Using Analogy to Promote Conceptual Modeling Reuse

Karin K. Breitman, Simone D.J. Barbosa, Marco A. Casanova,
Antonio L. Furtado, Michael G. Hinchey
Departamento de Informática – Pontifícia Universidade Católica do Rio de Janeiro
Rua Marquês de S. Vicente, 225 – Rio de Janeiro, Brazil – CEP 22451-900
{karin, simone, casanova, furtado}@inf.puc-rio.br, mike.hinchey@usa.net
http://www.inf.puc-rio.br

Abstract. This paper argues in favor of a database conceptual schema and Semantic Web ontology design discipline that explores analogy mappings to reuse the structure and integrity constraints of conceptual models, stored in a repository. We presuppose that a team of expert conceptual designers would build a standard repository of source conceptual models, which less experienced designers would use to create new target conceptual models in other domains. The target models will then borrow the structure and the integrity constraints from the source models by analogy. The concepts are expressed in the contexts of Description Logics, the RDF model and OWL to reinforce the basic principles and explore additional questions, such as the consistency of the target model.

1 Introduction

Metaphor is not merely a rhetorical device, characteristic of language alone. Lakoff and Johnson (1980) argue that "the human conceptual system is fundamentally metaphorical in nature. The essence of metaphor is understanding and experiencing one kind of thing in terms of another." Holyoak and Thagard (1995, p. 220) argue that "metaphor uses the same mental processes as analogical thinking ... a metaphor is understood by finding an analogy mapping between the target domain (the topic of the metaphor) and the source domain. The degree to which an analogy is viewed as metaphorical will tend to increase the more remote the target and source domains are from each other."

In this paper, we claim that analogy mappings facilitate conceptual modeling by allowing the designer to reinterpret fragments of familiar conceptual models in other contexts. This may have applications in developing new versions of ground control systems or flight software for a new spacecraft, for example. Exploitation of such an approach requires a sound approach and a methodology to support conceptual modeling. To that end, we propose a discipline for database conceptual schema design, and Semantic Web ontologies as well, that we call conceptual modeling by analogy and metaphor.

The discipline is based on two simple ideas Breitman et al. (2007). First, a team of expert conceptual designers would build a standard repository of *source* conceptual models that cover commonly found conceptual design patterns and that are expressed in familiar terms. The source conceptual models will naturally contain fully formalized integrity constraints, as defined by the conceptual design experts. For example, instead of a generic weak entity pattern, the collection will contain a sample model of employees and their dependents, which is typically used to explain weak entities and is more intuitive to apprehend than any abstract formulation of the concept. Second, naïve designers would then create new *target* conceptual models in other domains by defining analogy mappings with the source conceptual models in the repository. The target models will then borrow the structure and the integrity constraints from the source models by analogy – essentially a combination of a straightforward renaming process with consistency checking.

The discipline for database conceptual schema and Semantic Web ontologies design would then consist of the gradual expansion of conceptual models for specific domains basically by repeatedly defining analogy mappings.

The contributions of this paper lie in two directions. We define a simple construct, ISLIKE, to specify analogy mappings. We adopt the weak entity construct as a running example since it is well-known, it has a concise description, and yet it looks rather sophisticated to most naïve designers. Next, we rephrase our proposal in the context of Description Logics (DL). We chose DL mostly because it is fairly well-known, it has been applied to provide precise definitions for ER and UML concepts, and it permits discussing consistency of the target conceptual model, an issue that must not be overlooked. Finally, we discuss how to specify analogy mappings in the context of the RDF model and OWL – Web Ontology Language. We introduce two new properties, isLike and Source, and show how they can be used to3 creat3target OWL ontologies from source ontologies by analogy.

Winston (1980) is an early reference that describes a theory of analogy with applications to AI systems. The Berkeley Master Metaphor List is an example of a relatively old (linguistics) metaphor repository Lakoff et al. (1991). Metaphors have been used to improve human-computer interface design Blackwell (2006), Barbosa and de Souza (2001), Catarci et al. (1996), in particular, and software design, in general Lippert et al. (2003). Goguen (1999) describes a formalization of user interface design based on semiotic morphisms. The analogy mappings we describe throughout the paper are, in some sense, morphisms that preserve the structure and the integrity constraints of the source schemas.

This paper is organized as follows. Section 2 formalizes the strategy in Description Logics. Section 3 specifies the conceptual modeling strategy we propose in the context of the RDF model and OWL. Finally, Section 4 contains the conclusion.

2 Analogy in DL

In this section, we introduce analogy mappings in the context of the Description Logic (DL). We first briefly review the very basic concepts of DL, referring the reader to Baader and Nutt (2003) for an introductory account of DL. Then, we formalize the concepts of analogy mappings and terminologies generated by analogy. For the sake of brevity, we do not introduce the concept of metaphor mapping. The reader familiar with the basic concepts of DL may proceed directly to Section 3.2.

2.1 A Brief Review of Attributive Languages

Description logic (DL) refers to a family of knowledge representation formalisms that model the application domain by defining the relevant concepts of the domain and then using these concepts to specify properties of objects and individuals occurring in the domain Baader and Nutt (2003). We summarize in this section the syntax and semantics of the *family of attributive languages*, or AL-*family*.

An *attributive language* L is characterized by an *alphabet* consisting of a set of *atomic concepts*, a set of *atomic roles*, and the special symbols T and ⊥, respectively called the *universal concept* and the *bottom concept*. (Note that the notion of alphabet here is entirely similar, but not identical, to that introduced in Section 2.1).

The set of *concept descriptions* of L is inductively defined as follows:

Any atomic concept and the universal and bottom concepts are concept descriptions.

If *A* is an atomic concept, *C* and *D* are concept descriptions, and *R* is an atomic role, then the following expressions are concept descriptions.

$\neg A$	(atomic negation)
$C \sqcap D$	(intersection)
$\forall R.C$	(value restriction)
$\exists R.\mathrm{T}$	(limited existential quantification)

The other classes of languages of the AL-family expand the set of concept descriptions to include expressions of one of the following forms.

If *C* and *D* are concept descriptions, *R* is an atomic role, and *n* is a positive integer, then the following expressions are concept descriptions.

$\neg C$	(arbitrary negation)
$C \sqcup D$	(union)
$\exists R.C$	(full existential quantification)
$(\geq n\ R)$	(at-least restriction, a cardinality restriction)
$(\leq n\ R)$	(at-most restriction, a cardinality restriction)
R^{-}	(inverse role)

An *interpretation* I for an attributive language L consists of a nonempty set Δ^{I}, the *domain* of I, whose elements are called *individuals*, and an *interpretation function* such that:

$\mathrm{T}^{\mathrm{I}} = \Delta^{\mathrm{I}}$ and $\perp^{\mathrm{I}} = \varnothing$

For every atomic concept *A* of L, the interpretation function assigns a set $A^{\mathrm{I}} \subseteq \Delta^{\mathrm{I}}$

For every atomic role *R* of L, the interpretation function assigns a binary relation $R^{\mathrm{I}} \subseteq \Delta^{\mathrm{I}} \times \Delta^{\mathrm{I}}$

Informally, the interpretation function is extended to concept descriptions of L inductively as follows:

$(\neg A)^{\mathrm{I}}$ denotes the complement of A^{I} with respect to the domain

$(C \sqcap D)^{\mathrm{I}}$ denotes the intersection of C^{I} and D^{I}

$(\forall R.C)^{\mathrm{I}}$ denotes the set of individuals that *R* relates only to individuals in C^{I}, if any

$(\exists R.\mathrm{T})^{\mathrm{I}}$ denotes the set of individuals that *R* relates to some individual of the domain

For the extended family, we have:

$(\neg C)^{\mathrm{I}}$ denotes the complement of C^{I} w.r.t. the domain

$(C \sqcup D)^{\mathrm{I}}$ denotes the union of C^{I} and D^{I}

$(\exists R.C)^{\mathrm{I}}$ denotes the set of individuals that *R* relates to some individual in C^{I}

$(\geq n\ R)^{\mathrm{I}}$ denotes the set of individuals that *R* relates to at least *n* individuals

$(\leq n\ R)^{\mathrm{I}}$ denotes the set of individuals that *R* relates to at most *n* individuals

$(R^{-})^{\mathrm{I}}$ denotes the inverse of *R*

Let L be a language in any of the classes of the AL-family. A *terminological axiom* (written) in L or, simply, an *axiom*, is an expression of the form $C \sqsubseteq D$, called an *inclusion*, or of the form $C \equiv D$, called an *equality*, where C and D are concept descriptions in L. A *definition* (written) in L is an equality $A \equiv D$ such that A is an atomic concept and D is a concept description of L.

Let I be an interpretation for L. Then, I *satisfies* $C \sqsubseteq D$ iff $C^I \subseteq D^I$, and I *satisfies* $C \equiv D$ iff $C^I = D^I$. Let T be a set of axioms. Then, I *satisfies* T, or I *is a model of* T, iff I satisfies each axiom in T. Two sets of axioms are *equivalent* iff they have the same models.

A *terminology* (written) in L is a set of axioms T such that, for any atomic concept A of L, there is at most one definition in T whose left-hand side is A, called *the definition of A* in T. We may therefore partition (with respect to T) the atomic concepts of L into *defined concepts* (with respect to T) that appear in the left-hand side of the definitions in T and *primitive concepts* (with respect to T) that do not appear in the left-hand side of the definitions in T.

Our interest in these concepts lies exactly in that there are fairly efficient decision procedures that test satisfiability of terminologies for various dialects of the AL-family.

2.2 Analogy Mappings between Terminologies

Given two attributive languages, S and T, with alphabets A(S) and A(T), an *analogy mapping* from A(S) into A(T) is one-to-one mapping α:A(S)$\rightarrow$A(T) such that, for any $s \in$A(S), if $\alpha(s)$ is defined, then s and $\alpha(s)$ both are atomic concepts or both are atomic roles. We say that A(S) is the *source alphabet* and A(T) is the *target alphabet* of α.

We extend α to the concept expressions and axioms of S in the obvious way: for each concept expression or axiom t of S, $u=\alpha(t)$ is the concept expression or axiom of T obtained by replacing each symbol s of A(S) that occurs in t by $\alpha(t)$, if $\alpha(t)$ is indeed defined; if there is a symbol occurring in t for which α is undefined, then $\alpha(t)$ is undefined. If $\alpha(t)$ is defined, we also say that $u=\alpha(t)$ is *induced* by α. (It is left for the reader to show that u is indeed a concept expression or axiom of T).

Given two attributive languages, S and T, a terminology S written in S, and an analogy mapping α:A(S)$\rightarrow$A(T), we say that a terminology T is *created by analogy with* S *using* T *and* α iff T is the set of all axioms t in T such that there is $s \in$ S such that $\mu(s)$ is defined and $t=\mu(s)$. Intuitively, each axiom s of S is translated into an axiom t of T, if the analogy mapping μ is indeed defined for all symbols that occur in s. We also say that S is the *source terminology* and T in the *target terminology* of μ. Note that, as for the ER model, the designer has a certain degree of freedom when using analogy mappings, since he may block reusing some of the original axioms simply by omitting the mapping of certain symbols of the source alphabet A(S).

We have to extend the above definitions, though, since the designer may want to: (i) repeatedly apply analogy mappings to create a new terminology out of others; (ii) revise the axioms induced by an analogy mapping to restore consistency. The first point is entirely analogous to the discussion in Section 2.3. However, we raised the second point only here since DL, but not the original ER model, comes equipped with a family of fairly efficient decision procedures that test satisfiability of terminologies for various dialects of the AL-family, as already anticipated.

Let S and T be two attributive languages, B be a terminology written in T, S be a terminology written in S, and α:A(S)→A(T) be an analogy mapping. We say that a terminology T *expands* B *by analogy with* S *using* T *and* α iff T = B ∪ U, where U is the terminology created by analogy with S using T and α. Note that T is not necessarily consistent since U adds new axioms to B that may potentially be inconsistent with the original axioms of B.

This brings us to the second point. We briefly discuss three alternatives here. First, we may treat T as a typical knowledge base and leave it to the designer the problem of restoring consistency by deprecating axioms, induced by an analogy mapping or not. The designer would have to resort to whatever DL reasoner he has in hand to help him in this task.

Second, we may revise the notion of extending a terminology using an analogy mapping to avoid inconsistencies from the onset. Adopting this second approach, we define that T *consistently expands* B *by analogy with* S *using* T *and* α iff T is a maximal consistent subset of B ∪ U, where U is the terminology created by analogy with S using T and α. This second approach is just theoretical, since T may not unique and, therefore, the designer may end up not knowing what would be the outcome of an expansion by analogy.

The third approach is more pragmatic, and is being adopted as part of our proof-of-concept implementation. We start by considering that the axioms of a source terminology are ranked. Then, we modify the expansion by analogy process to introduce the axioms from the source terminology, one by one, in rank order, testing for consistency at each step. If an inconsistency is found, the process stops with the last consistent expansion found (and asks for user intervention). This is again possible by combining a DL reasoner with the implementation of the repository of source conceptual models.

As an example of terminologies created by analogy, consider the terminology EMPLOYEE defined with an alphabet E consisting of:

Atomic concepts:	`Emp`	(the set of employees)
	`Dep`	(the set of dependents)
	`No`	(the set of positive integers)
Atomic roles:	`EmpNo`	(assigns an employee number to an employee)
	`DepNo`	(assigns a sequential number to a dependent)
	`isDepOf`	(a binary relation that indicates the dependent of an employee)

The axioms of terminology EMPLOYEE are:

$\texttt{Emp} \sqsubseteq (\geq 1\ \text{EmpNo}) \sqcap (\leq 1\ \text{EmpNo}) \sqcap \forall \text{EmpNo}.\text{No}$
(employees have exactly one employee number)

$\texttt{No} \sqsubseteq (\leq 1\ \text{EmpNo}^-) \sqcap \forall \text{EmpNo}^-.\text{Emp}$
(employee number is a key of employee)

$\texttt{Dep} \sqsubseteq (\geq 1\ \text{DepNo}) \sqcap (\leq 1\ \text{DepNo}) \sqcap \forall \text{DepNo}.\text{No}$
(dependents have exactly one sequential number)

$\texttt{Dep} \sqsubseteq (\geq 1\ \text{isDepOf}) \sqcap (\leq 1\ \text{isDepOf})$
$\sqcap \forall \text{isDepOf}.\text{Emp}$
(dependents are associated with exactly one employee)

We refer the reader to Calvanese and Giacomo (2003) for a discussion on how to translate UML concepts into DL and to Borgida and Brachman (2003) for a discussion of conceptual modeling in DL, in general.

However, we note that the Discriminating Attribute Property is not expressible in any language of the AL-family defined in Section 3.1 since, intuitively, it is equivalent to a composite key defined by concatenating the employee number and the dependent number Calvanese and Giacomo (2003). Using a first-order language derived from the alphabet E, this property would be written as follows:

$\forall x \forall y \forall w \forall z \forall n$ (Dep(x) $\wedge$ Dep(y)
isDepOf(x,z) $\wedge$ isDepOf(y,z)
DepNo(x,n) $\wedge$ DepNo(y,n) $\Rightarrow$ $x=y$)

Consider now a second alphabet, B, defined as follows:

Atomic concepts:	`Book`	(the set of books)
	`Edition`	(the set of book editions)
	`No`	(the set of positive integers)
Atomic roles:	`ISBN`	(assigns an ISBN to a book)
	`EdNo`	(assigns a number to an edition)
	isEdOf	(a binary relation that associates an edition to a book)

Then, we may define a analogy mapping μ:E→B such that:

μ(`Emp`) = `Book`	μ(`EmpNo`) = `ISBN`
μ(`Dep`) = `Edition`	μ(`DepNo`) = `EdNo`
μ(`No`) = `No`	μ(isDepOf) = isEdOf

The extension of EMPLOYEE induced by analogy mapping μ would be the terminology with the following axioms:

`Book` $\sqsubseteq$ ($\geq$ 1 ISBN) $\sqcap$ ($\leq$ 1 ISBN) $\sqcap$ $\forall$ISBN.No
(books have exactly one ISBN)

`ISBN` $\sqsubseteq$ ($\leq$ 1 ISBN$^-$) $\sqcap$ $\forall$ISBN$^-$.Book
(ISBN is a key of book)

`Edition` $\sqsubseteq$ ($\geq$ 1 EdNo) $\sqcap$ ($\leq$ 1 EdNo) $\sqcap$ $\forall$EdNo.No
(editions have exactly one edition number)

`Edition` $\sqsubseteq$ ($\geq$ 1 isEdOf) $\sqcap$ ($\leq$ 1 isEdOf)
$\sqcap$ $\forall$isEdOf.Book
(editions are associated with exactly one book)

$\forall x \forall y \forall w \forall z \forall n$ (`Edition`(x) $\wedge$ `Edition`(y)
isEdOf(x,z) $\wedge$ isEdOf(y,z)
EdNo(x,n) $\wedge$ EdNo(y,n) $\Rightarrow$ $x=y$)

The point of this example is again that the designer may reuse a fairly sophisticated set of axioms by just defining an analogy mapping and relying on his intuitive understanding of the source terminology to generate the target terminology. Indeed, if he has a sufficiently rich library of terminologies, he may specify a new terminology largely by defining analogy mappings.

3 Analogy in OWL

In this section, we discuss how to specify analogy mappings in the context of the RDF model and OWL – Web Ontology Language. We first review the RDF model Manola and Miller (2004), OWL McGuinness and Harmelen 92004) and the TRIPLE language Decker et al. 92005), which we adopt to complement OWL due to its similarities with first-order languages. Then, we introduce two new properties, isLike and Source, and show how they can be used to create OWL ontologies with the help of analogy mappings.

3.1 A Brief Review of the RDF Model, OWL and TRIPLE

We assume that the reader is familiar with the basic XML concepts. In particular, recall that a *resource* is anything identified by an URIref and that an XML *namespace* or a *vocabulary* is a set of URIrefs. A *literal* is a character string that represents an XML Schema datatype value.

An *RDF statement* (or simply a *statement*) is a triple *(S, P, O)*, where

S is a URIref, called the *subject* of the statement

P is a URIref, called the *property* (or *predicate*)

of the statement, that denotes a binary relationship is either a URIref or a literal, called the *object* of the statement; if *O* is a literal, then *O* is also called the *value* of the property *P.*

The *Web Ontology Language* (OWL) describes classes, properties, and relations among these conceptual objects in a way that facilitates machine interpretability of Web content. OWL is defined as a vocabulary, just as are RDF and RDF Schema, but it has a richer semantics. Hence, an ontology in OWL is a collection of RDF triples, which uses such vocabulary.

For example, the class and property declarations of the `Employee` ontology would be written in OWL as follows:

```
  <?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
     <!ENTITY xsd
           "http://www.w3.org/2001/XMLSchema#">
     <!ENTITY owl
           "http://www.w3.org/2002/07/owl#">]>
<rdf:RDF
  xml:base="www.metaphor.org/Employee/"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdf=
 "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs=
 "http://www.w3.org/2000/01/rdf-schema#">
  <owl:Ontology rdf:about="">
     <rdfs:label>Employee</rdfs:label>
</owl:Ontology>
  <owl:Class rdf:about="Emp"/>
  <owl:Class rdf:about="Dep"/>
  <owl:DatatypeProperty rdf:about="EmpNo">
     <rdf:type rdf:resource=
                 "&owl;FunctionalProperty"/>
     <rdf:type rdf:resource=
            "&owl;InverseFunctionalProperty"/>
     <rdfs:domain rdf:resource="Emp"/>
     <rdfs:range rdf:resource=
                         "&xsd;positiveInteger"/>
20.   </owl:DatatypeProperty>
21.   <owl:DatatypeProperty rdf:about="DepNo">
22.      <rdf:type rdf:resource=
                         "&owl;FunctionalProperty"/>
23.      <rdfs:domain rdf:resource="Dep"/>
24.      <rdfs:range rdf:resource=
                         "&xsd;positiveInteger"/>
25.   </owl:DatatypeProperty>
26.   <owl:ObjectProperty rdf:about="isDepOf">
27.      <rdf:type rdf:resource=
                         "&owl;FunctionalProperty"/>
28.      <rdfs:domain rdf:resource="Dep"/>
29.      <rdfs:range rdf:resource="Emp"/>
30.   </owl:ObjectProperty>
31. </rdf:RDF>
```

In particular, the value of the rdf:about attribute of owl:Ontology indicates the URIref of the ontology. If the value is empty, as in line 10, the URIref of the ontology is the base URI of the document. Line 11 provides a perhaps more readable name for the ontology.

The above XML document defines a set of triples, labeled `Employee`. However, not all set of triples T using the OWL vocabulary defines an OWL ontology. For example, T may have a triple (*C*, `rdfs:domain`, *D*) indicating that *D* is the domain of *C* and yet T does not contain any triple indicating that *C* is a datatype or object property. Without providing a detailed definition for the sake of brevity, we say that a set of triples `T` using the OWL vocabulary is *well-formed* iff `T` defines an OWL ontology.

TRIPLE supports namespaces, sets of RDF statements, reification, and rules with syntax close to that of first-order logic. We refer the reader to Sintek and Decker (2002) and Decker et al. (2005) for a full description of TRIPLE.

An *RDF statement* is an expression of the form "S[P→O].", where `S` is the subject, `P` is the predicate, and `O` is the object, and denotes an ordinary RDF triple `(S,P,O)`. An *RDF model* is a set of RDF statements. A model can be made explicit in TRIPLE, receive a name (i.e., a resource denoting a model), and be attached to an atom, RDF statement, or molecule to indicate that the statement holds in that model.

An *atomic formula* is an atom, an RDF statement, or a molecule. A *formula* is either an atomic formula or an expression recursively defined by composing atomic formulas and formulas with the usual logical connectives ($\neg$, $\wedge$, $\vee$) and quantifiers ($\forall$, $\exists$). All variables must be introduced via the universal or existential quantifier.

A *clause* is either a fact or a rule. A *fact* is an atomic formula. A *rule* is an expression of the form $\forall X\ C \leftarrow A$, where *X* is a list of variables, *C* is a conjunction of atomic formulas, and *A* is a formula.

The `Employee` ontology also includes an additional TRIPLE rule that captures the discriminating property for dependents (if two dependents, X and Y, are dependents of the same employee Z and they have the same sequential number N, then they are actually the same dependent, i.e., X=Y):

```
FORALL X,Y,Z
X=Y <- X[seqNo→N]   AND Y[seqNo→N] AND
     X[isDepOf→Z] AND Y[isDepOf→Z].
```

3.2 **Analogy Mappings in OWL**

We first discuss how to define analogy mappings in the context of RDF triples, which parallels the development of Section 3.2, and then show how to use analogy mappings to generate OWL documents.

We capture analogy mappings with the help of a vocabulary with just two properties, md:isLike and md:Source, where md: abbreviates the URIref http://www.metaphor.org/term/, adopted for the sake of our discussion.

Given two vocabularies, S and T , an *analogy mapping* from S into T is a set L of RDF triples of the form *(t,* md:isLike*, s)*, where *s* and *t* are URIrefs of the vocabularies S and T, respectively, such that, for any two triples *(t',* md:isLike*, s')* and *(t",* md:isLike*, s")* in L, we have that *t'=t"* iff *u'= u"*. Note that L defines a one-to-one mapping α:S$\rightarrow$T such that $\alpha(s)=t$ iff *(t,* md:isLike*, s)* occurs in L. We say that S is the *source vocabulary* and T is the *target vocabulary* of L.

In what follows, assume that an OWL ontology O is expressed as a set of RDF triples, and that A(O) denotes the vocabulary of O. Given an OWL ontology S, a vocabulary T and an analogy mapping L from A(S) into T, we say that an OWL ontology T is *created by analogy with* S *using* T *and* L iff

(triples analogy) $(t_1,t_2,t_3)\in$T iff there is $(s_1,s_2,s_3)\in$S such that, for $i\in\{1,2,3\}$, either s_i is a term of the OWL vocabulary and $t_i = s_i$ or t_i is a term of T and there is a triple of the form $(t_i,$ md:isLike$, s_i)\in$L. (Intuitively, each RDF triple s in S generates a triple t in T iff t is the translation of s to the vocabulary T using the translations defined in L).

(structural consistency) T is a well-formed OWL ontology.

Note that condition (1) defines how to translate triples in S using the analogy mapping defined by L, but it leaves out those triples for which L does not define a complete translation. This may create inconsistencies in T. Thus, condition (2) requires that T indeed be a well-formed OWL ontology.

We also say that an OWL ontology T is *created by metaphorical analogy with* S *using* T *and* L iff T is created by analogy with S using *using* U *and* M, where

U =T $\cup$ {$s\in$A(S) / ($\forall t\in$T)(*(t,* md:isLike*, s)*$\notin$L)}

M = L $\cup$ {*(s,* md:isLike*, s)* /
($\forall t\in$T)(*(t,* md:isLike*, s)*$\notin$L)}

Since M contains all the necessary triples, by construction, T will be a well-formed OWL ontology, if S is. Therefore, metaphorical analogies are simpler to construct.

We now show how to accommodate these concepts in RDF/XML with the help of the following example:

```
   <?xml version="1.0"?>
 <!DOCTYPE rdf:RDF [
   <!ENTITY xsd
      "http://www.w3.org/2001/XMLSchema#">
<!ENTITY we
      "http://www.metaphor.org/weakEntity/">]>
 <rdf:RDF
   xml:base="http://www.books.com/owl-schema/"
   xmlns:owl="http://www.w3.org/2002/07/owl#"
   xmlns:rdf=
  "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
   xmlns:rdfs=
   "http://www.w3.org/2000/01/rdf-schema#"
```

```
10.   xmlns:mp="http://www.metaphor.org/term/">
11.   <owl:Ontology rdf:about="">
12.     <rdfs:label>Books Catalogue</rdfs:label>
13.   </owl:Ontology>
14.   <md:Source
    rdf:resource="www.metaphor.org/weakEntity/"/>
15.     <rdf:Description rdf:about="Book">
16.       <md:isLike rdf:resource="&we;Emp/"/>
17.     </rdf:Description>
18.     <rdf:Description rdf:about="Edition">
19.       <md:isLike rdf:resource="&we;Dep/"/>
20.     </rdf:Description>
21.     <rdf:Description rdf:about="isEdOf">
22.       <md:isLike rdf:resource="&we;isDepOf/"/>
23.     </rdf:Description>
24.     <rdf:Description rdf:about="ISBN">
25.       <md:isLike rdf:resource="&we;EmpNo/"/>
26.     </rdf:Description>
27.     <rdf:Description rdf:about="EdNo">
28.       <md:isLike rdf:resource="&we;DepNo/"/>
29.     </rdf:Description>
30.    </md:Source
31. </rdf:RDF>
```

Line 12 contains the name of the target ontology, `Books Catalogue`, in this case. Line 14 indicates that the OWL ontology defined at the URI `www.metaphor.org/weakEntity/` is a source ontology for `Books Catalogue`. Lines 15 to 17 indicate that `Book` is like `Emp` of the `Employee` ontology, and similarly for lines 18 to 29. Note that the `md:isLike` declarations are nested inside the md:Source declaration. This syntactical structure permits more than one source ontology to be specified for the same target ontology, as exemplified at the end of Section 2.3. The `Books Catalogue` ontology would then expand to:

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
  <!ENTITY xsd
         "http://www.w3.org/2001/XMLSchema#">
  <!ENTITY owl
          "http://www.w3.org/2002/07/owl#">]>
<rdf:RDF
  xml:base="http://www.books.com/owl-schema/"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:rdf=
"http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs=
     "http://www.w3.org/2000/01/rdf-schema#">
  <owl:Ontology rdf:about="">
     <rdfs:label>Books Catalogue</rdfs:label>
  </owl:Ontology>
  <owl:Class rdf:about="Book"/>
  <owl:Class rdf:about="Edition"/>
  <owl:DatatypeProperty rdf:about="ISBN">
     <rdf:type rdf:resource=
                 "&owl;FunctionalProperty"/>
     <rdf:type rdf:resource=
           "&owl;InverseFunctionalProperty"/>
     <rdfs:domain rdf:resource="Book"/>
     <rdfs:range rdf:resource=
                    "&xsd;positiveInteger"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="EdNo">
     <rdf:type rdf:resource=
                    "&owl;FunctionalProperty"/>
     <rdfs:domain rdf:resource="Edition"/>
     <rdfs:range rdf:resource=
                        "&xsd;positiveInteger"/>
  </owl:DatatypeProperty>
  <owl:ObjectProperty rdf:about="isEdOf">
     <rdf:type rdf:resource=
                    "&owl;FunctionalProperty"/>
     <rdfs:domain rdf:resource="BookVolume"/>
     <rdfs:range rdf:resource="Book"/>
  </owl:ObjectProperty>
</rdf:RDF>
```

Finally, we state, omitting the details, that the rewriting scheme introduced above generalizes to TRIPLE in the obvious way, so that the definition of a source ontology may also include rules in TRIPLE.

For example, the Books Catalogue ontology also includes an additional TRIPLE rule that captures that two book editions must have different edition numbers (or, equivalently, if two book editions are editions of the same book and they have the same edition number, then they are actually the same book edition):

```
  FORALL X,Y,Z
 X=Y <- X[EdNo→N]  AND Y[EdNo→N] AND
     X[isEdOf→Z] AND Y[isEdOf→Z].
```

4.Conclusion

Holyoak and Thagard (1995) introduce analogies as mappings between target domain and source domain. They argue that “the mapping can be used to enrich understanding of the target by generating new inferences, and it can lead to formation of a schema based on the relational structure common to the target and the source. In addition, analogy goes hand in hand with the formation of schemas – new categories that embrace both the source and the target, thus changing our understanding of both.” Along these lines, but with a different purpose, we introduce analogy mappings as a way to generate new schemas from previously defined ones, considered to be design archetypes.

We argued in favor of a database conceptual schema and Semantic Web ontology design discipline that explores analogy mappings to reuse the structure and integrity constraints of conceptual models, stored in a repository. The discipline presupposes that a team of expert conceptual designers would build a standard repository of *source* conceptual models, which less experienced designers would use to create new *target* conceptual models in other domains. The target models will then borrow the structure and the integrity constraints from the source models by analogy. The concepts were expressed in the contexts of Description Logics, the RDF model and OWL to reinforce the basic principles and explore additional questions, such as the consistency of the target model.

Acknowledgement

This work is partially supported by CNPq under grants 550250/2005-0, 551241/2005-5 and 311794/2006-8.

References

Baader, F.; Nutt, W. (2003) Basic description logics. In: Baader, F.; Calvanese, D.; McGuiness, D.L.; Nardi, D.; Patel-Schneider, P.F. (Eds) *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, Cambridge, UK.

Barbosa, S.D.J.; Souza, C.S. (2001) "Extending software though metaphors and metonymies". *Knowledge-Based Systems*, 14, Elsevier Science, pp. 15-27.

Blackwell, A.F. (2006) "The reification of metaphor as a design tool". *ACM Trans. on Computer-Human Interaction*, Vol. 13, Issue 4 (Dec. 2006), pp. 490-530.

Borgida, A.; Brachman, R.J. (2003) Conceptual Modeling with Description Logics. In: Baader, F.; Calvanese, D.; McGuiness, D.L.; Nardi, D.; Patel-Schneider, P.F. (Eds) *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, Cambridge, UK.

Breitman, K. K., Barbosa, S.D.J., Casanova, M.A., Furtado, A.L., Conceptual Modeling by Analogy and Metaphor". In Proc. of the ACM Sixteenth Conference on Information and Knowledge Management (CIKM 2007), November 2007, Lisbon (Portugal), to appear.

Calvanese, D.; De Giacomo, G. (2003) "Description Logics for Conceptual Data Modeling in UML". In: Proc. 15th European Summer School in Logic Language and Information, August 2003, Vienna (Austria), pp. 18-29.

Catarci, T.; Costabile, M.F.; Matera, M. (1996) "Which Metaphor for Which Database?". In: Proc. HCI'95 Conf. on People and Computers. Huddersfield, UK, pp. 151-165.

Chen, P. P-S. (1976) "The entity-relationship model—toward a unified view of data". *ACM Transactions on Database Systems*, Vol. 1, Issue 1 (March 1976), p. 9–36.

Decker, S. et al. (2005) TRIPLE - An RDF rule language with context and use cases. In: Proc. W3C Workshop on Rule Languages for Interoperability, USA, pp. 27–28.

Goguen, J. (1999) "An introduction to algebraic semiotics, with applications to user interface design". In: Nehaniv, C (ed.) *Computation for Metaphors, Analogy and Agents*. LNAI, Vol. 1562. Springer, pp. 242-291.

Holyoak, K.J.; Thagard, P. (1995) *Mental Leaps – Analogy in Creative Thought*. The MIT Press, Cambridge, MA, USA.

Lakoff, G.; Johnson, M. (1980) *Metaphors We Live By*. The University of Chicago Press, Chicago, 1980.

Lakoff, G.; Espenson, J.; Schwartz, A. (1991) *Master metaphor list*. Second draft copy. University of California Berkeley. Available at: http://cogsci.berkeley.edu/lakoff/

Lippert, M.; Schmolitzky, A.; Züllighoven, H. (2003) "Metaphor Design Spaces". In: Proc. 4th Int. Conf. Extreme Programming and Agile Processes in Software Engineering, XP 2003. Genova, Italy, May 25-29, 2003.

Manola, F.; Miller, E. (Eds) (2004) RDF Primer. W3C Recommendation, 10 February 2004. Available at: http://www.w3.org/TR/rdf-primer/.

McGuinness, D.L.; Harmelen, F.V. (Eds) (2004) OWL Web Ontology Language Overview. W3C Recommendation, 10 February 2004. Latest version available at: http://www.w3.org/TR/owl-features/

Winston, P.H. (1980) "Learning and reasoning by analogy". *Communications of the ACM*, 23, pp. 689-703.