ISWC-2003
Workshop Program

Semantic Integration Workshop (SI-2003)

AnHai Doan
Alon Halevy
Natasha Noy

Second International Semantic Web Conference

October 20, 2003
Sanibel Island, Florida, USA

In numerous distributed environments, including today's World-Wide Web, organizational intranets, and the emerging Semantic Web, the applications will inevitably use the information described by multiple ontologies and schemas. Interoperability among applications depends critically on the ability to map between them. Today, matching between ontologies and schemas is still largely done by hand, in a labor-intensive and error-prone process. As a consequence, semantic integration issues have now become a key bottleneck in the deployment of a wide variety of information management applications.

The high cost of this bottleneck has motivated numerous research activities on methods for describing mappings, manipulating them, and generating them semi-automatically. This research has spanned several communities (Databases, AI, WWW), but unfortunately, there has been little cross fertilization between the communities considering the problem.

This workshop examines semantic integration issues, with an emphasis on schema and ontology matching, ontology integration, and object matching and fusion. It brings together researchers from different communities to examine cutting-edge approaches to semantic integration, to consider how different communities can leverage each other's strengths, and to discuss additional challenges brought by new application contexts.

Workshop topics:

Workshop topics include, but are not limited to, the following:

- matching schemas and ontologies
- languages to express semantic mappings
- tools for user-driven mapping between ontologies and schemas
- reasoning with semantic mappings
- mapping negotiation, semantic negotiation
- using semantic mappings for query answering, data transformation, and other applications
- object matching and fusion
- maintenance of semantic mappings
- discovery and usage of approximate mappings
- evaluation of matching techniques
- merging schemas and ontologies
- model management
- specialized matching techniques for specific application contexts

Table of Contents

Research papers

Demo descriptions

Position statements

Workshop schedule
(tentative)

8:00-9:15	**Breakfast, poster/demo setup**
9:15-9:20	Introduction (organizers)
9:20-10:20	**Invited talk**: Phillip Bernstein "Generic Model Management: A Database Infrastructure for Schema Manipulation"
10:20-11:05	**Paper Presentation Session**, chair AnHai Doan G. Marquet, C. Golbreich, A. Burgun “From an ontology-based search engine towards a mediator for medical and biological information integration” M.Missikoff, F.Schiappelli , F.Taglino “A Controlled Language for Semantic Annotation and Interoperability in e-Business Applications” J. Akahani, K. Hiramatsu, and T. Satoh ""Approximate Query Reformulation for Integration"""
11:05-11:30	**Coffee break, poster/demo setup**
11:30-12:00	**Paper Presentation Session**, chair Natasha Noy M. Schorlemmer Y.Kalfoglou "On Semantic Interoperability and the Flow of Information " C. Menzel, P. Hayes "SCL: A Logic Standard for Semantic Integration"
12:00-1:00	**Panel discussion** "What are they smoking? Controversial issues in semantic integration" Panelists: Alon Halevy (chair), Pat Hayes, Len Seligman, Chris Welty
1:00-2:00	**Lunch, poster/demo setup**
2:00-3:00	**Invited talk**: Eduard Hovy "Building Large Ontologies"
3:00-4:00	**Poster/demo session**, coffee break
4:00-5:15	**Panel discussion** "Let’s get down to business: Mapping definition and discovery" Panelists: Michael Grüninger (moderator), Phil Bernstein, Jérôme Euzenat, Fausto Giunchiglia, Li Xu Panelists will also give overview of their papers in the workshop proceedings: M. Grüninger and J. Kopena "Semantic integration through invariants" F. Giunchiglia and P. Shvaiko "Semantic matching" J. Euzenat, P. Valtchev "An integrative proximity measure for ontology alignment" Li Xu and D. Embley "Using Domain Ontologies to Discover Direct and Indirect Matches for Schema Elements"
5:15-5:20	**Break**
5:20-6:00	**Panel and general discussion** "Where are we going? Research directions" Panelists: Mike Uschold (moderator), Alon Halevy, Eduard Hovy

Research papers

Approximate Query Reformulation for Ontology Integration

Jun-ichi Akahani, Kaoru Hiramatsu, and Tetsuji Satoh
NTT Communication Science Laboratories, NTT Corporation
2-4 Hikaridai, Seika-cho, Soraku-gun, Kyoto 619-0237, Japan
{akahani, hiramatu, satoh}@cslab.kecl.ntt.co.jp

Abstract

This paper proposes an approximate query reformulation framework for integrating heterogeneous ontologies. In order to achieve semantic interoperability in the Semantic Web, multiple ontologies have to be integrated. Ontology integration requires approximation mechanisms, since often no perfectly corresponding ontologies exist. However, most previous research efforts on ontology integration have not provided clear semantics for approximation. We have therefore proposed a framework for approximate query reformulation. In this framework, a query represented in one ontology is reformulated approximately into a query represented in another ontology based on an ontology mapping specification. In this paper, we focus on a fragment of OWL DL and provide a reformulation method for value restrictions and negation.

1 Introduction

Ontologies play a central role in the Semantic Web. However, the decentralized nature of the Web makes it difficult to construct or standardize a single ontology. Regional information is one reason for this, because ontologies vary from region to region due to the cultural differences. Ontologies also vary over time. Different people may update or customize an ontology independently. In such cases, one may query based on an updated ontology for the original ontology, or vice versa. We thus have to integrate heterogeneous ontologies both in temporal and spatial dimensions.

When integrating ontologies, it is rare to find those that correspond exactly; for example, there may be no corresponding class for Cajun restaurants in a Japanese ontology for restaurants. In such a case, one may use an approximation mechanism to replace "Cajun" with the "American" restaurant class in the Japanese ontology. However, most previous research efforts on ontology integration have not provided clear semantics for approximation.

We have therefore proposed a basic framework for approximate query reformulation [2]. In this framework, a query represented in one ontology is reformulated approximately into a query represented in another ontology based on an ontology mapping specification which is also described as an ontology. In order to characterize *closer* reformulation, the framework introduces two types of reformulation: *minimally-containing* reformulation and *maximally-contained* reformulation. However, this paper focuses on a simple ontology language and one-to-one subsumption mapping between ontologies.

Ontology description languages such as OWL [6] have much expressive power by providing constructs for value restrictions (`allValuesFrom` and `someValuesFrom`) and negation (`disjointWith` and `complementOf`). We therefore extend the approximate query reformulation framework for a fragment of OWL DL [6]. We focus on minimally-containing reformulation because they require non-standard inferences in Description Logics. Based on the formal framework, we provide a reformulation method for value restrictions and negation.

In the following sections, we first present an approximate query reformulation framework. We then provide a reformulation method for value restrictions and negation. Finally, we relate our framework to previous efforts in the field and present our conclusions.

2 Approximate Query Reformulation

In the approximate query reformulation framework [2], a query represented in one ontology is reformulated approximately into a query represented in another ontology based on an ontology mapping specification, as shown in Figure 1. This section and the next present a formal framework for approximate query reformulation. Throughout these sections, we use the simple example in the figure to illustrate our framework. More complex examples will be shown in the latter sections.

2.1 Queries in Ontologies

In this paper, we focus on a fragment of OWL DL [6] to describe ontologies. We use a Description Logic syntax for the sake of simplicity. Our ontology description language provides class description constructs for value restrictions ($\forall P.C$ for `allValuesFrom` and $\exists P.C$ for `someValuesFrom`) and negation ($\neg C$ for `complementOf`). We distinguish classes and properties in each ontology as follows.

Definition 1 (Class Description) *Given a set $\mathcal{C}_0^i$ of atomic classes and a set $\mathcal{P}^i$ of properties in an ontology O^i, a set $\mathcal{C}^i$ of classes in the ontology O^i consists of the following class descriptions:*

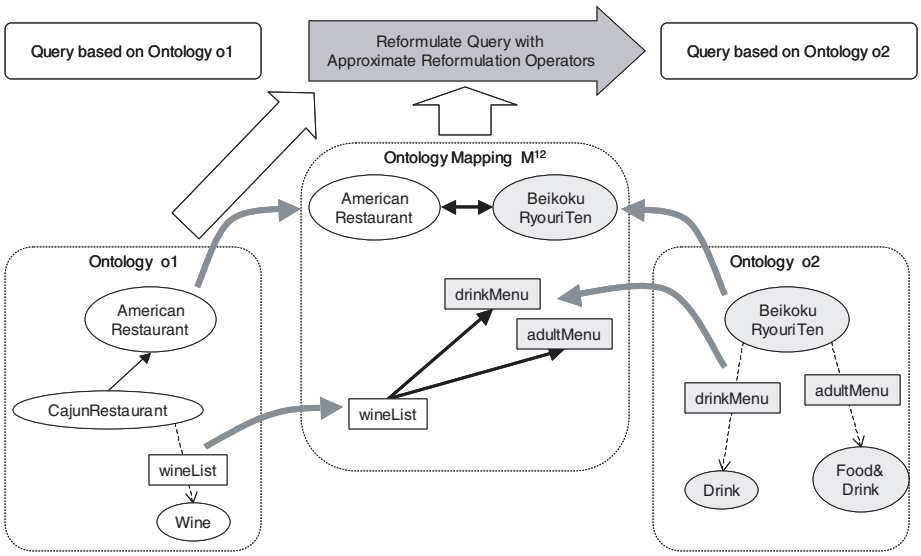


Figure 1: Approximate Query Reformulation Framework

- C^i *where* $C^i \in \mathcal{C}^i_0$.
- $\forall P^i.C^i$, $\exists P^i.C^i$, $\neg C^i$ *where* $C^i \in \mathcal{C}^i$, *and* $P^i \in \mathcal{P}^i$.

Ontologies are described by subsumption relations for classes and properties, and disjointness of classes. For the sake of simplicity, we do not distinguish individuals and data literals.

Definition 2 (Ontology Description) *Given a set $\mathcal{C}^i$ of classes and a set $\mathcal{P}^i$ of properties in an ontology O^i, and a set $\mathcal{L}$ of individuals and data literals, an ontology description of ontology O^i is a set $\mathcal{O}^i$ of axioms of the following forms:*

- $C^i_1 \sqsubseteq C^i_2$, $P^i_1 \sqsubseteq P^i_2$, $C^i_1 \sqsubseteq \neg C^i_2$,
- $a : C^i$, $\langle a, b\rangle : P^i$,

where $C^i_1, C^i_2, C^i \in \mathcal{C}^i$, $P^i_1, P^i_2, P^i \in \mathcal{P}^i$, and $a, b \in \mathcal{L}$.

For example, the ontology description $\mathcal{O}^1$ in ontology `o1` shown in Figure 1 contains the following axiom:

- $\texttt{CajunRestaurant}^1 \sqsubseteq \texttt{AmericanRestaurant}^1$.

(We denote the subsumption relation (i.e., `subClassOf` and `subPropertyOf`) as a solid arrow for easy visualization.)

The semantics of our ontology description language is defined using an interpretation function $\mathcal{I}$ in the usual way. Table 1 summarizes the syntax and semantics of our ontology description language. We denote $S \models A$ iff an axiom A logically follows from a set of axioms S.

We next define queries in ontologies. A query is a conjunction of a query about classes and a query about properties.

Definition 3 (Query) *Let $\mathcal{V}$ be a set of variables disjoint from $\mathcal{L}$. A query Q^i in ontology O^i is of the form*

$$Q^i_C \wedge Q^i_P,$$

where

- Q^i_C *is a conjunction of* $C^i(x)$ *where* $C^i \in \mathcal{C}^i$ *and* $x \in \mathcal{L} \cup \mathcal{V}$,
- Q^i_P *is a conjunction of* $P^i(x, y)$ *where* $P^i \in \mathcal{P}^i$ *and* $x, y \in \mathcal{L} \cup \mathcal{V}$.

For example, the following denotes a query about a Cajun restaurant that has Merlot on its wine list in ontology `o1`.

- $\texttt{CajunRestaurant}^1(x) \wedge \texttt{wineList}^1(x, \text{`}\texttt{Merlot}'\text{)}$.

The answer to a query Q^i can be obtained by substituting variables $v_1, \cdots, v_n$ contained in Q^i by a tuple $\langle a_1, \cdots, a_n\rangle$ of objects. We denote this substitution σ. The answer set $A(Q^i)$ is a set of tuples such that $\mathcal{O}^i \models Q^i\sigma$. The semantic relation between different queries are defined as follows.

Definition 4 (Query Containment) *For queries Q_1 and Q_2, Q_1 is said to be contained in Q_2 (denoted by $Q_1 \sqsubseteq Q_2$) if $A(Q_1) \subseteq A(Q_2)$.*

2.2 Mapping among Multiple Ontologies

In our framework, a query represented in one ontology is reformulated approximately into a query represented in another ontology by using an ontology mapping specification. In this paper, we describe ontology mapping specifications using the ontology description language.

Definition 5 (Ontology Mapping Specification) *Ontology mapping M^{ij} between ontology O^i and O^j is a set of axioms of the following forms:*

- $C^i \sqsubseteq C^j$, $C^j \sqsubseteq C^i$, $C^i \sqsubseteq \neg C^j$, $C^j \sqsubseteq \neg C^i$,
- $P^i \sqsubseteq P^j$, $P^j \sqsubseteq P^i$,

where $C^i \in \mathcal{C}^i$, $C^j \in \mathcal{C}^j$, $P^i \in \mathcal{P}^i$ and $P^j \in \mathcal{P}^j$.

The mapping range $R(M^{ij})$ of ontology mapping M^{ij} is defined to be a set of classes and properties in ontology O^j that appear in M^{ij}.

For example, the ontology mapping M^{12} between ontologies `o1` and `o2` in Figure 1 contains the following axioms:

- $\texttt{AmericanRestaurant}^1 \sqsubseteq \texttt{BeikokuRyouriTen}^2$.

OWL Construct	Syntax	Semantics
allValuesFrom	$\forall P^i.C^i$	$\{x \mid \forall y.\langle x,y\rangle \in \mathcal{I}(P^i) \supset y \in \mathcal{I}(C^i)\}$
someValuesFrom	$\exists P^i.C^i$	$\{x \mid \exists y.\langle x,y\rangle \in \mathcal{I}(P^i) \wedge y \in \mathcal{I}(C^i)\}$
complementOf	$\neg C^i$	$\mathcal{I}(\Delta) \setminus \mathcal{I}(C^i)$
subClassOf	$C_1^i \sqsubseteq C_2^i$	$\mathcal{I}(C_1^i) \subseteq \mathcal{I}(C_2^i)$
subPropertyOf	$P_1^i \sqsubseteq P_2^i$	$\mathcal{I}(P_1^i) \subseteq \mathcal{I}(P_2^i)$
disjointWith	$C_1^i \sqsubseteq \neg C_2^i$	$\mathcal{I}(C_1^i) \cap \mathcal{I}(C_2^i) = \phi$
(individual axioms)	$a : C^i$	$\mathcal{I}(a) \in \mathcal{I}(C^i)$
(individual axioms)	$\langle a,b\rangle : P^i$	$\mathcal{I}(\langle a,b\rangle) \in \mathcal{I}(P^i)$

Table 1: Syntax and Semantics of Ontology Description Language

- $\texttt{BeikokuRyouriTen}^2 \sqsubseteq \texttt{AmericanRestaurant}^1$.
- $\texttt{wineList}^1 \sqsubseteq \texttt{drinkMenu}^2$.
- $\texttt{wineList}^1 \sqsubseteq \texttt{adultMenu}^2$.

There may be many possible reformulated queries, but we prefer closer reformulation. We therefore adapt and extend the notion of *maximally-contained* reformulation [4] in the database literature. Specifically, we characterize two kinds of reformulation: minimally-containing reformulation and maximally-contained reformulation. In minimally-containing reformulation, the reformulated query minimally covers the original query. On the other hand, in maximally-contained reformulation, the reformulated query is maximally covered by the original query.

Assuming that ontology mapping M^{ij} is consistent with ontologies $\mathcal{O}^i$ and $\mathcal{O}^j$, we characterize approximate query reformulation using query containment in the merged ontology $\mathcal{O}^i \cup M^{ij} \cup \mathcal{O}^j$. We extend the definition of the answer set $A(Q)$ to be a set of tuples such that $\mathcal{O}^i \cup M^{ij} \cup \mathcal{O}^j \models Q\sigma$. Approximate query reformulation is defined as follows.

Definition 6 (Approximate Query Reformulation) *Let Q^i be a query in ontology O^i and Q^j be a query in ontology O^j described by classes and properties in the mapping range $R(M^{ij})$ of ontology mapping M^{ij} .*

- *Q^j is an equivalent reformulation of Q^i if $Q^j \sqsubseteq Q^i$ and $Q^i \sqsubseteq Q^j$.*
- *Q^j is a* **minimally-containing** *reformulation of Q^i if $Q^i \sqsubseteq Q^j$ and there is no other query Q_1^j such that $Q^i \sqsubseteq Q_1^j$ and $Q_1^j \sqsubseteq Q^j$.*
- *Q^j is a* **maximally-contained** *reformulation of Q^i if $Q^j \sqsubseteq Q^i$ and there is no other query Q_1^j such that $Q^j \sqsubseteq Q_1^j$ and $Q_1^j \sqsubseteq Q^i$.*

Recall the example query above. A reformulated query

- $\texttt{BeikokuRyouriTen}^2(x) \wedge \texttt{drinkMenu}^2(x, \text{`}\texttt{Merlot}')$

is not a minimally-containing reformulation, as there is a minimally-containing reformulated query as follows;

- $\texttt{BeikokuRyouriTen}^2(x) \wedge \texttt{drinkMenu}^2(x, \text{`}\texttt{Merlot}') \wedge \texttt{adultMenu}^2(x, \text{`}\texttt{Merlot}')$

3 Approximate Reformulation Operators

In this section, we provide approximate reformulation operators, which we call the most special generalizers, for minimally-containing reformulation.

A reformulated query consists of classes and properties appeared in the *range* of ontology mapping. Intuitively, classes and properties in a minimally-containing reformulation should minimally subsume those in the original query. Therefore, it is necessary to calculate the least upper bounds and the greatest lower bounds for classes and properties. We first define the least upper bounds and greatest lower bounds for a class and a property. This definition is an extended version of [7].

Definition 7 (Least Upper Bounds and Greatest Lower Bounds) *Let C be a class, O be a ontology description, and TC be a set of classes, then the least upper bounds $LUB(C,O,TC)$ and greatest lower bounds $GLB(C,O,TC)$ are defined as follows:*

- *$LUB(C,O,TC) = \{C' \mid C' \in TC, O \models C \sqsubseteq C'$ and there is no other $C_1' \in TC$ such that $O \models C \sqsubseteq C_1'$ and $O \models C_1' \sqsubseteq C'\}$.*
- *$GLB(C,O,TC) = \{C' \mid C' \in TC, O \models C' \sqsubseteq C$ and there is no other $C_1' \in TC$ such that $O \models C' \sqsubseteq C_1'$ and $O \models C_1' \sqsubseteq C\}$.*

The least upper bounds and greatest lower bounds for a property are defined similarly.

Minimally-containing reformulation requires calculation of the least upper bounds of classes and properties in the original query with respect to the merged ontology. For example, the least upper bounds of a class $\texttt{CajunRestaurant}^1$ with respect to ontology $\mathcal{O}^1 \cup M^{12} \cup \mathcal{O}^2$ in the mapping range $R(M^{ij})$ is the following set.

- $LUB(\texttt{CajunRestaurant}^1, \mathcal{O}^1 \cup M^{12} \cup \mathcal{O}^2, R(M^{12})) = \{\texttt{BeikokuRyouriTen}^2\}$.

Similarly, the least upper bounds of a property $\texttt{wineList}^1$ with respect to ontology $\mathcal{O}^1 \cup M^{12} \cup \mathcal{O}^2$ in the mapping range $R(M^{12})$ is the following set.

- $LUB(\texttt{wineList}^1, \mathcal{O}^1 \cup M^{12} \cup \mathcal{O}^2, R(M^{12})) = \{\texttt{drinkMenu}^2, \texttt{adultMenu}^2\}$.

Using the least upper bounds, we can define the most special generalizers for class queries and property queries.

Definition 8 (Most Special Generalizers) *Let $\mathcal{O}^i$ and $\mathcal{O}^j$ be ontology descriptions in ontology O^i and O^j and M^{ij} be an ontology mapping. A most special generalizer for a class query $C^i(x)$ is defined as follows:*

$$MSG(C^i(x), \mathcal{O}^i, \mathcal{O}^j, M^{ij}) = C_1^j(x) \wedge \cdots \wedge C_n^j(x),$$

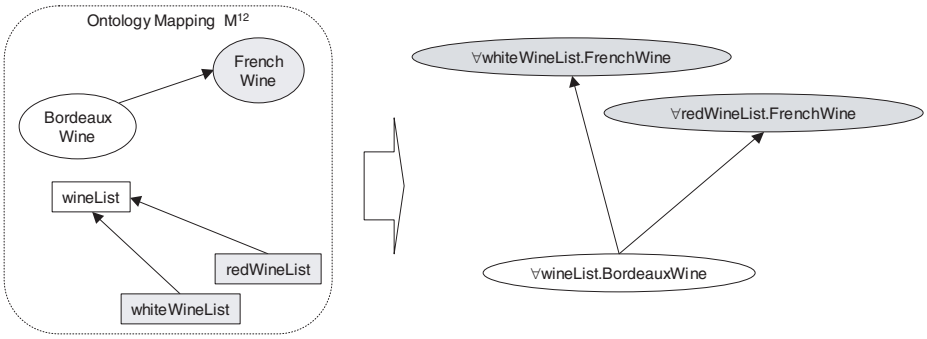


Figure 2: Least Upper Bounds for Value Restrictions

where $LUB(C^i, \mathcal{O}^i \cup M^{ij} \cup \mathcal{O}^j, R(M^{ij})) = \{C_1^j, \cdots, C_n^j\}$.

A most special generalizer for a property query $P^i(x, y)$ *is defined as follows:*

$$MSG(P^i(x,y), \mathcal{O}^i, \mathcal{O}^j, M^{ij}) = P_1^j(x,y) \wedge \cdots \wedge P_n^j(x,y),$$

where $LUB(P^i, \mathcal{O}^i \cup M^{ij} \cup \mathcal{O}^j, R(M^{ij})) = \{P_1^j, \cdots, P_n^j\}$.

Based on the above examples of least upper bounds, we have the following most special generalizers.

$MSG(\texttt{CajunRestaurant}^1(x), \mathcal{O}^1, \mathcal{O}^2, M^{12}) =$
$\texttt{BeikokuRyouriTen}^2(x)$.

$MSG(\texttt{wineList}^1(x, \texttt{"Merlot"}), \mathcal{O}^1, \mathcal{O}^2, M^{12}) =$
$\texttt{drinkMenu}^2(x, \texttt{"Merlot"})$
$\wedge\ \texttt{adultMenu}^2(x, \texttt{"Merlot"})$.

Applying these most special generalizers, the query in ontology $\circ 1$

- $\texttt{CajunRestaurant}^1(x) \wedge \texttt{wineList}^1(x, \text{`}\texttt{Merlot}\text{'})$

is reformulated approximately into the query in ontology $\circ 2$

- $\texttt{BeikokuRyouriTen}^2(x) \wedge \texttt{drinkMenu}^2(x, \text{`}\texttt{Merlot}\text{'}) \wedge$ $\texttt{adultMenu}^2(x, \text{`}\texttt{Merlot}\text{'})$.

The following theorem assures the correctness of our framework.

Theorem 1 *Let* Q^i *be a query in ontology* O^i*, then*

- *if* Q^i *is reformulated into* Q_g^j *in ontology* O^j *by the most special generalizers, then* Q_g^j *is a minimally-containing reformulation of* Q^i.

4 Computing Least Upper Bounds

As we have seen in the previous section, our approximate query reformulation framework requires computation of least upper bounds. This computation of least upper bounds for classes and properties varies according to the expressive power of ontology description languages. Our ontology description language only allows subsumption relationship for properties that are used in normal ontology description languages, such as OWL. It is therefore easy to compute the least upper bounds for properties using subsumption relationship.

However, our ontology description language allows value restrictions and negation for class description. In this section, we provide least upper bounds for value restrictions and negation. In the following, we write $LUB(C)$ instead of $LUB(C, \mathcal{O}^i \cup M^{ij} \cup \mathcal{O}^j, R(M^{ij}))$ for simplicity.

4.1 Value Restrictions

Least upper bounds for a value restriction such as $\forall P^i.C^i$ are a set of value restrictions in the target ontology. Because a value restriction consists of a class and a property, we have to take into consideration both class and property hierarchies.

We start with the observation that the following holds.

- If $P_1 \sqsubseteq P_2$ and $C_1 \sqsubseteq C_2$, then $\forall P_2.C_1 \sqsubseteq \forall P_1.C_2$ and $\exists P_1.C_1 \sqsubseteq \exists P_2.C_2$.

For example, if a white wine list property is a sub-property of a wine list property:

- $\texttt{whiteWineList}^2 \sqsubseteq \texttt{wineList}^1$,

and a Bordeaux wine class is a sub-class of a French wines class:

- $\texttt{BordeauxWine}^1 \sqsubseteq \texttt{FrenchWine}^2$,

then we have the following.

- $\forall \texttt{wineList}^1.\texttt{BordeauxWine}^1$ $\sqsubseteq \forall \texttt{whiteWineList}^2.\texttt{FrenchWine}^2$

Intuitively, a class that has only Bordeaux wines on its wine list is subsumed by a class that has only French wines on its white wine list, because the latter may also have Italian rose wine.

Thus, a value restriction that subsumes $\forall P^i.C^i$ consists from a sub-property (e.g., $\texttt{whiteWineList}^2$) of property P^i (e.g., $\texttt{wineList}^1$). Therefore, computation of the least upper bounds for value restriction $\forall P^i.C^i$ requires that of the greatest lower bounds of property P^i. A greatest lower bound is semantically a disjunction of all the element of the greatest lower bound. As properties occur negatively in $\forall P.C$, negation of the greatest lower bound is a conjunction of negation of each element.

Strictly speaking, the following proposition holds from the semantics of value restrictions.

Proposition 1 (Least Upper Bounds for Value Restrictions) *The least upper bounds for value restrictions are defined as follows:*

- $LUB(\forall P^i.C^i) = \{\forall P_k^j.C_l^j \mid P_k^j \in GLB(P^i)$ *and* $C_l^j \in LUB(C^i)\}$.
- $LUB(\exists P^i.C^i) = \{\exists P_k^j.C_l^j \mid P_k^j \in LUB(P^i)$ *and* $C_l^j \in LUB(C^i)\}$.

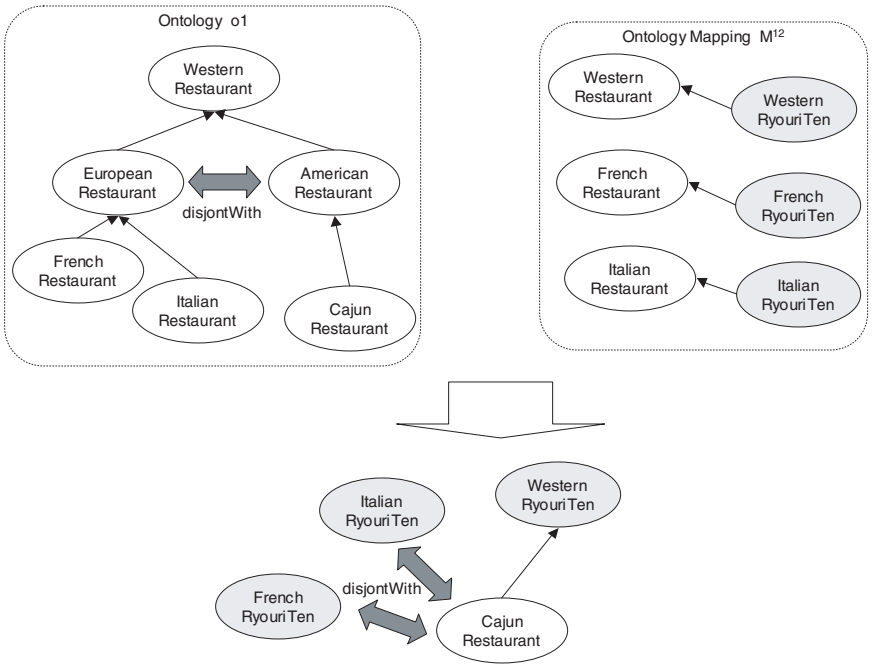


Figure 3: Least Upper Bounds for Negation

For example, the following can be calculated from the ontology mapping in Figure 2.

- $GLB(\texttt{wineList}^1) = \{\texttt{whiteWineList}^2, \texttt{redWineList}^2\}$.
- $LUB(\texttt{BordeauxWine}^1) = \{\texttt{FrenchWine}^2\}$.

We thus have the following as shown in Figure 2.

- $LUB(\forall \texttt{wineList}^1.\texttt{BordeauxWine}^1 = \{\forall \texttt{whiteWineList}^2.\texttt{FrenchWine}^2, \forall \texttt{redWineList}^2.\texttt{FrenchWine}^2\}$.

Note that least upper bounds for a value restriction are computed recursively. A naive algorithm may cause an infinite loop. However, a simple blocking mechanism is sufficient to avoid infinite recursion.

4.2 Negation

Negation is useful in practical applications. Consider, for example, the ontology description and the ontology mapping in Figure 3. The ontology description contains the following axioms. Suppose that one would like to make a query about a "Cajun" restaurant in ontology $\texttt{o2}$. There is no class corresponding to "Cajun" nor "American" restaurants. However, the "American" restaurant class is defined to be disjoint with "European" restaurant class, and there are corresponding classes for subclasses (e.g., "French" and "Italian" restaurants) of the "European" restaurant class. It is possible to construct a reformulated query for the query about a "Cajun" restaurant using these classes.

The ontology $\texttt{o1}$ contains the following.

- $\texttt{AmericanRestaurant}^1 \sqsubseteq \neg\texttt{EuropeanRestaurant}^1$.

Thus, computation of $LUB(\texttt{CajunRestaurant}^1)$ requires that of $LUB(\neg\texttt{EuropeanRestaurant}^1)$. It is easy to show that

- if $C_1 \sqsubseteq C_2$, then $\neg C_2 \sqsubseteq \neg C_1$.

Therefore, the class $\neg\texttt{EuropeanRestaurant}^1$ is subsumed by negation of each subclass as follows:

- $\neg\texttt{EuropeanRestaurant}^1 \sqsubseteq \neg\texttt{FrenchRestaurant}^1$.
- $\neg\texttt{EuropeanRestaurant}^1 \sqsubseteq \neg\texttt{ItalianRestaurant}^1$.

Strictly speaking, the following proposition holds from the semantics of negation.

Proposition 2 (Least Upper Bounds for Negation) *The least upper bounds for negation are defined as follows:*

- $LUB(\neg C^i) = \{\neg C^j_k \mid C^j_k \in GLB(C^i)\}$.

Applying the proposition to above example, we have

- $LUB(\neg\texttt{EuropeanRestaurant}^1) = \{\neg\texttt{FrenchRyouriTen}^2, \neg\texttt{ItalianRyouriTen}^2\}$.

Thus, a class for "Cajun" restaurant is approximated to a generalized class "WesternRyouriTen" except "French" and "Italian" as shown in Figure 3.

- $LUB(\texttt{CajunRestaurant}^1) = \{\texttt{WesternRyouriTen}^2, \neg\texttt{FrenchRyouriTen}^2, \neg\texttt{ItalianRyouriTen}^2\}$.

Computation of greatest lower bounds may require the greatest lower bounds for value restrictions. The following is a dual of Proposition 1.

Proposition 3 (Greatest Lower Bounds for Value Restrictions) *The greatest lower bounds for value restrictions are defined as follows:*

- $GLB(\forall P^i.C^i) = \{\forall P^j_k.C^j_l \mid P^j_k \in LUB(P^i)$ *and* $C^j_l \in GLB(C^i)\}$.
- $GLB(\exists P^i.C^i) = \{\exists P^j_k.C^j_l \mid P^j_k \in GLB(P^i)$ *and* $C^j_l \in GLB(C^i)\}$.

5 Related Work

Approximate terminological query framework [8] provides a formal framework for query approximation. In this framework, query approximation is used to improve the efficiency in a single ontology. Thus, the authors did not provide ontology mapping. They also introduce query containment, but it is used as a measure for the degree of query approximation. On the other hand, we address approximate query reformulation between ontologies and introduce minimally-containing reformulation and maximally-contained reformulation.

The approximate information filtering framework [7] has also been proposed. However, the author only dealt with simple class hierarchies and the maximally-contained reformulation in our framework. On the other hand, we deal with complex class description such as value restrictions and negation and minimally-containing reformulation, which requires non-standard inference in Description Logics.

Most previous research efforts on ontology integration have used ad-hoc mapping rules between ontologies (as surveyed in [9]). This approach allows flexibility in ontology integration, but most works do not provide semantics for the mapping rules. One exception is the Ontology Integration Framework [3] which provides clear semantics for ontology integration by defining *sound* and *complete* semantic conditions for each mapping rule. However, each mapping rule and its semantic conditions have to be specified by users. It is therefore difficult to ensure consistency in the mapping rules. In contrast, our framework can generate *sound* and *complete* mapping rules by specifying ontology mapping. It is relatively easy to check the consistency, since ontology mapping specifications are described as an ontology.

6 Conclusions

In this paper, we presented an approximate query reformulation framework for a fragment of OWL DL. In our framework, a query in one ontology is reformulated approximately into a query in another ontology based on an ontology mapping specification. To characterize closer reformulation, we introduced two types of reformulation: minimally-containing reformulation and maximally-contained reformulation. For the former, we provided the most special generalizers to reformulate a class (or property) expression in an original query into conjunction of the least upper bounds of the class (or property). We also provided a reformulation method for value restrictions and negation.

This paper focused on a fragment of OWL DL. Specifically, our ontology description language lacks unnamed conjunctions of classes (`intersectionOf`), disjunctions of classes (`unionOf`), and number restrictions (`cardinality`, etc.). However, number restrictions can be incorporated into our framework. The main reason for this restriction is that our framework reformulates each conjunct of queries. Further investigation is necessary for this direction.

As ontology mapping specifications are described in an ontology language, our framework is useful for dealing with updated or customized ontologies. If one makes queries based on an updated ontology for the original ontology, our framework can reformulate the queries based on ontology mapping between the original and updated ontologies.

The approximate query reformulation framework has been incorporated into the GeoLinkAgent system [1]. In the prototype system, agents coordinate regional information services provided by the GeoLink system, which is used in the Digital City Kyoto prototype [5]. Approximate query reformulation is required for such domains that have cross-cultural aspects, because ontologies vary from region to region due to cultural differences.

References

[1] J. Akahani, K. Hiramatsu, Y. Furukawa, and K. Kogure. Agent-based coordination of regional information services. In *Digital Cities II: Computational and Sociological Approaches*, LNCS 2362, pages 283–291. Springer-Verlag, 2002.

[2] J. Akahani, K. Hiramatsu, and T. Satoh. Approximate query reformulation based on hierarchical ontology mapping. In *Proc. of International Workshop on Semantic Web Foundations and Application Technologies (SWFAT)*, pages 43–46, 2003.

[3] D. Calvanese, G. De Giacomo, and M. Lenzerini. A framework for ontology integration. In I. Cruz, S. Decker, J. Euzenat, and D. McGuinness, editors, *The Emerging Semantic Web*, pages 201–214. IOS Press, 2002.

[4] A. Y. Halevy. Theory of answering queries using views. *SIGMOD Record*, 29(4):40–47, 2000.

[5] T. Ishida, J. Akahani, K. Hiramatsu, K. Isbister, S. Lisowski, H. Nakanishi, M. Okamoto, Y. Miyazaki, and K. Tsutsuguchi. Digital City Kyoto: Towards a social information infrastructure. In *Proc. of International Workshop on Cooperative Information Agents (CIA-99)*, pages 23–35, 1999.

[6] P. F. Patel-Schneider, P. Hayes, and I. Horrocks. Owl web ontology language semantics and abstract syntax. http://www.w3.org/TR/2003/WD-owl-semantics-20030331/, 2003.

[7] H. Stuckenschmidt. Approximate information filtering with multiple classification hierarchies. *International Journal of Computational Intelligence and Applications*, 2(3):295–302, September 2002.

[8] H. Stuckenschmidt and F. van Harmelen. Approximating terminological queries. In *Proc. of the 4th International Conference on Flexible Query Answering Systems (FQAS'02)*. Springer-Verlag, 2002.

[9] H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Huebner. Ontology-based integration of information – a survey of existing approaches. In *Proc. of IJCAI 2001 Workshop on Ontologies and Information Sharing*, 2001.

Fund Finder: A case study of database-to-ontology mapping

Jesús Barrasa, Oscar Corcho, Asunción Gómez-Pérez
(Ontology Group, Departamento de Inteligencia Artificial, Facultad de Informática, Universidad Politécnica de Madrid, Spain
(jbarrasa@eui.upm.es, ocorcho@fi.upm.es, asun@fi.upm.es)

Abstract: The mapping between databases and ontologies is a basic problem when trying to "upgrade" deep web content to the semantic web. Our approach suggests the declarative definition of mappings as a way to achieve domain independency and reusability. A specific language (expressive enough to cover some real world mapping situations like lightly structured databases or not 1st normal form ones) is defined for this purpose. Along with this mapping description language, the ODEMapster processor is in charge of carrying out the effective instance data migration. We illustrate this by testing both the mappings definition and processor on a case study.

Keywords: database-to-ontology mapping, ontology population, information integration.

1 Introduction

It is a well known fact that there is a large quantity of existing data on the web stored using relational database technology. This information is often referred to as the Deep Web [Bergman, 2001] as opposed to the surface web comprising all static web pages. Deep Web pages don't exist until they are generated dynamically in response to a direct request. As a consequence traditional search engines cannot retrieve its content and the only manageable way of adding semantics to them is attacking directly its source: the database.

The case study presented in this paper has been developed in the context of the ESPERONTO[1] project. This project aims to bridge the gap between the actual World Wide Web and the Semantic Web by providing a service to "upgrade" existing content to Semantic Web content, retrievable and exploitable in an automatic and efficient way by Semantic Web tools. In this effort, ontologies play a key role, aiming at unifying, bridging and integrating multiple heterogeneous digital content.

The Fund Finder application is about migrating relational database content to the semantic web. Typically the input to this kind of problem is a database that contains the data to be migrated and an ontology that we want to populate with instances extracted from the database.

The important idea behind the approach described in this paper is that mappings between entities, relationships and attributes in the database's relational schema and the corresponding concepts, relations and attributes of the ontology will be defined declaratively in a mapping document. This mapping document will be the input of a processor charged of carrying out the effective migration in an automatic way. The fact of defining these mappings declaratively will make our solution domain independent and reusable.

The level of complexity of the mappings to be defined will depend on the level of similarity of the ontology's conceptual model and the E/R model underlying the database. Normally, one of them will be richer, more generic or specific, better structured, etc., than the other. This paper is organized as follows: Section 2 contains a description of the specific test case in which the study is based. Section 3 describes the system's architecture and components. Section 4 gives a global view of our approach to database-to-ontology declarative mapping definition and a set of possible mapping situations. Section 5 describes the most important features of the eD2R mapping description language. Section 6 describes how our work relates to other experiences and approaches. And finally, section 7 comments and evaluates the results and conclusions of our case study and gives a glimpse of some future trends.

2 Case study

The database we want to migrate (FISUB) contains incentives and funds provided by the Catalan and Spanish Governments and by the European Union, for companies or entrepreneurs located in the Spanish region of Catalonia. It contains more than 300 registers that are updated manually on a daily basis.

The reason why we want to migrate these contents to the Semantic Web is to be able to aggregate to them information from other web resources related to funding in the European Union and to allow web users to ask intelligent queries about funding resources according to some parameters like their profile, to look for complementary ones, to check compatibilities and incompatibilities between types of funding, and so on.

The FISUB database is very lightly structured as it stores almost all information on a main table called *FUND_OPP* (funding opportunity). This table has 19

[1] http://www.esperonto.net

columns and among them, the most important ones (which will be used for our examples) are the following:

- *TITLE* stores the name or accronym assigned to the funding opportunity.
- *BEGIN_END* stores important dates related to the funding opportunity as the beginning and end of validity.
- *LEG_REF* stores the legal announcement or approval of the funding opportunity.
- *FUND_OP_TYPE* stores a short description about the type of funding: A text in natural language describing whether it is a prize, a credit, a tax discount or other.
- *URL* stores the funding's home page if it has one.

Some other tables like *SECTOR* (activity sector) and *AIM* are used to add information about the activity sector covered and the objectives aimed by a funding opportunity. These satellite tables are linked to the main table *FUND_OPP* through standard foreign key fields. The main elements in the relational database schema can be seen in figure 1.

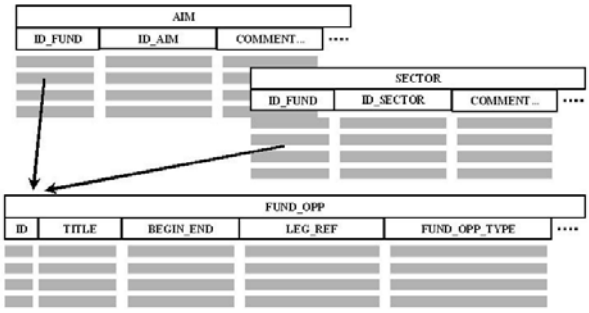


Figure 1: Excerpts from database tables.

The ontology to be populated is the Funding Opportunity ontology, which adds more structure and organization as well as enhanced inference and search capabilities to the legacy database. Figure 2 shows an excerpt of the ontology's concepts and relations.

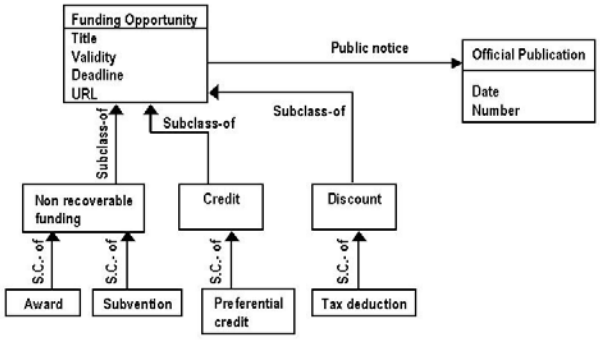


Figure 2: Excerpts from the Funding Opportunity ontology.

The mapping process is expected to extract instance data from the database and generate a set of instances committing to the funding opportunity ontology. Figure 2 shows graphically some of the expected results of this mapping. As can be seen, some record fields map directly their corresponding ontology attribute or relation (i.e. TITLE) but for some others this correspondence is not immediate (i.e. BEGIN_END) and some transformation is required. Let's have a look at some of these mapping situations.

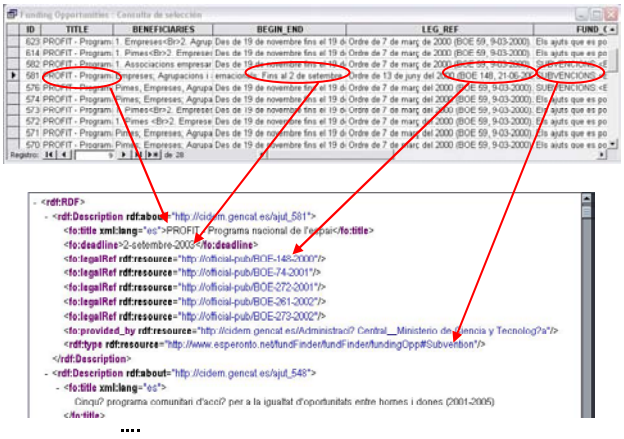


Figure 3: Results of the execution of the mapping between the FISUB database and the Funding Opportunity ontology.

- The TITLE field on the database maps directly the title property on the ontology because both refer to the same thing. The database field contains a string with the name or acronym that identifies the funding opportunity plus an optional short comment. In the example, "PROFIT" is the Spanish technical research support program.
- The BEGIN_END field on the database, needs to be transformed. It stores together the dates when the fund call opens and closes. In the ontology, the opening and closing dates are separate attributes, so some extraction needs to be done on the database field.
- The type of funding is determined by analysing the content of the field FUND_OPP_TYPE. If the keyword "subvention" appears in the field value, then the funding opportunity will be classified as a Subvention. If the keyword "prize" is found instead, then the type of the instance is Award, etc. As can be seen, keyword search particularly suits this case.
- The case of the LEG_REF data field is slightly more complicated. It stores a string referencing the (one or more) official publication in which the funding opportunity was proposed, approved, modified, cancelled, etc. by the competent authority. The corresponding element in the ontology is the LegalRef property and the fact of having more than one official publication mentioned into the LEG_REF field, which means the database is not in

first Normal Form (1NF), will lead to the generation of multiple relations to different instances from this single field value. As the official publications usually have alphanumeric codes as identifiers, regular expression evaluation seems adequate for this case.

3 System's architecture

Figure 4 resents the Fund Finder architecture. We have distinguished two layers: The modelling layer and the implementation layer (we are ignoring the formalism layer for the sake of clarity). At the first one we have the ontology conceptual model in the WebODE [Azpírez,2001] platform and the E/R model underlying the database. At the implementation layer, we have the ontology implemented in several ontology languages (OWL, DAML+OIL, RDF(S)…) using WebODE translators and the SQL implementation of the database relational model. An instance data sub-layer would contain instance data from the database (records) and instances of the ontology. The grey area in the figure shows the mapping definition and execution key elements.

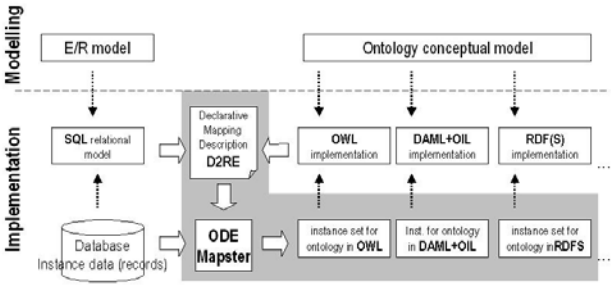


Figure 4: Diagram showing interactions between elements in our mapping approach.

- ***A declarative mapping description document: eD2R.*** This document contains the declarative definitions of the mappings between components in the SQL implementation of the relational database model and the ones in the ontology implementation. This documents is written in the eD2R mapping description language.
- ***The ODE Mapster processor*** is the software in charge of the mapping execution according to the directives of the aforementioned mapping document. The execution occurs automatically once the mappings are defined.
- ***A database*** containing the data to be migrated as instances of the ontology.
- ***An ontology*** to be populated with the data extracted from the database. The ontology can be expressed in any ontology implementation language, but instances of the ontology are generated in RDF in the first version of the processor.
- The automatically generated ***instance sets*** in RDF.

4 Global approach to database-to-ontology mapping

4.1 Declarative mappings

A declarative mapping is a set of explicit correspondences between components of two models. A mapping can be defined at different levels. In our case, it will be defined at the implementation level between a database's SQL description and an ontology's implementation. Furthermore, the intended direction of the mappings is from the database to the ontology, which means that we perform a process of data extraction from the database and we populate the ontology with the extracted information. That is why these correspondences will actually have the following form and not the other way round.

$OntologyComponent_i = Transformation(DatabaseComponent_j, DatabaseComponent_k ...)$

Where $OntologyComponent_i$ is any concept, attribute or relation in the target ontology and $DatabaseComponent_j$ is any database table or column.
A mapping between a database schema and an ontology can then be defined as a set of basic mapping expressions or *mapping elements* between components in both models like the one showed before. Inspired on the proposal of [Mena et al., 2001] and conveniently adapted to the specific case of databases, a basic mapping expression for a ***concept*** in the ontology will be defined as a 2-tuple <Rel, (a_1.. a_n)> where Rel is a SQL expression and a_1… a_n are columns of Rel that identify its objects (key columns). In other words, instances of concepts will be the records extracted from the database with an SQL query.

CONCEPT C1 : <Rel, (a_1..a_n)>

For an ***attribute*** or ***relation*** in the ontology, a basic mapping expression will be defined as a 4-tuple <Rel, (a_1.. a_n), (a_{n1}.. a_{nm}), f_{rl}> where Rel is a SQL expression; a_1.. a_n are attributes of Rel that identify its objects (the key columns); a_{n1}.. a_{nm} are columns of Rel that contain the attribute or relation values being mapped; and frl is a function f_{rl}:D_1x...xD_m → R that allows the transformation of the stored field data into the final values of the attribute or relation (D_i is the domain of field a_{ni} in the database and R is the range of the ontology's attribute or relation being described). In other words the value of an attribute or relation of the ontology will be obtained from one or more columns of an SQL expression directly or through the application of a transformation function.

ATT A1.1 : <Rel, (a_1..a_n), (a_{n1}…a_{nm}), fr_1>

The two mapping elements defined can be compacted in the following way:

CONCEPT C1 : <Rel, (a_1..a_n)>
ATT A1.1 : <(a_{n1}…a_{nm}), fr_1>
ATT A1.2 : <(a_{n1}…a_{nm}), fr_2> …

Where the ATT A1.i attribute mapping expressions inherit the two first elements (the SQL query Rel and the set of key columns a_1.. a_n) from their container CONCEPT C1.
What follows is an example of a mapping. We can see intuitively how the concept *FundingOpportunity* (the prefix *fo:* means that the concept is defined in the funding opportunity 'fo' ontology) maps all funding opportunities in the database marked as *new*. The mapping expression groups those records of the table *FUND_OPP* with value 1 in the field *NEW*. The different values of attribute ID identify the different records (ID is the key of the database table).
Within this concept mapping element a set of attribute or relation mapping elements can be defined. In the example the property *fo:title* maps directly the *TITLE* column and no function is applied to it's values. Attribute *fo:deadline* maps the *BEGIN_END* column after applying the function *getDeadline*. The same happens to the *fo:legalRef* relation, the column *LEG_REF* and the function *getLegalRef*. Functions used in the definitions should also be described in terms of the primitives provided by the mapping language being used, which will be discused later.

CONCEPT fo:FundingOpportunity :
*<[select * from FUND_OPP where*
FUND_OPP.NEW=1], FUND_OPP.ID>
ATTRIBUTE fo:title :
<FUND_OPP.TITLE, none>
ATTRIBUTE fo:deadline :
< FUND_OPP.BEGIN_END,getDeadline>
RELATION fo:legalRef :
< FUND_OPP.LEG_REF,getLegalRef>

4.2 Mapping cases

Based on the experience with the test case described in section 2, we have identified some mapping situations between the database implementation components and the concepts in the ontology. They are described and summarized in table 1. The second column in this table presents the database elements that can be mapped to an ontology concept, and the third column describes shortly the mapping case.

Table 1: Concept mapping cases

	Database implementation SQL element	Description
#1	View[2]	A view maps exactly one concept in the ontology.
#2	SELECT C1,…Cn FROM View	A subset of the columns in the view map a concept in the ontology.
#3	SELECT * FROM View WHERE f(C1,…Cn)	A subset (selection) of the records of a database view map a concept in the ontology.
#4	ImplicitSelect(View)	A subset of the records of a database view map a concept in the ontology but the selection cannot be made using SQL.
#5	T(Column)	One or more concepts can be extracted from a single data field.

Case #1 reflects the simplest mapping situation: The view in the database is semantically equivalent to the concept in the ontology and every record in the view corresponds to an instance of the ontology concept.

Case #2 is similar to case #1: the ontology concept and the database view refer to the same thing but two things may happen:

- The database view describes it with a higher level of detail by adding columns.
- In the view the relevant information for the specific concept we are interested in is merged with other concepts in the same view for optimisation purposes or just because of a bad structure of the database.

In **case #3**, the ontology concept is a subclass of the concept represented by the database table. The records in the database table being instances of the ontology concept can be extracted with an SQL query.

The same can be said for **case #4** with a peculiarity: the set of database records being instance of the ontology concept cannot be extracted with standard SQL and more complex techniques (i.e. keyword search, regular expression matching, natural language processing…) have to be applied on its data fields.

Finally **case #5** corresponds to situations in which a concept can be created out of a single column value. Or even more than one in the case of tables which are not in 1NF.

For ontology attributes and relations we have identified the following situations (the columns in table 2 are organized in the same way as those in table 1) :

[2] A view represents a single database table or any join of more than one table.

Table 2: Attributes and relations mapping cases

	Database element	Description
#1	Column	A column in a database view maps directly an attribute or a relation.
#2	T(Column)	A column in a database view maps an attribute or a relation after some transformation.
#3	n Column	A set of columns in a database view map an attribute or a relation.

Case #1 reflects the simplest mapping situation: both the column in the database is semantically equivalent to the attribute or relation in the ontology and share the same representation format. The correspondence is then direct.

Case #2 can cover three different cases:

1. The column in the database represents conceptually the same as the attribute or the relation in the ontology but they use a different representation format (i.e. currency unit transformation) and so the mapping needs a transformation function.
2. The column in the database stores the information needed to populate the ontology's attribute or relation but the information is mixed with other (noise) and it has to be extracted. Again a transformation function will be needed.
3. The same as the preceding one but furthermore, the column in the database stores more than one value (Not in 1NF) and each one of them needs to be extracted.

In **case #3**, the ontology's attribute or relation groups more than one database column. That means that the ontology property is less structured than its corresponding in the database. Let's take as an example the case of a postal address stored in a database using three columns one for the road name and number, another one for the postal code and a third one for the town name. These three fields would map one non-structured single field from the ontology containing the whole postal address resulting of the concatenation of the three column values in the database.

5 eD2R mapping description language

eD2R (extended D2R) is an extension of D2R MAP[3] which is a declarative, XML-based language to describe mappings between relational database models and ontologies implemented in RDFS developed at Freie Universität Berlin [Bizer, 2003].

D2R uses SQL statements in the mapping rules giving the possibility of handling highly normalized table structures, where instance data is spread over several tables. On the other hand, it fails to map low structured databases because of its limited expressiveness and we have enhanced with new primitives.

In D2R, basic concept mappings are defined using class maps. The class map is also the container of a set of attribute and property mapping elements called bridges (datatype property bridges and object property bridges respectively).

eD2R adds Operation and condition elements expressed in terms of elemental functions (Operation and Condition items) allowing the definition of complex and conditional transformations on field values based on techniques such as keyword search, regular expression matching, natural language processing and others. They cover all three case#2 attribute and relation mapping situations.

Classifier elements are used to apply what we called in section 4.2 Implicit Selections (selections which are not feasible via SQL queries) to classify elements in a taxonomy of concepts in the ontology.

Finally eD2R's field map elements are used for concept extraction from data fields and correspond to case #5 in the concept mapping cases table.

A detailed explanation of the eD2R mapping description language can be found at [Aguado, 2003]. The diagram in figure 5 shows the original elements in D2R and the ones in eD2R.

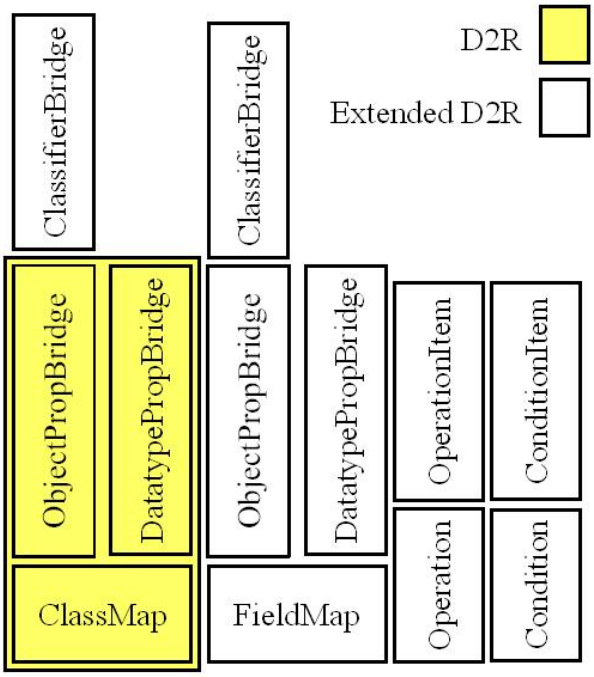


Figure 5: D2R and eD2R mapping description languages' elements.

[3] D2R MAP (Database to RDF) is available at: http://www.wiwiss.fu-berlin.de/suhl/bizer/d2rmap/D2Rmap.htm

6 Related work

Recent approaches like [Stojanovic et al.,2002] define mappings between a database and a ontology semi-automatically generated from the database's relational model. The level of similarity between both models is very high and mappings are consequently quite direct. They don't deal with complex mapping situations like the ones defined in section 4.2.

The same stands for REVERSE[4], an early prototype for mapping relational database content to ontologies, which is integrated in the Karlsruhe Ontology and Semantic Web Tool Suite (KAON).

[Handschuh et al., 2003] facilitates the manual definition of mappings, through the use of a server-side web page markup with information about the underlying database and its relation with the web page content (Web site cooperativity assumption). Their approach doesn't seem to deal with complex mapping situations like the ones tackled in this paper.

[Beckett and Grant, 2003] surveys and discusses mapping approaches to and from relational schemas.

Similar approaches to this work can be also found in the Intelligent Information Integration area, in which data from existing heterogeneous databases are extracted according to ontologies and then combined. Examples of such systems are Observer [Mena et al., 2000] and Picsel [Goasdoué et al., 2000], among others. The main differences with respect to our approach is that in these systems the mapping between the ontologies and the databases from which the ontology instances are extracted are not created declaratively but with ad-hoc software implementations.

7 Results, conclusions and future work

To sum up, the main outcomes of our experience are the following:

- The identification and characterization of a significant set of mapping situations when content stored in database is migrated into an ontology.
- Extension of D2R MAP with new features covering all the situations mentioned in section 2.
- Implementation of the ODEMapster processor to carry out the effective migration according to the definitions expressed using eD2R.
- Experimentation on a real world test case. The Fund Finder application.

Regarding the future trends of our work, intensive testing with other databases is being carried out and will continue as well as the enhancements to eD2R language.

The eD2R language has become quite complex as a counter-effect to its expressivity and the creation of a mapping document becomes a tedious, time consuming and error-prone task. A graphical user interface to support this activity is actually under development.

Acknowledgements

This work is partially supported by a FPU grant from the Spanish Ministry of Education (AP2002-3828), and by the IST project Esperonto (IST-2001-34373).
We would like to thank Raúl Blanco and Carles Gómara from CIDEM for providing the database and all information needed.

References

[Aguado, 2003] Aguado G, Barrasa J, Corcho O, Gómez-Pérez A, Suárez M, Blanco R, Gómara C. *Accompanying document to D8.3 Test Case application development. Fund Finder.* Esperonto project deliverable. October 2003.

[Azpírez,2001] Azpírez J, Corcho O, Fernández-López M, Gómez-Pérez A. *WebODE: a Workbench for Ontological Engineering.* First International Conference on Knowledge Capture (K-CAP01). Victoria B.C., Canada. October 2001

[Beckett and Grant, 2003] Beckett D, Grant J (2003) SWAD-Europe Deliverable 10.2: Mapping Semantic Web Data with RDBMSes. Technical report.

[Bergman, 2001] Bergman MK. *The Deep Web: Surfacing hidden value*. White paper. Sept 2001.

[Bizer, 2003] Bizer C. *D2R MAP – A Database to RDF Mapping Language*. 12th International World Wide Web Conference, Budapest. May 2003.

[Goasdoué et al., 2000] Goasdoué F, Lattes V, Rousset M (2000) *The Use of CARIN Language and Algorithms for Information Integration: The PICSEL Project*. International Journal of Cooperative Information Systems (IJCIS) 9(4):383–401

[Handschuh et al., 2003] Handschuh S, Staab S, Volz R. *On deep annotation.* 12th International World Wide Web Conference, Budapest. May 2003.

[Mena et al., 2000] Mena E, Illarramendi A, Kashyap V, Sheth AP (2000) *OBSERVER: An Approach for Query Processing in Global Information Systems based on Interoperation across Pre-existing Ontologies*. International Journal on Distributed and Parallel Databases 8(2):223–271

[Mena et al., 2001] Mena E, Illaramendi A. *Ontology-based query processing for global information systems*. Kluwer Academic Publishers. Pags:86-88. 2001.

[Stojanovic et al.,2002] Stojanovic L, Stojanovic N, Volz R. *Migrating data-intensive web sites into the Semantic Web.* Proceedings of the ACM Symposium on Applied Computing, Madrid 2002.

[4] http://kaon.semanticweb.org/alphaworld/reverse/view

Building an Ontology with MOMIS*#

Domenico Beneventano[1,2], Sonia Bergamaschi[1,2], Francesco Guerra[1]

DII - Università di Modena e Reggio Emilia
Via Vignolese 905 - Modena
{lastname.firstname}@unimo.it

[2]IEIIT-CNR Istituto di Elettronica e di Ingegneria dell'Informazione
e delle Telecomunicazioni
Viale Risorgimento 2 – Bologna

1. Introduction

Nowadays the Web is a huge collection of data and its expansion rate is very high. Web users need new ways to exploit all this available information and possibilities. A *new vision of the Web*, the Semantic Web[1], where resources are annotated with machine-processable metadata providing them with background knowledge and meaning, arises. A fundamental component of the Semantic Web is the ontology; this "*explicit specification of a conceptualization*" [6] allows information providers to give a understandable meaning to their documents.

MOMIS (Mediator envirOnment for Multiple Information Sources) [3] is a framework for information extraction and integration of heterogeneous information sources. The system implements a semi-automatic methodology for data integration that follows the *Global as View* (GAV) approach [11]. The result of the integration process is a global schema, which provides a reconciled, integrated and virtual view of the underlying sources, GVV (Global Virtual View). The GVV is composed of a set of (global) classes that represent the information contained in the sources. In this paper, we focus on the MOMIS application into a particular kind of source (i.e. web documents), and show how the result of the integration process can be exploited to create a conceptualization of the underlying domain, i.e. domain ontology for the integrated sources. GVV is then semi-automatically annotated according to a lexical ontology. With reference to the Semantic Web area, where generally the annotation process consists of providing a web page with semantic markups according to an ontology, we firstly markup the local metadata descriptions and then the MOMIS system generates an annotated conceptualization of the sources. Moreover, our approach "builds" the domain ontology as the synthesis of the integration process, while the usual approach in the Semantic Web is based on "a priori" existence of ontology.

2. The MOMIS system

In this section, we describe the information integration process for building the GVV of a web pages' set. The process is shown in Figure 1.

2.1 ODL$_{I3}$

For a semantically rich representation of schemas and object patterns, MOMIS uses an object-oriented language called ODL$_{I3}$, which is an evolution of the OODBMS standard language ODL. ODL$_{I3}$ extends ODL with the following relationships expressing intra- and inter-schema knowledge for the source schemas:

- SYN (synonym of) is a relationship defined between two terms t_i and t_j that are synonyms in every involved source.
- BT (broader terms) is a relationship defined between two terms t_i and t_j, where ti has a broader, more general meaning than t_j. The opposite of BT is NT (narrower terms).
- RT (related terms) is a relationship defined between two terms t_i and t_j that are generally used together in the same context in the considered sources.

By means of ODL$_{I3}$, only one language is exploited to describe both the sources (the input of the synthesis process) and the GVV (the result of the process). The translation of ODL$_{I3}$ descriptions into one of the Semantic Web standards such as RDF, DAML+OIL, OWL is a straightforward process. In fact, from a general perspective an ODL$_{I3}$ concept

[1] http://www.w3.org/2001/sw

[#] A complete version of this work appears on IEEE Internet Computing's special "Zen of the Web" issue, Sep-Oct 2003, 42-51

corresponds to a *Class* of a the Semantic Web standard, and ODL$_{I3}$ relationships are translated into *properties* (in particular the ISA ODL$_{I3}$ relationships are *subclassof* in the Semantic Web standards).

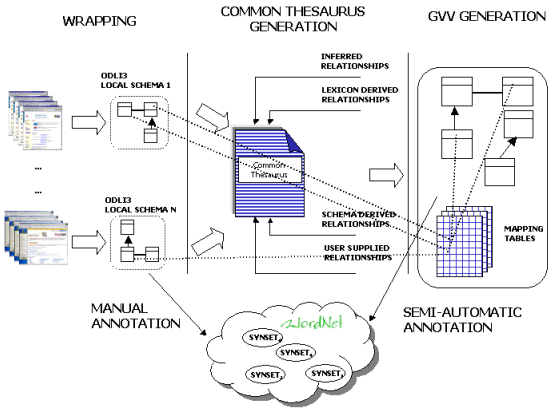


Figure 1: An overview of the ontology integration process

2.2 Wrapping: extracting data structure for sources

A wrapper logically converts the source data structure into the ODL$_{I3}$ information model. The wrapper architecture and interfaces are crucial, because wrappers are the focal point for managing the diversity of data sources.

For conventional structured information sources (e.g. relational databases), schema description is always available and can be directly translated. For semistructured information sources, a schema description is in general not directly available at the sources. A basic characteristic of semistructured data is that they are "self-describing" hence information associated with the schema is specified within data. Thus, a wrapper has to implement a methodology to extract and explicitly represent the conceptual schema of a semi-structured source. We developed a wrapper for XML/DTDs files. By using that wrapper, DTD elements are translated into semi-structured objects, according to different proposed methods [1], and in particular the OEM model [12].

Information is available on the Web mainly in HTML pages that are human-readable but cannot easily be automatically accessed and manipulated. In particular, HTML language does not separate data structure from layout. Thus, we need a further preliminary step of extraction: by means of Lixto [5], we translate the content of a web page (data and data structure) into a XML file, then we exploit the previously developed wrapper XML/DTD to acquire the source descriptions.

2.3 Running example

We consider how to build an ontology from two web sources related to the University domain. By means of a Lixto generated wrapper, the source content is translated into XML files

University Site (UNI)

```
<!ELEMENT UNI(People*)>
<!ELEMENT People (Research_Staff* |
School_Member*)>
...
<!ELEMENT Research_Staff(name,
  e-mail, Section*, Article*)>
<!ELEMENT Section(name, year.  period)>
<!ELEMENT Article(title, year,  journal,
conference)>
<!ELEMENT School_Member(name,  e-mail)>
<!ELEMENT name (#pcdata)> ...
```

Computer Science Site (CS)

```
<!ELEMENT CS(Person*)>
  ...
<!ELEMENT Person(Professor*|Student*)>
<!ELEMENT Professor(
first_name,last_name, e-mail,
Publication*)>
<!ELEMENT Student(name, e-mail)>
<!ELEMENT Course(denomination,
Professor)>
<!ELEMENT Publication(title,
journal, editor)>
<!ELEMENT School_Member(name, e-mail)>
<!ELEMENT name (#pcdata)>...
```

according to the DTDs sketched in Table 1.

Table 1: A fragment of the University (UNI) and Computer Science (CS) DTDs

By means of the XML/DTD wrapper, the obtained DTDs are translated into ODL$_{I3}$ descriptions. An example of the classes obtained in this step is shown in Table 2.

2.4 Annotation of a local source with WordNet

The WordNet database [10] contains 146,350 lemma organized in 111,223 synonym sets. WordNet's starting point for lexical semantics comes from the conventional association between the forms of the words - that is, the way in which words are pronounced or written - and the concept or meaning they express. These associations give rise to several

properties, including synonymy, polysemy, and so forth. The correspondence between the words form (F_i) and their meaning (M_j) is synthesized in the so-called Lexical Matrix LM, where the element Lmij is true if the word form F_i can be

```
University Site (UNI)
...
Interface Research_Staff
(Source Un_site.dtd)
{ attribute string name;
  attribute string email;
  attribute set < Section > section;
  attribute set < Article > article;}

Interface Article
(Source Un_site.dtd)
{ attribute string title;
  attribute string journal;
  attribute string conference;
  attribute string year; }
```

```
Computer Science Site (CS)
…
Interface Professor
(Source Sc_site.dtd)
{  attribute string first_name;
   attribute string last_name;
   attribute string email;
   attribute set < Publication > publication;}

Interface Publication
(Source Sc_site.dtd)
{ attribute string title;
  attribute string journal;
  attribute string editor }
```

Table 2: A piece of the University (UNI) and Computer Science (CS) sources in ODL$_{I^3}$

used to express word meaning M_j.If $LM_{i,1}$, …, $LM_{i,k}$ k>1 are true, then the word form F_i is polysemous (i.e. it can be used to represent more than one meaning, M_1, …, M_k); if $LM_{1,j}$, …, $LM_{p,j}$ p>1 are true, then the word form $F_{i, \ldots,}$ Fp are synonyms.

The integration designer has to manually choose the appropriate WordNet meaning for each element of the conceptual schema. The annotation phase is composed of two different steps:

- **Word Form choice.** In this step, the WordNet morphologic processor aids the designer by suggesting a word form corresponding to the given term.
- **Meaning choice.** The designer can choose to map an element on zero, one or more senses.

This phase assigns a name, LEN (this name can be the original one or a word form chosen from the designer), and a set (eventually empty) of meanings, LEM_i (a class or attribute meaning is given by the disjunction of its set of meanings), to each local element (class or attribute) LE of the local schema:

$LE = \langle LEN, \{LEM_1, \ldots, LEM_k\} \rangle, \; k \geq 0$

For example:

$CS.Course = \langle course, \{course\#1\} \rangle$

where Course#1 = 'education imparted in a series of lessons or class meetings'

In order to improve the accuracy of local source annotations with WordNet, we are evaluating how to extend WordNet. If a source description element (i.e. a class or an attribute name) has no correspondent in the reference lexical ontology (WordNet in our case), the designer may add a new meaning and proper relationships to the existing meanings

2.5 Common Thesaurus Generation

MOMIS constructs a Common Thesaurus describing intra and inter-schema knowledge in the form of SYN, BT, NT, and RT relationships. The Common Thesaurus is constructed through an incremental process in which relationships are added in the following order:

1. *schema-derived relationships*: relationships holding at intra-schema level are automatically extracted by analyzing each schema separately. For example, analyzing XML data files, BT/NT relationships are generated from couples IDs/IDREFs and RT relationships from nested elements.
2. *lexicon-derived relationship:* we exploit the annotation phase in order to translate relationships holding at the lexical level into relationships to be added to the Common Thesaurus. For example, the hypernymy lexical relation is translated into a BT relationship.
3. *designer-supplied relationships*: new relationships can be supplied directly by the designer, to capture specific domain knowledge. If a nonsense or wrong relationship is inserted, the subsequent integration process can produce a wrong global schema;
4. *inferred relationships*: Description Logics techniques of ODB-Tools [2] are exploited to infer new relationships, by means of subsumption computation applied to a "virtual schema" obtained by interpreting BT/NT as subclass relationships and RT as domain attributes.

In our running example, some of the relationships automatically obtained by MOMIS and proposed at the integration designer are the following (the number denotes the kind of derivation of relationships):

```
1 CS.Professor NT CS.Person
2 UNI.Article NT CS.Publication
3 UNI.Research_Staff SYN
CS.Professor
4 UNI.Research_Staff NT
CS.Person
```

2.6 GVV generation

The MOMIS methodology allows us to identify similar ODL$_{I^3}$ classes, that is, classes that describe the same or semantically related concept in different sources. To this end, *affinity coefficients* are evaluated for all possible pairs of ODL$_{I^3}$ classes, based on the relationships in the Common Thesaurus properly strengthened. Affinity coefficients determine the degree of matching of two classes based on their names (*Name Affinity* coefficient) and their attributes (*Structural Affinity* coefficient) and are fused into the *Global Affinity* coefficient, calculated by means of the linear combination of the two coefficients [4]. Global affinity coefficients are then used by a hierarchical clustering algorithm, to classify ODL$_{I^3}$ classes according to their degree of affinity.

For each cluster Cl, a **Global Class** GC, with a set of **Global Attributes** GA_1, …, GA_N , and a **Mapping Table** MT, expressing mappings between local and global attributes, are defined.

The Mapping Table is a table whose columns represent the local classes (LC), which belong to the Global Class and whose rows represent the global attributes. An element `MT[GA][LC]` is a function which represents how local attributes of LC are mapped into the global attribute GA :

`MT[GA][LC]= f(LAS)`

where LAS is a subset of the local attributes of LC.

Some simple and frequent cases of such function are the following:

- *identity*: LAS is a singleton, LAS = {LA}, and f is the identity function; in this way we express that the GA value is equal to the `LA` value; we denote this case as `MT[GA][LC] = LA`
- *constant*: `GA` assumes into LC a constant value set by the designer; we denote this case by `MT[GA][LC] = const`
- *undefined*: `GA` is undefined into `LC`; we denote this case as `MT[GA][LC] = null`.

The Global Class and Mapping Table generation is a synthesis activity performed interactively with the designer. A preliminary set of Global Attributes GA_1, …, GA_N and mappings are automatically generated, and proposed to the designer, as follows.

First, local attributes of the local classes belonging to GC are grouped on the basis of SYN and BT/NT relationships among local attributes.

Formally, let $\leftrightarrow$ be a relation defined between two local attributes LA_1 and LA_2 as follows:

$LA_1 \leftrightarrow LA_2$ iff LA_1 SYN LA_2 or LA_1 BT LA_2 or LA_1 NT LA_2 is in the Common Thesaurus.

Let $\Leftrightarrow$ be the equivalence relation defined as the transitive-reflexive-symmetric closure of $\leftrightarrow$.

Given a local attributes LA, [LA] denotes the equivalence class of LA w.r.t. $\Leftrightarrow$. Given a Global Class GC, we consider a Global Attribute GA, for each element of the set

{[LA] | LA is an attribute of LC and LC $\in$ GC}

For each element of the mapping table `MT[GA][LC]=f(LAS)` the proposed set LAS is the set of the attributes of the local class LC which belong to the equivalence class related to GA. This set can be:

- *empty*: GA does not have any representation in the local class LC: in this case the designer has to choose between the undefined (default) or the constant function;
- *a singleton* : the function f may represent the identity function (default), i.e. GA and LA represent the same information, or f is a translation function.

In our running example the clustering process gives rise to three global classes:

```
Global1:(UNI.Section, CS.Course)
Global2: (UNI.Article,
CS.Publication)
Global3: (UNI.Research_Staff,
UNI.School_Member, CS.Professor,
CS.Student)
```

and, for Global2, the following Mapping Table, where all the maps are automatically produced except for $Const_1$ set by the designer, is generated.

	UNI.Article	CS.Publication
Title	Title	Title
Year	Year	$Const_1$ [2]
Journal	Journal	Journal
Conference	Conference	NULL
Editor	NULL	Editor

Table 4: Mapping Table of the global class Global2 (Publication)

3 Global Virtual View Annotation

In this section, we propose a semi-automatic methodology annotate a GVV, i.e. to assign a name, GEN, and a set (eventually empty) of meanings, GEM_i (a class or attribute meaning is given by the disjunction of its set of meanings) to each global element (class or attribute) GE:
$GE = \langle GEN, \{GEM_1, \ldots, GEM_p\} \rangle, p \geq 0$

3.1 Global Class Annotation

In order to semi-automatically associate an annotation to each global class, we consider the set of all its "broadest" local classes, w.r.t. the relationships included in the Common Thesaurus, denoted by GC_B:

$GC_B = \{ LC \in GC \mid \neg\exists y \in GC, (LC\ NT\ y) \}$

In our example:

	GC	GC_B
GC_1	CS.Course, UNI.Section	CS.Course, UNI.Section
GC_2	CS.Publication, UNI.Article	CS.Publication
GC_3	CS.Professor, CS.Person,UNI.School_Member, UNI.Research_Staff, CS.Student	CS.Person

On the basis of GC_B, the designer will annotate the global class GC as follows:

- **name choice**: the integration designer is responsible for the choice of the GC name: the system only suggests a list of possible names. The designer may select a name, i.e. a label to identify the GC, within the proposed list or select another name not belonging to the list.
- **meaning choice**: the union of the meanings of the local class names in GC_B are proposed to the designer as meanings of the Global Class. The designer may change this set, by removing some meanings or by adding other ones.

With respect to our example, the proposed annotations are the following:

GC	Names	Meanings
GC_1	course or section	course#1
GC_2	Publication	Publication#1
GC_3	University_Member	person#1

Table 5: University GVV annotation

3.2 Global Attributes Annotation

We extend the previously used approach for names and meanings of the attributes. Given a global attribute GA of the global class GC, we consider the set LGA of local attributes, which are mapped into GA:

$LGA = \{LA \mid \exists LC \in GC, LA \in LC \wedge MT[GA][LA] \neq null\}$

and the set of all its ``broadest'' local attributes, denoted by LGA_B:

$LGA_B = \{LA \in LGA \mid \neg\exists y \in LGA, (LA\ NT\ y)\}$

On the basis of LGA_B, the designer will annotate the global attribute as described for global classes. Moreover, according to mapping function, we may develop some specific policy to automatically select meanings.

4 Concluding remarks and future work

In this paper, we presented a methodology for supporting the semi-automatic building, annotation of a domain ontology obtained by integrating web documents with the MOMIS system. Some methodologies that aid the generation process of semantic mappings between data sources and mediated schema, starting from annotated schemas, have been presented; as pointed in [7], generating semantic

[2] For example, in order to specify all the publications of CS source are published on 2003, the designer may set $Const_1$=2003

mappings is a current challenge in data integration. In this paper, we do not take into account problems arising when two o more schemas are merged [13].

The annotated ontology may be exploited to support dynamics issues, i.e. to have ontology consistent with the domain that refers to. Many interesting solutions have been developed with regard to this topic [8,9] and an outstanding idea is to exploit multiple variants of the same ontology to cope with changes. This approach, called o*ntology versioning*, is different from our idea where a single ontology has to be kept consistent with the sources, which refer to. So, if new sources are added/deleted, or if some changes occur in the sources, the corresponding GVV has to change. In order to restart the integration process from scratch, we are developing a methodology for integrating a new source, which exploits the previous integration work, i.e., a built-up GVV, without restarting the integration process from scratch.

The Momis methodology is currently adopted in the Sewasie (Semantic Web Agents in Integrated Economies) European research project (www.sewasie.org). Sewasie's goal is to design and implement an advanced search engine that enables intelligent access to heterogeneous data sources on the Web via semantic enrichment that provides the basis for structured secure Web-based communication. To achieve this goal, Sewasie realizes a virtual network, whose nodes are Sewasie Information Nodes (SINode). SINodes are mediator-based systems that represent a virtual view of the overall information managed within any SINode and consists of the managed information sources, wrappers, and a metadata repository. We think that the methodology implemented in Momis could be exploited to create the kernel of an SINode

Bibliografy

[1] S. Abiteboul, P. Buneman, and D. Suciu. "Data on the Web - From Relations to Semistructured Data and XML". Morgan Kaufmann, 2000.

[2] D. Beneventano, S. Bergamaschi, C. Sartori, M. Vincini "ODB-QOptimizer: a tool for semantic query optimization in OODB." Int. Conference on Data Engineering ICDE97, UK, April 1997.

[3] S. Bergamaschi, S. Castano, D. Beneventano, M. Vincini: "Semantic Integration of Heterogeneous Information Sources", DKE, Vol. 36, Num. 1, Pages 215-249, Elsevier Science B.V. 2001.

[4] S. Castano, V. De Antonellis, S. De Capitani di Vimercati. Global viewing of heterogeneous data sources. IEEE Transactions on Data and Knowledge Engineering, 13(2), 2001.

[5] R. Baumgartner, S. Flesca, G. Gottlob: Visual Web Information Extraction with Lixto. VLDB 2001: 119-128

[6] T. R. Gruber. A translation approach to portable ontologies. Knowledge Acquisition, 5(2):199-220, 1993

[7] A. Halevy Data Integration: a Status Report. Proceedings of the German Database Conference, BTW-03

[8] J. Heflin, J. Hendler Dynamic Ontologies on the Web. In Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000). AAAI/MIT Press, CA, 2000. pp. 443-449.

[9] M. Klein, D. Fensel Ontology Versioning on the Semantic Web. In 1th Semantic Web Working Symposium. 2001.

[10] A.G. Miller. A lexical database for English. Communications of the ACM, 38(11):39:41,1995

[11] M. Lenzerini Data Integration: A Theoretical Perspective. PODS 2002: 233-246

[12] Y. Papakonstantinou, H. Garcia-Molina, J. Widom. Object exchange across heterogeneous information sources. In Proc of ICDE95, Taiwan, 1995

[13] R.A. Pottinger, P. A. Bernstein: Merging Models Based on Given Correspondences, University of Washington Technical Report UW-CSE-03-02-03. February 2003.

An algorithm for semantic coordination

P. Bouquet[2,3], L. Serafini[3], S. Zanobini[2] and M. Benerecetti[1],

[1]Dept. of Physical Sciences
University of Naples
Via Cinthia – 80126 Napoli (Italy)

[2]Dept. of Information and Communication Technology
University of Trento
Via Sommarive, 10 – 38050 Trento (Italy)

[3]ITC-IRST – Istituto per la Ricerca
Scientifica e Tecnologica
Via Sommarive, 14 – 38050 Trento (Italy)

bene@na.infn.it, bouquet@dit.unitn.it, serafini@itc.it, zanobini@dit.unitn.it

Abstract

The problem of finding an agreement on the meaning of heterogeneous semantic models is one of the key issues in the development of the Semantic Web. In this paper, we propose (i) a *general algorithm* which implements a new approach, called CTXMATCH, for discovering (semantic) relationships across distinct and autonomous *generic structures* and (ii) a *specific algorithm* specializing the algorithm to the discovering of mappings across *hierarchical classifications*. This approach shifts the problem of semantic coordination from the problem of computing linguistic and/or structural similarities between semantic-based structures (what most other proposed approaches do), to the problem of deducing relations between sets of logical formulas that represent the meaning of concepts belonging to different structures.

1 Introduction

The approach to semantic coordination we proposed in [6, 7] is based on the intuition that there is a huge conceptual difference between coordinating abstract structures (e.g., arbitrary labelled graphs) and coordinating structures labeled with expressions of a language spoken by the community of their users. The latter ones give us the chance to exploit the complex degree of semantic coordination implicit in the way a community uses the language from which the labels are taken.

We believe that at least three distinct levels of semantic knowledge are needed in order to semantically coordinate structures labelled with natural language:

• **Lexical knowledge:** knowledge about the words used in the labels. For example, the fact that the word 'image' can be used to mean a picture or a personal facade;

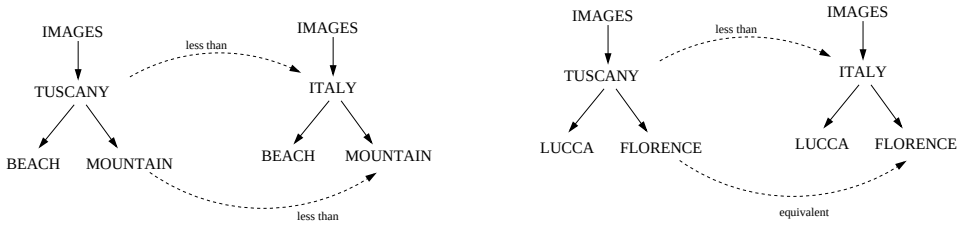
• **Domain knowledge:** knowledge about the relation between senses of labels in the real world or in a specific domain. For example, the fact that Florence is both a city of Italy and of Tuscany;

• **Structural knowledge:** knowledge deriving from the way the labels are arranged in a given structure. For example, the fact that the node `MOUNTAIN` in Figure 1.a can be used to classify images of mountains, and not books.

In [6, 7] we deeply motivate this choice. To summarize these motivations, consider the hierarchical classifications (hereafter HC) in Figure 1 used to classify images in two multi-media repositories. We want to discover the semantic relation between the nodes labelled `MOUNTAIN` in the two HCs in Figure 1.a, and between the two nodes `FLORENCE` in Figure 1.b. Human reasoners understand almost immediately that the relation between the first pair of nodes is "less general than" (after all, the images that one would classify as 'images of mountains in Tuscany' are a subset of the images one would classify under 'images of mountains in Italy'), while that the relation between the second pair of nodes is "equivalent" (in fact, the images that one would classify as 'images of Florence in Tuscany' are the same as the images that one would classify under 'images of Florence in Italy'). Notice that the two relations are different, even though the two pairs of HCs are structurally equivalent. Using the three semantic levels mentioned above, we can account for this difference. Consider the mapping between the two nodes `MOUNTAIN`. *Linguistic knowledge* tells us that the sense of the two labels is the same. *Domain knowledge* tells us, among other things, that Tuscany is a region of Italy. Finally, *structural knowledge* tells us that the intended meaning of the two nodes `MOUNTAIN` refers to images of mountains of Tuscany (left HC), and images of Italian mountains (right HC) respectively. All these facts together allow us to conclude that one node is less general than the other one. We can use a similar reasoning for the two nodes `FLORENCE`. But, exploiting *domain knowledge*, we can add the fact that Florence is both in Tuscany and in Italy (such a relation doesn't hold between mountains and Italy or Tuscany in the first example). This further piece of knowledge allows us to conclude that, despite structural equivalence, the relation is different.

2 CTXMATCH: the general algorithm

The general framework described in Section 1 can be used for discovering relations between any *structures labelled with natural language*. In this section, we introduce the structure and purpose independent part of the algorithm, namely the steps that do not depend on the use nor on the type of structure. This *generic algorithm* must be obviously enriched with *spe-*



(a) (b)

Figure 1: Coordinating HCs

cific structure and purpose dependent functions, i.e. with different functions for each particular type and use of the structures we want to match. In Section 3 we present the specific functions we use to match Hierarchical Classifications, i.e., tree-like structures used for classifying documents.

To make things clearer, imagine the following scenario: an agent *A* (the seeker) has a set of documents organized into a tree–structure. To collect new documents, he can send a query to a provider (an agent *B*). In our approach, the agent can formulate the query using his own structure: for example, imagine that seeker *A* uses the structure on the right-hand side of Figure 1.b to classify his documents. Then, he can select node FLORENCE to formulate the query 'Images of Florence in Italy'. Furthermore, imagine that the provider employs the left-hand structure in Figure 1.b. After receiving the query, he has the following tasks: (i) to interpret the query he receives, (ii) to find semantic relations holding between the query and his structures, and (iii) to return relevant documents (if any). In particular, in this paper we focus on the tasks (i) and (ii).

The algorithm needs two **inputs**:

query *Q***:** A seeker sends a query composed by a node fl in a structure *FS*. It means simply that the seeker wants to find nodes semantically related to the node fl in *FS*;

context *C***:** The context is composed by the three elements of the local knowledge, namely a structure *LS*, a lexicon *LL* and an ontology *LO*. The context is the target of the query[1].

The main goal of the algorithm CTXMATCH is to find the semantic relations between node fl in the query *Q* and all the nodes belonging to the local structure *LS* in the context *C*. For the sake of simplicity, in this paper we focus on the procedure for matching the node fl in the query with a single nodes ll in the context *C*. Therefore, for this simplified version of CTXMATCH, we add a third element in the input: a label ll of the structure *LS*. The **output** of the algorithm will simply be the semantic relation holding between the two nodes.

The algorithm also employs a data–type 'concept' $\langle \phi, \alpha \rangle$, constituted by a pair of logical formulas, where ϕ approximating the *individual concept* represented by a node of a structure and α expressing the relations between the current individual concept and other individual concepts in the structures (*local relevant axioms*). E.g., the formulas associated with the node labeled FLORENCE in rightmost structure in Figure 1.b will approximate the statements 'images of Florence in Italy' (the individual concept) and 'Florence is in Italy' (the local relevant axiom).

Algorithm 1 CTXMATCH(Q,C,ll)

▷ *query* $Q = \langle fl, FS \rangle$ *where* fl *is the foreign term*
FS is the foreign structure

▷ *context* $C = \langle LS, LL, LO \rangle$ *where LS is the local structure*
LL is the local lexicon
LO is the local onltology

▷ *label* ll *is the label of the local node to be matched*

VarDeclarations
context *QC*;
concept $\langle \phi, \alpha \rangle, \langle \psi, \beta \rangle$; ▷ *concepts are pairs of formulas*
relation *R*;

1 $QC \leftarrow \langle FS, LL, LO \rangle$;
▷ *QC represents the virtual query context*
2 $\langle \phi, \alpha \rangle \leftarrow$ BUILD–CXT–MEANING(fl, QC);
3 $\langle \psi, \beta \rangle \leftarrow$ BUILD–CXT–MEANING(ll, C);
▷ *compute the concepts expressed by label* ll *and* fl
4 $R \leftarrow$ SEMANTIC–COMPARISON$(\langle \phi, \alpha \rangle, \langle \psi, \beta \rangle, LO)$;
▷ *R represents the semantic relation between the two concepts*
5 **return** *R*;

In line 1, CTXMATCH first builds the 'virtual' query–context *QC*. The reason of it is that we want the query *Q* to be locally interpreted within the local lexicon and ontology. An important consequence is that the relation returned by the algorithm is **directional**: it expresses *the relation holding between the two nodes from the provider's point of view*. Indeed, the seeker could have different lexicon and ontology and could calculate different relation for the same nodes.

Then, line 2 builds a concept, i.e. a pair of logical formulas, approximating the meaning of the node fl in the virtual context *QC*. Line 3 similarly builds the concept for the node ll in the local context *C*. Finally, line 4 computes the *semantic relation* between the two concepts. The following two subsections describes in more detail this two top-level operations, implemented by the functions BUILD–CTX–MEANING and SEMANTIC–COMPARISON.

2.1 Building the contextual meaning

This step has the task of building the concept expressed by a generic node *t* in a generic context *GC*. Before analyzing the corpus of the algorithm, it's important to focus our attention on the array of senses *SynS*. A synset (set of synonyms) is a set of senses, i.e. of concepts, expressed by an expression

[1] We call context the ensemble of the three levels of knowledge because they express the local representation that an agent has of a portion of the world.

of the natural language[2]. For example the word 'Florence' has, in WORDNET, two senses (i.e. it may express two different concepts): 'city of Tuscany' and 'town of South Caroline'. The array *SynS* records these senses, so that, for example, *SynS*[Florence] is the synset containing the two senses above, while *SynS*[Florence][0] is the first of the two senses.

Let us now look at the algorithm. Line 1 determines the focus of a node t, i.e. the subgraph of the structure T useful to extract the meaning of t. This step is performed essentially for efficiency reasons, as it reduces as much as possible the node space to take into account. Lines 2-3 associate to each node within the focus the synsets found in the Lexicon. Consider the Figure 1.b: the two synsets 'city of Tuscany' and 'town of South Caroline' are associated to the label FLORENCE.

Lines 4-5 try to filter out unreasonable senses associated to t. In our example, 'town of S.C.' is discarded since it is incompatible with the other labels in the focus of t (in fact, node FLORENCE refers clearly to the city in Tuscany – see Algorithm 4).

Algorithm 2 BUILD−CTX−MEANING(GC,t)
▷ *context* $GC = \langle T, L, O\rangle$, *where* T *is a structure*
L is a lexicon
O is an onltology
▷ *label* t *is a generic label*
VarDeclaratrions
sense *SynS*[][] ▷ *array of senses*
structure F
fomula α, η
1 $F \leftarrow$ DETERMINE−FOCUS(t, T);
▷ *the focus* F *is a substructure of* T
2 **for** each label e in F **do**
3 *SynS*$[e] \leftarrow$ EXTRACT−SYNSET(e, L);
▷ *extracts the senses associated to each label in the structure* F
4 **for** each label e in F **do**
5 *SynS*$[e] \leftarrow$ FILTER−SYNSET$(F, O, SynS, e)$;
▷ *unreasonable senses are discarded*
6 $\delta \leftarrow$ INDIVIDUAL−CONCEPT$(t, SynS, F, O)$;
7 $\eta \leftarrow$ EXTRACT−LOCAL-AXIOMS$(F, SynS, O)$;
8 **return** $\langle \delta, \eta \rangle$;

Finally, lines 6 and 7 build the two component of the concept expressed by node t, computing the *individual concept* and the *local relevant axioms*, as we explained in describing Algorithm 1.

2.2 Comparing the concepts

The main task when comparing two concepts is to find the semantic relation holding between them. The algorithm employs the data–type 'deductional–pair': this is an array of pairs $\langle$*relation, formula*$\rangle$, where the *formula* expresses the condition under which the semantic *relation* between the concepts holds. E.g., the deductional–pair $\langle \equiv, \alpha \rightarrow \beta \rangle$ means that if $\alpha \rightarrow \beta$ is valid, then the relation holding between the two concepts is the equivalence ($\equiv$).

Line 1 extracts *global axioms*, i.e. the relations holding between individual concepts belonging to different structures. Consider, for example, the nodes ITALY AND TUSCANY in Figure 1.b: the global axioms express the fact that, for example, 'Tuscany is a region of Italy'. Line 2 builds the array of deductional–pair. It's important to note that the relations, their number and the associated conditions depend on the type of structure to match. In Section 3 we report the pairs relation/condition relevant for matching HCs. Lines 3–6 look for the "correct" relation holding between two concepts. This is done by checking the formulas in each deductional–pair, until a valid one is found[3]. If a valid formula is found, the associated relation is returned.

It's important to observe that the problem of finding the semantic relation between two nodes $t \in T$ and $t' \in T'$ is encoded into a satisfiability problem involving both the formulas extracted in the previous phase, and some further *global relevant axioms*. So, to prove whether the two nodes labeled FLORENCE in Figure 1.b are equivalent, we check the logical equivalence between the formulas approximating the statements 'Images of Florence in Tuscany' and 'Images of Florence in Italy' (individual concepts), given the formulas approximating the statements 'Florence is in Tuscany' and 'Florence is in Italy' (local axioms) and 'Tuscany is a region of Italy' (global axiom).

Algorithm 3 SEMANTIC−COMPARISON($\langle \phi, \alpha \rangle, \langle \psi, \beta \rangle, O$)
▷ *concept* $\langle \phi, \alpha \rangle$
▷ *concept* $\langle \psi, \beta \rangle$
▷ *ontology* O
VarDeclaratrions
formula γ
deductional-pair k[] ▷ *array of pairs* $\langle relation, formula \rangle$
1 $\gamma \leftarrow$ EXTRACT−GLOBAL−AXIOMS(ϕ, ψ, O);
2 $k \leftarrow$ BUILD−DEDUCTIONAL−FORMULAS$(\langle \phi, \alpha \rangle, \langle \psi, \beta \rangle, \gamma)$;
3 **for** each deductional-pair i in k
4 **if** SATISFIES$(\neg k[i].formula)$ **then**
5 **return** $k[i].relation$;
6 **else return** *Null*;

The three functions above constitute the *top-level algorithm*, i.e. the procedure followed to match generic structures labelled with natural language. All remaining functions (see below) are specific to the particular type of structures we need to match.

3 Semantic coordination of Hierarchical Classifications

Intuitively, a classification is a grouping of things into classes or categories. When categories are arranged into a hierarchical structure, we have a hierarchical classification. Prototypical examples of HCs are the web directories of many search engines, for example the Google™ Directory, the Yahoo!™ Directory, or the Looksmart™ web directory. In this section we show how to apply the general approach described in the previous section to the problem of coordinating HCs.

[2] See for example [3] for the use of synsets in a Lexicon.

[3] Note that a formula ϕ is valid exatcly in the case its negation $\neg\phi$ is not satisfi able.

The main algorithm is CTXMATCH, which is essentially the version of CTXMATCH where the input context contains a HC. It returns a relationship between the query node fl and the local node ll. Due to space limitation, we limited the descriptiom to the most relevant functions (see [6, 7] for a more detailed description). In the version of the algorithm presented here, we use WORDNET as a source of both lexical and domain knowledge. WORDNET could be replaced by another combination of a linguistic and domain knowledge resources[4].

HC–specific functions for BUILD-CTX-MEANING

BUILD-CTX-MEANING first needs to compute the focus of the label t and the synsets of each label in the structure. This is done by the functions DETERMINE−FOCUS and EXTRACT−SYNSET, respectively. We only give an intuitive description of these two functions.

Given a node s belonging to a structure S, DETERMINE−FOCUS has the task to reduce S to the minimal one without loosing the capability of rebuilding the meaning associated to the node s. For HC–CTXMATCH we define the focus F of a structure S given a node $s \in S$ as the smallest structure containing s and all its ancestors with their children.

EXTRACT−SYNSET associates to each node all the possible linguistic interpretations (synsets) provided by the Lexicon. In order to maximize the possibility of finding an entry into the Lexicon, we use bot a postagger and a lemmatizator over the labels.

The next function FILTER−SYNSET is applied to each node t of the focus. Its goal is to eliminate those senses associated to a node which seem to be incompatible with the meaning expressed by the node. To this end, it employs three heuristic rules, which take into account domain information provided by the ontology. This information concerns the relations between the senses associated to the node t and the senses associated to the other nodes in the focus.

Intuitively, the situation is as follows. Consider the node `FLORENCE` in the rightmost structure of Figure 1.b. The function EXTRACT−SYNSET associates to this node the two senses 'town in South Caroline' ('`florence#1`') and 'a city in central Italy' ('`florence#2`'). The structure also contains the node `ITALY`, which is an ancestor of `FLORENCE`. This node has a sense `italy#3` (namely, 'Italy the european state'), for which the relation '`italy#3` hyperonym `florence#2`' holds, meaning that 'Florence is in Italy'. Therefore, the sense '`florence#1`' can be discarded by exploiting knowledge about the sense of an ancestor node. We can then conclude that the term 'Florence' refers to the 'city in Italy' and not to the 'town in South Caroline'. The function ACCESS−ONTOLOGY allows us to discover relations between senses by traversing the ontology O (the WORDNET relations are reported in the left-hand side of Table 1).

Algorithm 4 FILTER−SYNSET$(T, O, SynS, t)$

▷ *structure T*
▷ *ontology O*
▷ *sense SynS[][] array of senses for the labels in T*
▷ *label t*

VarDeclaratrions
relation R_1, R_2, Rel_1, Rel_2 ▷ *initialized to Null*
sense $sense_{t1}, sense_{t2}, sense_y$

1 **for** each pair $sense_{t1} \neq sense_{t2}$ in $SynS[t]$ **do**
2 **for** each ancestor y of t in T **do**
3 **for** each $sense_y$ in $SynS[y]$ **do**
4 $R_1 \leftarrow$ ACCESS−ONTOLOGY$(sense_y, sense_{t1}, O)$;
5 **if** R_1 = 'hyperonymy' **then** $Rel_1 \leftarrow$ 'hyperonymy';
6 $R_2 \leftarrow$ ACCESS−ONTOLOGY$(sense_y, sense_{t2}, O)$;
7 **if** R_2 = 'hyperonymy' **then** $Rel_2 \leftarrow$ 'hyperonymy';
8 **if** $(Rel_1 = Null$ & $Rel_2 \neq Null)$ **then**
9 remove $sense_{t1}$ from $SynS[t]$;
10 $Rel_1 \leftarrow Rel_2 \leftarrow Null$;
11 **for** each pair $sense_{t1} \neq sense_{t2}$ in $SynS[t]$ **do**
12 **for** each descendant y of t in T **do**
13 **for** each $sense_y$ in $sense[y]$ **do**
14 $R_1 \leftarrow$ ACCESS−ONTOLOGY$(sense_y, sense_{t2}, O)$;
15 **if** R_1 = 'hyponymy' **then** $Rel_1 \leftarrow$ 'hyponymy';
16 $R_2 \leftarrow$ ACCESS−ONTOLOGY$(sense_y, sense_{t1}, O)$;
17 **if** R_2 = 'hyponymy' **then** $Rel_2 \leftarrow$ 'hyponymy';
18 **if** $(Rel_1 = Null$ & $Rel_2 \neq Null)$ **then**
19 remove $sense_{t1}$ from $SynS[t]$;
20 $Rel_1 = Rel_2 = Null$;
21 **for** each $sense_{t1}$ in $SynS[t]$ **do**
22 **for** each sibling y of t in T **do**
23 **for** each $sense_y$ in $SynS[y]$ **do**
24 $R_1 \leftarrow$ ACCESS−ONTOLOGY$(sense_{t1}, sense_y, O)$;
25 **if** R_1 = 'contradiction' **then** $Rel_1 \leftarrow$ 'contradiction';
26 **if** $(Rel_1 \neq Null)$ **then** remove $sense_{t1}$ from $SynS[t]$;
27 **return** SynS[t];

Lines 1–10 applies this heuristic to a sense s_n associated to a node t. Formally, it discards s_n if the following two conditions are satisfed: (i) no relation is found between this s_n and any sense associated to some ancestor, and (ii) some relation is found between a sense $s_m \neq s_n$ and some sense associated with an ancestor of t. Lines 11–20 do the same for descendants. Finally, lines 21–26 discard a sense if it is in 'contradiction' with some sense associated to a sibling of t.

The function INDIVIDUAL−CONCEPT builds a formula approximating the meaning expressed by a node t. This is done by combining the linguistic interpretation (the synsets $SynS$ associated to the nodes of the focus) with structural information (T) and domain knowledge (O), in input to the function. A critical choice is the formal language used to describe the meaning. Our implementation for HCs adopts propositional logic, whose primitive terms are the synsets of WORDNET associated to each node.

Lines 1–6 look for some ontological relation between the senses of the siblings and, if anyone is found, the interpretation of the node is refined. For example, imagine we have a node `IMAGES` with two children `EUROPE` and `ITALY`, and that the functions EXTRACT−SYNSET and FILTER−SYNSET associate to the nodes `EUROPE` and `ITALY` respectively the senses

[4] It's important to note that WORDNET is not a merged and shared structure, namely a kind of average of the structures to be matched (as in the GAV and LAV approaches). Indeed, it represents the result of linguistic mediation in centuries of use by human speakers. Using WORDNET instead of merged and shared structures, shifts the problem of sharing 'view of the world' to the more natural problem of 'sharing natural language'.

WORDNET relation	axiom
s#k synonym t#h	$s\#k \equiv t\#h$
s#k hyponym t#h	$s\#k \rightarrow t\#h$
s#k hypernym t#h	$t\#h \rightarrow s\#k$
s#k contradiction t#h	$\neg(t\#k \wedge s\#h)$

Table 1: WORDNET relations and their axioms.

europe#3 and italy#1. Since there exists an ontological relation 'europe#3 hyperonym italy#1' (Italy is in Europe) the meaning associated to node EUROPE is not longer europe#3, but it becomes europe#3 $\wedge\neg$ italy#1. In fact we imagine that a user wants to classify under node EUROPE images of Europe, and not images of Italy.

Algorithm 5 INDIVIDUAL−CONCEPT$(t, SynS, T, O)$

▷ *label t*
▷ *sense SynS[][]*
▷ *structure T*
▷ *ontology O*

VarDeclaratrions
formula $\eta = Null$
relation $R = Null, Rel = Null$
path P

1 **for** each $SynS[t][i]$ in $SynS[t][]$ **do**
2 **for** each sibling y of t in T **do**
3 **for** each $SynS[y][k]$ in $SynS[y][]$ **do**
4 $R \leftarrow$ ACCESS−ONTOLOGY$(SynS[t][i], SynS[y][k], O)$;
5 **if** R = 'hyperonymy' **then** $Rel \leftarrow$ 'hyperonymy';
6 **if** $(rel \neq Null)$ **then** replace $SynS[t][i]$ in $SynS[t][]$ with '$SynS[t][i] \wedge \neg SynS[y][k]$';
7 $P \leftarrow$ path from root to t in T; ▷ *Path from root to node t.*
8 $\eta \leftarrow \bigwedge_{e \in P} (\bigvee_i SynS[e][i])$;
9 **return** η;

Lines 7-8 compute the formula approximating the structural meaning of the concept t. This formula is the conjunction of the meanings associated to all of its ancestors (i.e., the path P). The meaning of a node is taken to be disjunction of all the (remaining) senses associated to the node. For example, if you consider the node FLORENCE in the rightmost structure of Figure 1.b, the function returns the formula (images#1 $\vee$ images#2) $\wedge$ italy#3 $\wedge$ florence#2, where (images#1 $\vee$ images#2) means that we are not able to discard anyone of the senses.

Function EXTRACT−LOCAL−AXIOMS extracts the local relevant axioms, i.e. the axioms relating concepts within a single structure. The idea is to rephrase the ontological relations between senses into logical relations. Consider again the senses florence#2 and italy#3 associated to the nodes FLORENCE and ITALY in Figure 1.b. The ontological knowledge tells us that 'italy#3 hyperoym florence#2'. This can be expressed by the axiom 'florence#2$\rightarrow$italy#3'. In HC-CTXMATCH, local axioms are built by translating WORDNET relations into formulas according to Table 1.

HC–specific functions for SEMANTIC−COMPARISON

The top–level function SEMANTIC−COMPARISON calculates the semantic relation between the formulas approximating the meaning of two nodes. In this section we describe the structural dependent functions called by this function: EXTRACT−GLOBAL−AXIOMS and BUILD−DEDUCTIONAL−FORMULAS.

EXTRACT−GLOBAL−AXIOMS works exactly as EXTRACT−LOCAL−AXIOMS. The only difference is that the axioms extracted express relations between concepts belonging to different structures. Consider for example that the two senses tuscany#1 and italy#3 have been associated respectively to nodes TUSCANY and ITALY in Figure 1.b. The ontological relation is 'italy#3 hyperonym tuscany#1', which can be expressed as 'tuscany#1 $\rightarrow$ italy#3'. The rules of translation from WORDNET senses to axioms are the same as for the function EXTRACT−LOCAL−AXIOMS.

In our approach, the problem of finding the relation between two nodes is encoded into a satisfiability problem. BUILD−DEDUCTIONAL−FORMULAS defines the satisfiability problems needed by defining (i) the set R of possible relations holding between concepts and, for each such relation $r \in R$, (ii) the formula which expresses the truth conditions for this relation. Clearly, the set R of possible relations depends on the intended use of the structures we want to map. For HC-CTXMATCH we choose the following set–theoretical relations: $\equiv$, $\subseteq$, $\supseteq$, $\perp$ ($\perp$ means that the two concepts are disjoint).

Relation	Formula
$\perp$	$(\alpha \wedge \beta \wedge \gamma) \rightarrow \neg(\phi \rightarrow \psi))$
$\equiv$	$(\alpha \wedge \beta \wedge \gamma) \rightarrow (\phi \equiv \psi))$
$\subseteq$	$(\alpha \wedge \beta \wedge \gamma) \rightarrow (\phi \rightarrow \psi))$
$\supseteq$	$(\alpha \wedge \beta \wedge \gamma) \rightarrow (\psi \rightarrow \phi))$

Table 2: The satisfiability problems for concepts $\langle \phi, \alpha \rangle$ and $\langle \psi, \beta \rangle$, with global axioms γ.

Table 2 reports the pairs ⟨relation,formula⟩ representing the satisfiability problems associated to each relation between concepts we consider, given two concepts $\langle \phi, \alpha \rangle$, $\langle \psi, \beta \rangle$, and the formula γ representing the global axioms. The result of this function is simply an array $k[]$ containing these pairs.

Consider the problem of checking whether FLORENCE in the right-hand structure in Figure 1.b is, say, equivalent to the node FLORENCE in the left-hand structure. Following are the concepts and axioms exteacted by the two strcutures:

$$\text{concept 1:} \quad \text{image\#1} \wedge \text{tuscany\#1} \wedge \text{florence\#2} \quad (1)$$
$$\text{local axiom 1:} \quad \text{florence\#2} \rightarrow \text{tuscany\#1} \quad (2)$$
$$\text{concept 2:} \quad \text{image\#1} \wedge \text{italy\#3} \wedge \text{florence\#2} \quad (3)$$
$$\text{local axiom 2:} \quad \text{florence\#2} \rightarrow \text{italy\#3} \quad (4)$$
$$\text{global axiom:} \quad \text{tuscany\#1} \rightarrow \text{italy\#3} \quad (5)$$

Checking equivalence then amounts to checking the following logical consequence $2 \wedge 4 \wedge 5 \models (1 \equiv 3)$. By the properties of propositional consequence, we can rephrase it as follows: $\models (2 \wedge 4 \wedge 5) \rightarrow (1 \equiv 3)$. It is easy to see that this latter formula is valid. So we can conclude that the relation holding between the two nodes FLORENCE is "equivalence", which is the intuitive one.

In particular, the function SATISFIES checks for the validity of a formula. In our implementation a standard SAT–solver is used for this task.

4 Testing the algorithm

In this section, we report from [5] some results of the first tests on CTXMATCH. The tests were performed on real HCs (i.e., pre-existing classifications used in real applications), and not on *ad hoc* HCs.

Matching Google with Yahoo!. We evaluated CTXMATCH over portions of Google™ and Yahoo!™ Directories looking for overlapping domains. The test was performed on the two sub-hierarchies 'Architecture' and 'Medicine' available in both Google™ and Yahoo!™. The results, measured in terms of precision and recall, are reported in the following table:

		Architecture		**Medicine**	
Relations		Pre.	Rec.	Pre.	Rec.
equivalence	$\xrightarrow{\equiv}$	.75	.08	.88	.09
less general than	$\xrightarrow{\subseteq}$	.84	.79	.86	.61
more general than	$\xrightarrow{\supseteq}$	.94	.38	.97	.35

We observe that the use of domain knowledge allowed us to discover non trivial mappings. For example, an inclusion mapping was found between `Architecture /History/Periods_and_Styles/Gothic/Gargoyles` and `Architecture/History/ Medieval` as a consequence of the relation between `Medieval` and `Gothic` provided by WORDNET. This kind of semantic mappings are very difficult to find using a keyword–based approach.

Product Re-classification. The second test was in the domain of e–commerce. In the framework of a collaboration with a worldwide telecommunication company, the matching algorithm was applied to re-classify the HC of the 'equipment and accessories' office (used to classify company suppliers) into UNSPSC[5] (version 5.0.2). We compare the results of the re-classification using CTXMATCH and the baseline matching process[6]:

	Baseline classification		**Matching classification**	
Total items	194	100%	194	100%
Rightly classified	75	39%	134	70%
Wrongly classified	91	50%	16	8%
Non classified	27	14%	42	22%

Given the 194 items re-classify, the baseline process found 1945 possible nodes, only 75 of which turned out to be correct. The baseline, a simple string-based matching method, is able to capture a certain number of re-classifications, but the percentage of error is quite high (50%) with respect to correctness (39%). With CTXMATCH the percentage of success is significantly higher (70%) and, even more relevant, the percentage of error is minimal (8%).

[5] UNSPSC (Universal Standard Products and Services Classification) is an open global coding system that classifies products and services. UNSPSC is extensively used around the world for electronic catalogs, search engines, e–procurement applications and accounting systems.

[6] The baseline has been performed by a simple keyword based matching which worked according to the following rule: for each item description (made up of one or more words) gives back the set of nodes, and their paths, which maximize the occurrences of the item words.

5 Conclusions and related work

In this paper we presented a new approach to semantic coordination in open and distributed environments. In particular we define in detail (i) a top algorithm (called CTXMATCH) for finding relations between structures labelled with natural language, and (ii) an implementation for finding set–theoretical relationships between nodes of hierarchical classifications (HCTXMATCH).

In [6, 7] we compare CTXMATCH with other proposed works, in particular with generic graph matching, CUPID [4], MOMIS [1] and GLUE [2]. We refer to this paper for related work.

References

[1] S. Bergamaschi, S. Castano, and M. Vincini. Semantic integration of semistructured and structured data sources. *SIGMOD Record*, 28(1):54–59, 1999.

[2] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of WWW-2002, Hawaii*, 2002.

[3] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. The MIT Press, Cambridge, US, 1998.

[4] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *The VLDB Journal¡*, pages 49–58, 2001.

[5] B. M. Magnini, L. Serafini, A. Doná, L. Gatti, C. Girardi, , and M. Speranza. Large–scale evaluation of context matching. Technical Report 0301–07, ITC–IRST, Trento, Italy, 2003.

[6] P. Bouquet B. Magnini, L. Serafini, and S. Zanobini. A SAT–based algorithm for context matching. In *Proc. of the 4th International and Interdisciplinary COnference on Modeling and Using Context (CONTEXT-03). Stanford University (CA), June 23-25, 2003*, volume 2680 of *Lecture Notes in Artificial Intelligence*. Springer Verlag, 2003.

[7] P. Bouquet L. Serafini and S. Zanobini. Semantic coordination: a new approach and an application. In *Proc. of the 2nd International Semantic Web Conference (ISWO'03). Sanibel Islands, Florida, USA*, October 2003.

Exploiting Ontologies to Achieve Semantic Convergence Between Different CC/PP-like RDF Schemes for Representing Devices' Capabilities: the SADiC Approach

Francesco Cannistrà
fracan@inwind.it
The-Web-Middle-Earth.com

ABSTRACT. *The CC/PP and the UAProf are two related frameworks that aim at defining a general and extensible format to describe the capabilities of the user-end terminals for accessing contents and services provided by the Internet and by the Web in particular. Both CC/PP and UAProf are based on RDF and have logically equivalent architectures. However, notwithstanding their logical bindings, they appear to be parallel standards, i.e., equivalent but not compatible. This paper explains the reasons for the incompatibility between CC/PP and UAProf and presents the approach followed by the Semantic API for the Delivery Context (SADiC) in order to achieve rigorously the required semantic convergence between these frameworks – as well as, in general, between all the CC/PP-like RDF schemes – by exploiting the concepts of the Semantic Web, without influencing the standards' bodies themselves.*

1 Introduction

Nowadays the edge population of the Internet is growing through the proliferation of heterogeneous and special purpose terminals (e.g. mobile devices) hooked up to specific network access channels (e.g. wireless networks) and offering to the users intrinsically limited service fruition capabilities. In this new scenario the users' expectation to access the services provided by the Internet (and by the Web in particular) pervasively – regardless of the specific characteristics of the device used from time to time – is fostering the service providers to engage issues regarding the device independent provision of information contents [3]. The parameters that can influence the way a user perceives and enjoys contents are many and span from the capabilities of the used device and its equipments to the constraints imposed by the network access channel, possibly including also the preferences of the user. The set of all these attributes that characterize a client fruition environment is called the *delivery context* [3, 4]. Provided with the delivery context information, the Web servers should be able to select or to adapt the output of a service for the specific requirements of the client that requested it in order to deliver a functional representation of contents that is suitable for their fruition by means of the peculiar characteristics of the *access mechanism* exploited by each user [4].

Recently the Composite Capability/Preference Profiles (CC/PP) [6, 7] – being developed by the World Wide Web Consortium (W3C) [2] – and the related User Agent Profile (UAProf) [9, 10, 11] – from the Open Mobile Alliance (OMA, formerly the WAP Forum) [8] – are emerging as standards that define a general format for expressing the delivery context information by means of profiles. They are both based on the W3C Resource Description Framework (RDF) [5] and they describe a profile as a structured set of RDF assertions. Even tough the documents specifying the CC/PP and the UAProf emphasize the need for converging the two frameworks as a shared belief, in point of fact the two working groups have proceeded almost in parallel with their respective standardization efforts so that, from a rigorous point of view, the CC/PP and UAProf now appear as equivalent but not compatible standards.

The Semantic API for the Delivery Context (SADiC) [1] acknowledges the problems actually affecting the interoperability of CC/PP with UAProf. SADiC is a Java API for processing and interrogating CC/PP and UAProf profiles. SADiC provides many features and, in particular, it succeeds in achieving rigorous semantic convergence between CC/PP and UAProf – as well as between all the RDF-based schemes implementing the basic semantics of CC/PP.

The remainder of this paper is structured in order to introduce gradually the approach of SADiC to achieve the required semantic convergence between CC/PP and UAProf. Section 2 introduces the CC/PP, focusing on its original aspects concerning the addressing of interoperability and extensibility issues. Section 3 discusses the reasons because of which UAProf is not compatible with CC/PP. A certain emphasis is given to these two sections, since it's the author's opinion that the points there discussed have not yet been taken in the right consideration by the research community. Then section 4 presents the approach of SADiC and section 5 concludes the article.

Even tough some efforts have been spent to present the contents of this paper as clearly as possible, it would be preferred that, in order for a full comprehension of the paper, the readers have, at least, a basic knowledge of RDF.

2 The CC/PP as an extensible framework providing interoperability

The CC/PP aims at defining an extensible framework as a basis for interoperability of applications that exchange delivery context information on the Internet and on the Web in particular.

Basically, the CC/PP is founded on two main ideas. Firstly, it introduces a semantic structure for representing the delivery context information by means of profiles, and provides the formal means to instance and to recognize such a structure. Secondly, it provides the formal means to define the vocabularies of attribute properties that can be used to populate the structure of a profile in order to express the specific attributes of an actual delivery context. It's in this way that CC/PP tries to address the interoperability-extensibility binomial:

1. different applications interoperate by sharing the concept of profile and the formal means to instance and to recognize this concept: profiles constructed by an application are recognizable by all others;
2. the information that can be conveyed by a profile is extensible: each application can create its own vocabulary that defines attributes useful to represent specific capabilities, and such a vocabulary can even be used by all other applications (possibly in conjunction with other vocabularies) to construct profiles.

To implement a framework with such prerogatives, CC/PP founds itself on RDF. The CC/PP does define a RDF vocabulary acting as a shared *structural vocabulary* that is the backbone of the entire conceptual framework, since it defines the RDF constructs to be used in order to instance the structure of a profile through the RDF data model, and the RDF primitives to be extended in order to define vocabularies of attributes by means of RDF schemas. This way, profiles are constructed by instancing always the same skeleton structure and then by populating this structure with actual attributes taken from different vocabularies defined by time.

A CC/PP profile can be viewed as a two-levels hierarchical structure made up of components and attributes: the attributes represent the specific capabilities of the delivery context being described, while the components group these capabilities possibly with respect to a certain global aspect (e.g., hardware or software characteristics). Figure 1 shows an excerpt of an hypothetical vocabulary defining two component types (i.e., *voc:Hardware* and *voc:Software*) and the associated attribute properties (e.g., *voc:ScreenSize* and *voc:JavaCapable*), and then shows how such a vocabulary can be exploited to build an actual profile.

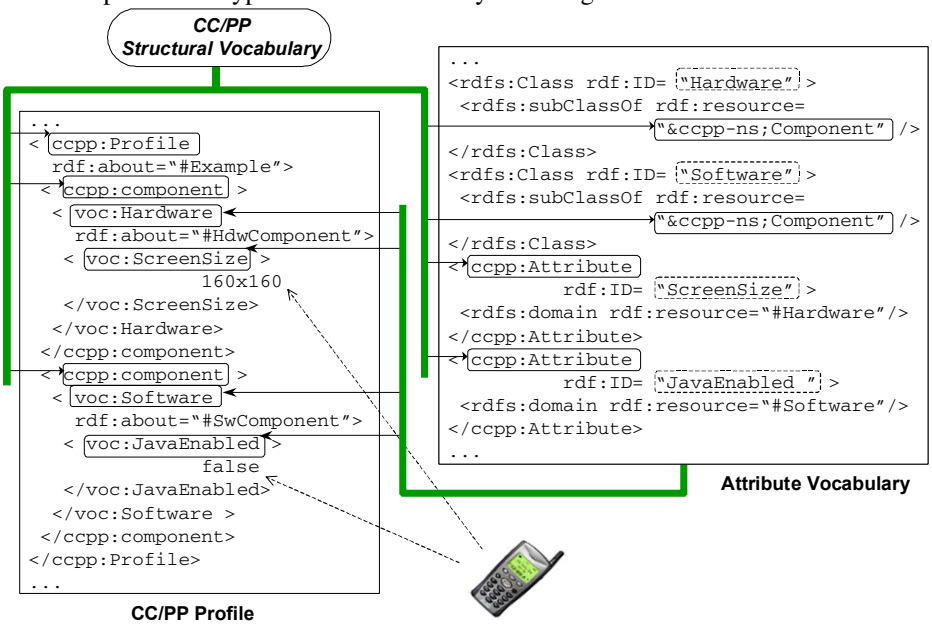


Fig. 1. A simple example showing how to define vocabularies and to construct profiles through CC/PP.

3 UAProf and its incompatibility with the CC/PP

CC/PP is vocabulary-agnostic, in the sense that it does not aim at defining any specific vocabulary of attributes that could be exploited to describe the characteristics of an actual delivery context. The CC/PP schema is just the backbone of a conceptual framework for defining vocabularies and for utilizing them in order to express the capabilities of an actual delivery context by means of an RDF description (i.e. a profile). On the contrary, the UAProf was originally invented as a specific extension of CC/PP mainly aiming at defining a rich vocabulary of attributes for constructing actual profiles. UAProf was designed to be broadly and seamlessly interoperable with the CC/PP. A precise and explicit goal of its creators was to build UAProf on the model of CC/PP as a specific implementation of it that would have also provided a vocabulary of attributes for constructing the profiles of a large range of terminals (WAP devices in particular).

Unfortunately, the development of the two frameworks has reached a status at which, if we look at them from a rigorous point of view, they can be considered only parallel, i.e., equivalent but not entirely compatible standards. The incompatibility ensues from the fact that UAProf does not rely on the RDF elements defined in the CC/PP structural vocabulary. Instead, the RDF schema introducing the UAProf vocabulary also replaces the definition of the RDF elements sustaining the CC/PP conceptual structure. The structural semantics defined by the UAProf schema is almost the same as in the CC/PP schema, but, since the RDF elements for utilizing in practice the corresponding structural concepts are tied to a different naming space, such a logical equivalence cannot be recognized at RDF level.

In fact, one of the basic principles of RDF in order to provide a language for expressing *machine-understandable* facts is that a vocabulary schema, on the one hand, explicitly introduces unequivocal terms (through URI references) – and either expresses the constraints on their use – and, on the other hand, implicitly identifies the semantics of the terms themselves. In this way, a machine, provided in advance with the knowledge base of a vocabulary and being able to recognize unambiguously the terms on which this relies, can associate terms with concepts and actual resources, and so can deduce actual relationships and meanings. The name-spacing is

just an additional facility not directly related to the RDF data model. Tying the vocabulary-defined elements to an univocal naming space helps applications to recognize groups of terms relating to the same vocabulary's context, and permits the reuse of identical (tough relative) terms within different contexts. However, since a naming space is unambiguously identified by a namespace URI that is either the shared prefixed part of all the vocabulary-defined terms, terms tied to different naming spaces denote different concepts for an RDF engine. Therefore, since both the CC/PP and UAProf are based on RDF and since the RDF elements they provide for leveraging semantically-equivalent structural concepts are defined through different RDF schemas with different namespaces, it is a consequence that they appear as different RDF applications that, although equivalent, are not compatible.

Another related problem concerns extensibility. As we have seen in section 2, the basic idea of CC/PP to achieve extensibility is that the structural vocabulary can be exploited to define whatever actual vocabulary of attributes so that, provided that the profiles' structure does not rely on any specific attribute vocabulary (but is instanced through the RDF properties defined by the CC/PP schema), a profile can be populated with actual attributes coming from different vocabularies. Moreover, if an application already defined its own vocabulary (or is using an existing one) and wants to extend this vocabulary by adding new attributes, then it should formally define a new vocabulary schema that contains the definition of the added attributes only. This way, the core vocabulary used by the application would look like a super-vocabulary made up of a set of vocabularies defined throughout subsequent schemas, and so the semantic integration between profiles that reference the different vocabularies would be assured as well.

UAProf did not acknowledge this basic idea because it intended the possibility to extend or to make corrections to the vocabulary it introduces as if each time the vocabulary schema could be completely redefined (including the basic structural concepts) by replicating it with just a few modifications and then tying the updated version to a new namespace URI. As a consequence, it was attained a situation where there exist multiple instances of the UAProf schema with different namespaces and each one of these is formally incompatible with each other for analogous reasons as those explained above when comparing the CC/PP and the UAProf in general.

It is straightforward that the problems outlined in this section are a serious hindrance to the use of the CC/PP and UAProf in wide practice and make the authoring of profiles and the development of profile processors quite cumbersome, since the risk for both profiles and processors to be not widely compliant or to became suddenly meaningless is more than concrete. However, the most important concern should be about the assurance of having wide semantic compatibility at RDF level, so that the really original prerogatives of CC/PP can be actually exploited and can then provide the intended advantages as regards interoperability and extensibility. In fact, if vocabularies and profiles were created basing on always different RDF schemes that, although intended to rely on the conceptual structure of the CC/PP, are not compatible with this structure at RDF level, then the advisability itself to have built CC/PP on top of RDF would not make sense any more.

In summary, the RDF heterogeneity between CC/PP and UAProf is paradoxically leading the Web accessibility towards a vertical segmentation as depicted in figure 2. A restatement of UAProf that would obey more to the basic interoperability principles of the CC/PP and of RDF would certainly improve the situation and would be an auspicial as well. However, this would not solve the problems at all, since the industry manufactures have already started to use UAProf and, at the moment, virtually all the CC/PP enabled devices use UAProf, provided that it also defines a rich vocabulary of actual attributes that can be utilized in practice to express device capabilities. Therefore, a general and rigorous approach that would assure of formal semantic interoperability without affecting the state of the art with the standards is now, of course, required.

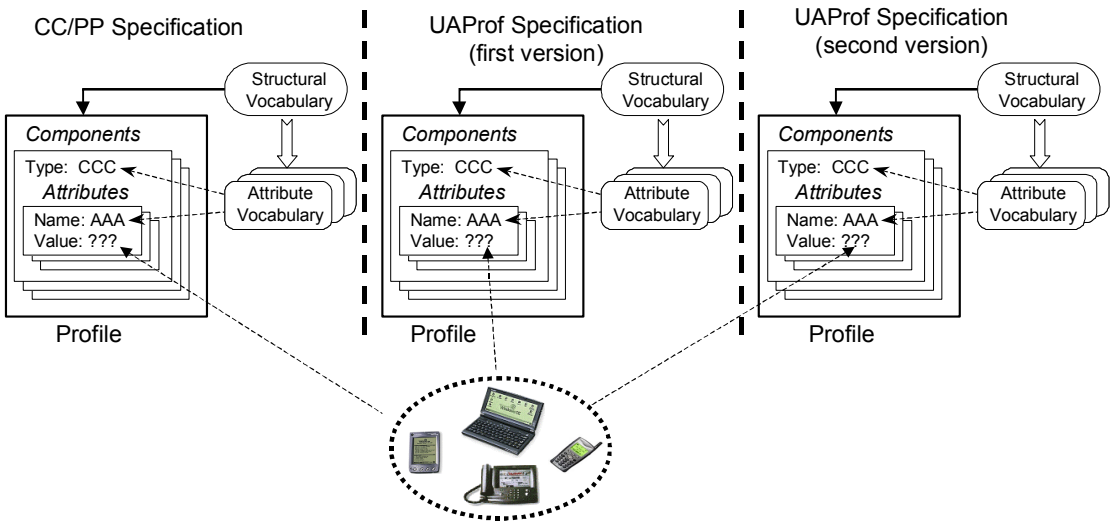


Fig. 2. The vertical segmentation of the Web accessibility ensuing from the lack of interoperability.

4 The approach of SADiC for the semantic integration of CC/PP and UAProf

Provided with the full understandings of the interoperability problems pointed out in the previous section, the Semantic API for the Delivery Context (SADiC) has been designed also to achieve rigorous semantic integration between CC/PP and UAProf – and, in general, between all the RDF-based schemes that provide a parallel implementation of the CC/PP conceptual framework – so that all these schemes can be used concurrently or jointly in wide practice, being assured of semantic interoperability without the need for any particular effort. For this aim, SADiC exploits the concepts of the Semantic Web [12] and, in particular, the notion of *ontology*.

Basically, an ontology is a collection of axioms that describe computer-usable concepts – and either introduce the vocabulary of terms that relate to them – in the perspective of representing a *domain* (i.e., an area of knowledge) for *machine-understanding* purposes. The basic idea introduced by SADiC exploits the fact that the knowledge encoded by an ontology can be imported and reused by other domains. In this way, it is possible to build complex domains that grow each over each other and that represent different levels of abstraction of the same knowledge base, possibly specializing and/or extending and/or enhancing this for a particular application purpose.

SADiC defines and relies on a core ontology that expresses the abstract knowledge base required to build an RDF-based conceptual framework implementing the basic semantics of the CC/PP architecture. The elements defined by this ontology represent the semantic abstraction of the basic structural concepts introduced by the CC/PP – e.g., the abstract concept of *attribute property* (i.e., the class of RFD properties that express the delivery context attributes), the abstract concepts of *structural properties* (i.e., the RDF properties that let instance the semantic structure of a profile within an RDF data model) and the concept of profile component (i.e., the *Component* class that acts as the root component type for all profile components).

The key aspect of the core ontology is that it does not supersede the CC/PP structural vocabulary. The core ontology just represents the lowest level of abstraction of the logical domains corresponding to all the CC/PP-like conceptual frameworks, and houses formally the shared semantics of the basic structural concepts already introduced by the CC/PP specification, not the terms that are to be used to exploit these concepts in practice. An actual domain can import the basic concepts of the core ontology and map them to its own terms. Such a domain is intended as a structural domain, since it provides an effective naming space for the CC/PP concepts and allows to utilize them through the specific terms it defines. Therefore, many lexically-different but semantically-equivalent domains can be introduced: these all exploit the same semantics of the CC/PP structural concepts, but allow to refer to them through different terms afferent to different namespaces.

Let's consider, for example, the case of the structural vocabulary proposed by the CC/PP specification and the vocabularies corresponding to the various version of the UAProf specification. Within SADiC all these vocabularies are associated with domains that just host suitable terms to refer to the shared semantics of the CC/PP structural concepts, but that do not define the concepts themselves. Since these concepts are defined elsewhere – i.e. in the core abstract ontology – and the different terms through which they can be referenced are formally mapped to them, then the wished semantic convergence and cross-interoperability are achieved automatically and rigorously.

Figure 3 sketches a simplified view of the semantic hierarchy introduced by SADiC. Note that at the leaf level there are the pure application domains, i.e. the domains corresponding to the actual vocabularies of attributes for describing a delivery context, which are defined through RDF schemas that reference and utilize a structural vocabulary associated with a specific CC/PP structural domain.

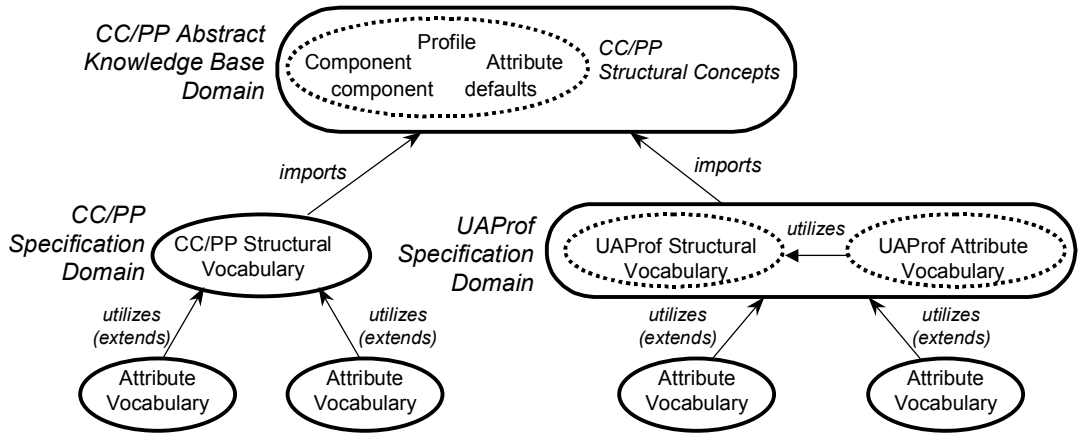


Fig. 3. The semantic hierarchy introduced by SADiC.

```
<owl:Ontology rdf:about="">
  <owl:sameAs rdf:resource="&ccpp-spec-schema-URI;"/>
</owl:Ontology>
<owl:Ontology rdf:about="&ccpp-spec-schema-URI;">
  <owl:imports rdf:resource="&ccpp-abs-ontology-URI;"/>
</owl:Ontology>
<owl:Class rdf:about="&ccpp-spec-schema-URI;#Component" >
  <owl:sameAs rdf:resource="&ccpp-abs-ontology-URI;#Component"/>
</owl:Class>
<owl:Class rdf:about="&ccpp-spec-schema-URI;#Attribute">
  <owl:sameAs rdf:resource=&ccpp-abs-ontology-URI;#Attribute"/>
</owl:Class>
<rdf:Property rdf:about="&ccpp-spec-schema-URI;#component">
  <owl:sameAs rdf:resource=&ccpp-abs-ontology-URI;#component"/>
</rdf:Property>
<rdf:Property rdf:about="&ccpp-spec-schema-URI;#defaults">
  <owl:sameAs rdf:resource=&ccpp-abs-ontology-URI;#defaults"/>
</rdf:Property>
```

Fig. 4. An excerpt of the OWL ontology for the structural domain of the CC/PP specification.

In order to define a domain, SADiC makes use of the Web Ontology Language (OWL) [14, 15, 16], the language for representing ontologies on the Web. OWL is based on RDF and is still being developed by the W3C as a component of the Semantic Web Activity [13]. SADiC requires that only the structural domains are to be explicitly defined: the pure application domains are defined implicitly by the corresponding RDF vocabulary schemas (provided that these schemas extend correctly the RDF schema defining the structural vocabulary of an already recognized structural domain). A structural domain is defined by means of a simple OWL ontology that expresses the basic facts that semantically make of such a domain a CC/PP structural domain. For this goal, it is sufficient to state that the terms introduced by an RDF schema (within its own naming space) to refer to the semantics of the CC/PP structural elements have the same *intentional meaning* as the concepts defined in the core abstract ontology – i.e., both RDF elements, though denoted by different terms, are semantically equivalent.

The listing in figure 4 shows a fragment of the OWL ontology introducing the domain corresponding to the structural vocabulary schema defined by the CC/PP specification.

This way, SADiC succeeds in mapping the semantics between structural vocabularies corresponding to different RDF-based schemes that provide a parallel implementation of the CC/PP architecture, and, therefore, the potential vertical segmentation depicted in the previous section is brilliantly avoided within a purely semantic context (see figure 5).

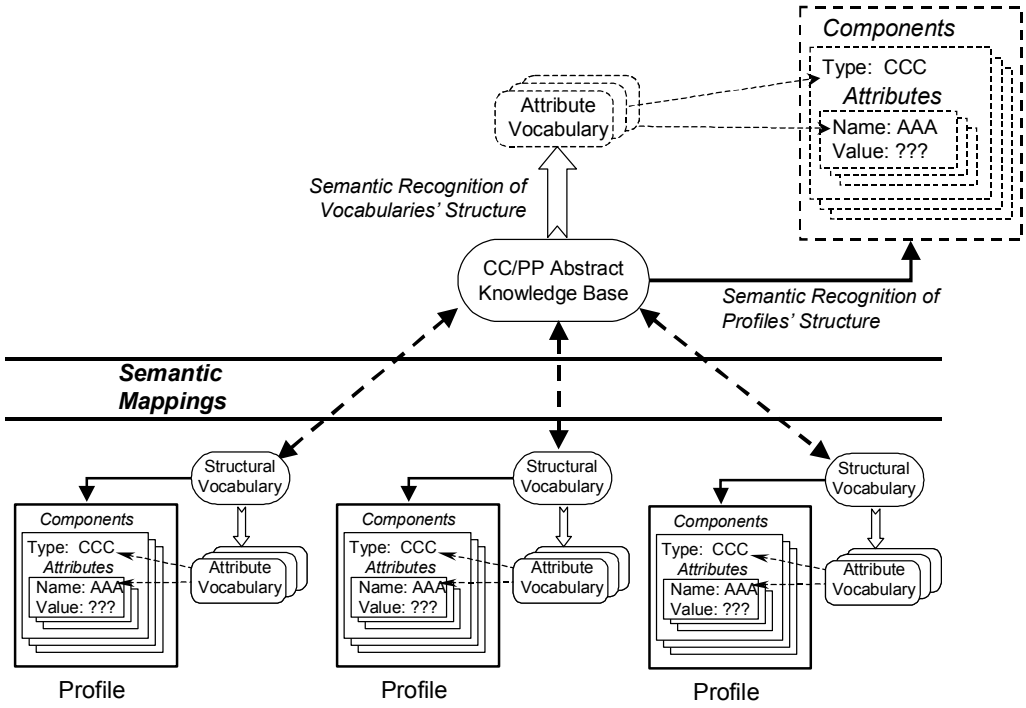


Fig. 5. How SADiC achieves interoperability.

Further to overcoming the RDF incompatibilities at structural level, SADiC also addresses problems related to the multi-versioning of attribute vocabularies. In particular, the ontology for specifying a domain can be exploited to assert that two RDF schemas – tied to different namespaces – are semantically equivalent or are subsequent versions of the same logical vocabulary. In such cases, SADiC is able to manage properly this kind of equivalence so that, for example, segmental profiles constructed relying on different versions of the same logical vocabulary can be merged together consistently.

As we have seen in section 3, the proliferation of multiple namespace URIs to refer to the same logical vocabulary is an incongruity actually affecting UAProf. In particular, there are two kinds of slightly different problems: the referencing of the vocabulary schema tied to a certain version of UAProf through different namespace URIs, and the extending the UAProf vocabulary through new RDF schemas that completely supersede the older ones (and have different namespaces either). To address the latter problem, it is sufficient to assert, through the OWL ontology defining the domain corresponding to a certain version of UAProf, that the RDF schema associated with such a version is the subsequent version of an earlier RDF schema:

```
<owl:Ontology rdf:about="&uaprof;">
  <owl:backwordCompatibleWith
     rdf:resource="&uaprof-previous;"/>
</owl:Ontology>
```

Note that the above statements also indicate that all the local terms tied to the previous naming space have the same intended interpretations in the naming space of the new version.

Instead, in order to assert that the defined domain could even be referenced through a namespace URI different from the canonical namespace URI of the considered UAProf version (and that acts as an alias for this), an *owl:backwordCompatibleWith* statement should be coupled with an *owl:sameAs* statement:

```
<owl:Ontology rdf:about="&uaprof-alias;">
  <owl:backwordCompatibleWith
     rdf:resource="&uaprof;"/>
  <owl:sameAs rdf:resource="&uaprof;"/>
</owl:Ontology>
```

5 Conclusions

This paper has discussed of the cumbersome problems actually thwarting the cross-compatibility between the CC/PP and the UAProf frameworks, and of the danger of attaining a vertical segmentation of the Web accessibility that would be quite the contrary of the original goals of the CC/PP. The paper has introduced the Semantic API for the Delivery Context (SADiC), showing how this approaches the abovementioned problems and succeeds in achieving the required formal semantic convergence between CC/PP and UAProf – as well as between all the RDF-based schemes that are intended to rely on a CC/PP-like conceptual architecture. The approach of SADiC exploits the notion of ontology in order to build an extensible hierarchy of semantically overlapping RDF domains, and uses the Web Ontology Language (OWL) in order to represent a domain and to map semantics between domains.

Even though SADiC focuses on a specific application context, its semantic approach introduces simple and general ideas that could be exploited in order to address analogous issues of semantic interoperability for other RDF-based contexts as well.

References

1. SADiC: the Semantic API for the Delivery Context, *http://www.the-web-middle-earth.com/sadic/*
2. World Wide Web Consortium (W3C), *http://www.w3.org*
3. Device Independence Activity (DIA) at W3C, *http://www.w3.org/2001/di/*
4. R. Gimson et al., "Device Independence Principles". W3C Note, Version 1 September 2003 available at: *http://www.w3.org/TR/2003/NOTE-di-princ-20030901/*
5. W3C Resource Description Framework (RDF), *http://www.w3.org/RDF/*
6. Early CC/PP Working Group at W3C, *http://www.w3.org/Mobile/CCPP/*
7. G. Klyne, F. Reynolds, C. Woodrow, H. Ohto, J. Hjelm, M. Butler, L. Tran et al., "Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies". W3C Working Draft, Version 28 July 2003 available at: *http://www.w3.org/TR/2003/WD-CCPP-struct-vocab-20030728/*
8. Open Mobile Alliance (OMA), *http://www.openmobilealliance.org*
9. WAP Forum, User Agent Profiling Specification (WAP-174-UAPROF-19991110-a). November 1999, available from: *http://www.wapforum.org/what/technical_1_2_1.htm* (See also subsequent WAP SINs)
10. WAP Forum, User Agent Profiling Specification (WAP-248-UAPROF-20011020-a). October 2001, available from: *http://www.wapforum.org/what/technical.htm* (See also subsequent WAP SINs)
11. OMA, User Agent Profile Version 1.1. December 2002, available from: *http://www.openmobilealliance.org/documents.html*
12. T. Berners-Lee, J. Hendler, and O. Lassila, "The semantic web". Scientific American, May 2001.
13. Semantic Web Activity at W3C, *http://www.w3.org/2001/sw/*
14. Web Ontology (WebOnt) Working Group at W3C, *http://www.w3.org/2001/sw/WebOnt/*
15. D. L. McGuinness, F. van Harmelen et al., "OWL Web Ontology Language Overview". W3C Candidate Recommendation, Version 18 August 2003 available at: *http://www.w3.org/TR/2003/CR-owl-features-20030818/*
16. S. Bechhofer, F. van Harmelen, J. Hendler, I. Horrocks, D. L. McGuinness, P. F. Patel-Schneider, L.A. Stein et al., "OWL Web Ontology Language Reference". W3C Candidate Recommendation, Version 18 August 2003 available at: *http://www.w3.org/TR/2003/CR-owl-ref-20030818/*
17. J. Heflin et al., "Web Ontology Language (OWL) Use Cases and Requirements". W3C Candidate Recommendation, Version 18 August 2003 available at: *http://www.w3.org/TR/2003/CR-webont-req-20030818/*

An integrative proximity measure for ontology alignment*

Jérôme Euzenat
INRIA Rhône-Alpes
Jerome.Euzenat@inrialpes.fr

Petko Valtchev
Université de Montréal
Petko.Valtchev@umontreal.ca

Abstract

Integrating heterogeneous resources of the web will require finding agreement between the underlying ontologies. A variety of methods from the literature may be used for this task, basically they perform pair-wise comparison of entities from each of the ontologies and select the most similar pairs. We introduce a similarity measure that takes advantage of most of the features of OWL-Lite ontologies and integrates many ontology comparison techniques in a common framework. Moreover, we put forth a computation technique to deal with one-to-many relations and circularities in the similarity definitions.

1 The ontology alignment problem

Like the Web, the semantic Web will necessarily be distributed and heterogeneous. Therefore, the integration of resources found on the semantic Web is a key issue. A standard approach to the resulting problem lies in the use of ontologies for data description. However, the available ontologies could themselves introduce heterogeneity: given two ontologies, the same entity can be given different names in each of them or simply be defined in different ways, whereas both ontologies may express the same knowledge but in different languages.

Semantic interoperability can be grounded in ontology reconciliation. The underlying problem, which we call the "ontology alignment" problem, can be described as follows: given two ontologies each describing a set of discrete entities (which can be classes, properties, rules, predicates, etc.), find the relationships (e.g., equivalence or subsumption) that hold between these entities. Alignment results can be used for various purposes such as displaying the correspondences, transforming one source into another or creating a set of bridge axioms between the ontologies. An overview of alignment methods is presented in §2.

The present paper focuses on automatic and autonomous ontology alignment, although more interactive scenarios may be built on top of the proposed technique (e.g., complete a partial alignment or use the result as a suggestion to the user). It will be also assumed that the ontologies are described within the same knowledge representation language: OWL-Lite (§3).

The language is based on various features: classes and subsumption, properties and type constraints, etc., and the goal of this paper is to define a similarity measure that encompasses all those features (§4.1) while overcoming major alignment problems such as circularities (§4.2) and the presence of external data types. Our approach is based on previous work on object-based knowledge representation similarity which is here adapted to the current web languages. Interested readers are refered to [14] for a detailed discussion of the proposed measure.

2 Alignment methods

There has been important background work that can be used for ontology alignment: in discrete mathematics for matching graphs and trees [7], in databases for reconciling and merging schemas [11], in machine learning for clustering compound objects described in a restricted FOL [1].

Basically, aligning amounts at defining a pair-wise distance between entities (which can be as reduced as an equality predicate) and computing the best match between them,

*This work has been partially supported by grants from the French consulate in Montréal and the Centre Jacques Cartier. We thank an anonymous reviewer for interesting critical remarks that have helped improving this presentation.

i.e., the one that minimizes the total distance (or maximizes a similarity measure). But there are many different ways to compute such a distance. Roughly speaking, they can be classified as (this complements the taxonomy provided in [11] and only consider features found in actual systems):

terminological (T) comparing the labels of the entities; **string-based (TS)** does the terminological matching through string structure dissimilarity (e.g., editing distance); **terminological with lexicons (TL)** does the terminological matching modulo the relationships found in a lexicon (i.e., considering synonym as equivalent and hyponyms as subsumed);

internal structure comparison (I) comparing the internal structure of entities (e.g., the value range or cardinality of their attributes);

external structure comparison (S) comparing the relations of the entities with other entities; **taxonomical structure (ST)** comparing the position of the entities within a taxonomy; **external structure comparison with cycles (SC)** an external structure comparison robust to cycles;

extensional comparison (E) comparing the known extension of entities, i.e. the set of other entities that are attached to them (in general instances of classes);

semantic comparison (M) comparing the interpretations (or more exactly the models of the entities).

Some contributions can be found in Table 1, we only provide some salient points for each of them: [3] matches conceptual graphs using terminological linguistic techniques and comparing superclasses and subclasses. [12] computes the dissimilarity between two taxonomies by comparing for each class the labels of their superclasses and subclasses. FCA-Merge [13] uses formal concept analysis techniques to merge two ontologies sharing the same set of instances while properties of classes are ignored. Anchor-Prompt [10] uses a bounded path comparison algorithm with the originality that anchor points can be provided by the users as a partial alignment. Cupid [8] is a first approach combining many of the other techniques. It aligns acyclic structures taking into account terminology and data types (internal structure) and giving more importance to leaves. [9] creates a graph whose nodes are candidate aligned pairs and arcs are shared properties. Arcs are weighted by their relevance to the nodes and similarity values are propagated through this graph until a fixed point is reached. T-tree [5] infers dependencies between classes (bridges) of different ontologies sharing the same set of instances based only on the "extension" of classes. Semantic similarity is comparable to the work on subsumption in description logics. In addition, a number of other systems use machine learning techniques for finding class similarity from instances [4].

Many of these algorithm use various techniques for finding a alignment, though they still neglect some aspects of the ontology definitions. Moreover, they are not often robust to cycles in definitions, e.g., the fixed-point computation in [9] is not proven to converge. Our goal is to design a measure that integrates all aspects of OWL-Lite and can deal with cyclic definitions.

3 Ontology representation

For that purpose, we will first exhibit a representation for OWL-Lite ontologies (§3.1) that emphasises entities and their relationships (§3.2).

3.1 The web ontology language OWL

OWL [2] is a language for expressing ontologies on the web. Due to space restrictions, we only present here the ontology constructors proposed by the language (the reader can find elsewhere more information on their semantics). OWL can be thought of as a description logic embedded in a frame-like syntax. It comes in three flavors: OWL-Lite, OWL-DL, and OWL-Full. We concentrate on OWL-Lite which is sufficient for many purposes while creating various difficulties for alignment algorithms.

OWL-Lite is an extension of RDF which allows the definition of individuals as instances of a class and the expression of relations between individuals. Additionally[1], OWL-Lite:

- uses RDF Schema keywords (`rdfs:subClassOf`, `rdfs:Property`, `rdfs:subPropertyOf`, `rdfs:range`, `rdfs:domain`) for defining taxonomies of classes and properties and restricting the range of properties;

[1] We do not present all the constructors, some of them can be easily defined from the others. E.g., `owl:sameClassAs` can be defined through reciprocal `rdfs:subClassOf` assertions. Any semantically grounded measure should be able to account for these equivalences.

Reference	T	TS	TL	I	S	ST	SC	E	M
Dieng & Hug [3]	x		x			x			
Staab & Mädche [12]		x			x	x			
FCA-Merge [13]						x			x
Anchor Prompt [10]	x			x	x	x			
Cupid [8]	x	x		x	x				
Similarity flooding [9]		x			x		x		
T-tree [5]					x			x	

Table 1: Various contributions to alignment at a glance.

- allows the definition of a class (`owl:Class`) as more specific or equivalent to the intersection of other classes;
- allows the assertion of equality (`owl:sameAs`) or difference (`owl:differentFrom`) between two individuals;
- characterizes properties as transitive (`owl:TransitiveProperty`), symmetric (`owl:SymmetricProperty`) or inverse of another property (`owl:inverseOf`);
- can restrict the range of a property in a class to be another class (`owl:allValuesFrom`) or assert that some objects of a particular class must be in the property (`owl:someValuesFrom`).
- can restrict the number of object in a particular relation with another one through the use of cardinality constraints (`owl:minCardinality` and `owl:maxCardinality`). In OWL-Lite, these constraints can only take values 0, 1, or infinite.

OWL makes use of external data types. In particular it relies on the XML Schema data types without having to know them.

3.2 Representation

Instead of computing similarity on an OWL-Lite syntax, it will be computed on a corresponding graph based syntax. Such a graph will contain several types of nodes: class (C), object (O), relation (R), property (P), property instance (A), datatype (D), datavalue (V), property restriction labels (L). These nodes are linked by various kinds of relationships:

- `rdfs:subClassOf` between two classes or two properties ($\mathcal{S}$);
- `rdf:type` ($\mathcal{I}$) between objects and classes, property instances and properties, values and datatypes;
- $\mathcal{A}$ between classes and properties, objects and property instances;
- `owl:Restriction` ($\mathcal{R}$) expressing the restriction on a property in a class;
- valuation ($\mathcal{U}$) of a property in an individual

The relation symbols will be used as set-valued fonctions ($\mathcal{F}(x) = \{x; \exists y; \langle x, y\rangle \in \mathcal{F}\}$). Additionaly, each node is identified ($\lambda : C \cup O \cup R \cup P \cup D \cup A \longrightarrow URIRef$) by a URI reference and can be attached annotations.

Finally, to provide the most complete basis for comparison, one may wish to bring knowledge encoded in relation types to the object level. This could be done by adding some edges between objects that are reverse, symmetric or transitive for an existing edge or a pair of edges. Relation types can be handled by saturation of the graph or in a lazy way: for `owl:TransitiveProperty` by adding transitivity arcs; for `owl:SymmetricProperty` by adding symmetric arcs; for `owl:inverseOf` by adding the reverse arcs (both in generic and individual descriptions); for `owl:FunctionnalProperty` by adding a cardinality constraint; `owl:InverseFunctionnalProperty` is not accounted for at that stage.

4 Principles of similarity

Alignment amounts at finding the best correspondance between entities of two ontologies. This requires the definition of a similarity on entity pairs (§4.1). Since relation-

ships between entities constitute a major part of the ontological knowledge, a sensible similarity measure must process them suitably, in particular, by comparing two entities with respect to the sets of "surrounding" entities in the corresponding ontologies. Consequently, relationships entail dependencies between similarity values which further require an effective computation mechanism to avoid the pitfalls of circularity (§4.2).

4.1 Similarity measure

The graphic representation chosen for OWL-Lite highlights the various categories of entities, of links between entities and of descriptive features for entities. The target correspondence between two ontologies maps entities from one ontology to the most similar entities of the other one, a principle that is based on a dedicated similarity measure. We choose to use a similarity measure for ease of explaination. A dual disimilarity can be obtained by an easy transformation. The measure ranks a pair of entities to a real number in $[0\ 1]$ whereby 0 (1) stands for completely different (similar) entities. It is based on two key assumptions:

- all the components of an entity category are *a priori* relevant for similarity assessment, although their relative importance can be tuned through weights. This is backed by most of the techniques used for ontology alignment (see §2);
- the entities within each category are dealt with in the same way, but comparison means for different categories may diverge.

In summary, the approach followed here consists in assigning each entity category, e.g., a class, a specific measure which is defined as a function of the results computed on the related entity categories, e.g., a property, a sub-class, etc., by the respective measures. We choose to aggregate the various components through a weighted sum. Some other aggregation operators could be used but at the expense of the difficulty to find a solution. Weights allow to tune the importance of a component in the similarity whereby a zero weight amounts to completely ignoring the component. E.g., for two classes classes c, c' :

$$\begin{aligned} Sim_C(c,c') = \pi_L^C sim_L(\lambda(c),\lambda(c')) \\ + \pi_O^C MSim_O(\mathcal{I}(c),\mathcal{I}'(c')) \\ + \pi_S^C MSim_C(\mathcal{S}(c),\mathcal{S}'(c')) \\ + \pi_P^C MSim_P(\mathcal{A}(c),\mathcal{A}'(c')) \end{aligned}$$

The similarity is normalised: the sum of all weights is 1, i.e., $\pi_L^C + \pi_S^C + \pi_O^C + \pi_P^C = 1$, whereas set similarities ($MSim$) are basically averages of components similarities, as illustrated by the measure for super-class sets:

$$MSim_C(S,S') = \frac{\sum_{\langle c,c'\rangle \in Pairing(S,S')} Sim_C(c,c')}{max(|S|,|S'|)}$$

Here $Pairing(S,S')$ is a mapping of element of S to elements of S' which maximises the $MSim_C$ similarity. Thus, the similarity between the sets is the average of the values on matched pairs (see definition in [14]). Table 2 lists all the defined measures.

The target similarity values ultimately depend on the similarities between data types, values and URIRef and the way these are propagated through the relationships in the graphs. Measures for data types and values should be provided together with an abstract data type definition, URIRef can be compared by an equality predicate or by a string similarity applied to suffixes.

4.2 Computing similarities

One may notice from the above example that $Sim_C(c,c')$ depends on the result of Sim_C on other classes, both through specialization and properties. In the second case, the dependancy may easily lead to a "deadlock" where $Sim_C(c_1,c_2)$ depends on $Sim_C(c_3,c_4)$ and *vice versa*. Consequently, similarities can only be expressed as equations. More precisely, a system is composed in which a variable corresponds to an entity pair whereas an equation is drawn from the definition of that pair similarity, namely by substituting all similarity occurences by the corresponding variables:

$$\begin{cases} x_{1,1} = Sim_C(c_1,c'_1) & y_{1,1} = Sim_P(p_1,p'_1) \\ x_{1,2} = Sim_C(c_1,c'_2) & y_{1,2} = Sim_P(p_1,p'_2) \\ \ldots \end{cases}$$

Function	Node	Factor	Measure
Sim_O	$o \in O$	$\lambda(o)$	sim_L
		$a \in A$, $(o, a) \in \mathcal{A}$	$MSim_A$
Sim_A	$a \in A$	$r \in R$, $(a, r) \in \mathcal{R}$	Sim_R
		$b \in O \cup V$	$MSim_V / MSim_O$
Sim_V	$v \in V$	value literal	type dependent
Sim_C	$c \in C$	$\lambda(c)$	sim_L
		$p \in P$, $(c, p) \in \mathcal{R}$	$MSim_P$
		$c' \in C$, $(c, c') \in \mathcal{S}$	$MSim_C$
sim_D	$d \in D$	$\lambda(r)$	XML-Schema specific
Sim_R	$r \in R$	$\lambda(r)$	sim_L
		$c \in C$, $(r, \texttt{domain}, c) \in \mathcal{R}$	$MSim_C$
		$c \in C$, $(r, \texttt{range}, c) \in \mathcal{R}$	$MSim_C$
		$d \in D$, $(r, \texttt{range}, d) \in \mathcal{R}$	Sim_D
		$r' \in R$, $(r, r') \in \mathcal{S}$	$MSim_R$
Sim_P	$p \in P$	$r \in R$, $(p, r') \in \mathcal{S}$	Sim_R
		$c \in C$, $(p, \texttt{allValuesFrom}, c) \in \mathcal{R}$	$MSim_C$
		$n \in \{0, 1, +\infty\}$, $(p, \texttt{cardinality}, n) \in \mathcal{R}$	equality

Table 2: Similarity function decompositon.

In case some similarity values (or some similarity or dissimilarity assertions) are provided as an input to the program, the corresponding equation can be replaced by the assertion of the similarity between the objects.

If each of the $MSim$ were deterministic (only one entity is compared to another), this system would be solvable directly because all variables are of degree one. However, in the case of OWL-Lite, the system is not linear since there could be many candidate pairs for the best match. Nevertheless, the resolution of the resulting system can still be carried out as an iterative process that simulates the computation of the fixed point of a vector function, as shown by Bisson [1]. The trick consists in defining an approximation of the $MSim$-measures, solving the system, replacing the approximations by the newly computed solutions and iterating. The first values for these $MSim$-measures are the maximum similarity found for a pair, without considering the dependent part of the equations. The subsequent values are those of the complete similarity formula filled by the solutions of the system. The system converges: the similarities cannot decrease between steps – in an equation, the "ground" part remains steady while dependencies may only propagate their own increase – and the similarity is bounded by 1 – no variable value can exceed 1 since none of its components can (inductively). The process halts when none of the values increases by more than ϵ with respect ot the previous iteration. The algorithm may well converge to a local optimum, i.e., a different matching in one equation may, at least theoretically, lead to a different global solution. A solution could lay in a random change of some matchings.

The result of the process is an approximation of the similarity between entities from opposite ontologies. The ultimate alignment goal is a satisfactory mapping between ontologies which uses the similarity values as a basis for the ranking of entity pairs.

5 Conclusion

In order to be able to align ontologies written in OWL-Lite, we adapted a method developed for measuring object-based similarity to OWL-Lite. This method has the benefit of considering many of the features of ontology descriptions in computing the alignment: it deals successfully with external data types, internal structure of classes as given by their properties and constraints, external structure of classes as

given by their relationships to other classes and the availability of individuals.

This is an improvement towards other methods that take advantage of only a subpart of the language features. The proposed methods does not only compose linearly individual methods for assessing the similarity between entities, it uses an integrated similarity definition that make them interact during computation. Moreover, it copes with the unavoidable circularities that occur within ontologies.

Yet, this measure does not cover all syntactic constructions of OWL-Lite (e.g., `owl:AllDifferent`, `owl:InverseFunctionnalProperty`). A more complete description of this similarity measure is in preparation [6]. We also plan to neatly integrate some features of OWL-DL (e.g., `owl:oneOf`). Moreover, thorough tests of our measure must be performed to find weights and external similarity measures that provide satisfactory results.

The proposed similarity measure is not semantically justified, but exhibits good features such as not imposing injective mapping. However, we would like it to be at least syntax-independent for OWL-Lite. To that extent, we must ensure that whatever the description of two entities, if they are semantically equivalent, they behave identically with respect to the similarity measure. This will amounts to either normalising the graph (adding `owl:minCardinality` constraints for each `owl:someValueFrom` for instance) or comparing heterogeneous components.

References

[1] Gilles Bisson. Learning in FOL with similarity measure. In *Proc. 10th American Association for Artificial Intelligence conference, San-Jose (CA US)*, pages 82–87, 1992.

[2] Mike Dean and Guus Schreiber (eds.). OWL web ontology language: reference. Working draft, W3C, 2003. http://www.w3.org/TR/owl-ref/.

[3] Rose Dieng and Stefan Hug. Comparison of "personal ontologies" represented through conceptual graphs. In *Proc. 13th ECAI, Brighton (UK)*, pages 341–345, 1998.

[4] An-Hai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Ontology matching: A machine learning approach. In Steffen Staab and Rudi Studer, editors, *Handbook on Ontologies in Information Systems*, pages 397–416. Springer-Verlag, Heildelberg (DE), 2003.

[5] Jérôme Euzenat. Brief overview of T-tree: the Tropes taxonomy building tool. In *Proc. 4th ASIS SIG/CR workshop on classification research, Columbus (OH US)*, pages 69–87, 1994. ftp://ftp.inrialpes.fr/pub/sherpa/publications/euzenat93c.ps.gz.

[6] Jérôme Euzenat and Petko Valtchev. Alignment in OWL-Lite, 2003. in preparation.

[7] John Hopcroft and Robert Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4):225–231, 1973.

[8] Jayant Madhavan, Philip Bernstein, and Erhard Rahm. Generic schema matching using Cupid. In *Proc. 27th VLDB, Roma (IT)*, pages 48–58, 2001. http://research.microsoft.com/ philbe/CupidVLDB01.pdf.

[9] Sergey Melnik, Hector Garcia-Molina, and Erhard Rahm. Similarity flooding: a versatile graph matching algorithm. In *Proc. 18th International Conference on Data Engineering (ICDE), San Jose (CA US)*, 2002.

[10] Natalya Noy and Mark Musen. Anchor-PROMPT: Using non-local context for semantic matching. In *Proc. IJCAI 2001 workshop on ontology and information sharing, Seattle (WA US)*, pages 63–70, 2001. http://sunsite.informatik.rwth-aachen.de/Publications/CEUR-WS/Vol-47/.

[11] Erhard Rahm and Philip Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[12] Steffen Staab and Alexander Mädche. Measuring similarity between ontologies. *Lecture notes in artificial intelligence*, 2473:251–263, 2002.

[13] Gerd Stumme and Alexander Mädche. FCA-merge: bottom-up merging of ontologies. In *Proc. 17th IJCAI, Seattle (WA US)*, pages 225–230, 2001.

[14] Petko Valtchev. *Construction automatique de taxonomies pour l'aide à la représentation de connaissances par objets*. Thèse d'informatique, Université Grenoble 1, 1999.

Semantic Matching

Fausto Giunchiglia and Pavel Shvaiko

DIT – Dept. of Information and Communication Technology
University of Trento, 38050 Povo, Trento, Italy
{fausto, pavel}@dit.unitn.it

Abstract

We think of *match* as an operator that takes two graph-like structures (e.g., database schemas or ontologies) and produces a mapping between elements of the two graphs that correspond semantically to each other. The goal of this paper is to propose a new approach to matching, called *semantic matching*. The contributions of this paper are (i) a rational reconstruction of the major matching problems and their articulation in terms of the more generic problem of matching graphs; (ii) the identification of semantic matching as a new approach for performing generic matching; and (iii) a proposal of implementing semantic matching by testing propositional satisfiability.

1 Introduction

Due to the progress of information and communication technologies the number of different information resources is rapidly increasing, and the problem of semantic heterogeneity is becoming more and more severe, see for instance [Washe *et al*., 2001], [Goh, 1997], [Giunchiglia and Zaihrayeu, 2002]. One proposed solution is matching. *Match* is an operator that takes two graph-like structures (e.g., database schemas or ontologies) and produces a mapping between elements of the two graphs that correspond semantically to each other. So far, with the noticeable exception of [Serafini *et al*, 2003], the key intuition underlying *all* the approaches to matching has been to map labels (of nodes) and to look for similarity (between labels) using syntax driven techniques and syntactic similarity measures; see for instance [Do and Rahm, 2002], [Madhavan *et al*., 2001]. We say that all these approaches are different variations of *syntactic matching*. In syntactic matching semantics are not analyzed directly, but semantic correspondences are searched for only on the basis of syntactic features.

In this paper we propose a novel approach, called *semantic matching*, with the following main features:

- We search for semantic correspondences by mapping meanings (concepts), and not labels, as in syntactic matching.
- We use semantic similarity relations between elements (concepts) instead of syntactic similarity relations. In particular, we consider relations, which relate the extensions of the concepts under consideration (for instance, more/less general relations).

The contributions of this paper are (i) a rational reconstruction of the major matching problems and their articulation in terms of the more generic problem of matching graphs; (ii) the identification of semantic matching as a new approach for performing generic matching; and (iii) a proposal of using a decider for propositional satisfiability (SAT) as a possible way of implementing semantic matching. The algorithm proposed works only on Directed Acyclic Graphs (DAG's) and *is-a* links. It is important to notice that SAT deciders are correct and complete decision procedures for propositional logics. Using SAT allows us to find only and all possible mappings between elements. This is another major advantage over syntactic matching approaches, which are based on heuristics. The SAT-based algorithm discussed in this paper is a minor modification/extension of the work described in [Serafini *et al*, 2003].

The rest of the paper is organized as follows. Section 2 defines the notion of matching and discusses the essence of semantic matching. Section 3 provides guidelines to the implementation of semantic matching. Section 4 overviews the related work. Section 5 reports some conclusions.

2 Matching

We assume that all the data and conceptual models (e.g., relational db schemas, OODB and XML schemas, concept hierarchies and ontologies) can be represented as graphs, see for a detailed discussion [Giunchiglia and Shvaiko, 2003]. Therefore, the problem of matching heterogeneous and autonomous information resources can be decomposed in two steps:

1. extract graphs from the data or conceptual models,
2. match the resulting graphs.

Notice that this allows for the statement and solution of a more *generic matching problem*, very much along the lines of what done in Cupid [Madhavan *et al*., 2001], and COMA [Do and Rahm, 2002].

Let us define the notion of matching graphs more precisely. *Mapping element* is a 4-tuple $< m_{ID}, N^i_1, N^j_2, R >$, $i=1...h$; $j=1..k$; where m_{ID} is a unique identifier of the given mapping element; N^i_1 is the *i-th* node of the first graph, h is the number of nodes in the first graph; N^j_2 is the *j-th* node of the second graph, k is the number of nodes in the second graph; and R specifies a *similarity relation* of the given nodes. A *Mapping* is a set of mapping elements. *Matching* is the process of discovering mappings between two graphs through the application of a matching algorithm. There exist two approaches to graph matching, namely *exact matching* and *inexact* or *approximate matching*. For obvious reasons we are interested in inexact matching.

We classify matching into *syntactic* and *semantic matching* depending on how matching elements are computed and on the kind of similarity relation R used.

- In *syntactic matching* the key intuition is to map labels (of nodes) and to look for the similarity using syntax driven techniques and syntactic similarity measures. Thus, in the case of *syntactic matching*, mapping elements are computed as 4-tuples $< m_{ID}, L^i_1, L^j_2, R >$, where L^i_1 is the *label* at the *i-th* node of the first graph; L^j_2 is the *label* at the *j-th* node of the second graph; and R specifies a similarity relation in the form of a coefficient, which measures the similarity between the *labels* of the given nodes. Typical examples of R are coefficients in [0,1], for instance, *similarity* coefficients [Madhavan *et al*., 2001]. Similarity coefficients usually measure the closeness between the two elements linguistically and structurally. For instance, based on linguistic analysis, the similarity coefficient between elements "telephone" and "phone" from the two hypothetical schemas could be 0,7.
- As from its name, in *semantic matching* the key intuition is to map meanings (concepts). Thus, in the case of *semantic matching*, mapping elements are computed as 4-tuples $< m_{ID}, C^i_1, C^j_2, R >$, where C^i_1 is the *concept* of the *i-th* node of the first graph; C^j_2 is the *concept* of the *j-th* node of the second graph; and R specifies a similarity relation in the form of a semantic relation between the *extensions of concepts* at the given nodes. Possible R's between nodes are *equality* (=), *overlapping* ($\cap$), *mismatch* ($\perp$), or *more general/specific* ($\subseteq$, $\supseteq$).

These ideas are schematically represented in Figure 1. It is important to notice that all past approaches to matching we are aware of, with the exception of [Serafini *et al*, 2003], are based on syntactic matching.

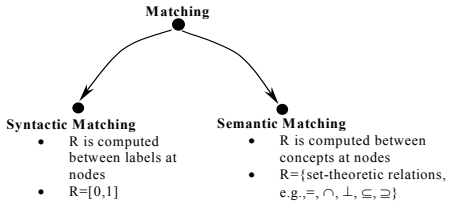


Fig.1. Matching problems

Let us consider some examples, which make the consequences of the observation described above clearer. For any example we also report the results produced by the state of the art matcher, Cupid [Madhavan *et al*., 2001], which exploits very sophisticated syntactic matching techniques. Notationally, A stands for the label at a node; C_A stands for the concept denoted by A; $\boldsymbol{C_i}$ stands for the concept at the node i (in the following we sometimes confuse concepts with their extensions), numbers in circles are the unique identifiers of the nodes under consideration. In order to keep track of the graph we refer to we index nodes, labels, concepts and their extensions with the graph number (which is "1" for the graph on the left and "2" for the graph on the right). Thus we have, for instance, A_1, 5_1, C_{A_1}, $\boldsymbol{C}_{5_1}$.

Analysis of siblings. Let us consider Figure 2. Structurally the graphs shown in Figure 2 differ in the order of siblings. Suppose that we want to match node 5_1 with node 2_2.

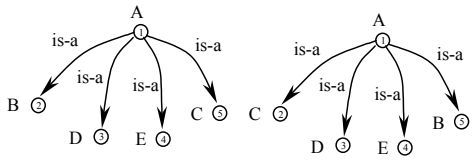


Fig.2. Analysis of siblings. Case 1

Cupid finds the similarity coefficient between labels at the given nodes, which equals to 0,8. This is because $A_1=A_2$, $C_1=C_2$ and we have the same structures on both sides. . A semantic matching approach compares concepts $C_{A_1} \cap C_{C_1}$ with $C_{A_1} \cap C_{C_1}$ and produces $\boldsymbol{C}_{5_1} = \boldsymbol{C}_{2_2}$.

Analysis of ancestors. Let us consider Figure 3. Suppose that we want to match nodes 5_1 and 1_2.

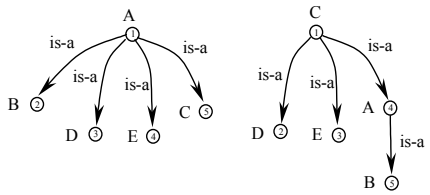


Fig.3. Analysis of ancestors. Case 1

Cupid does not find a similarity coefficient between the nodes under consideration, due to the significant differences in structure of the given graphs. In semantic matching, the concept denoted by the label at node 5_1 is C_{C_1}, while the concept at node 5_1 is $\boldsymbol{C}_{5_1} = C_{A_1} \cap C_{C_1}$. The concept at the node 1_2 is $\boldsymbol{C}_{1_2} = C_{C_2}$. By comparing the concepts denoted by the labels at nodes 5_1 and 1_2 we have that, being identical, they denote the same concept, namely $C_{C_1}=C_{C_2}$. Thus, the concept at node 5_1 is a subset of the concept at node 1_2, namely $\boldsymbol{C}_{5_1} \subseteq \boldsymbol{C}_{1_2}$.

Let us complicate the example shown in Figure 3 by allowing for an arbitrary distance between ancestors, see Figure 4. The asterisk means that an arbitrary number of nodes are allowed between nodes 1_2 and 5_2. Suppose that we want to match nodes 5_1 and 5_2.

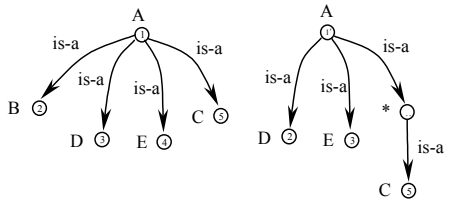


Fig.4. Analysis of ancestors. Case 2

Cupid finds out that the similarity coefficient between labels C_1 and C_2 is 0,86. This is because of the identity of labels (A_1=A_2, C_1=C_2), and due to the fact that nodes 5_1 and 5_2 are leaves. Notice how Cupid treats very differently the two situations represented here and in the example above, even if, from a semantic point of view, they are similar. Following semantic matching, the concept at node 5_1 is $\boldsymbol{C}_{5_1} = C_{A_1} \cap C_{C_1}$; while the concept at node 5_2 is $\boldsymbol{C}_{5_2} = C_{A_2} \cap * \cap C_{C_2}$. Since we have that $C_{A_1} = C_{A_2}$ and $C_{C_1} = C_{C_2}$, then $\boldsymbol{C}_{5_2} \subseteq \boldsymbol{C}_{5_1}$.

Enriched analysis of siblings. Suppose that we want to match nodes 2_1 and 2_2, see Figure 5.

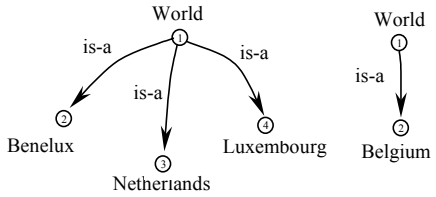


Fig.5. Analysis of siblings. Case 2

Cupid without thesaurus doesn't find a match; with the use of thesaurus it finds out that the similarity coefficient between nodes with labels $Benelux_1$ and $Belgium_2$ is 0,68. This is mainly because of the entry in the thesaurus specifying *Belgium* as a part of *Benelux,* and due to the fact that the nodes with labels $Benelux_1$ and $Belgium_2$ are leaves.

Following semantic matching, both concepts $C_{Benelux_1}$ and $C_{Belgium_2}$ are subsets of the concept $C_{World_{1,2}}$. Let us suppose that an oracle, for instance WordNet, states that *Benelux* is a name standing for *Belgium*, *Netherlands* and *Luxembourg*. Therefore, we treat $\boldsymbol{C}_{2_1}$ in Figure 5 as $C_{Benelux_1} \cap C_{Netherlands_1} \cap C_{Luxembourg_1} = C_{Belgium_1}$. Thus, $\boldsymbol{C}_{2_1} = \boldsymbol{C}_{2_2}$.

3 Implementing Semantic Matching

There are two levels of granularity while performing semantic (and also syntactic matching) matching: *element-level* and *structure-level*. Element-level matching techniques compute mapping elements between individual labels/concepts at nodes; structure-level techniques compute mapping elements between subgraphs.

3.1 Element-level Semantic Matching

Element-level semantic techniques analyze individual labels/concepts at nodes. At the element-level we can exploit all the techniques discussed in the literature, see for instance [Do and Rahm, 2002], [Melnik *et al*., 2002], [Rahm and Bernstein, 2001]. The main difference here is that, instead of a syntactic similarity measure, these techniques must be modified to return a semantic relation R, as defined in Section 2. We distinguish between *weak semantics* and *strong semantics* element-level techniques. *Weak semantics* techniques are syntax driven techniques: examples are techniques, which consider labels as strings, or analyze data types, or soundex of schema elements. Let us consider some examples.

Analysis of strings. String analysis looks for common prefixes or suffixes and calculates the distance between two strings. For example, the fact that the string "phone" is a substring of the string "telephone" can be used to infer that "phone" and "telephone" are synonyms. Before analyzing strings, a matcher could perform some preliminary parsing, e.g., extract tokens, expand abbreviations, delete articles and then match tokens. The analysis of strings discovers only equality between concepts.

Analysis of data types. These techniques analyze the data types of the elements to be compared and are usually performed in combination with string analysis. For example, the elements "phone" and "telephone" are supposed to have the same data type, namely "string" and therefore can be found equal. However, "phone" could also be specified as an "integer" data type. In this case a mismatch is found. As another example the integer "Quantity" is found to be a subset of the real "Qty". This kind of analysis can produce any kind of semantic relation.

Analysis of soundex. These techniques analyze elements' names from how they sound. For example, elements "for you" and "4 U" are different in spelling, but similar in soundex. This analysis can discover only equality between concepts.

Strong semantics techniques exploit, at the element-level, the semantics of labels. These techniques are based on the use of tools, which explicitly codify semantic information, e.g. thesauruses [Madhavan *et al*., 2001], WordNet or combinations of them [Castano *et al*., 2000]. Notice that these techniques are also used in syntactic matching. In this latter case, however, the semantic information is lost before moving to structure-level matching and approximately codified in syntactic relations.

Precompiled thesaurus. A precompiled thesaurus usually stores entries with synonym and hypernym relations. For example, the elements "e-mail" and "email" are treated as synonyms from the thesaurus look up: *syn key* - "e-mail:email = syn". Precompiled thesauruses (most of them) identify equivalence and more general/specific relations. In some cases domain ontologies are used as precompiled thesauruses [Mena *et al.*, 1996].

WordNet. WordNet is an electronic lexical database for English (and other languages), where various *senses* (namely, possible meanings of a word or expression) of words are put together into sets of synonyms (synsets). Synsets in turn are organized as hierarchy. Following [Serafini *et al*, 2003] we can define the semantic relations in terms of senses. *Equality*: one concept is equal to another if there is at least one sense of the first concept, which is a synonym of the second. *Overlapping*: one concept is overlapped with the other if there are

some senses in common. *Mismatch*: two concepts are mismatched if they have no sense in common. *More general / specific*: One concept is more general than the other iff there exists at least one sense of the first concept that has a sense of the other as a hyponym or as a meronym. One concept is less general than the other iff there exists at least one sense of the first concept that has a sense of the other concept as a hypernym or as a holonym. For example, according to WordNet, the concept "hat" is a holonym for the concept "brim", which means that "brim" is less general than "hat".

3.2 Structure-level Semantic Matching

The approach we propose is to translate the matching problem, namely the two graphs and our *mapping queries* into a propositional formula and then to check it for its validity. By mapping query we mean here the pair of nodes that we think will match and the semantic relation between them. In the following we show how, limited to the case of DAG's and *is-a* hierarchies, we can check validity by using propositional satisfiability (SAT) decider. Notice that SAT deciders are correct and complete decision procedures for propositional satisfiability and therefore will exhaustively check for all possible mappings. Being complete, they automatically implement all the examples described in the previous section, and more. This is another advantage over syntactic matching, whose existing implementations are based only on heuristics.

Our SAT based approach to semantic matching incorporates six steps. We describe below its intended behavior by running these six steps on the example shown in Figure 3 and by matching nodes 5_1 and 1_2 (steps 2-5 are taken from [Serafini *et al*, 2003]).

1. **Extract the two graphs.** Notice that during this step, in the case of DB, XML or OODB schemas, it is necessary to extract useful semantic information, for instance in the form of ontologies. There are various techniques for doing this, see for instance [Davis and Aiken, 2000], [Mena *et al.*, 1996]. The result is the graph in Figure 3.
2. **Compute element-level semantic matching.** For each node, compute semantic relations holding among all the concepts denoted by labels at nodes under consideration. In this case C_{A_1} has no semantic relation with C_{C_2} while we have that $C_{C_1} = C_{C_2}$.
3. **Compute concepts at nodes.** Starting from the root of the graph, attach to each node the concepts of all the nodes above it. Thus, we attach $\boldsymbol{C}_{1_1} = C_{A_1}$ to node 1_1; $\boldsymbol{C}_{5_1} = C_{A_1} \cap C_{C_1}$ to node 5_1; $\boldsymbol{C}_{1_2} = C_{C_2}$ to node 1_2 in the *is-a* hierarchy. As it turns out we have that $\boldsymbol{C}_{5_1} \subseteq \boldsymbol{C}_{1_2}$.
4. **Construct the propositional formula**, representing the matching problem. In this step we translate all the semantic relations computed in step 2 into propositional formulas. This is done according to the following transition rules:

$$C_{A_1} \supseteq C_{A_2} \Rightarrow C_{A_2} \rightarrow C_{A_1}$$
$$C_{A_1} \subseteq C_{A_2} \Rightarrow C_{A_1} \rightarrow C_{A_2}$$
$$C_{A_1} = C_{A_2} \Rightarrow C_{A_1} \equiv C_{A_2}$$
$$C_{A_1} \perp C_{A_2} \Rightarrow \neg(C_{A_1} \wedge C_{A_2})$$

Subset translates into implication; equality into equivalence; disjointness into the negation of conjunction. In the case of Figure 3 we have that $C_{C_1} \equiv C_{C_2}$ is an axiom. Furthermore, since we want to prove that $\boldsymbol{C}_{5_1} \subseteq \boldsymbol{C}_{1_2}$, our goal is to prove that $((C_{A_1} \wedge C_{C_1}) \rightarrow C_{C_2})$. Thus, our target formula is $((C_{C_1} \equiv C_{C_2}) \rightarrow (C_{A_1} \wedge C_{C_1}) \rightarrow C_{C_2}))$.

5. **Run SAT.** In order to prove that $((C_{C_1} \equiv C_{C_2}) \rightarrow (C_{A_1} \wedge C_{C_1}) \rightarrow C_{C_2}))$ is valid, we prove that its negation is unsatisfiabile, namely that a SAT solver run on the following formula $((C_{C_1} \equiv C_{C_2}) \wedge\neg\ (C_{A_1} \wedge C_{C_1}) \rightarrow C_{C_2}))$ fails. A quick analysis shows that SAT will return FALSE.
6. **Iterations.** Iterations are performed re-running SAT. We need iterations, for instance, when matching results are not good enough, for instance no matching is found or a form of matching is found, which is too weak, and so on[1]. The idea is to exploit the results obtained during the previous run of SAT to tune the matching and improve the quality of the final outcome. Let us consider Figure 6.

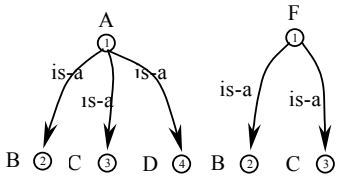


Fig.6. Not good enough answer

Suppose that we have found out that $\boldsymbol{C}_{2_1} \cap \boldsymbol{C}_{2_2} \neq \emptyset$, and that we want to improve this result. Suppose that an oracle tells us that $C_{A_1} = C_{F_2} \cup C_{G_2}$. In this case the graph on the left in Figure 6 can be transformed into the two graphs in Figure 7.

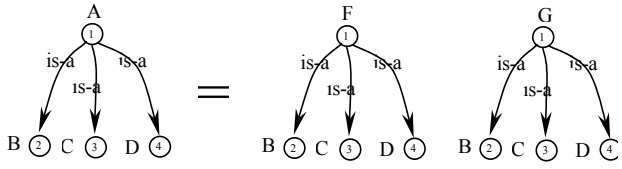


Fig.7. Extraction of additional semantic information

After this additional analysis we can infer that $\boldsymbol{C}_{2_1} = \boldsymbol{C}_{2_2}$. As a particular interesting case, consider the following situation, see Figure 7.1

[1] [Giunchiglia and Zaihrayeu, 2002] provides a long discussion about the importance of dealing with the notion of "good enough answer" in information coordination in peer-to-peer systems.

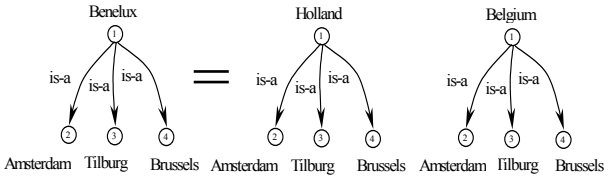


Fig.7.1. Extraction of additional semantic information. Example

In this case the concept *Brussels* in the graph on the left (after the sign "=") becomes inconsistent (empty intersection) and can be omitted; and the same for the concepts at nodes *Amsterdam* and *Tilburg* in the graph on the right. The resulting situation is as follows:

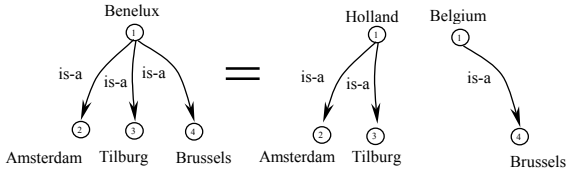


Fig.7.2. Extraction of additional semantic information. Example

Another motivation for multiple iterations is to use the result of a previous match in order to speed up the search of new matches. Consider the following example.

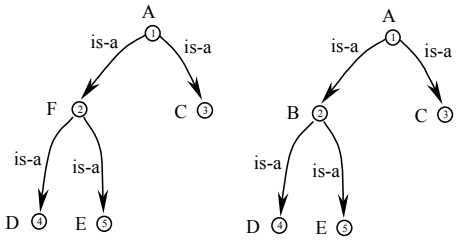


Fig.8. Iterations

Having found that $C_{2_1} \subseteq C_{2_2}$, we can automatically infer that $C_{5_1} \subseteq C_{5_2}$, without rerunning SAT, for obvious reasons, and the same for C_{4_1} and C_{4_2}. As a particular case consider the following situation:

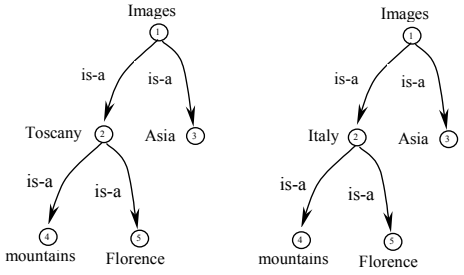


Fig. 8.1. Iterations. Example

Our algorithm allows us to find that $C_{5_1} \subseteq C_{5_2}$, while, being Tuscany in Italy we actually have $C_{5_1} = C_{5_2}$. This is an acceptable result as long as we are not looking for the strongest possible relation holding between two nodes.

4 Related Work

From a technical point of view the matcher we have proposed in this paper is a function *MatchNodesR*(G_1,G_2,n_1,n_2,R) which takes two graphs, two nodes, and a relation and returns a Yes/No answer. Most matchers proposed in the literature are a function *Match*(G_1,G_2) which takes two graphs and returns a set of mappings (n_1, n_2, R). However, it is easy to see how we can build an analogous function. The naive approach being to triple loop on the nodes of the graphs and on the set of proposed relations and, at each loop, call *MatchNodesR*.

At present, there exists a line of semi-automated schema matching and ontology integration systems, see for instance [Madhavan *et al*., 2001], [Do and Rahm, 2002], [Li and Clifton, 2000], [Castano *et al*., 2000], [Arens *et al*., 1996], [Mena *et al*., 1996], [Doan *et al*., 2002], etc. Most of them implement syntactic matching. A good survey, up to 2001, is provided in [Rahm and Bernstein, 2001]. The classification given in this survey distinguishes between individual implementations of match and combinations of matchers. Individual matchers comprise instance- and schema-level, element- and structure-level, linguistic- and constrained-based matching techniques. Individual matchers can be used in different ways, e.g. simultaneously (hybrid matchers), see [Li and Clifton, 2000], [Castano *et al*., 2000], [Madhavan *et al*., 2001] or in series (composite matchers), see for instance [Doan *et al*., 2002], [Do and Rahm, 2002].

The idea of generic (syntactic) matching was first proposed by Phil Bernstein and implemented in Cupid system [Madhavan *et al*., 2001]. Cupid implements a complicated hybrid match algorithm comprising linguistic and structural schema matching techniques, and computes normalized similarity coefficients with the assistance of a precompiled thesaurus. COMA [Do and Rahm, 2002] is a generic schema matching tool, which implements more recent composite generic matchers. With respect to Cupid, the main innovation seems to be a more flexible architecture.

A lot of state of the art syntactic matching techniques exploiting weak semantic element-level matching techniques have been implemented. For instance, in COMA, schemas are internally encoded as DAG's, where the elements are the paths, which are analyzed using string comparison techniques. Similar ideas are exploited in Similarity Flooding (SF) [Melnik *et al*., 2002]. SF is a hybrid matching algorithm based on the ideas of similarity propagation. Schemas are presented as directed labeled graphs; the algorithm manipulates them in an iterative fix-point computation to produce mappings between the nodes of the input graphs. The technique uses a syntactic string comparison mechanism of the vertices' names to obtain an initial mapping, which is further refined within the fix-point computation.

Some work has also been done in strong semantics element-level matching. For example, [Castano *et al*., 2000] utilizes a common thesaurus, while [Madhavan *et al*., 2001] has a precompiled thesaurus. In MOMIS [Castano *et al*., 2000] element-level matching using a common thesau-

rus is carried out through a calculation of the name, structural and global affinity coefficients. The thesaurus presents a set of intensional and extensional relations, which depict intra- and inter-schema knowledge about classes, and attributes of the input schemas. All these systems implement syntactic matching and, when moving from element-level to structure-level matching, don't exploit the semantic information residing in the graph structure, and just translate the element-level semantic information into affinity levels.

As far as we know the only example where element-level and a simplified version of structure- level strong semantics matching have been applied is CTXmatch [Serafini *et al*, 2003]. The main problem of CTXmatch is that its rather limited in scope (it applies only to concept hierarchies), and it is hard to see the general lessons behind this work. This paper provides the basics for a better understanding of the work on CTXmatch.

5 Conclusions and Future Work

In this paper we have stated and analyzed the major matching problems e.g., matching database schemas, XML schemas, conceptual hierarchies and ontologies and shown how all these problems can be defined as a more generic problem of matching graphs. We have identified semantic matching as a new approach for performing generic matching, and discussed some of its key properties. Finally, we have identified SAT as a possible way of implementing semantic matching, and proposed an iterative semantic matching approach based on SAT.

This is only very preliminary work, some of the main issues we need to work on are: develop an efficient implementation of the system, do a thorough testing of the system, also against the other state of the art matching systems, study how to take into account attributes and instances, and so on.

Acknowledgments

Thanks to Luciano Serafini, Paolo Bouquet for many discussions on CTXmatch. Stefano Zanobini and Phil Bernstein have proposed very useful feedback on the semi-final version of this paper. Also thanks to Michail Yatskevich for his work on running SAT solvers on our matching problems.

References

[Arens *et al.*, 1996] Yigal Arens, Chun-Nan Hsu, and Craig A. Knoblock. Query processing in the SIMS information mediator. In *Advanced Planning Technology*. AAAI Press, California, USA, 1996.

[Buneman, 1997] Peter Buneman. Semistructured data. In *Proc. of PODS*, pages 117–121, 1997.

[Castano *et al*., 2000] Castano S., V. De Antonellis, S. De Capitani di Vimercati. Global Viewing of Heterogeneous Data Sources. *IEEE Trans. on Knowledge and Data Engineering*, 2000

[Doan *et al*., 2002] A. Doan, J.Madhavan, P. Domingos, and A. Halvey. Learning to map between ontologies on the semantic web. In *Proc. Of WWW-02, 11th International WWW Conf.*, Hawaii 2002.

[Do and Rahm, 2002] Hong H. Do, Erhard Rahm. COMA – A System for Flexible Combination of Schema Matching Approach. *VLDB Journal*, pages 610-621, 2002

[Goh, 1997] Cheng Hian Goh. *Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources*. Phd, MIT, 1997.

[Giunchiglia and Shvaiko, 2003] Fausto Giunchiglia and Pavel Shvaiko. Semantic Matching. *Technical Report #DIT-03-013*. http://www.dit.unitn.it/~p2p/

[Giunchiglia and Zaihrayeu, 2002] Fausto Giunchiglia and Ilya Zaihrayeu. Making peer databases interact - a vision for an architecture supporting data coordination. *Proceedings of the Conference on Information Agents*, Madrid, September 2002.

[Li and Clifton, 2000] W. Li, C. Clifton: SEMINT: A tool for identifying attribute correspondences in heterogeneous databases using neural networks. *Data & Knowledge Engineering*, 33(1): 49-84, 2000.

[Davis and Aiken, 2000] Kathi Hogshead Davis, Peter H. Aiken: Data reverse engineering: a historical survey, WCRE'00, 2000.

[Madhavan *et al*., 2001] Jayant Madhavan, Philip A. Bernstein, and Erhard Rahm. Generic schema matching with Cupid. *VLDB Journal*, pages 49-58, 2001

[Melnik *et al*., 2002] Melnik, S., H. Garcia-Molina, E. Rahm: Similarity Flooding: A Versatile Graph Matching Algorithm. *ICDE*, 2002.

[Mena *et al.*, 1996] E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. Observer: An approach for query processing in global information systems based on interoperability between pre-existing ontologies. In *Proceedings 1st International Conference on Cooperative Information Systems*. Brussels, 1996.

[Rahm and Bernstein, 2001] Erhard Rahm and Philip A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4): 334-350, 2001.

[Serafini *et al*, 2003] Luciano Serafini, Paolo Bouquet, Bernardo Magnini, and Stefano Zanobini. An Algorithm for Matching Contextualized Schemas via SAT. In *Proc. of CONTEX 03,* June 2003.

[Washe *et al*., 2001] H. Wache, T. Voegele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Huebner. Ontology-based integration of information - a survey of existing approaches. In *Proc. of IJCAI*, August 2001.

Semantic Integration through Invariants

Michael Grüninger
Manufacturing Systems Integration Division
National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, MD 20899-8260
gruning@cme.nist.gov

Joseph B. Kopena
Geometric and Intelligent Computing Laboratory
College of Engineering, Drexel University
3141 Chestnut St
Philadelphia, PA 19104
joe@plan.mcs.drexel.edu

Introduction

A semantics-preserving exchange of information requires mappings between logically equivalent concepts in each ontology. The challenge of semantic integration is therefore equivalent to the problem of generating such mappings, determining that they are correct, and providing a vehicle for executing the mappings, thus translating terms from one ontology into another.

Current approaches to semantic integration ((5), (7)) emphasize the use of generic techniques that do not exploit the model-theoretic structures of the ontologies. In this paper we will show how the classification of models within the PSL Ontology can serve as the basis for generating semantic mappings between applications.

The Process Specification Language (PSL) ((2), (4), (6)) has been designed to facilitate correct and complete exchange of process information among manufacturing systems[1], such as scheduling, process modeling, process planning, production planning, simulation, project management, workflow, and business process reengineering. PSL is intended to be used as a mediating ontology that is independent of the applications' ontologies and that is used as a neutral interchange ontology ((1)). The semantic mappings between application ontologies and PSL can be semi-automatically generated from invariants (properties of models preserved by isomorphism). Since these invariants are also used to characterize the definitional extensions within the PSL Ontology, the semantic mappings can be verified prior to integration.

PSL Ontology

The PSL Ontology is a set of theories in the language of first-order logic. Theories that introduce new primitive concepts are referred to as core theories, while theories containing only conservative definitions are referred to as definitional extensions[2].

[1]PSL has been accepted as International Organisation of Standardisation project ISO 18629; as of June 2003, part of the work is under review as a Draft International Standard.

[2]The complete set of axioms for the PSL Ontology can be found at `http://www.mel.nist.gov/psl/psl-ontology/`.

Core Theories

All core theories within the ontology are consistent extensions of PSL-Core (T_{psl_core}). The purpose of PSL-Core is to axiomatize a set of intuitive semantic primitives that is adequate for describing the fundamental concepts of manufacturing processes. Specifically, PSL-Core introduces four disjoint classes: activities, activity occurrences, timepoints, and objects. Activities may have zero or more occurrences, activity occurrences begin and end at timepoints, and timepoints constitute a linearly ordered set with endpoints at infinity. Objects are simply those elements that are not activities, occurrences, or timepoints. Extensions to PSL-Core defining the core theories include:

Occurrence Trees The occurrence trees that are axiomatized in the core theory $T_{occtree}$ are partially ordered sets of activity occurrences—for a given set of activities, all discrete sequences of their occurrences are branches of a tree. An occurrence tree contains all occurrences of *all* activities, not simply the set of occurrences of a particular (possibly complex) activity. As each tree is discrete, every activity occurrence in the tree has a unique successor occurrence of each activity.

There are constraints on which activities can possibly occur in some domain. This intuition is the cornerstone for characterizing the semantics of classes of activities and process descriptions. Although occurrence trees characterize all sequences of activity occurrences, not all of these sequences will intuitively be physically possible within the domain. We will therefore want to consider the subtrees of the occurrence trees that consist only of *possible* sequences of activity occurrences; such a subtree is referred to as a legal occurrence tree.

Discrete States The core theory T_{disc_state} introduces the notion of fluents (state). Fluents are changed only by the occurrence of activities, and fluents do not change during the occurrence of primitive activities. In addition, activities have preconditions (fluents that must hold before an occurrence) and effects (flu-

Core theories are indicated by a .th suffix and definitional extensions by a .def suffix. As of June 2003, the ontology is in version 2.0.

ents that always hold after an occurrence).

Subactivities The PSL Ontology uses the *subactivity* relation to capture the basic intuitions for the composition of activities. This relation is a discrete partial ordering in which primitive activities are the minimal elements.

Atomic Activities The core theory T_{atomic} axiomatizes intuitions about the concurrent aggregation of primitive activities. This is represented by the occurrence of concurrent activities, rather than concurrent activity occurrences.

Complex Activities The core theory $T_{complex}$ characterizes the relationship between the occurrence of a complex activity and occurrences of its subactivities. Occurrences of complex activities correspond to sets of occurrences of subactivities; in particular, these sets are subtrees of the occurrence trees. An activity tree consists of all possible sequences of atomic subactivity occurrences beginning from a root subactivity occurrence. In a sense, activity trees are a microcosm of an occurrence tree, in which we consider all of the ways in which the world unfolds *in the context of an occurrence of the complex activity.*

Definitional Extensions

Many ontologies are specified as taxonomies or class hierarchies, yet few ever give any justification for their classification scheme. If we consider ontologies of mathematical structures, we see that logicians classify models by using properties of models, known as invariants, that are preserved by isomorphism. For some classes of structures, such as vector spaces, invariants can be used to classify the structures up to isomorphism; for example, vector spaces can be classified up to isomorphism by their dimension. For other classes of structures, such as graphs, it is not possible to formulate a complete set of invariants. However, even without a complete set, invariants can still be used to provide a classification of the models of a theory.

Following this methodology, the set of models for the core theories of PSL are partitioned into equivalence classes defined with respect to the set of invariants of the models. Each equivalence class in the classification of PSL models is axiomatized using a definitional extension of PSL. In particular, each definitional extension in the PSL Ontology is associated with a unique invariant; the different classes of activities or objects that are defined in an extension correspond to different properties of the invariant. In this way, the terminology of the PSL Ontology arises from the classification of the models of the core theories with respect to sets of invariants and intuitively corresponds to classes of activities and objects.

Many of the invariants with definitional extensions in the PSL Ontology are related to the automorphism groups[3] for different substructures of the models. For example, we can consider mappings that are permutations of activity occurrences that map the predecessor of a legal occurrence of an activity **a** to other predecessors of legal occurrences of **a** in an occurrence tree. This set of mappings forms a group, which is referred to as $OP(\mathbf{a})$. Each invariant related to occurrence constraints is based on subgroups of this group.

The most prevalent class of occurrence constraints is the case of Markovian activities, that is, activities whose preconditions depend only on the state prior to the occurrences; the class of Markovian activities is defined in the definitional extension *state_precond.def* (see Figure 1). The invariant associated with this extension is the group[4] $\mathcal{P}^{\mathcal{F}}(\mathbf{a})$, which is the maximal normal subgroup of $Aut(\mathcal{F})$ that is also a subgroup of $OP(\mathbf{a})$. If $\mathcal{P}^{\mathcal{F}}(\mathbf{a}) = Aut(\mathcal{F})$, then these permutations preserve the legal occurrences of an activity, and the activity's preconditions are strictly Markovian; this is axiomatized by the *markov_precond* class in Figure 1. If $\mathcal{P}^{\mathcal{F}}(\mathbf{a})$ is only a subgroup of $Aut(\mathcal{F})$, then there exist additional nonmarkovian constraints on the legal occurrences of the activity; this is axiomatized by the *partial_state* class in Figure 1. If $\mathcal{P}^{\mathcal{F}}(\mathbf{a})$ is the trivial identity group, then there are no Markovian constraints on the legal occurrences of the activity; this is axiomatized by the *rigid_state* class in Figure 1.

Additional relations are defined to capture the action of the automorphism groups on the models. Two activity occurrences o_1, o_2 are *state_equiv* iff there exists a permutation in $Aut(\mathcal{F})$ that maps o_1 to o_2; the two activity occurrences are *poss_equiv* iff there exists a permutation in $OP(\mathbf{a})$ that maps o_1 to o_2.

Translation Definitions

Translation definitions specify the mappings between PSL and application ontologies. Such definitions have a special syntactic form—they are biconditionals in which the antecedent is a class in the application ontology and the consequent is a formula that uses only the lexicon of the PSL Ontology.

Translation definitions are generated using the organization of the definitional extensions, each of which corresponds to a different invariant. Every class of activity, activity occurrence, or fluent in an extension corresponds to a different value for the invariant. The consequent of a translation definition is equivalent to the list of invariant values for members of the application ontology class.

[3] An automorphism is a bijection from a structure to itself that preserves the extensions of the relations and functions in the structure. Intuitively, it is a symmetry in the structure.

[4] In this example, $\mathcal{F}$ is the structure isomorphic to the extension of the *prior* relation. $Aut(\mathcal{F})$ is the group of permutations that map activity occurrences only to other activity occurrences that agree on the set of fluents that hold prior to them.

$$(\forall o_1, o_2)\ state\_equiv(o_1, o_2) \equiv \quad (1)$$
$$(\forall f)\ (prior(f, o_1) \equiv prior(f, o_2))$$
$$(\forall a, o_1, o_2)\ poss\_equiv(a, o_1, o_2) \equiv \quad (2)$$
$$(poss(a, o_1) \equiv poss(a, o_2))$$
$$(\forall a)\ markov\_precond(a) \equiv \quad (3)$$
$$((\forall o_1, o_2)\ state\_equiv(o_1, o_2) \supset poss\_equiv(a, o_1, o_2))$$
$$(\forall a)\ partial\_state(a) \equiv \quad (4)$$
$$(\exists o_1)\ ((\forall o_2)\ state\_equiv(o_1, o_2) \supset poss\_equiv(a, o_1, o_2))$$
$$\wedge(\exists o_3, o_4)\ state\_equiv(o_3, o_4) \wedge \neg poss\_equiv(a, o_3, o_4)$$
$$(\forall a)\ rigid\_state(a) \equiv \quad (5)$$
$$(\forall o_1)(\exists o_2)\ state\_equiv(o_1, o_2) \wedge \neg poss\_equiv(a, o_1, o_2)$$

Figure 1: Classes of activities with state-based preconditions (from the definitional extension *state_precond.def*).

For example, the concept of *AtomicProcess* in the DAML-S Ontology ((3)) has the following translation definition:

$$(\forall a)\ AtomicProcess(a) \equiv$$
$$primitive(a) \wedge markov\_precond(a)$$
$$\wedge((markov\_effects(a) \vee context\_free(a))$$

This methodology has been implemented in the PSL project's Twenty Questions mapping tool[5]. Each question corresponds to an invariant, and each possible value of the invariant is a possible answer to the question. Any particular activity, activity occurrence, or fluent will have a unique value for the invariant; however, if we are mapping a class of activities, occurrences, or fluents from some application ontology, then different members of the class may have different values for the same invariant. In such a case, one would respond to a question by supplying multiple answers.

For example, consider the question displayed in Figure 2. The invariant corresponding to this question is $\mathcal{P}^{\mathcal{F}}(\mathbf{a})$, and the classes of activities corresponding to values of this invariant are axiomatized in Figure 1. Selecting the first answer would generate the translation definition:

$$(\forall a)\ myclass(a) \equiv markov\_precond(a)$$

Selecting the first two answers would give the translation definition:

$$(\forall a) myclass(a) \equiv (markov\_precond(a) \vee partial\_state(a))$$

In this latter case, some activities in *myclass* will have markov preconditions while other activities will not.

[5]Available at `http://ats.nist.gov/psl/twenty.html`.

2. Constraints on Atomic Activity Occurrences based on State

Are the constraints on the occurrence of the atomic activity based only on the state prior to the activity occurrence?

- ☐ Any occurrence of the activity depends only on fluents that hold prior to the activity occurrence.
- ☐ Some (but not all) occurrences of the activity depend only on fluents that hold prior to the activity occurrence.
- ☐ There is no relationship between occurrences of the activity and the fluents that hold prior to occurrences of the activity.

Figure 2: One of the Twenty Questions, used to classify activities with state-based preconditions.

When building translators, we are faced with the additional challenge that almost no application has an explicitly axiomatized ontology. However, we take the Ontological Stance ((4)), in which we model a software application as if it were an inference system with an axiomatized ontology, and use this ontology to predict the set of sentences that the inference system decides to be satisfiable. The Twenty Questions tool supports this by allowing the application designer to specify the intended semantics of her ontology by using the classes in the PSL Ontology.

Process Information Exchange Profiles

In addition to providing mappings between an application and PSL, we can also use the translation definitions to directly specify the relationship between two application ontologies. For example, suppose we have a scenario in which two software agents, Alice and Bob, need to exchange process information. Alice's designer specifies the semantic mapping (translation definitions) between Alice's ontology and the PSL ontology, and Bob's designer specifies the semantic mapping between Bob's ontology and the PSL ontology. When Alice and Bob first interact, they use these previously specified mappings to automatically generate the semantic mappings between each other's ontologies. In this way, the PSL Ontology mediates the mapping between the agent ontologies.

The set of translation definitions for all concepts in a software application's ontology is the profile for the application. If the PSL Ontology has m invariants and each invariant n values, then an application profile will have the form:

$$(\forall a)\ C_1^{onto}(a) \equiv$$
$$(p_{11}(a) \vee ... \vee p_{1n}(a)) \wedge ... \wedge (p_{m1}(a) \vee ... \vee p_{mn}(a))$$
$$\vdots$$

$$(\forall a)\, C_k^{onto}(a) \equiv$$

$$(p_{11}(a) \vee ... \vee p_{1n}(a)) \wedge ... \wedge (p_{m1}(a) \vee ... \vee p_{mn}(a))$$

For example, we may have:

$$(\forall a)\, C_1^{alice}(a) \equiv unconstrained(a)$$

$$\wedge(markov\_effects(a) \vee context\_free(a))$$

$$(\forall a)\, C_1^{bob}(a) \equiv$$

$$(unconstrained(a) \vee markov\_precond(a)) \wedge context\_free(a)$$

In general, we want to use PSL and the profiles to determine the relationship between the application ontologies. The mapping for the above example would be:

$$T_{psl} \models (\forall a) markov\_precond(a) \supset (C_1^{alice}(a) \supset C_1^{bob}(a))$$

$$T_{psl} \models (\forall a) markov\_effects(a) \supset (C_1^{bob}(a) \supset C_1^{alice}(a))$$

These mappings will in general take the form of:

$$(\forall a)\, (p_{11}(a) \vee ... \vee p_{1n}(a)) \wedge ... \wedge (p_{m1}(a) \vee ... \vee p_{mn}(a))$$

$$\supset (C_i^{alice}(a) \supset C_j^{bob}(a)))$$

The antecedents of these sentences can be considered to be guard conditions that determine which activities can be shared between Alice and Bob. This can either be used to support direct exchange between Alice and Bob, or simply as a comparison between the application ontologies for Alice and Bob. In the example, Alice can export any *unconstrained* activity description to Bob and Bob can export any *context_free* activity description to Alice; however, Alice cannot export *markov_precond* activity descriptions to Bob and Bob cannot export any *markov_effects* activity descriptions to Alice.

Summary

In this paper we have described how the use of model-theoretic invariants can be used to specify translation definitions between application ontologies and PSL. The sets of models for the core theories of PSL are partitioned into equivalence classes defined with respect to the invariants of the models. Each equivalence class in the classification of PSL models is axiomatized using a definitional extension of PSL. The Twenty Questions tool that is based on these invariants and definitional extensions supports the semiautomatic generation of semantic mappings between an application ontology and the PSL Ontology. This approach can be generalized to other ontologies by specifying the invariants for the models of the axiomatizations. Future work in this area includes developing software to generate mappings based on profiles created with the Twenty Questions tool and application to translation between PSL and other ontologies (such as DAML-S) and translators for existing process modelers and schedulers.

References

Ciocoiu, M., Gruninger M., and Nau, D. (2001) Ontologies for integrating engineering applications, *Journal of Computing and Information Science in Engineering*, 1:45-60.

Gruninger, M. (2003) A Guide to the Ontology of the Process Specification Language", in *Handbook on Ontologies in Information Systems*, R. Studer and S. Staab (eds.). Springer-Verlag.

McIlraith, S., Son, T.C. and Zeng, H. (2001) Semantic Web Services, *IEEE Intelligent Systems*, Special Issue on the Semantic Web. 16:46–53, March/April, 2001.

Menzel, C. and Gruninger, M. (2001) A formal foundation for process modeling, *Second International Conference on Formal Ontologies in Information Systems*, Welty and Smith (eds), 256-269.

Noy, N. and Musen, M. (2000) PROMPT: Algorithm and tool for automated ontology merging and alignment, Proceedings of AAAI-2000.

Schlenoff, C., Gruninger, M., Ciocoiu, M., (1999) The Essence of the Process Specification Language, *Transactions of the Society for Computer Simulation* vol.16 no.4 (December 1999) pages 204-216.

Stuckenschmidt, H. and Visser, U. (2000) Semantic Translation Based on Approximate Reclassification. In Proceedings of the Seventh International Conference on Knowledge Representation and Reasoning, Breckenridge, Colorado.

Semantic Integration in the IFF

Robert E. Kent
Ontologos
rekent@ontologos.org

Abstract

The IEEE P1600.1 Standard Upper Ontology (SUO) project aims to specify an upper ontology that will provide a structure and a set of general concepts upon which domain ontologies could be constructed. The Information Flow Framework (IFF), which is being developed under the auspices of the SUO Working Group, represents the structural aspect of the SUO. The IFF is based on category theory. Semantic integration of object-level ontologies in the IFF is represented with its fusion construction[*]*. The IFF maintains ontologies using powerful composition primitives, which includes the fusion construction.*

1. The Information Flow Framework

The IEEE P1600.1 Standard Upper Ontology (SUO)[1] project aims to specify an upper ontology that will provide a structure and a set of general concepts upon which object-level domain ontologies could be constructed. These object-level domain ontologies will utilize the SUO for "applications such as data interoperability, information search and retrieval, automated inferencing, and natural language processing". A central purpose of the SUO project is interoperability.

The Information Flow Framework (IFF)[2] is being developed to represent the structural aspect of the SUO. It aims to provide semantic interoperability among various object-level ontologies. The IFF supports this interoperability by its architecture and its use of a particular branch of mathematics known as category theory (Mac Lane, 1971). A major reason that the IFF uses the architecture and formalisms that it does is to support modular ontology development. Modularity facilitates the development, testing, maintenance, and use of ontologies. The categorical approach of the IFF provides a principled framework for modular design via a structural metatheory of object-level ontologies. Such a metatheory is a method for representing the structural relationships between ontologies.

The IFF provides mechanisms for the principled foundation of a metalevel ontological framework – a framework for sharing ontologies, manipulating ontologies as objects, relating ontologies through morphisms, partitioning ontologies, composing ontologies via fusions, noting dependencies between ontologies, declaring the use of other ontologies[3], etc. The IFF takes a building blocks approach towards the development of object-level ontological structure. This is a rather elaborate categorical approach, which uses insights and ideas from the theory of distributed logic known as information flow (Barwise and Seligman, 1997) and the theory of formal concept analysis (Ganter and Wille, 1999). The IFF represents metalogic, and as such operates at the structural level of ontologies. In the IFF, there is a precise boundary between the metalevel and the object level.

The modular architecture of the IFF consists of metalevels, namespaces and meta-ontologies. There are three metalevels: top, upper and lower. This partition, which corresponds to the set-theoretic distinction between small (sets), large (classes) and generic collections, is permanent. Each metalevel services the level below by providing a language that is used to declare and axiomatize that level. The top metalevel services the upper metalevel, the upper metalevel services the lower metalevel, and the lower metalevel services the object-level. Within each metalevel, the terminology is partitioned into namespaces[†]. The number of namespaces and the content may vary over time: new namespaces may be created or old namespaces may be deprecated, and new terminology and axiomatization within any particular namespace may change. In addition, within each level, various namespaces are collected together into meaningful composites called meta-ontologies. At any particular metalevel, these meta-ontologies cover all the namespaces at that level, but they may overlap. The number of meta-ontologies and the content of any meta-ontology may vary over time: new meta-ontologies may be created or old meta-ontologies may be deprecated, and new namespaces within any particular meta-ontology may change (new versions).

The top IFF metalevel provides an interface between the simple IFF-KIF language and the other IFF terminology. By analogy, the simple IFF-KIF language is like a machine language and the top IFF metalevel is like an assembly language. There is only one namespace and one meta-ontology in the top metalevel: the Top Core (meta) Ontology. This meta-ontology represents generic collections. In a sense, it bootstraps the rest of the IFF into existence. The single namespace, the meta-ontology and the top metalevel can be identified with each other. The upper and lower IFF metalevels represent the structural aspect of the SUO. By analogy, the structural aspect of the SUO is

[*] Throughout this paper, we use the intuitive terminology of mathematical context, passage/construction, pair of invertible passages and fusion for the mathematical concepts of category, functor, adjunction and colimit, respectively.

[†] The IFF terminology is disambiguated via the disjoint union of local namespace terminology. A fully qualified term in the IFF is of the form "ν$τ", where the namespace prefix label "ν" is a "." separated sequence of alphabetic strings that uniquely represents an IFF namespace, and the local unqualified term "τ" is a unique lowercase alphanumeric-dash string within that namespace. For example: the term

"`th.col.psh$coequalizer-diagram`"

represents the coequalizer diagram underlying a pushout diagram of theories within the theory pushout namespace in the lower IFF metalevel.

like a high level programming language such as Lisp, Java, ML, etc. There are three permanent meta-ontologies in the upper metalevel: the Upper Core (meta) Ontology represents the large collections called classes; the Category Theory (meta) Ontology represents category theory; and the Upper Classification (meta) Ontology represents information flow and formal concept analysis. There will eventually be many meta-ontologies situated in the lower IFF metalevel[‡]. Currently there are only four: the Lower Core (meta) Ontology represents the small collections called sets; the Lower Classification (meta) Ontology is a small and more specialized version of its upper counterpart; the Algebraic Theory (meta) Ontology represents equational logic; and the Ontology (meta) Ontology represents first order logic and model theory. All versions of these meta-ontologies are listed as links in the SUO IFF site map[4].

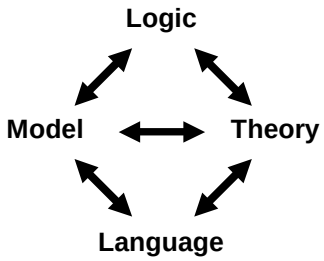


Figure 1: IFF-OO Architecture

The IFF, which is situated at the metalevel, represents form. The ontologies, which are situated at the object level, represent content[§]. By analogy, the content aspect of the SUO is like the various software applications, such as word processors, browsers, spreadsheet software, databases, etc. The distinction between content and form is basic in the general grammar of natural languages, in logic and in ontology. In all of these realms, but especially in logic and ontology, the IFF offers a coherent principled approach to form. Such form is realized in the structuring, mapping and integration of ontologies. The IFF offers axiomatization and techniques for the hierarchical structuring of object-level ontologies via the lattice of theories, the mapping between ontologies via syntax directed translation, and the semantic integration of ontologies via mediating or reference ontologies. To paraphrase John Sowa[5], developing the tools and methodologies for extending, refining, and sharing object-level ontologies is more important than developing the content for those ontologies.

2. Basic Concepts of the IFF-OO

The metalevel axiomatic framework for object-level ontologies represented in first order logic and model theory is concentrated in the lower metalevel IFF Ontology (meta) Ontology (IFF-OO). The IFF-OO is a generic framework for the representation and manipulation of object-level ontologies. The architecture of the IFF-OO (Figure 1) consists of four central mathematical contexts[*] interconnected by five pairs of invertible passages[*]. Each of the four contexts represents a basic concept axiomatized in the IFF-OO. These four concepts are language, theory, model and logic. The context of first order logic *languages*[6] sits at the base of the IFF-OO – everything depends upon it. The three other contexts – models, theories and logics – are situated above the language context. Models provide the interpretive semantics for object-level ontologies, *theories* provide the formal or axiomatic semantics, and logics provide the combined semantics. Any theory is based on a language, and the context of theories is connected to the context of languages by the base passage. An object-level ontology is populated when it has instance data. Unpopulated object-level ontologies are represented by IFF theories, whereas populated object-level ontologies are represented by IFF logics. This paper deals only with formal, axiomatic semantics for object-level ontologies. Interpretive semantics will be combined with this in future work.

The concept of an IFF language is many-sorted – the definition follows (Enderton, 1972), generalizing the standard notion of a single-sorted language. The IFF terminology is somewhat different from Enderton – it uses the two polarities of entities versus relations and instances versus types: an IFF entity type corresponds to a sort, an IFF relation type corresponds to a predicate, and an IFF function type corresponds to a function symbol. In this paper, we ignore function types for simplicity – these are adequately handled in the IFF Algebraic Theory (meta) Ontology. Note that an IFF language deals only with type information. Constants are regarded as nullary function types. Languages are comparable via *language morphisms*, and theories are comparable via *theory morphisms*. Any language $\boldsymbol{L}$ determines a *lattice of theories* $\mathit{fiber}(\boldsymbol{L})$[**], a base passage fiber[††]. Any language morphism $\boldsymbol{f} : \boldsymbol{L}_1 \to \boldsymbol{L}_2$ determines a function $\mathit{expr}(\boldsymbol{f}) : \mathit{expr}(\boldsymbol{L}_1) \to \mathit{expr}(\boldsymbol{L}_2)$ by induction, and from this a *lattice morphism of theories*

$$\mathit{fiber}(\boldsymbol{f}) = \langle \mathit{inv}(\boldsymbol{f}), \mathit{dir}^{\exists} \rangle : \mathit{fiber}(\boldsymbol{L}_2) \to \mathit{fiber}(\boldsymbol{L}_1),$$

[‡] A module in the IFF lower metalevel should represent a well-researched area. In addition to the IFF-OO, which represents first order logic and model theory, other non-core lower metalevel modules are also being considered: a module for the "soft computation" of both rough sets and fuzzy logic; a module for theories of semiotics; a module for game-theoretic semantics; etc.

[§] Many current object-level ontologies contain generic axiomatizations for notions such as binary relations, partial orders, etc. In the IFF, these are not needed, since such axiomatizations are included in the Lower Core (meta) Ontology, etc. When compliant with the IFF, object-level ontologies can concentrate on their core axiomatics.

[**] The *lattice of theories* $\mathit{fiber}(\boldsymbol{L})$ for a language $\boldsymbol{L}$ is the complete lattice of all theories with base language $\boldsymbol{L}$ using entailment order between theories: $\boldsymbol{T}_2 \leq \boldsymbol{T}_1$ means that $\boldsymbol{T}_2$ is more specialized than $\boldsymbol{T}_1$ in the sense that $\boldsymbol{T}_1$ is contained in the closure of $\boldsymbol{T}_2$; or equivalently, that any theorem of $\boldsymbol{T}_1$ is entailed by the axioms of $\boldsymbol{T}_2$.

[††] A *fiber* of a passage $\boldsymbol{P} : \mathbf{C} \to \mathbf{B}$ for fixed object $b \in \mathbf{B}$ is analogous to the inverse image of b along $\boldsymbol{P}$, thus forming the sub-context $\mathit{fiber}_{\boldsymbol{P}}(b) \subseteq \mathbf{C}$ of all $\mathbf{C}$-objects that map to b and all $\mathbf{C}$-morphisms that map to the identity at b.

the fiber invertible passages of direct/inverse image operators – the (*existential*) *direct image* operator

$$dir^{\exists}(f) = (\exists expr(f))^{op} : fiber(L_1) \rightarrow fiber(L_2)^{\ddagger\ddagger}$$

and the *inverse image* operator

$$inv(f) = (expr(f)^{-1})^{op} : fiber(L_2) \rightarrow fiber(L_1).$$

The mapping of unpopulated object-level ontologies is represented by IFF language/theory morphisms. In particular, the IFF represents ontology mapping as the movement of theories back and forth between lattices of theories by using the above lattice morphism of theories over a language morphism.

A recent vote by the SUO Working Group approved a proposal by John Sowa to develop a library of modules structured in a hierarchy. This library of modules will include modules derived from other object-level ontologies. The hierarchical structure framing such a library of modules is a lattice of theories. Sowa has offered a step-wise approach for building a library of modules[7]. However, the processing involved here can be applied to any system of ontologies, and each step of Sowa's process of "building the hierarchy" is represented in the IFF. To do this we represent a module as an IFF theory. A library of modules, regarded as a generalization-specialization hierarchy, is conceptually situated within the context of a lattice of theories[**] and its correlated structure known as the *truth concept lattice*[§§]. In the IFF, an unpopulated monolithic object-level ontology is represented as an IFF theory, the same as a module. The IFF regards a library of modules to be an unpopulated modularized object-level ontology. This is represented in the IFF as a *diagram of theories*[***]. In other terminology, an IFF diagram of theories represents a system of object-level ontologies. Diagrams of theories are comparable via *theory diagram morphisms*[6]. Any diagram of theories $\boldsymbol{T}$ indexed by a shape graph $\boldsymbol{G}$ has a base diagram of languages $\boldsymbol{L} = base(\boldsymbol{T})$ of the same shape, where the language (language morphism) at any indexing node (edge) of graph $\boldsymbol{G}$ is the underlying base language (language morphism) of the theory (theory morphism) at that node (edge). Generalizing the fiber over a language, any language diagram $\boldsymbol{L} : \boldsymbol{G} \rightarrow |\mathsf{Language}|$ determines a *lattice of theory diagrams* $fiber(\boldsymbol{L})$[6]. Generalizing the fiber adjoint pair over a language morphism, any language diagram morphism φ determines a *lattice morphism of theory diagrams* $fiber(\varphi)$[6].

3. Fusion of a System of Ontologies

The IFF can utilize the fusion construction[*] in various mathematical contexts. Since this paper only discusses the formal, axiomatic semantics of integration, here we limit ourselves to the fusion construction for languages and theories. The fusion of theories is defined in terms of the fusion of languages (Table 1).

Table 1: The Fusion Construction[†††]

1. Informally, identify the theories to be used in the construction.
2. Formally, create a diagram of theories $\boldsymbol{T}$ of shape (indexing) graph $\boldsymbol{G}$ that indicates this selection. This diagram of theories is transient, since it will be used only for this computation. Other diagrams could be used for other fusion constructions.
3. Form the fusion theory $\boldsymbol{T}^{\cdot} = \sum \boldsymbol{T}$ of this diagram of theories, with theory fusion cocone $\tau : \boldsymbol{T} \Rightarrow \boldsymbol{T}^{\cdot}$.
   a. Compute the base diagram of languages $\boldsymbol{L} = base(\boldsymbol{T})$ with the same shape. In more detail, $\boldsymbol{L} = base(\boldsymbol{T})$ $= \{\boldsymbol{L}_n\} + \{\boldsymbol{L}_e : \boldsymbol{L}_m \rightarrow \boldsymbol{L}_n\}$ $= \{base(\boldsymbol{T}_n)\} + \{base(\boldsymbol{T}_e) : base(\boldsymbol{T}_m) \rightarrow base(\boldsymbol{T}_n)\}$.
   b. Form the fusion language $\boldsymbol{Ł} = \sum \boldsymbol{L}$ of this diagram, with language fusion cocone $\lambda : \boldsymbol{L} \Rightarrow \boldsymbol{Ł}$. In more detail, $\lambda = \{\lambda_n : \boldsymbol{L}_n \rightarrow \boldsymbol{Ł}\}$, satisfying the conditions $\lambda_m = \boldsymbol{L}_e \cdot \lambda_n$ for $\boldsymbol{G}$-edge $e : m \rightarrow n$.
   c. Move (the individual theories $\{\boldsymbol{T}_n\}$ in) the diagram of theories $\boldsymbol{T}$ from the lattice of theory diagrams $fiber(\boldsymbol{L})$ along the language morphisms in the fusion cocone $\lambda : \boldsymbol{L} \Rightarrow \boldsymbol{Ł}$ to the lattice of theories $fiber(\boldsymbol{Ł})$ using the direct image function, getting the homogeneous diagram of theories $dir(\lambda)(\boldsymbol{T})$ with the same shape $\boldsymbol{G}$, where each theory $dir(\lambda)(\boldsymbol{T})_n = dir(\lambda_n)(\boldsymbol{T}_n)$ has the same base language $\boldsymbol{Ł}$ (the meaning of homogeneous).
   d. Compute the meet (union) of the diagram $dir(\lambda)(\boldsymbol{T})$ within the lattice $fiber(\boldsymbol{Ł})$ getting the fusion theory $\boldsymbol{T}^{\cdot} = \sum \boldsymbol{T} = meet(\boldsymbol{Ł})(dir(\lambda)(\boldsymbol{T}))$.
   e. The language fusion cocone is the base of the theory fusion cocone $\lambda = base(\tau) : base(\boldsymbol{T}) \Rightarrow base(\boldsymbol{T}^{\cdot})$.

As mentioned before, any diagram of theories $\boldsymbol{T}$ has a base diagram of languages $\boldsymbol{L} = base(\boldsymbol{T})$ of the same shape. It is important to note that the indexed theories within $\boldsymbol{T}$ do not necessarily have the same base language. To semantically compare these theories and to conceptually situate them within a lattice of theories, we move them to the lattice of theories over the fusion language $\boldsymbol{Ł} = \sum \boldsymbol{L}$, with this movement guided along the language morphisms in the fusion cocone $\lambda : \boldsymbol{L} \Rightarrow \boldsymbol{Ł}$. The latter is a $node(\boldsymbol{G})$-indexed collection of language morphisms, whose source is the language diagram $\boldsymbol{L}$ and whose target is the language $\boldsymbol{Ł}$. For any diagram of theories $\boldsymbol{T}$ in $fiber(\boldsymbol{L})$, the *direct image fiber* operator $dir(\lambda)$ moves $\boldsymbol{T}$ along the fusion cocone to

[‡‡] In the following, we abbreviate this as $dir(f) = dir^{\exists}(f)$.

[§§] Intuitively, the truth concept lattice is the lattice of closed theories. The lattice order is reverse subset inclusion. The truth concept lattice is the concept lattice for the *truth classification*, the fundamental example 4.6 introduced in (Barwise and Seligman, 1997).

[***] A *diagram of theories* $\boldsymbol{T} : \boldsymbol{G} \rightarrow |\mathsf{Theory}|$ consists of two collections, theories and theory morphisms, indexed by a *shape* graph $\boldsymbol{G}$: each $\boldsymbol{G}$-node n indexes a theory $\boldsymbol{T}_n$ and each $\boldsymbol{G}$-edge $e : m \rightarrow n$ indexes a theory morphism $\boldsymbol{T}_e : \boldsymbol{T}_m \rightarrow \boldsymbol{T}_n$. The size of a diagram corresponds to the cardinality of the node and edge sets of its shape graph. Although these can be infinite, in most practical situations they are finite – there are empty diagrams, single theory diagrams, diagrams with only two theories and one theory morphism, etc.

[†††] The two operations of (1) forming *sums* of theories and (2) specifying *endorelations* and then computing their *quotients*, offer an alternate method for the fusion construction of diagrams of theories: *coequalizers* of theories can be constructed as quotients of endorelations; and *pushouts* of theories can be constructed in terms of sums of components and then quotients of endorelations.

$dir(\lambda)(\boldsymbol{T})$, a homogeneous diagram of shape $\boldsymbol{G}$ in the lattice of theories over $\boldsymbol{L}$. Homogeneous means that all the indexed theories in $dir(\lambda)(\boldsymbol{T})$ have the same base language $\boldsymbol{L}$, and hence can be semantically compared via the theory entailment order. The fusion of the diagram of theories $\boldsymbol{T}$ resolves into $\sum\boldsymbol{T} = meet(\boldsymbol{L})(dir(\lambda)(\boldsymbol{T}))$ – the fiber direct image $dir(\lambda)$ along the base diagram fusion cocone, followed by the meet $meet(\boldsymbol{L})$ in the lattice of theories over $\boldsymbol{L}$, the base diagram fusion language.

Two new ideas have emerged recently in the discussion of the SUO Working Group: the idea of a polycosmos and the idea of mapping closure. Both of these ideas are important in the theory of semantic integration. However, it was not possible to succinctly express these ideas without the use of theory fusions.

- The idea of a *polycosmos*[‡‡‡] was first expressed[8] by Patrick Cassidy: a polycosmos is an unpopulated modular object-level "ontology that has a provision for alternative possible worlds, and includes some alternative logically contradictory theories as applying to alternative possible worlds". The mathematical formulation of polycosmic[9] was immediately given by the author in terms of the fusion of a diagram of theories. A diagram of theories $\boldsymbol{T}$ is *monocosmic* when the fusion theory $\sum\boldsymbol{T}$ is consistent. A diagram of theories $\boldsymbol{T}$ is *pointwise consistent* when each indexed theory in $dir(\lambda)(\boldsymbol{T})$ is consistent. A monocosmic diagram of theories is pointwise consistent by default. A diagram of theories $\boldsymbol{T}$ is *polycosmic* when it is pointwise consistent, but not monocosmic; that is, when there are (at least) two consistent but mutually inconsistent theories in $dir(\lambda)(\boldsymbol{T})$. In the IFF[§§§], there are some extreme polycosmic diagrams of theories, where any two theories are either equivalent or mutually inconsistent. Each of the theories in these diagrams lies at the lowest level in the lattice of theories, strictly above the bottom inconsistent theory containing all expressions.
- The idea of *mapping closure* was first expressed[10] by the author. Any mapping of ontologies involves this notion of mapping closure. For any morphism of languages $\boldsymbol{f} : \boldsymbol{L}_1 \to \boldsymbol{L}_2$, the mapping closure of $\boldsymbol{f}$ applied to any source theory $\boldsymbol{T}_1 \in fiber(\boldsymbol{L}_1)$ is the closure associated with the fiber adjoint pair: $clo(\boldsymbol{f})(\boldsymbol{T}_1) = inv(\boldsymbol{f})(dir(\boldsymbol{f})(\boldsymbol{T}_1))$. Since language morphisms and endorelations are in a sense equivalent[****], the idea of mapping closure is also induced by a language endorelation. An endorelation based on a language $\boldsymbol{L}$ defines by induction an equivalence relation on variables, entity types, relation types and expressions. One expression is equivalent to another expression when the constituent terms[††††] in each are equivalent. Any expression that is equivalent to a theorem of a theory $\boldsymbol{T} \in fiber(\boldsymbol{L})$ is included in the mapping closure[‡‡‡‡].

  Any morphism of languages $\boldsymbol{f} : \boldsymbol{L}_1 \to \boldsymbol{L}_2$ determines a *lattice morphism of theories*

$$\langle dir^{\forall}(\boldsymbol{f}), inv(\boldsymbol{f})\rangle : fiber(\boldsymbol{L}_1) \to fiber(\boldsymbol{L}_2)$$

  with the (*universal*) *direct image* operator

$$dir^{\forall}(\boldsymbol{f}) = \forall expr(\boldsymbol{f})^{op} : fiber(\boldsymbol{L}_1) \to fiber(\boldsymbol{L}_2)$$

  and the *inverse image* operator. In summary, for any morphism of languages $\boldsymbol{f} : \boldsymbol{L}_1 \to \boldsymbol{L}_2$ there are two linked pairs of invertible monotonic functions:

$$dir^{\forall}(\boldsymbol{f}) \dashv inv(\boldsymbol{f}) \dashv dir^{\exists},$$

  with $dir^{\forall}(\boldsymbol{f})$ and $inv(\boldsymbol{f})$ preserving joins (intersections), and $inv(\boldsymbol{f})$ and $dir(\boldsymbol{f}) = dir^{\exists}$ preserving meets (unions). Two questions arise. (1) What is the significance of the mapping closure? (2) Which quantificational direct image operator should be used for moving theories? In the IFF view, mapping theories along a language morphism requires a commitment to mapping closure. In other words, if one is willing to use a language morphism to map a theory, then one is committing oneself to the mapping closure of that theory; that is, one is essentially asserting all of the additional axioms in the difference between the theory and its mapping closure. The existential direct image operator is seen to be important by its use in the fusion construction. However, what about the universal direct image operator? The fact is that the two operators are identical on the mapping closure of a theory. Hence, if we commit ourselves to the mapping closure of a theory, it does not matter which direct image operator we use, since they are both equal in this case.

4. Maintenance of a System of Ontologies

This section discusses how the notions of modularity and centralization are represented in the IFF. As the author has discussed[11] and demonstrated[12], each step of Sowa's process of "building the hierarchy"[7] is represented in the IFF. All steps take place in the context of theories. However, in the general maintenance of a diagram of theories, these processing steps can be used in any fashion deemed necessary. The following are various operations that are

[‡‡‡] According to the dictionary, a cosmos is an orderly harmonious systematic universe.

[§§§] Since IFF models have a set of tuples (= relation instances) as one component, they are more refined than traditional model-theoretic structures and are better able to represent the intuitive notion of context – some IFF models even have only one tuple.

[****] Any language morphism has a kernel (equivalence) endorelation based on the source language, where two source types are equivalent when they are mapped to the same target type. Conversely, any language endorelation generates an epimorphic language morphism onto the quotient language of the endorelation.

[††††] By terms, we mean the variables and the entity, relation and function types used in the language $\boldsymbol{L}$. Constants are nullary function symbols.

[‡‡‡‡] The IFF notion of language endorelation is a theory of relative synonymy – synonymy relative to the base language, and hence relative to the conceptual structures of whatever community owns and manages the corresponding ontology. Such a theory of relative synonymy may be related to any linguistic/philosophical discussion of synonymy, such as (Quine, 1951).

possible in the IFF in order to practically maintain a diagram of theories.

- **Consistency checking:** Any theory in a homogeneous diagram of theories may be inconsistent (equivalent to the bottom of the lattice of theories). A basic and non-trivial operation is to check for the consistency of the indexed theories in a diagram. Of course, any theory that comes with its own special model is already consistent.
- **Sum theory:** This is a procedure for distinguishing the various terms used in a discrete diagram of theories. Every theory in such a diagram has a unique theory index, and all terms in the standard theory sum are distinguished by 'labeling" with the index of their theory of origin. This is the process of forming the sum in the context of theories and the underlying context of languages.
- **Endorelation and Quotient theory:** The quotient of a theory is based upon an endorelation over that theory[§§§§]. The identification of pairs of terms[††††] corresponds to the mathematical process of forming the *quotient* of the sets of terms in a theory via a suitable *endorelation*. This is the process of forming the quotient in the context of theories and the underlying context of type languages.
- **Subtheory:** Often it is helpful in maintaining a diagram of theories to extract smaller (and hence more generic) subtheories from larger more specific ones. This makes the diagram of theories more flexible to use. In particular, when fusing theories, one may need to only use some smaller more generic parts. Each extracted theory is more general than its theory of origin, and thus higher in the lattice hierarchy.
- **Alignment:** For alignment in particular and integration in general, we follow the definitions of the ontology working group of the NCITS T2 Committee on Information Interchange and Interpretation as recorded by Sowa[3]. Ontological alignment consists of the sharing of common terminology and semantics through a mediating or reference ontology (Kent, 2000). The intent of alignment is that mapped types are equivalent. Such equivalence can be automatically computed via the FCA-Merge process (Stumme and Mädche, 2001)[*****]. To formalize this, we represent an equivalence pair of types as a single type in a mediating or reference theory, with two mappings from this new type back to the participant theory types. Thus, alignment is represented as a span or 'Λ'-shaped diagram of three theories and two theory morphisms. The mediating or reference ontology in the middle represents both the equivalenced types and the axiomatization needed for the desired degree of compatibility with the participant ontologies, whether partial or complete. Since the theoretical alignment links preserve this axiomatization, compatibility will be enforced[†††††].
- **Sum diagram:** Given two diagrams of theories $\boldsymbol{T}_1$ and $\boldsymbol{T}_2$ of shapes $\boldsymbol{G}_1$ and $\boldsymbol{G}_2$, respectively, the sum diagram of theories $\boldsymbol{T} = [\boldsymbol{T}_1, \boldsymbol{T}_2]$ has the sum shape $\boldsymbol{G}_1 + \boldsymbol{G}_2$ with object function $obj(\boldsymbol{T})$ that maps nodes in $node(\boldsymbol{G}_1 + \boldsymbol{G}_2) = node(\boldsymbol{G}_1) + node(\boldsymbol{G}_2)$ according to component: $obj(\boldsymbol{T})(n_1) = obj(\boldsymbol{T}_1)(n_1)$ and $obj(\boldsymbol{T})(n_2) = obj(\boldsymbol{T}_2)(n_2)$; similarly for edges.
- **Removal:** Any theory in a diagram might be mark for deletion for various reasons – the theory may have been proven inconsistent, or the theory may no longer be of interest to the community federation maintaining the system of ontologies.
- **Fusion (or Unification):** It may be desirable at any time to create a customized theory. One example of such a customized theory is a "great big hierarchy with modules copied in, frozen into place, and relabeled to avoid inconsistencies" as described[13] by John Sowa. This is built as the fusion construction of a sub-diagram of theories (Table 1). The fusion $\boldsymbol{T}\bullet$, the desired theory to be constructed, is just another theory. The other theories in the diagram being maintained have been left in place undisturbed. Forming the meet is a special case of the fusion construction for a homogeneous sub-diagram of theories.
- **Theory Creation:** Often a small theory of specialized axioms is needed. This may occur when defining a customized theory as the fusion of a diagram with the small theory as one indexed component.

5. Future Prospects

Full semantic integration involves the notion of information flow (Barwise and Seligman, 1997). Special cases of this have appeared in the papers (Kent, 2000 and 2003), (Kalfoglou and Schorlemmer, 2002) and (Schorlemmer and Kalfoglou, 2003). In particular, the papers by the author argue that the semantic integration of ontologies is the two-step process of alignment and unification. Ontological

[§§§§] This is a systematic procedure for specifying the pairs of terms to be semantically identified. One can assume that the terms in the sum of a (discrete) diagram of theories are coordinated with one another in the following sense. In a theory sum, (1) any two terms from independently developed component theories should not be identified; however, (2) two identical terms from different component theories should be identified if these theories originated by subsetting from a third more specialized theory.

[*****] In fact, although we recognize that it can serendipitously discover new relationships, we view FCA-Merge as predominately an automatic process for ontology alignment. It is important to note that FCA-Merge requires interpretative or combined semantics, since it crucially depends upon instance data and classifications. Hence, this approach to alignment uses logics, not just theories.

[†††††] In general, alignment acts through community ontology port(al)s. Before two ontologies can be aligned, it may be necessary to introduce new subtypes or supertypes of terms in either ontology in order to provide suitable targets for alignment. In addition, when any participant ontology has some distinct instance data, alignment may quotient that participant. Hence, alignment is represented by a 'W'-shaped diagram, with the original participating ontologies at the two upper outer vertices, the mediating or reference ontology at the upper center vertex, and the participant port(al) ontologies at the two lower vertices.

alignment consists of the sharing of common terminology and semantics through a mediating or reference ontology. Ontological unification, concentrated in a virtual ontology of community connections, is fusion of the alignment diagram of participant community ontologies – the quotient of the sum of the participant port(al)s modulo the ontological alignment structure. The current paper contributes to this "information flow approach to semantic integration" by describing how the IFF represents formal semantic integration through its general fusion construction and situates formal semantic integration in the on-going maintenance of a system of ontologies. However, true information flow, and hence combined semantic integration, both formal and interpretive, occurs at the level of logics. The correct formulation of this requires the notion of free logics and the notion of fusions of logics. The current version of the IFF-OO has axiomatizations for free logics and for fusions of theories. However, fusions of logics cannot be constructed. The problem is that the current version of the IFF-OO follows too closely Enderton's notion of a sorted language. In particular, IFF languages using reference functions (sort functions in Enderton's terminology) cause problems when trying to construct the coproduct of models or logics. This has been remedied in the new version of the IFF-OO to be posted soon. See the discussion of the "IFF Work in Progress"[14] for more on this.

In summary, we argue that the principled framework of the IFF realizes the information flow approach to semantic integration, and we hope that this theoretical approach and its implementation[‡‡‡‡‡] will contribute to realization of the "gold standard for semantic integration" (Uschold and Gruninger, 2002).

6. References

Barwise, Jon and Jerry Seligman. Information Flow: The Logic of Distributed Systems. Cambridge Tracts in Theoretical Computer Science 44. Cambridge University Press. 1997.

Enderton, Herbert B. A Mathematical Introduction to Logic. New York: Academic Press. 1972.

Ganter, Bernhard and Rudolf Wille. Formal concept analysis: mathematical foundations. Heidelberg: Springer. 1999.

Kalfoglou, Yannis and Marco Schorlemmer. Information-Flow-based Ontology Mapping. On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE, Lecture Notes in Computer Science 2519, 1132–1151. Springer. 2002.

Kent, Robert E. The Information Flow Foundation for Conceptual Knowledge Organization. In: Dynamism and Stability in Knowledge Organization. Proceedings of the Sixth International ISKO Conference. Advances in Knowledge Organization 7, 111–117. Ergon Verlag, Würzburg. 2000.

Kent, Robert E. The IFF Foundation for Ontological Knowledge Organization. In: Knowledge Organization and Classification in International Information Retrieval. Cataloging and Classification Quarterly. The Haworth Press Inc., Binghamton, New York. 2003.

Mac Lane, Saunders. Categories for the Working Mathematician, New York/Heidelberg/Berlin: Springer-Verlag. 1971. (New edition 1998).

Quine, Willard V.O. Two Dogmas of Empiricism. Philosophical Review. 1951.

Schorlemmer, Marco and Yannis Kalfoglou. Using Information-Flow Theory to Enable Semantic Interoperability. Sisè Congrés Català d'Intelligència Artificial. Palma de Mallorca, 2003.

Sowa, John F. Knowledge Representation: Logical, Philosophical and Computational Foundations. Brookes/Coles. 2000.

Stumme, Gerd and Alexander Mädche. FCA-Merge: Bottom-Up Merging of Ontologies. In: B. Nebel (Ed.): Proc. 17th Intl. Conf. on Artificial Intelligence (IJCAI '01). Seattle, WA, USA, 2001, 225–230. Long version: Ontology Merging for Federated Ontologies for the Semantic Web. In: E. Franconi, K. Barker, D. Calvanese (Eds.): Proc. Intl. Workshop on Foundations of Models for Information Integration (FMII'01), Viterbo, Italy, Sept. 16–18, 2001. LNAI, Springer 2002.

Uschold, Michael and Michael Gruninger. Creating Semantically Integrated Communities on the World Wide Web. Invited paper. International Workshop on the Semantic Web. 2002.

‡‡‡‡‡ The IFF takes the high road to implementation. There is work in progress on an IFF representation of the Meta Object Facility (MOF) and the Model Driven Architecture (MDA) of the Object Management Group (OMG). In the other direction, there are on-going explorations to demonstrate how the MOF can be used for a high level specification of the IFF.

[1] "IEEE P1600.1 Standard Upper Ontology (SUO) Working Group" web site. [http://suo.ieee.org/].

[2] "The SUO Information Flow Framework (SUO IFF)" main web page. [http://suo.ieee.org/IFF/].

[3] Sowa, John F. Building, Sharing, and Merging Ontologies. Unpublished paper. (2001) [http://www.jfsowa.com/ontology/ontoshar.htm].

[4] "The SUO IFF Site Map" web page. [http://suo.ieee.org/IFF/site-map.html].

[5] Sowa, John F. "Reply: Ontology Structure & Content". SUO list message. (13 January 2001) [http://grouper.ieee.org/groups/suo/email/msg02765.html].

[6] Kent, Robert E. "IFF Lattice of Theories (LOT) Glossary" Unpublished document. [http://suo.ieee.org/IFF/LOT-glossary.pdf].

[7] Sowa, John F. "Building the hierarchy". SUO list message. (16 May 2003) [http://suo.ieee.org/email/msg09453.html].

[8] Cassidy, Patrick. "Reply: "\SUO Ballot with 2 Questions – 'monolithic'?". SUO list message. (10 Jun 2003) [http://grouper.ieee.org/groups/suo/email/msg09701.html].

[9] Kent, Robert E. "Mathematical Definitions for Monocosmic and Polycosmic". SUO list message. (14 Jun 2003) [http://grouper.ieee.org/groups/suo/email/msg09768.html].

[10] Kent, Robert E. "Mapping Closure". SUO list message. (5 Jul 2003) [http://grouper.ieee.org/groups/suo/email/msg10356.html].

[11] Kent, Robert E. "Building the Hierarchy by Language and Module Processing". SUO list message. (2 Jun 2003) [http://grouper.ieee.org/groups/suo/email/msg09555.html].

[12] Kent, Robert E. "Language and Module Processing" Unpublished document – process diagram. (31 May 2003) [http://suo.ieee.org/IFF/language-module-processing.html].

[13] Sowa, John F. "Reply: Charter vs. Consensus". SUO list message. (27 Jun 2003) [http://grouper.ieee.org/groups/suo/email/msg10139.html].

[14] Kent, Robert E. "IFF Work in Progress" web page. (4 April 2003) [http://suo.ieee.org/IFF/work-in-progress.html].

Evolution Management for Interconnected Ontologies

Michel Klein and **Heiner Stuckenschmidt**
Vrije Universiteit Amsterdam
de Boelelaan 1081a, 1081 HV Amsterdam
{michel.klein, heiner}@cs.vu.nl

Abstract

Mappings between ontologies are easily harmed by changes in the ontologies. In this paper we explain a mechanism to define modular ontologies and mappings in a way that allows for local containment of terminological reasoning. We have also developed a change detection and analysis method that predicts the effect of changes on the concept hierarchy. This method determines whether the changes in one ontology affect the reasoning inside other ontologies or not. Together, these mechanisms allow ontologies to evolve without unpredictable effects on other ontologies. In this paper, we also apply these methods in a case study that is undertaken in a EU IST project.

1 Motivation

When mappings are created between ontologies, it is essential that the evolution of ontologies is managed, because a change in one ontology could have extensive effects in other ontologies. This is especially important when ontologies are used as basis for formal reasoning tasks.

To handle this problem, we have developed a mechanism to define modular ontologies and mappings between them that allows for local containment of terminological reasoning [10]. This modularization mechanism makes it possible to perform subsumption reasoning within an ontology without having to access other ontologies. We have also developed a change detection and analysis method that predicts the effect of changes on the concept hierarchy. This method determines whether the changes in one ontology affect the reasoning inside other ontologies or not. Together, these mechanisms allow ontologies to evolve without unpredictable effects on other ontologies.

In this paper, we will show how these methods work in a realistic example. For this we use the case study that is undertaken in the WonderWeb project[1]. We describe the case study and explain the overall approach in the next paragraphs. Section 2 defines the modularization approach in more detail, and shows how we use this to define mappings to the case study ontology. In section 3, we explain the change analysis mechanism and show the results for our example. Finally, in section 4 we conclude with a discussion of open issues and future work.

[1]The WonderWeb project aims at developing scalable infrastructure for the semantic web. For more information, see `http://wonderweb.semanticweb.org/`.

1.1 The WonderWeb Case Study

In the WonderWeb case study, an existing database schema in the Human Resource (HR) domain is used as the basis for an ontology. The first version of the ontology is created by a tool that automatically converts a schema into an ontology [11]. In the next phase, the quality of the ontology is improved by relating this ontology to the foundational ontology DOLCE [5]. First, the HR ontology is aligned with the DOLCE ontology, and in several successive steps the resulting ontology is further refined. During this process, the ontology changes continuously, which causes problems when other ontologies refer to definitions in the evolving ontology. Therefore, in our case study, evolution management is important during the entire life-cycle of the ontology development process.

Besides this DOLCE+HR ontology, we assume that we have another ontology (we call it the *local ontology*) that uses terms and definitions from the evolving DOLCE+HR ontology (the *external ontology*). As an example, we define a very simple ontology about employees (see Figure 1). Our example ontology introduces the concept 'FulltimeEmployee' and defines a superclass 'Employee' and two subclasses 'DepartmentMember' and 'HeadOfDepartment' using terms from the DOLCE+HR ontology.

The specific problem in our case is that the changes in the DOLCE+HR ontology could affect the reasoning in the local ontology. We want to be able to predict whether or not the reasoning in the local ontology is still valid for specific changes in the external ontology.

Changes in DOLCE+HR

The evolution of the DOLCE+HR ontology consisted of several steps, which are prescribed by the DOLCE methodology. Each of these steps involves some typical changes.

In the aligning phase, the concepts and properties in the HR ontology are connected to concepts and properties in the DOLCE ontology via subsumption relations. For example, the concept 'Departments' from the HR ontology is made a subclass of 'Social-Unit' in DOLCE.

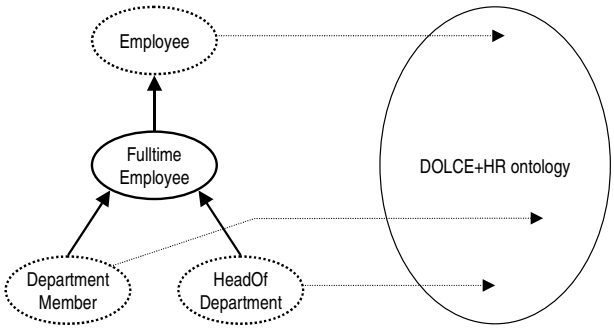


Figure 1: A simple ontology (left) with some concepts (dashed ovals) that are defined using terms from the DOLCE+HR ontology (schematically representation by a large oval).

The refinement step involves a large number of changes. Some property restrictions are added, and some additional concepts and properties are created to define the HR concepts more precisely. For example, the concept 'Administrative-Unit' is introduced as a new subclass of 'Social-Unit', and the concept 'Departments' is made a subclass of it. Also, the range of the property 'email' is restricted from 'Abstract-Region' to its new subclass 'Email'.

In the next step, a number of concepts and properties are renamed to names that better reflect their meaning. For example, 'Departments' is renamed to 'Department' (singular), and the two different variants of the relation 'manager_id' are renamed to 'employee_manager' and 'department_manager'.

In the final step, the tidying-up step, all properties and concepts that are not necessary anymore are removed and transformed into property restrictions. For example, the property 'employee_email' is deleted and replaced by an existential restriction in the class 'Employee' on the property 'abstract_location' to the class 'Email'.

1.2 Approach for Ontology Mappings and Change Management

The main design ideas behind our approach are the following. A detailed description with examples will be given in the next sections.

View-Based Mappings: We adopt the approach of view-based information integration. In particular, ontology modules are connected by conjunctive queries. This way of connecting modules is more expressive than simple one-to-one mappings between concept names but less expressive than the logical language used to describe concepts. We decide to sacrifice a higher expressiveness for the sake of conceptual simplicity and desirable semantic properties such as independence of the ontology langauge used.

Compilation of Implied Knowledge: In order to make local reasoning independent from other modules, we use a knowledge compilation approach. The idea is to compute the result of each mapping query off-line and add the result as an axiom to the ontology module using the result. During reasoning, these axioms replace the query thus enabling local reasoning.

Change Detection and Automatic Update: Once a query has been compiled, the correctness of reasoning can only be guaranteed as long as the concept hierarchy of the queried ontology module does not change. In order to decide whether the compiled axiom is still valid, we propose a change detection mechanism that is based on a taxonomy of ontological changes and their impact of the concept hierarchy.

2 Modular Ontologies

We will now explain the modularization mechanism and the compilation of implied subsumption relations in more detail. In Section 2.3, we show how we use these mechanisms in the case study.

In order to get a general notion of ontological knowledge, we define the general structure of an ontological module and its instantiation independent of a concrete language.

Definition 1 (Ontology Module) *A module is a triple $M = \langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$ where $\mathcal{C}$ is a set of concept definitions, $\mathcal{R}$ is a set of relation definitions and $\mathcal{O}$ is a set of object definitions. Further, we define the signature of a module $\langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$ to be a triple $\langle \mathcal{CN}, \mathcal{RN}, \mathcal{ON} \rangle$, where $\mathcal{CN}$ is the set of all names of concepts defined in $\mathcal{C}$, $\mathcal{RN}$ the set of all relation names in $\mathcal{R}$ and $\mathcal{ON}$ the set of all object names occurring in $\mathcal{O}$.*

2.1 Internal and External Definitions

We divide the set of concepts in a module into internally defined concepts $\mathcal{C}_I$ and externally defined concepts $\mathcal{C}_E$ resulting into the following definition of $\mathcal{C}$:

$$\mathcal{C} = \mathcal{C}_I \cup \mathcal{C}_E,\ \mathcal{C}_I \cap \mathcal{C}_E = \emptyset$$

Internally defined concepts are specified by using class expressions in the spirit of description logics [1]. We do not require a particular logic to be used.

Definition 2 (Internal Concept Definition) *An internal concept definition is an axiom of one of the following forms*

$$C \sqsubseteq D, C \equiv D$$

where $C \in \mathcal{CN}$ and D is a class expression of the form $f(t_1, \cdots, t_n)$ where the terms t_i are either class names or class expressions and f is an n-ary class building operator.

Besides the standard way of defining concepts, we consider externally defined concepts that are assumed to be equivalent to the result of a query posed to another module in the modular ontology. This way of connecting modules is very much in spirit of view-based information integration which is a standard technique in the area of database systems [6].

Definition 3 (External Concept Definition) *An external concept definition is an axiom of the form $C \equiv M : Q$ where M is a module and Q is an ontology-based query over the signature of M.*

A modular ontology is now simply defined as a set of modules that are connected by external concept definitions. In particular we require that all external definitions are contained in the modular system. Queries over ontological knowledge are defined as conjunctive queries, where the conjuncts are predicates that correspond to classes and relations of an ontology. Furthermore, variables in a query may only be instantiated by constants that correspond to objects in that ontology.

Definition 4 (Ontology-Based Queries) *Let V be a set of variables disjoint from $\mathcal{ON}$ then an ontology-based query Q over a module $M = \langle \mathcal{C}, \mathcal{R}, \mathcal{O} \rangle$ is an expressions of the form $Q(\bar{X}) \leftarrow q_{1_i} \wedge \cdots \wedge q_{m_i}$ where q_i are query terms of the form $x : c$ or $(x, y) : r$ such that $x, y \in V \cup \mathcal{ON}$, $c \in \mathcal{CN}$ and $r \in \mathcal{RN}$ or are of the form $x = o$ where $x \in V$ and $o \in \mathcal{ON}$*[2].

The fact that all conjuncts relate to elements of the ontology allows us to determine the answer to ontology-based queries in terms of instantiations of the query that are logical consequences of the knowledge base.

2.2 Compilation and Local Reasoning

We now turn our attention to the issue of reasoning in modular ontologies. For the sake of simplicity, we only consider the interaction between two modules in order to clarify the basic principles. Furthermore, we assume that only one of the two modules contains externally defined concepts in terms of queries to the other module.

Implied Subsumption As mentioned in the introduction, we are interested in the possibility of performing local reasoning. For the case of ontological reasoning, we focus on the task of deriving implied subsumption relations between concepts within a single module. For the case of internally defined concepts, this can be done using well established reasoning methods [3]. Externally defined concepts, however, cause problems: being defined in terms of a query to the other module, a local reasoning procedure will often fail to recognize an implied subsumption relation between these concepts. Consequently, subsumption between externally defined concepts requires reasoning in the external module as the following theorem shows.

Theorem 1 (Implied Subsumption) *Let E_1 and E_2 be two concepts in module M_i that are externally defined in module M_j by queries Q_1 and Q_2, then $E_1 \sqsubseteq E_2$ if $Q_1 \sqsubseteq Q_2$ in module M_j.*

The result presented above implies the necessity to decide subsumption between conjunctive queries in order to identify implied subsumption relations between externally defined concepts. In order to decide subsumption between queries, we translate them into internally defined concepts in the module they refer to. A corresponding sound and complete translation is described in [7]. Using the resulting concept definition, to which we refer as *query concepts*, we can decide subsumption between externally defined concepts by local reasoning in the external ontology.

Compilation and Integrity We can avoid the need to perform reasoning in external modules each time we perform reasoning in a local module using the idea of knowledge compilation [2]. The idea of compilation is to perform the external reasoning once and add the derived subsumption relations as axioms to the local module. These new axioms can then be used for reasoning instead of the external definitions of concepts. If we want to use the compiled axioms instead of external definitions, we have to make sure that this will not invalidate the correctness of reasoning results. At the time of applying the compilation this is guaranteed by theorem 1, however, integrity cannot be guaranteed over the complete life-cycle of the modular ontology. The problem is, that changes to the external ontology module can invalidate the compiled subsumption relationships. In this case, we have to perform an update of the compiled knowledge.

2.3 Modularization and Local Reasoning in the Case Study

If we now consider the problem statement from the case study, we have a local ontology with a concept hierarchy that is built up by the following explicitly stated subsumption relations (see Figure 1 again):

$$\begin{aligned} FulltimeEmployee &\sqsubseteq Employee \\ DepartmentMember &\sqsubseteq FulltimeEmployee \\ HeadOfDepartment &\sqsubseteq FulltimeEmployee \end{aligned}$$

This ontology introduces 'Full time employee' as a new concept, not present in the case study ontology. Consequently, this concept is only defined in terms of its relation to other concepts in the local ontology.

All other concepts are externally defined in terms of ontology based queries over the case study ontology. The first external definition concerns the concept 'Employee' that is equivalent to the 'Employee' concept in the case study ontology. This can be defined by the following trivial view:

$$Employee \equiv HR : Employee(x)$$

Another concept that is externally defined is the 'Head of Department' concept. We define it to be the set of all instances that are in the range of the 'department manager' relation. The definition of this view given below shows that our approach is flexible enough to define concepts in terms of relations.

$$HeadOfDepartment \equiv HR : \exists y[departmentManager(y, x)]$$

An example for a more complex external concept definition is the concept 'department member' which is defined using a query that consists of three conjuncts, claiming that a department is an employee that is in the has_member relation with a Department.

$$DepartmentMember \equiv HR : \exists y[Department(y) \wedge has\_member(y, x) \wedge Employee(x)]$$

[2]Note that this may include data-type expressions as the type itself is can be considered to be a class, the actual value an instance of that class and the comparison operator a special relation.

Implied subsumption relations

If we now consider logical reasoning about these external definitions, we immediately see that the definition of Employee subsumes the definition of DepartmentMember, as the former occurs as part of the definition of the latter.

$$\models DepartmentMember \quad \sqsubseteq \quad Employee \qquad (1)$$

At a first glance, there is no relation between the definition of a Head of Department and the other two statements as it does not use any of the concept- or relation names. However, when we use the background knowledge provided by the external ontology we can derive some implied subsumption relations. The reasoning is as follows. Because the range of the department_manger is set to 'Department' and the domain to 'Manager', the definition of HeadofDepartment is equivalent to:

$$\exists y[Department(y) \wedge department\_manager(y, x) \wedge Manager(x)]$$

As we further know that Manager is a subclass of Employee and department_manager is a sub-relation of has_member, we can derive the following subsumption relation between the externally defined concepts:

$$\models HeadOfDepartment \quad \sqsubseteq \quad Employee \qquad (2)$$
$$\models HeadOfDepartment \quad \sqsubseteq \quad DepartmentMember \qquad (3)$$

When the relations 1–3 are added to the local ontology, it possible to do subsumption reasoning without having to access the DOLCE+HR ontology anymore.

3 Change Detection and Analysis

The changes in the DOLCE+HR ontology could invalidate the local reasoning. In principle, testing the integrity of the mappings might be very costly as it requires reasoning within the external ontology. In order to avoid this, we propose a heuristic change detection procedure that analyzes changes with respect to their impact on compiled subsumption relations, i.e. relations 1–3 from the previous section. This is a three-steps procedure: 1) find out what the differences are between two distinct versions of the ontology, 2) characterize the effect of these changes on individual concepts, and 3) determine the impact of changes of individual concepts on the compiled subsumption relations. The next sections describe these steps.

3.1 Finding Changes

To find changes in ontologies, we have developed a mechanism and a tool to compare ontologies. This change detection mechanism is described in [8]. The algorithm that we developed works for all ontology languages that can be represented in the RDF data model [9], including RDF Schema and OWL. For each changed definition, it produces a list of change operations that are necessary to transform the old version into the new version.

To standardize the description of changes, we have developed an ontology of all possible change operations for an OWL-lite ontology. An actual description of a change between two versions of an ontology can be seen as an instantiation of the ontology of change operations. The change ontology is extendable to other knowledge models. We have chosen the OWL-Lite model because of its simplicity and the central role of OWL in the WonderWeb project. A snapshot of the change ontology can be found online.[3]

Apart from *atomic change operations* — like add range restriction or delete subclass relation — our change ontology also contains some *complex change operations*, which consist of multiple atomic operations and/or incorporate some additional knowledge. The complex changes are often more useful to specify effects than the basic changes. For example, for operations like concept moved down, or range restricted, we can specify the effect more accurately than for the atomic operations subclass relation changed and domain modified.

The case study ontology in our example is expressed in OWL-Lite, which is based on RDF. Therefore, we can use rule-based change detection mechanism. If we look at the changes in the definition of 'Departments', we see that three things happened:

- the comment is reformulated,
- the superclass is changed from 'Social-Unit' to 'Administrative-Unit', and
- there is a property restriction added for 'temporary-component-of' to the class 'Organization'.

This results in three change operations: 1) superclass changed (from 'Social Unit' to 'Administrative-Unit', 2) comment changed, and 3) property restriction added.

3.2 Characterizing Changes

Now we have detected the change operations that are required to transform the old version of the ontology into the new version, we look at the effect of the change operations on individual concepts. Assuming that C represents the concepts under consideration before and C' the concept after the change there are four ways in which the old version C may relate to the new version C':

1. the meaning of concept is not changed: $C \equiv C'$ (e.g. because the change was in another part of the ontology, or because it was only syntactical);
2. the meaning of a concept is changed in such a way that concept becomes more general: $C \sqsubseteq C'$
3. the meaning of a concept is changed in such a way that concept becomes more specific: $C' \sqsubseteq C$
4. the meaning of a concept is changed in such a way that there is no subsumption relationship between C and C'.

We want to know what the effect of specific operations on the interpretation of a concept is (i.e. whether it becomes more general or more specific). As our goal is to determine the integrity of mappings without having to do classification, we describe what theoretically could happen to a concept as result of a modification in the ontology. To do so, we have

[3] `http://ontoview.org/changes/1/3/`

determined the effect for all possible change operations that we distinguish in the 'finding changes' phase.

Table 1 contains some examples of operations and their effect on the classification of concepts. The table only shows a few examples, although our full ontology of change operations contains around 120 operations. This number is still growing as we define new complex changes.

	Operation	**Effect on C**
1	Attach a slot to class C	Specialized
2	*Complex:* Change the superclass of class C to a class lower in the hierarchy	Specialized
3	*Complex:* Restrict the range of a slot S (*effect specified for all classes C that have a slot restriction with S*)	Specialized
4	Remove a superclass relation of a class C	Generalized
5	Change the class definition of C from primitive to defined	Generalized
6	Add a class definition A	Unknown
7	*Complex:* Add a (not further specified) subclass A of C	No effect

Table 1: Some ontology change operations and their effect on the classification of concepts in the hierarchy.

If we apply this to our example, we can only give a useful characterization of the effect to some of the concepts. For example, the concept 'Departments', underwent several changes during the whole process: its superclass has changed to a subclass of the original superclass (change 2 in Table 1) but there are also some property restrictions removed. Both changes have an opposite effect. As a result, we have to characterize the effect of the change as "Unknown". On the contrary, the effect on the relation 'department_manager', is clear: the relation is renamed from 'manager_id' — which has no conceptual effect — and the range is changed from 'Employee' to 'Manager'. Because 'Manager' is a subclass of 'Employee', this change makes it more specific (change 3 in Table 1).

3.3 Update Management

With the elements that we described in this section, we now have a complete procedure to determine whether compiled knowledge in other modules is still valid, and thus whether the mappings are still usable. The complete procedure is as follows:

1. create a list of concepts and relations that are part of the "subsuming" query of any compiled axiom;
2. create another list of concepts and relations that are part of the "subsumed" query of any compiled axiom;
3. achieve the modifications that are performed in the external ontology;
4. use the modifications to determine the effect on the interpretation of the concepts and relations.
5. check whether there are concepts or relations in the first, "subsuming", list that became more specific, or concepts or relations in the second, "subsumed", list that became more general, or concepts or relations in any of the lists with an unknown effect; if not, the integrity of the mapping is preserved.

All the steps can be automated. The tool that we mentioned in the previous section currently helps with steps 3 and 4. It detects the changes between two versions and produces a list op change operations.

We can now use this procedure to check whether the implied subsumption relations in our case study are still valid. For the sake of simplicity, we restrict us here to relation 3:

$$\models HeadOfDepartment \sqsubseteq DepartmentMember$$

For this compiled axiom, the list of 'subsuming' concepts and relations would contain 'Department', 'has_member', and 'Employee', while the list of subsumed concepts and relations would be 'Department', 'department_manager', and 'Manager'.

We will now illustrate that the conclusions of the procedure are correct by studying the impact of changes mentioned in the problem statement.

Example 1: The Employee Concept The first change we observed is the removal of properties from the Employee concept. Our rules tell that this change makes the new version more general compared to its old version:

$$Employee \sqsubseteq Employee'$$

According to our procedure, this shouldn't be a problem because Employee is in the 'subsuming list'.

When we analyze this change, we see that it has an impact on the definition of the concept DepartmentMember as it enlarges the set of objects allowed to take the first place in the has_member relation. This leads to a new definition of $DepartmentMember'$ with $DepartmentMember \sqsubseteq DepartmentMember'$. As DepartmentMember was already more general than HeadOfDepartment and the Employee concept is not used in the definition of the latter the implied subsumption relation indeed still holds.

Example 2: The department_manager Relation The second example, we have to deal with a change affecting a relation that is used in an external definition. The relation department_manager is specialized by restricting its range to a more specific concept making it a subrelation of its previous version:

$$department\_manager \sqsupseteq department\_manager'$$

Again, this is harmless according to our procedure, as department_manager is in the 'subsumed list'.

The analysis shows that this change has an impact on the definition of the concept HeadOfDepartment as it restricts the allowed objects to the more specific Class Manager. The new definition $HeadOfDepartment'$ is more specific that the old one: $HeadOfDepartment' \sqsubseteq HeadOfDepartment$.

As the old version was already more specific than the definition of DepartmentMember and the department_manager relation is not used in the definition of the latter the implied subsumption is indeed still valid.

Example 3: The Department Concept The different changes of the definition of the department concept left us with no clear idea of the relation between the old and the new version. In this specific case, however, we can still make assertions about the impact on implied subsumption relations. The reason is that the concept occurs in both definitions. Moreover, it plays the same role, namely restricting the domain of the relation that connects an organizational unit with the set of objects that make up the externally defined concept. As a consequence, the changes have the same impact on both definitions thus not invalidating the implied subsumption relation. *In summary, an implied subsumption relation is still valid if the changed concept occurs in and plays the same role in both definitions involved.*

4 Discussion

In this paper we discussed the problem of ontology evolution in situations where mappings between ontologies existed. We presented two main contributions towards a better understanding and management of dependencies in the light of changes to an ontology.

- We presented a formal model for describing dependencies between different ontologies. We proposed conjunctive queries for defining concept using elements from another ontology and presented a model-based semantics in the spirit of distributed description logics that provides us with a notion of logical consequence across different ontologies. This clear semantic account of dependence makes it possible to study the impact of changes on a semantic level.
- We described a method for detecting changes in an ontology and for assessing their impact. The main feature of this method is the derivation of conceptual changes from purely syntactic criteria. These conceptual changes in turn provide input for a semantical analysis of the effect on dependent ontologies, in particular on the validity of implied subsumption relations.

The effect analysis procedure that we have proposed uses quite coarse-grained heuristics. As a result, it often concludes that a validity of a subsumption relation cannot be guaranteed, while it is in fact still valid. In order to be able to provide more precise answers we will have to develop a more formal characterization of changes like it has been done in the area of schema evolution for database systems [4]. Based on such a formal characterization, we have to investigate conditions under which implied knowledge is still valid in a more generic way.

References

[1] F. Baader, D. Calvanese, D. McGuiness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook - Theory, Implementation and Applications*. Cambridge University Press, 2003.

[2] M. Cadoli and F. Donini. A survey on knowledge compilation. *AI Communications*, 10(3-4):137–150, 1997.

[3] F. Donini, M. Lenzerini, D. Nardi, and A. Schaerf. Reasoning in description logics. In G. Brewka, editor, *Principles of Knowledge Representation*, Studies in Logic, Language and Information, pages 193–238. CSLI Publications, 1996.

[4] E. Franconi, F. Grandi, and F. Mandreoli. A sematic approach to schema evolution and versioning in object-oriented databases. In *Proceesing of CL 2000*, volume 1861 of *Lecture Notes in Artificial Intelligence*, pages 1048–1062. Springer Verlag, 2000.

[5] A. Gangemi, N. Guarino, C. Masolo, A. Oltramari, and L. Schneider. Sweetening ontologies with DOLCE. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, volume 2473 of *Lecture Notes in Computer Science*, page 166 ff, Sigüenza, Spain, Oct. 1–4, 2002.

[6] A. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[7] I. Horrocks and S. Tessaris. A conjunctive query language for description logic aboxes. In *AAAI/IAAI*, pages 399–404, 2000.

[8] M. Klein, D. Fensel, A. Kiryakov, and D. Ognyanov. Ontology versioning and change detection on the web. In *13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02)*, number 2473 in LNCS, page 197 ff, Sigüenza, Spain, Oct. 1–4, 2002.

[9] O. Lassila and R. R. Swick. Resource Description Framework (RDF): Model and Syntax Specification. Recommendation, World Wide Web Consortium, Feb. 1999. See http://www.w3.org/TR/REC-rdf-syntax/.

[10] H. Stuckenschmidt and M. Klein. Integrity and change in modular ontologies. In *Proceedingso of the 18th International Joint Conference on Artificial Intelligence*, Acapulco, Mexico, Aug. 2003.

[11] R. Volz, D. Oberle, S. Staab, and R. Studer. Ontolift prototype. Deliverable D11, EU/IST Project WonderWeb, 2002.

From an ontology-based search engine towards a more flexible integration for medical and biological information

G. Marquet, C. Golbreich, A. Burgun
Laboratoire d'Informatique Médicale, Faculté de Médecine, 35033 Rennes, France

Christine.Golbreich@uhb.fr {Gwenaelle.Marquet, Anita.Burgun}@univ-rennes1.fr

Abstract. *Better understanding pathologies-genes relationships requires semantic integration of heterogeneous information distributed in multiple 'medical' and 'biological' sources. This paper presents an ongoing project that aims at developing an information integration system providing a unified access to biomedical resources. The basic idea is to use for semantic integration the existing knowledge available in standard domain terminologies e.g. GeneOntology™, UMLS® and databanks e.g. HUGO, GOA. A first tool, BioMeKe, has been achieved in that perspective. BioMeKe is an ontology-based search engine designed to facilitate the extraction and connection of biological and medical information, accessible from multiple public resources and biologists local repositories, for a system devoted to liver transcriptome analysis. The paper presents existing resources, describes BioMeke. Then, general lessons learnt from this practical experience are discussed.*

1 Introduction

The Word-Wide Web has made available a tremendous amount of biomedical information, but it remains tedious and time-consuming for biologists and physicians to access the information relevant to their queries. Multiple public resources are available in genomics including databanks such as SWISS-PROT[1], OMIM[2], LocusLink[3], GenBank[4], as well as many others, and some systems e.g. TAMBIS [27] are being developed to provide transparent access to bioinformatics sources. But a step further is needed for better understanding of the pathological processes that are involved in human diseases. To develop research, suggest new hypotheses about molecular mechanisms of human diseases and take advantage of recent research for patient care (e.g. [7]), biologists and physicians need to access and to relate numerous information from both genomics *and* medicine,. Therefore, tools to acquire and connect relevant data from existing resources are required. The problem is that there is considerable semantic heterogeneity between the sources, both intra and inter-domain. Different resources use different conceptualizations or different terms for the same concept or the same individual, although standard terminologies have been defined for each domainsuch as GeneOntology™ (GO) for molecular biology and genomics, and the Unified Medical Language System® UMLS® for the biomedical domain. The goal of the project is to develop a semantic integration system that offers a uniform interface for querying multiple heterogeneous sources *both* in genomics-molecular biology and in medecine, together with services for combining pieces of medical and molecular biology information relevant to answer queries. The basic idea is to use for semantic integration the knowledge available in the existing standard domain terminologies, namely GO and UMLS® and in databanks. Section 2 presents the main existing terminologies and databanks, section 3 describes BioMeKe (Biological and Medical Knowledge Extraction) [21], an ontology-based tool achieved to facilitate the access and association of knowledge from Web or local resources, for liver transcriptome analysis. Then, general lessons learnt from this practical experience are discussed.

2 Molecular biology and medicine resources

Information in the biomedical domain is scattered through multiple public databanks and bibliographic systems. But for each domain, « ontologies » have been defined to provide a unified and controlled vocabulary.

2.1 Ontologies

- **GeneOntology™** (GO)[5] is an ontology for molecular biology and genomics. GO is organized with three top categories Molecular Function, Biological Process, and Cellular Component. In May 2003 GO contained 7172 processes, 5386 molecular functions and 1265 component concepts. GO itself is not populated with gene products. It provides a controlled vocabulary for annotating sequences and gene products. GO concepts are broadly used as attributes in many public databases e.g. SWISS-PROT, as well as in

[1] http://us.expasy.org/sprot/
[2] http://www.ncbi.nlm.nih.gov/omim/
[3] http://www.ncbi.nlm.nih.gov/LocusLink/
[4] http://www.ncbi.nlm.nih.gov/Genbank/

[5] http://www.geneontology.org

specific applications. In the context of microarray experiments, biologists use GO for annotating the genes they are studying (Table 3).

- The **UMLS®** is a medical ontology intended to help health professionals and researchers use biomedical information from different sources [19]. It has two major components, the Metathesaurus®, a large repository of concepts (900,551 concepts in the 2003AA release), built by merging more than 100 families of vocabularies (including MeSH), and in grouping synonymous terms under a same concept and the Semantic Network, a limited network of 135 semantic types. The Metathesaurus concepts are assigned to one or more semantic types. The Metathesaurus is. In addition to the standard MeSH, the US National Library of Medicine created and maintains the **MeSH-ST**, ST standing forSupplementary Terms, which contains records that cover the fields of chemicals and molecular biology. (134,749 records in the 2003 release). The MeSH-ST files are updated continuously. MeSH-ST terms are integrated in the UMLS, making most of the terms, but not all the information provided by MeSH-ST accessible through the UMLS.

2.2 Multiple heterogeneous public databanks

Multiple public databanks provide information on genes, sequences and proteins, discovered upon a published experiment e.g. **SWISS-PROT** (SW), **GenBank**, **LocusLink, HUGO, G0 Annotation @EBI :**

- **LocusLink**[6] is a genes database to unify kowledge about genes. It provides official nomenclature, aliases, sequence accessions, cross-references to other banks via identifiers (EC Id, MIM Id, etc.).
- **HUGO**[7] (Human Gene Nomenclature Database) provides official gene names e.g. *ferritin, heavy polypeptide 1*, their synonyms, official symbol e.g. *FTH1*, and various links to other databases LocusLink, SWISS-PROT, OMIM, etc. via identifiers (e.g. ID: P02794 , LocusLink ID: 2495, OMIM ID: 134770).
- **GO Annotation @EBI**[8] (GOA) objective is to assign GO terms to *gene products*. GOA provides a file of human proteins assigned with GO terms, and a specific file of SWISS-PROT-TrEMBL data with their GO assignments. For each entry, GOA gives links towards GO molecular function, biological process, and cellular component (Table 1) and many cross-references towards public databanks, GenBank, LocusLink, MedLine, IPI, Ensembl, HUGO and RefSeq via an accession number, which is a means to get for a protein all the information and bibliography stored other databanks.

Molecular Function	Binding activity, Ferric iron binding activity, iron ion binding activity, iron ion homeostasis
Biological Process	Intracellular iron ion storage, Iron ion transport, Cell proliferation
Cellular Component	Ferritin complex

Table 1 : Assignments of GO terms to the protein ferritin heavy chain in GOA

There are also many public databanks in medicine. Among them **OMIM**[9] a database relating human genes and genetic disorders, and **MedLine**[10], which contains 12,000,000 biomedical journal citations accessed through the PubMed service of the National Library of Medicine.

2.3 Existing mappings and links

Many mappings and relations between standard ontologies and databanks are stored in these online resources.

2.3.1 Mappings and links databanks ↔ standard ontologies

- **Databanks ® GO**. For many biological databases, mappings to GO[11] ontology concepts are explicitely defined e.g. mappings of SW keywords to G0 terms (Table 2), mapping of Enzyme Commission Numbers entries to GO function ontology enzymes etc. Moreover, there are also implicit mappings since many public banks e.g. SWISS-PROT-TrEMB data are indexed with GO concepts thanks to GOA (§2.2)

```
!date: 2003/07/14 21:07:05
! Evelyn Camon, SWISS-PROT.
!Mapping of SWISS-PROT KEYWORDS to GO terms
SP_KW:Metal-thiolate cluster > GO:metal ion
binding ; GO:0046872
SP_KW:Metalloenzyme inhibitor > GO:enzyme
inhibitor activity ; GO:0004857 …
```

Table 2 Mappings of SW keywords to G0 terms

- **GO ® Databanks**. Reversely, GO terms are connected to various databanks (Prosite, InterPro, SW etc.) :

'External References' (Figure 1) defines links

[6] http://www.ncbi.nlm.nih.gov/LocusLink/
[7] http://www.gene.ucl.ac.uk/nomenclature/
[8] http://www.geneontology.org/#annotations
[9] http://www.ncbi.nlm.nih.gov/omim/
[10] http://www.ncbi.nlm.nih.gov/entrez/query.fcgi
[11] http://www.geneontology.org/doc/GO.indices.html

from GO terms to entries or indexes of external databanks, e.g. *iron ion homeostasis* is mapped to the SW keyword *Iron storage*.

'Associated Genes' associates GO terms to a list of Gene Products e.g. *iron ion homeostasis* is associated with *PF14_0518*, *CERU_HUMAN* etc. Reversely, for a Gene Product e.g. *CERU_HUMAN*, the field '**Associated to Terms**' provides its GO annotations e.g. *copper ion homeostasis* , *extracellular space*, *ferroxidase activity*, *iron ion homeostasis*

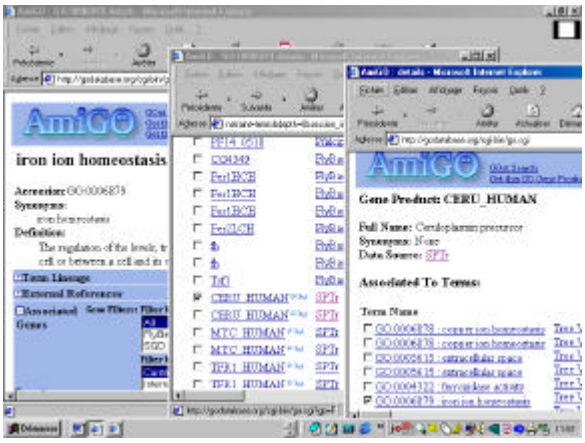


Figure 1 Browsing by chaining links: from a biological process, to a gene, and its function

- **Databanks ↔ UMLS®** There are also links from medical databases to the UMLS®, since the UMLS® is built by integration of dozens of existing terminologies that are used to code data in medical databanks, e.g. MeSH, which is used for indexing the biomedical literature in MedLine.

2.3.2 Links between databanks

Most databanks provides cross-references to other databases via accession numbers (§2.2). HUGO and GOA provide links particular useful for gene annotation systems:

HUGO relates gene to gene products, providing for a given gene the SWidentifier of its associated gene products. For instance, the gene. *ferritin, heavy polypeptide* (FTH1), is related to the SWID: P02794 of its corresponding protein. Accessing it then enables to get its stored information eg. its name *ferritin heavy chain* (FRIH_HUMAN), and synonym *Ferritin H*.

GOA relates gene products and GO terms. It provides for SWentries their relations with GO molecular function, biological process, and cellular component term. From these associations, it is possible to get for a protein, e.g. *Ferritin heavy chain* its GO assignments e.g. its molecular function "*iron ion binding activity*", biological process "*intracellular iron ion storage*".

2.4 Needs of information integration

It is really tedious for biologists and physicians looking for pathologies-genes relationships to browse the relevant information along such mappings and links (Figure 1). The problem is that the knowledge about the sources, their content, links to standard ontologies and between them, is not explicitely represented. An intelligent information integration system is needed providing them with a uniform access to sources both in genomics and medicine. A first tool has been achieved to meet urgent needs of researchers at INSERM U522, which study molecular mechanisms involved in human liver diseases (§3). The more long term objective is to build a more flexible system providing a unified access and services to combine information from various resources accessible on the Web or from local repositories, and to answer complex queries such as find « all the *metalloproteins* involved in *iron homeostasis* that have a *copper ion binding activity* and possible relationships to *liver* diseases » or « all *gene products* involved in proccesses such as *cell proliferation* and *ferric iron binding* with possible relationships to diseases *hemochromatosis and cataract* ».

3 BioMeKe

Biologists and physicians of INSERM U522 and LIM at Rennes study molecular mechanisms involved in human liver diseases, by means of transcriptome analysis. The objective is to find out the genes that are expressed in liver, to correlate them with patient data, in order to better understand pathological processes in liver. But for example, more than 3,000 SWentries are isolated from the tissue « Liver ». BioMeKe (Biological and Medical Knowledge Extraction), has been achieved to help them to extract and to associate medical and biological information accessible from multiple public sources, GenBank, Swissprot, LocusLink, MedLine, etc, and to correlate it to the biologists data laying in their local repository (Gedaw [10]).

3.1 Components and functionalities

BioMeKe, is an ontology-based tool composed of two parts: a core ontology and a query processor :

- **BioMeKe Core Ontology** (BCO) includes the main standard of the biomedical domain: for the medical domain, the UMLS® plus MeSH-ST, for genomics, GO plus GOA which has been added. since GO itself is not populated with gene products nor genes. Different synonyms may be used for a single gene in different databases and all synonymous are not necessarly found in a given database. Therefore, HUGO which adresses such issues is integrated into BCO. All terminologies are separatly stored in a MySQL relational database.

Links between items are dynamically created during the search for a given term or an annotation request.

- **BioMeKe Query Processor** uses BCO knowledge to search information in the external sources. It has three components. The *heterogeneity manager* (HM) uses HUGO and the UMLS for semantic unification of the different names and cross-references, HM returns for a gene its official name and symbol, and SWidentifiers. The *biological search module* (BS) is in charge of searching for biological information in GO, and to provide access to information of several public databanks. For a given term, BS searches for it in GO, GOA. If it is not directly found, it calls HM. If the term is matched and some synonyms provided, the search is done again for those new terms. If it is still unsuccessful, the SWor LocusLink ID provided by HM is then used to access GOA. Since GOA provides cross-references to other databanks, they can then be browsed to pick up relevant information. This unified access to the external banks is possible in the interactive mode, but not in the automatic mode (see 21 for details). The *medical search module* (MS) is in charge of searching for medical information in UMLS. For a given term, MS searches for it in the UMLS. If the term is found its context is displayed, including co-occurrences in MedLine, thus MedLine abstracts can be accessed through MeSH.

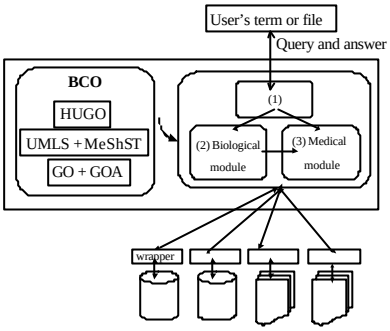


Figure 2 BioMeke

Implementation of the BioMeKe system relies on a MySQL relational database and JAVA A set of JAVA functions (wrappers) have been implemented to access to the content of several public databases, the BCO databases content is accessed thanks SQL queries.

BioMeKe prototype can be used either in an interactive or automatic mode. The automatic mode allows biological and medical annotation for a gene. The interactive mode offers a unified interface that enables, for a term entered by the user, to get biomedical information from the UMLS and GO and to browse across several public databanks the information related to a gene product.

Example. A user may search for biological and medical annotations for the gene *ferritin, heavy polypeptide 1*. BS searches for it in GO, GOA but does not find it, so it calls HM who returns the SWand LocusLink IDs of the corresponding protein *Ferritin heavy chain* (found in HUGO), from which the wanted biological information is obtained thanks GOA (Table 1). These accession numbers can also serve for browsing relevant information in other public databases. The user can search for the item in the UMLS but, the query *Ferritin heavy chain* is unsuccessful in UMLS. Indeed the term that is broadly used in medecine for this gene is *Ferritin H*. Reformulaing the query for the synonym *Ferritin H* provides its context, i.e. here the table MRCOC, from which concepts that co-occur in MedLine (e.g. liver, hemochromatosis, cataract) can be extracted and abstracts accessed (Table 1).

3.2 Application

Gene Name	*Ceruloplasmin (ferroxidase)*	*Ferritin*
Molecular Function	Oxidoreductase activity, Copper ion binding, Multicopper ferroxidase iron transport mediator activity	Binding activity, Ferric iron binding activity
Biological Process	Iron ion homeostasis, Copper ion homeostasis	Iron ion transport, Intracelluar iron ion storage, Cell proliferation, Iron ion homeostasis
Cellular Component	Extracellular space	Ferritin complex
Co-occurences Disease or Syndrom	Nervous system diseases, Iron overload[12], s etc.	Hemochromatosis, Cataract, etc.

Table 3: BioMeKe automatic annotation (extracts)

BioMeKe is being evaluated for the automatic annotation of genes for transcriptome analysis in the domain of liver diseases [10]. The process has three main steps:

Step1: Synonyms management. In order to reconciliate all the identifiers stored in the datawarehouse, and to solve gene synonymy problems, Locuslink identifiers are extracted from the GenBank file, then the HQ module provides the official names and symbols, and SW identifiers.

Step2: GO annotation. From SW identifiers, the BS module returns GO biological information via GOA

Step 3: UMLS annotation. The MS module uses the names and symbols provided at step 1 as inputs to search information in the UMLS. The UMLS annotations are filtered by semantic types to keep the 25 most relevant types to relate genotypes to

[12] the generated report contains 80 associated diseases

phenotypes (e.g 'Disease or Syndrom') (Table 3).

3.3 Limitations

BioMeKe main innovation is to be an ontology-based tool. However, the ontologies are non formal. Second, it is a procedural tool, and it provides semantic integration, but it is still limited.

- **Limits of non formal ontologies.**

GO and UMLS are not structured according formal principles, and exhibit many inconsistencies. For example (Fig. 3) in GO *Multicopper ferroxidase iron transport mediator activity* is child of *metal ion transporter activity*, which is sibling of *cation transporter activity*, while in another subtree, *metal ion homeostasis* is defined as child of *cation homeostasis*. Hierarchies for the *copper ion binding*, *copper ion transporter* functions, the *copper ion transport* process, all have a different pattern (resp. iron) etc. !

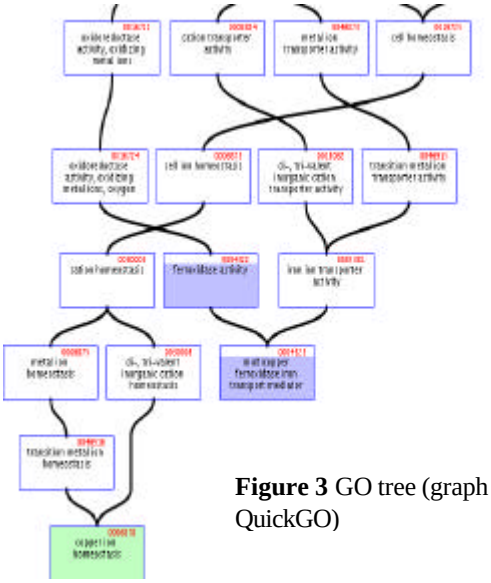


Figure 3 GO tree (graph from QuickGO)

Since GO 'is-a' hierarchy is not rigorous, it entails that gene annotation, may exhibit inconsistencies, redundancies, lacking, or heterogeneity., e.g. BioMeKe relates *Ceruloplasmin* to *Multicopper ferroxidase iron transport mediator activity* and to one arbitrary subsumer *Oxidoreductase activity*, but not to the others, e.g. *ion transporter activity*. Informal ontologies are clearly not appropriate in a context of integration.

- **Limits of the query engine**

BioMeKe is mainly grounded on various "mappings" and relations between the standard ontologies and databanks, or between databanks (by cross-references). However, since this knowledge remains implicit, many tasks are still grounded on user's skill and own responsibility: reformulation, selection of databanks to browse etc. Even assisted by BioMeKe, it remains difficult for researchers looking for pathologies-genes relationships, to navigate along such mappings and links across databanks to get the relevant information, and useful relations may easily be missed. BioMeKe is a procedural system, based on a fixed process. As the number of online databanks always increases, more automatization and more flexibility are required, providing extensibility and dynamic sources selection possibilities

- **Limits in semantic integration.**

BioMeKe management of heterogeneity is limited. First, it is mainly based on the synonyms found either in HUGO or the UMLS, but it does not exploit other information available in external databanks e.g. the synonymy of *ferritin heavy chain* and *Ferritin H* is asserted in SWISS-PROT. Second, GO, UMLS, HUGO must be frequently updated. Third, BCO has to be customized for specific use, e.g. a lexical database associating official gene names with complementary simplified names has to be added for liver transcriptome analysis. Moreover, heterogeneity concerns not only the data, but also at a more generic level, ontology concepts and relations. Although GO and UMLS have been recently merged on a lexical basis [25], generalizing mappings between ontologies is difficult. even with recent interactive tools such as PROMPT.

4 Lessons learnt

Some improvements are possible in BioMeKe. But, addressing all above problems clearly requires a *declarative* (knowledge-based or database) approach, allowing an explicit representation of the knowledge (ontology, mappings, queries) and an inference (query) engine with powerful services in particular for ontology automatic classification, consistency checking, and dynamic chaining of mappings. There is clear needs of formal ontology web languages, and of more flexible integration.

4.1 Needs of formal ontologies

Most people now agree about the limits of non formal ontologies and benefits of a formal language ontologies, for the Web in general [26] and in the biomedical domain [23] [27] [8]. First, "multiple viewpoints" is an old problem in biomedicine. For example, in GO functions, processes hierarchies are organized from a biochemical viewpoint derived from the EC Enzyme Commission classification, or from the chimical substances they act on*metal ion*, *cation*, *transition metal ion*, *iron*, *copper*. Multiple viewpoints are source of inconsistencies, when the ontology structuration is not automatized (§ 3.3). Moreover biologists and physicians are interested in clustering diseases, genes according to different dimensions, e.g. genes according to their functions or related pathologies, also in identifying all the gene products that share a same feature. Description Logics (DL) provide powerful services

for that, and the next W3C standard Ontology Web Language OWL[13] comes with useful tools e.g. the FaCT automatic classifier [14], the OilEd editor [2].

Example. The following example shows how constructing a global formal ontology for genomics in OWL will prevent from many inconsistencies. GO concepts below (in DL syntax) are based on MeSH definitions expressing that a cation is a postively charged atom, a cation divalent has valence of plus 2, an ion metal is a cation and a metal etc.

Cation:= Ion $\wedge$ ($\geq$ charged PositiveCharge)

CationDivalent:=Cation $\wedge$ ($\leq$ 2 charged PositiveCharge)

IonMetal: = Cation $\wedge$ Metal

TransitionMetalIon:= CationDivalent $\wedge$ ($\forall$ belongsto PeriodicGroup 3-12)

The "root" concepts Transport, Binding are using explicit roles "transported" "bound" relating them to the Chemical ontology concepts.

Transport:= Activity $\wedge$ ($\forall$ transported Chimical)

TransitionMetalIonTransportActivity:= Transport $\wedge$ ($\forall$ transported TransitionMetalIon)

Binding:= Activity $\wedge$ ($\forall$ bound Chimical)

TransitionMetalIonBinding:= Binding $\wedge$ ($\forall$ bound TransitionMetalIon)

Then all the sub-ontologies stemming from these concepts are globally consistent (and more generally so built ontologies, provided the related ontologies consistence e.g. Chemical). Such a formal ontology, also enables defining rigourous rules for gene annotation, for example "annotation must be done with the most specific function (resp. process, etc.)" since the others can be infered.

4.2 Needs of a more flexible information integration

Extensibility and real-time data are crucial requirements. Bioinformatics is a very fast-moving field. Web sources are multiple, with huge and constantly evolving contents. New online ontologies and specialized databanks often appear. Datawarehouses are not well appropriate and more flexible integration, such as mediator-based *centralized* systems, or new approaches proposing *distributed* integration are quite attractive [4]. *Local as view* (LAV) mediators defining the content of sources in terms of views over the global ontology, might be preferred to *global as view* (GAV), defining the global ontology in terms of views over the sources e.g. Tambis [27]. But although mediators are a significant progress, they may be not even flexible enough for scaling up the Web, and distributed systems are perhaps more appropriate. As described, databanks are not only data "sources" but also include precious links and mappings, through their cross-references to ontologies and other databanks. Such local relations between sources should be explicitly represented and directly exploited to infer new information. Peer-based integration where "every participant should be able to contribute new data and relate it to existing concepts and schemas, define new schemas that others can use as frames or reference for their queries or define new relationships between existing schema or data providers" is therefore a challenging approach to meet the extensibility and distribution encountered in biomedical information integration. But, whatever mediator or peer-based integration systems, rich formal languages are required for representing ontologies, queries, and mappings,. [9]

[13] http://www.w3.org/TR/owl-ref/

[14] http://www.cs.man.ac.uk/~horrocks/FaCT/

5 Discussion

Other systems have been achieved for gene annotations e.g. Source [5], or MatchMiner [14]. BioMeKe and Source annotation results have been compared on a sample of 364 genes : among the 250 gnes annotated by both systems, Source provide a more complete annotation for 15%, while BioMeKe for 38%. BioMeKe is based on GO and the UMLS, but several other ontologies exist like GALEN [23], TaO [1] for molecular biology and bioinformatics OMB (Ontology for Molecular Biology). The next perspective is to developp either a LAV mediator, opposed to TAMBIS GAV approach [27] or a distributed system. A LAV mediator requires a *global* ontology for genomics *and* medicine. Building such a formal ontology joins recent projects aiming at migrating GO to DL [30] or at merging the UMLS and GO [24]. Another perspective is to build an hybrid tool combining a search based on the formal ontology together with a classical search based on GO and UMLS.

6 Conclusion

BioMeKe is a first ontology-based tool facilitating the access and search of biological and medical information related to gene or gene products. An automatic mode allows annotation of gene files. However, selecting the sources to be explored and the information to extract is still too much grounded on the user's own skills and responsability. The current challenge is to provide a more automatized and flexible integration. A formal Web ontology language like OWL, and mediators or Peer-based distributed integration seem to be promising techniques. Main challenges are now to combine them, and to provide a language for mappings. Another bottleneck is to represent huge ontologies like GO and the UMLS in OWL and source mappings definitions for so multiple sources. Partial automatization seems the only reasonable solution.

7 References

1. Baker,P. et al. (1999) An ontology for bioinformatics applications. Bioinformatics, 15, 510–520.
2. Bechhofer S., et al. OILEd: a Reason-able Ontology Editor for the Semantic Web. Proc KI2001, 396-408. 2001.
3. Bernstein. P. Applying model management to classical meta data problems. Proc of the Conf. on Innovative Database Research (CIDR), 2003.
4. Bernstein P et al. Data management for peer-to-peer computing: A vision, Workshop WebDB 2002.
5. Cantor MN, Lussier YA. A knowledge framework for computational molecular-disease relationships in cancer. Proc AMIA Symp 2002;101-5
6. Diehn M et al. SOURCE: a unified genomic resource of functional annotations, ontologies, and gene expression data. Nucleic Acids Research, 2003, 31, 1: 219-23
7. Goasdoue, F, Lattes, V, Rousset, MC. The Use Of Carin Language and Algorithms for Information Integration: The PICSEL System Int J Coop Infor Sy stems, 9(4), 383-401, 2000.
8. Golbreich, C et al. Web ontology language requirements w.r.t expressiveness of taxononomy and axioms in medecine, ISWC 2003, Springer.
9. Golbreich, C., B., Burgun A. Challenges for Biomedical Information. Position Statement Paper. Semantic Integration Workshop, ISWC 2003
10. Guérin E et al. UML modeling of Gedaw: A gene expression data warehouse specialised in the liver. JOBIM; 2002 France, Saint-Malo. p. 319-334.
11. Hahn, U., Schulz S. Turning Lead into Gold? Feeding a Formal Knowledge Base with Informal Conceptual Knowledge. EKAW 2002: 182-196
12. Halevy A. Y. Answering queries using views. The VLDB Journal, 10(4):270-294, 2001.
13. Halevy AY, Zachary G, Ives Suciu D, Tatarinov I. Schema mediation in peer data management sy stems. ICDE, 2003.
14. Kimberly J.B. et al. MatchMiner: a tool for batch navigation among gene and gene product identifiers. Genome Biology, 2003 4(4):R27
15. Kirk, T, Levy, AY., Sagiv Y, D .Srivastava. The Information Manifold, Information Gathering from Heterogeneous, Distributed Environments, AAAI Spring Symposium Series, Stanford University, March 1995.
16. Levy A. Y, Logic-Based Techniques in Data Integration Logic Based Art Int , J Minker. Ed Kluwer,, 2000
17. Levy A. Y, Rousset MC, The Limits on Combining Recursive Horn Rules with Description Logics, AAAI/IAAI, Vol. 1 (1996)
18. Li Q, Shilane P, Noy NF, Musen MA. Ontology acquisition from on-line knowledge sources. Proc AMIA Symp. 2000;497-501.
19. Lindberg DAB, Humphreys BL, McCray AT, The Unified Medical Language System. Meth Inform Med, 1993, 32(4): 281-91
20. Lucie Xyleme: A dynamic warehouse for XML Data of the Web. IEEE Data Eng Bull 24(2): 40-47 (2001)
21. Marquet G et al. BioMeKe: an ontology-based biomedical knowledge extraction system devoted to transcriptome analysis, MIE 2003
22. Povey S et al. The HUGO Gene Nomenclature Committee (HGNC).Hum Genet. 2001;109(6):678-80
23. Rector A. et al. The GRAIL concept modelling language for medical terminology. Art Int Med,9:139-171, 1997.
24. Sarkar, I et al. Linking biomedical language information and knowledge resources in the 21st century: GO and UMLS, *Pac Symp Biocomput*, 2003 **8**, 427-450.
25. Schulz S, Hahn U. Medical knowledge reengineering--converting major portions of the UMLS into a terminological knowledge base. Int J Med Inf. 2001 Dec;64(2-3):207-21.
26. Staab S Edt Ontologies'KISSES in standardization. IEEE Intelligent Systems, 70-79
27. Stevens R et al. TAMBIS: transparent access to multiple bioinformatics information sources. Bioinformatics. 2000 Feb;16(2):184-5.
28. The Gene Ontology Consortium. Creating the gene ontology resource:design and implementation. Genome Res 2001,11(8):1425-1433
29. Wiederhold G. Mediators in the architecture of future information systems. Computer, 1992, 25(3): 38-49
30. Wroe CJ, Stevens R, Goble CA, Ashburner M. A methodology to migrate the gene ontology to a description logic environment using DAML+OIL. Pac Symp Biocomput. 2003:624-35.

Acknowledgements

The authors thank F Moussouni, E Guérin, O Loréal, F Mougin for their participation in BioMeKe.

SCL: A LOGIC STANDARD FOR SEMANTIC INTEGRATION

CHRISTOPHER MENZEL AND PATRICK HAYES

The Knowledge Interchange Format (KIF) [2] is an ASCII-based framework for use in exchanging of declarative knowledge among disparate computer systems. KIF has been widely used in the fields of knowledge engineering and artificial intelligence. Due to its growing importance, there arose a renewed push to make KIF an official international standard. A central motivation behind KIF standardization is the wide variation in quality, style, and content — of logic-based frameworks being used for knowledge representation. Variations of all three types, of course, hinder the possibility of semantic integration. A well-crafted logic standard for the representation of declarative knowledge would impose some greatly needed syntactic and semantic uniformity on the current somewhat chaotic situation, uniformity that would in turn greatly enhance the capacity for semantic integration.

For all its potential advantages, however, the idea of a logic standard is problematic for at least two reasons:

- Standardization of a single syntax forces comformant users to write their logic in a form that is likely to be, at the least, unfamiliar, and, at worst, may in fact not be optimal for their representational needs. Call this the *uniformity* problem
- The standard might involve constructs that are neither needed nor desired for one's representational purposes. Call this the *excess baggage* problems.

KIF in fact seems particularly vulnerable to objections along these lines. Its LISP-like syntax is not universally held in high esteem. Moreover, it includes a variety of constructs that researchers find quirky and unnecessary, notably:

- Variable polyadicity — predicate constants and function symbols have not fixed arity, but can take any number of arguments;
- Pseudo-higher-order constructs — bound variables can occur in predicate position in atomic formulas.
- Type-freedom — predicates can occur as arguments to other predicates; semantically speaking, properties and relations are "first-class" objects that can be referred to and quantified over like any other individuals.
- Non-first-order expressiveness — KIF includes "sequence variables", the presence of which raises its expressive power beyond first-order to that of a weak infinitary logic.

To add to the confusion, KIF has lacked a rigorous model theory for its distinctive constructs.

Nevertheless, the idea of standardization is still a good one — widespread conformance to such a standard would go a long way toward enabling semantic integration between diverse knowledge bases. Moreover, something on the order of KIF's full first-order expressive power, at the least, is still needed, especially for the metalinguistic constructs that are inevitably needed to enable semantic integration. Finally, though superfluous in some context, KIF's additional constructs prove useful and convenient in others.

The solution sketched in this brief technical paper — the Simplified Common Logic (SCL) framework[1] — addresses the uniformity problem by defining a purely abstract syntax that specifies only the underlying structure that a conformant language must exhibit, leaving the concrete specifics of any given manifestation to the discretion of the user. SCL addresses the excess baggage problem by defining the grammatical framework flexibly enough to allow users to pick and choose from a variety of syntactic constructs depending on their representational needs and preferences. Finally, a rigorous general model theory is provided that yields definitions of denotation and truth for any given SCL language.[2]

1. LEXICONS

An SCL language is based upon an initial stock of primitive syntactic entities. Specifically, an SCL *lexicon* λ will consist of the following sets:

- A countable set *PCon* called the *predicate constants* of λ. This set will include a distinguished predicate *Id*. (Predicate constants will also be referred to simply as *predicates*.)
- A countable set *ICon* called the *individual constants* of λ.
- A countable set *FnSym* called the *function symbols* of λ.
- A denumerable set *GVar* called the *general variables* of λ;
- A set *SVar* called the *sequence variables* of λ. *SVar* will be either empty or denumerable.

If *SVar* is empty, then λ is known as a *first-order* lexicon.

[1] SCL is part of the Common Logic Standard effort; see [1]. The present paper is a distillation of some of the current SCL working document [4].

[2] We have recently been made aware of the language HiLog [3], which purportedly is syntactically and semantically quite similar to SCL (without sequence variables) . We have not had the time yet to study the framework full, so we will have to report on the similarities and differences in a further paper.

$Con = PCon \cup ICon$ is known as the set of *constants* of λ. $Var = GVar \cup SVar$ is known as the set of *variables* of λ. *GVar* and *SVar* shall be disjoint, and *Var* shall be disjoint from $Con \cup FnSym$. Let $PrimTrm = ICon \cup GVar$. *PrimTrm* is known as the set of *primitive terms* of λ.

Lexicons λ also come with a function *arity* that maps each predicate constant and function symbol into the set $\mathbb{N} \cup \omega$, where $\mathbb{N}$ is the set of natural numbers and ω is any object not in $\mathbb{N}$. For predicates π, *arity* will indicate the number of arguments π will take. (This will of course be expressed explicitly in the grammar below.) If $arity(\pi) = n \in \mathbb{N}$, then π is said to be an n-place predicate; otherwise π is *variably polyadic*. Variably polyadic predicates will be able to take any number of arguments. We let $PCon_n$ be the set of n-place predicates, and $PCon_\omega$ the set of variably polyadic predicates. Because we will interpret function symbols as functional relations, we will let the arity of a function symbol correspond to the arity of the relation it denotes rather than to the number of arguments it takes. This will also enable the predicates of an SCL lexicon to do double duty as function symbols — note that there is no requirement that *PCon* and *FnSym* be disjoint. Accordingly, for function symbols α, if $arity(\alpha) = n+1$, we say that α is an n-place function symbol; otherwise α is variably polyadic. We stipulate that $arity(\alpha) \neq 0$, for any function symbol α. We let $FnSym_n$ be the set of n-place function symbols, and $FnSym_\omega$ the set of variably polyadic function symbols.

Over and above presence of sequence variables, SCL lexicons differ from traditional first-order lexicons in three important ways. First, SCL generalizes the notion of arity by allowing (though not requiring) variably polyadic predicates and function symbols, i.e., predicate constants and function symbols that can take arbitrarily many arguments. Variably polyadicity is especially useful and appropriate in SCL languages containing sequence variables.

Second, it is not required that *PCon*, *ICon*, and *FnSym* be pairwise disjoint. This reflects SCL's goal of generality. Many knowledge representation languages are "type-free" to one extent or another; that is, they treat properties, propositions, classes, functions, and other so-called "higher-order" entities as "first-class citizens" in their own right, capable of being referred to and quantified over along with individuals. Natural language itself reflects this "dual role" that properties and their ilk can play in the gerundive construction, whereby verb phrases expressing properties and relations — e.g., *is a linguist* — are transformed into noun phrases — *being a linguist*. By allowing predicate constants and function sybols simultaneously to serve as individual constants, and by allowing variables to serve as predicable terms, SCL provides a formal correlate to these constructions and thereby provides a rigorous framework in which this common knowledge representation construction is fully sanctioned.

To illustrate SCL's flexibility, we explicitly pick out several important limiting cases of SCL languages that are determined by minimally or maximally tweaking arity and the degree of overlap among constants and function symbols. Thus, say that an SCL lexicon λ is *fully typed* if $(PCon \cup FnSym) \cap ICon = \varnothing$ (i.e., if there is no overlap between the predicates constants, function symbols, and individual constants of λ); *arity-fixed* if, for all predicate constants and function symbols κ, $arity(\kappa) = n$, for some $n \in \mathbb{N}$ (i.e., if every predicate constant and function symbol has a fixed arity); and *traditional first-order* (*TFO*) if λ is both fully-typed and arity-fixed. By contrast, say that λ is *arity-free* if, for all predicate constants and function symbols κ, $arity(\kappa) = \omega$; *type-free* if $PCon \cup FnSym \subseteq ICon$; and *unconstrained* if λ is both arity-free and type-free. In between the extremes of TFO and unconstrained lexicons, of course, lie any number of interesting intermediate possibilities.

2. GRAMMARS

2.1. **Terms.** Given an SCL lexicon λ, we define the notion of a term class based on λ. Intuitively, a term is either a primitive term (constant or variable) or the result of "applying" a function symbol to some nonempty sequence of terms. Because we are defining an abstract syntax, we do not want to specify the exact form that the application of a function symbol to its arguments should take. Hence, we simply specify the general constraints than any syntax of application must satisfy; we do this in terms of a certain type of syntactic function.

As groundwork for this definition, for any set M, let M^ω be the set of finite sequences of elements of M, i.e., $M^\omega = \bigcup n \in \mathbb{N} M^n$, where M^n is the set of all n-tuples of elements of M. Given this, say that T is a *term class for* λ if T contains all of the primitive terms of λ and is the smallest class closed under a one-to-one operation **App** — called a *term generator* for λ — such that

$$\textbf{App} : \bigcup_{n\in\mathbb{N}}\{FnSym_n \times T^n \cup (FnSym_\omega \times (T^\omega \cup (T^\omega \times SVar)))\} \longrightarrow T.$$

That is, for $\tau_1, ..., \tau_n \in T$, if α is an n-place function symbol, then $\textbf{App}(\alpha, \tau_1, ..., \tau_n) \in T$, and if α is a variably polyadic functional, then in addition for any sequence variable σ, $\textbf{App}(\alpha, \tau_1, ..., \tau_n, \sigma) \in T$;

We say that **App** *generates* the corresponding term class T. For any term generator **App** for λ, let $FnTrm = Range(\textbf{App})$. $FnTrm$ is the set of *function terms* of λ (relative to **App**).

So, for example, if `a` and `b` were among the constants of a lexicon λ and `f` and `g` among its function symbols, then any of the following might among the function terms

produced by different generators: `f(a,g(b),s)`, `(f a (g b) s)`, `s[bg]af` (somewhat perversely) and even the XML'ish

```
<term>
  <fnsym>f</fnsym>
  <indcon>a</indcon>
  <term>
    <fnsym>g</fnsym>
    <indcon>b</indcon>
  </term>
  <seqvar>s</seqvar>
</term>
```

2.2. **Type-Freedom and Predicability.** As hinted at above, and as will be spelled out in more detail in the model theory below, one of the important features of SCL is that it allows for a "type-free" semantics in which properties and relations are treated as first-class individuals. Languages with such a semantics will there be allowed to refer to and quantify over such "reified" entities directly. In particular, it is important to allow such languages to quantify over them in their predicative roles. Syntactically speaking, this means that we must allow variables to occur in predicate position in atomic formulas, e.g., in KIF:

```
(forall (?x ?y ?F)
        (impl (Symmetric ?F)
              (impl (?F ?x ?y) (?F ?y ?x))))
```

However, because it is important that SCL encompass more traditional first-order languages as well, type-freedom should be optional. Accordingly, whether or not variables (and other expressions, more generally) can occur in predicative position along with predicate constants will be specified in the grammar for a language, rather than being predetermined by the chosen lexicon. Consequently, the set $Pred_n$ of n-place predicables in an SCL grammar is allowed to be either simply the set $PCon_n \cup Pred_\omega$ (since variably polyadic predicates be predicated of any finite number of arguments — hence, in particular of n) or the set $PCon_n \cup Pred_\omega \cup GVar$. A similar generalization that allows variables to occur in function position in complex terms adds a certain elegance and convenience at the cost of a great deal of semantic complexity, but the gains are minimal for the purposes envisioned for SCL.

2.3. **Formulas.** In light of the above, we now do for formulas what we did for terms. Let λ be an SCL lexicon, and let *Trm* be the term class for λ generated by some term generator **App**. First, we need a class of basic formulas. Let **Holds** be a one-to-one function on $\bigcup_{n\in\mathbb{N}}\{Pred_n \times T^n \cup (Pred_\omega \times (T^\omega \cup (T^\omega \times SVar)))\}$. That is, given an n-place predicable and n terms, or a variably polyadic predicable, n terms and a sequence variable, **Holds** returns a unique formula. Any such function **Holds** is said to be a *predication operation for λ based on* **App**. As with term generators, the outputs of different predication functions might take very different forms. The only constraint is that distinct inputs always yield distinct outputs. Given a term generator, the range of a predication operation **Holds** for λ is said to be the class of *atomic formulas* for λ generated by **Holds**.

Let *At* be the class of atomic formulas for λ based on a predication operator **Holds**. Say that F is a *formula class* for λ, relative to **Holds**, if it is the smallest class that includes *At* and is closed under a set *Op* — known as a *formula generator* for λ based on **Holds** — of operations **Id**, **Neg**, **Conj**, **Disj**, **Cond**, **Bicond**, **EQ**, **UQ** that satisfy the following conditions:

- Each operation is one-to-one;
- The ranges of the operations are pairwise disjoint, and disjoint from *Trm*
- **Id** : $Trm \times Trm \longrightarrow F$
- **Neg** : $F \longrightarrow F$
- **Conj** : $F^* \longrightarrow F$
- **Disj** : $F^* \longrightarrow F$
- **Cond** : $F \times F \longrightarrow F$
- **Bi** : $F \times F \longrightarrow F$
- **EQ** : $(GVar \cup (GVar \times (PCon_1 \cup PCon_\omega)))^* \times F \longrightarrow F$
- **UQ** : $(GVar \cup (GVar \times (PCon_1 \cup PCon_\omega)))^* \times F \longrightarrow F$

Let *Fla* be range of the operations in *Op*. We say that *Fla* is the formula class *generated by Op*.

As with terms, depending on one's choice of term generator, predication operation, and generator set, SCL languages can come in many different concrete forms. So, for example, the standard, first-order "logical form" of 'Every boy kissed a girl' in terms of our abstract syntax is

UQ$(\nu_1,$ **Cond**$($**Holds**$(\pi_1,\nu_1),$ **EQ**$(\nu_2,$ **Conj**$($**Holds**$(\pi_2,\nu_2),$ **Holds**$(\pi_3,\nu_1,\nu_2)))))$,

where π_1, π_2, and π_3, are "slots" for the predicates constants of the appropriate arity chosen from any particular lexicon to represent boyhood, girlhood, and kissing, and ν_1 and ν_2 represent some choice of variables. In one SCL language, this form might be realized by its familiar introductory text-book form:

$$(\forall x)(Boy(x) \rightarrow (\exists y)(Girl(y) \wedge Kissed(x,y))).$$

A conceptual graph interchange form (CGIF) implementation has a rather different appearance:

```
[@every*x][If:(Boy ?x)[Then:[*y](Girl ?y)(Kissed ?x ?y)]].
```

As does a KIF-like implementation:

```
(forall (?x ?y)
        (impl (Boy ?x))
              (exists (?y)
                      (and (Girl ?y)
                           (Kissed ?x ?y))))
```

not to mention the following XML'ish monstrosity:

```
<formula>
  <forall>
    <var>x</var>
    <formula>
      <implies>
        <formula>
          <atom>
            <con>Boy</con>
            <var>x</var>
          </atom>
        </formula>
        <formula>
          <exists>
            <var>y</var>
            <formula>
              <and>
                <formula>
                  <atom>
                    <con>Girl</con>
                    <var>x</var>
                  </atom>
                </formula>
                <formula>
                  <atom>
                    <con>Kissed</con>
                    <var>x</var>
                    <var>y</var>
                  </atom>
                </formula>
              </and>
            </formula>
          </exists>
        </formula>
      </implies>
   </forall>
</formula>
```

It is important to observe that, because the operations in a generator set for a formula class *Fla* for λ are all one-to-one and disjoint in their ranges, every element of *Fla* will have exactly one "decomposition" under the inverses of those operations, and that all such decompositions are finite. Let $\varphi \in Fla$. An object ϵ in the decomposition of φ is an *atom* of φ just in case ϵ is an element of the lexicon λ. ψ is a *subformula* of φ if $\psi \in Fla$ and ψ is in the decomposition of φ.

2.4. **Languages.** Let **App** be a term generator for λ, where *Trm* is the set generated by **App**, and let **Holds** be based upon **App**. Let *Op* be a formula generator for λ based on **Holds**, and let **L** be the formula class generated by *Op*. We define any such set **L** to be an *SCL language* for the SCL lexicon λ, and we say that λ *underlies* **L**. *Trm* is said to be the set of *terms* of **L**. If λ and λ' are SCL lexicons with the same sets of constants and function symbols, and **L** and **L**' are SCL languages for λ and λ', respectively, then **L** and **L**' are said to be *equivalent*. If λ is a first-order lexicon, then a language for λ is said to be a *first-order* SCL language. In particular, on ths definition, every familiar first-order language turns out to be an instance of an SCL language whose underlying lexicon is traditional first-order (i.e., "TFO" — see the end of Section 1 above). We therefore call any such language a *TFO* language.

3. INTERPRETATIONS

Let λ be an SCL lexicon. An *SCL interpretation* **I** for λ is a 4-tuple $\langle I, R, ext, V\rangle$ satisfying the following conditions. First, I and R are nonempty sets. Intuitively, I represents the set of *individuals* of **I**, and will serve as the range of the quantifiers and its members will serve as the denotations of terms. R is the set of relations[3] whose members serve as possible denotations of predicate constants. To allow for type-freedom, there is no requirement that I and R be disjoint; indeed any degree of overlap, from partial to complete, is allowed. Those relations that are also members of I are said to be *reified*. Intuitively, reified relations are relations that can also be thought of as individuals. Accordingly, they can also be the values of individual constants and individual variables.

R is itself the union of countable sets R_ω, R_1, R_2, R_3, ... All are possibly empty with the exception of R_2, which contains a distinguished element **Id**, intended to serve as the identity relation. Intuitively, R_ω is the set of variably polyadic relations, and each R_n the set of n-place relations. Accordingly, *ext* is a corresponding extension function from R into $Pow(I^\omega)$ subject to the constraint that, for any natural number $n > 0$, if $r \in R_n$, then $ext(r) \subseteq I^n$; in particular, $ext(\mathbf{Id}) = \{\langle a, a\rangle : a \in I\}$.

Intuitively, of course $ext(r)$ represents the extension of r. For elements r of R_ω, if $ext(r)$ is a total (extensional) function on I^ω, then we say that r is a *function* on I^ω. For $n+1$-place relations r, if $ext(r)$ is a total (extensional) function on I^n, then we also say that r is a *function on* I^n, or an *n-place function*.

Finally, V is a "denotation" function that assigns appropriate values to the constants and function symbols of **L**. Specifically,

- If κ is an individual constant, then $V(\kappa) \in I$;
- If π is a predicate constant, then $V(\pi) \in R_{arity(\kappa)}$.
- If α is a function symbol, then $V(\alpha)$ is a function on $I^{arity(\alpha)}$.

Note, importantly, that it is not required that I and R be disjoint. This is the semantic correlate of the type-freedom permitted (though not required) in SCL languages. Specifically, an SCL language **L** can allow a primitive term κ to do double duty as both a predicate constant

[3]It is possible to model of the members of R extensionally as sets, though this will in general require non-well-founded set theory, since a relation, qua individual, can be in its own extension.

and an individual constant. Consequently, the denotation function V in any interpretation **I** of **L** must by definition map κ, qua predicate constant, to an element of R; it must also map κ, qua individual constant, qua individual constant, to an element of I. Consequently, to satisfy these constraints, **I**, $V(\kappa)$ will have to be in both I and R, i.e., it will have to be both a relation and an individual. And this is just what the semantics allows. In a similar fashion, predicate constants can do double duty as function symbols.

A question might arise about interpretation in which only some members of R are members of I. In fact, it is likely that in the most common intended interpretations overlap will either be nonexistent or complete. However, there is a reasonably natural idea corresponding to partial overlap, namely, that some predicates indicate real properties of things and others are just convenient ways of categorizing things. For example, in an biological ontology, "is an arm" may not be thought of as a genuine property of anything, but only a convenient way of classifying things that play a certain functional role in a biological organism. By contrast "is a cell" might be thought of as indicating a genuine biological property of a thing that one might wish to include the genuine inventory of one's ontology. Partial overlap provides a natural way of preserving this distinction.

4. Denotations and Truth

Given the notion of an interpretation for a lexicon λ, we can now define what it is for a formula of an SCL language L based on λ to be *true* in an interpretation.

Some additional apparatus will be useful in defining truth for quantified formulas (i.e., formulas in the range of **EQ** and **UQ**). First, given an interpretation **I**, define a *variable assignment* for **I** to be a function that maps individual variables into I and sequence variables into I_ω. To define the semantics of quantification, what we need is the notion of a variable assignment v' that is exactly like a given assignment v except that it might not agree with v on what to assign to some finite set of individual variables. The idea is straightforward, but the presence of restricted quantifiers forces us to proceed with some care. Let $\mathbf{I} = \langle I, R, ext, V\rangle$ be an interpretation for **L**, and let v be a variable assignment for **I**. In our syntax, a quantifier can bind an entire sequence consisting of (individual) variables and variable/predicate pairs. So let $\chi_1, ..., \chi_n$ be such a sequence, and say that a variable assignment v' for **I** is a $[\chi_1, ..., \chi_n]$-*variant* of v iff

- if χ_i is a variable / predicate-constant pair $\langle \nu, \kappa\rangle$ and $V(\kappa)$ is a relation, then $v'(\nu)$ is in the extension of $V(\kappa)$; and
- $v'(\nu) = v(\nu)$, if ν is distinct from all the variables in the sequence $\chi_1, ..., \chi_n$ and all of the variables occurring in variable/constant pairs in the sequence.

So let **L** be an SCL language for a lexicon λ, where **App** generates the set *Trm* of terms of **L**, and let $\mathbf{I} = \langle I, R, ext, V\rangle$ be an interpretation for **L**. Given **I** and a variable assignment v, let V_v be $V \cup v$. Given **I** and a variable assignment v, the denotations of the function terms of **L** in **I** are completely determined by V_v. This can be expressed in terms of a unique extension $V_v^\#$ of V such that, for any term $\tau \in Trm$:

- If τ is an individual constant, then $V_v^\#(\tau) = V(\tau)$.
- If τ is a variable, then $V_v^\#(\tau) = v(\tau)$.
- If τ is a function term $\mathbf{App}(\alpha, \tau_1, ..., \tau_n)$, then:
  - If τ_n is a sequence variable and $v(\tau_n) = \langle e_1, ..., e_m\rangle$, then $V_v^\#(\tau) = V(\alpha)(V_v^\#(\tau_1), ..., V_v^\#(\tau_{n-1}), e_1, ..., e_m)$.
  - If τ_n is not a sequence variable, then $V_v^\#(\tau) = V^\#(\alpha)(V_v^\#(\tau_1), ..., V_v^\#(\tau_n))$;

Given V, we define satisfaction for the formulas of **L** by a variable assignment v for our interpretation **I** as follows. Let $\varphi \in \mathbf{L}$:

- If $\varphi = \mathbf{Holds}(\kappa, \tau_1, ..., \tau_n)$, then:
  - If τ_n is a sequence variable and $v(\tau_n) = \langle e_1, ..., e_m\rangle$, then v satisfies φ iff $V(\kappa) \in R_{arity(\kappa)}$ and $\langle V_v^\#(\tau_1), ..., V_v^\#(\tau_{n-1}), e_1, ..., e_m\rangle \in ext(V(\kappa))$.
  - If τ_n is not a sequence variable, then v satisfies φ iff $V(\kappa)$ is a relation and $\langle V_v^\#(\tau_1), ..., V_v^\#(\tau_n)\rangle \in ext(V(\kappa))$.
- If $\varphi = \mathbf{Id}(\tau, \tau')$, then v satisfies φ iff $V_v^\#(\tau) = V_v^\#(\tau')$.
- If $\varphi = \mathbf{Neg}(\psi)$, then v satisfies φ iff ψ is not true in **I**.
- If $\varphi = \mathbf{Disj}(\psi_1, ...\psi_n)$, then v satisfies φ iff v satisfies ψ_i for some i, $1 \leq i \leq n$.
- If $\varphi = \mathbf{Conj}(\psi_1, ...\psi_n)$, then v satisfies φ iff v satisfies ψ_i for each i, $1 \leq i \leq n$.
- If $\varphi = \mathbf{Cond}(\psi, \psi')$, then v satisfies φ iff v does not satisfy ψ or v satisfies ψ'.
- If $\varphi = \mathbf{Bi}(\psi, \psi')$, then φ v satisfies φ iff iff v either satisfies both ψ and ψ' or satisfies neither.
- If $\varphi = \mathbf{EQ}(\chi_1, ..., \chi_n, \psi)$, then v satisfies φ iff some $[\chi_1, ..., \chi_n]$-variant of v satisfies ψ.
- If $\varphi = \mathbf{UQ}(\chi_1, ..., \chi_n, \psi)$, then v satisfies φ iff every $[\chi_1, ..., \chi_n]$-variant of v satisfies ψ.

Finally, then, a formula φ is *true in* **I** iff every variable assignment for **I** satisfies φ.

Note that, on this semantics, free individual variables are implicitly universally quantified; that is, if φ is a formula containing a free individual variable ν, then φ is true in **I** iff $\mathbf{UQ}(\nu, \varphi)$ is true in **I**. We do not have a similar metatheorem for formulas with free sequence variables because sequence variables are not explicitly quantified. It should be clear, however, that the above definition of

truth treats free sequence variables as if they were universally quantified as well: a formula φ containing a free sequence variable σ will be true in an interpretation **I** iff every variable assignment v satisfies φ, and hence iff every $[\sigma]$-variant of every variable assignment satisfies φ.

5. SCL AND TRADITIONAL FOL

We conclude with an important observation about the relation between SCL and first order logic. Consider, the following sentence from an unconstrained SCL language **L**:

$$(\forall x)(Px \leftrightarrow \neg Qx) \wedge (\forall xy)x = y$$

Because **L** is unconstrained, there is no distinction between predicate constants and individual constants. Hence, all such terms denote individuals in the domain. Such languages are useful, recall, in contexts where properties and relations are themselves considered "first-class citizens" and hence are included the domain of individuals. By the first conjunct in the above sentence, the individuals p and q that 'P' and 'Q' denote individuals must be distinct, as they must differ in their extensions. By the second conjunct, however, there is exactly one individual, and hence p and q cannot be distinct. Therefore, the sentence is false in all interpretations of **L**.[4]

This might lead one to charge that SCL's model theory does violence to the logical properties of traditional first-order logic. But it does not. The logical properties of the sentence above change only with respect to SCL languages that incorporate features that extend traditional first-order languages. Considered as a sentence of a TFO language (and many others midway between TFO and unconstrained), the the sentence is satisfiable relative to SCL's model theory no less than it is in traditional "Tarskian" model theory. More generally, then: The logical properties of TFO languages — those SCL language with no sequence variables, no variably polyadic predicates, no type-freedom, and no variables in predicate position — are *identical* regardless of whether they are interpreted according to the usual Tarskian semantics or according to SCL semantics; a formula of such a language will be true in all SCL interpretation iff it is true in all Tarskian interpretations. (The proof of this is quite simple, as it is easy to transform one type of interpretation into the other in a way that preserves truth.) Moreover, if one is unhappy with the differences in logical properties that can arise in a less constrained SCL language, there is a simple translation function that maps such a language to a theory in TFO language that has exactly the same expressive power.[5]

SCL is thus in a very precise sense a "conservative" extension of traditional first-order logic; it encompasses traditional first-order logic in all its many guises, but allows as well for the definition of much more powerful and flexible comformant languages. SCL thereby provides elegant solutions to both the uniformity problem and the excess baggage problem.

REFERENCES

[1] The Common Logic Working Group, "Common Logic Standard," URL = `http://cl.tamu.edu`.

[2] KIF Working Group, "Knowledge Interchange Format: Draft proposed American National Standard (dpANS)," NCITS.T2/98-004, URL = `http://logic.stanford.edu/kif/dpans.html`.

[3] Chen, W., M. Kifer, and D. S. Warren, "HiLog: A Foundation for higher-order logic programming," *Journal of Logic Programming* **15**(3), February 1993, pp. 187–230.

[4] The Common Logic Working Group, "Abstract Syntax and Semantics for SCL," URL = `http://cl.tamu.edu/docs/scl/scl-latest.html`.

DEPARTMENT OF PHILOSOPHY, TEXAS A&M UNIVERSITY, COLLEGE STATION, TX 77843-4237
E-mail address: `cmenzel@tamu.edu`

IHMC, UNIVERSITY OF WEST FLORIDA, PENSACOLA, FL 32501
E-mail address: `phayes@ihmc.us`

[4] We thank Ian Horrocks for the example, who came up with it to illustrate his dissatisfaction with an earlier incarnation of SCL. In that incarnation, there was no distinction between predicate and individual constants in any SCL language, and hence the sentence above turned out to be logically false. This pointed out an admittedly disturbing disconnect between the logical properties of SCL sentences relative to SCL's model theory and their logical properties relative to traditional Tarskian model theory. Revisions since then have added flexibility to SCL that undermines this objection.

[5] Briefly, one introduces new predicates $Holds_n$ for all n and maps every atomic sentence '$P(t_1, \ldots, t_n)$' of the non-TFO language in which 'P' is serves as both an individual and predicate constant into the sentence '$Holds(P, t_1, \ldots, t_n)$'.

A Controlled Language for Semantic Annotation and Interoperability in e-Business Applications[1]

M.Missikoff, F.Schiappelli, F.Taglino

LEKS, Lab for Enterprise Knowledge and Systems,
IASI-CNR (Italy),
{missikoff, fed_schi, taglino}
@iasi.cnr.it
http://leks.iasi.rm.cnr.it/home

Abstract. In this paper we present the basic ideas underlying a solution for software application interoperability in a business context. Key elements of our solution are a *Reference Ontology*, aimed at modelling the key aspects of a business domain, and a *Semantic Annotation Language*, SMAIL, used to associate semantic expressions, defined in terms of the reference ontology, to business elements.

Keywords. Semantic annotation, formal languages, ontology, business, interoperability.

1 Introduction

The considerable impact of the Internet on computer interconnection raised a high expectation in the area of application software interoperability. However, the experience shows that, despite the advances of current technology, two different legacy systems hardly can cooperate to carry out a common business task, even if data and procedures deal with the same business entities.

Our work is based on the idea that, to achieve interoperability among different systems, it is necessary to expose the actual semantics of data and programs, often deeply concealed by superficial differences, such as naming, syntactic and structural discrepancies.

To provide an effective solution to this problem it is necessary to shift towards a semantic level of interaction, i.e., semantic interoperability; to this end we need to make explicit the semantics hidden in, e.g., an application interface. A promising approach to achieve semantic interoperability requires the use of a Reference Ontology (RO) and a Semantic Annotation Language, based on the former.

Semantic Annotation (SA) has been proposed in literature mostly to annotate documents and web pages [SeWeb]. Only few proposals are aimed at the creation of additional structures that represent (in a formal, controlled way) the semantic content of a web resource (e.g., a document, a business process or an eService).

Among typical applications of semantic annotation, we can find:

- *Document Management*, for semantic search;
- *Knowledge Management*, for organization and retrieval of enterprise knowledge;
- *Web Services* publishing and discovery, with semantic matchmaking of requested and offered services;
- *Semantic Interoperability*, by annotating local resources (information and processes) to support business cooperation among enterprise software applications.

In the literature, the first two applications have attracted most of the attention. They are addressed by solutions referred to as "human-oriented" annotations. This kind of annotation solutions are provided by systems such as Annotea [KPS01], Annotation System for Semantic Web [VR02], Trellis [GV02]. Such systems are interactive environments that allow users to add, in a ***descriptive*** way (plain text), an annotation representing the content of the documents.

A second important class is represented by the solutions referred to as "machine-oriented" annotations. This kind of SA is provided by systems such as MnM [VM*02], SMORE [KP*02], SHOE Knowledge Annotator [LS*97], COHSE [BG01]. These solutions aim at representing, in a ***formal*** way, the conceptual content of a given web resource. The user annotates segments of text, typically in a web page, using tags based on the concepts defined in an *Ontology*. This activity is known as "ontology driven mark-up".

Our work evolves along the second line, since we propose a solution for ontology-driven semantic annotation, aimed at the interoperability of software applications in an e-business context. The main difference, with respect to the previously mentioned

[1] This work was partially supported by IST European Project Harmonise

tools, is that they mainly aim at enriching a web page, embedding the SA in the document itself. Our method allows formal, ontology-based, structures to be created externally (but tightly connected) to the web resource, on the line of the OntoMat Annotizer [HS02, HS03] approach. We refer to this formal structure as the "semantic image" of the resource. In this way, the annotation, being not embedded, can be associated to any kind of resource, such as a video, a sound, or a web service. Furthermore, starting from a collection of web resources, it is possible to gather their semantic images to build a semantic index for a Semantic Web architecture. Semantic search and matchmaking can be implemented for fast retrieval of web resources, based on the actual knowledge they carry.

Another important characteristic of our approach is that the proposed annotation language is tightly controlled, based on a reference ontology that, in its terminological content, is part of the language.

2 Semantic Annotation for Enterprise Interoperability

The goal of interoperability is to allow different software applications to exchange data and services, despite the fact that the two software systems were not originally conceived for cooperation. It is well known that, even if data and procedures deal with the same business entities, existing software applications exhibit deep differences in their internal organization, database schemas, software architectures, and other important technical characteristics, that hinder a smooth cooperation.

To solve such a problem, it is necessary to identify the business entities addressed, i.e., the semantics of the information elements and operations that are involved in the cooperation, beyond the syntactic and structural differences.

The problem that we address bears some similarity with the area of heterogeneous information sources integration, addressed in the database field. In this area, two basic approaches have been proposed: global-as-view (GAV) [MP*97, TRV98, GB*99] and local-as-view (LAV) [Hal01, Lenz02]. The LAV approach (for information interoperability) implies that each application system interacts with any other system as if its own data organisation was the only existing solution, i.e., as if all the other software applications were organised in the same way. The LAV approach [KLS95, AD98, CD*01] does not require a global schema to be built, but the existence of a common view of the business scenario where the cooperation takes place. This common view, in our approach, is represented by a shared Reference Ontology (RO). The RO, built by a team of domain experts, provides precise and formal (therefore, computer processable) definitions of relevant (for the business context) domain entities. The terminological component is used to annotate the resources managed by the cooperating systems.

In this paper we restrict the focus to information interoperability; the set of information elements of a given software application, participating in the interoperability process, will be referred to as PLCS (Public Local Conceptual Schema).

2.1 The Annotation Process

The semantic annotation process is a critical one; it represents a first phase in which a given legacy system is confronted with its inherent inclination to interoperate within a given business community. In fact, the reference ontology RO is assumed to be a proper representation of the business domain, and in particular of the part that will be involved in the networked business activities. It is fair to assume that a given information or service that is not defined in the RO is not of interest of the community. Therefore, in the semantic annotation process, a PLCS element that cannot be annotated is assumed to be (at that moment) of scarce interest for the networked business. But, since the reality continuously evolves, we assume that suitable mechanisms will be implemented to update the RO whenever a sufficient consensus is reached in order to modify it.

Besides the cases where some PLCS elements fall outside the (ontological) scope of the business domain, there are other cases where an annotation that precisely captures the intended meaning of a PLCS element is not possible. We refer to these cases as "annotation mismatches". In fact we can have:

Lossless annotation: when the annotation fully captures the intended meaning,

Lossy annotation: when the annotation fails to fully representing the intended meaning.

In the first case, a PLCS element exactly corresponds to a concept in the RO or its meaning can be precisely expressed by a suitable composition of concepts. In the second case, the meaning of a PLCS element does not have a matching concept in the ontology, nor the possibility of compositionally express it, since either:

- the intended meaning is outside the scope of the RO;

- the PLCS element is not sufficiently refined (i.e., it does not match the accuracy level of the ontology) (*underspecification*)
- the PLCS element present a level of refinement not deemed useful, that does not match the level of refinement of the RO (*overspecification*).

Annotation mismatches may derive from different organizations of information in the PLCS and RO, but also from different views of the world. However, having in mind a specific concept (or a set of concepts), represented in two different models, there are a limited number of possible divergences. We have a first list of differences that the knowledge engineer must consider in annotating a PLCS. We present them divided in the two broad categories previously introduced: lossless and lossy (see an excerpt in Table 1).

Lossless mismatches	
Path-Naming	different labels for the same content (the attribute names *Name* and *Denomination* to indicate a hotel name)
Encoding	different formats of data or units of measure (a *Price* expressed in dollars and in euro)
Structuring	different structures for the same content (an *Address* represented as a string or a composition of the *Street_name*, and *Street_number* fields)
Lossy mismatches	
Content	different content denoted by the same concept - typically expressed by enumeration (the *hotel services* concept described by different enumeration items)
Coverage	presence/absence of information (the *mobile phone* in the PLCS, but not in the RO)
Precision	the accuracy of information (the *distance* expressed by an integer value or by strings such as *near, far*)
Abstraction	level of specialisation refinement of the information (the distinction bw *indoor* and *outdoor swimming pool* versus a generic *swimming pool* concept)
Granularity	level of decomposition refinement of the information (the *restaurant* represented as a whole or as an aggregation of a *terrace* and an *indoor_rooms*)

Table 1. Sorts of mismatches

Semantic annotation is a critical process that requires deep knowledge on the domain, the RO and the legacy system. Given a software application, and in particular its Public Local Conceptual Schema, its semantic annotation is accomplished by performing the following steps:

- *Identification of the PLCS elements*. The first step consists in the identification of the elements, essentially information (provided or requested), necessary for the system to participate in the cooperation with other systems.
- *Identification of the intended meaning*. Then, each PLCS element is clearly assigned with an "intuitive" semantics, by associating the business elements represented (informal annotation).
- *Identification of the related concepts in the RO*. The most appropriate concepts that express the intended meanings are then chosen.
- *Definition of the semantic expressions*. By using the selected RO concepts, an expression that specifies the intended meaning is constructed. For each PLCS element, the best fitting annotation expression is built (formal semantic annotation expressions).
- *Semantic coverage assessment*. The intended meaning of the PLCS element is contrasted with the semantic annotation expression, to see if there is any loss of semantics.
- *Association of the semantic annotation expressions to the PLCS elements*. Finally, each semantic annotation expression can be associated to the correspondent PLCS element, using the correct connective (for lossless or lossy cases). In the following section, this process is further elaborated.

3 SMAIL: a Controlled Semantic Annotation Language

To create the semantic annotation expressions, we propose to use SMAIL (Semantic Mediation and Application Interoperability Language).

It is important to note that SMAIL is characterized by a closed vocabulary. This means that, unlike the other annotation languages, the user cannot define his/her own terms nor named concepts. The sentences of SMAIL can be constructed only using the terms (i.e., concepts) defined in the Reference Ontology. A naming policy, inventing labels for variables, subroutines, relation names, etc., is one of the most critical aspects of the development of an information system. For this reason, one of the main characteristics of SMAIL is the fact that, in building a semantic expression, the user needs to look at the ontology and can only select terms denoting defined concepts. Therefore the terminological elements of

the ontology become part of the language; the generation of the annotation expressions is performed by a composition and/or transformation of ontology elements.

More in detail, an annotation expression is composed of a left-hand-side and a right-hand-side. On the left-hand-side only a name (or path) of an information element in the PLCS can appear; it identifies the PLCS element to be annotated. On the right-hand-side only ontology elements, and SMAIL constructors, can appear. As anticipated we have lossless and lossy annotations and we introduce specific connectives to express these kinds of annotation.

Lossless annotations can be expressed with a Semantic Equivalence connective

PLCS_elem =: SA_expression

Lossy annotations can be expressed with Over/Underspecification connectives

PLCS_elem >: SA_expression (Local Overspecification)

PLCS_elem <: SA_expression (Local Underspecification)

3.1 The SMAIL Grammar

In the following we give a formal specification of the SMAIL language by defining the grammar that generates it.

$$G_{SMAIL} = (N,T,P,\Sigma)$$

where

- N is the set of non-terminal symbols,
- T is the set of terminal symbols, where labels are terms in the ontology
- P is the set of production rules,
- Σ is the start symbol.

In Fig.2,3 the element of the 4-tuple are presented in detail. Terminal symbols are in *italic*, while non-terminal symbols are in UPPER CASE.

- N = {SE,OE,OE_SEQ,COND,EXP}
- T = $V_{ont} \cup O \cup A \cup D \cup \{\perp,\varepsilon\}$
  - V_{ont} = set of the ONTOLOGY TERMS
  - O = {*,* , *;* , *and* , *or* , *(* , *)* }
  - A = {*>*, *>=*, *<*,*<=*, *=*, *+*,*-*,***,*/*,*%*}
  - D={*0..9*}

Figure 2: Nonterminal and Terminal sets for G_{SMAIL}

SE ::= ⊥
  | EXP
EXP ::= OE COND
  | OE_SEQ COND
  | ARITM_EXP COND
  | BOOL_EXP
OE ::= *oelem* ... *oelem* ∈ V_{ont}
OE_ SEQ ::= OE
  | OE, OE_SEQ
ARITM_EXP ::= grammar for the arithmetic expressions, where the operands are either ontology elements (oelem) or real numbers
BOOL_EXP ::= *(*EXP*)*
  | *(*EXP*) and* BOOL_EXP
  | *(*EXP*) or* BOOL_EXP
COND ::= ε | *;* ...grammar for the boolean expressions, where the operands are either ontology elements (oelem) or real numbers

Figure 3: Production rules for G_{SMAIL}

Please, note that SMAIL is not intended for direct use by a knowledge engineer. It is at the basis of the annotation tool associated to SymOntoX [MT02], the Ontology Management System developed at LEKS, IASI-CNR. Therefore, in actual applications, the complexity of the annotation language is shielded from the user by a friendly graphical user interface. Furthermore, we are currently working on a version of OWL [GH03] referred to as SMOWL, to cast the annotation approach of SMAIL into an XML-based ontology language.

4 A few examples

Some examples of semantic mismatches introduced in Table 1 and the SMAIL expressions aimed at solving them are shown in Tables 2-3. In the *Address* case, a simple string is annotated with a concatenation of two strings. Please note that syntactical details, such as separators in the PLCS *Address*, are not dealt with here since we focus on the semantic aspect of annotation. Such implementation details will be addressed in later phases, when semantic annotation will be used to build semantic adaptors for interoperability [MT03]. Please note that the nihil symbol (assumed to be defined in the RO) is used to denote the undefinedness. Furthermore, the *Swimming Pool* example, a concept *Swimming_Pool* is supposed to exist in the RO, defined as a generalization of the *Indoor_Sw_Pool* and *Outdoor_Sw_Pool* concepts.

PLCS **Hotel**	**RO** **Hotel**	**Mismatch**
Name: *literal*	Denomination: *literal*	Naming
Address: *literal*	Address [Street_name: *literal* Street_number: *literal*]	Structuring
Services: enum (‘security box’, ‘hamam’, ‘parking’)	Services: enum (‘safe’, ‘sauna’, ‘ironing center’)	Content
Telephone: *literal* Mobile-phone: *literal* Fax: *literal*	Contact_Info [Phone: *literal* Fax: *literal* Email: *literal*]	Coverage, Structuring, Naming
Location: enum (‘near city’, ‘far city’, ‘near airport’, ‘far airport’)	Location [Distance: *literal* from: enum (‘city’, ‘airport’)]	Precision
SwimmPool: enum (‘yes’, ‘no’)	Facilities: [Indoor_Sw_Pool: enum (‘yes’, ‘no’) Outdoor_Sw_Pool: enum (‘yes’, ‘no’)]	Abstraction, Structuring, Naming
Restaurant: enum (‘yes’, ‘no’)	Restaurant: [Terrace: enum (‘yes’, ‘no’) Indoor_Room: enum (‘yes’, ‘no’)]	Granularity

Table 2: Mismatches examples

Semantic Annotation	**Mismatch**
PLCS.Hotel.Name=: RO.Hotel.Denomination	Naming
PLCS.Hotel.Address.Location =: RO.Hotel.Address.Street_name, RO.Hotel.Address.Street_number	Structuring
PLCS.Hotel.Services(“security box”) =: RO.Hotel.Services(“safe”) PLCS.Hotel.Services(“hamam”) =: RO.Hotel.Services(“sauna”) PLCS.Hotel.Services(“parking”) <: ⊥	Content
PLCS.Hotel.Mobile-phone <: ⊥	Coverage
PLCS.Hotel.Location(“near city”) =: (RO.Hotel.Location.From(“City”)) and (RO.Hotel.Location.Distance(); RO.Hotel.Location.Distance<=2) PLCS.Hotel.Location(“near airport”) =: (RO.Hotel.Location.From(“Airport”)) and (RO.Hotel.Location.Distance(); RO.Hotel.Location.Distance<=2) …	Precision
PLCS.Hotel.SwimmPool =: RO.Hotel.Facilities.Swimming_Pool	Abstraction
The ontology is richer than the PLCS	Granularity

Table 3: Semantic annotation examples

5 Conclusions

In this paper we briefly presented the main issues of SMAIL, an ontology-based semantic annotation language conceived for semantic interoperability among software applications. The proposed language is used to assign meaning to elements of legacy systems that are exchanging information with each other. Semantic annotation is the first, preliminary phase, to allow semantic interoperability.

The proposed language is based on a Reference Ontology that determines the expressions that can be built. In this way any possible expression has a precise, unambiguous semantics. SMAIL is, therefore, a controlled language with a closed, ontology-based, vocabulary.

Along this line, a first solution for semantic interoperability has been developed within the European Project Harmonise [Harmo],[Miss*03], that originated the presented work.

References

[AD98] Abiteboul, S., Duschka, O.: Complexity of answering queries using materialized views. In: Proc. Of PODS'98. (1998) 254–265

[BG01] Sean Bechhofer, Carole Goble. Towards Annotation using DAML+OIL. K-CAP 2001 workshop on Knowledge Markup and Semantic Annotation, Victoria B.C, October 2001

[CB84] D.J.Cook, H.E.BEZ; Computer Mathematics; Cambridge University Press, 1984

[CD*01] Calvanese, D., De Giacomo, G., Lenzerini, M., Nardi, D., Rosati, R.: Data integration in data warehousing. Int. J. of Cooperative Information Systems 10 (2001) 237–271

[GB*99] Goh, C.H., Bressan, S., Madnick, S.E., Siegel, M.D.: Context interchange: New features and formalisms for the intelligent integration of information. ACM Trans. on Information Systems 17 (1999) 270–293

[GH03] Deborah L. McGuinness and Frank van Harmelen eds. Web Ontology Language (OWL): Overview, W3C Last Call Working Draft 31 March 2003.

[GV02] Gil, Yolanda and Varun Ratnakar: Trusting Information Sources One Citizen at a Time. Proceedings of the First International Semantic Web Conference (ISWC), Sardinia, Italy, June 2002.

[Hal01] Halevy, A.Y.: Answering queries using views: A survey. VLDB Journal 10 (2001) 270–294

[Harmo] www.harmonise.org

[HS02] S. Handschuh and S. Staab. Authoring and annotation of web pages in CREAM. In The Eleventh International World Wide Web Conference (WWW2002), Honolulu, Hawaii, USA 7-11 May, 2002.

[HS03] S. Handschuh, S. Staab, R.Volz: On Deep Annotation, WWW-2003, Budapest, Hungary, May 20-24, 2003

[HU79] J.E.Hopcroft, J.D.Ullmann; Introduction to Automata Theory, Languages and Computation; Addison-Wesley, 1979

[KLS95] Kirk, T., Levy, A.Y., Sagiv, Y., Srivastava, D.: The Information Manifold. In: Proceedings of the AAAI 1995 Spring Symp. on Information Gathering from Heterogeneous, Distributed Enviroments. (1995) 85–91

[KP*02] Kalyanpur, A., Bijan Parsia, James Hendler, Jennifer Golbeck. SMORE - Semantic Markup, Ontology, and RDF Editor. in Submitted to WWW2003. 2002

[KPS01] J. Kahan, M. Koivunen, E. Prud'Hommeaux and R. Swick: Annotea: Open RDF Infrastructure for Shared Web Annotations. In Proc. of the WWW10 International Conference. Hong Kong, 2001.

[Lenz02] Lenzerini, M.: Data Integration: A Theoretical Perspective, Proceedings. of PODS Conference, 2002.

[LS*97] S. Luke, L. Spector, D. Rager, and J. Hendler. Ontology-based Web agents. In Proceedings of the First International Conference on Autonomous Agents, pages 59–66. ACM, 1997.

[Miss*03] M.Missikoff et Al.; The Architecture of the Project Harmonise; Proc. Of ENTER Conference, Helsinki, Jan 2003.

[MP*97] Garcia-Molina, H., Papakonstantinou, Y., Quass, D., Rajaraman, A., Sagiv, Y., Ullman, J.D., Vassalos, V., Widom, J.: The TSIMMIS approach to mediation: Data models and languages. J. of Intelligent Information Systems 8 (1997) 117–132

[MT02] M.Missikoff, F.Taglino; Business and Enterprise Ontology Management with SymOntox; Proc. Of ISWC 02, Sardinia, June 2002.

[MT03] M.Missikoff, F.Taglino; The Architecture of an Ontology-based Platform for Semantic interoperability; The Handbook of Ontologies in Information Systems (S.Staab and R.Studer, Eds), Springer Verlag, to appear, 2003.

[NLF99] Naumann, F., Leser, U., Freytag, J. C.: Quality-driven integration of heterogeneous information systems. In Proceedings. of the 25th Int. Conf. on Very Large DataBases (VLDB'99), pages 447–458, 1999.

[SeWeb] annotation.semanticweb.org

[TRV98] Tomasic, A., Raschid, L., Valduriez, P.: Scaling access to heterogeneous data sources with DISCO. IEEE Trans. on Knowledge and Data Engineering 10 (1998) 808–823

[UG96] M.Uschold, M.Gruninger; Ontologies: Principles, Methods and Applications; The Knowledge Engineering Review, V.11, N.2, 1996.

[VM*02] Maria Vargas-Vera, Enrico Motta, John Domingue, Mattia Lanzoni, Arthur Stutt and Fabio Ciravegna, MnM: Ontology Driven Tool for Semantic Markup. European Conference on Artificial Intelligence (ECAI 2002). In proceedings of the Workshop Semantic Authoring, Annotation & Knowledge Markup (SAAKM 2002). Lyon France, July 22-23, 2002.

[VR02] S.Venkatasubramani, R.K.V.S.Raman: Annotations In Semantic Web. In the Eleventh International World Wide Web Conference (WWW2002)

On Semantic Interoperability and the Flow of Information

Marco Schorlemmer[1] Yannis Kalfoglou[2]

[1]School of Informatics, The University of Edinburgh.
[1]Escola Universitària de Tecnologies d'Informació i Comunicació, Universitat Internacional de Catalunya.
[2]School of Electronics and Computer Science, University of Southampton.
marco@cir.unica.edu; y.kalfoglou@ecs.soton.ac.uk

Abstract. We discuss current approaches that, for the sake of automation, provide formal treatments to the problem of semantic interoperability and integration, and we reflect upon the suitability of the Barwise-Seligman theory of information flow as a candidate for a theoretical framework that favours the analysis and implementation of semantic interoperability scenarios.

1 Introduction

In a large-scale, distributed, and often deregulated environment such as the World Wide Web, systems integration is seen as the viable solution in order to cross organisational and market boundaries and hence enable applications deployment in a wide variety of domains, ranging from e-commerce to e-Science Grid projects. Although systems integration has been studied and applied for years in closed and controlled environments within organisational boundaries and vertical market segments, the situation is quite different in the emergent Semantic Web [17].

One of the ambitious goals of the Semantic Web is for systems to be able to exchange information and services with one another in semantically rich and sound ways [4]. The semantics, being a key aspect of the Semantic Web, should therefore be exposed, interpreted, and used to enable services and to support distributed applications. This means that semantics should be understood, verified against an agreed standard, and used to endorse and validate reliable information exchange. These high-level goals were similar to those pursued within the context of database schema and information integration, where the problem of semantic heterogeneity among different data sources had to be tackled [21, 9]. If these goals were achieved, two systems could be interoperable, moreover, semantically interoperable.

Semantic Interoperability and Integration

Semantic interoperability and semantic integration are much contested and fuzzy concepts which have been used over the past decade in a variety of contexts and works. As reported in [17], in addition, both terms are often used indistinctly, and some view these as the same thing.

The ISO/IEC 2382 Information Technology Vocabulary defines interoperability as "the capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units." In a debate on the mailing list of the IEEE Standard Upper Ontology working group, a more formal approach to semantic interoperability was advocated: Use logic in order to guarantee that after data were transmitted from a sender system to a receiver, all implications made by one system had to hold and be provable by the other, and there should be a logical equivalence between those implications.[1]

With respect to integration, Uschold and Grüninger argue that "two agents are semantically integrated if they can successfully communicate with each other" and that "successful exchange of information means that the agents understand each other and there is guaranteed accuracy" [25]. According to Sowa, to integrate two ontologies means to derive a new ontology that facilitates interoperability between systems based on the original ontologies, and he distinguishes three levels of integration [22]: *Alignment*—a mapping of concepts and relations to indicate equivalence—, *partial compatibility*—an alignment that supports equivalent inferences and computations on equivalent concepts and relations—, and *unification*—a one-to-one alignment of all concepts and relations that allows any inference or computation expressed in one ontology to be mapped to an equivalent inference or computation in the other ontology.

Although these definitions of semantic interoperability and integration are by no means exhaustive, and despite the blurred distinction between these two concepts, they are indicative of two trends: on one hand, we have deliberately abstract and rather ambiguous definitions of what semantic interoperability and integration could potentially achieve, but not how to achieve it; and on the other hand, we have formal and mathematically rigorous approaches, which allow for the automatisation of the process of establishing semantic interoperability and integration.

[1]Message thread on the SUO mailing list initiated at http://suo.ieee.org/email/msg07542.html

2 Formal Approaches to Semantic Interoperability

The above definitions also reveal a common denominator, that of *communication*. For two systems to interoperate there must be an established form of communication and the right means to achieve this efficiently and effectively. To provide the means for the former, practitioners have been studying and applying consensual formal representations of domains, like ontologies; these act as the protocol to which systems have to agree upon in order to establish interoperability. However, there is an ongoing debate with regard to the later means. The argument goes like that: *Having established a protocol to which communication will be based, i.e., ontologies, what is the best way to effectively make those semantically interoperable and to integrate them?*

A practical angle of viewing this problem is when we focus on the notion of equivalence. That is, we would like to establish some sort of correspondence between the systems, and subsequently their ontologies, to make them interoperable and that could be done by reasoning about equivalent constructs of the two ontologies. However, equivalence is not a formally and consensually agreed term, neither do we have mechanisms for doing that. Hence, if we are to provide a formal, language-independent mechanism of semantic interoperability and integration, we need to use some formal notion of equivalence. And for a precise approximation to equivalence the obvious place to look at is Logic.

In this sense first-order logic seems the natural choice: Among all logics it has a special status due to its expressive power, its natural deductive systems, and its intuitive model theory based on sets. In first-order logic, equivalence is approximated via the precise model-theoretic concept of *first-order equivalence*. This is the usual approach to formal semantic interoperability and integration; see e.g., [3, 5, 16, 25]. In Ciocoiu and Nau's treatment of the translation problem between knowledge sources that have been written in different knowledge representation languages, semantics is specified by means of a common ontology that is expressive enough to interpret the concepts in all agents' ontologies [5]. In that scenario, two concepts are equivalent if, and only if, they share exactly the same subclass of first-order models of the common ontology.

But this approach has its drawbacks. First, such formal notion of equivalence requires the entire machinery of first-order model theory, which includes set theory, first-order structures, interpretation, and satisfaction. This appears to be heavyweight for certain interoperability scenarios. Madhavan et al. define the semantics in terms of instances in the domain [14]. This is also the case, for example, in Stumme and Maedche's ontology merging method, FCA-Merge [23], where the semantics of a concept symbol is captured through the instances classified to that symbol. These instances are documents, and a document is classified to a concept symbol if it contains a reference that is relevant to the concept.[2] For FCA-Merge, two concepts are considered equivalent if, and only if, they classify exactly the same set of documents.

Menzel makes similar objections to the use of first-order equivalence and proposes an axiomatic approach instead, inspired on property theory [24], where entailment and equivalence are not model-theoretically defined, but axiomatised in a logical language for ontology theory [15].

Second, since model-theory does not provide proof mechanisms for checking model equivalence, this has to be done indirectly via the theories that specify the models. This assumes that the logical theories captured in the ontologies are complete descriptions of the intended models (Uschold and Grüninger call these *verified ontologies* [25]), which will seldom be the case in practice.

Furthermore, Corrêa da Silva at al. have shown situations in which even a common verified ontology is not enough, for example when a knowledge base whose inference engine is based on linear logic poses a query to a knowledge base with the same ontology, but whose inference engine is based on relevance logic [6]. The former should not accept answers as valid if the inference carried out in order to answer the query was using the contraction inference rule, which is not allowed in linear logic. Here, two concepts will be equivalent if, and only if, we can infer exactly the same set of consequences on their distinct inference engines.

Last, but certainly not least, first-order model theory was originally devised for mathematics in order to precisely describe the mathematical concepts of *truth* and *proof*. This semantics proved ill-suited for tackling problems which lay outside the scope of the mathematical realm, such as common-sense reasoning, natural language processing, or planning. Since the early days of AI, the community has been exploring several extensions of first-order logic in order to overcome these shortcomings [8].

But in spite of despising a model-theoretic approach to semantic interoperability, we want to step back and reflect on the necessity of settling upon a particular understanding of semantics for the sake of formalising and automating semantic interoperability. A careful look at the several formal approaches to semantic integration mentioned above reveals many different understandings of semantics depending on the interoperability scenario under consideration. Hence, what we need in order to successfully tackle the problem of semantic interoperability is not so much a framework that establishes a particular semantic perspective (model-theoretic, property-theoretic, instance-based, etc.), but instead we need a framework that successfully captures semantic interoperability despite the different treatments of semantics.

[2] This is done by means of a linguistic pre-analysis.

An Information-Centred Approach

In this paper we observe that, in order for two systems to be semantically interoperable (or semantically integrated) we need to align and map their respective ontologies such that *the information can flow*. Consequently, we believe that a satisfactory formalisation of semantic interoperability can be built upon a mathematical theory capable of describing under which circumstances information flow occurs.

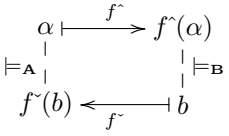
Although there is no such theory yet, the most promising effort was initiated by Barwise and Perry with situation semantics [1], which was further developed by Devlin into a theory of information [7]. Barwise and Seligman's channel theory is currently the latest stage of this endeavour [2], in which they propose a mathematical model that aims at establishing the laws that govern the flow of information. It is a general model that attempts to describe the information flow in any kind of distributed system, ranging form actual physical systems like a flashlight connecting a bulb to a switch and a battery, to abstract systems such as a mathematical proof connecting premises and hypothesis with inference steps and conclusions. Barwise and Seligman's theory is therefore a good place to start establishing a foundation for formalising semantic interoperability.

In channel theory, each component of a distributed systems is represented by an *IF classification* $\mathbf{A} = (tok(\mathbf{A}), typ(\mathbf{A}), \models_{\mathbf{A}})$, consisting of a set of *tokens* $tok(\mathbf{A})$, a set of *types* $typ(\mathbf{A})$ and a *classification relation* $\models_{\mathbf{A}} \subseteq tok(\mathbf{A}) \times typ(\mathbf{A})$ that classifies tokens to types.[3] It is a very simple mathematical structure that effectively captures the local syntax and semantics of a community for the purpose of semantic interoperability.

For the problem of semantic interoperability that concerns us here the components of the distributed systems are the ontologies of the communities that desire to communicate. We model them as IF classification, such that the syntactic expressions that a community uses to communicate constitute the types of the IF classification, and the meaning that these expressions take within the context of the community are represented by the way tokens are classified to types. Hence, *the semantics is characterised by what we choose to be the tokens of the IF classification*, and depending on the particular semantic interoperability scenario we want to model, types, tokens, and its classification relation will vary. For example, in FCA-Merge [23], types are concept symbols and tokens particular documents, while in Ciocoiu and Nau's scenario [5] types are expressions of knowledge representation languages and tokens are first-order structures. The crucial point is that *the semantics of the interoperability scenario crucially depends on our choice of types, tokens and their classification relation for each community*.

[3] We are using the prefix 'IF' (information flow) in front of some channel-theoretic constructions to distinguish them from their usual meaning.

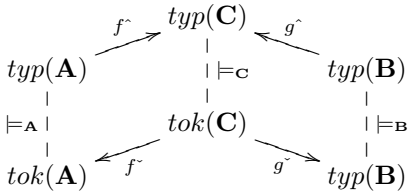
The flow of information between components in a distributed system is modelled in channel theory by the way the various IF classifications that represent the vocabulary and context of each component are connected with each other through *infomorphisms*. An infomorphism $f = \langle f^{\wedge}, f^{\vee} \rangle : \mathbf{A} \rightleftarrows \mathbf{B}$ from IF classifications $\mathbf{A}$ to $\mathbf{B}$ is a contravariant pair of functions $f^{\wedge} : typ(\mathbf{A}) \to typ(\mathbf{B})$ and $f^{\vee} : tok(\mathbf{B}) \to tok(\mathbf{A})$ satisfying the following fundamental property, for each type $\alpha \in typ(\mathbf{A})$ and token $b \in tok(\mathbf{B})$:



$$f^{\vee}(b) \models_{\mathbf{A}} \alpha \quad \text{iff} \quad b \models_{\mathbf{B}} f^{\wedge}(\alpha)$$

A *distributed IF system* $\mathcal{A}$ consists then of an indexed family $cla(\mathcal{A}) = \{\mathbf{A}_i\}_{i \in I}$ of IF classifications together with a set $inf(\mathcal{A})$ of infomorphisms all having both domain and codomain in $cla(\mathcal{A})$.

A basic construct of channel theory is that of an *IF channel*—two IF classifications $\mathbf{A}$ and $\mathbf{B}$ connected through a core IF classification $\mathbf{C}$ via two infomorphisms f and g:



This basic construct captures the information flow between components $\mathbf{A}$ and $\mathbf{B}$. Crucial in Barwise and Seligman's model is that it is the particular tokens that carry information and that information flow crucially involves both types and tokens.

In fact, as we shall see next, our approach uses this model to approximate the intuitive notion of equivalence necessary for achieving semantic interoperability with the precise notion of a type equivalence that is supported by the connection of tokens from $\mathbf{A}$ with tokens from $\mathbf{B}$ through the tokens of the core IF classification $\mathbf{C}$. This provides us with the general framework of semantic interoperability we are after, one that accommodates different understandings of semantics—depending on the particularities of the interoperability scenario—whilst retaining the core aspect that will allow communication among communities: a connection through their semantic tokens.

3 Semantic Interoperability via Information Channels

The key channel-theoretic construct we are going to exploit in order to outline our formal framework for seman-

tic interoperability is that of a *distributed IF logic*. This is the logic that represents the information flow occurring in a distributed system. In particular we will be interested in a restriction of this logic to the language of those communities we are attempting to integrate. As we proceed, we will hint at the intuitions lying behind the channel-theoretical notions we are going to use; for a more in-depth understanding of channel theory we point the interested reader to [2].

IF Theory and Logic

Suppose two communities $\mathbf{A}_1$ and $\mathbf{A}_2$ need to interoperate, but are using different ontologies in different ontologies. To have them semantically interoperating will mean to know the semantic relationship in which they stand to each other. In terms of the channel-theoretic context, this means to know an *IF theory* that describes how the different types from $\mathbf{A}_1$ and $\mathbf{A}_2$ are logically related to each other.

Channel theory has been developed based on the understanding that information flow results from regularities in a distributed system, and that it is by virtue of regularities among the connections that information of some components of a system carries information of other components. These regularities are implicit in the representation of the systems' components and its connections as IF classifications and infomorphisms, but, in order to derive a notion of equivalence on the type-level of the system we need to capture this regularity in a logical fashion. This is achieved with IF theories and IF logics in channel theory.

An *IF theory* $T = \langle typ(T), \vdash \rangle$ consists of a set $typ(T)$ of types, and a binary relation $\vdash$ between subsets of $typ(T)$. Pairs $\langle \Gamma, \Delta \rangle$ of subsets of $typ(T)$ are called *sequents*. If $\Gamma \vdash \Delta$, for $\Gamma, \Delta \subseteq typ(T)$, then the sequent $\Gamma \vdash \Delta$ is called a *constraint*. T is *regular* if for all $\alpha \in typ(T)$ and all sets $\Gamma, \Gamma', \Delta, \Delta', \Sigma', \Sigma_0, \Sigma_1$ of types:

1. *Identity:* $\alpha \vdash \alpha$
2. *Weakening:* If $\Gamma \vdash \Delta$, then $\Gamma, \Gamma' \vdash \Delta, \Delta'$
3. *Global Cut:* If $\Gamma, \Sigma_0 \vdash \Delta, \Sigma_1$ for each partition $\langle \Sigma_0, \Sigma_1 \rangle$ of Σ', then $\Gamma \vdash \Delta$.[4]

Regularity arises from the observation that, given any classification of tokens to types, the set of all sequents that are satisfied[5] by all tokens always fulfil these three properties. In addition, given a regular IF theory T we can generate a classification $Cla(T)$ that captures the regularity specified in its constraints. Its tokens are partitions $\langle \Gamma, \Delta \rangle$ of $typ(T)$ that are *not* constraints of T, and types are the types of T, such that $\langle \Gamma, \Delta \rangle \models_{Cla(T)} \alpha$ iff $\alpha \in \Gamma$.[6]

[4] A partition of Σ' is a pair $\langle \Sigma_0, \Sigma_1 \rangle$ of subsets of Σ', such that $\Sigma_0 \cup \Sigma_1 = \Sigma'$ and $\Sigma_0 \cap \Sigma_1 = \emptyset$; Σ_0 and Σ_1 may themselves be empty (hence it is actually a quasi-partition).

[5] Defined further below.

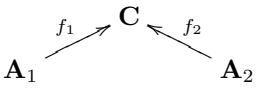
[6] These tokens may not seem obvious, but these sequents code the content of the classification table: The left-hand sides of the these sequents indicate which to which types they are classified, while the right-hand sides indicate to which they are not.

The IF theory we are after in order to capture the semantic interoperability between communities $\mathbf{A}_1$ and $\mathbf{A}_2$ is an IF theory on the union of types $typ(\mathbf{A}_1) \cup typ(\mathbf{A}_2)$ that respects the local IF classification systems of each community—the meaning each community attaches to its expressions—but also interrelates types whenever there is a similar semantic pattern, i.e., a similar way communities classify related tokens. That is the type language we speak in a semantic interoperability scenario, because we want to know when type α of one component corresponds to a type β of another component. In such an IF theory a sequent like $\alpha \vdash \beta$, with $\alpha \in typ(\mathbf{A}_1)$ and $\beta \in typ(\mathbf{A}_2)$, would represent an implication of types among communities that is in accordance to how the tokens of different communities are connected between each other. Hence, a constraint $\alpha \vdash \beta$ will represent that every α is a β, together with a constraint $\beta \vdash \alpha$ we obtain type equivalence.

Putting the idea of an IF classification with that of an IF theory together we get an *IF logic* $\mathfrak{L} = \langle tok(\mathfrak{L}), typ(\mathfrak{L}), \models_{\mathfrak{L}}, \vdash_{\mathfrak{L}}, N_{\mathfrak{L}} \rangle$. It consists of an IF classification $cla(\mathfrak{L}) = \langle tok(\mathfrak{L}), typ(\mathfrak{L}), \models_{\mathfrak{L}} \rangle$, a regular IF theory $th(\mathfrak{L}) = \langle typ(\mathfrak{L}), \vdash_{\mathfrak{L}} \rangle$ and a subset of $N_{\mathfrak{L}} \subseteq tok(\mathfrak{L})$ of *normal tokens*, which satisfy all the constraints of $th(\mathfrak{L})$; a token $a \in tok(\mathfrak{L})$ satisfies a constraint $\Gamma \vdash \Delta$ of $th(\mathfrak{L})$ if, when a is of all types in Γ, a is of some type in Δ. An IF logic $\mathfrak{L}$ is *sound* if $N_{\mathfrak{L}} = tok(\mathfrak{L})$.

Distributed IF Logic

The sought after IF theory is the IF theory of the distributed IF logic of an IF channel



that represents the information flow between $\mathbf{A}_1$ and $\mathbf{A}_2$. This channel can either be stated directly, or indirectly by some sort of partial alignment of $\mathbf{A}_1$ and $\mathbf{A}_2$.

The logic we are after is the one we get from *moving* a logic on the core $\mathbf{C}$ of the channel to the sum of components $\mathbf{A}_1 + \mathbf{A}_2$: The IF theory will be induced at the core of the channel; this is crucial. The distributed IF logic is the *inverse image* of the IF logic at the core.

Given an infomorphism $f : \mathbf{A} \rightleftarrows \mathbf{B}$ and an IF logic $\mathfrak{L}$ on $\mathbf{B}$, the *inverse image* $f^{-1}[\mathfrak{L}]$ of $\mathfrak{L}$ under f is the IF logic on $\mathbf{A}$, whose theory is such that $\Gamma \vdash \Delta$ is a constraint of $th(f^{-1}[\mathfrak{L}])$ iff $f^{\wedge}[\Gamma] \vdash f^{\wedge}[\Delta]$ is a constraint of $th(\mathfrak{L})$, and whose normal tokens are $N_{f^{-1}[\mathfrak{L}]} = \{a \in tok(\mathbf{A}) \mid a = f^{\vee}(b) \text{ for some } b \in N_{\mathfrak{L}}\}$. If $f^{\vee}$ is surjective on tokens and $\mathfrak{L}$ is sound, then $f^{-1}[\mathfrak{L}]$ is sound.

The type and tokens system at the core and the IF classification of tokens to types will determine the IF logic at this core. We usually take the *natural IF logic* as the IF logic of the core, which is the IF logic $Log(\mathbf{C})$ generated from an IF classification $\mathbf{C}$, and has as classification $\mathbf{C}$, as regular theory the theory whose constraints are the sequents satisfied by all tokens, and whose tokens are all

normal. This seems natural, and is also what happens in the various interoperability scenarios we have been investigating.

Given an IF channel $\mathcal{C} = \{f_{1,2} : \mathbf{A}_{1,2} \rightleftarrows \mathbf{C}\}$ and an IF logic $\mathfrak{L}$ on its core $\mathbf{C}$, the *distributed IF logic* $DLog_{\mathcal{C}}(\mathfrak{L})$ is the inverse image of $\mathfrak{L}$ under the sum infomorphisms $f_1 + f_2 : \mathbf{A}_1 + \mathbf{A}_2 \rightleftarrows \mathbf{C}$. This sum is defined as follows: $\mathbf{A}_1 + \mathbf{A}_2$ has as set of tokens the Cartesian product of $tok(\mathbf{A}_1)$ and $tok(\mathbf{A}_2)$ and as set of types the disjoint union of $typ(\mathbf{A}_1)$ and $typ(\mathbf{A}_2)$, such that for $\alpha \in typ(\mathbf{A}_1)$ and $\beta \in typ(\mathbf{A}_2)$, $\langle a, b\rangle \models_{\mathbf{A}_1+\mathbf{A}_2} \alpha$ iff $a \models_{\mathbf{A}_1} \alpha$, and $\langle a, b\rangle \models_{\mathbf{A}_1+\mathbf{A}_2} \beta$ iff $b \models_{\mathbf{A}_2} \beta$. Given two infomorphisms $f_{1,2} : \mathbf{A}_{1,2} \rightleftarrows \mathbf{C}$, the sum $f_1 + f_2 : \mathbf{A}_1 + \mathbf{A}_2 \rightleftarrows \mathbf{C}$ is defined by $(f_1 + f_2)\hat{}(\alpha) = f_i(\alpha)$ if $\alpha \in \mathbf{A}_i$ and $(f_1 + f_2)\check{}(c) = \langle f\check{}_1(c), f\check{}_2(c)\rangle$, for $c \in tok(\mathbf{C})$.

It is interesting to note that since the distributed IF logic is an inverse image, soundness is not guaranteed [2], which means that the semantic interoperability is not reliable in general. Even if $\alpha \nvdash \beta$ in the IF logic, there might be tokens (instances, situations, models, possible worlds) of the respective components for which this is not the case. Reliable information flow is only achieved for tokens that are connected through the core. The way in which infomorphisms from components to the core are defined in an interoperability scenario is crucial. If these infomorphisms are surjective on tokens, then the distributed IF logic will preserve the soundness of the IF logic of the core. Proving the token-surjectiveness is hence a necessary task in order to guarantee reliable semantic interoperability.

In this sense, in Stumme and Maedche's FCA-Merge scenario [23] reliable semantic interoperability is achieved, because tokens are shared among communities, and hence all infomorphisms have the identity as its token-level function, which is obviously surjective. But this is not the case in Ciocoiu and Nau's treatment of knowledge source translation [5], where reliable semantic interoperability is only achieved when sticking to first-order models of the common ontology, which play the role of tokens of the core of an IF channel, that connect the models of the various knowledge sources.

Four Steps Towards Semantic Interoperability

To summarise, in order to achieve the semantic interoperability we desire, for each scenario we will need to go through the following four steps:

1. We define the various contexts of each community by means of a distributed IF system of IF classifications.
2. We define an IF channel—its core and infomorphisms—connecting the IF classifications of the various communities.
3. We define an IF logic on the core IF classification of the IF channel that represents the information flow between communities.
4. We distribute the IF logic to the sum of community IF classifications to obtain the IF theory that describes the desired semantic interoperability.

These steps illustrate a theoretical framework and need not to correspond to actual engineering steps; but, since any effort to automatise semantic interoperability will need to be based to some extend on a formal theory of semantic interoperability, we claim that a sensible implementation of semantic interoperability can be achieved following this framework. In the next section we describe how this information-centred approach has been applied to various realistic interoperability scenarios.

4 Explorations and Applications

A significant effort to develop an information-centred framework around the issues of organising and relating ontologies is Kent's Information Flow Framework (IFF) [11, 13]. IFF uses channel theory in that it exploits the central distinction between types and tokens, in order to formally describe the stability and dynamism of conceptual knowledge organisation. Kent also describes a theoretical two-step process that determines the *core ontology of community connections* capturing the organisation of conceptual knowledge across communities. The process starts from the assumption that the *common generic ontology* is specified as an IF theory and that the several *participating community ontologies* extend the *common generic ontology* according to theory interpretations, and consists of the following steps: A *lifting step* from IF theories to IF logics that incorporates instances into the picture (proper instances for the community ontologies, and so called *formal instances* for the generic ontology); a *fusion step* where the IF logics of community ontologies are linked through a *core ontology of community connections*, which depends on how instances are linked through the concepts of the common generic ontology. IFF is currently further developed by the IEEE Standard Upper Ontology working group as a meta-level foundation for the development of upper ontologies [12].

Very close in spirit and in the mathematical foundations of IFF, Schorlemmer studied the intrinsic duality of channel-theoretic constructions, and gave a precise formalisation to the notions of *knowledge sharing scenario* and *knowledge sharing system* [19]. He used the categorical constructions of Chu spaces [18] in order to precisely pin down some of the reasons why ontologies turn out to be insufficient in certain scenarios where a common verified ontology is not enough for knowledge sharing [6]. His central argument is that formal analysis of knowledge sharing and ontology mapping has to take a duality between syntactic types (concept names, logical sentences, logical sequents) and particular situations (instances, models, semantics of inference rules) into account.

With respect to the particular task of mapping ontologies Kalfoglou and Schorlemmer present a step-wise method, IF-Map, which (semi-)automatically maps ontologies based on representing ontologies as IF classifications and automatically generating infomorphisms between IF classifications [10]. They demonstrated their approach by using the IF-Map method to map ontologies in the domain of computer science departments from five UK universities. The underlying philosophy of IF-Map follows the assumptions made in Section 3 where we argued that the way communities classify their instances with respect to local types reveals the semantics which could be used to guide the mapping process. Their method is also complemented by harvesting mechanisms for acquiring ontologies, translators for processing different ontology representation formalisms and APIs for web-enabled access of the generated mappings; these are in the form of infomorphisms as we introduced them in Section 3.

Finally, Schorlemmer and Kalfoglou used the framework described in Section 3 in an e-government alignment scenario [20]. In particular, they used the four steps described earlier to align UK and US governmental departments by using their units as types and their respective set of responsibilities as tokens which were classified against those types. This test bed was used to demonstrate the feasibility of the framework in a versatile and emerging paradigm, that of e-governments, where semantic interoperability is a prerequisite.

5 Conclusion

In order to achieve semantic interoperability and integration in an automated fashion we will need of a formal theory of semantic interoperability that suitably captures the idea of a "semantically integrated community" [25]. So far, efforts to formalise and automatise the issues arising with semantic interoperability have been focused on particular understandings of semantics, mainly based on first-order model theory. But it would be desirable to be provided with a theoretical framework that accommodates various different understandings of semantics depending on the semantic interoperability scenario addressed.

In this paper we have explored the suitability of Barwise and Seligman's channel theory to establish such a framework, by focusing our attention on the issues of information flow. Consequently, we have proposed a four-step methodology to enable semantic interoperability and have discussed various applications, theoretical and practical, of this methodology.

Acknowledgements. This work is supported under the Advanced Knowledge Technologies (AKT) Interdisciplinary Research Collaboration (IRC), which is sponsored by the UK Engineering and Physical Sciences Research Council under grant number GR/N15764/01. The AKT IRC comprises the Universities of Aberdeen, Edinburgh, Sheffield, Southampton and the Open University.

The first author is also supported by a 'Ramón y Cajal' Fellowship from the Spanish Ministry of Science and Technology.

References

[1] J. Barwise and J. Perry. *Situations and Attitudes*. MIT Press, 1983.

[2] J. Barwise and J. Seligman. *Information Flow: The Logic of Distributed Systems*. Cambridge University Press, 1997.

[3] T. Bench-Capon and G. Malcolm. Formalising ontologies and their relations. In *Database and Expert Systems Applications*, volume 1677 of *Lecture Notes in Computer Science*, pages 250–259, 1999.

[4] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[5] M. Ciocoiu and D. Nau. Ontology-based semantics. In *Principles of Knowledge Representation and Reasoning: Proceedings of the 7th International Conference*, pages 539–548. Morgan Kaufmann, 2000.

[6] F. Corrêa da Silva, W. Vasconcelos, D. Robertson, V. Brilhante, A. de Melo, M. Finger, and J. Agustí. On the insufficiency of ontologies: Problems in knowledge sharing and alternative solutions. *Knowledge-Based Systems*, 15(3):147–167, 2002.

[7] K. Devlin. *Logic and Information*. Cambridge University Press, 1991.

[8] M. Genesereth and N. Nilsson. *Logical Foundations of Artificial Intelligence*. Morgan Maufmann, 1987.

[9] A. Halevy. Answering queries using views. *The VLDB Journal*, 10:270–294, 2001.

[10] Y. Kalfoglou and M. Schorlemmer. Information-flow-based ontology mapping. In *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, volume 2519 of *Lecture Notes in Computer Science*, pages 1132–1151. Springer, 2002.

[11] R. Kent. The information flow foundation for conceptual knowledge organization. In *6th International Conference of the International Society for Knowledge Organization*, Toronto, Canada, July 2000.

[12] R. Kent. A KIF formalization of the IFF category theory ontology. In *IJCAI'01 Workshop of the IEEE Standard Upper Ontology*, Seattle, Washington, USA, 2001.

[13] R. Kent. The IFF foundation for ontological knowledge organization. In *Knowledge Organization and Classification in International Information Retrieval*, Cataloging and Classification Quarterly. The Haworth Press Inc., 2003.

[14] J. Madhavan, P. Bernstein, P. Domingos, and A. Halevy. Representing and reasoning about mappings between domain models. In *18th National Conferenc on Artificial Intelligence (AAAI'02)*, Edmonton, Canada, 2002.

[15] C. Menzel. Ontology theory. In *ECAI'02 Workshop on Ontologies and Semantic Interoperability*, Lyon, France, 2002.

[16] P. Mitra and G. Wiederhold. An algebra for the composition of ontologies. In *ECAI'02 Workshop on Knowledge Transformation for the Semantic Web*, Lyon, France, 2002.

[17] J. Pollock. The web services scandal: How data semantics have been overlooked in integration solutions. *eAI Journal*, pages 20–23, Aug. 2002.

[18] V. Pratt. The Stone gamut: A coordination of mathematics. In *10th Annual Symposium on Logic in Computer Science*, pages 444–454. IEEE Computer Society Press, 1995.

[19] M. Schorlemmer. Duality in knowledge sharing. In *7th International Symposium on Artificial Intelligence and Mathematics*, Ft. Lauderdale, Florida, USA, 2002.

[20] M. Schorlemmer and Y. Kalfoglou. Using information-flow theory to enable semantic interoperability. In *6è Congrés Català en Intel·ligència Artificial*, Palma de Mallorca, Spain, Oct. 2003. Also available as Informatics Report EDI-INF-RR-0161, The University of Edinburgh.

[21] A. Sheth and J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, 1990.

[22] J. Sowa. *Knowledge Representation and Reasoning: Logical, Philosophical, and Computational Foundations*. Brooks/Cole, 2000.

[23] G. Stumme and A. Maedche. FCA-Merge: Bottom-up merging of ontologies. In *17th International Joint Conference on Artificial Intelligence (IJCAI'01)*, Seattle, Washington, USA, Aug. 2001.

[24] R. Turner. Properties, propositions and semantic theory. In M. Rosner and R. Johnson, editors, *Computational linguistics and formal semantics*, chapter 5. Cambridge University Press, 1992.

[25] M. Uschold and M. Grüninger. Creating semantically integrated communities on the world wide web. In *WWW'02 Semantic Web Workshop*, 2002.

Developing Consensus Ontologies for the Semantic Web

Larry M. Stephens, Aurovinda K. Gangam, and Michael N. Huhns
Department of Computer Science and Engineering
University of South Carolina, Columbia, SC, 29208, USA

Abstract

This paper describes a methodology for associating, organizing, and merging large numbers of independently developed information sources. The hypothesis is that a multiplicity of ontology fragments, representing the semantics of the independent sources, can be related to each other automatically *without* the use of a global ontology. The methodology has been tested by merging small, independently developed ontologies for the domains of *Humans*, *Buildings*, and *Sports*. The methodology, which reinforces common parts of the component ontologies and deemphasizes unique parts, produces a *consensus* ontology.

1 Introduction

A search for information will typically uncover a large number of independently developed information sources—some relevant and some irrelevant. A common theme for refining searches is the creation, use, and manipulation of ontologies for describing both requirements and sources [2, 4, 6, 7, 9, 13, 16]. Unfortunately, ontologies are not a panacea unless everyone adheres to the same one, and no one has yet constructed an ontology that is comprehensive enough—even given ongoing attempts to create one such as [1, 10] and the Cyc Project [11], underway since 1984. Moreover, even if one did exist, it probably would not be adhered to, considering the dynamic and eclectic nature of the Web and other information sources.

This paper describes a methodology for merging and, therefore, relating small, independently developed ontologies automatically *without* the use of a global ontology. It is assumed that the sites have been annotated with ontologolical information [14]—a representation consistent with several visions for the Semantic Web [3, 8]. The domains of the sites must be similar—else there would be no interesting relationships among them—but they will undoubtedly have dissimilar ontologies, because they will have been annotated independently.

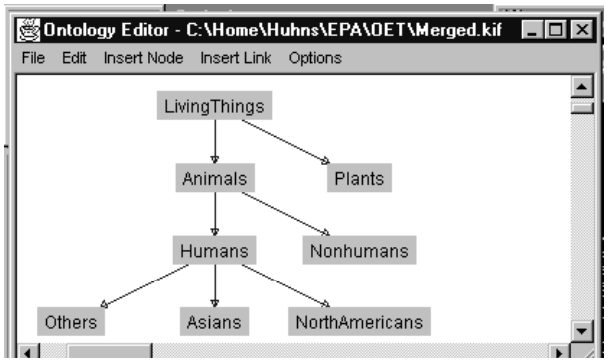


Figure 1: A typical small ontology used to characterize an information source about people (all links denote subclasses)

2 Experimental Methodology

To assess the methodology, we asked each student in a group of 54 computer science graduate students to construct a small ontology for the domain of *Humans-People-Persons.* A second group of 28 students constructed small ontologies for the *Buildings* domain, and a third group of 25 students developed ontologies for the *Sports* domain. The ontologies were written in OWL [5] and contained at least 8 classes organized with at least 4 levels of subclasses; a sample ontology is shown in Figure 1. In this and all other figures the directed link is from *superclass* to *subclass.*

We merge the files in each of the three domains using the syntactic and semantic information available in the component ontologies. The syntactic information is derived from the names of the nodes, for which we employ various string-matching techniques including detection of plural endings. The semantic information includes the meaning of the subclass link in the ontologies, prefixes that indicate antonyms, and evolving sets of synonyms for matching nodes. The synsets, which are used to track the progress of

merging and to monitor correctness, are seeded from WordNet [12]. The details of the node-merging algorithm are given in the Appendix.

Our system merges the component files one-at-a-time into a resultant merged file. For each node in the resultant file, we maintain a *reinforcement* value, which indicates how many times the node is matched as ontologies are merged. We also maintain reinforcement values for class-subclass links. The original work reported in [15] was dependent on the ontology sequencing; the work reported herein uses an algorithm that is commutative with respect to the ordering of component ontologies.

The enhanced algorithm also identifies and removes circularities in the merged ontologies, enforces disjoint-class definitions that are specified in the component ontologies, and identifies noun "classifiers," such as *Apartment* in *Apartment-Building* to determine subclass relationships. For noun-classifier identification, we use the heuristic of matching the shorter node name (*Building*—the candidate superclass) with the ending of the longer string (*ApartmentBuilding*—the candidate subclass).

The identification of noun-noun pairs is not straightforward if there is no space, hyphen, or case change between the nouns. The string "OfficeBuilding" is not recognized by WordNet, which correctly identifies both "office building" and "office-building." Ontology builders need a set of conventions for entering noun-classifier knowledge. We prefer the use of "camel-case," which allows words to be easily extracted. Without such conventions, extraction becomes difficult. From "warmbloodedanimal," one might extract "war," "warm," "arm," "blood," "loo," "ode," "animal," "ma," and "mal" to name a few.

3 Results

In the *Humans-People-Persons* domain, the component ontologies described 864 classes, while the merged ontology shown in Figure 2 contained 389 classes in a single graph with a root node of the OWL concept `owl:Thing`. All of the concepts were related, i.e., there was some relationship (path) between any pair of the merged concepts.

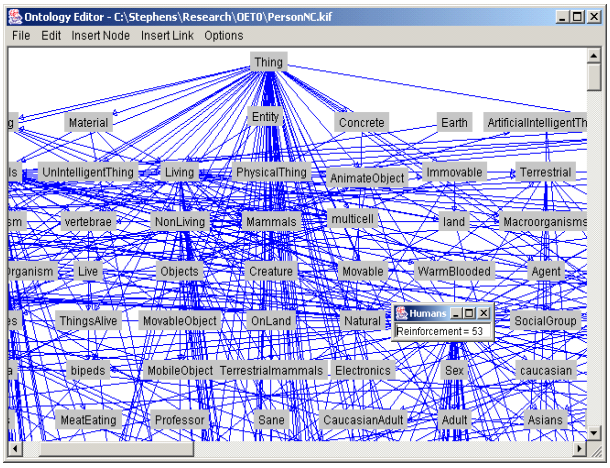


Figure 2: A portion of the ontology formed by merging 54 independently constructed ontologies for the domain Humans/People/Persons. The entire ontology has 389 concepts related by 696 subclass links.

Next, we constructed a *consensus ontology* by eliminating weakly reinforced nodes and links. In filtering the merged file, we sorted the links by their reinforcement values and found that, for the most part, the strongly reinforced nodes were associated with strongly reinforced links. This finding, while not surprising, makes constructing a consensus ontology more efficient.

The consensus ontology for the domain of *Humans* consists of 20 classes related by 25 subclass links (see Figure 3). The class *Humans* and its matching classes appeared 53 times (one of the 54 students used the term *Sapiens(Man),* which failed to match the other nodes). The subclass link from Mammals (and its matches) to Humans (and its matches) appeared 10 times. In this figure, all nodes are reinforced at least 5 times and all links, except as noted, reinforced at least 3 times. The weakly reinforced link *Female–Women* could be omitted but illustrates the transitive closure considerations, which are discussed next.

We considered removing from our merged ontologies all transitive closure class-subclass links, and reinforcing the remaining links. For example, if A has subclass B, and B has subclass C, then it appears needless to assert explicitly that A has subclass C. However, this approach can introduce results that clearly violate a consensus view. In Figure 3, *Humans* has subclass *Fe-*

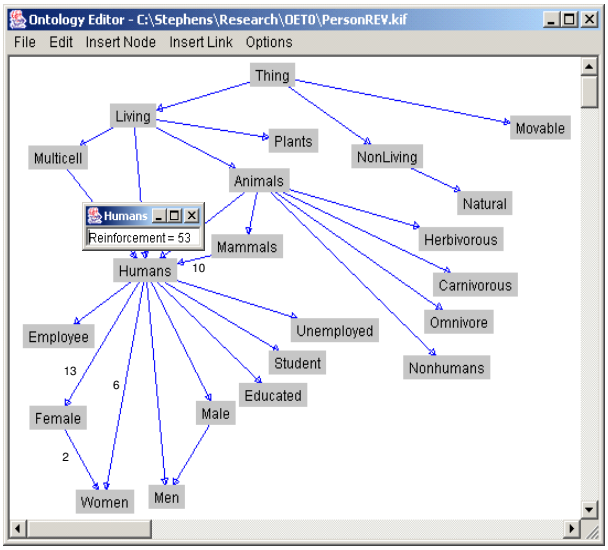


Figure 3: The consensus ontology for the *Humans* domain formed by merging concepts with common subclasses and superclasses from 54 component ontologies. The resultant ontology contains 20 concepts related by 25 subclass links.

male with reinforcement 13, *Female* has subclass *Women* with reinforcement 2, and the direct subclass link from *Humans* to *Women* has reinforcement 6. Removing the direct link and reinforcing the remaining links (as in Figure 4) would give the *Female–Women* link a reinforcement value of 8—much stronger than the consensus view indicates. Our conclusion was to abandon this procedure and leave link reinforcement values unchanged.

Figures 5 and 6 show the results for the domains of *Buildings* and *Sports*, which are based on 28 and 25 component ontologies, respectively. For these two domains, the reinforcement threshold for concepts and links is 3.

4 Discussion, Limitations, and Conclusions

A consensus ontology is perhaps the most useful organization for information retrieval by humans, because it represents the way most people view the world and its information. For example, if most people wrongly believe that crocodiles are a kind of mammal, then most people would find it easier to locate information about crocodiles if it were placed in a mammals grouping, rather

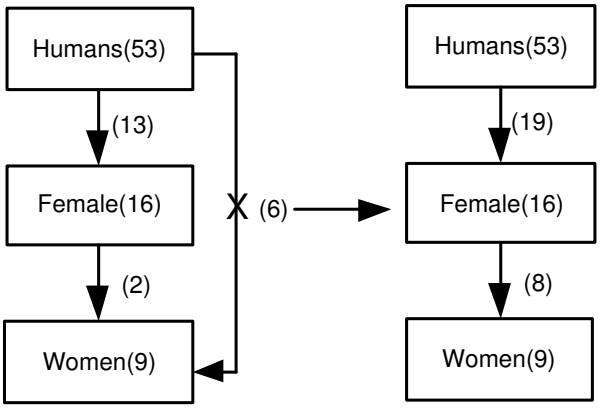


Figure 4: The consensus is that the concept *Women* is more strongly linked to *Humans* than *Female*. Removing the direct link from *Humans* to *Women* and reinforcing remaining links violates that consensus. Node and link reinforcements are shown in parentheses.

than where it factually belonged.

Our results could be useful in the following scenario: suppose a user, interested in a comparison of the conductivity of aluminum versus copper wire, initiates a simple search on the term *conductor*. A recent Google^{TM} search for *conductor* returned a ranked list of 1,980,000 Web pages, some of which concern orchestra and railroad conductors. Our methodology could be used to construct a merged ontology from the small ontologies associated with each of the first 100 or so pages. The merged ontology, centered on the term *conductor* and revealing the three mostly disjoint sub-ontologies for its three word senses, would be presented to the user, as shown in Figure 7. Based on this, the user could select a node to retrieve a page, or iterate by selecting a node from which to initiate a refined search.

Our experiments and analysis are preliminary and ongoing. However, the results so far support the hypothesis that a multiplicity of ontology fragments can be related automatically without the use of a global ontology. We used the following simplifications in our work:

- We did not make use of the properties of the classes, as would be the case for a complete implementation of subsumption.
- Our string-matching algorithm did not use a thorough morphological analysis to separate

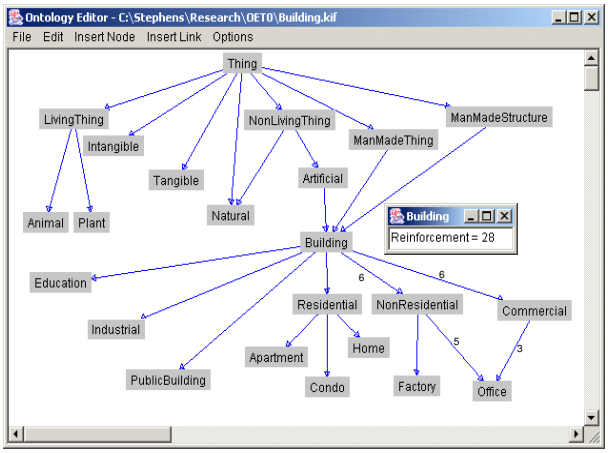


Figure 5: The consensus ontology for the *Building* domain contains 23 concepts and 26 links. *Office* is considered both *NonResidential* and *Commercial*. The concepts *Plant* (a subclass of *LivingThing*) and *Factory* (a subclass of NonLivingThing) appear in different branches of the ontology. The merged ontology is derived from 28 component ontologies.

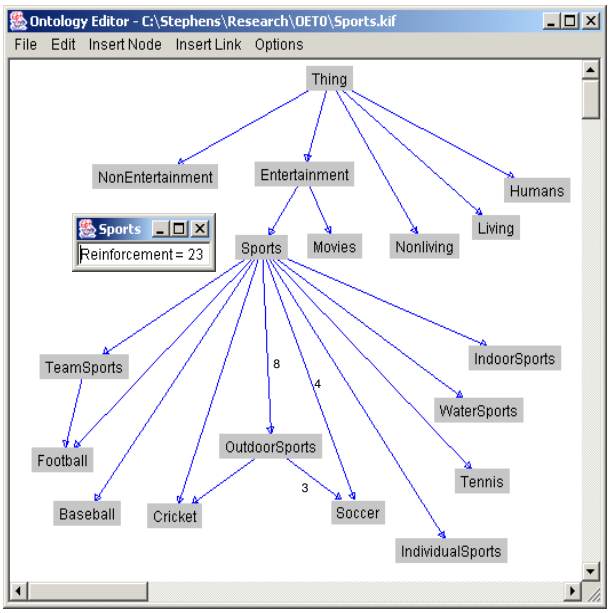


Figure 6: The consensus ontology for the *Sports* domain has 18 concepts and 20 links. *Soccer* is classified slightly more strongly as a subclass of *Sports* rather than of *OutdoorSports*.

the root word from its prefixes and suffixes. We do, however, handle singular and plural noun forms in most cases, and discriminate between obvious antonym pairs.

- Noun classifiers were detected by a string-matching heuristic. Breaks in compound nouns need to be identified in a more principled way, such has a blank space, hyphen, or case change. Unfortunately our data sets did not adhere to a uniform convention for compound noun representation.

- We used only subclass-superclass information, and have not yet made use of other important relationships, notably *partOf*.

The technology developed by our research would yield an organization of the received information, with the semantics of each document reconciled. This is a key enabling technology for knowledge-management systems. The technique could be applied off-line by search engines, thereby providing ontologies that do not exist today for refining queries.

Our premise is that it is easier to develop small ontologies, whether or not a global one is available, and that these can be automatically and *ex post facto* related. We are determining the efficacy of local annotation for Web sources, as well as the ability to perform reconciliation qualified by measures of semantic distance. The results of our effort will be (1) software components for semantic reconciliation, and (2) a scientific understanding of automated semantic reconciliation among disparate information sources.

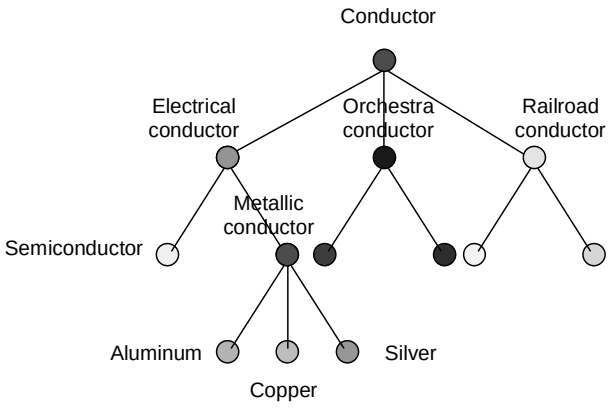


Figure 7: A merged ontology refines the domain concepts needed by users to satisfy their requests.

References

[1] E. Agirre, O. Ansa, E. Hovy, and D. Martnez, "Enriching very large ontologies using the WWW," in *Proceedings of the Ontology Learning Workshop*, ECAI, Berlin, Germany, July 2000.

[2] J. L. Ambite, Y. Arens, E. Hovy, A. Philpot, L. Gravano, V. Hatzivassilogluo, and J. Klavens, "Simplifying Data Access: The Energy Data Collection Project," *IEEE Computer*, 34(2), 47–54, 2001.

[3] T. Berners-Lee, J. Hendler, and O. Lassila, "The Semantic Web," *Scientific American*, 284(5), 34–43, 2001.

[4] B. Chandrasekaran et al., "What are ontologies, and why do we need them?," *IEEE Intelligent Systems*, 14(1), 20–26, 1999.

[5] M. Dean and G. Schreiber (eds.), "OWL Web Ontology Language 1.0 Reference," March 31, 2003, http://www.w3.org/TR/owl-ref/.

[6] S. Decker, M. Erdmann, D. Fensel, and R. Studer, "Ontobroker: Ontology Based Access to Distributed and Semi-Structured Information," in R. Meersman et al. (eds.), *Semantic Issues in Multimedia Systems, Proceedings of DS-8*, Kluwer Academic Publishers, Boston, 351–369, 1999.

[7] A. Farquhar, R. Fikes, and J. Rice, "Tools for Assembling Modular Ontologies in Ontolingua," in *Proceedings of AAAI-97*, AAAI Press, Menlo Park, CA, 436–441, 1997.

[8] J. Heflin and J. Hendler, "Dynamic Ontologies on the Web," in *Proc. 17th National Conference on AI (AAAI-2000)*, AAAI Press, Menlo Park, CA, 443–449, July 2000.

[9] V. Kashyap and A. Sheth, *Information Brokering across Heterogeneous Digital Data: A Metadata-based Approach*, Kluwer Academic Publishers, Boston, 2000.

[10] K. Knight and S. Luk, "Building a Large-Scale Knowledge Base for Machine Translation," *Proc. of the National Conference on Artificial Intelligence (AAAI)*, Seattle, WA, 773–778, 1994.

[11] D. B. Lenat and R. V. Guha, *Building Large Knowledge-Based Systems*, Addison-Wesley, Reading, MA, 1990, http://www.cyc.com.

[12] G. A. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM*, 1995 38(11), 39–41, 1995.

[13] M. Nodine, W. Bohrer, and A. Hee Hiong Ngu, "Semantic Brokering over Dynamic Heterogeneous Data Sources in InfoSleuth," *15th International Conference on Data Engineering*, Sydney, Australia, March 1999.

[14] J. M. Pierre, "Practical Issues for Automated Categorization of Web Sites," *Electronic Proc. ECDL 2000 Workshop on the Semantic Web*, Lisbon, Portugal, 2000, http://www.ics.forth.gr/proj/isst/SemWeb/program.html

[15] L. M. Stephens and M. N. Huhns, "Consensus Ontologies: Reconciling the Semantics of Web Pages and Agents," *IEEE Internet Computing*, 5(5), 92–95, 2001.

[16] C. Welty, "The Ontological Nature of Subject Taxonomies," in N. Guarino (ed.), *Formal Ontology in Information Systems*, IOS Press, Amsterdam 317–327, 1998.

Appendix: Node-Name Matching Algorithm

Our principle technique for merging two ontologies relies on simple string and substring matching. The name of a node from one ontology is systematically compared to each of the nodes from another ontology using the following prioritized rules:

- If an exact match is found, then the comparisons cease and a value of 1.0 is assigned as a match.
- If the node names are antonyms of each other, then the merging attempt is aborted. We detect antonyms formed by prefixes such

as *anti*, *dis*, *im*, *in*, *non*, and *un*. In general, antonym checking prevents some mergers and produces a correspondingly larger number of total classes compared to uninformed string matching. Antonyms are a convenient way to subdivide concepts or domains into subconcepts and opposites, and were widely used in the student-produced ontologies. For example, it is typical that *People* might be divided into *Students* and *NonStudents*, or *Citizens* and *NonCitizens*.

- If the names are not identical, then we check for plural pairs that follow the traditional rules of grammar such as building–buildings, calf–calves, knife–knives, and thesis–theses. The match value is set to 1.0 as if the node names were identical.

- If the shorter string is wholly contained at the *end* of the longer string, then the nodes are not merged but the node with the shorter string name is asserted to be a super class of the node having the longer name. For example, the string "Animal" matches the end of the string "WildAnimal," so "Animal" is assumed to a superclass of "WildAnimal."

- Otherwise, the match value is based on the extent to which the *leading* substring of the shorter name matches the *leading* substring of the longer name. For example, the first five characters of "Animal" and "Animate" are identical, and a match value of $5/7 = 0.71$ is assigned.

Evaluating Ontological Analysis

Christopher A. Welty, Ruchi Mahindru, and Jennifer Chu-Carroll
IBM T.J. Watson Research Center
{welty,rkalra,jencc}@watson.ibm.com

Introduction

Ontologies have often been proposed as a solution to the semantic integration problem, relying on the premise that a clear, high-quality ontology can act as an *interlingua* in which mappings between systems can unambiguously be expressed [Smith and Welty, 2001]. While this approach has not been realized in practice, one recent development in ontology research has been the specification of a formal methodology for ontological analysis, OntoClean [Guarino and Welty, 2002], that addresses the problem of defining just what "high quality" is for ontologies. Following this definition and approach, a high quality foundational ontology, Dolce, is being developed [Gangemi, et al, 2002].

While OntoClean appears to be a widely accepted analysis tool in the scientific community, there is still only a little evidence that it can have impact on semantic integration*. In fact, there appears to be a significant obstacle in understanding the methodology, and even without this "learning curve", significant manual effort must be expended to employ the methodology to develop actual "clean" ontologies. Finally, there has been no clear argument that such an expenditure will pay for itself in the long run. Indeed, "Why does it matter?" has been the most frequent criticism of the OntoClean approach.

We report here some preliminary results from a series of experiments using a knowledge-based search system to test the impact of *improving the quality of ontologies* on system performance. The use of search as a test system provides a well understood framework for empirical evaluation, and gives an excellent opportunity to address the, "Why does it matter?" question.

* OntologyWorks, a small company providing database integration services, has a proprietary analysis tool based on OntoClean (www.ontogyworks.com).

Background

The field of ontology has been sorely lacking in formal empirical analysis in general, however there have been numerous evaluations of the impact of structured knowledge (loosely construed as ontologies) on IR and search systems in general.

Most closely related to this work is the work of Clark, et al, at Boeing [2000], in which a search system was enhanced by employing an ontology-like artifact (a thesaurus of terms with more meaningful structure than a flat list of keywords). This work showed that precision and recall performance of a retrieval system can be significantly increased by adding this kind of information. It is important to note that while Clarke, et al, did discuss a process for improving the quality of the ontology; they did not formally evaluate the impact of the *improvement*. Furthermore, the AskJeeves corporation Enterprise solutions (www.jeevessolutions.com) has based their business on providing domain-specific knowledge-enhanced search, and have been turning a profit since 1Q 2002 [Ulicny, 2003].

Similar evaluations of the impact of ontologies on search-based systems have been done in the question-answering community. Moldovan and Mihalcea [2000] use a significantly enhanced version of WordNet to drastically improve question answering performance, and other groups including Woods, et al [2000], Hovy et al [2001], and Prager, et al [2001], have reported similar results. Again, as with the Boeing work, these groups report positively on the impact of adding an ontology to a search system, but make no attempt to determine whether good quality ontology would improve performance more. In fact, within the IR and QA communities, WordNet is the most common ontology-like artifact to employ, and previous work has shown that WordNet *viewed as an ontology* is not particularly of high quality [Oltramari, et al, 2002].

System Overview

The RISQUE system is an evolution of the system reported in [Chu-Carroll, et al, 2002]. This system provides a natural-language front end to a conventional search engine, but uses clues in the natural language question, a knowledge-base of industry terms, and knowledge of the web site structure (see below) to construct an *advanced search query* using the full expressiveness of the search engine. This search is limited to a corporate web site, in this case our knowledge is of the ibm.com buy and support pages for the ThinkPad™ and NetVista™ product lines.

The main components of Risque are a parser, the terminology, rules for question types, a hub page finder, a query relaxer, and the search engine.

The parser is a slot-grammar parser [McCord, 1980] that must be seeded with multi-word industry specific terms, so that e.g. "disk drive" will be parsed as a compound noun, rather than a head noun with a pre-modifier. These terms come from the knowledge-base. From the grammatical structure of the question, we extract the primary verb phrase and the noun phrases. The verb phrase information is the main evidence used to fire rules for recognizing question types, which themselves depend on the web site structure. The ibm.com web site, like many enterprise sites, is divided into a section for support and a section for sales. This gives Risque its two most basic question types, "buy" and "support".

The hub-page finder is a system of declarative rules that takes the noun phrases from the question and determines whether they correspond to products listed in the terminology, and if so finds the most appropriate "hub page" or "comparison page" for that product. For example, many questions about IBM Thinkpads can be answered with information on the "ThinkPad home page" on the IBM site. Directing users to these key pages is often the quickest path to an answer. Hub and comparison pages are described in the next section. The rules are broken into two parts, one set is derived directly from the knowledge base and includes *moreImportantThan* relationships and the taxonomy; the second includes rules expressing the relationships between linguistically-derived information and the hub pages. For example, if the question contains a superlative, as in "What is the fastest Thinkpad?", the rules indicate that the Thinkpad comparison page should be returned.

The system generates complex queries using knowledge of web site structure. The query may include URL restrictions, such as "only consider pages with 'support' in the URL for support questions", or "exclude pages with research.ibm.com in the URL for buy questions". The query will also make use of boolean connectives, disjunction to support synonym expansion, and conjunction of noun phrase terms. If this query does not return enough hits, the query relaxer will relax the query according to a number of heuristics, such as dropping the least important noun phrase. Knowledge of which terms are more important than others, based on manual analysis of the web site, is included in the knowledge-base.

The Risque system was tested and trained with a set of questions made up by team members. Later, it was evaluated with a set of questions made up by a domain expert from outside the group. The latter can be considered a fairly important element for evaluation, as it led us to the notion of an *expansion*, discussed below.

Role of Ontology

The central terminology of Risque is an ontology-like knowledge base of industry terms arranged in taxonomy according to specificity. In addition to the taxonomy, the knowledge base includes important information used by the system:

Hub page: Most terms at the top level have a corresponding "hub page" – a page that gives a general description of the things in that category. For example, there is a hub page for IBM Thinkpads, and also a hub page for IBM T-Series Thinkpads. The ibm.com website, along with most e-commerce websites, are designed to pack a lot of information in these particular pages, with links to as much information as the designers can imagine might be relevant to someone seeking support or seeking to purchase. The taxonomy is used to associate terms with *the most specific* hub page that is relevant. For example, if we know that a "ThinkPad A21" is a "ThinkPad A-Series Model", and the former has no hub page, then we infer it to be the latter's hub page. Furthermore, the hub page for all Thinkpads, would not be.

Comparison Page: Many e-commerce web sites including IBM's provide the ability to compare two or more similar products. Our knowledge-base stores information on how to find or

generate comparison pages for products. These pages will be displayed for questions like, "What is the fastest A-Series ThinkPad?" Similar to hub pages, the taxonomy is used to associate terms with the *most specific common* comparison page.

Synonyms: Synonyms account for simple variations on spelling, acronyms, abbreviations, etc., as well as traditional synonyms. This information is used to find the term being referenced in a question, as well as in query expansion. The use of synonyms in query expansion made the notion of an *expansion* (see below) more important.

MoreImportantThan: e-Commerce websites have an organization that is important to capture in interpreting questions. For example, IBM's web site is organized such that add-on accessory pages list which models they are compatible with, but computer pages do not list which accessories are compatible with them. This knowledge is explicit and intentional for the website maintainers, but is not necessarily obvious to a customer browsing the site for the first time. Thus, when an accessory and a computer are mentioned in the same question, such as, "What CD drive goes with my ThinkPad T23?" we consider the CD drive to be the more important term in the question. The more important term in the question will have its hub page returned in a higher position, and the less important term may, in some circumstances, not have its hub page appear at all. In addition, the less important term will be dropped first during query relaxation. The *MoreImportantThan* relation is considered to be transitive, and is also inherited down the taxonomy. Thus we only represent in the knowledge-base that accessories are *moreImportantThan* computers, and from this we infer that CD drive is *moreImportantThan* ThinkPad T23.

Expansions: An interesting situation that we had to account for in dealing with questions generated by a domain expert was that often people are confused about what industry terms mean. For example, many people think "SCSI" is a kind of disk drive, when in fact is it a type of communications bus. These types of errors do not appear in the web pages, thus making SCSI a simple synonym of "disk drive" would not be productive – synonyms are used in query expansion and therefore searches for disk drives would turn up communication bus technology pages. To solve this, the *expansion* relation between terms is treated as an asymmetric synonym. When "SCSI" appears in a query, it will be considered a synonym of disk-drive, however when disk-drive appears in a query, it will not be considered a synonym of "SCSI".

Clean-up Process

The original Risque system terminology, Quilt, was developed by domain experts with no experience with or knowledge of ontology engineering methods, and contained on the order of 3K synsets and 4.6K terms. We improved this terminology in a number of ways:

1) Developed a "backbone taxonomy" of terms
2) Analyzed terms and their position in the hierarchy.
3) Organized terms more logically
4) Ensured every term was grounded in the top level
5) Ensured terminology was logically consistent

We used three tools in performing this cleanup: The OntoClean methodology was used in analysis, and helped with #1-3; an ontology editor was used to view the taxonomy, this was critical in #2-4; a reasoner was used to ensure consistency and coverage of the *MoreImportantThan* relation.

The analysis and cleanup took on the order of one person-week, and resulted 3K synsets and 10.8K terms. This was largely due to the discovery of inconsistent, meaningless, redundant and disconnected terms in the original ontology, and a more consistent expansion of regular synonym patterns (such as T21, ThinkPad 21).

Experimental Setup

Although the main goal of the Risque system was to show an improvement over traditional web search, the particular experiment described in this paper was to isolate the process of improving the quality of the terminology using ontology-based analysis tools. The Risque system architecture treats the terminology as a pluggable module, which allowed us to isolate this particular change while holding all other aspects of the system constant. We then concentrated on how to compare a poorly structured terminology with a cleaned one.

After the cleanup was complete, we performed four evaluations as follows:

baseline: basic search over the IBM web pages using a traditional search engine

quilt: The full Risque system with the original Quilt terminology

clean: The full Risque system with the cleaned terminology

google: basic search using Google restricted to the ibm.com web pages

The evaluation was performed on 127 natural language questions about IBM products collected from a web site expert and hand-generated variations for broader domain coverage to meet a particular internal commitment. The experiments were run against the live ibm.com website, over which we had no control. As a result, we ran the experiments in parallel, with each question running through all four systems at the same time, in order to prevent changes in the web site from impacting performance of one system in isolation.

The google and baseline queries were formulated manually from a conjunction of all the words in the noun phrases from the natural language question. The answer to a question was considered correct if one of the pages in the top ten returned by the search contained an answer to the question – i.e. the answer is a single click (and some reading) away. For comparison questions, e.g. "What is the fastest desktop?", or "What is the lightest ThinkPad?" the answers were considered correct if the comparison page selected by Risque contained the relevant data for each type of computer, e.g. the processor speed of each desktop model, or the weight of each ThinkPad model.

Results and Analysis

Our results are shown in Table 1. These results are still preliminary, and we intend to also perform a recall measurement. Each experiment lists the number correct (of 127) and the percent.

	# correct	% correct	% improvement
Base	46	36%	
Quilt	77	61%	67%
Clean	92	72%	over baseline: 100% over Quilt: 19%
Google	43	33%	-6%

First of all, our results confirm the overwhelming evidence to date that ontologies can significantly improve search results. In this case we see a relative improvement of 67% over the baseline search even with a poorly constructed ontology. This result appears to come from the correct identification of industry terms for the parser (which is not dependant on ontology structure), and the association of more common industry terms with the proper hub pages. Again, as discussed above, hub pages on the ibm.com web site are designed to contain a lot of information.

Most notably, our results show a clear improvement of the search results when using the higher quality "cleaned" terminology, which *doubles the performance* of the baseline search and shows a 19% relative improvement over the original terminology. While the improved terminology contained more actual words, this expansion did not in itself account for the increase in precision. *Prima facie* most correct answers come from hub and comparison pages, so the fact that terms are more consistently connected through the taxonomy with these pages in the cleaned terminology was the major reason the cleanup improved precision. Another important factor was the proper derivation of the *moreImportantThan* relation between terms, which was incorrect in a number of cases in the original search because of missing links in the taxonomy.

The heavy reliance of our system on "hub pages" for correct answers would seem to indicate that link analysis, or a similar technique that ranks highly connected pages over less connected ones, would improve search considerably given the large number of incoming and outgoing links on these pages. If effective, such a technique would clearly be preferable over a knowledge-base, since it requires significantly less manual effort to maintain. This led us to perform an experiment using the Google™ search engine restricted to the ibm.com website. We were very surprised to find that this experiment was the worst performer of all, although the difference from baseline was not significant. Again, these results are preliminary, but we believe one important difference between the knowledge-based search and one based on link analysis is knowledge of the structure of the website, as reflected e.g. in the *moreImportantThan* relation. As discussed above, one of the things captured in the relation is the fact that information about compatibility is located on accessory pages, not the computer pages. Thus the highly-connected ThinkPad hub pages receive high scores from Google™ for questions like, "What modem goes with my ThinkPad t30?", but they do not contain an answer to the question - knowledge trumps statistics.

The main flaw in the evaluation was that the questions, though generated externally from the

Risque group, came from a domain expert and not from actual users.

Conclusion

We described a system for Knowledge-Based Natural Language Search called Risque, and focused on the knowledge-based components and their role in the system. We performed a controlled experiment to compare the precision of the search system with an unprincipled ontology to the same system with a principled ontology. Our results showed an 19% relative improvement in precision (from 61% to 72%) with no other changes in the system other than applying the OntoClean methodology to analyzing the ontology and cleaning it.

Though these results are still preliminary and undergoing more thorough analysis, we have shown evidence that improving the quality of an ontology does improve the performance of an ontology-based search. It stands to reason that any system that has a significant ontology component would benefit from improving the ontology portion.

Acknowledgements

Other contributors to the Risque system were John Prager, Yael Ravin, Christian Lenz Cesar, David Ferrucci, and Jerry Cwiklik. Our thanks to Mike Moran and Alex Holt for supporting the project.

References

Chu-Carroll, J., John Prager, Yael Ravin and Christian Cesar. 2002. A Hybrid Approach to Natural Language Web Search. *Proceedings of EMNLP-2002.*

Clark, P., John Thompson, Heather Holmback, and Lisbeth Duncan. 2000. Exploiting a Thesaurus-Based Semantic Net for Knowledge-Based Search. *Proceedings of IAAI-2000.* Pp. 988-995. AAAI Press.

Gangemi A., Guarino N., Masolo C., Oltramari, A., Schneider L. 2002. Sweetening Ontologies with DOLCE. *Proceedings of EKAW 2002*. Siguenza, Spain.

Guarino, Nicola and Chris Welty. 2002. Evaluating Ontological Decisions with OntoClean. *Communications of the ACM.* 45(2):61-65. New York: ACM Press.

Hovy, E, L. Gerber, U. Hermjakob, C.-Y. Lin, and D. Ravichandran. 2001. Toward Semantics-Based Answer Pinpointing. In *Proceedings of the DARPA Human Language Technology Conference (HLT).* San Diego, CA.

McCord, Michael C. 1980. Slot grammars. *American journal of Computational Linguistics*, **6**(1):255--286, January 1980.

Moldovan, D. and Rada Mihalcea. 2000. Using WordNet and Lexical Operators to Improve Internet Searches. *IEEE Internet Computing*. **4**(1): 34-43.

Oltramari, A., Aldo Gangemi, Nicola Guarino, and Claudio Masolo. Restructuring WordNet's Top-Level: The OntoClean approach. *Proceedings of LREC2002 (OntoLex workshop)*. Las Palmas, Spain

Prager, J., Jennifer Chu-Carroll, Krzysztof Czuba. Use of WordNet Hypernyms for Answering What-Is Questions. *Proceedings of TREC 2001.*

Smith, B. and C. Welty. 2001 Ontology: Towards a new synthesis. In *Formal Ontology in Information Systems.* Ongunquit:ACM Press.

Ulicny, B. Putting your customers' answers to work. 2003. Keynote address, *AAAI Spring Symposium on New Directions in Question Answering*.

Woods, W., Stephen Green, Paul Martin, and Ann Houston. Halfway to Question Answering. *Proceedings of TREC 2000.*

Using Domain Ontologies to Discover Direct and Indirect Matches for Schema Elements *

Li Xu
Department of Computer Science
University of Arizona South
lxu@email.arizona.edu

David W. Embley
Department of Computer Science
Brigham Young University
embley@cs.byu.edu

Abstract

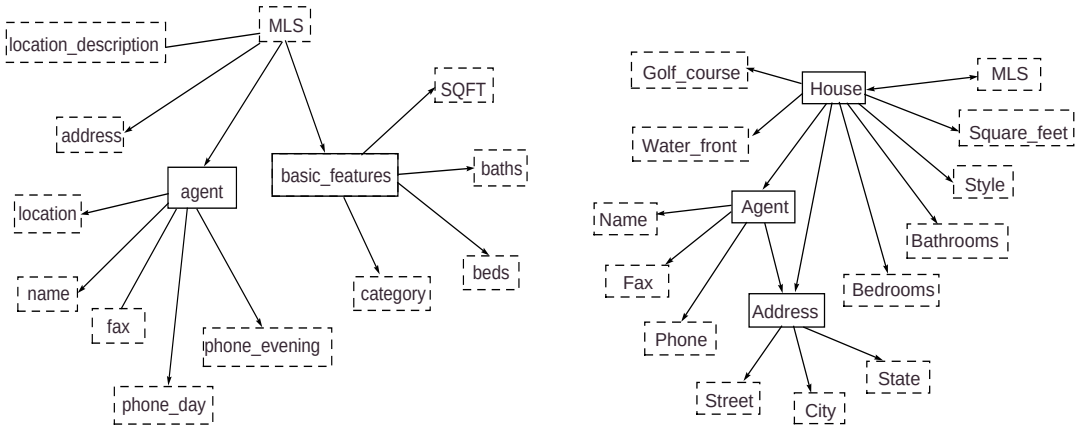
Automating schema matching is challenging. Previous approaches (e.g. [DDH01, RB01]) to automating schema matching focus on computing direct matches between two schemas. Schemas, however, rarely match directly. Thus, to complete the task of schema matching, we must also compute indirect matches. In this paper, we focus on recognizing expected values as a technique to discover many direct and indirect matches between a source schema and a target schema. This technique relies on domain ontologies, which must be handcrafted. The benefits appear to justify the cost as demonstrated in the experiments we have conducted over a real-world application. The experiments show that this technique increases the results by an increase about 20 percentage points, yielding an overall result above 90% precision and recall for both direct and indirect matches.

1 Introduction

In this paper, we focus on the long-standing and challenging problem of automating schema matching [RB01]. Schema matching is a key operation for many applications including data integration, schema integration, message mapping in E-commerce, and semantic query processing [RB01]. Schema matching takes two schemas as input and produces a semantic correspondence between the schema elements in the two input schemas [RB01]. In this paper, we assume that we wish to map schema elements from a *source* schema into a *target* schema. In its simplest form, the semantic correspondence is a set of *direct matches* each of which binds a source schema element to a target schema element if the two schema elements are semantically equivalent. To date, most research [DDH01, RB01] has focused on computing direct matches. Such simplicity, however, is rarely sufficient, and researchers have thus proposed the use of queries over source schemas to form virtual schema elements to bind with target schema elements [BE03, MHH00]. In this more complicated form, the semantic correspondence is a set of *indirect matches* each of which binds a virtual source schema element to a target schema element through appropriate manipulation operations over a source schema.

We assume that all source and target schemas are described using conceptual-model graphs (a conceptual generalization of XML). We augment schemas with sample data and regular-expression recognizers. For each application, we construct a domain ontology [ECJ+99], which declares the regular-expression recognizers for a set of concepts and relationships among the concepts. We use the regular-expression recognizers and relationships among the concepts to discover both direct and indirect matches between two arbitrary schemas. In this paper, we offer the following contributions: (1) a way to discover many direct and indirect semantic correspondences between a source schema S and a target schema T and (2) experimental results of our implementation to show that our approach to schema matching performs as well (indeed better) than other approaches for direct matches and also performs exceptional well for the indirect matches with which we work. The cost for this increased performance is the development of a domain ontology for a particular application. The benefits, as demonstrated in the experimental results, appear to justify the cost to develop the domain ontology. We present the details of our contribution as follows. Section 2 explains the internal representation of the input and output for schema matching. Section 3 describes the schema-matching technique by applying a domain ontology to discover both direct and indirect matches. Section 4 give an experimental result for a data set used in [DDH01] to demonstrate the contribution of applying domain ontologies to schema matching. In Section 5 we summarize, consider future work, and draw conclusions.

*This material is based upon work supported by the National Science Foundation under grant IIS-0083127.



(a) Schema 1

(b) Schema 2

Figure 1: Conceptual-model graphs for Schema 1 and Schema 2

2 Internal Representation

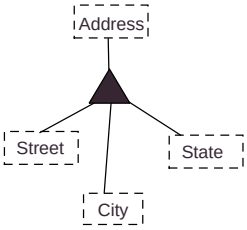
We use conceptual graphs to represent both the target schema and the source schemas as conceptual-model specifications. Each conceptual schema has an *object/relationship-model instance* that describes sets of objects, sets of relationships among objects, and constraints over object and relationship sets. An object set contains either data values or object identifiers, which we respectively call a *lexical object set* or a *nonlexical object set*. A relationship set contains tuples of objects representing relationships connecting object sets. Figure 1, for example, shows two schema graphs. In a schema graph we denote lexical object sets as dashed boxes, nonlexical object sets as solid boxes, functional relationship sets as lines with an arrow from domain object set to range object set, and nonfunctional relationship sets as lines without arrowheads. For either a target or a source schema, we use an object/relationship-model instance to represent schema-level information in our approach for schema mapping. An optional component of a conceptual schema is a set of *data frames*, each of which describes the data of a lexical object set. A data frame is like a type which describes data instances, but can be much more expressive. A data-frame description can be as simple as a list of potential values for an object set and can be as complex as a regular-expression specification that represents values for the object set. For target and source schemas in this paper, data frames are lists of actual or sample values.

In addition to the schema- and instance-level information available from the input source and target schemas, for a particular application domain, we can specify a domain ontology [ECJ$^+$99], which includes a set of concepts and relationships among the concepts, and associates with each concept a set of regular expressions that matches values and keywords expected to appear for the concept. Then using techniques described in [ECJ$^+$99], we can extract values from sets of data for source and target elements and categorize their data-value patterns based on the expected values and keywords declared for application concepts. The derived data-value patterns and the declared relationship sets among concepts in the domain ontology can help discover both direct and indirect matches for schema elements. Figure 2 shows three components in a real-estate domain ontology, which we used to automate matching of the two schemas in Figure 1 and also for matching real-world schemas in the real-estate domain in general. The three components include an address component specifying $Address$ as potentially consisting of $State$, $City$, and $Street$;[1] a phone component specifying $Phone$ as a possible superset of $Day\ Phone$, $Evening\ Phone$, $Home\ Phone$, $Office\ Phone$, and $Cell\ Phone$;[2] and a lot-feature component specifying $Lot\ Feature$ as a possible superset of $View$ and $Lot\ Size$ values and individual values $Water\ Front$, $Golf\ Course$, etc.[3] Behind a dashed box (or individual value), a regular-expression recognizer [ECJ$^+$99] describes the expected values and keywords for a potential application concept. The ontology explicitly declares that (1) the expected values in $Address$ match with a concatenation of the expected values for $Street$, $City$ and $State$; (2) the set of values associated with $Phone$ is a superset of the values in $Day\ Phone$, $Evening\ Phone$, $Home\ Phone$, $Office\ Phone$, and $Cell\ Phone$; and (3) the set of values associated with

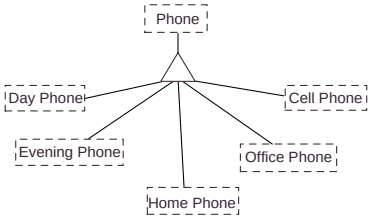
[1] Filled-in (black) triangles denote aggregation ("part-of" relationships).

[2] Open (white) triangles denote generalization/specialization ("ISA" supersets and subsets).

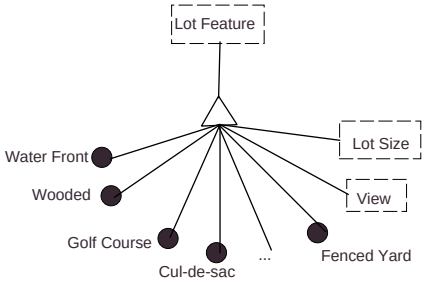
[3] Large black dots denote individual objects or values.



(a) Address



(b) Phone



(c) Lot Feature

Figure 2: Application domain ontology (partial)

$Lot\ Feature$ is a superset of the values associated with the set of $View$ values, the set of $Lot\ Size$ values, the singleton-sets including $Water\ Front$, $Golf\ Course$, $Wooded$, $Fenced\,Yard$, $Cul-de-sac$, etc.

For any schema H, which is either a source schema or a target schema, we let Σ_H denote the union of object sets and relationship sets in H. Our solution allows a variety of source derived data, including missing generalizations and specializations, merged and split values, and transformation of attributes with Boolean indicators into values and vice versa. Therefore, our solution "extends" the source schema elements in Σ_H to include view schema elements, each of which we call a *virtual* object or relationship set. We let V_H denote the extension of Σ_H with derived, virtual object and relationship sets. We consider a source-to-target mapping between a source schema S and a target schema T as a function f_{ST}. The domain of f_{ST} is V_S, and the range of f_{ST} is Σ_T. Thus we can denote a source-to-target mapping as a function $f_{ST}(V_S) \rightarrow \Sigma_T$. Intuitively, a source-to-target mapping represents an one-to-one mapping between a view-augmented source schema and a target schema.

3 Matching Technique

Provided with the domain ontology described in Figure 2 and a set of data values for elements in Schema 1 in Figure 1(a) and Schema 2 in Figure 1(b), we can discover indirect matches as follows. (We first introduce the idea with examples and then more formally explain how this works in general.)

1. *Merged/Split Values*. Based on the $Address$ declared in the ontology in Figure 2, the recognition-of-expected-values technique [ECJ$^+$99] can help detect that (1) the values of $address$ in Schema 1 of Figure 1(a) match with the ontology concept $Address$, and (2) the values of $Street$, $City$, and $State$ in Schema 2 of Figure 1(b) match with the ontology concepts $Street$, $City$, and $State$ respectively. Thus, if Schema 1 is the source and Schema 2 is the target, we can use manipulation *Decomposition* operators to split the values for $address$ in the source as the values for three virtual object sets such that the three virtual object sets match with $Street$, $City$, and $State$ respectively in the target. If we let Schema 2 be the source and Schema 1 be the target, based on the same information, we can identify an indirect match that declares a virtual object set derived by applying a manipulation *Composition* operator over the source to merge values in $Street$, $City$, and $State$ to directly match with $address$ in the target.[4]

2. *Superset/Subset Values*. Based on the specification of the regular expression for $Phone$, the schema elements $phone\_day$ and $phone\_evening$ in Schema 1 of Figure 1(a) match with the concepts $Day\ Phone$ and $Evening\ Phone$ respectively, and $Phone$ in Schema 2 of Figure 1(b) also matches with the concept $Phone$. $Phone$ in the ontology explicitly declares that its set of expected values is a superset of the expected values of $Day\ Phone$ and $Evening\ Phone$. Thus we are able to identify the indirect matching schema elements between $Phone$ in Schema 2 and $phone\_day$ and $phone\_evening$ in Schema 1. If Schema 1 is the source and Schema 2 is the target, we can apply a manipulation *Union* operator over Schema 1 to derive a virtual $Phone'$ whose values are a superset of values in $phone\_day$ and $phone\_evening$. Thus $Phone'$ can directly match with $Phone$ in Schema 2. If Schema 2 is the source and Schema 1 is the target, we may be able to recognize keywords such as *day-time*, *day*, *work phone*, *evening*, and *home* associated with each listed phone

[4]When applying the manipulation operations over sources in data-integration applications, the data-integration system requires routines to merge/split values so that correctly retrieving data from sources.

in the source. If so, we can use a manipulation *Selection* operator to sort out which phones belong in which specialization (if not, a human expert may not be able to sort these out either).

3. *Object-Set Name as Value.* In Schema 2 of Figure 1(b) the features $Water\_front$ and $Golf\_course$ are object-set names rather than values. The Boolean values "Yes" and "No" associated with them are not the values but indicate whether the values $Water\_front$ and $Golf\_course$ should be included as description values for $location\_description$ of $house$ in Schema 1 of Figure 1(a). Because regular-expression recognizers can recognize schema element names as well as values, the recognizer for $Lot\ Feature$ recognizes names such as $Water\_front$ and $Golf\_course$ in Schema 2 as values. Moreover, the recognizer for $Lot\ Feature$ can also recognize data values associated with $location\_description$ in Schema 1 such as "*Mountain View*", "*City Overlook*", and "*Water-Front Property*". Thus, when Schema 1 is the source and Schema 2 is the target, whenever we match a target-schema-element name with a source $location\_description$ value, we can declare "Yes" as the value for the matching target concept by applying a manipulation *Boolean* operator over the $location\_description$ value. If, on the other hand, Schema 2 is the source and Schema 1 is the target, we can declare that the schema element name should be a value for $location\_description$ for each "Yes" associated with the matching source element by applying a manipulation *DeBoolean* operator.

We now more formally describe these three types of indirect matches. Let c_i be an application concept, such as $Street$, and consider a concatenation of concepts such as $Address$ components. Suppose the regular expression for concept c_i matches the first part of a value v for a schema element and the regular expression for concept c_j matches the last part of v, then we say that the concatenation $c_i \circ c_j$ matches v. In general, we may have a set of concatenated concepts C_s match a source element s and a set of concatenated concepts C_t match a target element t. For each concept in C_s or in C_t, we have an associated hit ratio. Hit ratios give the percentage of s or t values that match (or are included in at least some match) with the values of the concepts in C_s or C_t respectively. We also have a hit ratio r_s associated with C_s that gives the percentage of s values that match the concatenation of concepts in C_s, and a hit ratio r_t associated with C_t that gives the percentage of t values that match the concatenation of concepts in C_t. To obtain hit ratios for Boolean fields recognized as schema-element names, we distribute the schema-element names over all the Boolean fields that have "Yes" values.

We decide if s matches with t directly or indirectly by comparing C_s and C_t when the hit ratios r_s and r_t are above an accepted threshold. If C_s equals C_t, we declare a *direct* match (s, t). Otherwise, if $C_s \supset C_t$ ($C_s \subset C_t$), we derive an *indirect* match (s, t) through a *Decomposition* (*Composition*) operation. If both C_s and C_t contain one individual concept c_s and c_t respectively, and if the values of concept c_s (c_t) are declared as a subset of the values of concept c_t (c_s), we derive an *indirect* match (s, t) through a *Union* (*Selection*) operation. When we have schema-element names as values, distribution of the name over the Boolean value fields converts these schema elements into standard schema elements with conventional value-populated fields. Thus no additional comparisons are needed to detect direct and indirect matches when schema-element names are values. We must, however, remember the Boolean conversion for both source and target schemas to correctly derive indirect matches.

We compute the confidence value for a mapping (s, t), which we denoted as $conf(s,t)$, as follows. If we can declare a direct match or derive an indirect match through manipulating *Union*, *Selection*, *Composition*, *Decomposition*, *Boolean*, and *DeBoolean* operators for (s, t), we output the highest confidence value 1.0 for $conf(s,t)$. Otherwise, we construct two vectors v_s and v_t whose coefficients are hit ratios associated with concepts in C_s and C_t. To take the partial similarity between v_s and v_t into account, we calculate a VSM [BYRN99] cosine measure $cos(v_s, v_t)$ between v_s and v_t, and let $conf(s,t)$ be $(cos(v_s, v_t) \times (r_s + r_t)/2)$.

Figure 3 shows the matrix containing confidence values computed based on expected values declared in the domain ontology of Figure 2 using Schema 1 in Figure 1(a) as a source schema and Schema 2 in Figure 1(b) as a target schema.[5] The schema elements along the top are source schema elements taken from Schema 1. The schema elements on the left are target schema elements taken from Schema 2. Observe that the technique correctly identifies the indirect matches between $location\_description$ in the source and $Golf\_course$ and $Water\_front$ in the target, between $phone\_day$ and $phone\_evening$ in the source and $Phone$ in the target, and between $address$ and $location$ in the source and $Street$, $City$, and $State$ in the target. Note that in Figure 3 there are several nonlexical object sets whose values are object identifiers in Schema 1 and Schema 2. An *NA* in the matrix denotes that the object identifiers associated with either the source object set in a column or the target object set in a row are not applicable for value analysis. Furthermore, for this example, we did not include the specifications for expected values or keywords of "bedrooms" and "bathrooms" in our domain ontology. The values for $Bedrooms$ and $Bathrooms$ in

[5] In order to make the matrix fit the page, we use several abbreviations of object-set names in the source schema.

the target and the values for $beds$ and $baths$ in the source do not match any concept in the domain ontology. If one set of data values corresponds to the expected values specified for a concept and another set of data values does not correspond to any concept in the ontology, the confidence is 0.0. For example, the confidence $conf(baths, Phone)$ is 0.0 because the values for $Phone$ in the target correspond to the concept *Phone* in the ontology, but the values for $baths$ in the source do not. If neither values of a pair corresponds to any concept specification in the ontology,[6] the entry is *NA*. For example, the *NA* for the pair ($baths$, $Bathrooms$) denotes that the data values for neither $baths$ in the source nor $Bathrooms$ in the target match any concept in the real-estate domain ontology. If the domain ontology is not complete with respect to an application, our approach needs other matching techniques to discover matches that are not discovered through comparing expected values in the domain ontology.

4 Experimental Results

We evaluate the performance of our approach based on three measures: precision, recall and the F-measure, a standard measure for recall and precision together [BYRN99]. We considered a real-world application, *Real Estate*, to evaluate our matching technique. We used a data set downloaded from the LSD homepage [DDH01] for the applications, and we faithfully translated the schemas from DTDs to conceptual-model graphs. The *Real Estate* application has five schemas. We decided to let any one of the schema graphs be the target and let any other schema graph be the source. We decided not to test any single schema as both a target and a source. In summary, we tested 20 pairs of schemas for the *Real Estate* application. In order to evaluate the contribution of the domain ontology-based matching technique, we tested two runs when comparing a source schema and a target schema. In the first run, we considered only matching techniques that compare object-set names and exploit structure properties [XE03]. In the second run, we added our matching techniques based on domain ontologies.

In the 20 pairs of application schemas, the problems of *Merged/Split Values* appear four times, the problems of *Superset/Subset Values* appear 48 times, and the problems of *Object-Set Name as Value* appear 10 times. With all other indirect and direct matches, there are a total of 876 object-set and relationship-set matches. In the first run, the performance reached 73% recall, 67% precision, and an F-measure of 70%. In the second run, which used a real-estate domain ontology, the performance improved dramatically and reached 94% recall, 90% precision, and an F-measure of 92%. By applying the domain ontology, the algorithm successfully found all the indirect matches related to the four problems of *Merged/Split Values* and all the indirect matches related to the 10 problems of *Object-Set Name as Value*. For the problem of *Superset/Subset Values*, the algorithm correctly found all the indirect matches related to 44 of 48 problems and incorrectly declared four extra *Superset/Subset Values* problems. Of these eight, six of them were ambiguous, making it nearly impossible for a human to decide, let alone a machine. In four of the six ambiguous cases there were various kinds of phones for firms, agents, contacts, and phones with and without message features, and in another two cases there were various kinds of descriptions and comments about a house written in free-form text. The two clearly incorrect cases happened when the algorithm unioned (selected) office and cell phones and mapped them to phones for a firm instead of just mapping office phones to firm phones and discarding cell phones, which had no match at all in the other schema.

One obvious limitation to our approach is the need to manually construct an application-specific domain ontology. Our experience in teaching others to use our system suggests that a domain ontology of the kind we use can be created in a few dozen person-hours. This is not inordinately long; indeed, it is comparable to the time it takes to make a training corpus for machine learning. Moreover, since we predefine a domain ontology for a particular application, we can compare any two schemas for the application using the same domain ontology, so that the work of creating a domain ontology is amortized over repeated usage. Further, the domain ontology does not necessarily need to cover all concepts and relationships in the application schemas, even though it can be revised to help discover more direct and indirect matches.

5 Conclusions and Future Work

Based mainly on expected values declared in domain ontologies, we presented a matching technique for automatically discovering many direct and indirect matches between sets of source and target schema elements. We detected indirect matches related to problems such as *Superset/Subset values*, *Merged/Split values*, as well as *Object-Set Names as Value*. Without this technique, the precision and recall results of the experiments we conducted were only in the neighborhood of 70%, whereas with this technique the results increased to over 90%.

Since domain ontologies appear to play an important role in indirect matching, finding ways to semi-automatically generate them is a goal worthy of some additional work. It is possible to use learning techniques to collect a set of informative and representative keywords

[6] We are not able to compare the expected values without the help of the domain ontology.

	MLS	bath.	bed.	cat.	SQ.	location_ desc.	basic_ features	agent	fax	phone_ day	phone_ evening	name	location	address
House	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Bathrooms	0.0	NA	NA	NA	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Bedrooms	0.0	NA	NA	NA	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
MLS	1.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Square_feet	0.0	0.0	0.0	0.0	1.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Water_front	0.0	0.0	0.0	0.0	0.0	1.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Golf_course	0.0	0.0	0.0	0.0	0.0	1.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0
Address	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Agent	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA	NA
Fax	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	1.0	0.0	0.0	0.0	0.0	0.0
Phone	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	1.0	1.0	0.0	0.0	0.0
Name	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	1.0	0.0	0.0
Street	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	1.0	1.0
State	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	1.0	1.0
City	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	1.0	1.0
Style	0.0	0.0	0.0	0.0	0.0	0.0	NA	NA	0.0	0.0	0.0	0.0	0.0	0.0

Figure 3: Expected-data-values confidence-value matrix

for application concepts in domain ontologies. Thus, without human interaction except for some labeling, we can make use of many keywords taken from the data of the application itself and thus specify some regular-expression recognizers for the application concepts in a semi-automatic way. Furthermore, many values, such as dates, times, and currency amounts are common across many application domains and can easily be shared.

References

[BE03] J. Biskup and D.W. Embley. Extracting information from heterogeneous information sources using ontologically specified target views. *Information Systems*, 28(3):169–212, May 2003.

[BYRN99] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*. Addison Wesley, Menlo Park, California, 1999.

[DDH01] A. Doan, P. Domingos, and A. Halevy. Reconciling schemas of disparate data sources: A machine-learning approach. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data (SIGMOD'01)*, pages 509–520, Santa Barbara, California, May 21–24 2001.

[ECJ+99] D.W. Embley, D.M. Campbell, Y.S. Jiang, S.W. Liddle, D.W. Lonsdale, Y.-K. Ng, and R.D. Smith. Conceptual-model-based data extraction from multiple-record Web pages. *Data & Knowledge Engineering*, 31(3):227–251, November 1999.

[MHH00] R. Miller, L. Haas, and M.A. Hernandez. Schema mapping as query discovery. In *Proceedings of the 26th International Conference on Very Large Databases (VLDB'00)*, pages 77–88, Cairo, Egypt, September 10–14 2000.

[RB01] E. Rahm and P.A. Bernstein. A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350, December 2001.

[XE03] L. Xu and D.W. Embley. Discovering direct and indirect matches for schema elements. In *Proceedings of the 8th International Conference on Database Systems for Advanced Applications (DASFAA 2003)*, pages 39–46, Kyoto, Japan, March 26–28 2003.

Translating Naive User Queries on the Semantic Web

Baoshi Yan, Robert MacGregor
Distributed Scalable Systems Division
Information Sciences Institute
University of Southern California
{baoshi,macgregor}@isi.edu

ABSTRACT

Query is an important way of information retrieval. One type of queries is those to search engines, which are lists of keywords without structures. Another type of queries is those to databases or knowledge bases, which must conform to the structure and terminology of the data source (e.g., SQL query to a database). In this paper, we deal with another type of queries: *naive user queries*–queries in users' own terms and structures. We envision that this type of queries would be a common phenomenon on the future Semantic Web. We propose an approach that, given a naive user query, translates it into a list of queries conforming to different data source schemas. The approach is based on partial alignment between some data sources. An early prototype showed that the result is promising.

1. INTRODUCTION

When people want to retrieve information, they usually issue a query. The most used form is queries to search engines, which is a combination of keywords. However, people cannot specify semantic structure between these keywords. This is more due to that search engines mostly deal with natural language texts that are hard to extract semantic structures from. On the other hand, databases and knowledge bases have semantic structures, but they require queries (e.g., a SQL query to a DB, or an ASK function to a KB) to conform to their terminology and structures. On the future Semantic Web[6], neither form of queries is sufficient. Search-engine-style queries don't impose restrictions on the terms used, but don't have semantic structure either, which put them in a disadvantaged situation in a web of semantic structures. Queries in DB or KB style have to conform to the schema used by individual data sources. On the Semantic Web, there'll be numerous data sources with different schemas. Requiring people to write different queries for individual data sources is a daunting task.

Thus we propose that it is necessary to deal with another type of user queries: *naive user queries*–queries in users' own terms and semantic structures. Instead of letting information seekers understand the schema used by an information provider, we could shift the load to the information provider to understand naive user queries.

Without losing generality, we represent a naive user query as a list of triple patterns (s,p,o)[1]. Table 1 lists some examples. We represent data source schemas as RDFS schemas [1](or ontologies). Equivalent queries can be written with different terms, as can be seen from first two queries in the table. Semantic structures between terms are binary relations: p is the kind of relationship between s and o. The triple patterns roughly conform to RDF [3] data model except that the terms here are users' descriptions, not URI's (Universal Resource Identifiers). The triple patterns allow people to specify semantic structures, and we think it keeps simple enough so that ordinary users can write it. It is also not hard to come up with a GUI to automatically add syntax sugar with little user effort.

Such type of user queries might not conform to the schemas of available data sources. In the rest of the paper, we discuss an approach to translate such queries into equivalent ones conforming to actual data source schemas.

2. PROBLEM FORMULATION

Translating naive user queries is difficult. This is because people could use all kinds of terms and structures to represent the same thing. However, we have two observations that would greatly ease the problem.

The first observation is that the variation of terms and structures used for a concept is much less than the variation of data schemas and queries. The huge variations of schemas or queries are mainly due to the combinatorial explosion of variations of terms and structures. For example, "moviename" and "movietitle" are two terms to represent a movie name; "rating" and "movierating" are two terms to represent rating of a movie. Their combination could lead to four data schemas. If we take into account that RDFS schemas qualify each term with URI's (so there might exist "http://com1#moviename", "http://com2#moviename"), then the number of schemas is infinite.

In a class at the University of Southern California, students are required to create a RDFS schema at their choices as a Semantic Web assignment. Seven of them happened to choose movie-related schemas. Table 2 shows parts of these seven schemas, in which many term duplicates can be seen.

The second observation is that among the terms used to represent the same concept, a few number of terms are used more often than others. The 20-80 law seems to apply here, which implies that a small number of terms will account for most usages.

[1]Although syntax doesn't affect our discussion, we use RDQL-alike [2] syntax for convenience.

.

#	Naive User Query	Meanings
1	select ?2 where (?1, moviename, "The Ring")(?1, director, ?2)	Find the director of the movie "The Ring"
2	select ?2 where (?1, title, "The Ring")(?1, directorname, ?2)	Find the director of the movie "The Ring"
3	select ?2 where (?1, type, ThrillerMovie) (?1, name, ?2)	Find all the thriller movies in the data source

Table 1: Some Naive User Query Examples

The above two observations led us to believe that, if we could accumulate a number of variations of terms and structures, we might be able to translate a great deal of naive user queries by matching their combinations against the given query. This belief is further strengthened by the fact that naive user queries tend to be short and simple.

Although it is possible that terms accumulated in one domain might help translating naive queries in other domains, we'll focus on only one domain in this paper. We are trying to solve the following kind of problems.

Problem Formulation:

Data A list of data schemas in the same domain, and some alignments between some properties and classes in some schemas (i.e., partial alignments).

For the examples in this paper, we use the seven movie schemas. Among the properties and classes specified in Table 2, we aligned the properties and classes in Schema 1 with those in Schema 6, and those in Schema 4 with those in Schema 5. This is all the alignment information we use throughout the paper.

Input Any given naive user query.

In the examples described in this paper, the given naive query is (?1, name, "The Ring") (?1, moviedirector, ?3) (?1, type, ThrillerMovie).

Output Translations of the given user query into each individual schema.

Goal The goal is to find as much correct translations as possible. We can use concepts of precision and recall from information retrieval as metrics. Precision measures the percentage of correct translations among all the generated translations. Recall measures how many correct translations are generated by the algorithm out of all possible correct translations.

We'll talk about how we handle this problem in the next section.

3. APPROACH

We start with two not-so-satisfactory approaches we've tried and then introduce our adopted approach. For illustration purpose, let's assume we are given a user query $(?1, name, ''TheRing'')(?1, moviedirector, ?2)$ and try to translate it into the movie schemas we have.

The first approach is to build a global schema. Just think of each column in the Table 2 as defining a concept in the global schema, with all the cells in the column as possible labels of the concept. Translation with this global schema is easy. However, constructing and maintaining such a global schema is difficult. Also, it is hard to represent complex alignments (e.g., one's parent whose gender is female is one's mother) in the global schema.

Another approach we tried without constructing a global schema is to map the problem into a path searching problem. The idea is to construct a graph $G(V, E)$ with V being the node set representing all the classes and properties from all schemas, and $E = E_1 \bigcup E_2$ where $E_1 = \{(v_1, v_2)|v_1 \approx v_2\}$, $E_2 = \{(v_1, v_2)|v_1$ and v_2 belong to the same schema$\}$. $v_1 \approx v_2$ means that there exist some kind of alignment between node v_1 and v_2. Given such a graph, we could look up the node (or a set of nodes) v_1 with label "name" and the node v_2 with label "moviedirector", and then we compute the paths [18] between v_1 and v_2. The intuition here is that if there's no path between v_1 and v_2 then there's no schema that would contain a translation of the query. After some post-processing on the resulted paths we could infer possible translations of the original query into different schemas. The problem with this approach is still the difficulty of representing complex alignments.

The approach we finally used is based on query rewriting. The intuition is to rewrite the original query based on available alignments. Then we check resulted translations to pick out those making sense. The algorithm works as following:

Decomposition Step: Decompose the original query into individual triple patterns t_1, t_2....

Query Rewriting Step: For each triple pattern $t = (s, p, o)$, find all possible rewritings of it according to the following rules. Repeat this step until no more new rewritings are produced.

Exact Name Matching: if p is a local name[2], find all properties whose local names match. If p is already a property name, find all properties with the same local name. For each matched property name p_i, produce a rewriting $t_i = (s, p_i, o)$. If p is a type predicate name, it indicates that o is a class name. Do the same thing to find out all classes with the same local name. For each matched class name C_j, produce a rewriting $t_j = (s, p, C_j)$.

Approximate Name Matching: if p is a local name, we also find all properties whose local name will match p after some manipulations. The manipulations are on local names of a property and its domain classes (i.e., the classes it adheres to). For example, if a property's local name is "movieName" and its domain class' local name is "Movie", then it matches the p of "name". If its local name is "name", then it matches "movieName", "hasName", and "hasMovieName". If p is a property name, we produce several variations of its local name and look for other properties with the same local name as those variations. We are not afraid of producing meaningless local names, because there won't be properties matching the meaningless names anyway. Note that some other approximate name matching techniques could

[2]In RDFS, a property is normally represented as a URI such as http://movie.org#movieName. The part after # is so-called local name.

also be used, such as the one based on edit distance [11]. As previous, we'll produce some new rewritings.

DESCRIPTION MATCHING: if p is a long string, it would be desirable to compare p with property descriptions. We haven't implemented this yet.

ALIGNMENT: We represent alignments as query rewriting rules. For example, the alignment between "movieName" in Schema 1 and "title" in Schema 3 is represented as $(?1, S1 : movieName, ?2) \leftrightarrow (?1, S3 : title, ?2)$. The "ThrillerMovie" class in S1 is aligned to the "Movie" class in Schema 5 with a "movieGenre" property of value "Thriller": $(?1, type, S1 : ThrillerMovie) \leftrightarrow (?1, type, S5 : Movie)(?1, S5 : movieGenre, ''Thriller'')$. Triple patterns matching either side of an alignment rule would result in a rewriting based on the other side of the rule.

INFERENCE: For a query on a class, its subclasses also matchthe query, i.e., for each C_1 that is a subclass of C_2 we have $(?1, type, C_2) \rightarrow (?1, type, C_1)$. Here $\rightarrow$ is not logical inference; it represents translation direction. For example, a "ThrillerMovie" matches a query looking for a "Movie". We expect that other kinds of inference rules could also be used.

As an example, Table 3 shows a sample series of rewritings starting with triple pattern $(?1, name, ''TheRings'')$.

Pruning Step 1: As a result of the last step, we'll have a list of rewritings for each individual triple pattern in the original query. Note that because of the use of alignment and inference rules, a rewriting for an original triple pattern might contain more than one triple pattern. Thus a rewriting might contain properties and classes from different schemas. Such rewriting doesn't make sense in the final answer, thus is removed. We'll illustrate this with examples in Table 4 later.

Pruning Step 2: For each schema used in the rewritings, we check whether all the original triple patterns have rewritings in that schema. If the answer is no and partial translation is not allowed, we could prune all the rewritings in that schema. We'll illustrate this with examples in Table 5 later.

Checking Step: Now for each schema left, for each original triple pattern we pick a rewriting of it in the schema. The picked rewritings form a possible translation of the original user query in the schema. We check the semantic structure of translation against that of the schema; the translation is removed if semantic structures don't match. We'll illustrate this with examples in Table 5 later.

The checking step is crucial. In the query rewriting step we try to come up with as many translations as possible, and the checking step ensures that wrong translations are actually removed and final answers make sense.

Let's illustrate our approach with one experiment. The data we used and the query we faced are those mentioned in the "Problem Formulation" of Section 2.

Table 4 shows that the rewriting $(?1, S4 : movieDirector, ?3)(?3, S5 : personName, ??3)$ for input query pattern $(?1, moviedirector, ?3)$ is pruned in this step, which is because the rewriting contains information from both Schema 4 and Schema 5. Thus the rewriting doesn't make sense as a part of the final translation.

Table 5 shows the result of Pruning Step 2 and Checking Step. The first translation in the table is pruned because it doesn't have a rewriting for the input triple pattern $(?1, type, ThrillerMovie)$. The third translation is pruned because its rewriting $(?1, S6 : Director, ?3)(?3, S6 : Title, ??3)$ doesn't match the semantic structure of Schema 6. In Schema 6 "S6:Title" is a property of the "S6:Movie" class and the value of the "S6:Director" property is a string which cannot have properties. Similarly, the fourth translation is pruned because an "S5:Movie" cannot have an "S5:personName" property.

The experiment turned out to be successful. It didn't contain erroneous translations, and found all correct translations. However, we have yet to prove that our approach will work in real life. This is more due to that we lack real data and real users to experiment with. Nevertheless, the preliminary results with our current (very limited) data are encouraging.

Another desirable feature of our approach is that the whole architecture is extensible. Different kinds of knowledge, from exact name matching to inference rules, can be utilized in the query rewriting step. The ability to incorporate inference rules is especially important for the Semantic Web.

4. EVALUATION

The ideal evaluation of our approach would be to test it with a number of real user queries, and to measure the performance with metrics like precision and recall as described in Section 2. It would also be helpful to adjust the number of alignments in the knowledge base and to see how the precision and recall change accordingly. However, collecting a large number of real user queries is difficult. Instead, for evaluation purpose, we searched for other movie schemas on the web, and use them construct naive user queries.

To search for movie schemas on the web, we picked a few keywords from current movie schemas and submitted them to Google. The HTML pages returned by Google often contain structured information that is like a schema. For example, below is a segment from the web page at http://www.moviepublicity.com/ppvvod/.

```
Director:Jim Isaac
Starring:Kane Hodder, Lexa Doig
Rating:R
Genre:Action/Horror
Run Time:91 minutes 30 seconds
Box Office: \$12,610,731
```

We gathered 17 movie schemas from the web in this way. For each schema we constructed a big query that involved all concepts in the schema. Note that some concepts like "Box Office" are not present in the 7 schemas. Thus we define "recall" on subqueries rather than on the whole big query. We define "recall" as the percentage of found translations out of all possible translations of subqueries. We define "precision" as the percentage of correct translations out of all translations of subqueries. The experiment showed a recall of 64% for all subquries. For the subquries semantically on the properties we have aligned (those in Table 2), it showed a recall of 72%. This result is already promising given that we've only aligned two pairs of schemas out of only 7 schemas. The experiment showed a precision of 100%. We attributed the high precision to the small knowledge base and the semantic correctness of the alignments. Given such a knowledge base, the algorithm either translates a subquery correctly or rejects it. We envision that with the size of knowledge base grows, the precision will decline and

SOME CLASSES IN DIFFERENT SCHEMAS

S1	[Movie]	[ThrillerMovie](subclass of [Movie])	[ComedyMovie](subclass of [Movie])
S2	[Movie]		
S3	[VideoLibraryItem]		
S4	[Movie]	[ThrillerMovie](subclass of [Movie])	[ComedyMovie](subclass of [Movie])
S5	[Movie]	[Movie]→movieGenre="Thriller"	[Movie]→movieGenre="Comedy"
S6	[Movie]	[Movie]→movieGenre→[Genre] →GenreName="Thriller"	[Movie] →movieGenre→[Genre] →GenreName="Comedy"
S7	[MovieInfo]		

SOME PROPERTIES IN DIFFERENT SCHEMAS

S1	movieName	directorName	actor
S2	Name	hasCrew→[Director]→PersonName	
S3	Title	Director	Actor
S4	movieName	movieDirector	leadingActor→[Artist]→artistName
S5	movieName	movieDirector→[Person]→personName	movieActor→[Person]→personName
S6	Title	Director	MainActor
S7	movieName	movieDirector	

S1	actress	rating→[Rating]→ratingType	
S2			genre
S3		Rating	
S4	leadingActress→[Artist]→artistName	movieRating	
S5		movieMPAARating	movieGenre
S6	MainActress	MPAA	movieGenre→[Genre]→GenreName
S7			

Table 2: Seven Movie Schemas

Step#	Rewriting Result	Rewriting Operation	Description
0	(?1,name,"The Ring")		Original Triple Pattern
1	(?1,S1:movieName,"The Ring")	localname matches URI	(?1,name,?2)→(?1,S1:movieName,?2)
2	(?1,S6:Title,"The Ring")	Property Alignment	(?1,S1:movieName,?2)↔(?1,S6:Title,?2)
3	...	...	...

Table 3: A Sample Series of Query Rewriting Steps

	Pruning Step 1
(?1,name,"The Ring")	(?1,S6:Title,"The Ring") (?1,S5:personName,"The Ring") (?1,S5:movieName,"The Ring") .....
(?1,moviedirector,?3)	~~(?1,S4:movieDirector,?3)(?3,S5:personName,??3)~~ ~~(?1,S3:Director,?3)(?3,S2:Name,??3)~~ (?1,S6:Director,?3)(?3,S6:Title,??3) (?1,S6:Director,?3) .....
(?1,type,ThrillerMovie)	~~(?1,type,S6:Movie)(?1,S5:movieGenre,?2)(?2,S6:GenreName,"Thriller")~~ ~~(?1,type,S6:Movie)(?1,S2:genre,?2)(?2,S6:GenreName,"Thriller")~~ (?1,type,S6:Movie)(?1,S6:movieGenre,?2)(?2,S6:GenreName,"Thriller") (?1,type,S5:Movie)(?1,S5:movieGenre,"Thriller") (?1,type,S4:ThrillerMovie) .....

Table 4: Pruning Step 1: Remove Rewritings Containing Different Schemas

	Pruning Step 2 & Checking Step
(?1,name,"The Ring") (?1,moviedirector,?3) (?1,type,ThrillerMovie)	(?1,S3:Title,"The Ring") (?1,S3:Director,?3) ~~()~~-
(?1,name,"The Ring") (?1,moviedirector,?3) (?1,type,ThrillerMovie)	(?1,S6:Title,"The Ring") (?1,S6:Director,?3) (?1,type,S6:Movie)(?1,S6:movieGenre,?2)(?2,S6:GenreName,"Thriller")
(?1,name,"The Ring") (?1,moviedirector,?3) (?1,type,ThrillerMovie)	(?1,S6:Title,"The Ring") (?1~~,S6:Director,?3)(?3,S6:Title,??3~~) (?1,type,S6:Movie)(?1,S6:movieGenre,?2)(?2,S6:GenreName,"Thriller")
(?1,name,"The Ring") (?1,moviedirector,?3) (?1,type,ThrillerMovie)	(?1~~,S5:personName,"The Ring~~") (?~~1,S5:movieDirector,?3~~) (?1,type,S5:Movie)(?1,S5:movieGenre,"Thriller")
.....	.....

Table 5: Pruning Step 2 and Checking Step: Pick Out Good Translations

the recall will increase. Note that our system does not guarantee 100% precision because we use a lot of guessing and aligning. The alignment between two terms seldom means these two terms are 100% equivalent.

5. APPLICATIONS AND FUTURE WORK

In this section, we discuss a couple of potential applications and extensions of our technique.

Information Search on the Semantic Web: The most natural application of our naive query translation technique will be to help information search on the Semantic Web. On the Semantic Web, we envision that numerous small schemas, rather than a few big schemas everyone must follow, will be created by people for their information management tasks. It is almost impossible to align all the schemas. It is also impossible for an information seeker to write all the different queries for different schemas. Thus it is important that, with a few alignments between a few schemas in the same domain, a naive user query can be translated into these schemas as well as others in the domain. The technique proposed in this paper represents our effort toward this great challenge.

In another project called "WebScripter" [19], we are developing a collaborative semantic annotation(CSA) tool for ordinary users to create metadata, and an easy-to-use report authoring tool for users to publish metadata as a user-friendly report as well as to align metadata from different schemas. With the grass-roots ontologies created by ordinary users using CSA and grass-roots alignment obtained from WebScripter, we hope that the technique we described in this paper would facilitate information sharing among WebScripter users.

Building Naive User Query Interface to an Existing Data Source: If we regard a naive user query as a mini-schema defined on-the-fly, it might be possible that, after accumulating and aligning a number of user queries, a system could interpret future naive user queries. Over time the system learns more rewriting rules and more synonyms, so it can produce increasingly robust and comprehensive response to new queries. A such-enhanced data source would provide a friendlier interface to information agents on the web.

Alignment-Carrying Information Agent: Other than enhancing a data source with a knowledge base of possible naive queries, if we arm an information agent with some alignment between some schemas of the domain of interest, will the agent be able to recognize future schemas it sees in the same domain, or at least, rewrite its task(a query) into those of the schemas? This kind of knowledgeable agent, or alignment-carrying agent, is likely to be more autonomous in information retrieval.

In a summary, there are interesting applications of the techniques we have developed. Exploring these applications, meanwhile testing the technique and discovering its deficiencies are our plan for the future work.

6. RELATED WORK

The research most related to our work is schema matching [17]. If we regard a naive user query as a mini-schema defined on-the-fly, translating naive user queries can be viewed as a special schema matching problem. However, there're significant differences between our work and schema matching, in the problem to be solved and in the techniques used.

Data schemas tend to be larger, more complex and rather static. Schema matching tries to make use of any helpful information such as name similarity, structure proximity [15], learning from data instances [9]. To further improve mapping accuracy, integration of all kinds of techniques into a single system is also used [8] [14]. Consequently, schema matching techniques are usually complex and time-consuming, and the matching process is generally assumed to happen offline. In contrast, naive user queries are normally short and dynamic. Timely response is also required.

Our translation approach makes use of a knowledge base of previous alignments. This distinguishes our work from many schema matching algorithms [15][9] [14] that only consider the two schemas at hand. The idea of reusing previous alignments is stated in [17] and further developed in [8]. However, [8] is not as flexible as our approach. In order to match $S1$ and $S2$ it requires the existence of S that has been already matched with $S1$ and $S2$, which makes it unusable for schemas unseen before. Alon Halevy [10] recently proposed to use a corpus of schemas to help schema matching. He also talked about the possibility of using such corpus to enable queries in users' own terminology. Our work goes one step further to also consider queries in users' own semantic structures. Furthermore, our idea of reusing previous alignments of *user queries* to translate future queries has not been seen in other's work.

There has been recent work [12] [4] [7] on enabling keyword-based search over relational databases. These systems try to compute a join of different table tuples which contains all the input keywords. Contrary to naive user queries, the relations between keywords are unclear, and it is difficult to specify what information users are looking for. As a result, users need to check that the relations between keywords in the returned tuples match user intents (For a set of keywords, there can be different join chains). Another human check is then required to extract the information users want from the tuples. Thus keyword-based search is more appropriate as a human activity rather than part of an automated computer program.

Natural language interface to databases provides another kind of query interface. Despite recent progress [16] [20], these systems remain difficult to implement. Natural language queries and SQL queries are two extremes and naive user queries are in the middle. We believe it is worthwhile to investigate whether a naive user query interface is easier to develop and performs better in terms of precision and recall.

Our query rewriting approach resembles a lot to those used in information integration systems [13] [5]. An information integration system translates a query between its global schema and local schemas. It assumes the existence of alignments between global schema and all local schemas. The query must be in one of the known schemas. In contrast, our work is on translating naive user queries that might not conform to any known schema. Our work is complementary to information integration in this sense.

7. CONCLUSION

The contributions of this paper are two-fold. First, we identified translating naive user queries as an important research problem. Translating naive queries is the process of translating queries in users' own terms and structures into those interpretable by data sources. Second, we proposed an approach to this problem. The approach utilizes schemas of different data sources and partial alignments between them to rewrite naive user queries into data-source-interpretable form. The approach is efficient and preliminary results were encouraging. We then showed how our technique could be an important component on the future Semantic Web. We also discussed possible applications of our technique, such as constructing naive-user-query translating interface to data sources and building Alignment-Carrying Information Agents on the current Web.

8. ACKNOWLEDGMENTS

The authors of the paper are supported by DARPA DAML program funding for WebScripter under contract number F30602-00-2-0576. We thank David Wilczynsky and Ellis Horowitz for providing their students' RDF Schemas. We thank Martin Frank for helpful comments.

9. REFERENCES

[1] Rdf vocabulary description language 1.0: Rdf schema. http://www.w3.org/TR/rdf-schema/.

[2] Rdql - rdf data query language. http://www.hpl.hp.com/semweb/rdql.htm.

[3] Resource description framework (rdf) model and syntax specification. http://www.w3.org/TR/1999/REC-rdf-syntax-19990222/.

[4] S. Agrawal and S. C. G. Das. Dbxplorer: A system for keyword-based search over relational databases. In *the 18th International Conference on Data Engineering (ICDE.02)*, 2002.

[5] Y. Arens, C. A. Knoblock, and W.-M. Shen. Query reformulation for dynamic information integration. *Journal of Intelligent Information Systems - Special Issue on Intelligent Information Integration*, 6(2/3):99–130, 1996.

[6] T. Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, May 2001.

[7] G. Bhalotia, A. Hulgeri, C. Nakhe, and S. Chakrabarti. Keyword searching and browsing in databases using banks. In *the 18th International Conference on Data Engineering (ICDE.02)*, 2002.

[8] H. Do and E. Rahm. Coma - a system for flexible combination of schema matching approaches, 2002.

[9] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map ontologies on the semantic web. In *The Eleventh International World Wide Web Conference*, 2002.

[10] A. Halevy, O. E. A. Doan, Z. Ives, and J. Madhavan. Crossing the structure chasm. In *the First Biennial Conference on Innovative Data Systems Research (CIDR)*, 2003.

[11] G. D. Hall, P. Approximate string matching. *Computing Survey*, 12(4):381–402, 1980.

[12] V. Hristidis and Y. Papakonstantinou. Discover: Keyword search in relational databases. In *the 28th VLDB Conference*, 2002.

[13] A. Y. Levy. Logic-based techniques in data integration. In J. Minker, editor, *Workshop on Logic-Based Artificial Intelligence, Washington, DC, June 14–16, 1999*, College Park, Maryland, 1999. Computer Science Department, University of Maryland.

[14] J. Madhavan, P. A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *The VLDB Journal*, pages 49–58, 2001.

[15] S. Melnik, H. Molina-Garcia, and E. Rahm. Similarity flooding: A versatile graph matching algorithm. In *the International Conference on Data Engineering (ICDE)*, 2002.

[16] A. Popescu, O. Etzioni, and H. Kautz. Towards a theory of natural language interfaces to databases. In *IUI*, 2003.

[17] E. Rahm and P. Bernstein. On matching schemas automatically. Technical report, Microsoft Research, Redmon, WA, 2001. MSR-TR-2001-17.

[18] R. Tarjan. Fast algorithms for solving path problems. *Journal of the ACM*, 3(28):591–642, 1981.

[19] B. Yan, M. Frank, P. Szekely, R. Neches, and J. Lopez. Webscripter: Grass-roots ontology alignment via end-user report authoring. In *the Second International Semantic Web Conference*, Octor 2003.

[20] J. M. Zelle and R. J. Mooney. Learning to parse database queries using inductive logic programming. In *the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, 1996.

Knowledge Augmentation for Aligning Ontologies: An Evaluation in the Biomedical Domain

Songmao Zhang and Olivier Bodenreider

U.S. National Library of Medicine, Bethesda, Maryland
National Institutes of Health, Department of Health & Human Services
{szhang|olivier}@nlm.nih.gov

Abstract

The objective of this study is to evaluate the contribution to semantic integration of the semantic relations extracted from concept names, representing augmented knowledge. Three augmentation methods – based on linguistic phenomena – are investigated (reification, nominal modification, and prepositional attachment). The number of concepts aligned in two ontologies of anatomy before and after augmentation serves as the evaluation criterion. Among the 2353 concepts exhibiting lexical resemblance across systems, the number of concepts supported by structural evidence (i.e., shared hierarchical relations) increased from 71% before augmentation to 87% after augmentation. The relative contribution of each augmentation method to the alignment is presented. The limitations of this study and the generalization of augmentation methods are discussed.

Introduction

Ontologies are often organized into concepts (e.g., *Heart*, *Mitral valve*) and semantic relations (e.g., <*Mitral valve*, PART-OF, *Heart*>). As a first approximation, concepts represent categories, while semantic relations represent assertions about the concepts. Both concepts and relations are useful for the semantic integration of ontological resources. Lexical resemblance among concept names may indicate similarity in meaning. Likewise, from a structural perspective, concepts sharing similar relations to other concepts tend to be similar in meaning.

However, the difference between concepts and semantic relations my not be as clear-cut as it seems. Although representing categories, concepts such as *Vein of leg* and *Subdivision of heart* also embed partitive assertions in their names. For example, the relation <*Vein of leg*, PART-OF, *Leg*> can be deduced from the name *Vein of leg*. And *Subdivision of heart* is equivalent to the relation <*X*, PART-OF, *Heart*> where *X* is a placeholder for any concept subsumed by *Subdivision of heart*, including *Mitral valve*. In addition, from the name of the concept *Sweat gland*, one can derive the assertion <*Sweat gland*, IS-A, *Gland*>.

More generally, concept names often embed assertions, i.e., implicit knowledge, not always represented explicitly through semantic relations. In this paper, we examine three linguistic phenomena (reification, nominal modification, and prepositional attachment), which usually embed semantic relations. We show how semantic relations extracted from these concept names contribute to improving the semantic integration – through alignment – of two ontologies of anatomy.

The general framework of this study is that of lexical semantics and knowledge acquisition. Lexical semantics [1] studies the link between linguistic phenomena and the semantic relations they encode. As such, lexical semantics contributes to knowledge acquisition from textual resources. While originally applied to general relations (e.g., hypernymy, meronymy) from general corpora (e.g., machine-readable dictionaries [2]), the same techniques have been adapted to the acquisition of specialized relations (e.g., the molecular interaction BINDS [3]) from the biomedical literature. Terminologies have also been used as specialized corpora for acquiring knowledge [4]. In this particular con-

text (controlled vocabulary, closed subdomain), there is often less ambiguity than in larger textual resources, which may facilitate knowledge extraction. In previous work, we studied semantic relations embedded in biomedical terms through nominal modification [5] and reification [6].

Although sharing with these studies some of the methods used for knowledge acquisition, this paper specifically evaluates the contribution to semantic integration of the semantic relations extracted from concept names through various methods. We demonstrate how each linguistic phenomenon under investigation contributes to improving the alignment of two ontologies of anatomy. This study is not evaluation of the alignment itself, but rather a quantification of the contribution of augmented knowledge to the alignment.

Resources and Methods

Ontologies of anatomy

The **Foundational Model of Anatomy**[1] (FMA) [August 30, 2002 version] is an evolving ontology that has been under development at the University of Washington since 1994 [7]. Its objective is to conceptualize the physical objects and spaces that constitute the human body. The underlying data model for FMA is a frame-based structure implemented with Protégé-2000. With 59,422 concepts, FMA claims to cover the entire range of gross, canonical anatomy. Concept names in FMA are pre-coordinated, and, in addition to preferred terms (one per concept), 28,686 synonyms are provided (up to 6 per concept). For example, there is a concept named *Uterine tube* and its synonym is *Oviduct.*

The Generalized Architecture for Languages, Encyclopedias and Nomenclatures in medicine[2] (**GALEN**) [version 5] has been developed as a European Union AIM project led by the University of Manchester since 1991 [8]. The GALEN common reference model is a clinical terminology represented using GRAIL, a formal language based on description logics. GALEN contains 25,192 concepts and intends to represent the biomedical domain, of which canonical anatomy is only one part. Unlike FMA, GALEN is compositional and generative. Concept names in GALEN are post-coordinated, and only one name is provided for each concept.

Both FMA and GALEN are modeled by *IS-A* and *PART-OF* relationships and allow multiple inheritance. Relationships in GALEN are finer-grained than in FMA. For the purpose of this study, we considered as only one *PART-OF* relationship the various kinds of partitive relationships present in FMA (e.g., *part of, general part of*) and in GALEN (e.g., *isStructuralComponentOf, isDivisionOf*).

Extracting relations from concept names

We used three methods for extracting relations from concept names. Each method takes advantage of one particular linguistic phenomenon. The relations embedded in concept names through these phenomena sometimes coexist with equivalent semantic relations represented explicitly in the ontology. However, cases where a relation is only embedded in a concept name in one ontology and only represented explicitly in the other are likely to impair semantic integration. In order to make ontologies more easily comparable, we systematically extracted the relations embedded in concept names. In this study, we focused on taxonomic (i.e., *IS-A* and *INVERSE-IS-A*) and partitive (i.e., *PART-OF* and *HAS-PART*) relations.

The **reification of *PART-OF*** consists of using a concept named *Part of W* to subsume a concept *P* instead of using a *PART-OF* relationship between the concept *P* (the part) and *W* (the whole). From a linguistic perspective, the concept name *Part of W* reifies the *PART-OF* relationship from concept *P* to *W*. The two representations, <*P*, *IS-A*, *Part of W*> and <*P*, *PART-OF*, *W*>, are equivalent for most purposes [9]. We systematically extracted <*P*, *PART-OF*, *W*> and <*W*, *HAS-PART*, *P*> from concept names such as *Subdivision of X, Organ component of X*, and *Component of X*, where *X* is a concept present in the ontology. For example, because the concept *Component of hand* subsumes *Finger*, we generated the two relations <*Finger*, *PART-OF*, *Hand*> and <*Hand*, *HAS-PART*, *Finger*>.

Nominal modification often represents a hyponymic relation involving the head of the noun phrase. For example, a *Cranial nerve* is a kind of *Nerve* and the *Carotid artery* is a kind of *Artery*. Therefore, the relations <*X Y*, *IS-A*, *Y*> and <*Y*,

[1] http://sig.biostr.washington.edu/projects/fm/AboutFM.html

[2] http://www.opengalen.org/

INVERSE-IS-A, *X Y* > can be tentatively extracted from the term *X Y*. However, this method is not applicable when the head of the noun phrase is polysemous in the domain under investigation. For example, *Body* (human body) does not subsume *Carotid body* (a small neurovascular structure). The problem here lies in the several senses of *body*: "the material part or nature of a human being" for the former and "a mass of matter distinct from other masses" in the latter. Domain knowledge is required for identifying such cases.

In anatomical terms, **prepositional attachment** using "of" (*X of Y*) often denotes a partitive relation between *X of Y* and *Y*. For example, we generated the relations <*Bone of femur*, *PART-OF*, *Femur*> and <*Femur*, *HAS-PART*, *Bone of femur*> from the term *Bone of femur*. Because it does not fully analyze the concept names, this method is not suitable for complex anatomical terms (e.g., names containing prepositions other than "of", such as *Groove for arch of aorta*).

Evaluation

The two ontologies of anatomy, FMA and GALEN, were aligned using a combination of lexical techniques (resemblance among concept names) and structural techniques (similarity and conflicts based on the semantic relations) [10]. In order to evaluate the role of the relations generated by augmentation, the alignment based on the explicit knowledge alone was compared to the alignment based on both explicit and augmented knowledge. In practice, structural techniques were used to refine the alignment of lexically related concepts, called anchors. Structural similarity, used as positive evidence, is defined by the presence of common hierarchical relations among anchors across systems. Conflicts, on the other hand, used as negative evidence, are defined by the existence of opposite hierarchical relationships (e.g., *PART-OF* and *HAS-PART*) between the same two anchors across systems.

Based on such evidence, the anchors (i.e., pairs of lexically related concepts *X* and *X'*) can be classified into three main groups:

1. anchors with no structural evidence (i.e., *X* and *X'* do not share any hierarchical relationships to other anchors),
2. anchors with positive evidence, (i.e., *X* and *X'* share similar relationships to other anchors), and
3. anchors with negative evidence (i.e., *X* and *X'* share opposite relationships to other anchors).

In order to quantify the contribution of augmented knowledge to the alignment of two ontologies, we compared the number of anchors in each group before and after augmentation. Since the augmentation methods applied to the two ontologies generate additional relations, it is expected that some of these new relations provide additional structural evidence to some anchors, thus reducing the number of anchors with no structural evidence.

Results

Number of relations generated

The number of concept names exhibiting the three linguistic phenomena under investigation (reification of *PART-OF*, nominal modification, and prepositional attachment) is presented in Table 1.With the exception of nominal modification, the lexical phenomena of interest in this study were more often present in FMA than in GALEN. This is especially true for prepositional attachment. Most names in FMA are anatomical terms and a majority FMA names contain the preposition "of" (e.g., *Muscle of pelvis*, *Nail of third toe*, *Cruciate ligament of atlas*, *Base of phalanx of middle finger*, etc.). In contrast, only part of GALEN concepts are related to the anatomical domain, which may explain the lexical differences observed between the two ontologies. Because a given name may exhibit more than one lexical phenomenon, the sum of the numbers of names for each phenomenon is greater than the total number of names.

The number of relations generated by the three augmentation methods described earlier is shown in Table 2. Note that a method may extract more than two relations (direct and inverse) from a concept name. This happens when the same linguistic phenomenon is present more than once in a name (e.g., from *Base of phalanx of middle finger* (*BoPoMF*), we generate both <*BoPoMF*, *PART-OF*, *Phalanx of middle finger*> and <*BoPoMF*, *PART-OF*, *Middle finger*>, as well as their inverses). A majority of relations extracted from the concept names are also explicitly represented in GALEN, but not in FMA. Because of some redundancy between explicit and extracted relations (and, to a lesser degree, among extracted relations), the total number of relations after augmentation is less than the sum of the numbers of explicit and extracted relations.

Additional structural evidence acquired

The alignment consists of identifying equivalent concept in FMA and GALEN. These anchors are concepts present in the two ontologies exhibiting the following two properties: lexical similarity (their names are lexically equivalent) and structural similarity (they share relationships to other anchors). 2353 lexically equivalent concepts, called anchors, were identified, of which 1668 (71%) also exhibited structural similarity before augmentation techniques were applied to FMA and GALEN. This proportion rose to 87% when relationships generated through augmentations were used.

The details of the alignment before and after augmentation are presented in Table 3. The relations generated by augmentation enable 388 anchors (+16%) to acquire positive evidence. Before augmentation, there was no support for these concepts to be considered either aligned (positive evidence) or distinct (negative evidence). Anchors acquiring positive evidence after augmentation include *Ciliary gland* (the sweat gland of eyelid), which acquired through augmentation *ISA* relation to *Gland* and *PART-OF* relation to *Head*, themselves anchors.

Not surprisingly, augmented knowledge also revealed a few more conflicts across systems. For example, the two anchors *Dorsum of Foot* and *Dorsal Region of Foot* originally received positive evidence through some shared hierarchical relations. After augmentation, they acquire negative evidence because the extracted relation <*Surface of dorsum of foot*, *PART-OF*, *Dorsum of foot*> in FMA conflicts with the explicit relation <*Dorsal Region of Foot*, *HAS-PART*, *Dorsum of Foot*> in GALEN (*Surface of dorsum of foot* and *Dorsal Region of Foot* are synonymous in FMA).

Relative contribution of each method

Before augmentation, the number of anchors not supported by structural evidence was 665, i.e., 28% of the 2353 anchors. If only one method were applied, this number would decrease by about 9%, since about 200 anchors acquire evidence through reification of *PART-OF* (203) and nominal modification (201), and by 7% with propositional attachment (158). This shows the relative contribution of the three augmentation methods in providing evidence for anchors.

Finally, Table 4 simulates what would happen if augmentation methods were applied only to one system and not to the other. The alignment mostly benefited from augmenting relations in FMA. The relations required for concepts to acquire evidence were generated from concept names in FMA in 364 cases out of 388 (94%).

Discussion

Generalization. Knowledge augmentation can be applied to other subdomains of biomedicine than anatomy and can be applied beyond the biomedical domain. Because of the prominence of hierarchical relations in anatomy, this study focused on *IS-A* and *PART-OF* relations. However, associative relations could benefit from the same approach. Roles and functions are often reified (e.g., *Iron transporter*, *Calcium channel blocker*). New rules would have to be developed to target specific relations.

Likewise, depending on the context, prepositions other than "of" could be used to identify relations (e.g., *Urine test for glucose*, where the preposition "for" expresses the relationship *analyzes*). Possibly, other linguistic phenomena such as appositives could be used as well. Finally, by increasing the number of relations available, knowledge augmentation should benefit not only semantic integration, but also other approaches relying on semantic relations such as semantic interpretation.

Limitations. One obvious limitation of this study is that no validation of the 2353 anchors has been performed yet. In the absence of a gold standard resulting from such a validation, it may be difficult to evaluate the actual benefit of any method generating the relations used as structural evidence in the identification of equivalent concepts across ontologies. Since the validation of 2353 anchors represents a significant effort involving domain experts, we elected to maximize the amount of structural evidence first (e.g., through augmentation) so that it could be used by the experts in a validation environment. Nevertheless, the results of this study suggest that relations generated by augmentation only provided structural evidence to a significant number of anchors (16%). An informal evaluation conducted on a limited number of anchors showed that, in most cases, anchors supported by structural evidence denote equivalent concepts across ontologies.

Alternative approaches. Our approach to aligning ontologies relies on lexical and structural similarity. In this regard, it is close to approaches such as PROMPT [11]. However, the augmentation techniques presented here are typically not used in their alignment algorithm. A different approach to aligning FMA and GALEN has been reported by Mork & al. [12]. These authors use a generic schema matching technique. While their approach is essentially generic, and therefore virtually domain-independent, ours takes advantage of domain knowledge. The augmentation techniques described in this paper are in many cases specific to anatomy. However, we believe that this study may be an illustration of the importance of domain knowledge in alignment techniques.

Conclusions

Knowledge augmentation based on semantic relations embedded in concept names through various linguistic phenomena has proved a powerful technique, generating as many relations as are represented explicitly in FMA. Moreover, knowledge augmentation also clearly benefited the alignment of FMA and GALEN, enabling 16% more anchors to acquire evidence (mostly positive, but also negative), compared to the use of explicit relations alone.

Acknowledgements

The research was supported in part by an appointment to the National Library of Medicine Research Participation Program administrated by the Oak Ridge Institute of Science and Education through an interagency agreement between the U.S. Department of Energy and the National Library of Medicine.

Thanks for their support and encouragement to Cornelius Rosse, José Mejino, and Kurt Richards for FMA and Alan Rector, Jeremy Rogers, and Angus Roberts for GALEN. Thanks also to Pieter Zanstra at Kermanog for providing us with an extended license for the GALEN server.

References

1. Cruse DA. *Lexical semantics*: Cambridge University Press; 1986.
2. Dolan W, Vanderwende L, Richardson SD. Automatically Deriving Structured Knowledge Bases From On-Line Dictionaries. Redmond, WA: Microsoft Corporation; 1993.
3. Rindflesch TC, Hunter L, Aronson AR. Mining molecular binding terminology from biomedical text. *Proc AMIA Symp* 1999:127-31.
4. Aussenac-Gilles N, Bourigault D, Condamines A, Gros C. How can Knowledge Acquisition benefit from Terminology? *Proceedings of the 9th Knowledge Acquisition Workshop* 1995;1:11-16.
5. Bodenreider O, Burgun A, Rindflesch TC. Lexically-suggested hyponymic relations among medical terms and their representation in the UMLS. *Proceedings of TIA'2001 "Terminology and Artificial Intelligence"* 2001:11-21.
6. Burgun A, Bodenreider O, Le Duff F, Mounssouni F, Loréal O. Representation of roles in biomedical ontologies: a case study in functional genomics. *Proc AMIA Symp* 2002:86-90.
7. Rosse C, Mejino JL, Modayur BR, Jakobovits R, Hinshaw KP, Brinkley JF. Motivation and organizational principles for anatomical knowledge representation: the digital anatomist symbolic knowledge base. *J Am Med Inform Assoc* 1998;5(1):17-40.
8. Rector AL, Bechhofer S, Goble CA, Horrocks I, Nowlan WA, Solomon WD. The GRAIL concept modelling language for medical terminology. *Artif Intell Med* 1997;9(2):139-71.
9. Schulz S. Bidirectional mereological reasoning in anatomical knowledge bases. *Proc AMIA Symp* 2001:607-11.
10. Zhang S, Bodenreider O. Aligning representations of anatomy using lexical and structural methods. *Proc AMIA Symp* 2003:(to appear).
11. Noy NF, Musen MA. PROMPT: algorithm and tool for automated ontology merging and alignment. *Proc of AAAI* 2000:450-455.
12. Mork P, Pottinger R, Bernstein PA. Challenges in precisely aligning models of human anatomy using generic schema matching. *Personal Communication* 2003.

Table 1. *Number of concept names exhibiting the three linguistic phenomena under investigation (a given name may exhibit more than one lexical phenomenon)*

	FMA		GALEN	
Reification of *PART-OF*	1,618	(2%)	227	(1%)
Nominal modification	19,395	(22%)	8,282	(33%)
Prepositional attachment	53,103	(60%)	1,886	(7%)
None	23,049	(26%)	15,353	(61%)
Total (unique names)	88,108		25,192	

Table 2. *Number of relations generated by the three augmentation methods (In parentheses is the percentage of relations not present before augmentation for each linguistic phenomenon and, on the last line, the percentage of relations only generated by augmentation techniques)*

	FMA		GALEN	
Before augmentation	342,889		322,092	
Reification of *PART-OF*	215,300	(93%)	58,358	(38%)
Nominal modification	55,328	(37%)	19,732	(21%)
Prepositional attachment	145,960	(74%)	3,886	(27%)
Total (unique relations)	658,749		349,366	
From augmentation only	*315,860*	*(48%)*	*27,274*	*(8%)*

Table 3. *Repartition of the 2353 anchors by type of evidence, before and after augmentation*

Type of evidence	Before		After		Difference	
None	665	(28%)	277	(12%)	-388	(-16%)
Positive	1668	(71%)	2054	(87%)	+386	(+16%)
Negative	20	(1%)	22	(1%)	+2	(+0%)

Table 4. *Number of anchors acquiring structural evidence (positive or negative) after augmentation, by method (first applied to each ontology separately, then applied to both)*

	FMA	GALEN	Both
Reification of *PART-OF*	193	13	203
Nominal modification	183	8	201
Prepositional attachment	137	10	158
All three combined	364	26	388

Demo descriptions

Efficient development of data migration transformations

Paulo Carreira
Oblog Consulting and FCUL, Portugal
paulo.carreira@oblog.pt

Helena Galhardas
INESC-ID and IST, Portugal
hig@inesc-id.pt

1 Introduction

Nowadays, the business landscape changes very fast. Organizations merge and joint-ventures have become common headlines. This reality requires system reengineering, information integration and migration of legacy data. In this paper, we address the data migration issue.

Current data migration applications aim at converting legacy data stored in sources with a certain schema into target data sources whose schema is predefined. Organizations often buy applicational packages (like SAP, for instance) that replace existing ones (e.g., supplier management). This situation leads to data migration projects that must transform the data model underlying old applications into a new data model that supports new applications. The migration process is first exhaustively tested and then applied in a one-shot operation, usually during a weekend. The original data sources become obsolete once the migration is performed. The transformation step of the ETL (Extract-Transform-Load) process involved in large-scale data migration projects has two kinds of requirements. The first one concerns the specification of migration transformations. The second deals with the project development and management.

Several issues arise when specifying data migration transformations. First, migration programs require more powerful languages than those supported by most of the commercial ETL tools currently available. In fact, those languages are usually not powerful enough to represent the semantics of the transformation rules involved. Typically, complex transformations are handled by ad-hoc programs coded outside the tools. Second, data migration programs need more than simple programmers. People that write migration code are often business experts as well. They prefer to use high-level constructs that can be easily composed. Third, the cost involved in the production and maintainability of migration programs must be minimized. Migration code must be short, concise and easily modifiable.

Data migration projects deal with large amounts of data and potentially involve a considerable number of transformations. Therefore, data migration programs are iteratively developed. In real world projects, easy prototyping is thus an imperative requirement. Moreover, as in any other software development effort, code and data must be logically organized into distinct packages. Managing such information is crucial for the success of the initiative.

Finally, migration processes deal with critical data. This means that project auditing is frequent and strict. Auditors want to be sure that the entire set of source data is migrated, i.e., that the migration transformations cover all source records. To ensure this, they need a tool that measures the progress of the migration, and reports which source fields have been migrated and which target fields have been populated.

DATA FUSION is a data transformation platform developed and commercialized by Oblog Consulting. It addresses the requirements of generic data transformation applications. In this paper, we describe DATA FUSION DM which is the DATA FUSION data migration component that has evolved from the requirements of real data migration problems.

1.1 The DATA FUSION DM component

DATA FUSION offers a domain-specific language named DTL (standing for *Data Transformation Language*) for writing concise and short programs. It also provides an Interactive Development Environment (IDE) for efficiently producing and maintaining code. In the rest of the paper, we will focus on the DTL primitives and IDE features supported by DATA FUSION DM.

DTL provides a set of abstractions appropriate for expressing the semantics associated to data transformations. The basic concept is a *mapper* that may enclose several rules. A *rule* encloses transformations with similar logics, e.g., populate fields with the *null* value (see Section 3 for an example of a mapper). The choice of providing such domain-specific language brings several advantages. First, migration solutions can be expressed in a language close to the problem domain. Second, programs are usually concise and easy to read and maintain. Due to these two features, DTL is appropriate for easy prototyping and testing, which are major requirements of data migration applications. Third, the compiler can check if the specific vocabu-

lary is correctly used. In DTL, for example, a target attribute cannot be assigned twice. Since DTL embodies domain knowledge, a number of optimizations that could not be identified otherwise, can be introduced. Finally, a debugger facility can be developed for data migration programs. The debugger facility implementation of DATA FUSION DM is in progress.

The DATA FUSION DM IDE supports the development of data migration projects. It follows the trend of modern environments for software development (like e.g. Eclipse or Visual Studio). It includes a text editor that supports known functionalities such as syntax highlighting and code templates. Moreover, the DTL compiler is integrated within the IDE and provides helpful hints when compilation errors occur. The user can parameterize DATA FUSION DM through the IDE, in order to differentiate among production and development modes. The types of errors that are allowed when writing and testing a migration application are not the same as the ones that may occur when migrating real data.

The IDE also supports project management. First, the code produced is organized into packages according to the functionality provided. This feature is extremely important in large-scale projects as is the case of data migration. Second, the IDE provides a project tracking facility that shows to be very useful in real data migration applications. The information to migrate is precious in the sense that every source record must be migrated and every slot of the target schema must be filled in. Auditing a data migration project is a very common activity. People owning data to be migrated frequently ask for periodically checking the progress of the data migration process. The IDE reports the state of all source and target fields, i.e., the association between all target and source fields, the percentage of source and target data already migrated, etc.

1.2 Related work

The commercial ETL tools currently available usually either provide an incredible number of operators (e.g., Sagent [Sag]) for transforming data or only a small set of operators (e.g. DataJunction [Dat]). The first group of tools is not easy to use, given the large number of abstractions that the programmer must be able to handle. In the second group of tools, complex transformation logics must be developed as external ad-hoc functions through programming interfaces. This feature has several drawbacks. First, programmers must be aware of at least two programming languages: the transformation language supplied by the tool and the programming language (usually Java or C) for writing external code. Second, migration programs that handle rich transformation semantics turn to be complex and difficult to optimize. In addition, debugging of migration transformations is difficult thus delaying the data migration development cycle. The DATA FUSION DM approach defends that functions should be defined in the transformation language to allow integrated development and debugging without having to switch among development environments and tracking down bugs through archaic mechanisms.

Commercial tools provide a GUI to specify the source-target mappings, often imposing weak forms of interaction when compared with a modern programming language editor. Finally, some of these tools (e.g., Compuware FileAid/Express [Fil]) neglect the development environment in favor of a more powerful set of data transformations. The application of DATA FUSION DM to solve real world data migration problems confirmed our expectations about the IDE usefulness.

Several research data transformation tools have been proposed in the last years. Potter's Wheel [RH01] is a tool for discrepancy detection that allows the user to successively apply simple schema and data transforms. However, the class of data transformations expressible with these transformation operators is limited and does not cover all data migration requirements. The semantics of the AJAX [GFS+01] map operator is similar to the semantics of a DATA FUSION DM rule, but rules can enclose more complex logics due to the expressiveness of DTL when compared to the map let clause. Express [SHT+77] is an early prototype for data transformation. As DATA FUSION DM, it offers a language for specifying transformations of source files into target files. However, unlike DTL, it does not support recursion. Clio [MHH+01] is a tool for interactive development of schema mappings. However, the set of transformations supported is a subset of SQL. As it will be shown in Section 3, DATA FUSION DM DTL can express transformations that cannot be written in SQL.

DTL was designed for capturing the semantics of arbitrarily complex data migrations. It is a domain specific language because it is oriented to a particular problem domain [vDKV00]. Many domain-specific languages have been used over the years (SQL, ASN.1, Makefiles, among others [vDKV00]), and the subject has been receiving increased attention from the research community [Kam97]. MedMaker MSL [PGMU96] and Squirrel ISL [ZHK96] are data integration languages whose main goal is to fusion data from several sources. DTL is intended for specifying a larger class of data transformations.

DATA FUSION DM assumes that the source-target schema mappings are known. The tool does not offer any facility for discovering schema mappings as it is the case of COMA [DR02], TranScm [MZ98] and others.

2 Architecture

The DATA FUSION platform follows the client-server architecture depicted in Figure 1. On the client side,

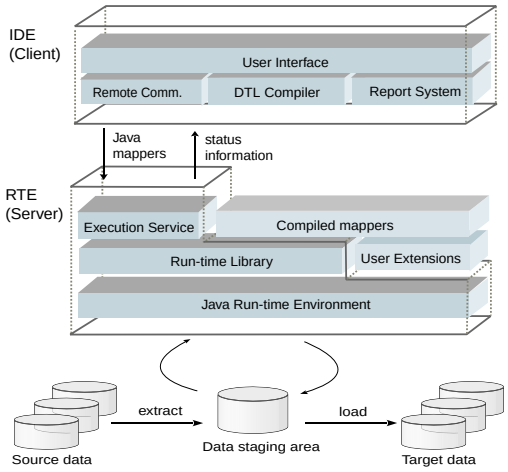


Figure 1: Architecture of DATA FUSION

the *Integrated Development Environment* (IDE) allows users to work in multiple data migration projects. On the server side, the *Run-Time Environment* (RTE) is responsible for compiling and parallelizing the data migration requests submitted from IDE instances. This client-server architecture attains scalability. An instance of the IDE may submit requests to multiple RTE instances and an instance of the RTE may run in parallel accepted submissions from multiple IDE instances.

The IDE is constituted by *(i)* the graphical user interface, which is a development environment for DTL specifications *(ii)* the remote communication subsystem in charge of submitting the compiled mappers and receiving the migration progress information, *(iii)* the DTL compiler that generates Java code from DTL mappers, and *(iv)* the report system that is responsible for displaying project tracking and auditing information.

The RTE is composed by *(i)* an execution service responsible for processing submission requests by compiling, launching and monitoring the execution of mappers, *(ii)* a run-time library that implements the semantic concepts of DTL and *(iii)* the Java Run-time Environment which is responsible for executing the Java code.

The transformations are executed by the RTE on a data staging area which can be supported by any RDBMS with a JDBC connection. Data extraction and loading are performed by third-party tools (e.g., Oracle SQL*Loader).

3 The Data Transformation Language

To motivate the unique features of the DTL language, we present a simple example which is a simplified version of typical real world problems found when migrating legacy data. The example show problems that are solved in a concise and self-contained way using DTL. To the best of our knowledge, complex restructuring sequences or manual coding would have to be used if tackled with currently available data transformation frameworks.

3.1 Migration of loan information

The source view `LOANS`, in Figure 2, stores the details of loans requested per account. The source column `ACCT` is the account number, `LOAN` is the loan number for each account and `AMT` is the amount requested. The target system does not support loan amounts superior to 100. When a loan amount greater than 100 is found in the source, it must be split into several loan payment entries in the target. In the target view `PAYMENTS`, `LOANNO` is the loan number and `AMOUNT` is the amount to be payed. The mapping requirements are as follows:

1. The column `LOANNO` is mapped by concatenating `ACCT` with `LOAN`.
2. The column `AMOUNT` is obtained by breaking down the value of `AMT` into multiple records with a maximum value of 100, in such a way that the sum of amounts for the same `LOANNO` is equal to the source amount for the same loan.

The mapper that implements these requirements is shown on the right side of Figure 2. The first requirement is implemented in the rule[1] that assigns the concatenation of the source columns `ACCT` and `LOAN` to the column `LOANNO`.

To implement the second requirement, an auxiliary variable `rec_amnt` is initialized with the value of `AMT` and is used to partition the total amount into parcels of 100. The dynamic creation of records is achieved by nesting an **insert** statement into a **while** loop. Each time an **insert** is executed, a new value for the target column is associated with the rule. Internally, values produced by the rules are represented by nodes in a graph. After executing all the rules for a source record, the values contained in the nodes are combined by a graph traversal algorithm to produce target records. In Figure 2, for each iteration of the loop, a node `AMOUNT` is loaded with 100. After the loop, an additional node `AMOUNT` is filled in with the remaining value. When both rules are executed for each source record, the values stored in node `LOANNO` and in nodes `AMOUNT` (for each `LOANNO` node, several `AMOUNT` nodes may exist) are combined to generate several records in the target view `PAYMENTS`.

The distinguishing feature illustrated by this example is as follows. The mapping logics used to load the target columns whose value is fixed (`LOANNO` in this example) is kept outside the loop. This is highly beneficial because in real-world examples, we often encounter target tables with tens of columns. By nesting

[1] No **rule** keyword is required when the rule is composed of a single statement.

LOANS

ACCT	LOAN	AMT
123	001	20,00
123	002	140,00
456	001	250,00

PAYMENTS

LOANNO	AMOUNT
123001	20,00
123002	100,00
123002	40,00
456001	100,00
456001	100,00
456001	50,00

```
mapper LoanConvert
  import master LOANS
  export PAYMENTS

  LOANNO = ACCT || LOAN

  AMOUNT = rule
    var rec_amnt: numeric
    rec_amnt = AMT
    while rec_amnt > 100 do
      @@ = 100
      rec_amnt = rec_amnt - 100
      insert
    end while
    @@ = rec_amnt
    insert
  end rule
end mapper
```

Figure 2: Specification of the LoanConvert mapper

all rules inside the loop would compromise their readability.

4 Scenario demonstrated

DATA FUSION DM has been used in real data migration projects. For example, it was applied by the Spanish software house INDRA [ind] to migrate financial data, and by Siemens to integrate three databases storing Portuguese public administration information.

In data migration projects, there is a common pattern. Proprietary applications are discontinued in favor of applicational packages which means that legacy data is migrated into a fixed target schema. The migration project that we will demonstrate intends to illustrate the generic characteristics of a data migration project driven by these requirements.

Due to confidentiality restrictions we cannot present real data used in our projects. Therefore, the scenario demonstrated is a constructed example of a banking migration. The Banking information system is composed of four applications: Clients, Accounts, Loans and Credit-cards. The data handled by these applications must be migrated into a pre-defined target schema.

With this demonstration, we want to outline the following points:

1. Complex legacy data transformations – We will illustrate a set of data migration transformations expressible in DTL that are either not tackled or are impractical in existing tools and frameworks.

2. IDE – We will show our development environment for DTL specifications (see a snapshot in Figure 3). In particular, we will present how the IDE project management handles the migration of real world financial data systems with thousands of tables.

3. Project tracking and auditing – By taking advantage of data dependency information supplied by the DTL compiler we are able: *(i)* to compute coverage metrics for source and target schema and *(ii)* to develop data dependency reports for source and target fields. We show how coverage metrics indicate the progress of rule coding. We also show how auditors take advantage of the data dependency reports to gain insight and confidence about the migration specification.

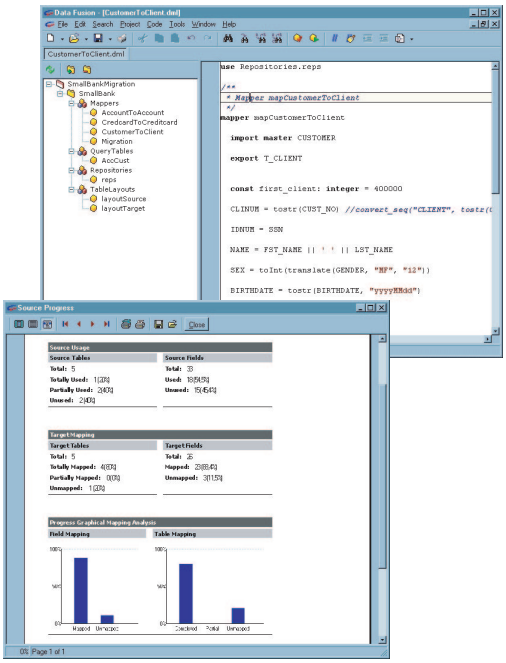


Figure 3: Snapshot of DATA FUSION IDE

References

[Dat] DataJunction. http://www.datajunction.com.

[DR02] Hong-Hai Do and Erhard Rahm. Coma – a system for flexible combination of schema matching approches. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB'02)*, Hong-Kong, August 2002.

[Fil] Compuware FileAid/Express. http://www.compuware.com/products/fileaid/express.htm.

[GFS+01] Helena Galhardas, Daniela Florescu, Dennis Shasha, Eric Simon, and Cristian-Augustin Saita. Declarative Data Cleaning: Language, Model, and Algorithms. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB'01)*, Rome, Italy, September 2001.

[ind] Indra. http://www.indra.es.

[Kam97] S. Kamin, editor. *First ACM SIGPLAN Workshop on Domain-Specific Languages (DSL'97)*, January 1997.

[MHH+01] R. J. Miller, L. M. Haas, M. Hernandéz, C. T. H. Ho, R. Fagin, and L. Popa. The Clio Project: Managing Heterogeneity. *SIGMOD Record*, 1(30), March 2001.

[MZ98] Tova Milo and S. Zhoar. Using schema matching to simplify heterogeneous data translation. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB'98)*, New York, USA, August 1998.

[PGMU96] Y. Papakonstantinou, H. Garcia-Molina, and J. Ullman. MedMaker: A Mediator System Based on Declarative Specifications. In *Proc. Int'l. Conf. on Data Engineering*, Naharia, Israel, March 1996.

[RH01] V. Raman and J. Hellerstein. Potter's Wheel: An Interactive Data Cleaning System. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB'01)*, Roma, Italy, 2001.

[Sag] Sagent. http://www.sagent.com.

[SHT+77] N. C. Shu, B. C. Housel, R. W. Taylor, S. P. Ghosh, and V. Y. Lum. EXPRESS: A Data EXtraction, Processing and REStructuring System. *ACM Transactions on Database Systems*, 2(2):134–174, June 1977.

[vDKV00] Arie van Deursen, Paul Klint, and Joost Visser. Domain-Specific Languages: An Annotated Bibliography. *SIGPLAN Notices*, 35(6):26–36, 2000.

[ZHK96] G. Zhou, R. Hull, and R. King. Generating Data Integration Mediators That Use Materialization. *Journal of Intelligent Information Systems*, 6(2/3):199–221, 1996.

Mediating Knowledge between Application Components

Monica Crubézy
Stanford University, CA 94305, USA
crubezy@smi.stanford.edu

Abstract

In such contexts as the Semantic Web, the components of an application increasingly rely on ontological models and content knowledge developed and maintained by independent contributors. These components also are designed to be building blocks of various applications. We advocate the use of a mediating component that defines and processes the knowledge transformations required to enable application components to exchange, and inter-operate on, knowledge and data. We present our approach and associated tools to support developers (1) in defining mapping relations between the ontologies involved in their application and (2) in running a mapping interpreter to mediate content knowledge and data among the corresponding ontology-based components.

1 Interoperation of Application Components

As a multi-contributor environment, the World-Wide Web fosters the formation of applications that involve multiple, distributed components. In light of the Semantic-Web approach, ontologies—models that define the concepts, properties and relations of a domain of discourse—are the communication interface (if not the backbone) of these components meant to be assembled in various applications. Increasingly, however, such ontology-based application components are contributed independently and hence cannot be expected to adhere to shared models nor to integrate with one another gracefully. Instead, different components impose different semantic, structural and syntactic views and expectations on knowledge and data, expressed by means of independent ontologies. For example in a travel-planning application, a flight-booking component would conceive travel time as the exact day and time of a flight (e.g., "Outbound on 05-01-2003 at 14h25min"), whereas a car-reservation component might only need the approximate rental period (e.g., "From Monday May 1st 2003 early evening to Sunday May 7th mid-morning plus or minus 1 day"). Such conceptual and representational mismatches need to be resolved at the ontological level in order to enable application components to exchange, and to interoperate on, a common set of data and knowledge elements.

Our solution centers around the design of a *mediating component*—one that isolates and processes the knowledge needed for configuring different knowledge-based components to work together in a particular application. This middle component encodes declarative *mapping relations* that express rules to resolve mismatches between the concepts and properties defined in the ontologies of two application components. The mediating component interprets mapping relations to transform knowledge and data from one component into knowledge and data expected by another component. We have developed associated tools for creating and processing mapping relations between any two ontologies, based on the Protégé[1] knowledge-modeling environment.

Our approach offers the advantages of maintaining the integrity of the original independent application components (hence increasing the reusability of the components in different knowledge systems) while localizing and making explicit the knowledge transformations involved in adapting the components to work together (hence reducing the effort needed to encode and modify the transformation operations). It is important to note that our solution accounts for ontology-level alignment operations as well as for content-level transformation operations. While the former is the focus of much of the current ontology-management research, the latter is more traditionally found in database integration approaches.

2 Ontology-based Mediation of Knowledge

Our approach to mediating knowledge and data between application components centers around the definition of a set of *mapping relations* that both bridge gaps between different components' ontologies and transform instance knowledge and data from one component's ontology to another. First, our solution introduces a generic ontology of the kinds of mapping relations that can be defined between the ontologies of any two application components. Instantiated mapping relations represent the correspondence links

[1] http://protege.stanford.edu

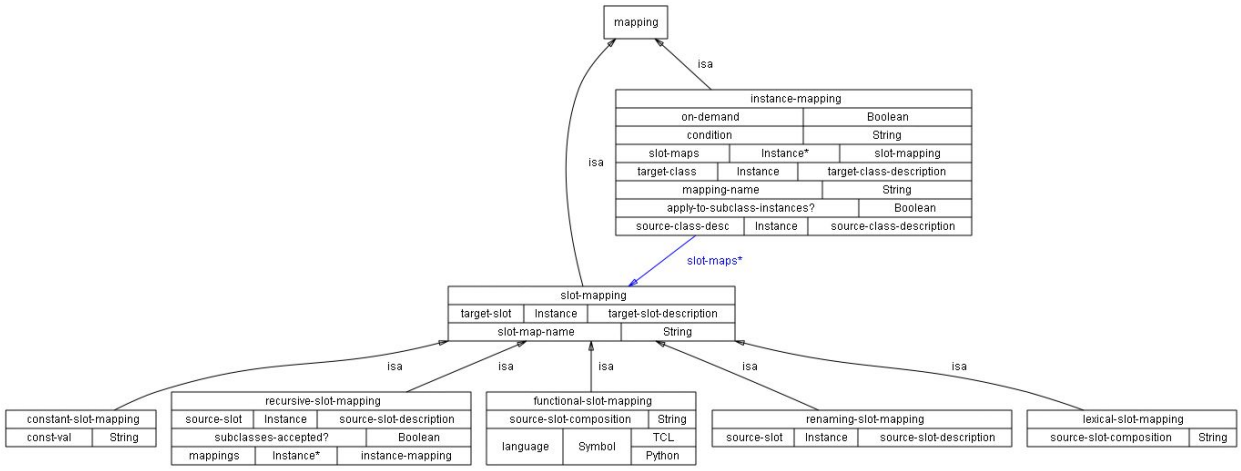


Figure 1: **Our generic ontology of mapping relations.** Each *instance-mapping* relation connects one or more source classes to one target class, and expresses how one instance of the target class is computed from each instance of the source class. Actual transformations of data and knowledge are specified in an associated set of *slot-mapping* relations that each defines the rules for computing the value of one slot of the target instance, possibly from the values of the source instance's slots. Types of slot mappings span the scope of operations that source knowledge can undergo to fit the format and semantics specified by the target ontology: from simple slot-value renaming to lexical expressions, to functional transformations. Recursive slot mappings are used for calculating instance-valued target slots, through a dependent instance mapping only processed in that context (*on-demand* flag). Finally, an instance mapping can be conditional upon properties of instances being mapped (*condition* slot), thus allowing for one-to-many instance-level mapping relations, and can be propagated to instances of subclasses of the source class (*apply-to-subclass-instances?* flag).

and transformation rules between the concepts and property values of the two components' underlying ontologies. Second, our approach includes a mapping interpreter that processes a set of mapping relations defined for two ontologies and migrates instantiated contents from one ontology to the other. Our approach is based on earlier work in our group that was aimed at studying the composition of knowledge systems from reusable domain knowledge bases and problem-solving methods [3, 4, 1].

We adopt a frame-based modeling view of ontologies. Accordingly, a set of *classes* are organized in a subsumption hierarchy to represent concepts in the domain of interest, and have *slots* attached to them to represent their properties. The values that slots can take are restricted by *facets*, such as cardinality, type and range. Classes are templates for individual *instances*, that have particular values for slots. Here, we adopt the notion of a knowledge base as an ontology populated with instances.

An Ontology of Mapping Relations

Mapping relations are defined between the ontologies of two—a *source* and a *target*—application components. Mapping relations hold the transformation operations to be applied on the source component's knowledge so that the target component is given the pieces and aspects of knowledge that it can operate on. According to a set of custom mapping relations, instances of the source component's concepts that are of interest to the target component are transformed (by a *mapping interpreter*, see below) into instances of corresponding target concepts, on which the target component is able to operate directly. Note that source and target roles for application components are dependent on the application's knowledge flow and are easily reversible.

It makes sense to categorize the types of mapping relations that can be expressed in any situation that requires mapping knowledge from one ontology to another. Such categorization allows us to conceptualize mapping relations in a better way and to design appropriate tool support for their definition and interpretation (see Section 3). We hence designed a small, generic *mapping ontology* that provides a structure for defining mapping relations between a source and a target ontology, in terms of conceptual alignment, of instance migration and of slot value computation. Figure 1 details the main aspects of our mapping ontology. To configure two components to work together in a system, a developer instantiates our mapping ontology with the set of mapping relations that link the ontologies of the source and target components. The developer thus creates a *mapping knowledge base* that contains rules to compute the target instances from source instance knowledge.

A mapping relation can be as simple as a one-to-one renaming correspondence between a source class and its slots, and a target class and its slots. In the travel-planning example, the ontologies of the flight-booking and car-reservation components might ex-

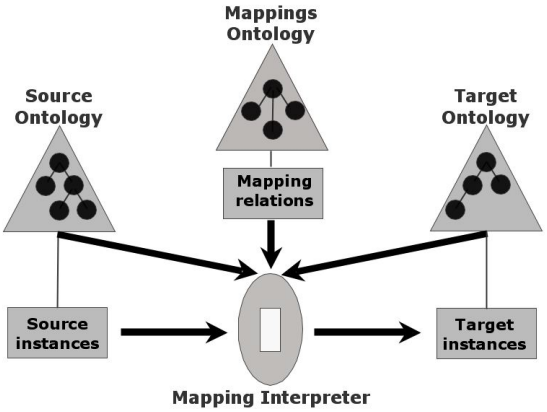


Figure 2: **Knowledge transformation and mediation performed by the mapping interpreter.**

a billing address. More complex instance-level mappings can express many-to-one, or many-to-many, aggregation relations between source and target concepts, as well as one-to-many concept-decomposition relations. Slot-level mappings also can express aggregation and decomposition operations, and include lexical, numerical and functional transformations of slot values. In the same example, the translation of the notion of time from one component to the other would involve several of those more complex operations, such as calculating a date interval from outbound and inbound flight dates, changing the date encoding, deriving approximate moments of the day from more precise flight times, etc.

Our mapping ontology provides the basis for expressing the adaptation knowledge needed to configure two components to work in a certain application. It is important to note that the core knowledge that is needed to create target instances out of source instances resides in the set of slot-level transformation operations attached to an instance-level mapping relation—operations that change the format and resolution of the source slot values to compute the required values of target slots. Eventually, a software component needs to operate on data structures that are derived from the filled-in instances of its ontology.

A Mapping Interpreter

We have developed the mapping interpreter as a piece of software associated with our mapping ontology that performs mediation of knowledge and data inside of a component-based application. As sketched on Figure 2, the mapping interpreter processes a given set of mapping relations between two ontologies—a mapping knowledge base—on a set of instances of the source ontology to produce a corresponding set of instances of the target ontology.

Specifically, in its default mode of operation the mapping interpreter cycles through all instance-mapping relations defined in the mapping knowledge base and creates one instance of the specified target class for each instance of the specified source class in a given instance mapping. The interpreter computes and fills-in the target instance's slot values according to each slot-mapping relation associated with the current instance mapping. A specific syntax that can be used in slot-value mapping expressions and in other mapping code such as conditions enables the interpreter to have local access to the source (sub-)instance's slot values. The mapping interpreter is also able to execute custom scripting and functional procedures (in TCL and Python), that provides additional mapping flexibility.

The mapping interpreter is written in Java and can be included in any component-based application. The mapping interpreter has a complete API for accessing its representation of a knowledge base (i.e., an ontology with instances). The mapping interpreter currently handles knowledge bases in the form of Protégé knowledge bases, Java collections of objects organized as in a frame-based knowledge base, and knowledge bases accessible from an OKBC [2] server. These formats can be extended with new ones.

3 Tool Support for Knowledge Mapping

We have developed an initial tool—the *Knowledge Mapping Tool*—to support an application developer in configuring ontology-based components to work together in a system, or simply to migrate knowledge from one ontology to another. Our tool is based on the Protégé knowledge-modeling environment. Ontologies have been at the heart of the Protégé methodology and tools since very early versions of the system [2]; Protégé hence is suited to provide the basis for the tool support that is necessary for mapping ontologies of application components. Protégé supports domain experts in modeling

[2] `http://www.ai.sri.com/~okbc/`

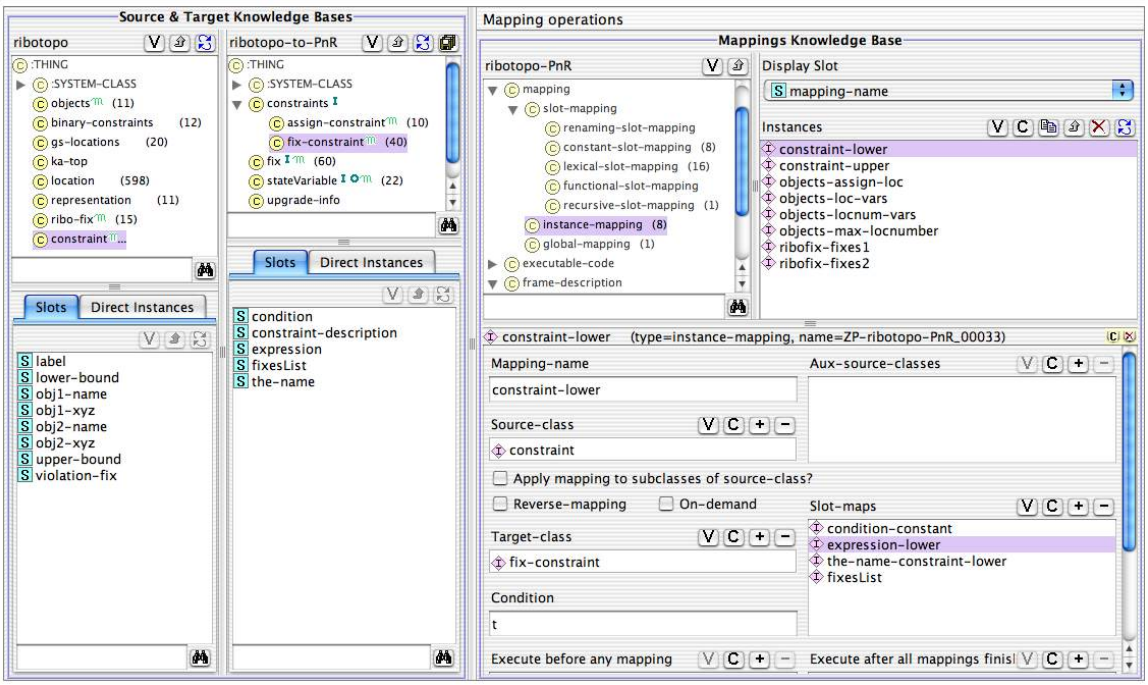


Figure 3: **Main view of the Knowledge Mapping tool.** The two left columns display, side-by-side, the source and target knowledge bases (classes in the upper panels, slots or instances in the lower panels). A small "m" icon next to a class name means that the class is part of a mapping relation with a class in the other ontology. At the right, the mapping panel displays the mapping knowledge base: At the top, the mapping ontology (left) and existing instances of mapping relations (right); below, the contents of the selected mapping relation instance, including its set of slot-level mappings. Double-arrow buttons at the top of each knowledge bases synchronizes all three panels according to the mappings defined for a selected class. For example, this screenshot shows the mapping relation "constraint-lower," from the "constraint" (source) class of the ribosome topology domain ontology and the "fix-constraint" (target) class of the propose-and-revise method ontology, for which four slot-level mapping relations have been defined to compute the values of the "condition," "expression," "the-name" and "fixesList" target slots. This example mapping relation specifies how to transform the lower bound value for the location of a ribosomal object into an actual distance-comparison expression and associated value-modification fixes to use when the expression is violated (see [1]). Note the "Mapping operations" menu at the very top of the mapping panel, that enables developers to create mapping relations from classes or slots selected in the source and target ontologies; to save the mapping knowledge base; and to run the mapping interpreter.

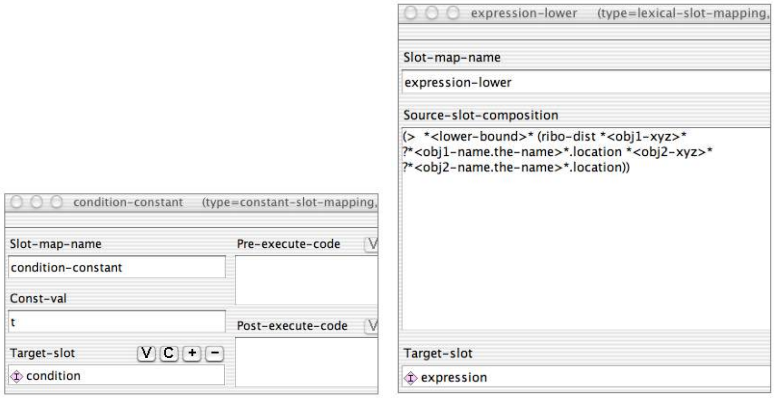


Figure 4: **Two particular slot-mapping relations**, defined in the scope of the instance mapping shown in Figure 3. (1) Left, is a simple constant slot mapping that specifies that for each instance of the target "fix-constraint" class created from an instance of the source "constraint" class, the value of the target slot "condition" should be filled-in with the value "t." (2) Right, is a lexical slot mapping that specifies a comparison predicate as value for the target "expression" slot involving the values of the source slots "lower-bound," "obj1-xyz," "obj2-xyz," "obj1-name," and "obj2-name." The $* < ... > *$ notation is used to access the actual values of the source (sub)instances' slots.

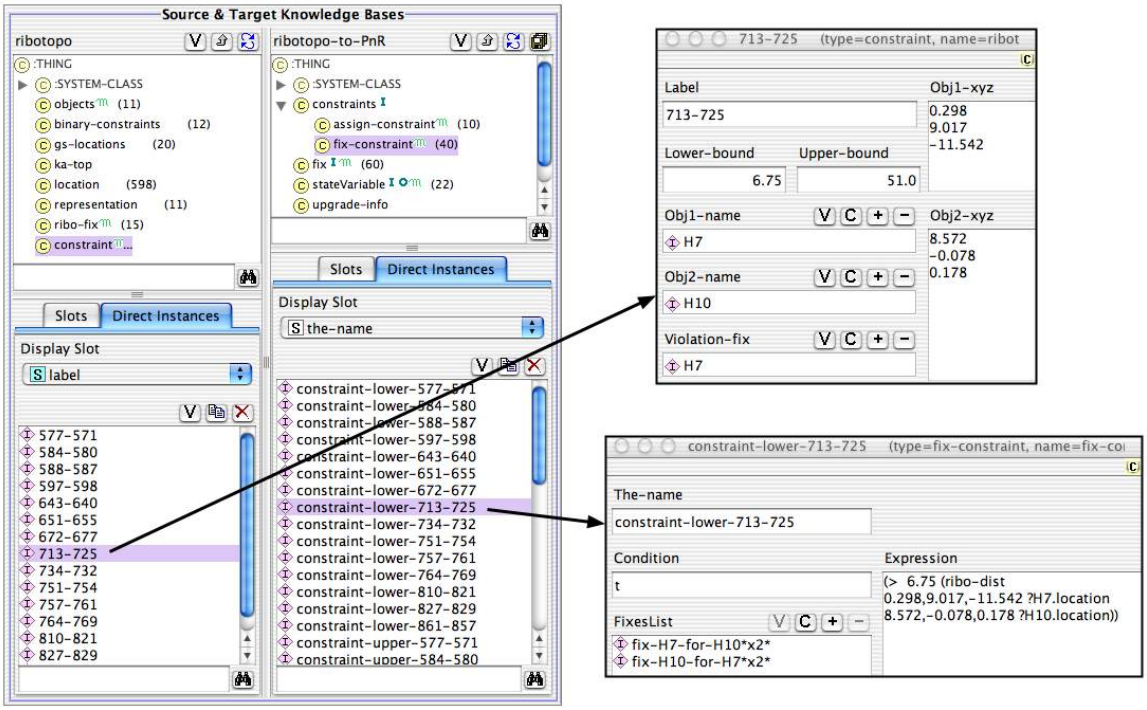


Figure 5: **Side-by-side inspection of the resulting target instances and their corresponding source instances.** As a result of running the mapping interpreter, the target ontology is populated with instances that are computed from the source ontology's instances and filled according to the set of mapping relations defined for these two ontologies. Note that instances of the target class can be the result of either instance mapping that has that class as a target class. Highlighted mapped instances are shown on the right: The contents of the instance of the source class "constraint" have been mapped partially to an instance of the target class "fix-constraint," according to the mapping relation shown in Figure 3. In particular, the value of the target slot "expression" contains the result of the lexical expression involving several source slots, as defined by the slot mapping shown in Figure 4.

relevant knowledge in an ontology and in customizing an associated knowledge-entry tool. We extended this native support of Protégé with a tool to help in creating and processing mapping relations between two ontologies. In particular, our mapping tool accesses knowledge bases from Protégé and reuses user interface elements of the base environment to provide a familiar, yet customized, interaction with system developers, as can be seen from our subsequent screen shots (Figures 3, 4 and 5).

The knowledge mapping tool allows application developers to perform the non-trivial activity of creating mapping relations between the entities of two ontologies. The tool provides a developer with an integrated and synchronized support for browsing and managing all three (source, target and mapping) ontologies involved, instead of switching manually between multiple ontology-editing windows. The mapping tool supports a developer in browsing the source classes and instances and the target classes side-by-side, and in creating or visualizing their mapping relations easily, as shown in Figure 3. A developer can populate, browse and edit the corresponding mapping knowledge base—a custom set of instances of the mapping ontology—that reflects the rules of mediating knowledge between the two ontologies.

The tool automatically creates a new mapping knowledge base for the two ontologies, or loads an existing one if available. Concretely, the developer then first creates a set of instance-level mapping relations between pairs of concepts of the two ontologies—relations that mean that for each instance of a source concept, an instance of the target concept will be created. For each instance-mapping relation between a source class and a target class, the developer also creates a set of slot-mapping relations—relations that express the way to compute the values of each slot of that target class, possibly from values of slots of that source class. The mapping tool helps the developer in making sure that all mappings are specified. The developer then can save the mapping knowledge base.

The knowledge mapping tool finally incorporates support to invoke the mapping interpreter on the three knowledge bases involved. Based on the mapping relations defined for two particular ontologies, the mapping interpreter computes a set of target instances that it fills with knowledge transformed from the source instances. After running the mapping interpreter, the knowledge mapping tool enables developers to inspect the computed instances in the newly populated target knowledge base (see Figure 5)—these instances hold the actual knowledge on which the target component will be able to operate directly.

4 Conclusion

We originally designed our solution for the task of assembling reusable domain knowledge bases with generic problem-solving methods into a working knowledge system, where a method defines a domain-independent ontology for its inputs and outputs, to be mapped to specific domain knowledge [1]. Our approach was key in assessing that PSMs and domain components could be reused in different applications. More generally, our generic mapping approach and tools are now applied to mediating knowledge and data between other kinds of knowledge-based application components in a wide range of situations. Recent applications of our tools include a high-performance architecture in which multiple public-health data sources and multiple analysis programs interact to perform syndrome-outbreak surveillance; a system for query transformation and dispatch to heterogeneous information sources; the migration of protocols from several clinical guideline and biological process formalisms to a generic workflow model. Provided adjustments of our tools to new ontology-modeling formalisms such as OWL [3], we are encouraged to believe that our approach will play a key role in Semantic-Web technology, along with ontology-management and database-integration solutions.

Acknowledgements

This research is based on seminal work by John Gennari and John Park. Recent work was joint with Zachary Pincus and was supported by the Defence Advanced Research Projects Agency.

References

[1] M. Crubézy and M. A. Musen. Ontologies in Support of Problem Solving. In Staab, S. and Studer, R., editor, *Handbook on Ontologies in Information Systems*, International Handbooks on Information Systems. Springer, In press. Also available as SMI Report SMI-2003-0957.

[2] J. H. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu. The Evolution of Protégé: An Environment for Knowledge-Based Systems Development. *International Journal of Human-Computer Studies*, 58(1):89–123, 2003.

[3] J.H. Gennari, S.W. Tu, T.E. Rothenfluh, and M.A. Musen. Mapping Domains to Methods in Support of Reuse. *International Journal of Human-Computer Studies*, 41:399–424., 1994.

[4] J.Y. Park, J.H. Gennari, and M.A. Musen. Mappings for Reuse in Knowledge-Based Systems. In *Eleventh Banff Knowledge Acquisition for Knowledge-Based Systems Workshop (KAW'98)*, Banff, Alberta, 1998.

[3] `http://www.w3.org/2001/sw/WebOnt/`

COntext INterchange (COIN) System Demonstration

Aykut Firat (aykut@mit.edu)
M. Bilal Kaleem (mbilal@mit.edu)
Philip Lee (philee@mit.edu)
Stuart Madnick (smadnick@mit.edu)
Allen Moulton (amoulton@mit.edu)
Michael Siegel (msiegel@mit.edu)
Hongwei Zhu (mrzhu@mit.edu)

MIT Sloan School of Management
30 Wadsworth Street, Cambridge, MA
02142, USA
http://context2.mit.edu/coin

Abstract. The Context Interchange (COIN) System provides tools for representing, processing, and reconciling heterogeneous data semantics. In this demonstration we show how COIN can be used to automatically resolve semantic conflicts. We demonstrate support tools for developing COIN-compatible applications and show the representation and resolution capabilities in COIN. We then show how the domain application merging capabilities in COIN allow us to rapidly develop new applications which combine the domain models, context, and elevation axioms of existing applications. This is done without rewriting existing domain knowledge. Instead a tool is used to create linking axioms. We demonstrate the construction of a Travel Agent application by merging existing airfare and car rental applications. The new application combines the strength of both application domains and resolves their semantic differences.

1 Introduction

Context Mediation technology addresses the important problem of data interpretation and deals directly with the integration of heterogeneous contexts (i.e. data meaning) in a flexible, scalable and extensible environment. The COntext INterchange (COIN) System [6] makes it easier and more transparent for receivers (e.g., applications, sensors, users) to exploit distributed sources (e.g., databases, web, information repositories, sensors). Receivers are able to specify their desired context so that there will be no uncertainty in the interpretation of the information coming from heterogeneous sources. The approach and associated tools significantly reduces the overhead involved in the integration of multiple sources and simplifies maintenance in an environment of changing source and receiver context.

An overview diagram of this approach is shown in Figure 1. The COIN project provides for a systematic representation and automatic processing of data semantics. Instead of explicitly capturing semantic conflicts, the COIN approach records data semantics declaratively and uses a mediation engine to detect all conflicts, which are reconciled by rewriting user queries to incorporate conversions that can be defined either internally or remotely on the network. This approach provides great extensibility.

We refer readers to [2,3] for a formal description of the COIN approach. The COIN framework is built on a deductive object-oriented data model where semantic data types and their properties are represented in an ontology. A modifier is a kind of property that determines how an instantiated semantic object is interpreted in different contexts. Data semantics are declared with 1) elevation axioms that map data elements to the semantic types in the ontology; and 2) context definitions that specify modifier values.

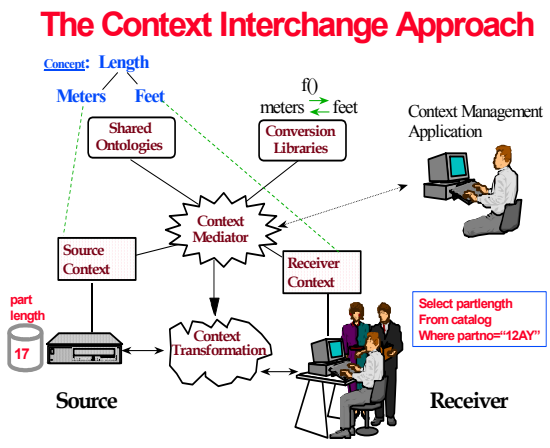


Figure 1. Context Interchange Approach

An abductive reasoning engine is used to detect semantic conflicts and rewrite the query

into one that resolves the conflicts. COIN also implements tools for authoring ontologies, interfacing with other representation (e.g., RDF, OWL)[7], specifying contexts, merging applications and domains, optimizing and executing queries.

2. Demonstration

In what follows, we will describe a COIN demonstration using several prototype applications that perform meaningful comparison of data from web sources. We demonstrate COIN, COIN development tools, and COIN application merging capabilities.

2.1 Context Mediation in a Single Application

The first part of the demonstration presents a Flight Reservation System. This application makes use of databases and web pages for flight scheduling and cost information. The sources may use differing meanings for flight information. The demonstration uses SQL as the intermediate query language and MIT's Chameleon Web extraction tool [8] to access semi-structured information from web pages. The COIN abduction engine identifies semantic conflicts through the comparison of modifier values (i.e. declarative context). Resolution of conflicts takes place automatically. The application returns a set of flight information in the context desired by the user.

The system accepts user queries in SQL, which are converted into Datalog by the query compiler. The user query is expressed as if all sources were in the user's context. The mediation engine then generates a mediated query that reconciles semantic differences, if any, between all sources involved and the user. The query optimizer and executioner [1] implements a capability aware and cost based distributed query optimization algorithm that takes advantage of parallel execution of subqueries in multiple sources.

We show the underlying components of the system including the ontology, contexts (i.e. modifiers) and elevation axioms. We then demonstrate the abduction and automatic rewriting capabilities of the system. This part of the demonstration also includes a look at the SQL interface to the set of sources of the application.

The specific application involves the aggregation of a number of web travel sources. The ontology for the Flight Reservation System is shown in Figure 2. In the ontology, semantic data types are shown in rounded boxes. Every type will be represented using primitive data types that are collectively called the *basic* type. Attributes and modifiers have these types as their domains and ranges. A context is a specification of all modifiers in the ontology. For instance, modifier *currency* may be specified to be "USD" for a U.S. context. What is included in *price* (e.g., taxes included or not) can be an ontological problem. But it is more flexible to model it as a context problem where each context is specified in modifier *type* and conversions between contexts are solved using the symbolic equation solver implemented in the mediation engine [9].

In Figure 3 we present the context issues for this demonstration. Here the user context are represented by Dora. Dora, for example, would like to see airfares that are in US dollars and that include paper ticket charges and service fees. The sources have context values (i.e., modifier values) different from Dora's. For example, TravelSelect provides airfares in British Pounds and does not include paper ticket charges or services fees.

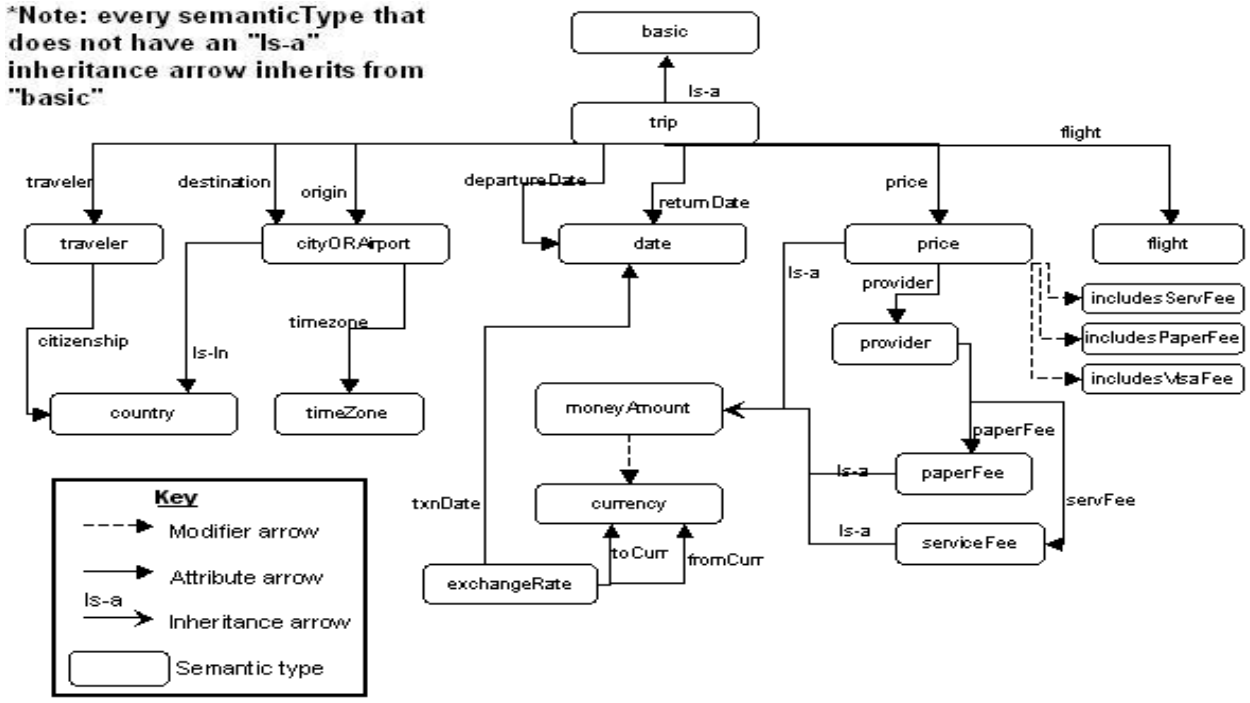


Figure 2. The Flight Reservation Ontology

Context Type	Context Name	includes-ServiceFee	includesPa-perTktCharge	currency
Receiver Contexts	Dora's Friend	No	No	GBP
	Dora	Yes	Yes	USD
Source Contexts	Yahoo	Yes	No	USD
	Expedia	Yes	No	USD
	Orbitz	No	No	USD
	Travelselect	No	No	GBP
	Itn	No	No	USD

Figure 3. Context Values for Sources and Receivers

Once the mediator has determined a conflict, conversion functions are used to provide the results in the users context. For currency conversion, we also need an internal source to generate the current date and an online source, *Olsen* at oanda.com, for exchange rate. Conversion functions for translating between the data level (*e.g.*, currency differences) and the ontological level (*e.g.*, what is included in price) are defined using rules.

Assume that Dora requests information from Travelselect and Yahoo. The conflicts are identified and the conversions, as shown in Figure 4 are executed automatically. The results are provided to Dora in her context.

Source	Conversion
Yahoo	No need to add taxes, service fees already included (do nothing), determine paper ticket charge and add it.
Travelselect	No need to add taxes, determine service fee and paper ticket charge for Orbitz and add them, convert everything from GBP to USD.

Figure 4. Conversion from Source Context to Dora's Context

A second application, for Car Rental, was developed and will be shown so that we can demonstrate the merging capabilities of COIN.

2.2 COIN Authoring Tools

We use a set of web-based authoring tools [4] to create and manage the ontology, the elevation axioms, and context definitions, which we call the knowledgebase for the application. This tool also imports RDF and exports RDF

[5,7]. By this means we can utilize ontologies developed by other applications. The tool provides both a text-based and a graphical interface. Using this tool we will demonstrate the ability to develop context knowledge, to add a new source and to modify context.

2.3 Merging Application Domains: A Travel Agent

Applications are developed in particular domains of interest. These domains are managed and used by the application using domain models (i.e., ontologies and context). It is important that the effort to develop these applications and associated domain models be reusable in other applications that may draw from one or more application domains. For this purpose we have developed technology for application domain merging. Unlike other approaches we utilize existing domain models intact. We have developed a tool that creates merging axioms that reside with the new application and operate over existing ontologies and contexts.

In the last part of the demonstration we will show the merging process. We will demonstrate a merged application (i.e., Travel Agency which includes Flight Reservation and Car Rental). This will show how new applications can be developed using existing applications covering multiple domains. The merged application provides a number of new capabilities by using the context representations and the conversion functions of the underlying applications. For example, currency as a context value is included in the airfare application but not the car rental application, in the merged application one can rent cars from agencies that price in currencies other than US dollars.

3. Summary

In this demonstration we show the capabilities of COIN for context mediation and application merging. We include a demonstration of the new symbolic equation solving [9] and multi-application merging capabilities. COIN can be used to solve a spectrum of data semantics problems, including representational, ontological and temporal semantics. We have demonstrated these capabilities in a number of application domains, such as financial services [9], online shopping [12], disaster relief efforts [4], corporate house holding knowledge engineering [11], and larger applications built by combining existing ones (*e.g.*, combine an airfare aggregator and a car rental shopper into a travel planner, see demos at our website). Efforts are also underway to use COIN framework as a cost effective alternative to standardization in the financial industry [10]. In addition, we have developed a .NET version of web wrapper and performed a preliminary study on accessing data and methods using Web Services. Progress in these areas will make COIN technology available to the Semantic Web community. Other planned extensions, such as temporal context, will further improve the applicability of COIN technology for various data integration needs.

References

1. Alatovic, T. (2002) “Capabilities Aware Planner/Optimizer/Executioner for Context Interchange Project”, MIT EECS Master’s Thesis.
2. Goh, C.H. (1997) Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems, PhD Thesis, MIT Sloan School of Management
3. Goh, C.H., Bressan, S., Madnick, S., Siegel, S. (1999) “Context Interchange: New Features and Formalisms for the Intelligent Integration of Information”, *ACM Transactions on Information Systems,* **17**(3), 270-293
4. Lee, P.W. (2003) Metadata Representation and Management for Context Mediation, MIT EECS Master’s Thesis.
5. Liburd, S. (2003) Conceptual Mapping and Structural Conversions between COIN and

RDF, technical report, MIT Sloan School of Management.
6. Madnick, S.E. (1999) “Metadata Jones and the Tower of Babel: The Challenge of Large-Scale Semantic Heterogeneity”, *Proceedings of the 1999 IEEE Meta-Data Conference*.
7. Manola, F. Miller, E. (2003) “RDF Primer”, W3C working draft.
8. Firat, A., Madnick, S., Siegel, M. (2000) “The Cameleon Web Wrapper Engine”, *Proceedings of the Workshop on Technologies for E-Services*, September 14-15, Cairo, Egypt.
9. Firat, A., Madnick, S., Grosof, B. (2002) Financial Information Integration in the Presence of Equational Ontological Conflicts, *12th WITS*, December 14-15, Barcelona, Spain.
10. Moulton A., Siegel, M.D., Madnick, S.E. (2003) Potential for Cost Savings from Application of Context Mediation Technology to Problems of Standards Interoperability in the Financial Services Industry, internal report, MIT Sloan School of Management.
11. Xian, X. (2003) Corporate Householding Knowledge Engineering and Processing using Extended COIN, MIT EECS Master’s Thesis.
12. Zhu, H. Madnick, S., Siegel, M. (2002) “Global Comparison Aggregation Services”, *1st Workshop on E-Business*, December 14-15, Barcelona, Spain.

OntoBuilder: Fully Automatic Extraction and Consolidation of Ontologies from Web Sources

Avigdor Gal
Technion - Israel Institute of Technology

Giovanni Modica
Mississippi State University

Hasan Jamil
Wayne State University

1 Introduction

Ontologies, formal specifications of domains, have evolved in recent years as a leading tool in representing and interpreting Web data. The inherent heterogeneity of Web resources, the vast amount of information on the Web, and its non-specific nature requires a semantically rich tool for extracting the essence of Web sources' content. The OntoBuilder project [10, 5] supports the extraction of ontologies from Web search interfaces, ranging from simple Search Engine forms to multiple-pages, complex reservation systems. Ontologies from similar domains are then consolidated into an ever improving single ontology with which a domain can be queried, either automatically or semi-automatically.

As an example, consider Figure 1. The figure presents partial screen shots of two forms in the domain of matchmaking. The forms require similar information to be gathered by the system, yet may use different formats to gather the information. For example, while one form asks explicitly for the "Level of Education," another form may ask for it implicitly, using a label "You are a." The similarity of the two fields can be observed only when considering the possible values to be filled. To be able to automatically match heterogeneous forms, a system must be equipped with semantic understanding of the domain, available through such ontological constructs as composition.

Given a sample form, filled by the user, and given a new form, from another Web site, OntoBuilder finds the best mapping between the two forms. This, in turn, can serve a system in automatically filling the fields (a sort of a query rewriting), according to the mapping suggested by OntoBuilder.

Unlike systems such as Protégé [4] and Lixto [2], OntoBuilder enables fully-automatic ontology matching, and therefore fall within the same category as Cupid [7] and GLUE [3]. The use of ontologies, as opposed to relational schema or XML, as an underlying data model allows a flexible representation of metadata, that can be tailored to many different types of applications. OntoBuilder contains several unique matching algorithms, that can match concepts (terms) by their data types, constraints on value assignment, and above all, the ordering of concepts within forms (termed *precedence*).

2 Overview of OntoBuilder

OntoBuilder was developed using Java, which makes it portable to various platforms and operating system environments. OntoBuilder also provides an applet version with the same features as the standalone version and the added functionality that allows users to access and use it within a Web client. The tool also runs under the Java Web Start technology. OntoBuilder generates dictionary of terms by extracting labels and field names from Web forms, and then it recognizes unique relationships among terms, and utilize them in its matching algorithms. The two types of relationships OntoBuilder is specifically equipped to deal with are composition and precedence, to be discussed in Section 2.2.

OntoBuilder is a generic tool and serves as a module for several projects, both at the Technion and at MSU. For example, we have designed a framework for evaluating automatic schema matching algorithms [1, 6], and we use OntoBuilder both for evaluation and for improving our methodology. This framework provides a sufficient condition (we term *monotonicity*) for a matching algorithm to generate "good"

Figure 1: Heterogeneous forms example

ontologies. Our empirical results with OntoBuilder show that its algorithms satisfy one of the forms of monotonicity we present in [6]. OntoBuilder is also envisioned to serve as an information integration tool in the EthoSource [8] public data repository. Finally, algorithms from OntoBuilder are being employed in an agent negotiation protocol for trading information goods.

The rest of this section presents the main features and highlights of OntoBuilder. The detailed description can be found in [10, 5, 9]. The process of ontology extraction and matching is divided into four phases, as depicted in Figure 2. The input to the system is an HTML page representing a Web site main page (*e.g.*, `http://www.avis.com`). In phase 1, the HTML page is parsed using a library for HTML/XML documents. All form elements and their labels are identified in phase 2. In phase 3, the system produces an initial version of global (target) ontology and local (candidate) ontologies. Later, in phase 4, the ontologies are matched in an iterative manner to produce a refined global ontology. We next focus on the extraction and matching processes.

2.1 Ontology extraction

Ontology extraction begins with accessing each Web source by the system browser and parsing each page into an ordered tree, called DOM tree (short for Document Object Model), which identifies page elements. This W3C standard can be used in a fairly straightforward manner to identify form elements, labels, input elements, *etc.* OntoBuilder performs suitable "cleaning" and filtering, *e.g.*, elimination of superfluous tags and removal of formatting and scripting tags, to overcome incorrect specification of the source HTML code.

The diversity of layout techniques and principles in Web design complicates the label identification process for input elements even in a well-structured DOM tree. In order to overcome this diversity we have created a set of *extraction rules*, learned from a representative set of HTML documents in different domains, to recognize an HTML page layout. The extendable set depicts all table and non-table input layouts we have encountered. Examples of input layout include text and image labels for input elements

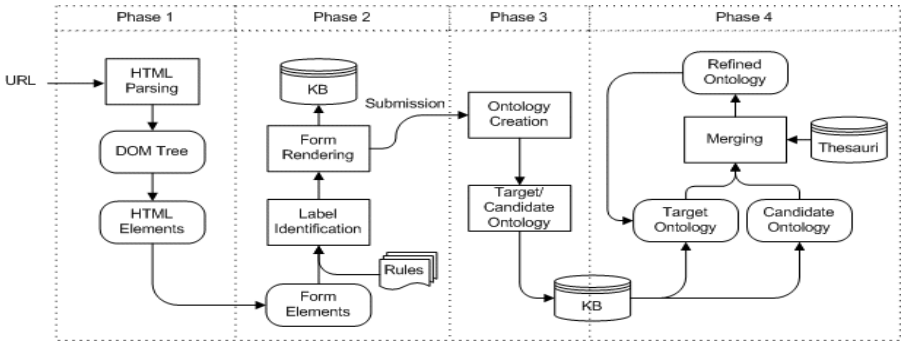


Figure 2: Ontology creation and matching process in OntoBuilder

(or forms), and table type and row type label and input field forms. The extraction process end result is an XML document containing the extracted terminology of the Web source (a dictionary).

The label identification algorithm employed in OntoBuilder is able to identify a high percentage of elements and their associated labels. Our experiments (reported in [9]) show that on average, the label identification algorithm achieves more than 90% effectiveness. These results do not include hidden fields, buttons, and images. Although these elements are used in the ontology extraction, they usually do not have an associated label (*e.g.*, hidden fields are not even shown to the user). With the release of the W3C standard for XHTML (basically well-formed HTML), Web application developers can have a solid foundation to make HTML pages easier to parse, assisting further in the task of ontology extraction. We plan on extending OntoBuilder capabilities to support XHTML as well.

Once terms are extracted, OntoBuilder analyzes the relationships among them to identify ontological structures of composition and precedence. **Composition** in Web forms is constructed through three techniques, namely *multiple term association*, *name similarity*, and *domain normalization*. *Multiple term association* involves the association of multiple terms with the same label, in which case all terms are named and grouped under that label. As an example, consider the American Airlines Web site presented in Figure 3, where the label `Departure Date:` relates to three different fields of month, day, and time. *Name similarity* groups entry labels that share identical prefix. *Domain normalization* involves the splitting of a term into subterms through recognition of known domains (such as day and time). Therefore, a time domain will be split into subterms, representing hours, minutes and AM/PM information. **Precedence** determines the order of terms in the application according to their relative order within a page and among pages. For example, car rental forms will present pickup information before return information. Also, airline reservation systems will introduce departure information before return information. It is our conjecture (supported by experiments) that precedence reflects time constraints of the application business rules and thus can be used to match better heterogeneous ontologies. For a concrete example, see Section 2.2.

2.2 Ontology matching

Ontology matching aims at refining domain information by mapping various ontologies **within the same domain**. OntoBuilder supports an array of matching and filtering algorithms. There are four main algorithms that form the core of OntoBuilder matching process, namely *word similarity, string matching, value normalization*, and *value matching*. Additional algorithms can be implemented and added to the tool as plug-ins. All algorithms are extensions of an abstract algorithm interface. The interface describes the signature (methods and functions) that matching algorithms must implement in order to be used in the tool. Algorithm parameters (such as weights) are specified using an XML configuration file which can be edited using a user-friendly interface.

Ontology matching is based on term and value matching, the former compares labels and field names using string matching, while the latter provides a measure of similarity among domains, as reflected by constrained data fields, such as drop-down lists and radio buttons. OntoBuilder provides several preprocessing techniques, based on Information Retrieval

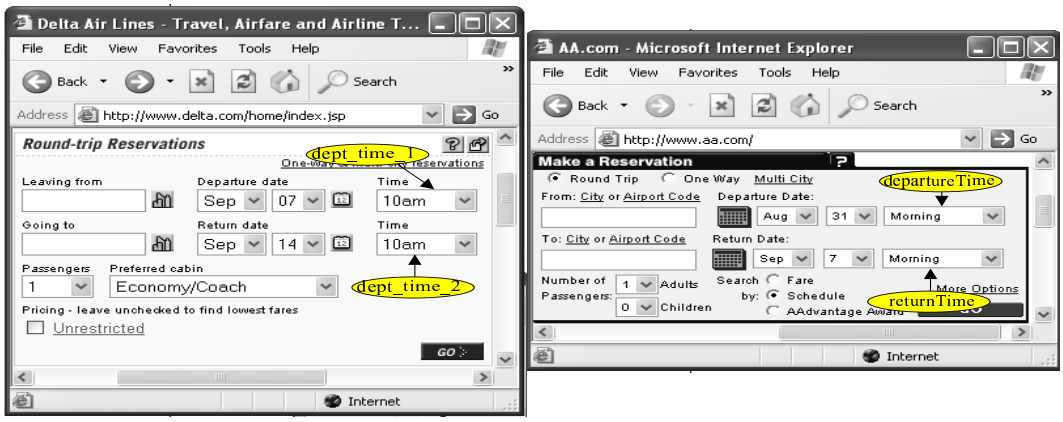


Figure 3: AA versus Delta

well-known algorithms such as *stoplists* and *dehyphenation*. It also supports automatic domain recognition and normalization to enhance the matching.

OntoBuilder employs unique algorithms for identifying structure similarity using **composition** and **precedence** constructs. Structure similarity is determined based on structure partitioning into subontologies, using terms as pivots, and comparison of subontologies. For example, using the precedence construct and two terms in two ontologies as pivots within their own ontology, OntoBuilder computes the similarity of subontologies that contain all terms that precede the pivots and also the subontologies that contain all terms that succeed the pivots (recall that Web forms enforce complete ordering of fields). A higher similarity among subontologies increases the similarity of the pivot terms themselves. This simple, yet powerful algorithm, has proven to be successful in a series of experiments performed with OntoBuilder on variety of Web sites. For example, consider Figure 3. The form of Delta airline reservation system contains two time fields, one for departure and the other for return. Due to bad design (or designer's error), the departure time entry is named `dept_time_1` while return time is named `dept_time_2`. Both terms carry an identical label, `Time`, since the context can be easily determined (by a human observer of course) from the positioning of the time entry with respect to the date entry. For American Airlines reservation system (see Figure 3 on the right), the two time fields of the latter were not labeled at all (relying on the proximity matching capabilities of an intelligent human observer), and therefore were assigned, using composition by association, with the label `Departure Date` and `Return Date`. The fields were assigned the names `departureTime` and `returnTime`. Term matching would prefer matching both `Time(dept_time_1)` and `Time(dept_time_2)` of Delta with `Return Date(returnTime)` of American Airlines (note that 'dept' and 'departure' do not match, neither as words nor as substrings). Value matching cannot differentiate the four possible combinations. Using precedence matching, OntoBuilder was able to correctly map the two time entries, since the subontologies of the predecessors of `Time(dept_time_2)` and `Return Date(returnTime)` match better than subontologies of other combinations.

2.3 Additional features

OntoBuilder supports the use of wizards, easy-to-use scripts. The *Ontology Creation Wizard* assists the user in extracting ontologies from HTML pages. The *Ontology Merging Wizard* supports the matching and merging of ontologies.

OntoBuilder provides an easy to use environment for ontology authoring. Therefore, it can be used to build ontologies from scratch or refine extracted ontologies. It also provides conversion capabilities to a variety of ontology formats, including the BizTalk schema format from Microsoft. In order to provide an intuitive interface to the user, the system implements common visualization techniques such as graph representations and hyperbolic views for ontologies, Web site maps, and document structures. Figure 4 provides a snapshot of OntoBuilder's user interface.

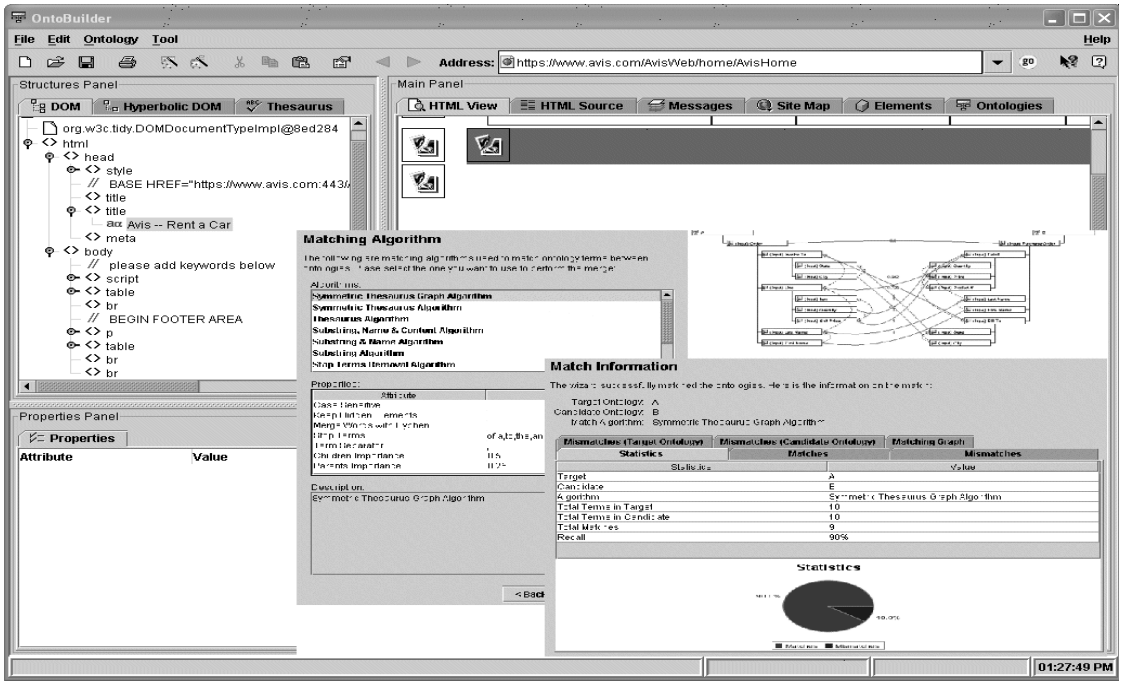


Figure 4: The OntoBuilder user interface

3 System demonstration

We will demonstrate OntoBuilder using an easy-to-follow example of matching Car rental ontologies. The system will create ontologies of car rental Web sites on-the-fly, and combine them into a global ontology. The benefits of OntoBuilder in resolving, in an automatic manner, semantic heterogeneity, including synonyms and designer errors, will be highlighted. OntoBuilder is available at `http://www.cs.msstate.edu/~gmodica/Education/OntoBuilder`.

References

[1] A. Anaby-Tavor, A. Gal, and A. Trombetta. Evaluating matching algorithms: the monotonicity principle. In S. Kambhampati and Craig A. Knoblock, editors, *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web*, pages 47–52, Acapulco, Mexico, August 2003.

[2] R. Baumgartner, S. Flesca, and G. Gottlob. Supervised wrapper generation with lixto. In *Proceedings of the International conference on very Large Data Bases (VLDB)*, pages 715–716, 2001.

[3] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the eleventh international conference on World Wide Web*, pages 662–673. ACM Press, 2002.

[4] N. Fridman Noy and M.A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 450–455, Austin, TX, 2000.

[5] A. Gal, G. Modica, and H.M. Jamil. Improving web search with automatic ontology matching. Submitted for publication. Available upon request from avigal@ie.technion.ac.il, 2003.

[6] A. Gal, A. Trombetta, A. Anaby-Tavor, and D. Montesi. A model for schema integration in heterogeneous databases. In *Proceedings of the 7th International Database Engineering and Application Symposium*, Hong Kong, China, July 2003.

[7] J. Madhavan, P.A. Bernstein, and E. Rahm. Generic schema matching with cupid. In *Proceedings of the International conference on very Large Data Bases (VLDB)*, pages 49–58, Rome, Italy, September 2001.

[8] Emilia Martins and Hasan Jamil. Ethosource. Internet Address: http://sunflower.bio.indiana.edu/~bbleakle/.

[9] G. Modica. A framework for automatic ontology generation from autonomous web applications. Master's thesis, Mississippi State University, July 2002.

[10] G. Modica, A. Gal, and H. Jamil. The use of machine-generated ontologies in dynamic information seeking. In C. Batini, F. Giunchiglia, P. Giorgini, and M. Mecella, editors, *Cooperative Information Systems, 9th International Conference, CoopIS 2001, Trento, Italy, September 5-7, 2001, Proceedings*, volume 2172 of *Lecture Notes in Computer Science*, pages 433–448. Springer, 2001.

A Collaborative Development Environment for Ontologies (CODE)

Pat Hayes, Raul Saavedra
Institute for Human & Machine Cognition
40 South Alcaniz Street
Pensacola, FL 32501
{phayes,rsaavedra}@ihmc.us

Thomas Reichherzer
Computer Science Department
Indiana University
150 S. Woodlawn Ave
Bloomington, IN 47405
treichhe@cs.indiana.edu

Submitted to Semantic Integration Workshop, October 2003, Sanibel Island.

ABSTRACT
Resolving conceptual conflicts between formalized ontologies is likely to become a major engineering problem as ontologies move into widespread use on the semantic web. We believe that in the immediate and medium-term future, conflict resolution will require the use of human collaboration, and cannot be achieved by automated methods except in simple cases. We are developing an integrated suite of software tools to provide for concept search, collaborative ontology composition and editing, and re-use of existing Web ontologies, based on the *CmapTools* collaborative Concept Map software in widespread use in education, training and knowledge capture applications. Concept maps provide a natural way to display and examine the structure of an ontology in a collaborative setting.

Keywords
Concept Maps, Collaboration, DAML, Ontology, OWL, RDF.

INTRODUCTION
Concept maps [3] or Cmaps, are a graphical representation for simple facts in the form of node-arc-node diagrams. In spite of, or perhaps because of, their simplicity, they have been used successfully in many educational settings as a technique for teaching conceptual thinking, as a knowledge-acquisition methodology and as an input modality for knowledge-acquiring software. The ease with which users of many ages, educational backgrounds and levels of technical expertise learn and use this style of representation is noteworthy; as is the fact that node-arc-node diagrams provide a natural way to display an RDF triple store, and RDF is the 'base' language of the Semantic Web architecture [3]. Noting this convergence suggested to us the possibility of using Cmap software to display and edit Web ontologies.

VIEWING ONTOLOGIES AS CONCEPT MAPS
We anticipate that the semantic web of tomorrow will provide a large set of general-purpose, domain-specific and standardized ontologies that will enable users to rapidly build their own ontologies by taking advantage of existing and agreed upon definitions of concepts and their properties. In this vision, there will be many more 'knowledge engineers' than there are at present, but this skill - which is presently considered somewhat arcane and specialized - will become simpler and more routine, rather as producing an HTML web page is now within the competence on millions of users worldwide. Nevertheless, there is clearly a need to display, search and manipulate ontologies in a form less forbidding to read than, say, RDF/XML.

The evolving W3C standards for ontology description on the Semantic Web comprises a suite of languages of increasing expressive power and complexity: RDF, RDFS, N3, DAML+OIL and OWL. All of these languages can be represented in RDF graph syntax, which consists of sets of triples of the form <subject, predicate, object>; the stronger semantic conditions of the subsequent languages can be viewed as *semantic extensions* imposed on particular RDF vocabularies (typically indicated by a URI Qname prefix, as for example in *rdfs:subClassOf* or *owl:sameAs*.) It is natural to display this syntax in a node-arc-node graphical format, which is indeed used in the specification documents themselves; and it is straightforward to apply graph-layout algorithms to generate a graphical rendering, in the form of a concept map, of any Web ontology represented as an RDF semantic extensions.

We found however that a straightforward graphical rendering of real RDF ontologies is often unreadable in practice, for a number of reasons. First, RDFS and OWL ontologies often supply full type information, which generates a lot of graphical clutter, with a few common 'type' nodes being densely linked to many other nodes in the graph; such information is useful for machine processing but redundant and distracting for human readers. Second, several of the more complex languages use constructions such as lists which are rendered into triples; these 'structural' triples tend to be distributed among the more contentful triples and fail to convey the intended meaning in a visually convincing fashion; and finally, the ontologies often contain typed literals, comment strings and other structures which are hard to follow when rendered graphically from the RDF syntax. For all these reasons, we have designed a special-purpose concept map tool which lays out Web ontologies more 'naturally', so as to reveal the essential content of the RDF graph while displaying lists as single nodes, literals and text in natural ways, and with all the 'obvious' typing information hidden from view in the graphical display. Together with several other graphical techniques, such as context-sensitive zooming, the resulting software is capable of automatically generating readable graphical layouts for DAML+OIL and OWL ontologies represented as RDF graphs containing many hundreds of concepts.

One of our main goals is to let the knowledge engineer see and edit the ontology as what the ontology really is in underlying structural terms: a graph. This ontology graph is created from the set of RDF triples defining the ontology. Now there we encounter the issue of where to place those ontology elements in space for visualization purposes. Ontology syntax and language(s) do not provide any placement information associated to any of its elements, so an automatic layout algorithm is therefore needed to place those graph elements over a canvas, and then connect those elements on the canvas to mirror the way they are connected in the ontology. The algorithm chosen was a variant of Sugiyama's algorithm for laying out hierarchical graphs [4]; this seems appropriate since many ontologies have a hierarchical subclass/type structure.

Some triples don't have to be directly represented in the Cmap. For example, comments can become just attributes of the node in the graph associated to the subject the comment applies to. In general, all objects that never appear as subjects in any triple can be hidden this way, making them attributes of the subject, without any effect on the underlying structure of the graph. "Type" triples are processed in a similar fashion. For instance, a triple of the form [A, rdf:type, T], can be also "hidden", keeping the type T information still accessible by making it just an attribute of the node A in the final graph. This reduces the graph density dramatically. If type triples are fully represented in the final graph, many of the type nodes T would become sinks of very many incoming edges, thus unnecessarily cluttering the graph, and making it harder to lay it out and visualize. We provide this hiding feature as a customizable option, so the user can choose what predicates or objects in the triples are to become attributes of the subject node, instead of separate nodes in the final graph.

Another important simplification is the replacement of all URI's and URL's with Qnames. For example, the URI http://www.w3.org/1999/02/22-rdf-syntax-ns#type gets transformed into "rdf:type". This basically reduces the length of all nodes on the final graph, making them easier to read, and also the graph becomes easier to navigate and to visualize as a whole.

After those attribute level simplifications are done on the graph, another stage in the graph transformation process identifies higher level constructs that involve several RDF triples. For example, instead of showing all the actual RDF nodes that constitute a daml list, the code collapses all those into just one special node labeled with the list of elements, and that node is connected to each of the separate element nodes contained in the list. DAML restrictions are identified and processed in a similar way.

The screen shots in Figure 1 illustrate the improvements in readability achieved by these methods. In a large Cmap the effects are even more marked. The software can also be used to edit or compose RDF ontologies by simple click-and-drag operations on the graphical display, and by select-and-paste operations on subgraphs of existing ontologies. This style of composition and editing of concept maps has proven useable by people with little technical background or special training. The 'natural' style of ontology display preserves the intuitive properties which make Cmaps useful.

Although Cmaps are used as the primary viewing and editing format for ontologies, CODE users can also view ontologies as lists of triples ordered in various ways, or as RDF/XML code; the software tracks these various views and maintains internal coherence, so that for example if a concept is found in the triples viewer – which often provides for a more rapid scan of the concepts in an ontology - then the Cmap window will be automatically centered to that place in the graph. Several studies have shown that the spatial metaphor of *location* of a node in a graph and of *navigating* through a graph is critical to the success of the Cmap user interface, particularly for large-scale graphs constructed and edited over an extended period (i.e. more than a single work session).

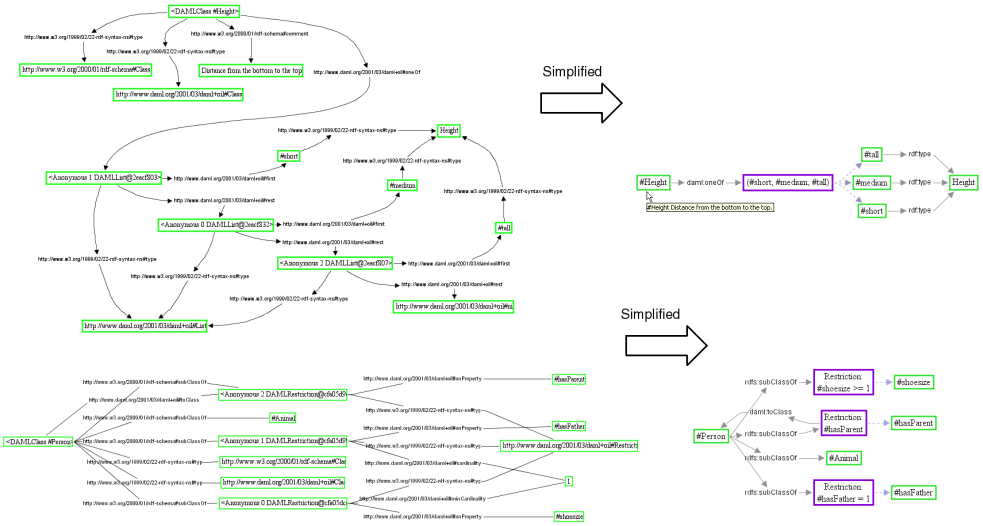


Figure 1: Original RDF graphs for Lists and DAML restrictions, and the simplified versions (right)

CMAPTOOLS AND COLLABORATIVE WORK

Our software is based on *CmapTools* [1](see also http://cmap.coginst.uwf.edu/), an integrated system for collaborative editing, storing and manipulating Cmaps which has been downloaded by users in approximately 150 countries. A similar Cmap interface based on that used by CmapTools was incorporated into the SHAKEN knowledge-entry system [2]; an application closely related to the one described here. Other applications are listed in the appendix below.

CmapTools supports collaboration among individuals at three different levels. First, users can collaborate by sharing sets of concept maps on public servers with controlled access. Second, users from different locations can start a session to synchronously edit a concept map, viewing simultaneously changes in the map as they are performed by the participants; the software is based on a comprehensive real-time agent environment which can maintain synchronicity globally, using internet protocols. And third, users can asynchronously attach notes and threads of discussion to map constituents. All of these functions adapt smoothly to the ontology tool suite we have implemented.

Among the servers in the *CmapTools* architecture, Cmap Servers are primarily responsible for making knowledge models publicly available across geographically-distant sites. They also enable users to collaborate on a knowledge model both synchronously and asynchronously. In support of asynchronous collaboration, servers facilitate discussion threads in concept maps and access control on the knowledge models. For synchronous collaboration, servers support simultaneous editing of concept maps by multiple users from different sites.

Cmap servers register themselves with a designated directory server in the *CmapTools* network that keeps track of available servers and services. Clients use the register of the directory server to present a list of places to the users where 'knowledge models' in the form of annotated Cmaps are published. While users can join the existing public *CmapTools* network with their own servers, they can also set up a private network that protects Cmap servers from becoming visible to others. Another type of server in the *CmapTools* architecture is the index server. Its purpose is to facilitate search capabilites enabling users to find resources.

Ontology Cruiser

In addition to the ontology-adapted Cmap editing and composing interface, CODE contains facilities for searching for ontologies and concepts described by existing ontologies.

To support knowledge engineers in building their own ontologies, we have developed a tool called the Ontology Cruiser that is integrated into the Cmap ontology editor. The tool provides a graphical user interface for browsing and bookmarking locally stored ontologies and ontologies from the web as well as searching for concepts within bookmarked ontologies or ontologies indexed by publicly accessible search engines. The tool's interface is designed similarly to an HTML browser, allowing users to view ontologies from the web or to manage frequently used and locally stored ontologies by means of bookmarking. However, unlike an HTML browser that enables users to view and navigate HTML pages, our focus has been to design the Cruiser such that it is particularly helpful in finding concepts within ontologies that may serve as the building blocks of new ontologies. To achieve our design goal, we focused on two aspects: (1) First, we allow knowledge engineers to browse ontologies in addition to XML text format as a set of RDF triples, or in the form of a concept map automatically generated from the original XML text. For the additional formats, we provide similar navigation and search aids as for the editor to assist knowledge engineers in dealing with large ontologies and finding concepts of interest. (2) Second, we index con-

cepts in bookmarked ontologies and provide a query interface to our locally stored index as well as public search engines for DAML and OWL. The Cruiser displays concepts that match a search query and downloads and depicts the corresponding ontology of a concept when selected in the interface. The concept index for bookmarked ontologies is kept in synch with the bookmarks to assure that concepts in user selected ontologies will be available for building new ontologies. Figure 2 shows a screenshot of the Ontology Cruiser, with two ontologies in view: the DAML'guide' ontology http://www.daml.org/2001/03/daml+oil-ex.daml as an autogenerated Cmap, and the Cyc upper ontology viewed in XML.

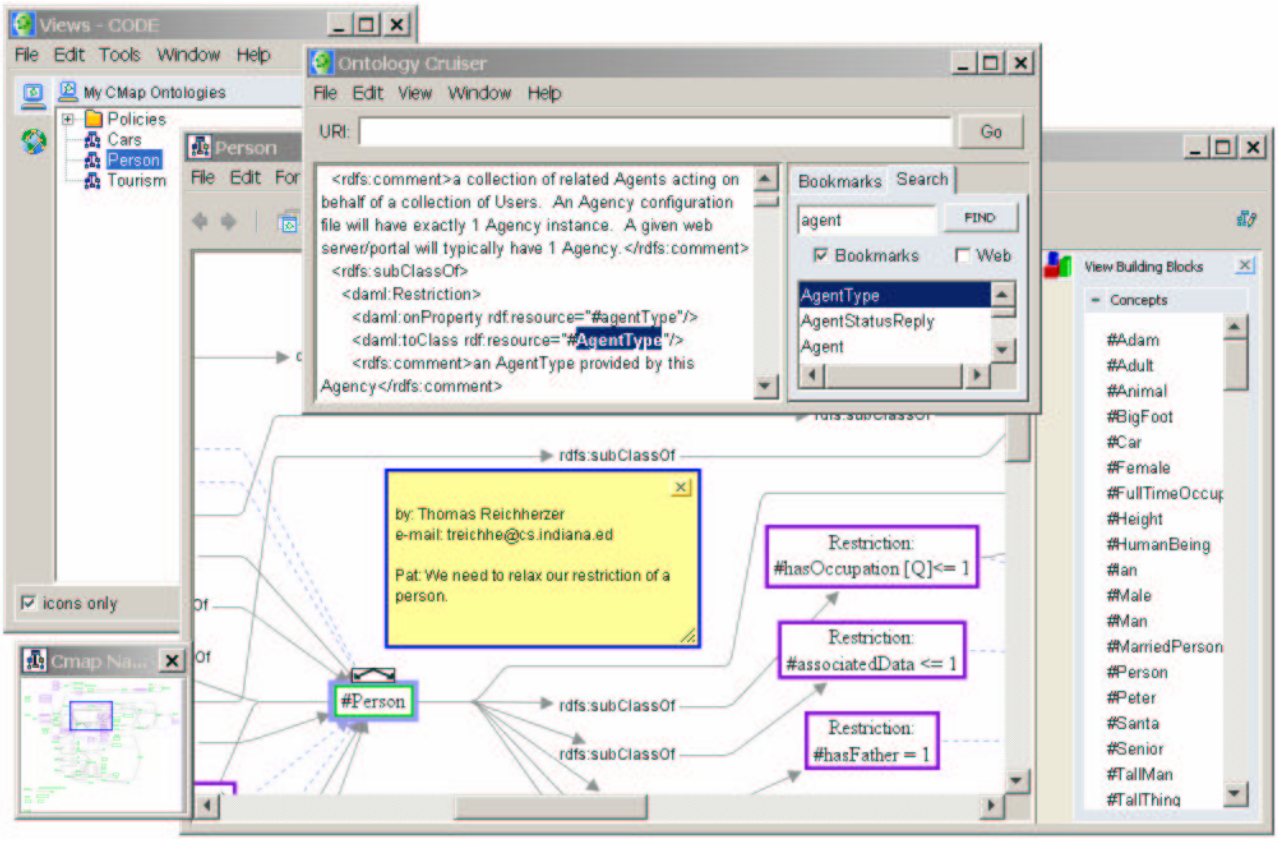


Figure 2: A Screenshot of the Collaborative Ontology Development Environment (CODE).

Building Ontologies

In our methodology for building ontologies, we envision knowledge engineers to start building new ontologies by searching through existing ontologies to collect a set of initial concepts that serve as the building blocks for the new ontology. These building blocks can be collected using the Ontology Cruiser. The Cruiser supports a simple drag-and-drop interface to drag concepts that were identified in a search into a concept map's side panel that is specifically designed for collecting concepts to be used for a new ontology. The side panel indicates graphically which concepts have been included into the new ontology. Knowledge engineers can navigate from the included concepts to the position where they occur in the concept map window by selecting the corresponding concept in the side panel. To insert a concept from the side panel, CODE supports a simple drag-and-drop interface that creates automatically a node labeled by the concept that was dragged into the map area.

CODE in Support of Collaborative Ontology Construction

Although the number of users who are actively engaged in writing Web ontologies has not yet grown to the point where one can make firm observations, we can predict several possible scenarios where concept integration issues arise and where the collaborative nature of the CODE framework could be useful.

In large-scale team efforts, the Cmap interface is of proven utility in maintaining a conceptual 'picture' of the evolving ontology over an extended development period, and the annotation and communication techniques supported by CODE (illustrated in Figure 2) can be used to focus group effort on concepts.

An 'ontology help service' could offer users attempting to write simple semantic markup detailed help by linking to the user's CODE tool, navigating and editing their ontology in real-time collaborative mode. This would enable the user to 'look over the shoulder' of the help service while the operations are being performed. Other Cmap applications have found this to be a powerful teaching and training device, particularly in concert with another communication channel (typically, a telephone contact).

The *CmapTool* software underlying CODE records every screen event in order to maintain real-time collaboration. The same technology can be used to 'record' a complete history of the ontology editing and construction process which can be replayed and analyzed off-line. We anticipate that this can provide a useful way to discover and correct conceptual errors during ontology construction.

The indexing and archiving provided by the ontology cruiser, joined to the Cmap viewing and navigation abilities, allow users to rapidly survey concepts in available ontologies and see them in context, as the example of figure 2 and to access other resources which have been attached to concepts, including comments, discussion threads, and links to other resources (such as movies, images, web pages, etc.) In this way, the Cmap tools substrate allows ontologies to be presented fully linked to a wide range of clarifying text and informative documentary supporting material. Experiments in using conventional Web search processes to discover useful concepts for informal Cmaps [6] suggest that this technique integrates well with existing Web search technology

APPENDIX

The *CmapTools* and the accompanying knowledge elicitation methodology have been used successfully for capturing, representing and sharing expertise in a variety of domains. In addition to many thousands of educational users, applications include a nuclear cardiology expert system; a prototype system to provide performance support and just-in-time training to fleet Naval electronics technicians [5]; a knowledge preservation model on launch vehicle systems integration at NASA [7], a large-scale knowledge modeling effort to demonstrate the feasibility of eliciting and representing local meteorological knowledge undertaken at the Naval Training Meteorology and Oceanographic Facility at Pensacola Naval Air Station, (http://www.coginst.uwf.edu/projects/STORMLK/) and a large multimedia knowledge model on Mars (http://www.cmex.arc.nasa.gov), constructed entirely by a NASA scientist, without the participation of knowledge engineers. Concept maps have provided a successful interface for subject matter experts to input knowledge to a computer system which automatically generates formal representations without the aid of knowledge engineers [2], see also http://www.ai.sri.com/project/SHAKEN.

ACKNOWLEDGMENTS

This research was supported in part by DARPA under contracts 2507-225-22 and 2507-221-22. We also thank the CmapTools development team for their many contributions to this project.

REFERENCES

1. Cañas, A. J., Ford, K. M., Novak, J. D., Hayes, P., Reichherzer, T., Suri, N., Online Concept Maps. The Science Teacher, 68, 4 (April 2001), 49-51.
2. Clark, P., Thompson, J., Barker, K., Porter, B., Chaudhri, V., Rodriguez, A. Thomere, J., Mishra, S., Gil, Y., Hayes, P., Reichherzer, T. (2001). Knowledge Entry as the Graphical Assembly of Components, Proceedings of the First International Conference on Knowledge Capture (K-Cap'01), pp. 22-29.
3. Novak, J. D., Gowin, D. B., Learning How to Learn. Cambridge University Press, Cambridge, UK (1984).
4. K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical systems. IEEE Trans. Syst. Man Cybern., SMC-11(2):109-125, 1981.
5. Cañas, A. J., J. W. Coffey, T. Reichherzer, N. Suri, R. Carff and G. Hill, El-Tech: A Performance Support System with Embedded Training for Electronics Technicians, Proceedings of the Eleventh Florida Artificial Intelligence Research Symposium, Sanibel Island, Florida, May 1997.
6. Cañas,A. J., M. Carvalho, M. Arguedas, Mining the Web to Suggest Concepts during Concept Mapping: Preliminary Results , XIII Simpósio Brasileiro de Informática na Educação – SBIE – UNISINOS 2002, November 2002, Brazil.
7. Coffey, J. W., R. Hoffman, A. J. Cañas and K. M. Ford, A Concept Map-Based Knowledge Modeling Approach to Expert Knowledge Sharing, IKS 2002- The IASTED International Conference on Information and Knowledge Sharing, November 2002, Virgin Islands.
8. Resource Description Framework (RDF), W3C Recommendation 22-February-1999; http://www.w3.org/TR/REC-rdf-syntax/.

SERVICE-ORIENTED ONTOLOGY MAPPING SYSTEM

Nuno Silva and João Rocha
GECAD - Knowledge Engineering and Decision Support Research Group
Instituto Superior de Engenharia do Porto
4200-072 Porto – Portugal
Nuno.Silva@dei.isep.ipp.pt

Abstract

Ontology mapping is the process whereby two ontologies are semantically related at conceptual level and the source ontology instances are transformed into target ontology entities according to those semantic relations. The objective of MAFRA–MApping FRAmework – is to cover all the phases of the ontology mapping process, including analysis, specification, representation, execution and evolution. The MAFRA Toolkit is an implementation of MAFRA, adopting an open architecture in order to observe the Semantic Web requirements, namely performance and transformation capabilities. One of the MAFRA Toolkit novelties respects its service-oriented approach, which claims that the capabilities of an ontology mapping system depend on what transformations are present. Independent, plug able services are then responsible for the instances transformations, but they also provide support for other ontology mapping tasks like automatic specification of semantic relations, negotiation and evolution. While this paper overview MAFRA Toolkit, the main contributions and novelties are the Automatic Bridging process and the Query Web Service.

1 Introduction

Ontologies, as means for conceptualizing and structuring knowledge, are seen as the key to the realization of the Semantic Web vision. Ontology allows the explicit specification of a domain of discourse, which permits to access to and reason about an agent knowledge. Ontologies raise the level of specification of knowledge, incorporating semantics into the data, and promote its exchange in an explicitly understandable form. Semantic Web and ontologies are therefore fully geared as a valuable framework for distinct applications, namely business applications like E-Commerce and B2B. However, ontologies do not overcome *per se* any interoperability problems, since it is hardly conceivable that a single ontology is applied in all kind of domains and applications. The ontology mapping aims to overcome semantic integration between ontology-based systems. According to the semantic relations (mapping relations) defined at conceptual level, source ontology instances are transformed into target ontology instances. Repositories are therefore kept separated, independent and distinct, maintaining their complete semantics and contents.

The work described in this paper has been developed in scope of MAFRA-MApping FRAmework [1]. MAFRA covers all the phases of the ontology mapping process, naming analysis, specification, representation, execution and evolution. MAFRA Toolkit is the current MAFRA implementation. It adopts a declarative specification of mappings, hiding the procedural complexity of specification and execution, while its service-oriented open architecture allows the integration of new semantic relations into the system, improving mapping capabilities as required.

This paper is organised as follow: Section 3 presents the MAFRA Toolkit service-oriented architecture. Section 4 describes the automatic semantic bridging process while section 5 describes the query web service, which allows independent agents to interoperate based on MAFRA based ontology mapping. Section 5 presents the Graphical User Interface proposed in MAFRA Toolkit. Section 6 describes related projects and compares them with this approach. In Section 7 some experiences are described, allowing a limited perspective of the capabilities of this approach. Finally, Section 8 makes a short overview of the work done so far and points out some current and future efforts.

2 Service-oriented approach

Ontology mapping aims to define semantic relations between source ontology entities and target ontology entities., which are further projected at instance level, transforming source ontology instances into target ontology instances. Semantic relations are realized through semantic bridges:

```
semanticBridge(TR, SE, TE, SC)
```

- *TR* is the process to apply in transforming instances of the source entities into instances of the target entities;
- *SE* is a subset of source ontology entities;
- *TE* is the subset of target ontology entities;
- *SC* is the set of condition expressions constraining the execution of the semantic bridge.

It is virtually impossible to provide all possible transformation requirements using a centralized static ontology mapping system. This simple observation lead to the adoption of a modular, decentralized approach, where independent transformation modules are attached to the system functional core modules (i.e. bridging, execution, negotiation, evolution, etc.) [1]. These independent transformation modules are called *Services* and provide their resources to the MAFRA core modules through the MAFRA Service Interface (Figure 1). Services are described and specified through a simple ontology, which allows MAFRA core modules to request specific features.

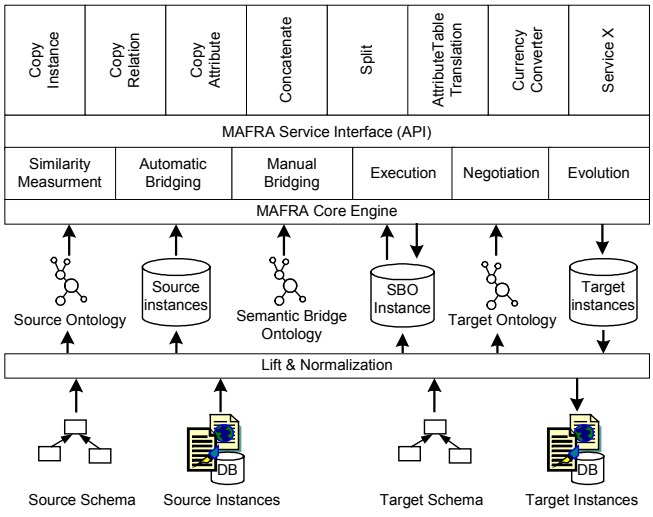


Figure 1. MAFRA Toolkit System Architecture

Simple observation shows that most of the transformation process depends upon transformation capabilities present in the system, which in turn constraint all previous phases. In fact, Services are responsible for many different tasks in the process, and not only for the transformations occurring at execution time. For example, the transformation service associated with a specific semantic bridge presumes a set of specific arguments specified in the Service description according to their characteristics (e.g. type, number, order). Recurring to this information it is possible to validate the semantic bridge arguments according to the attached Service.

The service-oriented approach suggested in this architecture advocates the need to exploit the knowledge and capabilities associated with each Service in order to increase automation and quality of the overall mapping process. Each Service interface is improved depending on the capabilities it provides to the MAFRA Core Modules. Automatic bridging and evolution are mapping phases that profit from this approach.

3 Automatic Bridging

Automatic bridging process concerns the discovery and definition of semantic bridges with minimal human intervention, based on semantic similarities between source and target ontologies entities and a set of available pool of services.

The set of semantic similarities between pairs of source and target ontologies entities play therefore a fundamental role in the discovery process. But while similarity pairs suggest that a semantic similarity exists between two entities, it says nothing about the transformation necessary to overcome the semantic heterogeneity.

It is therefore fundamental to identify the distinct sets of similarity pairs that fit in and fulfill the transformation service arguments. It is up to the service to determine the characteristics of the similarity pairs suited for its arguments. The more the service details and distinguish its arguments requirements from other services, the more perfect the automatic bridging process potentially becomes. Several and different similarity requirements are therefore required and exploited for each transformation service. Lexical and structural similarities between ontologies entities and similarities between source and target entities instances (if available) are examples of these factors. Others factors are less obvious and require deep domain expertise. In any case, similarity factors are combined into similarity pairs in the form of:

```
sp(se,te,c,v)
```

- *se* is the source ontology entity;
- *te* is the target ontology entity;
- *c* is the combination algorithm defined by the service;
- *v* is the value of the resulting similarity.

Each transformation service defines a threshold value for acceptance/rejection of similarity pairs. Currently, several independent similarity calculators are being developed and integrated into the system but for now the user manually defines the similarity factors.

To facilitate understanding the automatic bridging process described bellow, some basics on SBO [1] are necessary. Three types of semantic bridges exist:

- Concept Bridge: when the Copy Instance service is assigned. Concept Bridges form hierarchies of Concept Bridges, following the object-oriented approach, commonly applied in ontologies languages;
- Property Bridge: when any other service is assigned. These bridges are executed in scope of Concept Bridges;
- Alternative Bridges are containers for mutually exclusive semantic bridges.

Five steps compose the automatic bridging process:

1. The Similarity Inference step pre-processes similarity pairs to infer others. This situation occurs for all similarity pairs whose entities are relations (properties relating concepts). The problem arises due the fact that Property Bridges are defined in the scope of a Concept Bridge. This imply that domains and range concepts of properties are also semantic similar and further bridged. Consider `sp(name,surname,c1,x1)` a valid similarity pair for certain transformation service. The inference step determines that domain and range of *name* and *surname* are also semantic similar. If not previously defined, source concept *Person* and target concept *Employee* are stated as similar pair:

   ```
   sp(Person,Employee,c2,inferred)
   ```

   where the similarity value assumes the value *inferred*. The initial similarity pair is changed to:

   ```
   sp(Person.name,Employee,surname,c1,x1);
   ```

2. The Concept Bridge Specification step consists in pushing similarity pairs whose target entity is a concept, to the Copy Instance service (the service attached to Concept Bridges). For each valid similarity pair, a concept bridge is created. If the source entity is a property, the domain of property is bridged, and the property serves as extensional specification (see [2]);
3. The Property Bridge Specification step consists in pushing similarity pairs to each available service. The service itself accepts or rejects similarity pairs depending on its cardinality and similarity value (see example below). Only properties with the same domain concept are joined in a single Property Bridge;
4. The Inter-bridging step consists in creating the relations between semantic bridges. Two type of inter-bridges relations are defined: (i) Each Property Bridge is set in scope of Concept Bridges whose concepts are domains of the properties in the Property Bridge and (ii) Concept Bridges are set sub bridge of Concept Bridges whose concepts are the minimum super concept found;
5. The Alternative Bridge Specification step consists in setting certain group of bridges mutually exclusive. Similarity assignment strategy allows the same similarity pair to be applied in more then one semantic bridge. While not corresponding to a semantic mismatch, it is probable that the combination algorithm is not sufficiently distinctive from others. A special situation occurs when the exact same set of similarity pairs is used in more then on semantic bridge. In this case the process sets those bridges mutually exclusive. For that, an Alternative Bridge is created in the scope of the bridges and these are set as alternatives. The alternative bridge emphasizes the situation to the user, which is suggested to revise and customize the mapping resulting from the automatic process.

Example
Consider the automatic bridging process is trying to find the set of similarity pairs suited to the Copy Instance and Concatenation and Copy Attribute services. In its simplest form Copy Instance service takes one source concept and one target concept. The Concatenation service takes *n* source ontology attributes (strings) and concatenates their values into a single target ontology attribute (string), and Copy Attribute service takes one source attribute and copies its value to a single target attribute. After the similarity inference step, five valid similarity pairs exist:

```
sp(Person,Employee,c1,x1)                          (1)
sp(Person.givename,Employee.name,c2,x2)            (2)
sp(Person.surname,Employee.name,c2,x3)             (3)
sp(Person.givename,Employee.name,c3,x4)            (4)
sp(Person.surname,Employee.name,c3,x5)             (5)
```

where *c1*, *c2* and *c3* are the combination algorithms for the Copy Instance, Concatenation and Copy Attribute services respectively.
According to step 2, similarity pair (1) justifies the creation of a Concept Bridge (CB-Person-Employee). No other similarity pair is applicable in step 2, since no other concerns combination algorithm *c1*.
In step 3 each similarity pairs are forwarded to respective services. According to the Concatenation service cardinality, at least two different similarity pairs must have the same target attribute. Yet, attributes must have the same domain concept. Similarity pairs (2) and (3) respect these constraints and give raise to a new Property Bridge (PB-Concatenation-name). The Copy Attribute service requires that for each attribute, no other similarity pair exists (1:1 relation). Similarity pairs (4) and (5) do not respect the constraint and are therefore rejected.
In step 4, PB-Concatenation-name bridge is set in scope of CB-Person-Employee bridge, since Person is the domain concept of all source attributes, and Employee is the domain concept of the target attribute.
Step 5 has not effects once the previously created bridges correspond to completely different set of similarity pairs.

3.1 Bridging vs. Re-bridging

Two automatic bridging processes are available and used interchangeably according to domain expert requirements:

- Bridging process runs in scope of an empty semantic mapping. As consequence if a previous non-empty semantic mapping exists, the bridging process clears it, loosing all manual specification and customization of semantic bridges;
- Re-bridging process runs in scope of a previously existent non-empty semantic mapping. It preserves any manual modification or customization introduced by the domain expert, while encompassing changes in the semantic bridges arising from changes in the set of similarity pairs.

These two slightly different processes are necessary in order to fulfill the cyclic, iterative and interactive characteristics of the ontology mapping process advocated in scope of MAFRA.
Manual creation or deletion of semantic bridges implicitly implies changes in the set of similarity pairs. Such semantic bridges changes are not incorporated into the similarity pairs view unless requested.

4 Query Web Service

Even if ontology mapping might be applied in different contexts, our current efforts are focused in providing a functional system in the context of Semantic Web. We envisage an environment where autonomous agents need to transform excerpts of knowledge bases, according to momentary interactions with other agents. We advocate a transformation system centralized in a mediator responsible for the exchange of information between agents. Such mediator might be an autonomous entity or might be part of one of the interacting agents. Mediation process is preceded by a registration phase, concerned with the identification about each agent willing to participate in the community. In this phase, each agent provides self-identification (e.g. name, location), a set of ontologies it recognizes and a set of mappings it accepts, either as source or target agent. The query process runs according to the following algorithm:

```
query<-receiveQuery()                                  (1)
tOnto<-query.getOntology();                            (2)
mappings<-getAllMappingsForOntology(tOnto)             (3)
transf<-{}                                             (4)
tEntities<-query.getEntities()                         (5)
foreach Mapping m in mappings {                        (6)
  if(areAllEntitiesMapped(m,tEntities)){               (7)
    cbs<-m.getCBsWithEntities(tEntities)               (8)
    query<-constructQuery(m,query )                    (9)
    agents<-getRegisteredAgents(m)                    (10)
    sendTo(agents,query)                              (11)
    replies<-receiveFrom(agents,query)                (12)
    transf+=transformInst(replies,cbs)}}              (13)
reply<-runQuery(query,transf)                         (14)
sendTo(query.getAgent(),reply)                        (15)
```

The mediator receives a query from an agent (1). Accordingly to the query, the mediator identifies the ontology subjacent to the query (2 and 3) and identifies all semantic mappings related to that ontology (3). Each semantic map is then traversed in order to verify if all entities referred in the query are also mentioned in the mapping (5 and 7). If so, all concept bridges that relate each one of the target entities are identified (8). A new query is constructed, which will request all instances of all source concepts mentioned in all previous identified concept bridges (9). This new query is dispatched to all agents employing one of the mappings (10 and 11). The set of instances received from source agents (12) are then transformed through the previously identified concept bridges (13). After the cyclic process of lines (6) to (13), a set of transformed target instances is available. It is now necessary to run the original query against the set of resulting instances to filter instances accordingly to query (14). The result is finally sent to the requesting agent (15).

5 Graphical User Interface

The graphical user interface provides the domain expert with an extensive set of functionalities to specify and customize the semantic mapping and similarity pairs. The GUI evolved to a fully graphical representation of ontologies and semantic mapping (Figure 2), exploiting the graphical support available from KAON [3]. Using the same graphic approach the user is allowed to customize the mapping, matches or ontology entities. Each entity is represented using different shapes and colors.

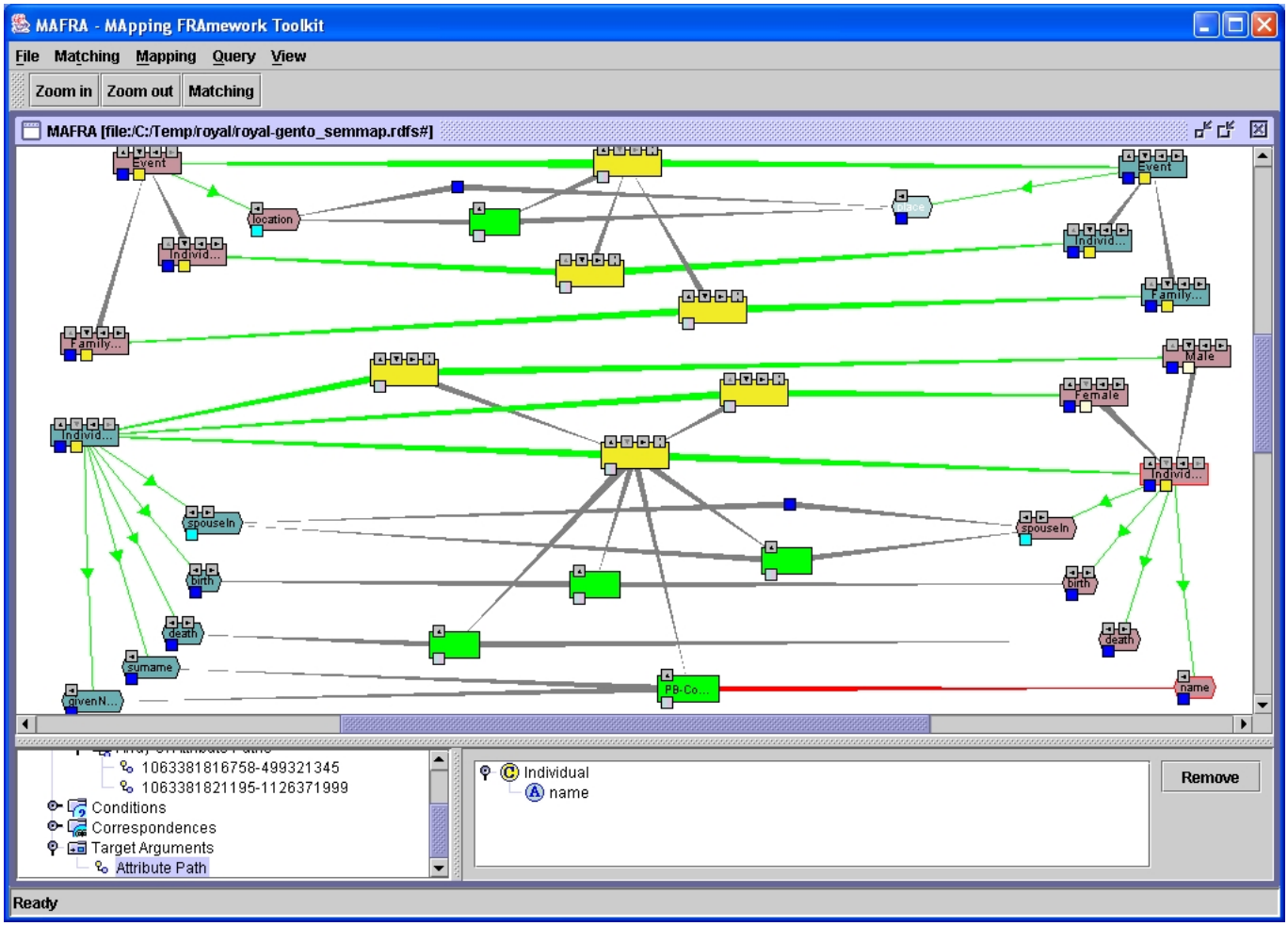


Figure 2 - MAFRA Toolkit screen-shot: specification or/and customization of semantic bridges

Ontology entities are connected to semantic bridges and similarity pairs using mouse-operated connections. Entities specific information is presented in the lower part of the window, allowing definition or customization of arguments, otherwise inaccessible. In special, it is necessary to specify transformation elements that do not exist in ontologies, like string separators, string patterns, currency converter factors, etc.

It is possible to manipulate the inter-relations of both ontologies and mapping elements, using the small colored control buttons in the border of the elements. Each button has two states (on/off) that expands or collapse the respective connections. Context menu with function like Hide, Hide Others and Fix nodes, permits to extensively define what is presented in the GUI.

6 Related Projects

Four ontology mapping projects are considered paradigmatic approaches. Park et al. [4] describes an extension to Protégé that consisted of a definition of the mapping between domain ontologies and problem solving methods. Different types of semantic relations are used depending on the complexity of the transformation, ranging from simple copy to functional transformations. The approach left several open points, especially concerning mapping between multiple concepts. Besides, there is no record of experiments that apply it to the Semantic Web environment. The second approach is RDFT [5], a meta-ontology that describes Equivalence and Versioning relations between either an XML DTD or RDFS document and another XML DTD or RDFS document. An RDFT instantiation describes the semantic relations between source and target documents, which will be further applied in the transformation of documents. Thirdly, the Buster project [6] applies information integration to the GIS domain. Two distinct approaches were proposed: rule-based transformation and reclassification. The rule-based approach applies a procedural transformation to instance properties, while classification applies class membership conditions to infer target classification through description-logic tools. However, these two approaches are not integrated, which limits mapping capabilities. The OntoMerge project [7] adopts a combination of merging and mapping techniques. The union of the two original ontologies creates the merged ontology. Elements common to both ontologies are identified and locally defined. Bridging axioms are then specified between each of these new elements and the respective elements in original ontologies. The merged ontology can be further used as any original ontology, allowing the conversion between a third ontology and the first two ontologies. This approach is based on an inference engine, which is responsible for its poor performance. The mandatory translation of ontologies and instances to and from an internal representation might also contribute to the poor performance. The great advan-

tage of this approach is the creation of a new ontology, allowing further mappings. However, the authors do not refer its usefulness and concrete application in real-world cases. How much ontologies can be merged while keep manageability, considering the poor performance of the system?

7 Experiences

MAFRA Toolkit was adopted as the development, representation and transformation engine in the Harmonise project [8]. This project intends to overcome the interoperability problems occurring between major tourism operators in Europe. Problems arise due to the use of distinct information representation languages like XML and RDF, and different business and information specifications, like SIGRT (http://www.dgturismo.pt/irt/c_pi.asp), TourinFrance (http://www.tourisme.gouv.fr) and FTB (http://www.mek.fi). Harmonise uses an "Interoperability Minimum Harmonisation Ontology" as *lingua franca* between agents. MAFRA is responsible for the acquisition, representation and execution of the ontology mapping between each agent specific ontology and IMHO. IMHO describes the tourism domain in about 64 concepts, 120 attributes and 213 inter-relations between concepts. IMHO and partner ontologies are very different. For example, the MEK ontology specifies 1 concept with 48 attributes and SIGRT defines about 50 concepts. Many different semantic and syntactic mismatches occur, but no conceptual limitations were detected in MAFRA Toolkit, and only a few refinements of the prototypal mapping relations were required.

Concerning performance issues, a very simple experience was made. Considering the lack of experience reports with ontology mapping tools, the report contained in [9] constitute a simple but valuable reference. They report the experience in transforming a dataset of 21164 instances respecting the Gedcom ontology (http://www.daml.org/2001/01/gedcom/gedcom.daml), into instances respecting the Gentology ontology (http://orlando.drc.com/daml/Ontology/Genealogy/3.1/Gentology-ont.daml). These are two very similar ontologies, whose mapping requires only simple semantic relations. The MAFRA Toolkit mapping was developed according to the semantic relations presented in the report and others gathered from the transformed data set, accessed from the web. No distinctions were detectable from both transformations. Ontologies are represented in DAML, which is not directly supported by MAFRA Toolkit. However, a representation translator from DAML to RDFS is available, which transform ontologies in a few seconds. Dataset is represented in RDF, thus excusing any transformation in MAFRA Toolkit execution. On the contrary, OntoMerge requires transformations if both ontologies and dataset. This might explain the huge difference in performance: while OntoMerge reports a 22 minutes execution time in a Pentium III at 800MHz, MAFRA Toolkit achieved the same results in less then 2 minutes in a Pentium II at 350 MHz. If a Pentium 4M 2.0Mhz is used, MAFRA requires 1 minute and 17 seconds. Unfortunately, it was not possible so far, to formally evaluate performance of the system.

8 CONCLUSIONS

This paper puts forward a new approach to ontology mapping, based on the notion of multi-dimensional service. Such services are responsible not only for the traditional instance transformation but also for other services dependent tasks like automatic bridging, negotiation and evolution. For the moment MAFRA Toolkit provides support in the four modules of the MAFRA framework: lift and normalization of source ontologies and datasets, automatic and manual specification and their representation of semantic relations, instance transformation and an easy and intuitive graphical user interface.

Currently, our efforts are focused in the evolution of the ontologies and its consequences to the ontology mapping process. It is not difficult for ontology mapping to become incoherent when a number of changes occur in mapped ontologies. The adopted service-oriented architecture provides a good starting point. A longer-term project should facilitate the mapping acquisition between different agents using meaning negotiation. This phase will also potentially benefice from the service-oriented architecture, relying on services the argumentation upon proposed semantic relations. While experiences and comparisons with other ontology mapping tools are insufficient, they showed that MAFRA Toolkit fulfils real-world requirements with a good performance.

9 Acknowledgements

This work is partially supported by the Portuguese MCT-FCT project POCTI/2001/GES/41830 under FEDER. Thanks to Jorge Santos for their comments and revisions.

10 REFERENCES

1. Maedche A., Motik, B., Silva, N., and Volz, R.; "MAFRA - A MApping FRAmework for Distributed Ontologies in the Semantic Web"; Workshop on Knowledge Transformation for the Semantic Web (KTSW 2002) at ECAI'2002; Lyon, France; 2002.
2. Silva N. and Rocha, J.; "Semantic Web Complex Ontology Mapping"; Web Intelligence 2003; Halifax, Canada; 2003.
3. Motik B., Maedche, A., and Volz, R.; "A Conceptual Modeling Approach for building semantics-driven enterprise applications"; ODBASE-2002; California, USA; 2002.
4. Park J. Y., Gennari, J. H., and Musen, M. A.; "Mappings for Reuse in Knowledge-based Systems"; 11th Workshop on Knowledge Acquisition, Modelling and Management (KAW 98); Banff, Canada; 1998.
5. Omelayenko B.; "RDFT: A Mapping Meta-Ontology for Business Integration"; Workshop on Knowledge Transformation for the Semantic Web (KTSW 2002) at ECAI'2002; Lyon, France; 2002.
6. Stuckenschmidt H. and Wache, H.; "Context Modeling and Transformation for Semantic Interoperability"; Workshop "Knowledge Representation meets Databases" (KRDB 2000) at ECAI'2000; Berlin, Germany; 2000.
7. Dou D., McDermott, D., and Qi, P.; "Ontology translation by ontology merging and automated reasoning"; EKAW Workshop on Ontologies for Multi-Agent Systems; Sigüenza, Spain; 2002.
8. Harmonise; IMHO; http://www.harmonise.org; 2003.
9. Dou D., McDermott, D., and Qi, P.; "Ontology translation on the semantic web"; ODBASE-2003; Catania (Sicily), Italy; 2003.

Resolving Schema and Value Heterogeneities for XML Web Querying

Nancy Wiegand and Naijun Zhou
University of Wisconsin
550 Babcock Drive
Madison, WI 53706
wiegand@cs.wisc.edu, nzhou@wisc.edu

Isabel F. Cruz and William Sunna
Computer Science Department
University of Illinois at Chicago
Chicago, Illinois 60607
ifc@cs.uic.edu, wsunna@cs.uic.edu

Abstract

To query XML data over the Web, query engines need to be able to resolve semantic differences between heterogeneous attributes that are conceptually similar. This demo presents a mapping tool and method to resolve semantic heterogeneity at the schema and value levels for data sets that are part of a Web-based information system. The mapping tool automatically produces agreement files. We enhanced a base prototype XML Web query system to include an ontology subsystem that generates subqueries using the agreement information. Other contributions include the use of minimal metadata to locate data sets, a formal language construct to support query re-write called a GeoSpace, and post-query aggregate statistics and spatial display.

1. Introduction

Semantic interoperability is necessary for querying distributed data over the Web. Our work is motivated by a proposed Wisconsin statewide land information system that will be a Web-based resource for local, regional, and state data (WLIS) [14]. We extend the clearinghouse vision of the original WLIS working group by incorporating DBMS-type querying over the distributed and highly heterogeneous data sets.

We illustrate our work by integrating and querying data containing land use codes. Land use data is an important component of WLIS because of its value for comprehensive planning decisions. However, land use codes are extremely heterogeneous because there is no standard code system and jurisdictions adapt code systems to emphasize their predominant types of land use.

Although we use land use data in this demo, our method is not limited to that theme. Our framework of semantic mapping and query rewrite can resolve any schema and value level differences. We particularly address the problem of values from heterogeneous domains that cannot be resolved in a straightforward manner. For example, although values in different units of measure can be easily converted, land use values cannot be resolved using a formula.

Related work has resolved heterogeneous schemas at the attribute level, e.g., [1] but has not addressed more complex value level differences. In our work, we demonstrate a method that captures mapping cardinalities and nuances of meaning at the value level.

2. The Semantic Problem

As stated, in addition to schema level mapping, we focus on a type of semantic problem in which formulas or algorithms cannot be used to resolve value level differences between conceptually related attributes in different data sets. We use land use coding systems as an example value domain [11, 13]. Land use coding systems vary by almost every jurisdiction that produces land use data. Example differences in levels of detail and semantics for residential codes between two counties are illustrated in Table 1. As can be seen, categories do not match between code systems.

Table 1. Example Coding Systems

Dane County	Racine County
111 Single Family	111 Single-Family
113 Two Family	120 Two-Family
115 Multiple Family	141 Multi-Family Low Rise (1-3 stories)
129 Group Quarters	142 Multi-Family High Rise (4 or more stories)
140 Mobile Home	150 Mobile Homes
142 Mobile Home Park	199 Residential Land Under Development
116 Farm Unit	
190 Seasonal Residence	

3. Method

The following subsections explain our ontology-enhanced XML query system shown in Figure 1.

3.1 Internet XML DBMS

To provide DBMS-type querying over distributed WLIS data, we use the Niagara Internet XML DBMS [9] as a base for our system. Niagara satisfies the need for general purpose querying over distributed XML data on the Web. However, Niagara does not have semantic integration facilities. To incorporate semantic integration, we modified the Niagara Java source code by adding an ontology subsystem to intercept queries (Figure 1). The ontology subsystem consults metadata indexes and ontology mappings to produce subqueries in local terms.

Our application has a type of query not found in conventional database usage. That is, to accommodate comprehensive or regional decision-making, a typical type of query has a common predicate applied over multiple geographic areas or jurisdictions. An example query for comprehensive land use planning is "*Find all the agricultural lands in Dane and Racine counties*." We call this type of query a GeoQuery because it covers a geographic area.

Niagara's "IN *" capability to range over all elements satisfying a predicate cannot be used here, even if the entire geographic area were specified, because of the heterogeneity of land use data created by independent agencies. Instead, this type of query must be intercepted by our subsystem which generates subqueries for each appropriate data set using semantic mappings.

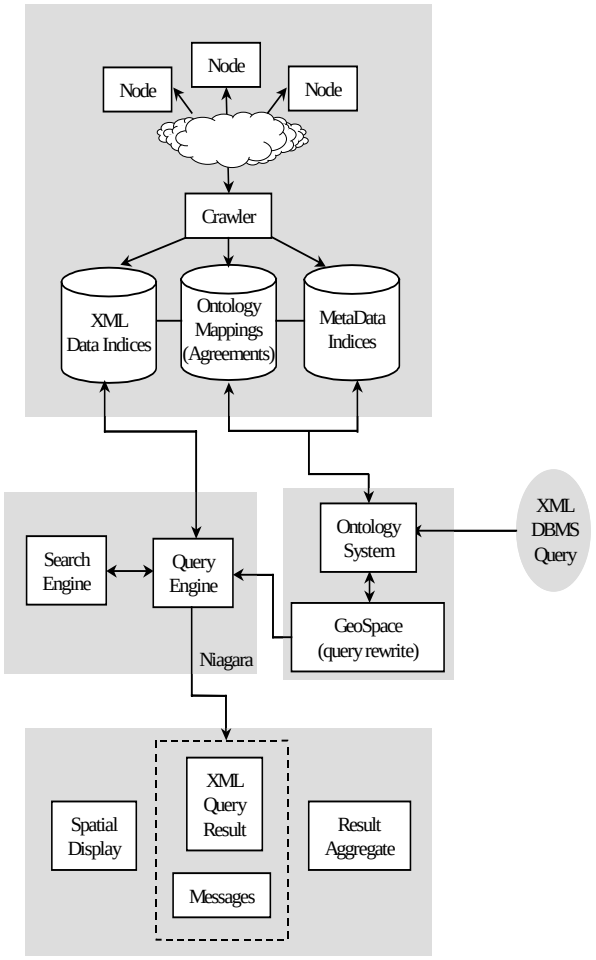


Figure 1. System Architecture

3.2 Ontology Mapping

To solve the heterogeneity problem, we developed an ontology of attributes in the land use theme and a subontology of values for the land use attribute, in particular. The ontologies can be considered to be master sets of terms from which a user can pose a land use query. We developed a tool in which a domain expert indicates schema and value level mappings between the master ontologies and each local data set (Figure 2). At the value level, our method captures the cardinality of the mapping between the ontology value and the local code. The domain expert can specify one to one, one to many, many to one, or one to null mappings. An example of each type of mapping is shown in Table 2.

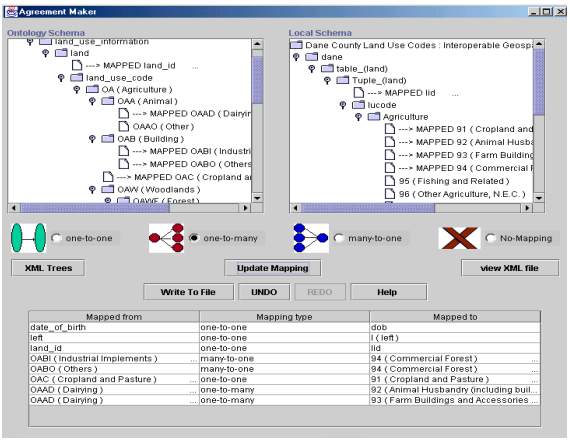


Figure 2. Tool to Create an Agreement File

The mapping tool automatically generates an XML agreement file (Figure 3). As can be seen, semantic information describing the mapping is expressed using the extensibility provided in XML tags. Furthermore, as an option, one to null mappings can be resolved. For example, Table 2 and Figure 3 show a detailed ontology code (multi-family) being resolved to a code at the next level up (residential) for a particular local code set. A complete description of agreement files is given in [2, 3].

Table 2. Value Level Semantic Mappings from the Ontology Codes to Local Codes

Mapping Cardinality	Ontology	Local Code
1 : 1	Cropland	Cropland
1 : N	Agriculture	Cropland, Pasture, etc.
N : 1	Plastics Rubber	Plastics & Rubber
1 : Null (*up a level*)	Multi-family	Residential (*resolved*)
1 : Null (*broader code at same level*)	Commercial Forest	Forest-Other (*resolved*)

```
<Ontology_to_localcode  value = "Agriculture">
   <mapping> one-to-many </mapping>
      <part> cropland </part>
      <part> pasture </part>
       …
</ Ontology_to_localcode >

<Ontology_to_localcode  value = "Multi-Family">
   <mapping> one-to-null </mapping>
      <level_up> Residential </level_up>
</ Ontology_to_localcode >
```

Figure 3. Part of an XML Agreement File

3.3 GeoSpace

To formally represent a GeoQuery, we developed a GeoSpace statement [12] for the XML-QL [4] query language (Figure 4). The GeoSpace statement has a variable that holds the list of URLs needed in the query. The variable also serves as a qualifier for the generic ontology terms in the formal expression of the query.

```
GEOSPACE Area = "www.co.wi.us/EauClaire.xml,
                 www.co.wi.us/Racine.xml"
WHERE <$*>
        <Area:LandUseCode> "agriculture" </>
        </> ELEMENT_AS $a
CONSTRUCT $a
```

Figure 4. GeoSpace Added to XML-QL

To send this query into the XML query engine, the ontology subsystem consults the agreement files to rewrite the formal query into multiple subqueries expressed in native terms. For example, the subquery pertaining to Eau Claire County is shown in Figure 5.

```
WHERE <$*>
        <Lu1> "AA" </Lu1>
        </> ELEMENT_AS $a
IN  www.co.wi.us/EauClaire.xml
CONSTRUCT $a
```

Figure 5. A Generated Subquery

3.4 Other Enhancements

We made further changes to the base XML query system to accommodate heterogeneous geospatial data.

3.4.1 Metadata Indexes

In an information system such as WLIS, users tend to select data sets for queries based on theme and either jurisdiction or spatial extent**.** To identify data sources, we created metadata files for each data set. Our minimal criteria include theme (e.g., land use), jurisdiction type (e.g., city), and jurisdiction name. This information is indexed in separate metadata indexes (Figure 1).

3.4.2 User Interface

Our user interface, shown in Figure 6, is designed to capture the minimal metadata needed

to locate data sources. We also include a spatial ability such that the user can click on a county on a map.

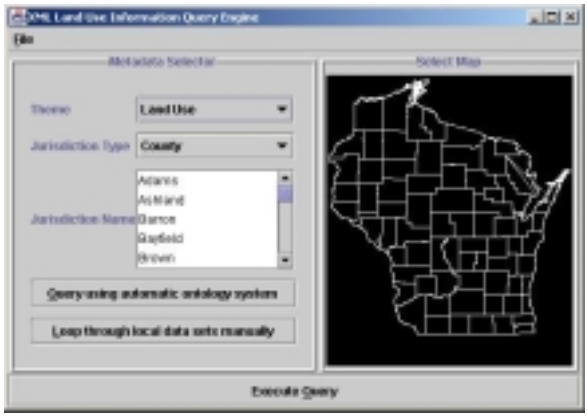


Figure 6. User Interface

3.4.2 User Output--Maps and Messages

Our test data is derived from ArcView [5] files which contain spatial coordinates in addition to nonspatial attribute tables. For each ArcView data set, we combined the spatial and nonspatial information into the same XML file using a feature-based approach. From the spatial data in the query results, MapObjects [5] was used to create spatial displays (Figure 7).

For each data set, we also output semantic information between the ontology code selected in the query and the mapped local code(s) so the user is informed of superset, subset, or resolved null mappings being returned.

Finally, because our potential users almost always asked for displays involving various summary statistics, we processed the client-side results to produce summary information, including sorted results, averages, and counts. For example, the total and average areas coded as agriculture within each jurisdiction are displayed.

Figure 7. Spatial Display

4. Related Work

Ontologies are now being used as a solution for semantic integration [6]. However, most work on ontologies has focused on the schema level and not the value level. Automatically creating ontologies is being explored, for example, in [7]. A use of ontologies in query processing can be found in [10] in which semantic similarities are obtained from a WordNet graph. They also introduced a similarity operator into an XML language. In our application, however, the land use code mappings cannot be found in a collection such as WordNet. Also, we need to hold precise semantic nuance information instead of retrieval relevance rankings. As a result, we needed to develop an automatic or semi-automatic ontology mapping method. Clio [8] represents related work in mapping but is a tool for mapping at the schema level.

5. Demo Description Summary

We are demonstrating a semantic integration query system for heterogeneous data that is built by enhancing an XML Internet DBMS. Our demo has two parts. One part is a tool used to create mappings between ontologies and local data sets (Figure 2). This tool also automatically creates XML agreement files.

The other part of the demo is the overall enhanced XML query system that uses Niagara [9] as a base. Our enhanced interface allows a user to select minimal metadata to determine relevant data sets and themes. The user then uses the appropriate ontology values to pose a query.

The type of query we address is one with a common predicate ranging over multiple data sets. This is typical for a comprehensive planning query that covers a geographic area. Our automated ontology subsystem resolves this type of query (GeoQuery) by generating specific local subqueries using lookups on the agreement mappings and metadata indexes. We formalized a representation of a GeoQuery by introducing a GeoSpace statement into an XML query language. Finally, we process client-side results to create aggregate statistics and spatial displays.

6. Acknowledgement

This work was supported by the Digital Government Program of NSF, Grant No. 091489.

7. References

[1] Bouguettaya, A.; Benatallah, B.; and Elmagarmid, A. Interconnecting Heterogeneous Information Systems. Kluwer Academic Publishers, 1998.

[2] Cruz, I. F. and Rajendran, A. "Semantic Data Integration in Hierarchical Domains", IEEE Intelligent Systems, Vol. 18, No. 2, March-April 2003, pp. 66-73.

[3] Cruz, I.F.; Rajendran, A.; Sunna, W.; and Wiegand, N. "Handling Semantic Heterogeneities Using Declarative Agreements", In Proceedings of ACM GIS, November 2002, pp. 168-174.

[4] Deutsch, A.; Fernandez, M.; Florescu, D.; Levy, A.; and Suciu, D. "XML-QL: A Query Language for XML", 1998, http://www.w3.org/TR/NOTE-xml-ql/.

[5] Environmental Systems Research Institute (ESRI), http://www.esri.com.

[6] Fensel, D. Ontologies: Silver Bullet for Knowledge Management and Electronic Commerce. Springer-Verlag, Berlin, 2001.

[7] Malyankar, R. "Vocabulary Development for Markup Languages – A Case Study with Maritime Information", WWW2002, Honolulu, Hawaii, May 2002, pp. 674-685.

[8] Miller, R.; Hernández, M.A.; Haas, L.M.; Yan, L.; Ho, C.T.H.; Fagin, F.; and Popa, L. "The Clio Project: Managing Heterogeneity", ACM SIGMOD Record, 2001, pp. 78-83.

[9] Naughton, J.; DeWitt, D.; Maier, D.; and others, "The Niagara Internet Query System", IEEE Data Engineering Bulletin, 24(2), 2001, pp. 27-33.

[10] Theobald, A. and Weikum, G. "The Index-Based XXL Search Engine for Querying XML Data with Relevance Ranking", In Proceedings of EDBT 2000, Jensen, C.S. (ed.), pp. 477-495.

[11] Wiegand, N.; Zhou, N.; Cruz, I.F.; and Rajendran, A. "Querying Heterogeneous GIS Land Use Data Over the Web", In Proceedings of GIScience 2002 Abstracts, Egenhofer, M. and Mark D. (eds.), Boulder CO, September 2002, pp. 207-210.

[12] Wiegand, N.; Zhou, N.; Patterson, E.D.; and Ventura, S. "A Domainspace Concept for Semantic Integration in a Web Land Information System", Demo, In Proceedings National Conference on Digital Government Research, dg.o2002, 2002, pp. 443-446.

[13] Wiegand, N.; Zhou, N.; and Cruz, I.F. "A Web Query System for Heterogeneous Geospatial Data", Scientific and Statistical Database Management, SSDBM, July 2003, pp. 262-265.

[14] Wisconsin Land Council Technical Working Group. Wisconsin Land Information System Technical Report, 1999.

Position statements

Semantic Integration in Biomedicine

Olivier Bodenreider and Songmao Zhang

U.S. National Library of Medicine, Bethesda, Maryland
National Institutes of Health, Department of Health & Human Services
{olivier|szhang}@nlm.nih.gov

Semantic integration research at NLM

In 1986, the National Library of Medicine (NLM) initiated a terminology integration project – the Unified Medical Language System® (UMLS®) – as "an effort to overcome two significant barriers to effective retrieval of machine-readable information": the variety of names used to express the same concept and the absence of a standard format for distributing terminologies. By integrating more than 60 families of biomedical vocabularies, the UMLS Metathesaurus® currently provides not only an extensive list of names (2.5 million) for its 900,551 concepts, but also over 12 million relations among these concepts. Its scope is broader and its granularity finer than that of any of its source vocabularies.

The major component of the UMLS is the Metathesaurus, a repository of inter-related biomedical concepts. The two other knowledge sources in the UMLS are the Semantic Network, providing high-level categories used to categorize every Metathesaurus concept, and lexical resources including the SPECIALIST lexicon and programs for generating the lexical variants of biomedical terms. The lexical resources play an important role in semantic integration by identifying lexically similar concepts. The potentially synonymous terms are reviewed by the Metathesaurus editors prior to being integrated into the UMLS.

As illustrated in Figure 1, by integrating the vocabulary of several subdomains of biomedicine, the Metathesaurus can be used for the integration of the various information systems and databases existing for these subdomains. For example, recently integrated terminologies include the NCBI taxonomy, used for identifying organisms, and Gene Ontology™, used for the annotation of gene products across various model organisms. The Metathesaurus also covers the biomedical literature with the Medical Subject Headings (MeSH), the controlled vocabulary used to index MEDLINE, a large bibliographic database. Core subdomains such as anatomy, used across the spectrum of biomedical applications, are also represented in the Metathesaurus with the Digital Anatomist Symbolic Knowledge Base. Finally, the subdomain represented best is probably the clinical component of biomedicine, with general terminologies such as SNOMED® International (and soon SNOMED-CT®), and the International Classification of Diseases, to name a few.

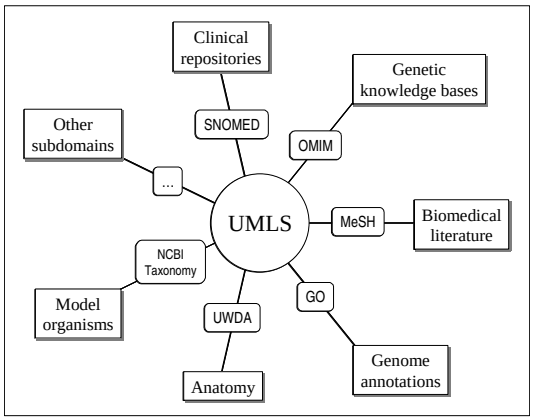


Figure 1. *The various subdomains integrated in the UMLS.*

More recently, the Medical Ontology Research project was initiated at NLM. The objective of this project is not to build an ontology of the biomedical

domain, but rather to develop methods whereby ontologies could be acquired from existing resources (including the UMLS Metathesaurus), as well as validated against other knowledge sources. Toward this endeavor, we have developed methods for aligning the UMLS with general ontologies (e.g., Cyc, WordNet) or specialized ones (e.g., the Gene Ontology). Additionally, methods have been developed for aligning UMLS knowledge sources (the Metathesaurus with the Semantic Network) and several biomedical ontologies outside the UMLS (the Foundational Model of Anatomy and GALEN). Related work developed as part of the Medical Ontology Research project also includes studying consistency and redundancy in biomedical terminologies and ontologies.

In the last eighteen months, we have been particularly interested in comparing two representations of anatomy: the Foundational Model of Anatomy and GALEN. Although the ultimate goal of this study is to compare the reasoning potential of these two ontologies, we have devoted most of the effort so far to aligning the two ontologies using a combination of lexical and structural techniques. We have also studied from both a quantitative and a qualitative perspective the contribution to the alignment of the different techniques used to obtain relationships from each ontology (knowledge augmentation, inference, etc).

Challenges and solutions

The challenging issues in semantic integration are many. In the biomedical domain, polysemy is one of them. For example, in molecular biology, a gene, the protein it produces, and the disease resulting from a mutation of this gene often have the same name. While geneticists and biologists usually have no problem identifying what is referred to by a particular name, this may not be the case for computer programs performing tasks such as information extraction or semantic interpretation.

While there are relatively few biomedical ontologies, there are, in contrast, many terminology systems developed for various purposes. Instead of building a medical ontology from the top-down (e.g., GALEN), the UMLS has attempted to integrate these terminology systems. Although the resulting Metathesaurus does not claim to be an ontology, we believe it can be used as the basis for building one. The biggest issue here is that the relations useful for organizing biomedical concepts for a given purpose (e.g., information retrieval) may not always be principled or consistent across terminological systems.

This approach to integrating many terminologies results in a semantic structure that may contain inconsistencies. On the other hand, redundancy is another feature of such systems that can be beneficial to semantic integration. The assumption here is that relations that appear in several sources are more likely to be semantically valid than relations asserted by one source only.

We also believe that domain knowledge can largely benefit semantic integration. Instead of using generic systems such as schema matching, we usually prefer to take advantage of the specific features of a given domain. For example, as illustrated in our paper, linguistic clues can be used reliably for extracting relations from anatomical concept names.

About the authors

Olivier Bodenerider is a Staff Scientist at the Lister Hill National Center for Biomedical Communications, US National Library of Medicine. He obtained a M.D. degree from the University of Strasbourg, France, in 1990 and a Ph.D. in Medical Informatics from the University of Nancy, France, in 1993. His research interests include terminology, knowledge representation, and ontology in the biomedical domain, both from a theoretical perspective and in their application to natural language understanding, reasoning, information visualization, and interoperability.

Songmao Zhang is currently a guest researcher at the Lister Hill National Center for Biomedical Communications, US National Library of Medicine. She obtained her PhD degree in computer science in 1992 at the Institute of Mathematics, Chinese Academy of Sciences where she is now an associate professor. Her research interests include ontology matching, knowledge representation, data mining, AI-based automatic animation, and natural language understanding.

Complex Alignments Between Ontology Universes

Alex Borgida
Dept. of Computer Science
Rutgers University
New Brunswick, USA
borgida@cs.rutgers.edu

The theme of this workshop is the synthesis of database and AI views of semantic integration.

We started working on issues of integrating Description Logic ontologies a few years ago, by examining to extent to which such formalisms are used as advertised: to *define* in a precise and formal way the meaning of every concept used in the terminology, rather than encoding meaning in the names of the terms [2]. Interestingly, that preliminary work was instigated by the definition of "conflict free schema integration" introduced in databases by Biskup and Convent [1].

However, let us start here by contrasting the treatment of *individuals* in approaches to semantic integration in the two fields.

In AI research on ontologies, it is almost always assumed (whether explicitly or not) that the areas of overlap between two ontologies being integrated concern the same individuals. In other words, if two concepts, C_1 from ontology O_1 and C_2 from ontology O_2, are to be related, then this is viewed as being based on a direct set-theoretic relationship — subset, equality, disjointness, overlap — between the sets of individuals denoted by C_1 and C_2. This still permits quite complex mappings, by allowing C_1 and C_2 to be complex concepts defined in terms of the terminologies O_1 and O_2 (e.g., [5]), but it does not make it possible to capture *systematic* relationships between individuals, such as the fact that the objects in one ontology (e.g., households in a census) correspond to sets of objects (e.g., persons living at that address) in another one.

In contrast, researchers in database integration, have recognized for a long time the need for complex translations between values in the databases being integrated: the early work of Kent [6] is replete with a variety of such examples, including the need to convert currencies, and convert different notions of income (before and after tax, net vs. gross, etc.). Kent's solution relies on functions expressed in a what is close to a general purpose programming language, equipped with loops and conditionals. He also provides examples where the relationship is not functional, such as the case when letter grades would need to be mapped to numeric values. This focus on complex mappings between individuals, evident in other work, such as the Clio project [7] for example, may also be due, in part, to the nature of the relational data model, which "dematerializes" individuals into tuples of values (integers and strings), and by the availability of powerful query languages to reconstruct values in the new, integrated database.

A natural question is to what extent complex correspondences between individuals are of interest to ontology integration and reasoning, or only in translating *database facts* (e.g., "Joan's salary_1 is 3000 dollars per week" to "Joan's salary_2 is 5000 Euros bi-weekly").

Consider the following example: One information source, IS_1, concerns literary works — novels, plays, poems, articles, authors, etc.; another information source, IS_2, is an entertainment guide for Southern Ontario, and maintains information about current and forthcoming events, such as sports events, performances of music, plays, etc. In integrating these two sources, we will want to match a literary work to its performances. Note that this correspondence is certainly not the identity function, and is not even a function: prefaces to plays and theater reviews are literary works that do not receive performances, and some plays are not being performed, while others are being performed on multiple nights (or receive multiple stagings). Suppose that as part of the process of semantic integration, we can be told information about this correspondence. For example, in this region plays are always performed in theaters, and all events occurring in Niagara on the Lake are performances of works by G.B. Shaw[1]. From this,

[1]Truth in advertising: Although this town does host a

one should be able to deduce that all events in Niagara on the Lake are theatrical performances.

Since Description Logics appear to be favored both as ontology representation languages, and as semantic representations for database schemas (they are more expressive than Entity Relationship diagrams), we have extended in [3, 4] the framework of Description Logics to allow such general binary relationships between individuals in the local domains of the information sources being integrated. (In fact, because the mapping is directional, we prefer to think of the resulting system as a federation of independent agents that import information to conduct local reasoning, rather than a single integrated entity.)

A central question in studying the resulting so-called "distributed description logics" is the language for expressing the properties of the correspondence R bbetween local domains. As usual, the choice of language affects the nature and complexity of reasoning in the resulting formalism.

It is obvious that allowing the mapping R to be represented by an arbitrary computable function prevents any kind of meaningful automatic reasoning in the resulting system. The papers mentioned above concentrate on simple restrictions of the form $\mathrm{R}(A) \subseteq D$ and $E \subseteq \mathrm{R}(B)$. But it is possible to view R as a *Description Logic role* (e.g., an OWL property [8]), in which case one can consider using the DL formalism to constrain it! In fact, a theorem proven in [4] shows that for a large class of description logic families this can be done using axioms involving property restrictions on R and its inverse, and then performing standard DL reasoning in a merged theory. For example, if we want to say that the mapping R is a bijection between the individuals in concepts A and D, we can assert

$A \sqsubseteq = 1\,\mathrm{R}$
$D \sqsubseteq = 1\,\mathrm{R}^-$
$A \sqsubseteq \forall \mathrm{R}.D$
$D \sqsubseteq \forall \mathrm{R}^-.A$

There are numerous open research problems concerning the extended formalism for expressing ontologies and mappings between them. These include

- problems introduced by the presence of datatypes in OWL
- the treatment of constraints on mappings, such as "$\mathrm{R}(A)$ and D overlap", which cannot be represented directly as subsumption axioms.
- the problem of expressing correspondences between complex objects in two ontologies, including the case when more than one element in one domain determines an individual in the other

Shaw festival, there are also other performances in town!

REFERENCES

[1] J. Biskup and B. Convent. "A Formal View Integration Method", *Proc. SIGMOD'86*

[2] A. Borgida and R. Ksters "What's not in a name: Some Properties of a Purely Structural Approach to Integrating Large DL Knowledge Bases", *Proc. Int. Workshop on Description Logics (DL'00)*, 2000.

[3] A. Borgida and L. Serafini, " Distributed Description Logics: Directed Domain Correspondences in Federated Information Sources", *Proc. CoopIS'02*, pp.36-53 (2002).

[4] A. Borgida and L. Serafini, "Distributed Description Logics: Assimilating Information from Peer Sources", *J. of Data Semantics*, to appear.

[5] T. Catarci and M. Lenzerini: "Representing and Using Interschema Knowledge in Cooperative Information Systems." *International Journal of Intelligent and Cooperative Information Systems 2(4)*, pp.375–398 (1993).

[6] W. Kent. "Solving Domain Mismatch and Schema Mismatch Problems with an Object-Oriented Database Programming Language." *Proc. VLDB'91*, Barcelona, Spain, pp. 147-160 (1991).

[7] R.J. Miller, L.M. Haas and M. Hernndez. "Schema Mapping as Query Discovery". *Proc VLDB'00*, Cairo, Egypt, pp.77-88 (2000).

[8] Peter F. Patel-Schneider, Patrick Hayes, Ian Horrocks (eds), "Web Ontology Language (OWL) Abstract Syntax and Semantics", `http://www.w3.org/TR/owl-semantics/` , August 2003

Position Statement: Efficient development of data migration transformations

Paulo Carreira
Oblog Consulting and FCUL
paulo.carreira@oblog.pt

Helena Galhardas
INESC-ID and IST
hig@inesc-id.pt

1 Introduction

Data migration and integration projects typically involve two phases. The first phase aims at establishing the schema and data mappings required for transforming schema and data. The second phase consists of developing and executing the corresponding schema and data transformations.

Several tools have been designed to assist the discovery of appropriate schema mappings [RB01], while data understanding solutions (e.g., Integrity [Val]) have been progressively adopted for helping to find out the correct data mappings. The development of the corresponding schema and data transformations is usually an ad-hoc process that comprises the construction of complex programs and queries.

Ideally, a framework should exist to assist the development, execution and correction of schema and data transformations. The execution of a given data or schema transformation usually gives further insight about the problem domain. For example, erroneous mappings are frequently detected when transformations are executed. In this case, the programs that implement the corresponding schema or data transformations must be modified. In real-world scenarios, this approach turns out to be infeasible due to the large number of modifications that must be introduced.

2 How we position ourselves

The computer-assisted development, execution and correction of schema and data mappings lay in the iterative refinement of mappings until the appropriate set of executable schema and data transformations is obtained.

Our work has been concerned with a particular case of the schema and data integration problem that consists of transforming a source schema into a fixed target schema. This problem frequently arises when a legacy system is migrated

into a modern system. We have developed a high-level language for specifying and refining schema and data mappings, and a system that supports its efficient execution. Since we have been involved in real-world projects, functionalities like a productive IDE (Integrated Development Environment) and mechanisms for project tracking and auditing have also received particular attention.

An initial version of our data migration tool-box named Data Fusion is already implemented and currently used in industrial settings. However, only a subset of the specification language is supported. The tool-box research and development are sponsored by Oblog Consulting.

3 Challenging issues

Due to their inherent complexity, schema and data mappings must be iteratively specified and refined. Shortening the time needed for each iteration results in a global reduction of the effort required to develop an entire data migration or integration project. The following features are required:

an adequate mapping specification language that provides the abstractions for conveniently specifying an refining the solutions to common data integration and transformation problems. This high-level language is also expected to fulfill the gap between business and implementation experts, thus reducing communication costs. Moreover, the language must be powerful enough so that ad-hoc programs are not needed. Domain specific languages [vDKV00] seem to be the approach to follow.

the partial execution of schema and data mappings enables the efficient deployment of potentially large and complex mappings and avoids the cost of entire compilations. This functionality is useful, for example, when predicting the effect of transforming data for a subset of fields. We plan to adapt and integrate a technique known as *program slicing* [Wei82, Tip95, Luc01] into our specification language for automatically computing simpler mappings given specific criteria.

the efficient execution of mappings is a major requirement in real-world scenarios for integrating and transforming millions of records in a limited time-frame. Several optimizations can be introduced both at compile and run time.

a debugging facility can greatly reduce the cost of locating anomalies in complex schema and data transformations. It should include a debugger and support partial executions and lineage tracing features [CW01].

4 About the authors

Paulo Carreira is a senior engineer at Oblog Consulting and a PhD student at FCUL (Faculty of Sciences of the University of Lisbon). He got his MSc on

automatic verification of object-oriented specifications. Helena Galhardas is a researcher at INESC-ID and professor at IST (Instituto Superior Técnico), the engineering school of the Technical University of Lisbon. Helena got her PhD on data cleaning.

References

[CW01] Y. Cui and J. Widom. Lineage tracing for general data warehouse transformations. In *Proc. of the Int'l Conf. on Very Large Data Bases (VLDB'01)*, Rome, Italy, September 2001.

[Luc01] A. De Lucia. Program slicing: Methods and applications. In *In 1st IEEE Int'l Workshop on Source Code Analysis and Manipulation*, pages 142–149, Florence, Italy, 2001. IEEE Computer Society Press, Los Alamitos, California, USA.

[RB01] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal: Very Large Data Bases*, 10(4):334–350, 2001.

[Tip95] F. Tip. A survey of program slicing techniques. *Journal of programming languages*, 3:121–189, 1995.

[Val] Vality. Home page of the integrity tool. http://www.vality.com/html/prodint.html.

[vDKV00] Arie van Deursen, Paul Klint, and Joost Visser. Domain-Specific Languages: An Annotated Bibliography. *SIGPLAN Notices*, 35(6):26–36, 2000.

[Wei82] M. Weiser. Programmers use Slicing when debugging. *Comunications of the ACM*, 25(7):446–452, July 1982.

Semantic Integration and Inconsistency

Steve Easterbrook
Department of Computer Science, University of Toronto
40 St George Street, Toronto, Ontario, M5S 2E4, Canada
http://www.cs.toronto.edu/~sme

Abstract

The management of inconsistency between multiple viewpoints has become a central problem in the development of large software systems. In this paper we argue that the same problem occurs in the development of the semantic web, and indeed that this is the central issue in semantic integration. A common approach is to attempt to remove inconsistencies, if necessary by discarding problematic information. We argue that this approach will greatly limit the utility of the semantic web. Instead, we argue the need for formal reasoning systems that can tolerate inconsistent information. A key observation is that the problem is essentially one of model management. Rather than seeking to build a single consistent model, the challenge is to reason about the inconsistencies and dependencies between a set of inter-related partial models, and to use paraconsistent logics when reasoning with information from inconsistent ontologies.

1. Viewpoint Integration in SE

For the past 15 years, we have been studying the problem of viewpoint integration in Software Engineering. Viewpoints are used in SE to support a loosely-coupled distributed approach to software development, in which different participants are able to maintain their own (partial) models of the system and its requirements, without being constrained by the need to be consistent with other participants' models [2]. By exploring the relationships between viewpoints, and the inconsistencies that arise when intended relationships do not hold, the participants discover disagreements, and understand one another's perspectives better.

The key insight of the viewpoints work is to see software development as a problem of model management, with the attendant goal of seeking coherence in information drawn from disparate sources. Software developers create models in a variety of notations to capture their current understanding of the problem and these models are rarely static. Developers analyze their models in various ways, and use the results of these analyses to improve them. They create multiple versions of their models to explore design options, and to respond to changing requirements. Hence, most of the time, design models are likely to be incomplete and inconsistent. Managing inconsistency as these models evolve is a major challenge.

In its narrowest sense, *consistency* is usually taken to mean *syntactic consistency*. In a good modeling language, syntactic consistency should correspond to the developer's intuitive notion of a "well-formed model". Hence, syntactic inconsistencies indicate simple mistakes, or slips, made by the designer. In this view, detection and resolution of inconsistency can be thought of as "model hygiene".

In our work, we have taken a much broader view of consistency. In our view, an inconsistency occurs whenever some relationship that *should* hold (of the model) has been violated. This definition has an intentional flavour: someone (e.g. the designer) *intends* that certain relationships hold. Such relationships may be internal to a model (e.g. the definition of an element should be consistent with its use), or may refer to external relationships (e.g. a model should be consistent with a particular choice of semantics, with existing standards, with good practice guidelines, or with another model, etc). This definition of inconsistency spans the semantics and pragmatics (i.e. the intended meanings and uses) of model elements, as well their syntax.

This view has several interesting consequences. Firstly, by this definition, most conceptual models are inconsistent most of the time, and attempting to remove all inconsistency is usually infeasible. Design involves finding acceptable compromises, rather than seeking perfection. Hence, in our work on consistency management, we don't view detection and removal of inconsistency as the main goal; instead, we focus on tools to explore the consistency relationships, and on reasoning techniques that tolerate inconsistency [7].

Secondly, most of the interesting consistency relationships arise implicitly as models are developed. If we wish to provide automated tools for consistency management, such consistency relationships have to be captured and represented. Thirdly, because of the intentional nature of these relationships, the set of relevant consistency relationships for a given model will change over time as the developer's intent changes.

We have made significant progress in the past 15 years in our study of these ideas.

- We have developed a number of representation schemes for capturing and managing the consistency relationships in modeling languages. These include a

first order logic for checking XML documents [6], a production rule approach for checking UML models [5] and a structural mapping technique based on graph morphisms for graphical notations [8]

- We have developed a number of reasoning techniques that tolerate inconsistency. In general, these make use of paraconsistent logics, i.e. non-classical logics whose entailment relations are not explosive under contradiction. For example, we have explored the use of a family of multi-valued logics identified by Fitting [3], and demonstrated that we can build practical reasoning engines for these logics [1].
- We have developed a theoretical framework for combining information from multiple, inconsistent sources, without first resolving the inconsistencies [8]. The composition technique we use in this framework preserves information about relative certainty and inconsistency of the source models.

2. Inconsistency in the Semantic Web

It now seems clear that if the semantic web is to be realized, it will not be by agreeing on a single global ontology, but rather a by weaving together a large collection of partial ontologies that are distributed across the internet [4]. We see the issues in semantic integration to be essentially the same as those in viewpoint management. In fact, the conceptual modeling tasks to which we have applied viewpoints are essentially ontology modeling tasks. For example, in requirements analysis, the models we build are domain ontologies, together with goal hierarchies and behaviour models that are based on them.

We can therefore make the following observations:

- By its very nature, the semantic web will be based on a heterogenous collection of viewpoints (partial ontologies), each constructed by a particular stakeholder for a particular purpose.
- These ontological components will not be static – they will evolve as the web services for which they were created evolve.
- For much of the time, these ontological components will be inconsistent with one another, in terms of the meanings attached to ontological elements, and the ways in which those elements are used.
- Semantic integration can only be achieved if (intentional) consistency relationships between ontological components can be captured and made explicit.
- Reasoning over the semantic web will only be possible if we have automated tools for testing these consistency relationships to identify inconsistencies.
- Fixing the inconsistencies will usually not be feasible, as this would require a globally distributed, disparate set of stakeholders to agree on and subscribe to a universal conceptual model.
- Hence, practical reasoning on the semantic web must be tolerant of inconsistency.

It should be clear by now that we believe *the* central problem in the semantic web will be managing inconsistency between ontologies. We believe our work on consistency management in the viewpoints framework suggests some promising ways forward. In particular, we believe we have practical solutions to two of the greatest challenges: representing the consistency relationships between ontologies, and reasoning over composite ontologies that contain inconsistencies. Several of the techniques described above are applicable.

We are currently investigating the application of the theoretical framework described in [8] to ontology integration. Briefly, this framework was developed for combining models in graph-based notations, where the combinations must take into account relative certainty and inconsistency of the source models. We explicitly tag elements of the models with labels indicating relative certainty and relative consistency. We call the resulting models *fuzzy viewpoints*. We then use graph morphisms to capture structural mappings between fuzzy viewpoints. Finally, we compute compositions of fuzzy viewpoints using the categorical construct of a pushout. The theoretical results on which this framework is based guarantee that we can always compute the composition, that it preserves the structure of the source models, and that no information is lost or gained in the composition. We believe that this theory provides an excellent foundation for ontology integration.

3. References

[1] M. Chechik, B. Devereux, S. M. Easterbrook & A. Gurfinkel "Multi-Valued Symbolic Model-Checking". To appear, IEEE Trans. on Software Engineering and Methodology, 2003.

[2] S. M. Easterbrook & B. A. Nuseibeh "Managing Inconsistencies in an Evolving Specification". 2nd IEEE Int. Symp. on Requirements Engineering (RE'95), York, UK, p48-55. Apr 1995.

[3] M. Fitting "Kleene's three-valued logics and their children". Fundamenta Informaticae, 20, 113-131, 1994

[4] J. Hendler, "Agents and the Semantic Web". IEEE Intelligent Systems, 16(2) 30--37, 2001.

[5] W. Liu, S. M. Easterbrook & J. Mylopoulos, "Rule-Based Detection of Inconsistency in UML Model". Workshop on Consistency Problems in UML-Based Software Development, 5th Int. Conference on the Unified Modeling Language, Dresden, Germany, Oct 1, 2002.

[6] C. Nentwich, W. Emmerich, A. Finkelstein and E. Ellmer, "Flexible Consistency Checking" ACM Trans. on Software Engineering and Methodology 12 (1) 28-63, 2003.

[7] B. A. Nuseibeh, S. M. Easterbrook & A. Russo, "Making Inconsistency Respectable in Software Development", J. of Systems and Software, 58 (2) 171-180. 2001.

[8] M. Sabetzadeh & S. M. Easterbrook "Analysis of Inconsistency in Graph-Based Viewpoints: A Category-Theoretic Approach". 18th IEEE Int. Conf. on Automated Software Engineering, Montreal, Oct. 6-10, 2003.

Towards composing and benchmarking ontology alignments

Jérôme Euzenat
INRIA Rhône-Alpes
Jerome.Euzenat@inrialpes.fr

Formal resources on the web will be expressed in the framework of an ontology. Integrating such resources requires finding agreement between ontologies (ontology alignment). Many methods have been put forward for aligning ontologies. They involve different techniques (linguistic, statistical, structural) and are based on various features of ontologies (names, internal structure, external structure, extension or semantics). It is necessary to allow these methods to cooperate and to be able to compare them in order to stimulate research on ontology alignment. We propose here a first attempt toward this based on a format for expressing the alignment independently of the methods used for building them and a first benchmarking framework for alignment methods. This sketch of foregoing work in the framework of the European FP6 'Knowledge web' is intended to be discussed and improved.

1 What's in an ontology alignment

Like the web, the semantic web will have to be distributed and heterogeneous. As such, the integration of resources found on this semantic web is its main problem. For contributing solving this problem, data will be expressed in the framework of ontologies. However, ontologies themselves can be heterogeneous and some work will have to be done for restoring interoperability.

Semantic interoperability can be grounded on ontology reconciliation: finding relationships between concepts belonging to different ontologies. We call this process "ontology alignment". The alignment result can be used for various purposes such as displaying the correspondances, transforming one source into the other, creating a set of bridge axioms between the ontologies or buiding query wrappers which rewrite queries for reaching a particular source.

The ontology alignment problem can be described in one sentence: given two ontologies which describe each a set of discrete entities (which can be classes, properties, rules, predicates, etc.), find the relationships (e.g., equivalence or subsomption) holding between these entities. Hence, in first approximation, an alignment is a set of pairs of elements from each ontology.

However, there are other aspects of alignments that can be added to this first approximation:

- There is not only equivalence/subsumption but more sophisticated operators (e.g., concatenation of firstname and lastname for instance considered in [2]).
- Another relevant point is to define the kind of alignment sought. Usual notations are 1:1, 1:m, n:1 or n:m. We prefer to note if the mapping is injective, surjective and total or partial on both side. We then end up with more alignment arities (noted with, 1 for injective and total, ? for injective, + for total and * for none and each sign concerning one mapping and its converse): ?:?, ?:1, 1:?, 1:1, ?:+, +:?, 1:+, +:1, +:+, ?:*, *:?, 1:*, *:1, +:*, *:+, *:*. These partial alignments (i.e. they align only one part of each ontology) could be provided as input (or constraints) of the alignment problem. This would allow iterative alignment: starting with a first alignment, followed by user feed-back, subsequent alignment rectification, and so on.
- Last, since many alignment methods compute a strength of the relation between entities, this strength can be provided as a normalized measure.

Then the alignment description can be stated as follows:

a set of pairs with characterization of the relation (default "=") and strength (default 1);
a statement of arity (default 1:1) and even more generaly a statement of the properties of the alignment when this can be provided by the alignment method (e.g., a subsumption preservation assertion),

This is simpler than the alignment representation of [1], but is supposed possible to produce by most alignment tools.

Having alignment results in a standardized format can be very useful for taking advantage of these alignments in various contexts (transformations, queries, etc.). It can be used for modularizing alignments; for instance, by first using terminological alignment methods for labels, having this alignment agreed or amended by a user and using it as input alignment for a structural alignment. But it can also be used for benchmarking alignment methods. To that extent, we will need a measure of the distance between such an alignment and an expected target alignment.

2 Benchmarking alignment methods

There are various methods for computing alignments, however, it seems sensible to ask alignment methods for an output alignment, given:

- two ontologies to be aligned;
- an input partial alignment (possibly empty);
- a characterization of the wanted alignment (1:+, ?:?, etc.).

A measure of the distance between alignments would allow to evaluate alignment methods. There are two kinds of benchmarks which seems useful: competence benchmarks and performance benchmarks.

2.1 Competence benchmark

Competence benchmarks aim at characterising the kind of task each method is good at. There are many different areas in which methods can be evaluated. One of them is the kind of features they use for finding matching entities (this complements the taxonomy provided in [2]):

terminological (T) comparing the labels of the entities trying to find those which have similar names;
internal structure comparison (I) comparing the internal structure of entities (e.g., the value range or cardinality of their attributes);
external structure comparison (S) comparing the relations of the entities with other entities;
extensional comparison (E) comparing the known extension of entities, i.e. the set of other entities that are attached to them (in general instances of classes);
semantic comparison (M) comparing the interpretations (or more exactly the models satisfying the entities).

A set of reference benchmarks, targetting one type of feature at a time can be defined. These benchmarks would caracterize the competence of the method for one of these particular features of the languages.

2.2 Performance benchmarks

Performance benchmarks are aimed at evaluating the overall behaviour of alignment methods in versatile real-life examples. It can be organised as a yearly or bi-annual challenge (à la TREC) for comparing the best compound methods. Such benchmarks should yield as a result the distance between provided output and expected result as well as traditional measures of the amount of resource consumed (time, memory, user input, etc.).

References

[1] Jayant Madhavan, Philip Bernstein, Pedro Domingos, and Alon Halevy. Representing and reasoning about mappings between domain models. In *Proc. 18th National Conference on Artificial Intelligence (AAAI 2002), Edmonton (CA)*, pages 122–133, 1998. http://citeseer.nj.nec.com/milo98using.html.

[2] Erhard Rahm and Philip Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

Evaluating Matching Algorithms: the Monotonicity Principle

Position Statement

Avigdor Gal
Technion – Israel Institute of Technology
avigal@ie.technion.ac.il

Traditionally, semantic reconciliation was performed by a human observer (a designer or a DBA) [8] due to its complexity [3]. However, manual reconciliation (with or without computer-aided tools) tends to be slow and inefficient in dynamic environments and does not scale for obvious reasons. Therefore, the introduction of the semantic Web vision and the shift towards machine understandable Web resources has unearthed the importance of automatic semantic reconciliation. Consequently, new tools for automating the process, such as GLUE [4], and OntoBuilder [11], were introduced.

Generally speaking, the process of semantic reconciliation is performed in two steps. First, given two attribute sets $\mathcal{A}$ and $\mathcal{A}'$ (denoted *schemata*) with n_1 and n_2 attributes, respectively,[1] a degree of similarity is computed **automatically** for all attribute pairs (one attribute from each schema),[2] using such methods as name matching, domain matching, structure (such as XML hierarchical representation) matching, and Machine Learning techniques. As a second step, a single mapping from $\mathcal{A}$ to $\mathcal{A}'$ is chosen to be the *best mapping*. Typically, the best mapping is the one that maximizes the sum (or average) of pair-wise weights of the selected attributes. We differentiate the best mapping from the *exact mapping*, which is the output of a matching process as would be performed by a human observer.

Automatic matching may carry with it a degree of uncertainty since "the syntactic representation of schemas and data do not completely convey the semantics of different databases" [10]. As an example, consider name matching, a common method in tools such as OntoBuilder [6], Protégé [5], and Ariadne [9]. With name matching, one assumes that similar attributes have similar (or even identical) names. However, the occurrence of synonyms (*e.g.*, `remuneration` and `salary`) and homonyms (*e.g.*, `age` referring to either human age or wine age) may trap this method into erroneous mapping. As a consequence, there is no guarantee that the exact mapping is always the best mapping.

We present the *monotonicity principle*, a sufficient condition to ensure that exact mapping would be ranked sufficiently close to the best mapping. Roughly speaking, the monotonicity principle proclaims that by replacing a mapping with a better one, score wise, one gets a more accurate mapping (from a human observer point of view), even if by doing so, some of the attribute mappings are of less quality. We have demonstrated, through theoretical [7] and empirical analysis,[2] that for monotonic mappings that satisfy the monotonicity principle, one can safely interpret a high similarity measure as an indication that more attributes are mapped correctly. An immediate consequence of this result is the establishment of a corroboration for the quality of mapping algorithms, based on their capability to generate monotonic mappings. We have experimented with a matching algorithm and report on our experiences in [2]. Our findings indicate that matching algorithms that generate monotonic mappings are well-suited for automatic semantic reconciliation. Another outcome of the monotonicity principle is that a good automatic semantic reconciliation algorithm would rank the exact mapping relatively close to the best mapping, thus enabling an efficient search of the exact mapping [1].

Monotonicity is not defined in "operational" terms, since it is compared to an initially unknown exact mapping. In fact, such an operational definition may not be generally developed, since algorithms may perform well only on some schema pairs. Therefore, a task for future research involves possible classification of application types on which

[1] The use of relational terms is in no way restrictive, and is used here to avoid the introduction of an extensive terminology that is of little benefit in this paper.

[2] Extensions to this basic model (*e.g.*, [10]) are beyond the scope of this statement.

certain algorithms would work better than others. Best mappings may also be subjective at times (less so in the type of applications we were exploring, though). It is not clear at this time how an operational definition can be developed in such cases without personalizing the algorithms to specific human observers. Taken to the extreme, an adaptive algorithm would rank erroneous mappings higher, simply by following a human observer presumptions. This line of research is also left for future investigation.

The recent steps taken in the direction of automating semantic reconciliation highlight the critical need of this research. As the automation of the process has already begun to take shape, often without the benefits of thorough research, the study is timely. We envision multitude of applications of automatic schema matching to the semantic Web. For example, we are currently designing smart agents that negotiate over information goods using schema information and can combat schema heterogeneity.

Acknowledgments

The work was partially supported by Technion V.P.R. Fund - New York Metropolitan Research Fund, the Ministry of Science, Culture, and Sport in Israel and by the CNR in Italy, and the IBM Faculty Award for 2002/2003 on "Self-Configuration in Autonomic Computing using Knowledge Management."

References

[1] A. Anaby-Tavor, A. Gal, and A. Moss. Efficient algorithms for top-k matchings. Submitted for publication. Available upon request from avigal@ie.technion.ac.il, 2003.

[2] A. Anaby-Tavor, A. Gal, and A. Trombetta. Evaluating matching algorithms: the monotonicity principle. In S. Kambhampati and Craig A. Knoblock, editors, *Proceedings of the IJCAI-03 Workshop on Information Integration on the Web*, pages 47–52, Acapulco, Mexico, August 2003.

[3] B. Convent. Unsolvable problems related to the view integration approach. In *Proceedings of the International Conference on Database Theory (ICDT)*, Rome, Italy, September 1986. In *Computer Science'* Vol. 243, G. Goos and J. Hartmanis, Eds. Springer-Verlag, New York, pp. 141-156.

[4] A. Doan, J. Madhavan, P. Domingos, and A. Halevy. Learning to map between ontologies on the semantic web. In *Proceedings of the eleventh international conference on World Wide Web*, pages 662–673. ACM Press, 2002.

[5] N. Fridman Noy and M.A. Musen. PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-2000)*, pages 450–455, Austin, TX, 2000.

[6] A. Gal, G. Modica, and H.M. Jamil. Improving web search with automatic ontology matching. Submitted for publication. Available upon request from avigal@ie.technion.ac.il, 2003.

[7] A. Gal, A. Trombetta, A. Anaby-Tavor, and D. Montesi. A model for schema integration in heterogeneous databases. In *Proceedings of the 7th International Database Engineering and Application Symposium*, Hong Kong, China, July 2003.

[8] R. Hull. Managing semantic heterogeneity in databases: A theoretical perspective. In *Proceedings of the ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS)*, pages 51–61. ACM Press, 1997.

[9] C.A. Knoblock, S. Minton, J.L. Ambite, N. Ashish, I. Muslea, A. Philpot, and S. Tejada. The Ariadne approach to web-based information integration. *International Journal of Cooperative Information Systems (IJCIS)*, 10(1-2):145–169, 2001.

[10] R.J. Miller, L.M. Haas, and M.A. Hernández. Schema mapping as query discovery. In A. El Abbadi, M.L. Brodie, S. Chakravarthy, U. Dayal, N. Kamel, G. Schlageter, and K.-Y. Whang, editors, *Proceedings of the International conference on very Large Data Bases (VLDB)*, pages 77–88. Morgan Kaufmann, 2000.

[11] G. Modica, A. Gal, and H. Jamil. The use of machine-generated ontologies in dynamic information seeking. In C. Batini, F. Giunchiglia, P. Giorgini, and M. Mecella, editors, *Cooperative Information Systems, 9th International Conference, CoopIS 2001, Trento, Italy, September 5-7, 2001, Proceedings*, volume 2172 of *Lecture Notes in Computer Science*, pages 433–448. Springer, 2001.

Position Statement

Fausto Giunchiglia
Full Professor of Computer Science,
Department of Information and Telecommunication Technology,
University of Trento

Pavel Shvaiko
First-year PhD Student in Information and Telecommunication Technology,
Department of Information and Telecommunication Technology,
University of Trento

Research Lines & Solutions

We are interested in the development of methodologies, theories, mechanisms, and technologies which will allow for an interaction of information sources (data bases, information systems, web sites, file systems, …) within distributed environments, e.g., P2P, World Wide Web, which must be effective, and implemented with real time constraints.

We propose a new approach, that we call *data coordination* that rejects the assumption, made in previous approaches, most noticeably in data integration, that the involved information sources act as if they were a single (virtual) source, for instance modeled as a global schema. We talk of coordination meaning that ... "... Coordination is managing dependencies between interacting information sources." From an operational point of view, the distinguishing feature of data coordination is that many of the parameters e.g., schema or ontology describing information source, influencing the interaction among applications or peers are decided at run time.

One of the main tools needed to make data coordination approach feasible, is to design and develop a generic semi-automated *semantic matching* approach, which provides interoperability at a semantic level among peers and data management applications at run time. The key intuition behind semantic matching approach is to calculate mappings between schema or ontology elements by computing *semantic relations* (for example, elements can be equal, more general, etc.) using propositional satisfiability deciders, instead of computing coefficients rating match quality in the [0,1] range. Notice that propositional satisfiability procedures are sound and complete, which allow us to find all and possible mappings, while other techniques which calculate coefficients in [0,1] are only based on heuristics.

The main conceptual tool at the basis of our proposed solution(s) is the notion of context. From a theoretical point of view, the model theory of context, the so called Local Models Semantics, provides a foundation to our approach. From an implementational point of view, a context is a data structure which can be used to index many things (for instance: a peer information source, a view on a peer information source, a user query, a user point of view, …). The context data structure allows us to know where any piece of data comes from, and to use this information to perform the most appropriate "matching" among the data coming from the many autonomous peer information sources.

These ideas have been developed within two research projects:

- Context2Context. http://www.dit.unitn.it/~p2p,
- Edamok. http://edamok.itc.it/,

and later exploited by a start-up company on distributed knowledge management:

- Dthink. http://www.dthink.biz/profilo.htm.

Future Directions

The most challenging issues we encountered so far and our future work can be declared as follows:

- Designing measures to assess the quality of query answering in settings of distributed systems, e.g., P2P, WWW;
- How to extract semantics from schemas (graphs);
- Development of a theory of matching via context theory;
- Development of semantic matching tool and a library of semantic element-level schema/ontology matchers;
- Development of a formal methodology for testing and evaluating of schema and ontology matching tools;
- Iterative semantic matching;
- Concept approximation techniques & semantic matching.

References

1. L. Serafini, F. Giunchiglia, J. Mylopoulos, P. Bernstein. *The Local Relational Model: A Logical Formalization of Database Coordination* // IRST Technical Report 0301-08. Also In Proceedings of CONTEX'03.

2. F. Giunchiglia. *Contextual Reasoning* // IRST Technical Report # 9211-20. Also in Epistemologia - Special Issue on "I Linguaggi e le Macchine", XVI:345-364, 1993.

3. F. Giunchiglia, P. Shvaiko. *Semantic Matching* // Technical Report # DIT-03-013.

4. P. Bernstein, F. Giunchiglia, A. Kementsietsidis, J. Mylopoulos, L. Serafini, and I. Zaihrayeu. *Data Management for Peer-to-Peer Computing: A Vision* // Technical Report # DIT-02-013. Also in proceedings of Web DB 2002.

5. C. Ghidini, F. Giunchiglia. *Local Models Semantics, or Contextual Reasoning = Locality + Compatibility* // Artificial Intelligence. 127(3):221-259, 2001.

Challenges for Biomedical Information Integration

C. Golbreich, A. Burgun
Laboratoire d'Informatique Médicale, Faculté de Médecine, 35033 Rennes, France
Christine.Golbreich@uhb.fr Anita.Burgun@univ-rennes1.fr

In the last decade significant progress have been done in Information Integration. Most systems for data integration issued from database and AI communities are mediator-based centralized systems. More recently, new approaches [4] [1] emerged, proposing distributed integration, that are quite attractive for Biological Information Integration (BII), such as functional genomics. Their deployment in BII depends on two main features. BII requires *flexible* integration and *expressive* representation languages.

1 Flexible information integration

Extensibility and real-time data are crucial requirements for BII. For example, Genomics is a very fast-moving field. Web sources are multiple, with huge and constantly evolving content (versioning of GO and UMLS). New online ontologies and specialized databanks frequently appear. Datawarehouses which can be quite powerful, providing high access performance are not well appropriate to such evolving data. More flexible integration, either centralized mediators or peer-based distributed integration might be more appropriate.

1.1 Mediator-based integration

A mediator includes a *global ontology* G (or mediated schema) and a set M of *mappings*, relating the global ontology G to the sources ontologies S. The query engine exploits this knowledge to reformulate the user query into queries that refer to the sources ontologies S. In bioinformatics or in medicine, new sources constantly appear and shall be added to S. Therefore, mainly for their easier extensibility, *local as view* (LAV) mediators defining the content of sources in terms of views over the global ontology, might be more appropriate than *global as view* (GAV) defining the global ontology in terms of views over the sources e.g. Tambis [8]. However, they still raise representation problems (§ 2).

1.2 Peer-based integration

Mediators are a significant progress, but for scaling up the Web, centralized integration may be not flexible enough, and distributed systems perhaps even better appropriate. As illustrated for bioinformatics [6], databanks are not only data "sources" but also include precious links and mappings, through their cross-references to general ontologies and to other databanks. Such local relations between sources should be explicitly represented and directly exploited to infer new information. Peer-based integration where "every participant should be able to contribute new data and relate it to existing concepts and schemas, define new schemas that others can use as frames or reference for their queries or define new relationships between existing schema or data providers" [4] is challenging to address the extensibility and distribution encountered in BII.

2 Rich languages for ontologies and mappings

Whatever mediator or peer-based integration systems, rich formal languages are required for representing ontologies, queries, and mappings, in the biomedical domain.

2.1 A DL extended by rules for ontologies

As advocated in [2] a rich language, that is expressive enough to allow a fine and precise representation of both structural (concepts, properties, and hierarchies) and deductive knowledge, is required in the biomedical domain. The next W3C standard OWL(-DL) is a good candidate for taxonomies, but is not sufficient and should be extended by rules for the deductive part. Rules are particularly needed to represent dependencies between relations, such as mereotopological (part-of) and topological relationships, propagation of relations along transitive role, or consistency constraints [2] etc., for instance location of a disease is inherited across partonymy: "has-location propagates via part-of" [7]. However, as the combination of an expressive DL e.g. $ALNR$ with rules e.g. Datalog enlarges the search space, a trade-off shall be found in limiting OWL or/and rules expressiveness, in order to remain decidable and to have sound and complete algorithms for subsumption and satisfiability. Second, using OWL as the ontology language in an integration system, fuels additional new questions, about (1) the query language: if rules are wanted to define conjunctive queries, the issue of a logical language combining OWL(-DL) with rules occurs again (2) the mappings language: how the mappings should be

represented; for example, by OWL subsumption or other axioms, by rules? (3) the query answering algorithm: decidability depends on the ontology, query, mapping languages. Thus, an integrated framework including OWL (or sublanguage) were queries reformulation is decidable is a key challenge for BII.

2.2 A metamodel and a logical language for the mappings

As illustrated in Bioinformatics (see [6]) the explicit representation of mappings play a key role in mediation. But there are several related problems to solve, in particular two main ones: the *modeling problem* "how to model the mappings between the sources and the global ontology (or between peers)"? and the *representation problem* "how to represent the mappings"?

A first challenge is to define a "*metamodel*" for *mappings*, at a conceptual level, independently of the representation language. For example, from the analysis of existing database or DL integration sytems, a first possible simple model[1] is to define, for a source s, *mappings* as triples (D, P, C), where D is a set of assertions relating the kinds of *data* that can be found in the source s to the concepts of the global G, where C is a set of *constraints* on its elements expressing restrictions on the data, or integrity constraints in terms of the global G, where P is a set of assertions relating *local properties* of the source s *to* G *properties*[2]. For example, for an integration system in genomics, where the global ontology G includes the concepts **Protein**, **Species,** **HumanSpecies** and properties **organism**, mappings for the source SWISS-PROT (SW) are defined as a set of assertions stating 1) that SW entries correspond to instances of **Protein**, 2) to which G entities, its lines are related, e.g. the OS line corresponds to the property **organism** and its content to instances of **Species**[3], 3) constraints e.g. the data of SW file "proteins of the non-redundant human proteome set" contains only human proteins. Thus, SWISS-PROT mappings are defined by the triple (D_{sw}, P_{sw}, C_{sw}), where D_{sw} = {SW-data → **Protein**, ...}, P_{sw}: = {SW-OS → **organism**, ...}, C_{sw}:= {OS-data → **HumanSpecies**, ...}

A second challenge is to define a logical language for representing mappings and semantics of "→". Most mediators represent mappings as views over databases [3]. But several issues are now re-opened (1) which logical formalism to use, DL (OWL), rule, else? (2) if OWL, then how to represent them? In principle, subclass or subrole axioms e.g. $V^{SP}_{data} \subset$ Protein, $V^{SP}_{OS} \subset$ organism, $V^{SP}_{data} \subset (\forall$ organism HumanSpecies) are possible. Another option, is to represent them by rules e.g. $V^{SP}_{data}(X) \Rightarrow$ Protein(X), $V^{SP}_{OS}(X,Y) \Rightarrow$ organism(X,Y), and to to have a more complex model, for instance allowing to map a local property to a G more complex expression. But the logical formalism to represent mappings with OWL ontologies is still an open issue. Indeed, as well studied [5] [3] the formalism has direct implications on the query reformulation problem, and as the formalism for expressing mappings becomes more expressive, it becomes harder. In conclusion, an hybrid formalism combining a subclass of OWL with rules, that allows to remain decidable and to have sound and complete algorithms for subsumption and satisfiability and if possible with good properties for the reformulation of queries using mappings is another key issue for BII.

Both mediator or peer-based integration raise a major question, that of available tools, ready to be used in BII.

3 References

[1] Bernstein P, Giunchiglia F, Kementsietsidis A, Mylopoulos J, Serafini L., Zaihrayeu I. Data management for peer-to-peer computing: A vision, Workshop WebDB 2002.

[2] Golbreich, C., Dameron, O., Gibaud, B., Burgun A. Web ontology language requirements w.r.t expressiveness of taxononomy and axioms in medecine, ISWC 2003, Springer, 2003.

[3] Halevy A. Y. Answering queries using views. The VLDB Journal, 10(4):270-294, 2001.

[4] Halevy, A. Y. Zachary G. Ives, Dan Suciu, and Igor Tatarinov. Schema mediation in peer data management systems. In ICDE, 2003.

[5] Levy A. Y, Rousset MC, The Limits on Combining Recursive Horn Rules with Description Logics, AAAI/IAAI, Vol. 1 (1996)

[6] Marquet G, Golbreich C., Burgun A From an ontology-based search engine towards a mediator for medical and biological information integration, Semantic Integration Workshop, ISWC 2003, Sanibel, Florida, 2003.

[7] Rector A. Analysis of propagation along transitive roles: Formalisation of the GALEN experience with Medical Ontologies, 2002 Int. Workshop on Description Logics DL2002, April 19-21, (2002)

[8] Stevens R, Baker P, Bechhofer S, Ng G, Jacoby A, Paton NW, Goble CA, Brass A. TAMBIS: transparent access to multiple bioinformatics information sources. Bioinformatics. 2000 Feb;16(2):184-5.

[1] presented on a LAV mediator, but it can be generalized to other approaches, including Peer-based integration

[2] "concept" and "property" refer to Class and Property in OWL.

[3] organism(s) which was (were) the source of the stored sequence

Semantic Integration : Position Statement

Michael Grüninger and **Joseph B. Kopena**
Manufacturing Systems Integration Division
National Institute of Standards and Technology
100 Bureau Drive
Gaithersburg, MD 20899-8260
gruning@cme.nist.gov, jkopena@cme.nist.gov

Overview of our Research

Much of the research in semantic integration has reduced the problem to ontology integration – two software applications can be integrated by specifying the semantic mappings between their respective ontologies. However, few applications in practice have explicitly specified ontologies, and even when they do, the ontologies are not fully axiomatized (that is, there exist intended interpretations that are not models of the axioms or there exist unintended models of the axioms). Consequently, ontology mappings are not sufficient to achieve semantic integration.

To address this problem, we adopt what we call the Ontological Stance – we model a software application as if it were an inference system with an axiomatized ontology, and use this ontology to predict the set of sentences that the inference system decides to be satisfiable. This is analogous to the intentional stance, which is the strategy of interpreting the behavior of an entity by treating it as if it were a rational agent who performs activities in accordance with some set of intentional constraints.

In our work, we focus on techniques for achieving the semantic integration of software applications directly by using ontologies as interlingua between the applications themselves. The distinguishing feature of this Interlingua architecture is the existence of a mediating ontology that is independent of the software applications' ontologies and which is used as a neutral interchange ontology. The semantic mappings between application and interlingua ontologies are manually generated and verified prior to interaction time. This process of creating the mapping between the application ontology and the interlingua ontology is identical to the process of creating a mapping directly between two application ontologies. However, it has two advantages. First, we only need to specify one mapping for each application ontology, rather than specifying a mapping for each pair of application ontologies. Second, if the application ontologies and the interlingua ontology are specified using the same logical language, then the translation can be accomplished by applying deduction to the axioms of the interlingua ontology and the ontology mappings. If these mappings have already been verified to preserve semantics between the application and interlingua ontologies, we are guaranteed that translation between the applications also preserves semantics.

The two tools that we are developing are described in our workshop paper. The Twenty Questions tool supports the semiautomatic generation of semantic mappings between the PSL Ontology and the terminology used by a software application. The Process Information Exchange Protocol compares the profiles generated by the Twenty Questions for different software applications to determine which concepts can be either fully or partially shared.

Challenges and Open Problems

What is Semantic Integration?

We still lack a precise characterization of the problem of semantic integration. In some sense, if the ontologies are using the same underlying logical language then the notion of relative interpretation is necessary for semantic integration. However, it is not sufficient – it does not capture all of our intuitions concerning partial translation and it does not distinguish between ontologies for different but overlapping domains.

Testing Semantic Mappings

Once semantic mappings have been proposed between two ontologies or software applications, we still need some methodology for evaluating the correctness and completeness of the mappings so that we can determine whether or not semantic integration has been achieved. If the ontologies are fully axiomatized, then we can provide a model-theoretic evaluation of the semantic mappings (e.g. preservation of models or submodels). However, as we observed above, most software applications do not use fully axiomatized ontologies; the best we can do in these cases is to use an empirical methodology to evaluate the semantic mappings between the terminology of the applications. Adopting the Ontological Stance, we can determine whether inferences performed by the applications are preserved by the mappings.

Implementation of Testbeds

There are several critical issues in semantic integration that can only be solved by empirical approaches. These include the expressiveness/decidability tradeoff for ontology representation languages, the evaluation of different mapping techniques, and determining whether the lack of ontology reuse is due to superficial or deep ontological commitments. We need to establish academic and industrial testbeds that consist of multiple agents and ontologies within different integration architectures, so that participants can carry out experiments to test the critical issues in semantic integration.

Authors

Michael Grüninger

Michael Gruninger is currently a Research Scientist in the Institute for Systems Research at the University of Maryland College Park and also a Guest Researcher at the National Institute for Standards and Technology (NIST). Michael was previously a Senior Research Scientist in the Enterprise Integration Laboratory of the Department of Mechanical and Industrial Engineering at the University of Toronto, where he was the project manager for numerous international projects in collaboration with industry, academia, and government.

His current research focuses on the design and formal characterization of ontologies and their application to problems in manufacturing and enterprise engineering. He is the project leader for the Process Specification Language project at NIST. He is also the project leader for ISO 18629 (Process Specification Language) within the International Standards Organization (ISO).

Joseph Kopena

Joseph Kopena is an undergraduate research assistant in Drexel University's Geometric and Intelligent Computing Laboratory. His research interests include knowledge representation and common sense reasoning in engineering domains and automated system integration through formal ontology. Contact him via joe@plan.mcs.drexel.edu.

What do we need for ontology integration on the Semantic Web
Position statement

Natasha F. Noy
Stanford University
251 Campus Drive, Stanford, CA 94305, USA
noy@smi.stanford.edu

Ontologies, integration, and the Semantic Web

In order for the Semantic Web participants to share information, they must have some agreement on what elements in their shared domain of interest exist and how these elements can relate to one another. A formal specification of such an agreement is called an **ontology**. An ontology for a domain enumerates and gives semantic descriptions of concepts in the domain of discourse, defining domain-relevant attributes of concepts and various relationships among them. For example, an ontology describing a wines will include such concepts as vintages, wine regions, wineries, grape varieties, and so on. It will also include relations such as produced by, made from, color, year, and body of wine.

The ultimate vision of using formal ontologies is to develop a single ontology or a small set of ontologies that everyone will conform to. Alternatively, on a smaller scale, there could be single organization-wide ontologies. Semantic integration then becomes a much easier, if not a trivial, problem since everyone shares the same set of definitions.

However, such a vision seldom, if ever, becomes a reality. Just as there is more than one Web directory (e.g., Yahoo!, ODP, etc.), more than one shopping site, more than one search engine on the today's web, there will be more than one ontology even for the same domain on the Semantic Web. The reasons for this diversity are both technical and non-technical.

On the technical side, the task for which ontology is going to be used greatly influences ontology design. For example, an ontology supporting an application of pairing wines with food is unlikely to have properties describing numbers of bottles or their exact prices, which is something an ontology supporting an inventory application for a restaurant will need. Furthermore, one ontology may classify wines based on the grapes that are used to produce them and another may use the region that the wine comes from as the classification criterion.

On the non-technical side, there are often cultural, organizational, or administrative reasons why, for example, different departments in an organization might undertake their own ontology-development efforts. These reasons range from the NIH (not invented here) syndrome to practical considerations such as having current software depend heavily on a particular ontology.

Therefore, **integration of ontologies** is a major challenge and research issue on the Semantic Web.

Challenges in ontology integration

Some of the specific challenges in ontology integration that we must address in the near future are:

- finding similarities and differences between ontologies in automatic and semi-automatic way
- defining mappings between ontologies
- developing an ontology-integration architecture
- composing mappings across different ontologies
- representing uncertainty and imprecision in mappings

In the Semantic Web, there will be multiple ontologies that will be developed independently but will interact with one another. These ontologies might reuse other ontologies and therefore share some of their content and frame of reference. They may make some changes to ontologies they are reusing, declare equivalence between their terms and terms in other ontologies, and so on.

The first challenge is to find similarities and differences between the ontologies in automatic or semi-automatic way. Differences could be as simple as the use of synonyms for the same concept. For example, one ontology may use the term "vintage" and another may use the term "year". There could be differences in the way ontologies organize concepts. For instance, one ontology can classify all wines based on their color, having Red, White and Rosé as the top-level categories. Another ontology can have color as a property of the wine. It may never be possible to find all mappings between ontologies in a completely automatic way since some of the intended semantics can only be discerned by humans. However, ontology-integration on the large scale will be possible *only* if we can make significant progress in identifying mappings automatically or semi-automatically.

Researchers have already made some progress in this direction. For example, Hovy and colleagues (1998) describe a set of heuristics that researchers at ISI/USC used for semi-automatic alignment of domain ontologies to a large central ontology. Their techniques are based mainly on linguistic analysis of concept names and natural-language definitions of concepts. PROMPT (Noy & Musen 2003) uses the structure of ontology definitions and the structure of a graph representing an ontology to suggest to ontology designers which concepts may be related. GLUE (Doan *et al.* 2002) applies machine-learning techniques to instance data conforming to ontologies to find related concepts.

Once we find the mappings, we need to define a formalism for representing them that would enable and facilitate various tasks that use the mappings. These task include (but are not limited to) the following:

- answering queries posed to one ontology in terms of another ontology
- transforming instance data conforming to one ontology into another ontology
- using one ontology to drive an application developed based on another ontology

One approach to expressing the mapping information is to use the statements in the ontology language itself to express the correlation. OWL for example, has such statements as `owl:sameClassAs` and `owl:samePropertyAs` that allows one to "bridge" two ontologies. A reasoning engine can then treat two ontologies as a single theory. Another approach is to express mappings as instances of concepts in a mapping ontology. Crubezy (2003) for example have developed such an ontology, which enables specification of extremely expressive mappings, including ones that require recursive definitions. More research is needed however to determine which approaches would best support specific integration tasks.

The next research issue is finding an optimal architecture for ontology integration. One possible architecture could be similar to information-integration architectures in which there is a global ontology which serves as an interface to a number of local ontologies (Genesereth *et al.* 1997; Calvanese *et al.* 2001). Queries are posed to the global ontology which translates them to the terms in the local ontologies. The drawback of such an architecture is the need to develop and, more important, agree on the global ontology.

Another possibility is a peer-to-peer architecture in which we create pairwise mappings between ontologies (Halevy *et al.* 2003). Compared to the global-ontology architecture, the number of mappings that we need to create is n^2 where n is the number of ontologies, compared to n mappings to the global ontology. At the same time, the peer-to-peer architecture preserves the de-centralized nature of the Semantic Web. We may not always need to map between each pair of ontologies and therefore in practice the number of mapping can be significantly smaller than n^2.

Reusing the mappings leads to the problem of mapping composition. Suppose we have two mappings: one mapping is between ontologies A and B and another one is between ontologies B and C. Can we use these mappings to derive the mapping between ontologies A and C? Can we compose the mappings in computationally complete and efficient way?

In many cases, in particular when using automatic means to find mappings, we may not be able to define mappings precisely. Sometimes a precise mapping simply will not exist. For example, one classification of wines may only have red and white wines (classifying rosé wines as white wines). Another ontology may have a separate class for rosé wines. This class in the second ontology will not have an exact counterpart in the first. A precise mapping may exist but our means for finding it automatically will not be able to find it but will suggest several likely candidates instead. And in some cases, we do not need precise mappings and knowing that a class A in one ontology is a subclass of a class B in another is sufficient. Challenges in these area include not only classifying and representing different types and sources of imprecise and approximate mappings but also using this information for such tasks as discovering new mapping information or performing reasoning services across the mapped ontologies.

While researchers are actively working on some of these challenges in ontology integration, they have only scratched the surface. The unique scale, decentralization, and lack of central control in the Semantic Web require significant new advances to make ontology integration possible on the Web scale.

References

Calvanese, D.; Giacomo, G.; and Lenzerini, M. 2001. Ontology of integration and integration of ontologies. In *Description Logic Workshop (DL 2001)*, 10–19.

Crubézy, M. 2003. Mediating knowledge between application components. In *Workshop on Semantic Integration at ISWC-2003.*

Doan, A.; Madhavan, J.; Domingos, P.; and Halevy, A. 2002. Learning to map between ontologies on the semantic web. In *The 11th Intl WWW Conference.*

Genesereth, M. R.; Keller, A. M.; and Duschka, O. 1997. Infomaster: An information integration system. In *ACM SIGMOD Conference.*

Halevy, A. Y.; Ives, Z. G.; Suciu, D.; and Tatarinov, I. 2003. Schema mediation in peer data management systems. In *19th Intel Conf on Data Engineering (ICDE).*

Hovy, E. 1998. Combining and standardizing largescale, practical ontologies for machine translation and other uses. In *First International Conference on Language Resources and Evaluation (LREC)*, 535–542.

Noy, N. F., and Musen, M. A. 2003. The PROMPT suite: Interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies, in press.*

Position Statement

The Authors

Marco Schorlemmer. Dr. Marco Schorlemmer is a *Ramón y Cajal* Research Fellow at the International University of Catalonia. Until recently he was a Research Fellow at The University of Edinburgh's School of Informatics, where he was involved in the Advanced Knowledge Technologies project, a multi-million pound, six-year collaboration between internationally recognised research groups at five UK universities. He obtained his PhD in Artificial Intelligence in 1999 from the Technical University of Catalonia, carrying out his research at the Artificial Intelligence Research Institute in Spain, and at SRI International and Indiana University in the USA. Dr. Schorlemmer is mainly interested in applying mathematical techniques from theoretical computer science to challenging engineering problems faced by software and knowledge engineers today. In particular, he has been using techniques from category theory and information-flow theory to automate the semantic mapping of ontologies. He has also published over 20 papers in journals and international workshop and conference proceedings in the fields of Formal Specification & Automated Theorem Proving, Diagrammatic Representation & Reasoning, Life Cycles of Knowledge Engineering & Management, and Semantic Interoperability & Integration.

Yannis Kalfoglou. Dr. Yannis Kalfoglou is a Senior Research Fellow at the University of Southampton working on the Advanced Knowledge Technologies (AKT) project; a large scale Interdisciplinary Research Collaboration (IRC) between five UK universities funded by the British government. He received a First Class Honours B.Sc. degree in Computer Studies from Portsmouth University, England, in 1995 and a Ph.D. degree in Artificial Intelligence from Edinburgh University, Scotland, in 2000. His research interest is focusing on ontologies, in particular ontology-based services in a variety of application areas. He has researched extensively the intersection of software and knowledge engineering with emphasis on intelligent support for the early stages of software design when working with knowledge models. He has published over 30 papers on this and similar subjects and recently his focus is on ontology mapping and merging technologies. He is an active member of the EU funded OntoWeb & KnowledgeWeb thematic networks of excellence where he is working on industry-strength ontology tools and environments.

Our Research on Semantic Integration

Our approach draws heavily on proven theoretical work but our work goes further in providing a systematic approach for ontology mapping with precise methodological steps. In particular, our method, Information-Flow based Ontology Mapping (IF-Map) [2], draws on the proven theoretical ground of Information Flow and channel theory [1], and provides a systematic and mechanised way for deploying the approach in a distributed environment to perform ontology mapping among a variety of different ontologies.

The IF-Map system formalises mappings of ontology constructs in terms of logic infomorphisms, the fundamental ingredient of Information Flow. These are well suited for representing the bi-directional relation of types and tokens, which corresponds to concepts and instances in the ontology realm. IF-Map is focusing on instances and how these are classified against ontology concepts. This reveals the operational semantics that the ontology's community has chosen by virtue of how it uses its instances. The IF-Map algorithm makes use of this information in order to map onto related concepts from another ontology with which its concepts classify the same instances.

We have used IF-Map with success in a variety of ontology mapping scenarios within and outside AKT such as mapping of computer science departments' ontologies from AKT participating universities [3]; mapping of e-government ontologies from a case study using UK and US governments ministries [6]. We have also conducted a large-scale survey of the state-of-the-art of ontology mapping [4] and we are currently exploring the role of Information Flow and the IF-Map approach in the wider context of semantic interoperability and integration [5].

Challenging Issues on Semantic Integration

One of the core aspects on semantic integration and interoperability research nowadays is to find ways to share knowledge across diverse systems and domains and make them semantically interoperable. A key challenge and starting point for achieving this, is to have their ontologies shared. One aspect of ontology sharing is to perform some sort of mapping between ontology constructs. That is, given two ontologies, one should be able to map concepts in one ontology onto those in the other. Further, research suggests that we should also be able to combine ontologies where the product of this combination will be, at the very least, the intersection of the two given ontologies. These are the dominant approaches that have been studied and applied in a variety of systems [4].

There are, however, some drawbacks that prevent engineers from benefitting from such systems. Firstly, the assumptions made in devising ontology mappings and in combining ontologies are not always exposed to the community and no technical details are disclosed. This is an important hindrance for progress within this newly founded and diverse community as the less information is exposed about an allegedly problem-solving technique the more difficult becomes to replicate and experiment with it.

Secondly, the systems that perform ontology mapping are often either embedded in an integrated environment for ontology editing or are attached to a specific formalism. This makes it difficult to assess their performance outside these nicely designed "sandy-boxes" which act as a controlled environment and cannot accommodate the dynamism of ontologies available in the real world and being attached to a specific formalism precludes familiarity with it and availability of translators for making it possible to work in a large scale.

Thirdly, in most cases mapping and merging are based on heuristics that mostly use syntactic clues to determine correspondence or equivalence between ontology concepts, but rarely use the meaning of those concepts, i.e., their semantics. This is a big assumption as in most of the cases syntax alone can say little or nothing about the actual meaning of a concept. The intended semantics of concepts are not revealed and the proposed outcome has to be manually verified by a human expert.

Fourthly, most, if not all approaches do not exploit ontological axioms or rules often found in formal ontologies. This will allow for mathematical proofs to performed on the mapping outcome which will, at least, increase the assurance that the proposed mapping conforms with the underpinning ontological knowledge.

Finally, ontology mapping as a term has a different meaning in different contexts due to the lack of a formal account of what ontology mapping is. There is an observed lack of theory behind most of the works in this area [4].

References

[1] J. Barwise and J. Seligman. *Information Flow: The Logic of Distributed Systems*. Cambridge University Press, 1997.

[2] Y. Kalfoglou and M. Schorlemmer. IF-Map: An ontology-mapping method based on information-flow theory. *Journal of Data Semantics*, (1)1, 2003.

[3] Y. Kalfoglou and M. Schorlemmer. Information-flow-based ontology mapping. In *On the Move to Meaningful Internet Systems 2002: CoopIS, DOA, and ODBASE*, volume 2519 of *Lecture Notes in Computer Science*, pages 1132–1151. Springer, 2002.

[4] Y. Kalfoglou and M. Schorlemmer. Ontology mapping: The sate of the art. *Knowledge Engineering Review*, 18(1), 2003.

[5] M. Schorlemmer and Y. Kalfoglou. On semantic interoperability and the flow of information. In *ISWC'03 Semantic Integration Workshop*, Sanibel Island, Florida, USA, Oct. 2003.

[6] M. Schorlemmer and Y. Kalfoglou. Using information-flow theory to enable semantic interoperability. In *6è Congrés Català en Intel·ligència Artificial*, Palma de Mallorca, Spain, Oct. 2003. Also available as Informatics Report EDI-INF-RR-0161, The University of Edinburgh.

Challenges in Making Context Interchange Part of the Semantic Web

Michael Siegel (msiegel@mit.edu)
Stuart Madnick(smadnick@mit.edu)
Aykut Firat (aykut@mit.edu)
Allen Moulton(amoulton@mit.edu)
Hongwei Zhu (mrzhu@mit.edu)

MIT Sloan School of Management
30 Wadsworth Street, Cambridge, MA 02142, USA
http://context2.mit.edu/coin

1. Introduction

The popularity and growth of the Web have dramatically increased the number of information sources available for use and the opportunity for important new information-intensive applications (e.g., massive data warehouses, integrated supply chain management, global risk management, in-transit visibility). Unfortunately, there are significant challenges to be overcome regarding *data interpretation*. Specifically, the existence of heterogeneous contexts, whereby each source of information and potential receiver of that information may operate with a different context, leading to large-scale semantic heterogeneity.

A context is the collection of implicit assumptions about the meaning of data. As a simple example, whereas most US universities grade on a 4.0 scale, MIT uses a 5.0 scale – posing a problem if one is comparing student GPA's. Another typical example might be the extraction of price information from the Web: but is the price in Dollars or Yen (If dollars, is it US dollars or Hong Kong dollars), does it include taxes, does it include shipping, etc. – and does that match the receiver's assumptions?

Contextual issues can be much more complex in other situations. For example, the meaning of "net sales" may vary – with "excise taxes" included for government reporting purposes in one context, but excluded for security analysis purposes in another. Also, one context may use information for a fiscal year as reported by the company, while another may use a standardized fiscal year to make all companies comparable. Furthermore, there may be multiple users that might want an answer to such a question, each with their own desired meaning (user context).

This context knowledge is often widely distributed within and across organizations. Solutions adopted to achieve interoperability must be scaleable and extensible. Thus, it is important to support the acquisition, organization, and effective intelligent usage of distributed context knowledge.

The COntext INterchange (COIN) System has been designed and implemented as a prototype at MIT. The prototype provides for a systematic representation and automatic processing of data semantics. Instead of explicitly capturing semantic conflicts, the COIN approach records data semantics declaratively and uses a mediation engine to detect all conflicts, which are reconciled by rewriting user queries to incorporate conversions that can be defined either internally or remotely on the network. This approach provides great extensibility. We refer readers to [1,2] for a formal description of the COIN approach.

2. Recent Developments in COIN

Recent developments by Firat [10] have provided a clear definition of concepts such as *context, conversion function, and ontology*. His work also resolved issues in *equational ontological conflicts* (EOC) that refer to the heterogeneity in the way data items are calculated from other data items in terms of definitional equations. Firat along with others at MIT have developed an approach to merging independently developed, ontology based COIN *applications*. Finally, there have been significant developments in providing for semantic integration using COIN on the Semantic Web. Specifically we have developed ways to make the context mediation approach compatible with web protocols (as in web services) and web-oriented representation languages such as RDF and OWL)[4,5].

We have demonstrated these new capabilities in a number of application domains, such as financial services [6], online shopping [9], disaster relief efforts [3], and corporate householding knowledge engineering [8]. We have also constructed larger applications by combining ontologies and context definitions from existing applications, such as an airfare aggregator and a car rental shopper combined into a travel planner (see demos at our website http://context2.mit.edu/coin/demos).

Efforts are also underway to use COIN framework as a cost effective method for resolving semantic ambiguities and differences to support semantic interoperability across multiple overlapping standards in the financial industry [7].

3. Making Context Mediation Ubiquitous on the Web: The Challenges

Our approach to semantic integration is data-oriented. As such, our goals are far more focused than many other visions of the potential for the Semantic Web. As a result, we are able to treat context interchange problems inherent in the Semantic Web in a tractable manner. For example, we

have a specific approach to merging ontologies that supports the merging of applications. This merging raises many of the issues that others have looked at but is nicely tied to the data requirements for new applications and focused on providing the context information needed to resolve semantic differences.

This focus on context knowledge and data integration has allowed us to make significant progress, however, challenges exist in making such an approach scalable, maintainable and usable in an open environment. We conclude this position paper with a number of these issues:

1. **Gathering, Representing and Maintaining Context Knowledge for Unknown Tasks** – Context Interchange capabilities have been used for specific applications. Though the semantic integration can be done at run-time for such an application, ad-hoc environments without predefined schemas and context knowledge will be more difficult to manage.
2. **Designing Ontologies to Include Context Knowledge** – We have developed ontology to support context knowledge. We have extended ontologies developed in RDF to include modifiers and other context information. However, we expect a wide range of ontology languages and representations. Context information must either be easily extracted from these ontologies or added through the use of context-authoring tools as developed on this project.
3. **Making Context Mediation Executable in non-SQL like environments** – We have taken a distinctly database-like approach to semantic integration on the Web. We developed data extraction tools that gather semi-structured data based on SQL queries issues to Web pages (along with structured data). Methods must be developed to include mediation in for other data representations.
4. **Automatically Gathering, Generating and Maintaining Context Knowledge** – Tools are needed to automatically assemble and maintain context knowledge.
5. **Complex Context Issues** – There remain a number of complex context issues related to temporal context, equational context, and partially resolvable context conflicts.

References

1. Goh, C.H. (1997) Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Systems, PhD Thesis, MIT Sloan School of Management
2. Goh, C.H., Bressan, S., Madnick, S., Siegel, S. (1999) "Context Interchange: New Features and Formalisms for the Intelligent Integration of Information", *ACM Transactions on Information Systems,* **17**(3), 270-293
3. Lee, P.W. (2003) Metadata Representation and Management for Context Mediation, MIT EECS Master's Thesis.
4. Liburd, S. (2003) Conceptual Mapping and Structural Conversions between COIN and RDF, technical report, MIT Sloan School of Management.
5. Manola, F. Miller, E. (2003) "RDF Primer", W3C working draft.
6. Firat, A., Madnick, S., Grosof, B. (2002) Financial Information Integration in the Presence of Equational Ontological Conflicts, *12th WITS*, December 14-15, Barcelona, Spain.
7. Moulton A., Siegel, M.D., Madnick, S.E. (2003) Potential for Cost Savings from Application of Context Mediation Technology to Problems of Standards Interoperability in the Financial Services Industry, internal report, MIT Sloan School of Management.
8. Xian, X. (2003) Corporate Householding Knowledge Engineering and Processing using Extended COIN, MIT EECS Master's Thesis.
9. Zhu, H. Madnick, S., Siegel, M. (2002) "Global Comparison Aggregation Services", *1st Workshop on E-Business*, December 14-15, Barcelona, Spain.
10. Firat, A. (2003) "Information Integration Using Contextual Knowledge and Ontology Merging" PhD Thesis, MIT Sloan School of Management.

Consensus Ontologies for Semantic Reconciliation

Larry M. Stephens and Michael N. Huhns
Department of Computer Science and Engineering
University of South Carolina, Columbia, SC, 29208, USA

Position Statement

Small, independently-developed ontologies can be related *without* the necessity of constructing a global ontology beforehand. We assume that Web pages for specific domains will be annotated with ontological information. Given these annotations, we have developed a methodology that merges individual ontologies into what we call a *consensus ontology*, which has the appearance of a global ontology for a particular query. The consensus ontology may be cached for later use. This approach allows consideration of additional information sources incrementally.

Information Retrieval

Information searches can involve data and documents both internal and external to an organization. The research reported at this workshop targets the following basic problem: a search will typically uncover a large number of independently developed information sources—some relevant and some irrelevant; the sources might be ranked, but they are otherwise unorganized, and there are too many for a user to investigate manually. The problem is familiar and many solutions have been proposed, ranging from requiring the user to be more precise in specifying search criteria, to constructing more intelligent search engines, or to requiring sources to be more precise in describing their contents. A common theme for all of the approaches is the creation, use, and manipulation of ontologies for describing both requirements and sources.

Unfortunately, ontologies are not a panacea unless everyone adheres to the same one, and no one has yet constructed an ontology that is comprehensive enough. Moreover, even if one did exist, it probably would not be adhered to, considering the dynamic and eclectic nature of the Web and other information sources.

There are three approaches for relating information from large numbers of independently managed sites: (1) all sites will use the same terminology with agreed-upon semantics (improbable), (2) each site will use its own terminology, but provide translations to a global ontology (difficult, and thus unlikely), and (3) each site will have a small, local ontology that will be related to those from other sites—our position.

The experimental methodology we developed relies on the ontological annotation of Web pages—a representation consistent with visions for the Semantic Web. However, a pre-existing global ontology is not required. The domains of the sites must be similar—else there would be no interesting relationships among them—but they will undoubtedly have dissimilar ontologies, because they will have been annotated independently.

Experimental Methodology

We assigned graduate students in computer science the task of constructing small ontologies for three domains. The ontologies are written in DAML/OWL and contain at least 8 classes organized with at least 4 levels of subclasses.

We merge the individual files for a particular domain one-at-a-time into a resultant merged file. Node merging is based on syntactic and semantic information. The syntactic information is derived from the names of the nodes, for which we employ various string-matching techniques including detection of plural endings. The semantic information includes the meaning of the subclass link in the ontologies, prefixes that indicate antonyms, and evolving sets of synonyms for matching nodes. The synsets, which are used to track the progress of merging and to monitor correctness, are seeded from WordNet.

For each node in the resultant file, we maintain a *reinforcement* value, which indicates how many times the node is matched as ontologies are merged. We also maintain reinforcement values for class-subclass links. Next, we construct a *consensus ontology* by eliminating weakly reinforced nodes and links. In filtering the merged file, we sort the subclass *links* by their reinforce-

ment values and find that, for the most part, the strongly reinforced nodes are associated with strongly reinforced links. This finding, while not surprising, makes constructing a consensus ontology more efficient.

The software for merging ontologies can be found at http://www.cse.sc.edu/research/cit/projects/DAML.html. Sample ontologies are also available on that page.

Challenges

Our work focuses only on the *class-subclass* relationship among concepts. Other relationships such as *partOf* offer semantics that can be exploited in generating and restricting a consensus view.

Biographies

Larry M. Stephens received the B.S. degree in electrical engineering from the University of South Carolina in 1968 and received the M.S. and Ph.D. degrees in electrical engineering from the Johns Hopkins University in 1974 and 1977, respectively. He is currently Professor of Computer Science and Engineering at the University of South Carolina and a member of the Center for Information Technology. Prior to his academic career, he served for four years as a U.S. Naval officer assigned to the Naval Reactors Program, Washington, DC.

From 1988 to 1989 he was on leave as a consultant to the Microelectronics and Computer Technology Corporation (MCC), Austin, Texas. At MCC he conducted research on distributed knowledge-based systems, reasoning architectures for synthesis tasks, and plausible inferencing based on the properties of semantic relations. At MCC, he did extensive work with the Cyc common-sense reasoning project. His current research interests include the fundamentals of knowledge representation, design of ontologies, and information retrieval from heterogeneous information sources.

Michael N. Huhns received the B.S.E.E. degree in 1969 from the University of Michigan, Ann Arbor, and the M.S. and Ph.D. degrees in electrical engineering in 1971 and 1975, respectively, from the University of Southern California, Los Angeles. He is a Professor in the Department of Computer Science and Engineering at the University of South Carolina, where he also directs the Center for Information Technology. Prior to this, he conducted research on the Argo, Antares, RAD, Carnot, and InfoSleuth projects at the Microelectronics and Computer Technology Corporation as a Senior Member of the Research Division. He was also an adjunct professor in computer sciences at the University of Texas. Before joining MCC, he was an associate professor at the University of South Carolina, a research assistant in image processing at the University of Southern California, and a radar systems engineer at Hughes Aircraft Company.

Dr. Huhns is currently teaching courses in information technology and conducting research in cooperative information systems, ontologies, cooperative information systems, software agent systems, and bioinformatics.

Semantic Matching in the SWAP Project

KR & R Group, Vrije Universiteit Amsterdam

1 Semantic Web and P2P

The current state-of-the-art in Knowledge Management solutions still focuses on one or a relatively small number of highly centralized knowledge repositories with ontologies as the conceptual backbone for knowledge brokering. As it turns out, this assumption is very restrictive, because, (i), it creates major bottlenecks and entails significant administrative overheads, especially when it comes to scaling up to large and complex problems; (ii), it does not lend itself to easy maintenance and the dynamic updates often required to reflect changing user needs, dynamic enterprise processes or new market conditions. In contrast Peer-to-Peer computing (P2P) offers the promise of lifting many of these limitations. At the same time, today's P2P solutions support only limited update, search and retrieval functionality, e.g. search in Napster is restricted to string matches involving just two fields: "artist" and "track". These flaws however make current P2P systems unsuitable for knowledge sharing purposes. The SWAP project aims at a P2P based knowledge management system that integrates the advantages of Semantic Web-based knowledge management technology developed in successful IST projects like On-To-Knowledge, KnowNet, or Comma. SWAP aims at benefits of a P2P based system that show just by installing the client software, viz. immediate automatic access to knowledge stored at peers. Of course, explicitly modelled ontologies may increase the benefits brought by any knowledge management solution, because they may improve the accuracy of knowledge access and sharing. SWAP solutions, however, may produce benefits even with near zero investment - in contrast to conventional knowledge management systems that need an extensive and expensive set-up phase. Conventional knowledge management repositories will still appear as just another, powerful peer in the network. Hence, a combined Semantic Web and P2P solution may always outperform the sophisticated, but conventional centralized system.

2 Semantic Matching

Information sharing in semantics-based P2P systems relies on the existence of mappings between the semantic models of different peers. The distributed and dynamic nature of P2P systems makes it unattractive to spend effort on creating fixed mappings manually. Instead, there is a need for automatic or semi-automatic matching algorithms that establish a connection between the semantic models of different peers on the fly. In this context, research in the SWAP project will be focussed on the following question: Are automatic matching methods powerful enough to support P2P information sharing in a practical setting ? In order to answer this question, the following research activities will be carried out by the academic partners in the SWAP project:

Assessment of Matching Methods A number of different approaches for matching semantic models have been proposed. These methods differ in the way they identify correspondences between models including, manual matching, lexical matching, structural matching, semantic matching and similarity-based matching. As a first step we will define a set of benchmark matching problems and compare the results of these different methods wrt. these problems. It is easy to see that one matching approach is not always better than another, but that certain matching approaches are better suited for certain matching problems. Based on the results of the evaluation of different matching approaches we will try to formulate guidelines for selecting a certain matching approach to solve a specific matching problem. For this purpose, we first have to identify relevant characteristics of the matching problem and relevant features of the matching method as well as their interaction.

Optimization of Matching Methods Initial experiments with real life data has shown that many matching algorithms are inherently complex and often fail to scale up to realistic scenarios. If we want to use matching in a practical setting, we have to make sure that the response time is still acceptable for the user. In this context, we will investigate the use of approximate matching methods instead of exact ones. In particular, out goal is to find the right trade-off between the run-time behavior of a matching algorithm and the accuracy of the matching result. For this purpose we will carry out experiments with different approximations of the same matching algorithm and compare the results wrt. matching accuracy and runtime. Based on the experiences gained in this experiments, we plan to develop efficient algorithms for semantic matching and implement them in an optimized way.

Position Statement

Nancy Wiegand, Isabel Cruz, Naijun Zhou, and William Sunna

Nancy Wiegand, Isabel Cruz, Naijun Zhou, and William Sunna are working together on a semantic integration project. Nancy Wiegand is a Research Scientist at the University of Wisconsin in Madison. Although her main focus is in Computer Science Database Management Systems, her background also includes interdisciplinary work in Geographic Information Systems, Civil Engineering, and Environmental Studies. Isabel Cruz is a Professor of Computer Science at the University of Illinois at Chicago. Naijun Zhou and William Sunna are graduate students at UW-Madison and Illinois, respectively. Naijun Zhou is a Ph.D. candidate in Geography and also has a Master's degree in Computer Science. William Sunna is studying Computer Science.

Our Computer Science and interdisciplinary backgrounds are being applied to a research project for a proposed Web-based statewide Wisconsin Land Information System. We are working on semantic integration over distributed, heterogeneous spatial and nonspatial data sets to enable DBMS-type querying. Our goal of DBMS querying is an extension of the clearinghouse vision of the original working group. Our research also includes methods for locating data sets and consideration of separate metadata files that describe data sources.

We developed a tool to map theme-based ontologies to local schemas, and, in particular, included the ability to map at the value level, in addition to the attribute level. This was necessary because various attributes in our data sources are conceptually similar, but their values are drawn from domains that differ in detail and expression. The mapping tool automatically produces agreement files which are consulted by an ontology subsystem for query re-writing. The ontology subsystem is embedded in a prototype XML Web-based query engine.

A hard problem in semantic integration is to provide easy extensibility for new ways of thinking about and relating information. Also, users should be able to trace and validate any automatically made semantic integration decisions to be able to confidently use results for decision-making.

Grass-roots Semantic Web Tools

Baoshi Yan, Robert MacGregor,
In-Young Ko, Juan Lopez
Distributed Scalable Systems Division
Information Sciences Institute
University of Southern California
{baoshi,macgregor,iko,juan}@isi.edu

ABSTRACT

One of the biggest challenges of the Semantic Web is to make its tools usable by ordinary users for grass-roots production and integration of semantic information. This paper introduces the ongoing research on this issue in our research group at the Information Sciences Institute.

1. RESEARCH OVERVIEW

Despite years of intense work and research on the Semantic Web, it has not become a reality. One of the biggest challenges is to make Semantic Web tools usable by ordinary users. Current tools for ontology creation, annotation, ontology alignment, and querying heterogeneous data sources are still too difficult for ordinary users. In this paper we'll discuss the various ongoing efforts in our research group aiming at creating Semantic Web tools that further lower the entrance barrier to Semantic Web for ordinary users.

1.1 Grass-roots Annotator

Metadata is the basis of the Semantic Web. There has been great effort on making metadata creation [1][6] easier for ordinary users. All these tools follow the same pattern: users are required to create an ontologies first, and then make annotations according to the created ontologies. However, ontology creation is an abstract activity, which is often difficult and unintuitive for ordinary users. As a result these tools are still difficult for ordinary users to use.

We are experimenting an extreme approach. Our Grass-roots Annotator (Figure 1) would allow users to create metadata first without creating any ontology. Users would be allowed to use whatever structures and terms they like to describe their data at hand without first defining these terms and structures. We would then try to induce ontologies from the metadata corpus.

The annotator is carefully designed so that some operations are indicative of possible ontologies. We are also developing techniques to mine the metadata corpus for patterns which indicates the existence of ontologies. Furthermore, our own experience with the tool shows that, with the metadata corpus growing, we tend to use same terminologies and structures to describe similar things in order to make it easier to manage the metadata. This indicates that it might be easier for users to generalize ontologies from the data they created than to create an ontology from scratch.

.

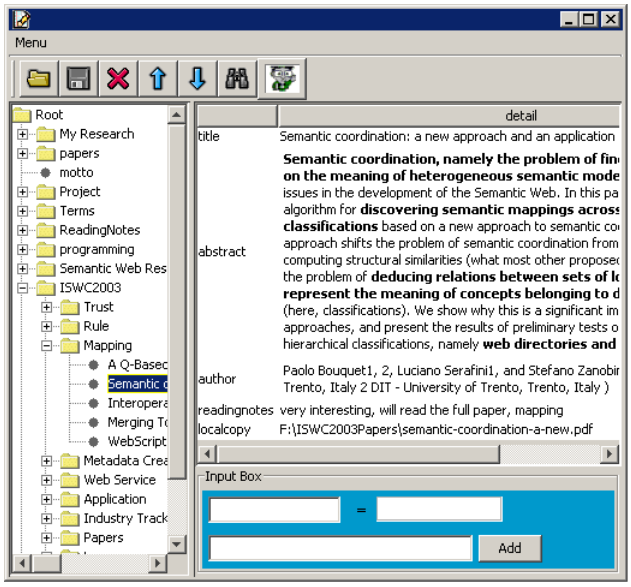


Figure 1: Grass-roots Annotator

1.2 WebScripter: Grass-roots Report Creation and Ontology Alignment

WebScripter[2](Figure 2) is a tool that enables ordinary users to easily and quickly assemble reports extracting and fusing information from multiple, heterogeneous Semantic Web sources. Different Semantic Web sources may use different ontologies. WebScripter addresses this problem by (a) making it easy for individual users to graphically align the attributes of two separate externally defined concepts, and (b) making it easy to reuse others' alignment work. At a high level, the WebScripter concept is that users extract content from heterogeneous sources and paste that content into what looks like an ordinary spreadsheet. What users implicitly do in WebScripter (without expending extra effort) is to build up an articulation ontology containing equivalency statements. We believe that in the long run, this articulation ontology will be more valuable than the data the users obtained when they constructed the original report. The equivalency information reduces the amount of work future WebScripter users have to perform.

The key difference we see between "traditional" ontology translation and WebScripter is that non-experts perform all of the translation - but potentially on a global scale, lever-

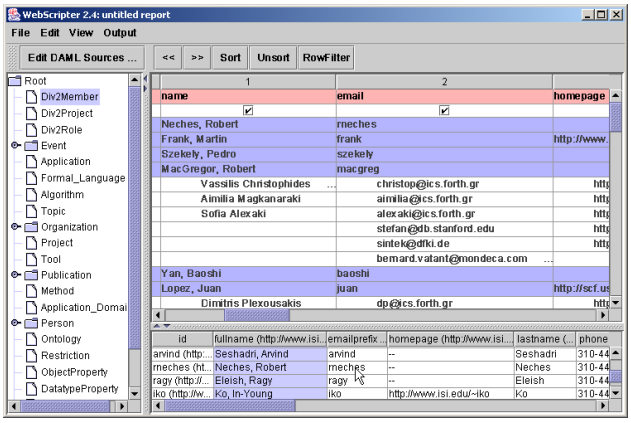


Figure 2: WebScripter Tool

aging each other's work.

1.3 Naive User Queries

Traditionally, users need to write queries conforming to the schema of a data source in order to retrieve information from it. On the Semantic Web, there will be numerous data schemas. Requiring people to write different queries for different schemas is a daunting task. Thus we propose that it's necessary to deal with another type of user queries: *naive user queries*–queries in users' own terms and own semantic structures. Without losing generality, we represent a naive user query as a list of triple patterns (s,p,o) (Although syntax doesn't affect our discussion, we use RDQL-alike [5] syntax for convenience). Semantic structures between terms are binary relations: p is the kind of relationship between s and o. Such type of user queries might not conform to the schemas of available data sources.

We propose an approach [8] that, given a naive user query, translates it into a list of queries conforming to different data source schemas. The approach is based on query-rewriting techniques. It utilizes partial alignment between different schemas, alignment between different naive user queries, similarities between term names, as well as other information as query rewriting rules. An early prototype showed that the result is promising.

1.4 Semantic Engineering Workbench (SEW)

ISI's n-Dimensional Information Management project is developing an integrated suite of tools, called the Semantic Engineering Workbench (SEW)[3][7](Figure 3), that provides an intelligent infrastructure for managing Semantic Web databases and developing Semantic Web applications. The SEW has been crafted by integrating key (open-source) software components into an integral whole. Retrieval capabilities and persistence is provided by combining Hewlett-Packard's Jena triple store with a relational database (we are currently using MySQL). Ontology editing is provided by Stanford's Protege Knowledge Acquisition tool. The SEW implements several layers of API's. The highest levels provide object-oriented representations of data objects, while lower-levels enable access to triples. The SEW transparently converts triples retrieved from Jena into Protege objects, using an on-demand strategy that imports data on an as-requested basis. The SEW is wholly implemented in Java, and currently runs on Windows PCs.

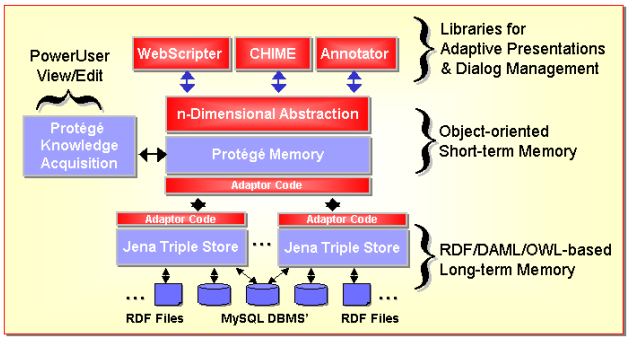


Figure 3: Semantic Engineering Workbench

The design of the SEW was motivated by the need to provide high-level support to three Semantic Web applications: the Annotator and WebScripter tool as explained in Sections 1.1 and 1.2, and the CHIME tool [4] that allows users to view n-dimensional data.

2. CONCLUSION

We've briefly introduced several research projects in our group including Grass-roots Anntator–an extremely easy-to-use tool for metadata creation, WebScripter–a tool for grass-roots report generation and ontology alignment, Naive User Query Processing–a technique to enable queries in users' own terms and structures, and finally Semantic Engineering Workbench–the infrastructure underlying all our tools.

3. ACKNOWLEDGMENTS

Effort sponsored by the Defense Advanced Research Projects Agency (DARPA) under agreement number F30602-00-2-0576, and the Advanced Research and Development Activity (ARDA) under contract number NMA401-02-1-2019.

4. ABOUT AUTHORS

Robert Macgregor is a senior project leader at University of Southern California's Information Sciences Institute. He is the project leader of the WebScripter and CHIME projects. Baoshi Yan and Juan Lopez are Ph.D. students participating in the WebScripter project. In-Young Ko is a post-doctoral research associate working on the WebScripter and CHIME projects.

5. REFERENCES

[1] http://protege.stanford.edu/.
[2] http://www.isi.edu/webscripter/.
[3] http://www.isi.edu/chime/sew.html.
[4] http://www.isi.edu/chime/.
[5] Rdql. http://www.hpl.hp.com/semweb/rdql.htm.
[6] S. Handschuh and S. Staab. Authoring and annotation of web pages in cream. In *The World Wide Web Conference*, 2002.
[7] R. Macgregor and I.-Y. Ko. Representing contextualized data using semantic web tools. In *1st International Workshop on Practical and Scalable Semantic Systems*, Octor 2003.
[8] B. Yan and R. Macgregor. Translating naive user queries on the semantic web. In *Semantic Integration Workshop*, Octor 2003.