Making Ontologies Talk: Knowledge Interoperability in the Semantic Web

Monika Lanzenberger, *Vienna University of Technology*
Jennifer Sampson, *Epsis AS*

Ontologies offer shared vocabularies. They're key to agent cooperation and seamless integration of knowledge systems, and they're fundamental to the Semantic Web. They let us precisely define the domain of a knowledge-based system. And they're increasing in number. With this increase, the need for new tools and techniques to reconcile different ontologies becomes crucial.

Ontology matching and alignment help establish agreement between different knowledge representations. The essays here exemplify some of the creative ways researchers are extending the state of the art in algorithms that can establish correspondences between different but related ontologies. The essays are based on six of the 12 papers accepted for the First Workshop on Ontology Alignment and Visualization, held in conjunction with the 2008 International Conference on Complex, Intelligent, and Software-Intensive Systems. The approaches described make original use of techniques such as queries, tagging, graph theory, and information visualization.

Ontology Interoperability and Matching Applications

Besides concepts, properties, and instances, axioms are essential ontology components. Frédéric Fürst and Francky Trichet describe a way for ontology matching to effectively take these axioms into account. They introduce the Ontology Conceptual Graphs Language, a graph-based knowledge representation and reasoning formalism. Their TooCom tool supports the definition of concepts and relations and the specification of axioms in a graphical way. They apply graph-theoretic operations to detect analogies between axioms of different ontologies.

Horst Kargl and Manuel Wimmer describe how to improve the quality of the simple one-to-one correspondences that constitute the typical output from automatic schema-matching tools. In an effort to address some shortcomings associated with existing schema-matching approaches, the authors present SmartMatcher, an orthogonal extension of these approaches that uses real-world examples to evaluate and improve computed alignments. They also introduce their prototype implementation for schemas defined in the Eclipse Modeling Framework.

For users to see ontology mapping as a benefit rather than an inconvenience, Colm Conroy, Declan O'Sullivan, David Lewis, and Rob Brennan look at interaction processes and user interfaces. Their focus is ontology mapping for casual Web users. They break the mapping process down into small tasks and apply a tagging approach. A small user experiment indicates that nonexperts can use their approach to produce mapping results on a par with ontology experts.

Challenges and Visions in Ontology Matching

José Ángel Ramos-Gargantilla and Asunción Gómez-Pérez survey different approaches to ontology mapping. They have designed an XML schema representation for covering mappings and their uses in the Semantic Web. Accordingly, their approach can be used to represent mappings that include not only ontologies but also other knowledge representations such as relational databases, thesauri, and so on.

Jérôme Euzenat, Axel Polleres, and François Scharffe propose to extend the SPARQL query language to express mapping between ontologies. They use SPARQL queries as a mechanism for translating RDF data of one ontology to another. Such functionality lets users exploit instance data described in one ontology while they work with an application that's been designed for another. The authors present an example translation of FOAF (friend-of-a-friend) files into vCards that shows how to use queries to extract data from the source ontology and generate new data for the target ontology.

Ontology alignment and matching still raise more questions than practical solutions for a broader audience. With the last essay, Konstantinos Kotis and Monika Lanzenberger give an overview of current dilemmas and crucial challenges in ontology matching. Their essay includes a sidebar for further resources, including the proceedings from which these essays originated.

Although further research is necessary for improving state-of-the-art algorithms and tools, ontology matching offers many ideas for supporting data interoperability. We hope you enjoy reading about some of them here.

Monika Lanzenberger is an assistant professor with the Institute of Software Technology and Interactive Systems at the Vienna University of Technology. Contact her at monika.lanzenberger@ifs.tuwien.ac.at.

Jennifer Sampson is a senior ontology engineer with Epsis in Bergen, Norway. Contact her at jsa@epsis.no.

1541-1672/08/$25.00 © 2008 IEEE
Published by the IEEE Computer Society

Ontology Matching with Axioms and Conceptual Graphs

Frédéric Fürst,
University of Picardie–Jules Verne
Francky Trichet, *University of Nantes*

Strategies for matching ontologies are diverse, but most of them consider only alignment between *lightweight ontologies*—that is, ontologies composed of concept and relation taxonomies. *Heavyweight ontologies* additionally include axioms to represent a domain's full semantics.[1] Not many real-world ontologies currently make substantial use of axioms, but the full functioning of the Semantic Web requires computers to have access both to structured collections of information and to sets of inference rules that support automated reasoning. So we think the need to develop heavyweight ontologies will inevitably increase. The World Wide Web Consortium's work toward standardizing the Semantic Web Rule Language, for example, is one instance of this trend.

We're working to define an ontology-matching approach based on the explicit use of all components of a heavyweight ontology. Our approach requires the explicit representation of axioms at the conceptual level, as opposed to the operational level, where most ontological engineering represents them. For instance, the Protégé knowledge-modeling environment uses the Protégé Axiom Language (PAL) to represent axioms directly via rules or constraints with fixed and predefined operational semantics.

Semantically speaking, finding a match for an axiom's operational form is difficult. At the conceptual level, an axiom has a formal semantics but not an operational one. At the operational level, an axiom has both formal and operational semantics, and the latter clearly limits reuse. An axiom's operational semantics, represented through a set of rules and constraints, expresses the way a computer can use the axiom to reason, whereas the formal semantics expresses how the axiom constrains the interpretation of its primitives—that is, the concepts and relations.[2]

Ontology Conceptual Graphs Language

To represent heavyweight ontologies at the conceptual level, we use the Ontology Conceptual Graphs Language (OCGL).[2] This modeling language is based on a graphical syntax inspired by conceptual graphs (CGs). First introduced as an operational knowledge-representation model,[3] CGs belong to the semantic-networks field and are mathematically grounded in both logics and graph theory.

OCGL is based on three building blocks: concepts, relations, and axioms. Representing an ontology in OCGL consists mainly in specifying a domain's conceptual vocabulary and specifying this vocabulary's semantics through axioms.

The conceptual vocabulary consists of a set of concepts and a set of relations that can be structured using well-known conceptual properties (schemata axioms) and domain axioms. Schemata axioms represent classical concept and relation properties, whereas domain axioms are totally specific to a domain. In our work, the term *axiom* means the union of these two axiomatic properties.

Figure 1 shows the OCGL graph representing the axiom "The enemy of my friend is my enemy." This is a domain axiom that can't be represented using classical properties. Compare it to the axiom "The friend of my friend is my friend," which is an OCGL schemata axiom that's represented by the transitivity of the relation called Friend(Human, Human).

OCGL has been implemented in TooCom (Tool to Operationalize an Ontology with the Conceptual Graph Model), a tool for editing and operationalizing domain ontologies. TooCom is available under the GNU GPL license at http://sourceforge.net/projects/toocom. It supports the definition of concepts and relations and the specification of schemata and domain axioms in a graphical way.[4]

MetaOCGL: An Ontology of Representation

To detect analogies between axioms represented as graphs, and then to detect analogies between the primitives corresponding to the graph nodes, axioms are transcribed from OCGL to a more abstract form that preserves the graphs' topological structures. These abstract representations are based on MetaOCGL, an ontology of representation. MetaOCGL expresses the OCGL language ontology in OCGL and is therefore a metalevel ontology.[5] MetaOCGL includes

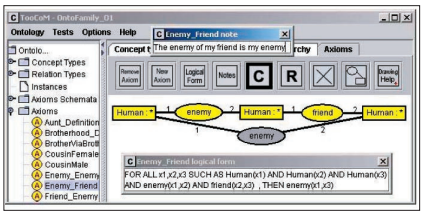

Figure 1. Representation of an axiom in TooCom.

- MetaOCGL concepts to represent OCGL primitives,
- MetaOCGL relations to represent the links between OCGL primitives,
- MetaOCGL schemata axioms used mainly to describe the properties of OCGL relations, and
- MetaOCGL domain axioms to express the formal OCGL semantics.

A MetaOCGL instance—that is, a MetaOCGL graph—can represent a domain ontology, just as OCGL graphs can represent domain facts. The MetaOCGL graph representing an ontology contains one part dedicated to the concept hierarchy's representation, one part dedicated to the relation hierarchy's representation, and as many parts as axioms in the ontology.

Figure 2 (see next page) shows the MetaOCGL graphs representing two axioms—"The enemy of my enemy is my friend" and "The enemy of my friend is my enemy"—and their corresponding metagraphs in MetaOCGL. TooCom automatically provides the MetaOCGL representation of an OCGL ontology. Correspondences between the domain-level and metalevel concepts appear in gold. The type-identity links denote domain-level nodes that are similar—that is, they have the same type. At the metalevel, the two graphs are similar without considering type-identity links; but with these links, they differ because the relations of the axiom Enemy-Enemy have the same type but the relations of the axiom Enemy-Friend do not.

The CG *projection* operator performs the comparisons between axioms represented in MetaOCGL. The projection operator is a graph-theoretic operation corresponding to homomorphism, which is sound and complete with regard to deduction in first-order logic. A projection from a graph G1 into a graph G2 is a specific graph morphism that can restrict the labels of the vertices; it corresponds to a logical implication between G1 and G2.

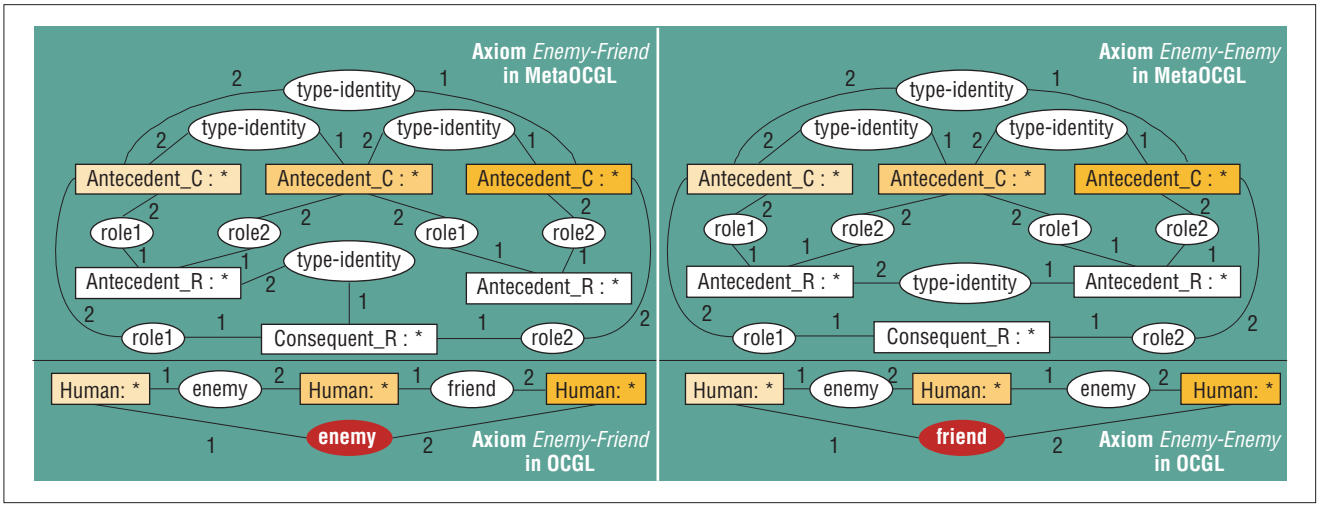

Figure 2. Two axioms represented in MetaOCGL.

Given two graphs G1 and G2, which represent two axioms A1 and A2 in MetaOCGL, if two projections exist from G1 into G2 and from G2 into G1, then A1 and A2 have the same structure. In this case, A1 and A2 express the same property type, and the analogy between the two axioms can extend to the primitives that appear in the axioms.

Axiom-Based Semantic Matching

Ontology matching aims to discover and evaluate semantic links between conceptual primitives of two given ontologies supposedly built on related domains. Our approach relies on using the ontologies' axiomatic level to discover semantic analogies between primitives that will reveal identities between them and calculate the similarity coefficient of these identities.

We use both schemata axioms and domain axioms to evaluate or discover primitive matchings. Each OCGL schemata axiom owns a predefined weight that modulates the axiom's influence on the matching process. The end user can modify the set of weights according to the kinds of ontologies or subjective preferences. Thus, these weights are algorithm parameters that users can change graphically to improve their results' precision.

Space limitations prevent us from presenting details on the use of the schemata and domain axioms to evaluate matchings, but details are elsewhere.[6]

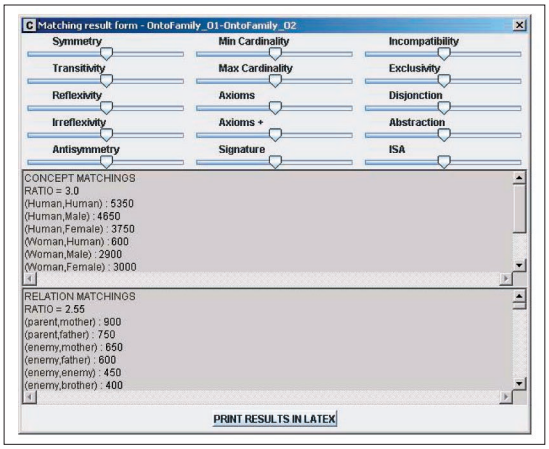

Figure 3. Experimental results in TooCom.

Experimental Results

Figure 3 shows an extract from applying our approach to two ontologies related to family relationships. This limited domain includes these notions: father, mother, grandfather, grandmother, son, daughter, cousin, nephew, niece, uncle, aunt, sister, brother, wife, husband, friend, and enemy. This example is easy to understand and necessarily requires domain axioms for defining such notions as "An aunt is either a female sibling of one of one's parents or the wife of an uncle who is the male sibling of a parent," and for specifying relations between notions such as "The enemy of my enemy is my friend." In other words, schemata axioms aren't sufficient for representing all the domain knowledge. It also means that OWL can't represent the ontologies (available in the XML storage format used for OCGL at http://sourceforge.net/projects/toocom).

In Figure 3, all the weights have the value 50. The TooCom interface (upper part of Figure 3) makes it possible to directly visualize the consequences (on the matchings) when modifying the values of the matching algorithm's property weights. This shows that the matching process itself is not sensitive to the weights assigned to the OCGL properties.

Moreover, it shows that TooCom provides a first step toward cognitive support for ontology mapping.[7] Indeed, very little research has addressed cognitive support for ontology mappings; researchers have focused on improving the performance of the algorithms themselves, largely ignoring the issue of end-user tools.

To deal with this new problem, Sean Falconer, Natalya Noy, and Margaret-Anne Storey have identified a set of 13 end-user tasks for an ontology-mapping tool.[7] Although our user-centered interface is perfectible, the current TooCom version already supports many of these tasks—for example, incremental navigation, browsable lists of candidate mappings, and conflict resolution/inconsistency detection.

Our method has the advantage of incorporating most descriptive features of a heavyweight ontology into the matching process, whereas current methods usually cover only subsets of a lightweight ontology. Of course, we know that our method, although applicable, isn't efficient for lightweight ontologies. However, as the need

for developing heavyweight ontologies increases over time, so will the need to focus on developing matching techniques dedicated to the reasoning power these ontologies can bring to the Semantic Web.

Acknowledgments

A longer version of this essay appeared in the *Proc. 2nd Int'l Conf. Complex, Intelligent, and Software-Intensive Systems* (CISIS 08), IEEE CS Press, pp. 853–858.

References

1. S. Staab and A. Maedche, *Axioms Are Objects Too: Ontology Engineering beyond the Modeling of Concepts and Relations*, research report 399, Inst. AIFB, Univ. Karlsruhe, 2000.
2. F. Fürst, M. Leclère, and F. Trichet, "Operationalizing Domain Ontologies: A Method and a Tool," *Proc. 16th European Conf. Artificial Intelligence* (ECAI 04), IOS Press, 2004, pp. 318–322.
3. J. Sowa, *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesley, 1984.
4. F. Fürst and F. Trichet, "Reasoning on the Semantic Web Needs to Reason Both on Ontology-Based Assertions and on Ontologies Themselves," *Proc. Int'l Workshop Reasoning on the Web* (RoW 06), 2006; www.aifb.uni-karlsruhe.de/WBS/phi/RoW06.
5. A. Gomez-Perez, M. Fernandez-Lopez, and O. Corcho, *Ontological Engineering*, Springer, 2003.
6. F. Fürst and F. Trichet, "Axiom-Based Ontology Matching," to be published in *Expert Systems: J. Knowledge Eng.*, vol. 25, no. 2, 2008.
7. S.M. Falconer, N.F. Noy, and M. Storey, "Towards Understanding the Needs of Cognitive Support for Ontology Mapping," *Proc. Int'l Workshop Ontology Matching* (OM 06), CEUR-WS, vol. 225, 2006; http://om2006.ontologymatching.org.

Frédéric Fürst is an associate professor of computer science at the University of Picardie–Jules Verne. Contact him at frederic.furst@u-picardie.fr.

Francky Trichet is an associate professor of informatics and computer science at the University of Nantes and a member of the research institute Laboratoire d'Informatique de Nantes Atlantique. Contact him at francky.trichet@univ-nantes.fr.

SmartMatcher: Improving Automatic Matching Quality

Horst Kargl and Manuel Wimmer,
Vienna University of Technology

Information integration deals with the problem of building a general view on different kinds of data. Its long history in computer science is rooted in database engineering from the early 1980s, when autonomous databases started to federate.[1] More recently, the Semantic Web and its schema-based technologies for describing, storing, and exchanging data have intensified the need to automate integration tasks.

Researchers have proposed several automated matching approaches and tools over the years. In general, these approaches fall into one of three categories.[2] *Schema-based* approaches use only schema information as input for the matching process, *instance-based* approaches use only instances as input, and *hybrid* approaches use schema and instance information. The typical outputs are simple one-to-one alignments, based mostly on schema information such as element name and structure similarities. These alignments, however, can't handle schema heterogeneities, which therefore remain problems that must be resolved manually. Furthermore, current tools can't automatically evaluate the alignment quality at the instance level because their matching approaches aren't bound to a specific integration scenario, such as transformation, merge, synchronization, or search.

The main requirement for matching solutions is to produce complete and correct mappings between schemas. Three problems complicate meeting this requirement:

- *Different mapping-execution scenarios.* Current matching approaches are general—that is, they apply to different kinds of integration problems. Because each integration scenario entails different conditions and interpretation, this generality makes it hard to cover all aspects of each scenario. Furthermore, most approaches lack a binding to an execution environment, which the actual integration solution will need.
- *Schema heterogeneity.* Matching approaches produce alignments that express correspondences between elements belonging to different schemas. Most schemas share similar semantics but describe their semantics with different structures. Current one-to-one alignments can't handle schema heterogeneities, so users must interpret and refine the alignment results manually.
- *Unreliable matching results.* Matching results are suggestions and not wholly reliable. In reality, the results often include mistakes, such as wrong or missed alignments. Assertions about alignment quality require quality measures.[3] To calculate these quality measures, the user must give all correct alignments, which means the user first has to solve the integration problem manually.

To tackle these problems, we've developed the SmartMatching approach to extend existing matching approaches orthogonally with a self-tuning component and thereby to improve the quality of automatically produced alignments for the transformation scenario. SmartMatcher is a hybrid approach that uses a real-world example to develop instances of the schema to be integrated. The example supports automatic evaluation of matching tools and improvement of their output results. We've implemented a prototype for schemas defined in the Eclipse Modeling Framework (EMF) Ecore metalanguage (www.eclipse.org/modeling/emf).

The SmartMatching Approach

Figure 1 (see next page) presents an overview of the SmartMatcher architecture and its integration process. Its three core components are the Initial Matcher, the Mapping Engine, and the Fitness Function. The workflow is an eight-step process.

1. Develop example instances. In this step, the user develops instances of semantically equivalent elements for each of the schemas to be integrated. Figure 2 (see next page) shows the general idea: the user must first define a real-world example that uses most of the schema elements. Describing the same real-world example with both schemas generates instances of semantically equivalent schema elements; nonoverlapping schema elements are filtered out.

Concrete examples are core elements of improving the mapping quality and supporting the SmartMatching self-tuning mechanism. The cloud at the bottom of Figure 2 stands for a real-world example

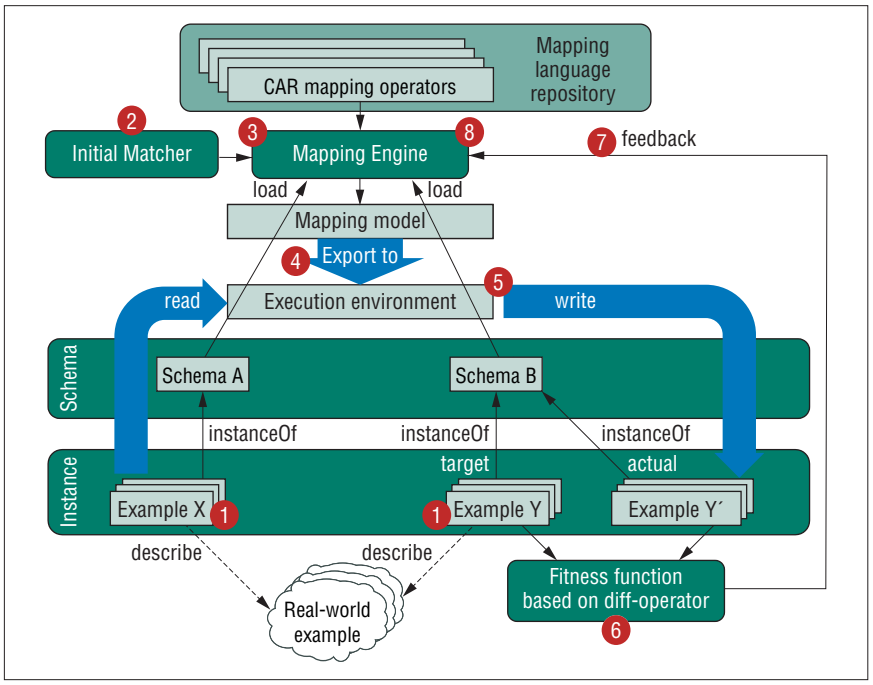

Figure 1. The SmartMatching approach architecture and process. SmartMatcher extends matching tools with self-tuning capabilities by providing self-evaluation (Fitness Function) and self-adaption (Mapping Engine) of mapping models as well as an iterative eight-step learning process.

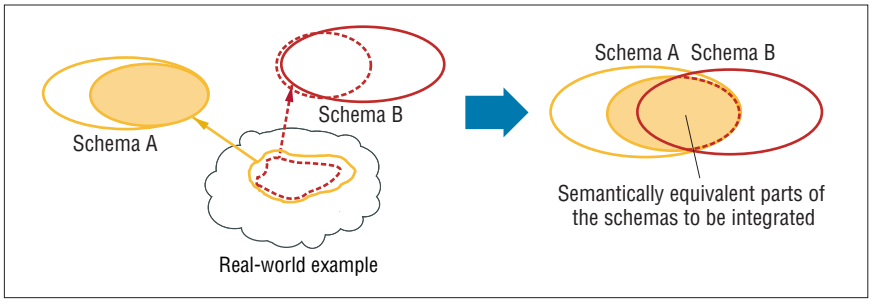

Figure 2. Building instances from a real-world example. First, the user defines a real-world example in natural language, which is then described in the schemas to be integrated.

of a specific domain. The user must describe this example in natural language and develop instances representing it for both schemas.

Other instance-based approaches also compare instance values to find similarities between schema elements. However, term ambiguities such as synonyms and homonyms keep the results of such a comparison from being trustworthy. Using the same real-world example and the same terms for the same concepts avoids these kinds of ambiguities.

The SmartMatcher uses the concrete examples to increase the completeness and correctness of found alignments. Comparing the actual instances generated by the transformation to the target instances developed by the user also supports evaluation of the mappings between two schemas. At the end of the SmartMatching process, the actual and target instances should be the same. If this is the fact, all automatically found mappings are correct.

2. Generate initial matching. We use existing matching tools to create basic alignments between similar schema concepts. We require the alignments to be expressed in the INRIA alignment format.[4] This lets us use all matching tools that deliver this format.

3. Interpret initial mappings. We can translate the alignments produced in step 2 to an initial mapping model based on the element types referenced by the alignments. The Initial Matcher increases Mapping Engine's performance by reducing the search space, compared to beginning with an empty mapping model.

4. Derive transformation. From the mapping model, a transformation is automatically generated, which transforms instances of one schema into instances of the other. In general, it's possible to generate different kinds of transformations from the mapping model to suit the schema languages. In our case, we generate transformations based on colored Petri nets[5] for transforming EMF-based models.

5. Transform instances. The execution environment is responsible for reading the instances conforming to one schema and transforming them into instances conforming to the other schema, according to the derived transformations.

6. Calculate differences. The Fitness Function compares the actual and target instances by means of their attribute values. Then it links and collects the differences in a diff model, which can be used to evaluate the quality of the mappings between Schema A and Schema B. Furthermore, in step 6 we have two termination conditions for the SmartMatching process. The first occurs when no further differences exist between the actual and target instances; in other words, the mapping is complete. The second termination condition occurs if the differences remain the same over several iterations; in this case, the process has reached a final point for a certain set of example instances.

7. Propagate differences. In this step, SmartMatcher propagates the differences calculated by the Fitness Function back to the Mapping Engine. More specifically, it propagates back missing and wrong values, expressed in the diff model of the actual and target instances.

8. Interpret differences and adjust mapping model. The Mapping Engine analyzes the propagated differences and adapts the current mappings between Schemas A and B by searching for and applying appropri-

ate mapping operators for missing or wrong mappings.

After step 8, a new iteration starts at step 4 until step 6. In step 6 the actual and the target models are compared again. If there are no more differences, the process is finished; otherwise the iteration continues until step 8, where a new iteration begins.

Compared with other automatic matching approaches, SmartMatcher needs more work in the preparation phase to establish the example instances. However, we hypothesize that building the instances for the real-world example costs less than manual evaluation and rework of alignments produced by other approaches. Furthermore, the real-world examples become available for reuse in other integration scenarios.

The SmartMatcher prototype we've implemented for EMF/Ecore implements a simple Initial Matcher component using the CAR (classes, attributes, and relationships) mapping language.[6] The prototype provides an import functionality for alignment models based on the INRIA alignment format. It also includes a Fitness Function implementation to compare the target model with the transformed actual model. The implementation can propagate differences between target and actual models to the Mapping Engine, which produces CAR mapping models that can be automatically converted to transformation definitions based on colored Petri nets.[5]

Using our first prototype implementation, we've evaluated our hypothesis that the SmartMatcher preparation is less work than the standard rework phase. First results have shown that the hypothesis holds true, especially in scenarios where the schemas to be integrated use different languages, naming conventions, or jargons. We also plan to conduct empirical experiments to evaluate our approach relative to completeness and correctness of the mappings as well as to verify the performance in terms of different mapping strategies' execution times. Furthermore, we're improving our prototype to support the development of appropriate test instances from real-world examples.

For additional information, see the SmartMatcher project homepage at http://big.tuwien.ac.at/projects/smartmatcher.

Acknowledgments

A longer version of this essay appeared in the *Proc. 2nd Int'l Conf. Complex, Intelligent, and Software-Intensive Systems* (CISIS 08), IEEE CS Press, 2008, pp. 879–885.

References

1. A.P. Sheth and J.A. Larson, "Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases," *ACM Computing Surveys*, vol. 22, no. 3, 1990, pp. 183–236.
2. E. Rahm and P. Bernstein, "A Survey of Approaches to Automatic Schema Matching," *VLDB J.*, vol. 10, no. 4, 2001, pp. 334–350.
3. G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1984.
4. J. Euzenat, "An API for Ontology Alignment," *Proc. Semantic Web Conf.* (ISWC 04), LNCS 3298, Springer, 2004, pp. 698–712.
5. T. Reiter, M. Wimmer, and H. Kargl, "Towards a Runtime Model Based on Colored Petri-nets for the Execution of Model Transformations," *Proc. 3rd Workshop Models and Aspects—Handling Crosscutting Concerns in MDSD*, *Forschungsberichte der Fakultät IV—Elektrotechnik und Informatik*, no. 6, 2007, pp. 19–23.
6. G. Kappel et al., "A Framework for Building Mapping Operators Resolving Structural Heterogeneities," *Information Systems and e-Business Technologies*, LNBIP 5, Springer, 2008, pp. 158–174.

Horst Kargl is a project assistant at the Vienna University of Technology's Institute for Software Technology and Interactive Systems. Contact him at kargl@big.tuwien.ac.at.

Manuel Wimmer is an assistant professor at the Vienna University of Technology's Institute for Software Technology and Interactive Systems. Contact him at wimmer@big.tuwien.ac.at.

Ontology Mapping for the Masses: A Tagging Approach

Colm Conroy, Declan O'Sullivan, David Lewis, and Rob Brennan, *Trinity College Dublin*

As ontologies become more commonplace, the need increases for tools to cope with their diversity and heterogeneity. A variety of techniques can automatically match a user's personal ontology to other domain models.[1] The research challenge lies in how to derive ontology mappings from the candidate matches. Fully automatic derivation of mappings isn't yet feasible,[2] and most state-of-the-art ontology-mapping tools rely on a classic side-by-side presentation of two ontologies' class hierarchies and some means for a user to express the mappings.[3] Moreover, most tool interfaces assume the user is an ontology engineer who performs the work during long mapping sessions.

We've developed an early prototype of an interface that makes ontology mapping as unintrusive and natural as possible. We want to engage casual Web users in ontology mapping by designing a process that doesn't require ontology-engineering experience and that, moreover, makes the benefits of mapping clear.

Mapping-Process Design

To make the mapping process less daunting, we deconstructed it so that it can occur over multiple sessions. This lets users see the impact of their decisions between sessions and correct or enhance their mappings over multiple sessions.

The mapping-process design has four main steps with rules for presenting mapping tasks to the user and a feedback loop to evaluate the user responses.

Step 1: When to present the mapping task. Calculating when to present a mapping task to the user is a key to making the process less intrusive. Because mapping generation occurs over multiple sessions, the mapping system must determine when to present each task. Example rules for this step include specifying regular time intervals, such as once every hour, or specifying "just-in-time" schedules, such as each time users want to submit a query but need to map their own ontology to another one for the query to work.

In each presentation case, the process should signal users that a mapping task is pending and ask if they wish to perform the task at this time.

Step 2: What mapping task to present. Deciding which mapping task to present is the second step. Several strategies are available. Priority-based rules are one example. Mapping tasks relevant to the user's current Web-browsing context is another.

Step 3: How to present the mapping task. Displaying the mapping informa-

Figure 1. The tagging interface. The concepts appear in a specific type of natural language, and users characterize their relation by either entering a new tag or choosing from existing tags.

tion in a way that's most natural to the user requires rules that can reflect user preferences. Example rules in this area include natural language versus graphical display and/or choices among various information filters.

Step 4: What mapping interaction to use. Given the mapping information available, what is the best design for the user interaction that generates mappings? One example is to ask questions that the user could answer with a simple yes or no. Another is to let the user "drag and drop" graphical connections. A third approach is to have users add a semantic tag representing the semantic relationship between the matching pairs.

Initial Process Implementation

We've been experimenting with implementations of this four-step design process over the past two years.

Our first experiment focused on the third and fourth steps.[4] To move away from classical approaches that typically assume a knowledge engineer, we adopted a natural language question-and-answer (Q&A) approach. This lets us introduce small discrete mapping sessions more easily. We also hypothesized that natural language would help a nontechnical user understand the information better than graphical structures.

Our initial implementation's main purpose was to test the usability of our prototype natural-language mapping tool and compare it to a current state-of-the-art graphic-based tool—specifically, we selected COMA++ for our experiment.[3] We split the user test group into three distinct groups: *ontology-aware* users had ontology work experience, *technology-aware* users had database or UML modeling experience but no ontology experience, and *nontechnical* users had basic computer experience but no database modeling or ontology experience.

On the positive side, results from this experiment suggested that ordinary users compared well with ontology-aware users in mapping effectiveness and efficiency. Using natural language seemed to help people understand the mapping information, and the Q&A approach helped in navigating through the mapping task.

On the negative side, ordinary users found the narrow range of mapping terminology to be limiting when answering questions. In addition, some users were unclear about the benefit of engaging in the mapping task.

To address the negative concerns, we focused the next phase of our research on how to be less restrictive in the way users could express mappings and how to overcome confusion about the rationale for undertaking the mapping task by clearly demonstrating benefits.

Tagging-Approach Design

Social networking sites like del.icio.us (http://del.icio.us) have become popular because people can use their own language to tag a link and so associate their own meaning with it. We decided to explore whether enabling users to map ontologies by tagging would make the mapping process easier and perhaps even lead to more expressive mappings.

In this approach, once a user chooses a tag, the system categorizes it according to top-level categories and their conceptual subcategories:

- *equivalent*—the same, subclass;
- *equivalent sometimes*—superclass, one of, union, intersection;
- *different*—different from, complement of, disjoint; and
- *corresponds/unknown*.

These categories align the user tags with the reasoning primitives typically used to express and execute mappings. The decision-making rules for assigning a matching pair to a top-level category are configurable. For example, a "majority rules" configuration would assign a matching pair to the "equivalent" category if the user submits a tag from each of "the same," "subclass," and "union" subcategories for the pair.

The fourth category accepts matching pairs that the system can't assign to one of the other three categories.

Tagging-Approach Implementation

The prototype currently uses the same default rules for each user. It generates the matches with the INRIA alignment API.[5]

Presentation: Browser extension. To make the mapping process as natural as possible, we used a Firefox browser extension to display the mapping information. The mapping question appears on a transparent interface over the Web page the user is currently browsing, as shown in Figure 1. The concepts appear in a specific type of natural language that represents their parents, siblings, and properties via bullet points and fixed statements, such as "A is a B" and "C is a type of A."

We modified the natural language used in our previous experiment to remove some confusing ontological terms (for example, "Thing"). We also limited the number of

properties shown to avoid cognitive overload. Future experiments might use different constructions.

Interaction: Tagging interface. The user has to tag the matching-pair relation with the tags they think represent the relationship. They can either type in a new tag or choose from an existing list that contains suggested tags. The top-level categories are equivalent, equivalent sometimes, and different. Users can enter multiple tags for the relationship if they wish.

Evaluate user response: Tag analyzer. After the user submits tags for a matching pair, a tag analyzer applies rules to the set of user-specified tags, categorizing the matching pair. The rules are configurable, but at present we set the same default rules for each user—for example, majority wins and a tie goes to the first specified category.

When a matching pair is categorized into "corresponds/unknown" via undefined tags, the system checks the tags to see if the matching pair is an object-property relationship; if so, the system will construct a new matching pair. In future experiments, we plan to categorize the unknown matching pairs through observation of other users and user-interaction patterns. Additionally, if a user isn't seeing any benefit from a mapping, we want to flag it as possibly incorrect.

To clarify the benefits of engaging in the mapping task, we're currently testing our prototype by offering users specific RSS items selected on the basis of mappings that users generate between their personal ontology and the domain ontology used by the RSS feeds. The user is alerted to new RSS feed information via a message in the browser extension—just like the message used to alert the user of pending mapping tasks. Users can ignore the message until later if they wish. The user test group is split into the same three group types as in the initial experiment.

This experiment is nearing completion. Some initial indications and feedback indicate that users have found the mapping prototype to be neither disruptive nor interfering. However, they would prefer that the alerts not display when they're busy. Most people think the mapping tasks are efficient, given their breakdown into small sessions. They liked the tagging approach because it was simple enough to use, although the quality of the generated tags has yet to be analyzed. Our next experiment will allow the user's browsing context to support adaptation of the mapping process. A wide-scale user trial over the Web is also planned.

Acknowledgments

A longer version of this essay appeared in the *Proc. 2nd Int'l Conf. Complex, Intelligent, and Software-Intensive Systems* (CISIS 08), IEEE CS Press, pp. 886–891. This work is part funded by the Irish Higher Education Authority's Nembes research project grant.

References

1. P. Shvaiko and J. Euzenat, "A Survey of Schema-Based Matching Approaches," *J. Data Semantics*, vol. 4, 2005, pp. 146–171.
2. N. Noy, "Semantic Integration: A Survey of Ontology-Based Approaches," *SIGMOD Record*, no. 4, 2004, pp. 65–70.
3. D. Aumüller et al., "Schema and Ontology Matching with COMA++," *Proc. 2005 ACM SIGMOD Int'l Conf. Management of Data*, ACM Press, 2005, pp. 906–908.
4. C. Conroy, D. O'Sullivan, and D. Lewis, "A Tagging Approach to Ontology Mapping," *Proc. 2nd Int'l Workshop Ontology Mapping* (ISWC 07); http://om2007.ontologymatching.org.
5. J. Euzenat et al., "Ontology Alignment with OLA," *Proc. 3rd Int'l Workshop Evaluation of Ontology-Based Tools*, CEUR-WS, 2004; http://ftp.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-128/EON2004_Proceedings.pdf.

Colm Conroy is a PhD student in the Department of Computer Science's Knowledge and Data Engineering Group, Trinity College Dublin. Contact him at coconroy@cs.tcd.ie.

Declan O'Sullivan is director of the Department of Computer Science's Knowledge and Data Engineering Group and a lecturer in the School of Computer Science and Statistics at Trinity College Dublin. Contact him at declan.osullivan@cs.tcd.ie.

David Lewis is a research lecturer at Trinity College Dublin. Contact him at dave.lewis@cs.tcd.ie.

Rob Brennan is a research fellow at Trinity College Dublin. Contact him at rob.brennan@cs.tcd.ie

Mappings for the Semantic Web

José Ángel Ramos-Gargantilla and Asunción Gómez-Pérez,
Universidad Politécnica de Madrid

Mappings usually relate two similar knowledge-aware resources. Mapping examples abound in thesauri, databases, and ontologies. Additionally, mapping systems can relate two different knowledge resources, such as databases and ontologies. All these mappings are operationally different and are sometimes named differently—for example, correspondences, semantic bridges, transformations, semantic relations, functions, conversions, and domain-method relations.

We've analyzed some of the existing mapping definitions and representations in the ontology world and its semantic neighborhood, and we propose a new definition and model to address the Semantic Web and its needs for format, access, and resource heterogeneity.

Knowledge-Representation Definitions

Drawing on the idea of mappings as a structured representation, Semantic Web research has focused mapping definitions on ontologies. For example, in 2002, Xiaomeng Su gave this definition:

> Given two ontologies A and B, mapping one ontology with another means that for each concept (node) in ontology A, we try to find a corresponding concept (node), which has the same or similar semantics in ontology B and vice versa.[1]

In Su's definition, the mapping elements are ontology concepts. Because mapping involves only two ontologies, the relation between elements is bidirectional and the semantics of the relation is of similarity or identity. Su's definition includes no idea of a conversion or transformation of elements.

In that same year, Alexandre Maedche and his colleagues proposed a definition that picked up on the transformation idea. They also extended the process vocabulary by introducing the term "semantic bridge" for mappings in which the transformation was not equivalent:

> An ontology mapping process is the set of activities required to transform instance

> of a source ontology into instances of a target ontology …. The mapping must define the two ontologies being mapped. Additionally, one may specify top-level semantic bridges which serve as entry points for the translation even if they are not mandatory. In this case the translation engine starts executing the Individual-Individual bridge.[2]

In 2003, Monica Crubézy and Mark Musen introduced yet another new dimension—namely, mapping between a domain and a problem-solving method (PSM) ontology:

> Our mapping ontology provides the basis for expressing the adaptation knowledge needed to configure a PSM for a certain application. In that sense, our mapping ontology extends the notion of domain-PSM bridges in the UPML [Unified Problem-Solving Method Description Language] framework by providing a structured and operational set of possible mapping axioms that bridge the ontologies of both components.[3]

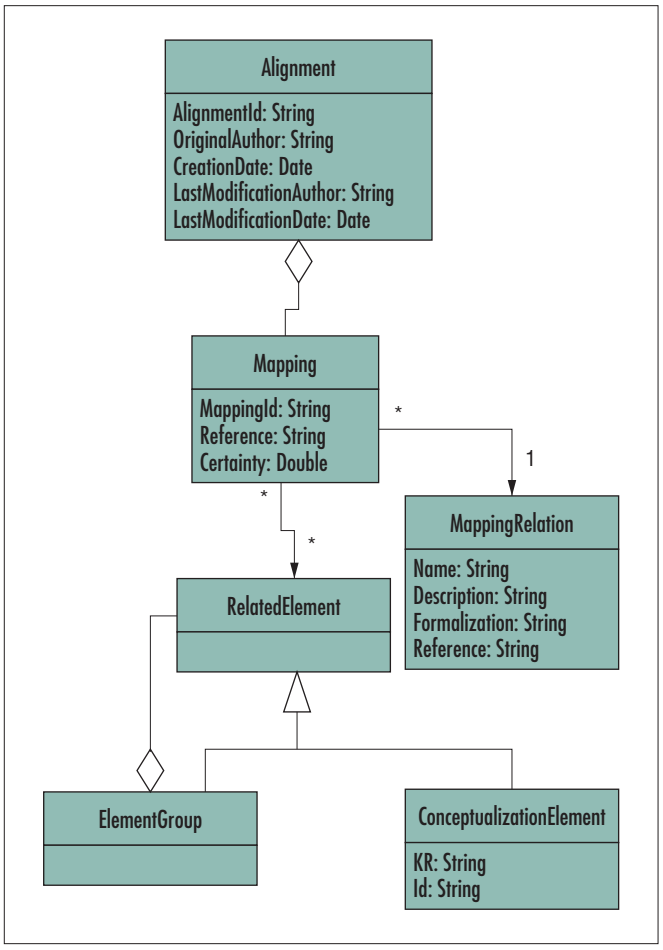

Figure 1. Mapping-model proposal. Mappings define relations between knowledge representations and their associated information (such as certainty, reference, and metadata).

This definition isn't classified into mappings or semantic bridges according to the complexity of functions. Mappings focus on configuring a PSM that will execute on concrete domain elements. The transformation idea is missing.

In 2004, a specification deliverable, led by Jerome Euzenat and Pavel Shvaiko for the EU's Knowledge Web project, provided a new definition of mapping between ontologies:

> A formal expression that states the semantic relation between two entities belonging to different ontologies. When this relation is oriented, this corresponds to a restriction of the usual mathematical meaning of mapping: a function (whose domain is a singleton).[4]

Again, mapping is defined here as an expression, without an explicit transformation objective. This definition upgrades the set of ontology components by extending Su's restricted mappings (only between concepts), and covers all complexity levels of expressions. Additionally, a new element appears—direction associated to the mapping when the relation is a function. This direction contradicts Su's bidirectional definition (because it covers only similarity and identity relations).

In their 2005 survey, Yannis Kalfoglou and Marco Schorlemmer defined ontology mapping as follows:

> A morphism, which usually will consist of a collection of functions assigning the symbols used in one vocabulary to the symbols of the other.[5]

They distinguished two mapping types: one oriented to correspondence between representation languages and the other oriented to correspondence between vocabularies. Such mappings have functions that assign the terms of one ontology to the terms of another. Therefore, their definition covers the mappings between PSM and domain ontologies, although it's restricted to only two ontologies.

Semantic Web Mappings

All these definitions between ontologies apply within the Semantic Web area. Although ontologies are the main knowledge representation of the Semantic Web, they aren't the only one. Integrated in the Semantic Web are systems and applications that work with other formats such as databases, natural language documents, annotated documents, Web pages, semantic networks, graphs, and navigation models. These knowledge-aware resources can be mapped with ontologies or between them.

Additionally, the Semantic Web includes systems that execute PSMs to obtain different results with different domain ontologies. So, Semantic Web mappings need to cover directional and not-predefined functions.

The Ontology Engineering Group at Universidad Politécnica de Madrid (UPM) has developed a mapping definition that covers the Semantic Web resources and functions:

> A mapping is a formal explicitation of a relation between elements, or a set of elements, of different knowledge resources (models and data).

In this definition, "explicitation" refers to a relation that's both explicit and formal, as in "machine-readable," and "element" refers to all components of a resource (concepts, nodes, columns of a table, value of an attribute in an instance, and so on). This definition doesn't limit the relation to a reciprocal function or declarative

transformations, as shown in the model in Figure 1. However, it supports mappings between all knowledge representation elements in any type of resource, without restriction to the number of elements or resources. Moreover, it encompasses all mappings that are part of Semantic Web processes, such as ontology alignment, heterogeneous-resource integration, and annotation.

Mapping Models

The literature offers several mapping models. For example, the Common Warehouse Model (CWM) represents mappings that are both generic and expressive,[6] but this model is also complex. It's composed of classes—Transformation, TransformationMap, ClassifierMap, FeatureMap, ClassifierFeatureMap, TypeMapping—and their properties and characteristics.

The RDF Transformation (RDFT) metaontology is based on the CWM. The RDFT specifies a small language for DTD (Document Type Definition) mappings of XML to RDF Schema and vice versa.[7] Its main class is Bridge, although it also includes Map, EventMap, Interface, Roles, Event2Event, DocumentMap, XMLBridge, VocabularyMap, and RDFBridge.

OWL defines equivalentClass and equivalentProperty as primitives, both of which can be considered mapping explicitations.[8]

C-OWL is a mapping-language proposal that can express relatively simple alignments between ontologies. The constructs in C-OWL are called bridge rules, and they can express a family of semantic relations between concepts/roles and individuals. C-OWL mappings provide eight semantic relations: equivalence, containment (contains and is contained in), overlap, and their negations.[9]

The SEKT (Semantically Enabled Knowledge Technologies) mapping language provides a set of constructs to express mappings between ontology classes, attributes, relations, and instances.[10] Several other languages express mappings, although we focus here on the language that is the most similar to our mapping concept—that is, INRIA's alignment format.[11]

Mapping Model Proposal

Starting from common elements of these models and taking into account that mapping could exist between elements of different types of resources, we designed a simple model for covering mappings and their uses in the Semantic Web. Figure 1 shows this model.

This model is independent of the knowledge resource; we can therefore use it to represent mappings between ontologies, between relational databases and ontologies, between some thesauri, and so on. Furthermore, mapping managers can define the relations they need because mapping relations are not limited. The model includes component metadata such as LastModificationDate and Reference, mainly for tracing information flow.

For making this representation usable, we present it as an XML Schema Definition.

UPM has a bilateral agreement with the Spanish National Geographic Institute (IGN) to integrate current heterogeneous databases using the definition and representation proposals presented here. IGN has four databases with geographic information in different scales. This information is classified into phenomena that have tremendously different granularity—for example, one catalog has 22 phenomena and another has 560. UPM and IGN have jointly developed an ontology of phenomena, called PhenomenOntology, and they are developing an automatic mapping discoverer between the ontology and the relational databases. Such mappings are represented using the model presented in Figure 1.

Additionally, the Ontology Engineering Group is working on extracting mappings of concept classification from textual semantic annotations. Such mappings could be used in ontology-learning or ontology-alignment applications, and we are representing them following our mapping model.

Acknowledgments

The Spanish National Project GeoBuddies (TSI2007-65677C02) and the bilateral collaboration UPM-IGN 2007-2008 supported the work reported here. A longer version of this essay appeared in the *Proc. 2nd Int'l Conf. Complex, Intelligent, and Software-Intensive Systems* (CISIS 08), IEEE CS Press, 2008, pp. 907–912.

References

1. X. Su, "A Text Categorization Perspective for Ontology Mapping," 2002; www.idi.ntnu.no/~xiaomeng/paper/Position.pdf.
2. A. Maedche et al., "MAFRA—A MApping FRAmework for Distributed Ontologies," *Proc. 13th Int'l Conf. Knowledge Eng. and Knowledge Management* (EKAW 02), LNCS 2473, Springer, 2002, pp. 235–250.
3. M. Crubézy and M.A. Musen, "Ontologies in Support of Problem Solving," *Handbook on Ontologies*, S. Staab and R. Studer, eds., Springer, 2003, pp. 321–341.
4. P. Bouquet et al., *D2.2.1 Specification of a Common Framework for Characterizing Alignment*, Knowledge Web (FP6-507482), 2004; http://knowledgeweb.semanticweb.org/semanticportal/deliverables/D2.2.1v1.pdf.
5. Y. Kalfoglou and M. Schorlemmer, "Ontology Mapping: The State of the Art," *Proc. Dagstuhl Seminar Semantic Interoperability and Integration*, 2005.
6. *CWM: Common Warehouse Model Specification*, v. 1.1, Object Management Group, 2003.
7. B. Omelayenko, "RDFT: A Mapping Meta-Ontology for Business Integration," *Knowledge Transformation for the Semantic Web*, B. Omelayenko and M. Klein, eds., IOS Press, 2003, pp. 137–153.
8. M. Uschold, "Achieving Semantic Interoperability using RDF and OWL," v. 4, Knowledge Web Deliverable 2.2.6, 2005; http://knowledgeweb.semanticweb.org/semanticportal/deliverables/D2.2.6.pdf.
9. P. Bouquet et al., "C-OWL: Contextualizing Ontologies," *Proc. 2nd Int'l Semantic Web Conf.*, LNCS 2870, Springer, 2003, pp. 164–179.
10. J. de Bruijn, D. Foxvog, and K. Zimmerman, *Ontology Mediation Patterns Library*, Knowledge Web Deliverable D4.3.1, Semantically Enabled Knowledge Technologies, 2004.
11. J. Euzenat and P. Shvaiko, *Ontology Matching*, Springer, 2007.

José Ángel Ramos-Gargantilla is a member of the Ontological Engineering Group at the Universidad Politécnica de Madrid and a PhD student in the Artificial Intelligence Department. Contact him at jarg@fi.upm.es.

Asunción Gómez-Pérez is the founder and director of the Ontological Engineering Group at the Universidad Politécnica de Madrid and a full professor in the Artificial Intelligence Department at UPM's Computer Science School. Contact her at asun@fi.upm.es.

SPARQL Extensions for Processing Alignments

Jérôme Euzenat, *INRIA and Laboratoire d'Informatique de Grenoble*
Axel Polleres, *National University of Ireland, Galway*
François Scharffe, *University of Innsbruck*

Heterogeneity between ontologies is often handled by establishing correspondences between the ontologies' entities and transforming data according to these correspondences, whether for integrating heterogeneous data sources or exchanging messages between services. Relations between aligned entities can be very complex, so we developed an alignment language for expressing such complexities.[1] The language is independent of concrete knowledge-representation and processing languages, but transforming concrete data requires processing the correspondences expressed in the alignment language. In particular, it requires translating the source ontology's data instances to instances of the target ontology.

We expect this scenario to become more common as an increasing number of ontologies and data are developed and published on the Web using the Resource Description Framework (RDF). A query language is a natural choice for translating data because it allows both data extraction and data transformation. Hence, while RDF, RDF Schema, and the Web Ontology Language (OWL) are the standards for describing data and ontologies on the Web, the Simple Protocol and RDF Query Language (SPARQL) seems the natural candidate for expressing and processing ontological correspondences.[2] However, SPARQL in its current version isn't yet powerful enough to cover the full expressivity of the alignment language we developed. We therefore propose combining two recent SPARQL extensions to handle complex alignments:

- SPARQL++ provides aggregates, value-generating built-ins, and (possibly recursive) processing of mappings expressed in SPARQL,[3] and
- PSPARQL provides queries on path expressions by allowing regular expression patterns.[4]

We illustrate our proposal with a data-translation problem between two commonly used ontologies: friend-of-a-friend (FOAF, http://xmlns.com/foaf/0.1) and vCard (www.w3.org/2006/vcard/ns). Both vocabularies describe information about persons and organizations, and both are used extensively on the Web. They cover complementary as well as overlapping aspects of personal information.

Alignment Representation

The *alignment format* is an extensible format for expressing alignments in XML/RDF.[5] It supports interchange between alignments created using ontology-matching algorithms and native representation of simple correspondences between ontological entities. The format is associated with an Alignment API,[6] which is organized around a small set of constructs that let users describe alignments through sets of correspondences, together with related metadata such as the alignment's purpose or the way it was built. Each set of correspondences gives a description of the alignment entities. Figure 1 shows a sample correspondence, expressing the equivalence between a vCard and a FOAF person.

The expressive alignment language we developed extends the alignment format so that it can represent more elaborate correspondences.[1] In particular, it offers

- operators to relate an entity in one ontology to a combination of entities in the other,
- conditions to restrict an entity's scope, and
- transformations for property values such as aggregations, functions, and data-type conversions.

The language provides a high-level description of ontology alignments and a convenient exchange format for matching algorithms, GUIs, and mediation languages.

Grounding

Ontology mediation is a complex mediation process involving two main phases.[7]

First, the alignment is constructed at design time. Typically, ontology engineers use matching algorithms to automatically discover correspondences between ontologies. Graphical mapping interfaces assist the process of refining these correspondences, which eventually involve correspondence patterns.[8] An expressive exchange format must carry the precise meaning of the correspondences.

Second, at runtime, the previously built alignments are executed in a mediation task using a specific target formalism. *Grounding* is the term we use for transforming the alignment expressed in an alignment-representation formalism—such as our expressive alignment language—into the concrete language or algorithm executable on the particular knowledge representation.

When translating instance data in RDF, SPARQL has advantages compared with upcoming rules language standards such as the Rule Interchange Format. SPARQL is declarative and already widely used for querying RDF Web data. This makes SPARQL-based data translation a more natural tool for Semantic Web users than rule-based languages or XML-based extraction techniques at the moment.

We can illustrate the process of grounding to concrete SPARQL expressions for data translations by using the example FOAF-vCard correspondences in Figure 1.

Data Translation Using SPARQL

SPARQL is the W3C recommendation for querying RDF.[2] Typically, SPARQL queries are used to select bindings of RDF terms to variables from a set of source RDF graphs (also called the dataset) according to a graph pattern. In a slightly simplified view, such a query follows the general structure:

```
CONSTRUCT { result pattern }
FROM dataset
WHERE { graph pattern }
```

Answers to a SPARQL query Q rely on computing the set of possible homomorphisms from Q's basic graph pattern(s) into the RDF graph representing the knowledge base (that is, the dataset). The resulting variable bindings for instantiating the pattern in the WHERE part are then used to construct a new graph by instantiating the result pattern. So, if we want to reuse instance data described in one ontology when our application has been designed for another one, we can use such SPARQL CONSTRUCT queries as a translation mechanism. This is a natural mechanism for writing mapping rules between RDF vocabularies. For instance, the query in Figure 2(a) illustrates a CONSTRUCT query translating a `foaf:Person` into a `vc:VCard`. However, this query only covers the simple concept correspondence. We must complete this correspondence to additionally translate—possibly recursively—a person's properties, such as name, address, or tele-

```
<Cell>
  <entity1 rdf:resource="&foaf;Person"/>
  <entity2 rdf:resource="&vc;VCard"/>
  <measure  rdf:datatype="&xsd;float">1.0
       </measure>
  <relation>equivalence</relation>
</Cell>
```

Figure 1. A sample correspondence in the Alignment format.

phone number. CONSTRUCT queries can likewise be used for these subcorrespondences. For example, Figure 2(b) shows how we can extend the Figure 2(a) query to also map names in vCard addresses to FOAF. Figure 2(c) shows a CONSTRUCT statement that models a more complex submapping of the correspondence in Figure 1.

We can then execute these queries on a set of instance data represented in RDF to yield the transformed ontology instances in the target ontology. However, in practical use cases, it turns out that the available constructs in SPARQL still aren't sufficient for an expressive mapping language as required by complex applications.

SPARQL Extensions for Accurate Translation

Three features that SPARQL currently lacks would be particularly useful for processing alignments—namely, aggregate computation, individual generation, and path expressions.

Aggregates. Definition of a Project (DOAP, http://trac.usefulinc.com/doap) is an open source project to create an XML/RDF vocabulary to describe software projects. The DOAP vocabulary contains a revision property—that is, version numbers of released project versions. With an aggregate function MAX, you could map DOAP information into the RDF Open Source Software Vocabulary (http://xam.de/ns/os), which provides a latest-release property, as Figure 3a shows. Other aggregates, such as count, average, or sum, might be needed for complex and complete mappings.

Individual generation. Completing the mapping between vCard and FOAF, if we try mapping from vc:workTel to foaf:phone, we observe that the former is a data-type property and the latter an object property in OWL/RDF. Basically, a mapping needs a conversion function, generating a new

```
CONSTRUCT { ?x rdf:type foaf:Person }
WHERE { ?x rdf:type vc:VCard }
(a)

CONSTRUCT { ?X foaf:name ?FN . }
WHERE { ?X vc:FN ?FN .
  FILTER isLiteral(?FN) }
(b)

CONSTRUCT { ?X rdf:type foaf:Person.
  ?X foaf:based_near "Grenoble"^^xsd:string. }
WHERE { ?X rdf:type v:VCard .
      OPTIONAL {?X v:workTel ?PH.}
  OPTIONAL {?X v:workAdr [v:locality ?L]}
  FILTER ( startsWith(?PH, "+33476")
      OR ?L = "Grenoble") }
(c)
```

Figure 2. SPARQL data-translation examples: (a) simple mapping from vCard to the FOAF person concept, (b) simple mapping of that concept's related properties, and (c) combined mapping including combination of properties (Examples omit FROM clauses.)

URI from a literal. Figure 3b shows such a mapping.

SPARQL doesn't allow such value generations at the moment, but they are defined and implemented in a recent extension called SPARQL++.[3] SPARQL++ also provides aggregates. Therefore, we consider SPARQL++ to be a valid basis for such a mapping language, but it doesn't yet address all the issues in complex relations. For example, RDF's blank nodes, which correspond to existential variables in CONSTRUCT queries, involve additional complications, which are discussed in more detail elsewhere.[3]

Paths. Another missing part for expressing complex mappings is path expressions over RDF graph patterns, which aren't expressible in SPARQL. This is fairly surprising for a language that claims to be a graph query language. Here, PSPARQL—another recent extension—SPARQL allows to replace the basic graph patterns—that is, RDF graphs with variables—by graphs with variables and regular path expressions in place of predicates.[4] We can view path expressions as complementary to aggregations: where aggregations join pieces together, path expressions extract them individually.

The example PSPARQL query in Figure 3c exhibits two path expressions in the WHERE clause using the indefinite composition (+) operator, extending SPARQL's

```
CONSTRUCT { ?P os:latestRelease
   MAX(?V : ?P doap:release ?R.
            ?R doap:revision ?V) }
WHERE { ?P rdf:type doap:Project . }
(a)

CONSTRUCT {?X a foaf:phone
   xsd:anyURI(
     fn:concat("tel:",fn:encode-for-uri(?T))).}
WHERE { ?X vc:tel ?T . }
(b)

CONSTRUCT { ?X rdf:type ex:PotentialSalesPerson }
WHERE {?X foaf:knows+ [ foaf:worksFor
       [ vc:adr [ vc:city "Innsbruck"]]] }
(c)
```

Figure 3. Example SPARQL extensions for accurate translation: (a) using aggregates to map DOAP to the Open Source Software Vocabulary, (b) using value-generation in CONSTRUCTs to map vCard telephone numbers (represented as strings) to FOAF telephone numbers (represented as URIs), and (c) a complex mapping using regular path expressions. (Examples omit FROM clauses.)

existing simple bracketed path expressions. This query maps to the class of potential salespersons for Innsbruck, by picking persons that indirectly know someone working in for a company based in Innsbruck.

We demonstrated that a query language is an adequate means for transforming RDF data according to some ontology alignment. However, the current SPARQL specification isn't yet powerful enough for supporting this task with the complex mappings that are necessary for describing alignments between ontologies on the instance level. The combination of SPARQL extensions—SPARQL++ and PSPARQL—can serve as a basis to ground expressive ontology alignments in concrete executable mappings between data RDF graphs adhering to different, overlapping ontologies.

To implement a complete alignment framework, we propose two things: first, an implementation of a SPARQL data transformation engine integrating PSPARQL and SPARQL++ and, second, a grounding of an abstract, expressive alignment language to this new PSPARQL++. We are currently working

on reconciling the different proposed extensions toward a common prototype.

Acknowledgments

The authors thank Faisal Alkhateeb, the main designer of PSPARQL. Alex Polleres' work is supported by the Science Foundation Ireland under the Lion project (SFI/02/CE1/I131) and by the European Commission under project inContext (FP6 IST-034718). A longer version of this essay appeared in the 2008 Proc. 2nd Int'l Conf. on Complex, Intelligent, and Software-Intensive Systems (CISIS 08), IEEE CS Press, pp.913–917.

References

1. J. Euzenat, F. Scharffe, and A. Zimmermann, *Expressive Alignment Language and Implementation*, tech. report D2.2.10, Knowledge Web Network of Excellence (EU-IST-2004-507482), 2007.
2. E. Prud'hommeaux and A. Seaborne, eds., *SPARQL Query Language for RDF*, W3C Recommendation, 2008.
3. A. Polleres, F. Scharffe, and R. Schindlauer, "SPARQL++ for Mapping between RDF Vocabularies," *Proc. 6th Int'l Conf. Ontologies, DataBases, and Applications of Semantics* (ODBASE 07), LNCS 4803, Springer, 2007, pp. 878–896.
4. F. Alkhateeb, J.-F. Baget, and J. Euzenat, *Extending SPARQL with Regular Expression Patterns*, Tech. report 6191, Institut National de Recherche en Informatique et en Automatique (INRIA), 2007.
5. J. Euzenat and P. Shvaiko, *Ontology Matching*, Springer, 2007.
6. J. Euzenat, "An API for Ontology Alignment," *Proc. 3rd Int'l Semantic Web Conf.*, Springer, 2004, pp. 698–712.
7. F. Scharffe et al., *Analysis of Knowledge Transformation and Merging Techniques and Implementations*, tech. Report D2.2.7, Knowledge Web Network of Excellence (EU-IST-2004-507482), 2007.
8. F. Scharffe et al., "Correspondence Patterns for Ontology Mediation," Ontology Matching Workshop, CEUR-WS Vol. 304, 2007, pp. 296-300.

Jérôme Euzenat is a senior research scientist at INRIA Grenoble Rhone-Alpes and Laboratoire d'Informatique de Grenoble. Contact him at jerome.euzenat@inrialpes.fr.

Axel Polleres is a lecturer and project leader at the Digital Enterprise Research Institute at the National Unviersity of Ireland, Galway. Contact him at axel.polleres@deri.org.

François Scharffe is a researcher at the Semantic Technology Institute and PhD candidate at the University of Innsbruck. Contact him at francois.scharffe@uibk.ac.at.

Ontology Matching: Status and Challenges

Konstantinos Kotis,
University of the Aegean
Monika Lanzenberger,
Vienna University of Technology

Ontology matching is a significant Semantic Web research topic and a critical operation in many domains—for example, heterogeneous systems interoperability, data warehouse integration, e-commerce query mediation, and semantic-services discovery. The matching operation takes two (sometimes more) ontologies as input, each consisting of a set of discrete entities and axioms, and outputs the relationships between the entities, such as equivalence or subsumption.

Researchers have proposed many solutions to the ontology-matching problem.[1,2] Although it's a different problem from schema matching, the techniques developed for each of them has benefitted the other. Most research has focused on specific criteria for evaluating and distinguishing between matching approaches according to

- *algorithm input*—for example, entity labels, internal structures, attribute types, and relationships with other entities;
- *matching-process characteristics*—for example, the approximate or exact nature of its computation or the way it interprets input data (syntactic, external, or semantic); and
- *algorithm output*—for example, a one-to-one matching or a one-to-many or many-to-many correspondence.

Other significant distinctions in the output results include the confidence and probability percentages of the mapping results and the kinds of relations provided (equivalence, subsumption, incompatibility, and so on).

Human involvement during the ontology-matching process is usually a trade-off with the results' precision and recall percentages. Fully automated tools are still looking for higher accuracy. International contests such as the Ontology Alignment Evaluation Initiative (http://oaei.ontologymatching.org) provide a forum and benchmarks for this task.[3] Still, both automated and semiautomated tools need to improve their performance. For instance, most of them can't handle large real-domain ontologies such as those in medicine and biology, although the research community has developed more and more realistic testbeds to evaluate tool solutions to the scalability problem.

Beyond ontology-matching methods, tools, and evaluation initiatives, recent efforts have focused on design frameworks for ontology-matching tools, such as AUTOMS-F.[4] Such frameworks let developers use APIs to not only develop ontology-matching methods but also, and more important, synthesize these methods into robust tools for producing more accurate mappings. Beyond this, existing methods need more work to improve their matching quality.

Dilemmas and Critical Questions

The research community's efforts to provide diverse solutions to the matching problem by developing a variety of tools haven't yet generated a dominant set of methods that can serve as a benchmark for designing other matching tools. This might reflect the variety of domain-specific user or application needs. In fact, we conjecture that the community couldn't nominate "the best tool" because so many critical questions arise within a specific problem-solving context.

For example, should the tool be fully or semiautomated? To answer this question satisfactorily requires knowing the extent of human involvement, if any, and how this influences the accuracy of mapping results. How much time must users spend validating mapping results? Can we ensure that mapping results are valid without users' involvement? Ultimately, the questions resolve to what is most critical for their application: investing in human involvement to validate resources or automating the ontology-matching tool? If user interaction is essential, you must provide the means to analyze the matching results and understand the source ontologies' characteristics.

Another important question is whether the tool should provide very high precision and indifferent recall or vice versa. What balance between these parameters does the user's application require? What are the trade-offs? Is there an optimal trade-off, and can users tune the methods to achieve it?

Performance involves another set of questions. Does the application call for a tool that supports high computational complexity and rather slow execution time, or are rapid results more important? What percentage of precision and recall are users willing to sacrifice to speed up the ontol-

ogy-matching process and, consequently, their application? In the end, what is more critical for the application; to achieve the tool's highest precision and recall or to obtain the mappings as fast as possible?

All these questions require answers within the context of specific application and user needs.

Challenges

Despite many years' progress toward solving ontology-matching problems, the research community still reports open issues that impose challenges and underline new directions for the future work.

Scalability. Most implemented and evaluated ontology-matching tools suffer performance problems in handling large ontologies. Real problems in specific application contexts require scalable solutions as a first priority. Future ontology-matching tools should provide this capability.

Tuning speed, automation, and accuracy. Tools currently emphasize maximizing specific performance parameters such as speed, automation, or accuracy. Most commonly, a tool will maximize one parameter's performance while neglecting—or even impeding—the performance of the others. Future research should support fine tuning all parameters.

Background knowledge. The ontology-matching process makes extensive use of domain-related background knowledge. Recent experiments to improve tool recall results have tried matching one ontology to another while using a third ontology (or more) that's larger and more detailed ontology from the same domain as background knowledge.[3] But this process doesn't seem to scale well.

The challenge here is to adopt an approach that doesn't sacrifice overall tool performance.

Ontology-matching frameworks. Some design frameworks for ontology-matching tools exist,[4] but their performance needs further investigation. Software developers need support not only for devising ontology-matching methods but also for synthesizing them into new tools that produce more accurate mappings. Scalability, speed, and compatibility between input ontology types also require further investigation to deliver a model framework that the research community could use to devise specific ontology-matching tools for specific user or application preferences.

Ontology-matching visualization. Humans must perform and decide several issues in ontology matching to ensure the quality, appropriateness, and relevance of the matching results. Interpreting an entity of one ontology in the context of the knowledge of another ontology is a cognitively difficult task that requires understanding the semantic relations among entities of different ontologies.[5] Visualizing ontology-matching results could support user understanding.

The challenges in this domain grow with every advance in IT and the emerging economic infrastructure it supports. Ontology-matching results can manifest the same difficulties as the source ontologies: they can be large, complex, and heterogeneous. Yet so long as the information the ontologies are organizing continues to expand and different ontologies turn up for the same information, both academic and industry researchers will proceed to address these challenges. ■

Further Reading on Ontology Alignment and Matching

The essays in this issue's Trends & Controversies appeared originally in longer versions as part of the First International Workshop on Ontology Alignment and Visualization (OnAV 2008). Space limitations restricted including all the workshop papers here, but they are available in the *Proceedings of the Second International Conference on Complex, Intelligent, and Software-Intensive Systems* (CISIS 08), IEEE CS Press, 2008.

Other conferences and workshops include the International Semantic Web Conference (ISWC), which is in its eighth year (http://iswc2008.semanticweb.org). ISWC is an international forum for Semantic Web research. Its conferences are organized and managed by the Semantic Web Science Association (SWSA, www.iswsa.org).

Since 2004, the Ontology Alignment Evaluation Initiative (OAEI, oaei.ontologymatching.org) has organized competitions aimed at evaluating ontology-matching technologies and establishing performance benchmarks. The OAEI 2008 campaign is associated with the ISWC Ontology Matching Workshop, which is in its third year (http://om2008.ontologymatching.org).

A treasure-house for papers and other scholarly information in this field is available at www.ontologymatching.org.

Acknowledgments

We acknowledge the research work on ontology alignment conducted by members of the AI-Lab, University of the Aegean. This work is a unique source of experience for realizing the work described here.

A longer version of this essay appeared in the *Proc. 2nd Int'l Conf. Complex, Intelligent, and Software-Intensive Systems* (CISIS 08), IEEE CS Press, 2008, pp. 924–927.

References

1. P. Shvaiko and J. Euzenat, "A Survey of Schema-Based Matching Approaches," *J. Data Semantics IV*, LNCS 3730, Springer, 2005, pp. 146–171.
2. Y. Kalfoglou and M. Schorlemmer: "Ontology Mapping: The State of the Art," *Knowledge Eng. Rev.*, vol. 18, no. 1, 2003, pp. 1–31.
3. C. Caracciolo et al., "First Results of the Ontology Alignment Evaluation Initiative 2008," OAEI, 2008; www.dit.unitn.it/~p2p/OM-2008/oaei08_paper0.pdf.
4. A. Valarakos et al., "AUTOMS-F: A Java Framework for Synthesizing Ontology Mapping Methods," *Proc. Int'l Conf. Knowledge Management* (I-KNOW 07), 2007; www.icsd.aegean.gr/ai-database/papers/AutomsF.pdf.
5. S.M. Falconer, N.F. Noy, and M.-A. Storey, "Ontology Mapping—a User Survey," *Proc. 2nd Int'l Workshop Ontology Matching* (OM 07), CEUR-WS, 2007, vol. 304, 2007; www.dit.unitn.it/~p2p/OM-2007/5-446ontology_mapping_survey.pdf.

Konstantinos Kotis is a research scientist and member of the Department of Communication and Information Systems' AI Lab at the University of the Aegean. Contact him at kotis@aegean.gr.

Monika Lanzenberger is a member of the academic faculty of the Institute of Software Technology and Interactive Systems at the Vienna University of Technology. Contact her at lanzenberger@ifs.tuwien.ac.at.