: (CS-3, CS-2, CS-1) (Discrete-Math, Calculus, Statistics), Networking: (CS-3, CS-2, CS-1, Computer-Architecture, Computer-Organization) (Discrete-Math, Calculus),
Artificial-Intelligence : (CS-3, CS-2, CS-1) (Statistics, Discrete-Math, Calculus), Semantic-Web : (Networking, CS-3, CS-2, CS-1, Computer-Architecture, Computer-Organization, Data-Bases, Artificial-Intelligence) (Calculus, Statistics, Discrete-Math)}

Recall that we have already identified name equivalences between concepts. Now we can use this information for establishing the semantic relation between them. For that, we compare justifications of the formulas describing courses with the same name from GSE(A) and GSE(B). The following three cases are possible:

Case 1. The two justifications are exactly the same. For example,
Computer-Architecture: (Computer-Organization, CS-2, CS-1) (Calculus) ∈ GSE(A)
Computer-Architecture: (Computer-Organization, CS-2, CS-1) (Calculus) ∈ GSE(B)

In this case, the two concepts Computer-Architecture(A) and Computer-Architecture(B) ***strongly agree***.

Case 2. The two justifications differ in their assumption lists only. For example,
CS-3 : (CS-2, CS-1) (Calculus) ∈ GSE(A)
CS-3 : (CS-2, CS-1) (Discrete-Math, Calculus) ∈ GSE(B)

In this case, we say that the two concepts, CS-3(A) and CS-3(B), ***partially agree***, and that CS-3(B) is ***stronger*** than CS-3(A) (that is, CS-3(A) < CS-3(B)).

Case 3. The two justifications differ in their required lists. For example,
Data-Bases: (CS-2, CS-1) (Calculus, Web-Technologies) ∈ GSE(A)
Data-Bases: (CS-3, CS-2, CS-1) (Calculus, Statistics, Discrete-Math) ∈ GSE(B)

In this case, we say that the two concepts, Data-Bases(A) and Data-Bases(B), are ***inconsistent***.

By the definitions of full and partial compatibility of two ontologies, introduced earlier in this paper, we can say that:

- O_1 and O_2 are **fully compatible** iff their GSEs contain only concepts that strongly agree.
- O_1 and O_2 are **partially compatible** iff their GSEs contain concepts that strongly or partially agree.
- O_1 and O_2 are **incompatible** iff their GSEs contain only concepts that are either inconsistent, or contain

inconsistent required prerequisites in their justifications.

Going back to John's query, the presented alignment technique will return the following results:

- CS-2(A) and CS-2(B), and Computer-Architecture(A) and Computer-Architecture(B) **strongly agree**.
- CS-1(A) and CS-1(B), and CS-3(A) and CS-3(B) **partially agree**. Note that CS-1(A) > CS-1(B), so John can transfer it to university B. However, CS-3(A) < CS-3(B), and thus John can not be certain about whether or not he will be allowed to transfer this one.
- The following three concepts are identified as **inconsistent**: Computer-Organization,Data-Bases, and Networking. The interpretation of such inconsistencies depends on the semantics of a posted query.

Inconsistencies are of special interest in our example application. Since the sources of detected inconsistencies have already been identified and recorded in course justifications, we can further process them to clarify the final response to John's query. First, consider the justifications for Computer-Organization concept:

Computer-Organization:(CS-2,CS-1) (Calculus) ∈ GSE(A)
Computer-Organization:(CS-1) ( ) ∈ GSE(B)

Computer-Organization(A) > Computer-Organization(B), because the required prerequisites of the latter are a subset of the required prerequisites of the former. Therefore, John must be allowed to transfer his Computer Organization course to university B. Now, compare the justifications for Data-Bases concept:

Data-Bases: (CS-2,CS-1)(Calculus, Web-Technologies)∈
∈ GSE(A)
Data-Bases: (CS-3, CS-2, CS-1) (Calculus, Statistics,
Discrete-Math) ∈ GSE(B)

Here Data-Bases(B) > Data-Bases(A), because (CS-2, CS-1) ⊆ (CS-3, CS-2, CS-1). Therefore, John will not be allowed to transfer this course to university B.

Conclusion

The paper addressed a special case of ontology alignment, which is one of the core tasks in many Web services. The presented alignment technique aims not only to establish the correspondences between two cooperating ontologies, but also to identify the discrepancies between them and explicate the culprits for those discrepancies. We have shown how heterogeneous ontologies comprised of concepts that are not fully specified and relations that are characterized with some degree of uncertainty, can be uniformly mapped into CTMS representation, and have shown how CTMS inference engine can be utilized to implement the alignment process.

Acknowledgement. Neli Zlatareva would like to thank the Fulbright Commission for Educational Exchange, under the sponsorship of which this joint research was made possible.

References

Davis J., R. Studer, P., and Warren, P. 2006. Conclusion and Outlook. In Davis, S. and Warren (eds.) *Semantic Web Technologies: Trends and Research in Ontology-Based Systems.* John Wiley and Sons.

de Bruijn, J. et al. 2006. Ontology Mediation, Merging and Alignment. In Davis, S. and Warren (eds.) *Semantic Web Technologies: Trends and Research in Ontology-Based Systems.* John Wiley and Sons.

Noy, N. and Musen, M. 2000. PROMPT: Algorithm and Tool for Automated Ontology Merging and Alignment. In *Proc. AAAI'2000*, Austin, Texas.

Doan, A. et al. 2004. Ontology Matching: A Machine Learning Approach. In Staab S, Studer R. (eds.) *Handbook on Ontologies in Information Systems*, Springer-Verlag.

Giunchiglia, F. and Shvaiko, P. 2003. Semantic Matching. *Knowledge Engineering Review*, 18(3): 265–280.

Dean, M. 2004. Semantic Web Rules: Covering the Use Cases. In G. Antoniou and Boley H. (eds.) *Rules and Rule Markup Languages for the Semantic Web*.Springer-Verlag.

Antoniou, G. and van Harmelen, F. 2004. A *Semantic Web Primer*. MIT Press.

Zlatareva, N. 1992. CTMS: A General Framework for Plausible Reasoning. In *International Journal of Expert Systems: Research and Applications*, 5(4).

Maedche, A. et al. 2002. MAFRA – A Mapping Framework for distributed Ontologies. In *Proc. 13th European Conference of Knowledge Engineering and Knowledge Management (EKAW'2002)*, Madrid, Spain.

Ghidini, C. and Serafini, L. 2006 Reconciling concepts and relations in heterogeneous ontologies. In *Proc. 3rd European Semantic Web Conference (ESWC'2006)*, Budva, Montenegro.