TOWARDS AUTOMATIC GENERATION OF APPLICATION ONTOLOGIES

Eveline R. Sacramento, Vânia M. P. Vidal, José Antônio F. de Macêdo, Bernadette F. Lóscio, Fernanda Lígia R. Lopes, Fernando Lemos
Department of Computing, FederalUniversity of Ceará, Fortaleza-CE, Brazil
{eveline, vvidal, jose.macedo, bernafarias, fernanda.ligia, fernandocl}@lia.ufc.br

Marco A. Casanova
Department of Informatics, PUC-Rio, Rio de Janeiro-RJ, Brazil
casanova@inf.puc-rio.br

Keywords: semantic heterogeneity, ontologies, ontology matching, data integration, schema mappings, rules.

Abstract: In the Semantic Web, domain ontologies can provide the necessary support for linking together a large number of heterogeneous data sources. In our proposal, these data sources are describe as local ontologies using an ontology language. Then, each local ontology is rewritten as an application ontology, whose vocabulary is restricted to be a subset of the vocabulary of the domain ontology. Application ontologies enable the identification and the association of semantically corresponding concepts, so they are useful for enhancing tasks like data discovery and integration. The main contribution of this work is a strategy to automatically generate such application ontologies and mappings, considering a set of local ontologies, a domain ontology and the result of the matching between each local ontology and the domain ontology.

1 INTRODUCTION

The Web is a complex and vast repository of information that is often stored in heterogeneous and distributed data sources. Problems that might arise due to heterogeneity of the data are already well known within the database community: syntactic heterogeneity and semantic heterogeneity.

In nearly all recent researches on data integration, ontologies provide a possible approach to address the problem of semantic heterogeneity. In general, two architectures for data integration can be identified: *two-level* and *three-level ontology-based architectures.*

The main components of the *two-level architecture* (Figure 1(a)) are: the domain ontology (DO); the local ontologies (LO), which describe the data sources using an ontology language; and the mapping that specifies the correspondences between the local ontologies and the domain ontology (LO-DO mappings). The work presented in (Calvanese et al., 2007) adopts this architecture. The main components of the *three-level architecture* (Figure 1(b)) are: the domain ontology (DO); the local ontologies (LO); the application ontologies (AO), which rewrite the local ontologies using a *subset* of the vocabulary of the domain ontology; the mapping that specifies the correspondences between the application ontologies and the domain ontology (AO-DO mappings); and the mapping that specifies the correspondences between the local ontologies and the application ontologies (LO-AO mappings). The work of (Lutz, 2006) adopts this architecture.

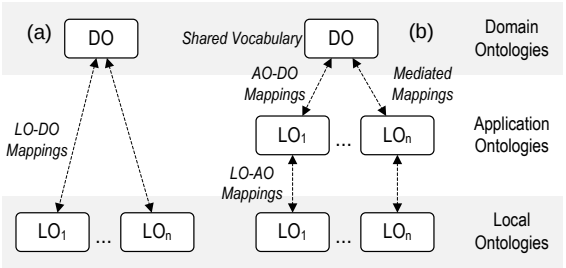

Figure 1: (a) Two-Level Ontology-Based Architecture. (b) Three-Level Ontology-Based Architecture.

The main problems that concern to both architectures are: (i) how to specify the mappings; and (ii) how to use the mappings to answer correctly the queries posed on the domain ontology. In the two-level architecture, the domain ontology is *only*

used for specifying the mediated schema. So the user has to define, which we call *heterogeneous mappings*, between entities of the local ontologies and the domain ontology, as such ontologies do not share the same vocabulary and also because of the structural heterogeneity. In the three-level architecture, the domain ontology is used for *both* specifying the mediated schema and as a shared vocabulary. As the application ontologies are *subsets* of the domain ontology, the user can define *homogeneous mappings* between these ontologies.

In our approach, application ontologies are used to divide the definition of the mappings into two stages: AO-DO mappings and LO-AO mappings. We use mediated mappings to define the classes and properties of the domain ontology in terms of the vocabularies of the application ontologies. The AO-DO and the mediated mappings are represented using a Description Logics (DL) formalism (Calvanese et al. 1998) to take advantage of ontological reasoning tasks. Since, we need to represent object restructuration; the LO-AO mappings are expressed in an extended rule-based formalism to overcome DL limitations.

This paper is organized as follows. Section 2 gives some definitions and presents an example. Section 3 presents concepts about ontology matching. Section 4 describes our approach for generating application ontologies and mappings.

2 BASIC DEFINITIONS

We use *extralite schemas* (Leme et al., 2009) that supports the definition of *classes* and *properties*, and that admit *domain* and *range* constraints, *subset* and *disjoint* constraints, *minCardinality* and *maxCardinality* constraints, with the usual meaning.

We present an example, adapted from (Casanova et al., 2009) of a virtual store mediating access to online booksellers. The user provides a domain ontology, describing data about virtual sales of products; and two local ontologies describing data about Amazon and eBay virtual stores. We use the namespace prefixes "*s:*", "*a:*" and "*e:*" to refer to the vocabulary of *Sales* domain ontology (Figure 2(a)); *Amazon* and *eBay* local ontologies (Figure 2(b)).

3 OWL SCHEMA MATCHING

Ontology matching is the process of finding correspondences between semantically related entities of different ontologies (Euzenat and Shvaiko, 2000). In the following, we present the two main steps of our strategy to the generation of the application ontologies, adapted from (Leme et al., 2009): (1) *vocabulary matching*, which generates the alignment between entities of two different ontologies; and (2) *concept mapping*, which induces the mapping rules from the ontology alignment.

3.1 Vocabulary Matching

Let O_S and O_T be two ontologies, and V_S and V_T be their vocabularies, respectively. Let C_S and C_T be the sets of classes and P_S and P_T the sets of datatype or object properties in V_S and V_T, respectively. A contextualized *vocabulary matching* (Leme et al., 2009) between the *source* ontology O_S and the *target* ontology O_T can be represented by a finite set Q of *quadruples* (v_1, e_1, v_2, e_2) such that: (i) if $(v_1, v_2) \in C_S \times C_T$, then e_1 and e_2 are the top class $\top$; and (ii) if $(v_1, v_2) \in P_S \times P_T$, then e_1 and e_2 are classes in C_S and C_T that must be subclasses of the domains, or the domains themselves, of v_1 and v_2, respectively.

If $(v_1, e_1, v_2, e_2) \in Q$, we say that: (i) Q matches v_1 with v_2 in the context of e_1 and e_2; (ii) e_i is the *context* of v_i; and (iii) (e_i, v_i) is a contextualized concept, for $i = 1, 2$.

Even though we do not focus on how these correspondences are created, we are aware that the correspondences obtained using an existing tool are often *incomplete* or *incorrect*; therefore, a user interaction might be necessary. Figures 3(a) and 3(b) show the vocabulary matching.

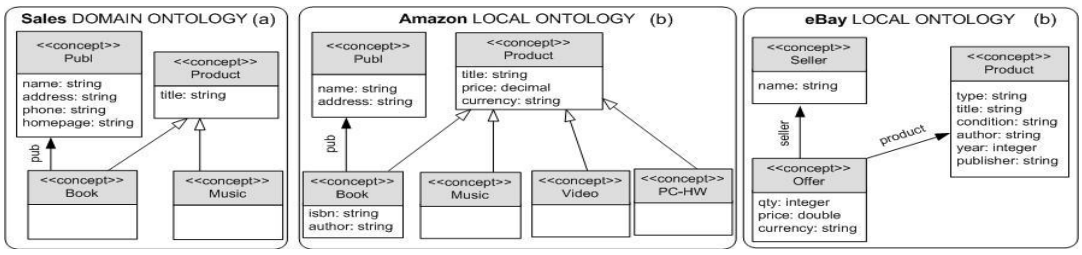

Figure 2: (a) Domain Ontology; (b) Local Ontologies.

3.2 Concept Mapping

In this work, c*oncept mapping* is induced from the vocabulary matching between ontologies. In general, a concept mapping from O_S into O_T is a set of expressions that define concepts of O_T in terms of concepts of O_S, in such a way that they semantically correspond to each other (Leme et al., 2009).

Concept mappings are usually represented by formalisms that deal with homogeneous mappings. DL, for example, can be used for inferring implicit taxonomic relationships between concepts or between concepts and individuals. However, it presents some limitations: DL cannot express ternary predicates and it does not define suitable mechanisms for the explicit building of object identifiers (OIDs). As both features are important in our approach, we use a Datalog variant with OID-invention (Hull and Yoshikawa, 1990) to represent concept mappings.

In the following definition consider that: (i) every variable v is a term; (ii) every constant c is a term; (iii) if $t_1, \ldots, t_n$ are terms, and f is an n-ary function symbol, then $f(t_1, \ldots, t_n)$ is a term.

Let O_S and O_T be two ontologies and R be a rule language. A concept mapping is specified through a set of *mapping rules*, each one of the form: $\beta_1(w_1) \Leftarrow \alpha_1(v_1),\ldots, \alpha_m(v_m)$ where $\alpha_1(v_1),\ldots, \alpha_m(v_m)$, called the *body* of the mapping, is an atom or a atom conjunction, where an atom α_i can be an atomic concept or an atomic role occurring in the source ontology O_S, and v_i is a sequence of terms; and $\beta_1(w_1)$, called the *head* of the mapping, is an atom that can be an atomic concept or an atomic role occurring in the target ontology O_T, and w_1 is a sequence of terms. This rule-based formalism supports *Skolem functions* (Hull and Yoshikawa, 1990) for the creation of OIDs of entities in O_T from one or more entities of O_S. In our work, the Skolem functions are simply used as URIref generators.

4 GENERATING APPLICATION ONTOLOGIES AND MAPPINGS

Given a local ontology LO, a domain ontology DO, a set of quadruples representing the vocabulary matching between LO and DO, our algorithm generates: (i) classes and properties of AO; (ii) a set of LO-AO mapping rules; and (iii) a set of mediated mappings. The algorithm checks if each quadruple satisfies one of the conditions of Table 1, in order to apply the corresponding actions. It follows the order of the cases listed in this table, and it is deterministic, as the number of quadruples is finite.

We now show the results obtained from the execution of our algorithm. Figure 4 shows the application ontologies. We use the namespace prefixes "*ap*:" and "*ep*:" to refer to the vocabularies of *Amazon* and *eBay* application ontologies, respectively.

Amazon		Sales	
a:title	a:Book	s:title	s:Book
a:pub	a:Book	s:pub	s:Book
a:Book	⊤	s:Book	⊤
a:title	a:Music	s:title	s:Music
a:Music	⊤	s:Music	⊤
a:name	a:Publ	s:name	s:Publ
a:address	a:Publ	s:address	s:Publ
a:Publ	⊤	s:Publ	⊤

Figure 3(a): Vocabulary matching between *Amazon* local ontology and *Sales* domain ontology.

eBay		Sales	
e:title	e:Product	s:title	s:Product
e:Product	⊤	s:Product	⊤
e:publisher	e:Product	s:name	s:Publ

Figure 3(b): Vocabulary matching between *eBay* local ontology and *Sales* domain ontology.

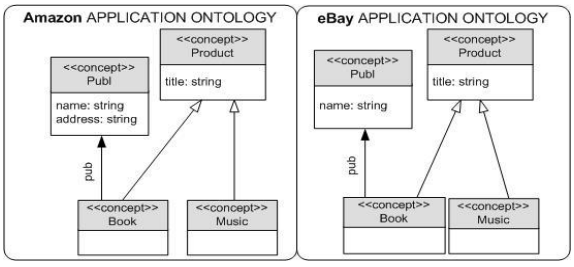

Figure 4: Application Ontologies.

```
#1: ap:Book(b) ⇐ a:Book(b)
#2: ap:Product(b) ⇐ a:Book(b)
#3: ap:Music(m) ⇐ a:Music(m)
#4: ap:Product(m) ⇐ a:Music(m)
#5: ap:Publ(p) ⇐ a:Publ(p)
#6: ap:title(b,t) ⇐ a:title(b, t), a:Book(b)
#7: ap:pub(b,p) ⇐ a:pub(b, p)
#8: ap:title(m,t)⇐ a:title(m, t), a:Music(m)
#9: ap:name(p, n) ⇐ a:name(p, n)
#10: ap:address(p,a)⇐ a:address(p, a)
```

Figure 5(a): Mapping rules from the *Amazon* local ontology to the *Amazon* application ontology.

```
#1:ep:Book(p) ⇐ e:Product(p),e:type(p,´book´)
#2:ep:Product(p)⇐ e:Product(p),e:type(p,´book´)
#3:ep:Music(p)⇐ e:Product(p),e:type(p,´music´)
#4:ep:Product(p)⇐ e:Product(p),e:type(p,´music´)
#5:ep:title(p,t)⇐ e:title(p,t),e:type(p,´book´)
#6:ep:title(p,t)⇐ e:title(p,t),e:type(p,´music´)
#7:ep:Publ(fpubl(n))⇐e:publisher(b,n),e:type(b,´book´)
#8:ep:name(fpubl(n),n)⇐e:publisher(b,n),e:type(b,´book´)
#9:ep:pub(b,fpubl(n))⇐ e:publisher(b,n),e:type(b,´book´)
```

Figure 5(b): Mapping rules from the *eBay* local ontology to the *eBay* application ontology.

```
Product ≡ ap:Product ⊔ ep:Product
title ≡ ap:title ⊔ ep:title
Book ≡ ap:Book ⊔ ep:Book   ...
```

Figure 6: Some of the mediated mappings.

Figures 5(a) and 5(b) show the LO-AO rules induced from the vocabulary matching of Figures 3(a) and 3(b). In Figure 5(b), the function *fpubl* is used to add an object of class *ep:Publ* and the properties *ep:name* and *ep:pub* in the application ontology. Figure 6 presents some mediated mappings, which allow the definition of a class (property) of the domain ontology through a *unique axiom*, composed by unions of classes (properties) of the application ontologies. They can be used for *unfolding* a query submitted over the domain ontology directly over the application ontologies

Table 1: From Vocabulary Matching to AO, LO-AO, AO-DO and mediated mappings.

Q = set of quadruples *qi* (lo:v1, lo:e1, do:v2, do:e2) $\mathcal{C}$ = set of classes of AO and $\mathcal{P}$ = set of properties of AO	$\mathcal{M}'$ = set of LO-AO mapping rules *M_concept* = set of mediated mappings of this *concept*
Condition analyzed for each *qi*	**Actions**
Case 1: lo:v_1 and do:v_2 are classes	$\mathcal{C} := \mathcal{C} \cup$ {ao:v_2}; *M_v2* := *M_v2* + "⊔" + {ao:v_2}; $\mathcal{M}' := \mathcal{M}' \cup$ {ao:v_2(x) ⇐ lo:v_1(x)}; for each superclass *S* of do:v2 do $\mathcal{M}' := \mathcal{M}' \cup$ {ao:*S*(x) ⇐ lo:v_1(x)}; if (ao:S ∉ $\mathcal{C}$) then $\mathcal{C} := \mathcal{C} \cup$ {ao:S}; *M_S*:= *M_S* + "⊔" + {ao:S};
Case 2: lo:v_1 and do:v_2 are properties. Let lo:e_1 and do:e_2 be the *contexts* of lo:v_1 and do:v_2, respectively:	
Case 2.1: *Q* matches lo:e_1 with do:e_2 and do:v_2 belongs to the class do:e_2 or to a *superclass S* of the class do:e_2.	$\mathcal{P} := \mathcal{P} \cup$ {ao:v_2}; *M_v2*:= *M_v2* + "⊔" + {ao:v_2}; $\mathcal{M}' := \mathcal{M}' \cup$ {ao:v_2(x, y) ⇐ lo:v_1(x, y), lo:e_1(x)};
Case 2.2: *Q does not* match lo:e_1 with do:e_2 but there is a *property path* (lo:p_{k1}, lo:p_{k2}, …, lo:p_{km}) in the source ontology corresponding to the alignment between lo:v_1 and do:v_2.	$\mathcal{P} := \mathcal{P} \cup$ {ao:v_2}; *M_v2*:= *M_v2* + "⊔" + {ao:v_2}; $\mathcal{M}' := \mathcal{M}' \cup$ {ao:v_2(x, y) ⇐ lo:pk_1(x, x_1), lo:pk_2(x_1, x_2),…, lo:pk_m(x_m-1,z), lo:v_1(z,y)};
Case 2.3: *Q does not* match lo:e_1 with do:e_2 and there is *no property path* that can align properties lo:v_1 and do:v_2, but the user can identify an *equivalence* between them:	$\mathcal{C} := \mathcal{C} \cup$ {ao:e_2}; *M_e2*:= *M_e2* + "⊔" + {ao:e_2}; $\mathcal{P} := \mathcal{P} \cup$ {ao:v_2}; *M_v2*:= *M_v2* + "⊔" + {ao:v_2};
Case 2.3.1: The user proposes a *selection condition* identifying a *property* lo:p_k in the source ontology that allows the alignment between properties lo:v_1 and do:v_2 and contexts lo:e_1 and do:e_2.	$\mathcal{M}' := \mathcal{M}' \cup$ {ao:e_2(x) ⇐ lo:e_1(x), lo:p_k(x, 'select value')}; $\mathcal{M}' := \mathcal{M}' \cup$ {ao:v_2(x, y) ⇐ lo:v_1(x,y), lo:p_k(x, 'select value')}; for each superclass *S* of *do:e2* do $\mathcal{M}' := \mathcal{M}' \cup$ {ao:S(x) ⇐ lo:e_1(x), lo:p_k(x, 'select value')}; if (ao:S ∉ C) then $\mathcal{C} := \mathcal{C} \cup$ {*ao:S*}; *M_S*:= *M_S* + "⊔" + {*ao:S*};
Case 2.3.2: The user proposes a *restructuring of information* in the enrolled ontologies creating a function *f* that allows the alignment between properties lo:v_1 and do:v_2 (*y* is an *inverse functional property* passed as argument to *f*).	$\mathcal{M}' := \mathcal{M}' \cup$ {ao:e_2(*f(y)*) ⇐ lo:v_1(x,y)}; $\mathcal{M}' := \mathcal{M}' \cup$ {ao:v_2(*f(y)*, y) ⇐ lo:v_1(x,y)}; $\mathcal{P} := \mathcal{P} \cup$ {*ao:*p_2}; *M_p2*:= *M_p2* + "⊔" + {*ao:*p_2}; $\mathcal{M}' := \mathcal{M}' \cup$ {ao:p_2 (x, *f(y)*) ⇐ lo:v_1(x,y)};

ACKNOWLEDGEMENTS

This work was partly supported by CNPq, under grants 301497/2006-0, 473110/2008-3 and 557128/2009-9, FAPERJ E-26/170028/2008, and by CAPES under grant CAPES/PROCAD NF 21/2009.

REFERENCES

Calvanese, D., De Giacomo, G., Lenzerini, M., Lembo, D., Poggi, A., Rosati, R., 2007. MASTRO-I: Efficient Integration of Relational Data through DL Ontologies. In: Proc. DL Workshop'07, pp. 227 – 234.

Calvanese, D., Lenzerini, M., Nardi, D., 1998. Description Logics for Conceptual Data Modeling. In: *Logics for Databases and Information Systems*. Kluwer Academic Publisher.

Casanova, M.A., Lauschner, T., Leme, L.A.P., Breitman, K.K; Furtado, A.L., Vidal, V. M. P., 2009. A Strategy to Revise the Constraints of the Mediated Schema. In: Proc. 28th Conf. on Conceptual Modeling, pp. 265-279, Gramado, Brazil.

Euzenat, J., Shvaiko, P., 2007. *Ontology Matching*. Springer, Heidelberg.

Hull, R., Yoshikawa, M., 1990. ILOG: Declarative Creation and Manipulation of Object Identifiers. In: Proc. VLDB 1990, pp. 455-468.

Leme, L. A. P., Casanova, M. A., Breitman, K. K., Furtado, A. L., 2009. Instance-based OWL Schema Matching. In: Proc. 11th International Conf. on Enterprise Information Systems, Milan, Italy.

Lutz, M., 2006. Ontology-based Discovery and Composition of Geographic Information Services. Phd Thesis, Institut für Geoinformatik.