

Semantic Interoperability via Category Theory

Isabel Cafezeiro¹

Edward Hermann Haeusler²

¹ Departamento de Ciência da Computação, Universidade Federal Fluminense, Niterói, Rua Passo da Pátria, 156 - Bloco E - 3 andar Boa Viagem - RJ CEP: 24.210-240, Email: isabel@dcc.ic.uff.br

² Departamento de Informática, Pontifícia Universidade Católica, Rua Marques de São Vicente 225, Gávea, Rio de Janeiro, RJ Email: hermann@inf.puc-rio.br

Abstract

This paper aims to bring the benefits of the use of Category Theory to the field of Semantic Web, where the coexistence of intrinsically different models of local knowledge makes difficult the exchanging of information. The paper uses categorical limit and colimit to define operations of breaking and composing ontologies, formalizing usual concepts in ontologies (alignment, merge, integration, matching) and proposing a new operation (the hide operation). The presented set of operations form a useful framework that makes easier the manipulation and reuse of ontologies.

Keywords: Ontology, Interoperability, Category Theory.

1 Introduction

Ontologies are used in computer science to describe real world things by hierarchically organizing concepts and enriching this hierarchy with relationships among concepts. An ontology is basically composed by *concepts (classes)* and *relations (properties)*. Concepts are structured in a taxonomy and relations set non-taxonomically connections. In addition, restrictions are stated by the use of logical axioms given in some expressive language whose model-theoretic semantics provides meaning. With this framework it is possible, not only to obtain accurate descriptions of real things, but also to enable intelligent interoperability among descriptions.

Researches Maedche (2001), Klein, M. (1987) have shown that technology based on ontology can be successfully applied to make viable the Semantic Web. But, the large applicability of this technology depends heavily on the mechanisms of mapping and integrating heterogeneous entities. Recent researches, like Kalfoglou, Y. & Schorlemmer, M. (2004), also remark the use of ontologies in knowledge engineering and point out difficulties in achieving effective interoperability. We believe that a formal view on ontologies can contribute a lot in this direction. But it is essential that the chosen formalism (i) emphasizes relationships between things, allowing mapping in an appropriate way; (ii) allows the coexistence of heterogeneous entities; and (iii) offers a good set of operations to put entities together. When concerning integration, interoperability of heterogeneous entities,

Copyright ©2007, Australian Computer Society, Inc. This paper appeared at the Twenty-Sixth International Conference on Conceptual Modeling - ER 2007 - Tutorials, Posters, Panels and Industrial Contributions, Auckland, New Zealand. Conferences in Research and Practice in Information Technology, Vol. 83. John Grundy, Sven Hartmann, Alberto H. F. Laender, Leszek Maciaszek and John F. Roddick, Eds. Reproduction for academic, not-for profit purposes permitted provided this text is included.

Category Theory is an appropriate formal framework. First of all, because of the focus that is put in relationship (categorical morphisms) and not in entities (categorical object). In agreement with (i), entities are described abstractly, accordingly to their interactions with other entities. Secondly, it is possible to define several categories, according to the kinds of entities to be described. These categories can be related by the definition of relationships between categories (categorical functors) that preserve properties of categories. Thus, as stressed by (ii), it is possible for heterogeneous entities to coexist. Finally, addressing (iii), Category Theory offers a set of ways of combining entities, some of which (as colimits) are traditionally used to integrate entities. Category Theory has been used successfully in situations where interoperability is a crucial point, as in formal specification of systems Ehrig & Mahr (1986) and in software architecture Haeusler & Meseguer (2000). We believe that Category Theory can contribute with the Semantic Web in the same way as it contributed in the areas of systems specification, systems architecture and many other fields: not only formalizing the object of study, but also giving good directions to operationalize the concept and facilitating its implementation. A direct gain of this formalization is the modularity and reuse of the framework: operations are defined succinctly and in a modular way, which makes it possible for new applications to reuse existing ontologies.

This paper is structured as follows: Section 2 presents a definition of category and the basic operations used in this paper. Section 3 presents the terminology to be adopted in this paper. Section 4 formalizes Ontology Category, concepts of Pushouts, Pullbacks and Equalizers, and shows how they merge, detect similarities and hide components of ontologies. Section 5 defines contextualized web queries, and shows how this idea fits in the categorical framework. Section 6 presents related works and 7 concludes the paper.

2 Categorical Theory, Limits and Colimits

Definition 1 A category \mathcal{C} is a structure $(O, M, Dom, Cod, \circ, id)$, where O is a collection of objects; M is a collection of morphisms $f : A \rightarrow B$, where $A, B \in O$; $Dom, Cod : M \rightarrow O$ are operations which associates to each morphism, its domain object and codomain object; \circ is an associative operation of morphism composition; id is a collection of identity morphisms id_A for each object A .

In the above definition the notion of morphism is the central idea. It is asserted that morphisms have domain, codomain and a composition operation. In contrast, almost nothing is stated about objects. In general, the entity to be described is an object, but it is focused by an external view, given by means of morphisms. Morphisms and objects are disposed in such

a way that (parts of) a category can be pictured as graphs (diagrams), what facilitates to reason about a category. For example, some definitions are obtained just by reversing “arrows” in the “graph”. In the following, we define limit and colimit by using diagrams.

Considering the pictorial arrangement of objects and morphisms, we call a *diagram* any set of objects and morphisms. We call a *cone* (Figure 1, left) a diagram with morphisms $f_i : o \rightarrow o_i$ such that for any morphism $g : o_i \rightarrow o_j$, $g \circ f_i = f_j$. We use the notation $\{f_i : o \rightarrow o_i\}$ for such a cone.

Definition 2 A *limit* for a diagram D with objects o_i is a cone $\{f_i : o \rightarrow o_i\}$ such that for any other cone $\{f'_i : o' \rightarrow o_i\}$, for D , there is a unique morphism $! : o' \rightarrow o$ for which $f_i \circ ! = f'_i$, with $f'_i : o' \rightarrow o_i$.

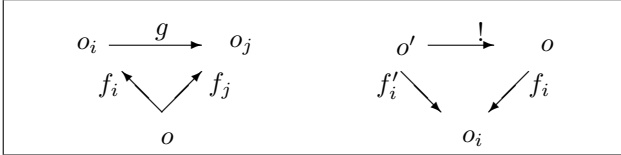


Figure 1: A cone and a Limit for a diagram with object o_i and morphisms $g : o_i \rightarrow o_j$.

The requirement “for any other cone $\{f'_i : o' \rightarrow o_i\}$, for D , there is a unique morphism $! : o' \rightarrow o$ for which $! \circ f_i = f'_i$, with $f'_i : o' \rightarrow o_i$ ” ensures that any other object that could also play the role of vertex of the cone, can be mapped in the limit. In other words, the limit is the more informative object in the category that can be mapped by the morphisms f_i to any object of the diagram.

There are special cases of diagrams whose limit gives very useful operations. For instance, the limit of an empty diagram results in the terminal object of the category. The limit of a diagram composed of two single objects, characterizes the product of these objects. The limit of a diagram as $o_1 \rightarrow o \leftarrow o_2$ is named pullback, and characterizes a kind of product of o_1 and o_2 , guided by o . The limit of a diagram as $o \rightrightarrows o_1$ is named equalizer, and expresses the similarities between both morphisms from o to o_1 .

We call *duality* the process of reversing all the morphisms in a category or diagram. It is an important mechanism in Category Theory because it usually results in interesting definitions. The concept of cocone is the result of dualizing cone diagram, the concept of colimit is the result of dualizing limit diagram.

Definition 3 A *colimit* for a diagram D with objects o_i is a cocone $\{f_i : o_i \rightarrow o\}$ such that for any other cocone $\{f'_i : o_i \rightarrow o'\}$, for D , there is a unique morphism $! : o \rightarrow o'$ for which $! \circ f_i = f'_i$, with $f'_i : o_i \rightarrow o'$.

The requirement “for any other cocone $\{f'_i : o_i \rightarrow o'\}$, for D , there is a unique morphism $! : o \rightarrow o'$ for which $! \circ f_i = f'_i$, with $f'_i : o_i \rightarrow o'$ ” ensures that the colimit can be mapped in any other object that could also play the role of vertex of the cocone. In other words, colimit is the less informative object that embodies all information from the diagram mapped by f_i .

The colimit of an empty diagram results in the initial object of the category, that is, the object from which there exists a unique morphism to any other object of the category. The colimit of a diagram composed of two single objects, characterizes the coproduct of these objects. The colimit of a diagram as $o_1 \leftarrow o \rightarrow o_2$ is named pushout. In this case the object colimit embodies the sum of o_1 and o_2 possibly collapsing parts according to the morphisms coming from o . We point the interested reader to MacLane (1997) for a complete reference on Category Theory.

3 Ontology Terminology

Considering that there is not a standardized terminology in the area, we present in this section the meaning of ontology structure and the operations to be formalized in this paper. We are considering Maedche (2001), which presents ontologies in a layered approach composed of an *ontology structure*, a *lexicon for the ontology structure*, a *knowledge base structure* and a *lexicon for the knowledge base structure*. As sections 4, 5 focus on the structural composition of ontologies, the axiom component of ontologies that appears in the definition of Ontology Structure will be omitted. In Cafezeiro & Haeusler (2007), the logical system is considered and it is shown that limit and colimit are well defined also in the syntactical and semantic categories of the logical system. In Cafezeiro & Haeusler (2007) the reader can find comparisons between the terminology adopted in this section and other approaches (for instance, Bench-Capon & Malcolm. (1999), Bruijn, J. et al. (2004), Kalfoglou, Y. & Schorlemmer, M. (2004)).

Definition 4 (Ontology Structure) An *ontology structure* is a tuple (C, R, H^C, rel, A) . The components, in the same order that appear in the tuple, are: *Concepts*, *Relations*, which are disjoint sets. $H^C \subseteq C \times C$ is a hierarchy of concepts - a taxonomic relation. $rel : R \rightarrow C \times C$ is a function that relates concepts non-taxonomically. Axioms specify other properties of concepts and relations. We assume that H^C is a partial order. By $(x_1, x_0) \in H^C$ we mean that x_1 is a subconcept of x_0 .

Ontology mapping is a total mapping from ontologies o_1 to o_2 which preserves hierarchy and conceptual relations and specify the semantic overlap between o_1 and o_2 .

Ontology alignment “is the task of establishing a collection of binary relations between the vocabularies of two ontologies. Since a binary relation can itself be decomposed into a pair of total functions from a common intermediate source, we may describe the alignment of two ontologies o_1 and o_2 by means of a pair of ontology mappings from an intermediate source ontology o .” Kalfoglou, Y. & Schorlemmer, M. (2004)

Ontology merging is the unification of two (or more) ontologies producing a new ontology that embodies the semantic differences and collapses the semantic intersection between the original ones.

Ontology matching is the task of finding commonalities between ontologies.

Hiding components of ontologies is the task of erasing a concept/relation of an ontology, preserving hierarchy, conceptual relations and semantic relations of the remaining components that do not depend on the erased component.

4 Ontologies in a Categorical View

Definition 5 (The category \mathcal{Ont} of ontologies) The category \mathcal{Ont} of ontologies has as objects ontology structures and as morphisms pairs of functions $(f, g) : O \rightarrow O'$ where $O = (C, R, H^C, rel)$ and $O' = (C', R', H^{C'}, rel')$ are ontology structures and $f : C \rightarrow C'$ and $g : R \rightarrow R'$ are such that (i) if $(C_1, C_2) \in H^C$ then $(f(C_1), f(C_2)) \in H^{C'}$, and (ii) if $(C_1, C_2) \in rel(r)$ then $(f(C_1), f(C_2)) \in rel'(g(r))$, for $r \in R$.

By condition (i), morphisms preserve hierarchy of concepts. By (ii), morphisms preserve relations.

Example 1 (Morphism for Mapping Ontologies) Consider the ontologies O_1 and O_2 defined in Figure

C' is a collapsed element from C_1 and C_2 . The notation $f_1(c) \mapsto c'$ in the definition of f'_1 means that the image of $f_1(c)$ under f'_1 is c' . In steps (2) and (3), the remaining concepts of C_1 and C_2 are also added to C' . At last, the hierarchy of H^{C_1} and H^{C_2} is copied to $H^{C'}$, considering the collapsed components (steps (4) and (5)). This construction ensures that, with respect to concepts, the diagram commutes. As morphisms preserve hierarchy, the composed $f'_i \circ f_i$ ensures that the $H^{C'}$ embodies H^{C_i} , considering collapsed components.

Pushout Algorithm

- Input: $(f_1, g_1) : O \rightarrow O_1, (f_2, g_2) : O \rightarrow O_2$
Output: $(f'_1, g'_1) : O_1 \rightarrow O', (f'_2, g'_2) : O_2 \rightarrow O'$
Initial Conditions: $O' = O_\epsilon, f'_i = \perp, g'_i = \perp$
- (1) For all $c \in C'$
 $C' := C' \cup c'$, where $c' \notin (C_1 \cup C_2)$
 $f'_1 := f'_1 \cup f_1(c) \mapsto c'$
 $f'_2 := f'_2 \cup f_2(c) \mapsto c'$
 - (2) For all $c \in C_1$ that is not in the image of f_1
 $C' := C' \cup c$
 $f'_1 := f'_1 \cup f_1(c) \mapsto c$
 - (3) For all $c \in C_2$ that is not in the image of f_2
 $C' := C' \cup c$
 $f'_2 := f'_2 \cup f_2(c) \mapsto c$
 - (4) For all $(c, d) \in H^{C_1}$
 $H^{C'} := H^{C'} \cup (f'_1(c), f'_1(d))$
 - (5) For all $(c, d) \in H^{C_2}$
 $H^{C'} := H^{C'} \cup (f'_2(c), f'_2(d))$

Example 3 (Merging Ontologies) The pushout of the alignment of Figure 3 results in an ontology that collapses the components identified by f_1, f_2 and embodies the components of both ontologies that are not in the image of f_1 or f_2 . The resulting ontology is shown in Figure 5.

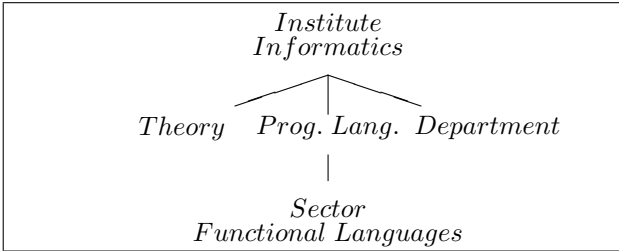


Figure 5: Pushout ontology of alignment of Figure 3.

4.2 Integration via Limits

The matching operation is important not only by its main purpose of detecting similarities, but also because in many frameworks it is part of the process of merging ontologies Bruijn, J. et al. (2004). In our approach the match results in a new ontology containing the similarities of the given ontologies and is formalized by limits.

Limit is the dual concept of colimit, that is, reversing the arrows of a colimit diagram yields a limit diagram. Reversing the arrows of a pushout diagram yields a pullback diagram. Thus, in a dual way of subsection 4.1, we use the operation *pullback* which is a particular case of limit concerning only three objects O, O_1 and O_2 linked by two morphisms: $O_1 \rightarrow O \leftarrow O_2$. We consider that O_1 and O_2 are the ontologies to be matched, and O is an intermediate ontology that guides the matching. The mappings O_1 to O and O_2 to O specify how concepts and relations of O_1 and O_2 are semantically correspondent to

concepts and relations of the ontology O . The resulting ontology embodies only the semantically identified concepts of O_1 and O_2 .

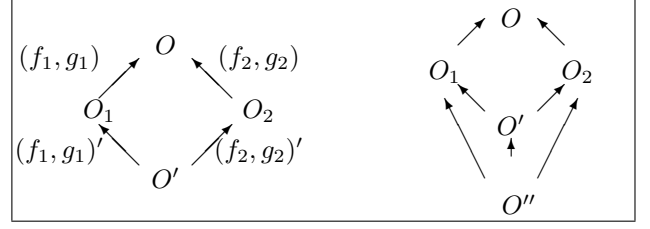


Figure 6: The Pullback commutativity and unicity.

Definition 7 (Pullback of ontologies) Consider the ontologies O_1, O_2 and O , and the ontology morphisms $(f_1, g_1) : O_1 \rightarrow O$ and $(f_2, g_2) : O_2 \rightarrow O$. The pullback is the ontology O' with morphisms $(f_1, g_1)' : O' \rightarrow O_1$ and $(f_2, g_2)' : O' \rightarrow O_2$ such that (i). $(f_1, g_1) \circ (f_1, g_1)' = (f_2, g_2) \circ (f_2, g_2)'$ and for any other ontology O'' , with morphisms $(f_1, g_1)'' : O'' \rightarrow O_1$ and $(f_2, g_2)'' : O'' \rightarrow O_2$ there exists a unique morphism $(f, g) : O'' \rightarrow O'$ with (ii). $(f_1, g_1)' \circ (f, g) = (f_1, g_1)''$ and $(f_2, g_2)' \circ (f, g) = (f_2, g_2)''$.

Condition (i) states that O' embody all information of O_1 and O_2 that is semantically equivalent. Condition (ii) states that O' is the greater ontology that can be constructed in this way: if there is any other ontology O'' that also has all semantic intersection, then it can be mapped in O' by the morphism from O'' to O' , as shows the right diagram of Figure 6.

The pullback algorithm searches for concepts c_i in C_i that have the same image in C by f_i . For each of these concepts, a new concept is added in C' , with mappings f'_i linking this concept c_i . The compositions $f_i \circ f'_i$ ensures the semantic intersection with respect to C . As C' is a subset of C , part of the hierarchy of H^C is copied to $H^{C'}$. The components R' and rel' and g'_i are calculated in a similar way.

Pullback Algorithm

- Input: $(f_1, g_1) : O_1 \rightarrow O, (f_2, g_2) : O_2 \rightarrow O$
Output: $(f'_1, g'_1) : O' \rightarrow O_1, (f'_2, g'_2) : O' \rightarrow O_2$
Initial Conditions: $O' = O_\epsilon, f'_i = \perp, g'_i = \perp$
For all $c_1 \in C_1$

If there is $c_2 \in C_2$ such that $f_1(c_1) = f_2(c_2)$

$$C' := C' \cup f_1(c_1)$$

$$f'_1 := f'_1 \cup f_1(c_1) \mapsto c_1$$

$$f'_2 := f'_2 \cup f_2(c_2) \mapsto c_2$$

Example 4 (Discovering Similarities) Let O_1 and O_2 be the ontologies in Figure 2. We define O , a new ontology to be the codomain of morphisms $(f_1, g_1), (f_2, g_2)$ from O_1 and O_2 (Figure 7 A). These morphisms map concepts of O_1 and O_2 in their similar in O . Pulling back this diagram will result in the ontology O' , with morphisms $(f'_1, g'_1), (f'_2, g'_2)$ (Figure 7 B).

The ontology O' is the intersection of O_1 and O_2 with respect to O . Both O_1 and O_2 were collapsed by f_1 and f_2 to the right branch of O , thus, with respect to O , x_0 and y_0 are similar concepts which we named z_0 . The concepts x_3 and y_2 are similar concepts and appear in O' as z_3 . The concepts x_1, x_2 and y_1 are considered to be similar with respect to O , but as $x_1 \neq x_2$, we have in O' two different concepts named z_1 and z_2 . As state f'_1 and f'_2 , z_1 expresses the similarity of x_1 and y_1 ($f'_1(z_1) = x_1$ and $f'_2(z_1) = y_1$), and z_2 expresses the similarity of x_2 and y_1 ($f'_1(z_2) = x_2$ and $f'_2(z_2) = y_1$). Integrity of O_1 and O_2 is ensured as, by definition of morphism, mapping preserves conceptual hierarchy and relationships.

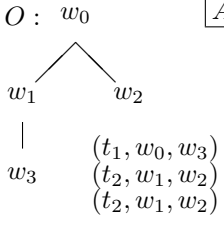
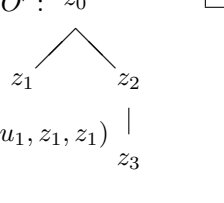
$f_1 : C_1 \rightarrow C$ $x_0 \mapsto w_0$ $x_1 \mapsto w_1$ $x_2 \mapsto w_1$ $x_3 \mapsto w_3$ $g_1 : R_1 \rightarrow R$ $r_1 \mapsto t_1$ $r_2 \mapsto t_2$	$f_2 : C_2 \rightarrow C$ $y_0 \mapsto w_0$ $y_1 \mapsto w_1$ $y_2 \mapsto w_3$ $g_2 : R_2 \rightarrow R$ $s_1 \mapsto t_1$ $s_2 \mapsto t_3$	$O : w_0$ 	A
$f'_1 : C' \rightarrow C_1$ $z_0 \mapsto x_0$ $z_1 \mapsto x_1$ $z_2 \mapsto x_2$ $z_3 \mapsto x_3$ $g'_1 : R' \rightarrow R_1$ $u_1 \mapsto r_2$	$f'_2 : C' \rightarrow C_2$ $z_0 \mapsto y_0$ $z_1 \mapsto y_1$ $z_2 \mapsto y_1$ $z_3 \mapsto y_2$ $g'_2 : R' \rightarrow R_2$ $u_1 \mapsto s_2$	$O' : z_0$ 	B

Figure 7: A: Morphisms $(f_1, g_1), (f_2, g_2)$ and ontology O . B: Morphisms $(f'_1, g'_1), (f'_2, g'_2)$ and pullback ontology O' .

4.3 Limits for Hiding Information

An equalizer is a special case of limit that is performed in a diagram as $O_1 \overrightarrow{=} O$. Each mapping in this diagram gives a view of O_1 in terms of O . As any kind of limit, the equalizer detects the similarities between these views. The resulting ontology O' with morphism $O' \rightarrow O_1$ has all concepts and relations of O_1 that are sent to the same concept or relation in O by both morphisms (by both views of O_1 into O).

Definition 8 (Equalizer of ontologies) Consider the ontologies O_1 and O , and the ontology morphisms $(f_1, g_1), (f_2, g_2) : O_1 \rightarrow O$. The equalizer is the ontology O' with morphism $(f, g)' : O' \rightarrow O_1$ such that (i). $(f_1, g_1) \circ (f, g)' = (f_2, g_2) \circ (f, g)'$ and for any other ontology O'' , with morphism $(f, g)'' : O'' \rightarrow O_1$ there exists a unique morphism $(f, g) : O'' \rightarrow O'$ with (ii). $(f, g)' \circ (f, g) = (f, g)''$.

Condition (i), is expressing that O' has the components of O_1 that are sent to the same component in O by both (f_1, g_1) and (f_2, g_2) . Condition (ii), is expressing that O' is the more complete ontology that satisfies (i), that is, if there is any other ontology O'' that could play the same role as O' then it can be mapped in O' .

Equalizers may be applied in a situation where information of ontology - a password, for instance - must be hidden. It must be ensured that any reference to the erased symbol is also eliminated. For example, if there are relations concerning a hidden concept they must also be erased. With equalizers one can hide a symbol by performing a unique operation, what ensures that there will be not a forgotten reference to the hidden symbol. The process of hiding symbols of an ontology O_1 consists in considering a similar ontology O , with the difference that O must have an additional concept and an additional relation to be the image of concept/relations to be erased. Morphism from O_1 to O are the identity, and an embedding that maps any concept/relation of O_1 to the same concept/relation in O , except those that will be erased, that will be mapped to the additional concept/relation in O . By definition, the equalizer O' will contain all components for which both morphisms agree, that is, everything of O_1 except the concept/relation mapped to the additional concept/relation.

Example 5 (Hiding components) Let O_1 (in Figure 2, B) be the original ontology, and O ,

with morphisms $(f_i, g_i), (f_e, g_e)$ defined in Figure 8. The equalizer is a new ontology O' that has just the part of O_1 that is sent to the same components in O by both (f_i, g_i) and (f_e, g_e) . By definition, the equalizer is the ontology $O' = (\{x_0, x_1, x_3\}, \{r_1\}, \{(x_1, x_0), (x_3, x_1)\}, \{(r_1, x_0, x_1)\})$ with the embedding $(f, g)' : O' \rightarrow O_1$.

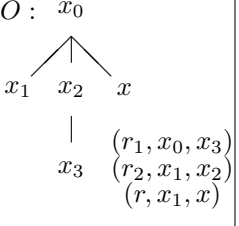
$f_i : C' \rightarrow C_1$ $x_0 \mapsto x_0$ $x_1 \mapsto x_1$ $x_2 \mapsto x_2$ $x_3 \mapsto x_3$ $g_i : R' \rightarrow R_1$ $r_1 \mapsto r_1$ $r_2 \mapsto r_2$	$f_e : C' \rightarrow C_2$ $x_0 \mapsto x_0$ $x_1 \mapsto x_1$ $x_2 \mapsto x$ $x_3 \mapsto x_3$ $g_e : R' \rightarrow R_2$ $r_1 \mapsto r_1$ $r_2 \mapsto r$	$O : x_0$ 
--	--	--

Figure 8: Morphisms $(f_1, g_1), (f_2, g_2)$ and ontology O for the equalizer.

5 Contextualizing Web Queries

Ontologies can be used to attach meaning to web pages. Using this semantic information, more accurate web consults can be implemented. The approach we describe here is to structure not only the web pages, but also the query themselves. We extend a given search algorithm to perform a pushout in a diagram composed by a contextualized query. In this way the search will result in ontologies that are semantically connected to the query. Morphisms from the query to results of the search give the semantic relationships.

We informally define *query* as a textual description used to request for information in the Semantic Web. A query is always associated to a domain of knowledge over which we consider a *context*, defined by Jannink, J. et al (1998), as a unit of encapsulation for well-structured ontologies from which we are able to assert correctness and consistency properties. Finally, we name *web consult* the searching for information, on the Semantic Web, that is semantically related to the query.

Describing queries by means of ontologies we are considering the possibility of having more sophisticated queries than unstructured textual information. Suppose also that a query is always linked to a context: an ontology (or a set of ontologies) that describes its domain of knowledge. Now, we can formally define *query* by establishing a connection between the ontology used to request for information and the (set of) ontology(ies) that compose the context. In this way, a *query* can be formally defined as a morphism in the category of ontologies.

Definition 9 (Query) A query is a morphism in *Ont*, where the domain ontology is the information to be searched and the codomain ontology is the context of the query.

Let us name O the domain of the query, O_1 the context, $search()$ some algorithm to perform a search in the Semantic Web, $\iota : O \hookrightarrow O_2$ an injection morphism from O to O_2 , $Morphisms(Ont)$ the set of morphisms of *Ont*, and $pushout()$, the algorithm to perform the pushout in a given diagram.

$Results := \emptyset$

For all $O_2 \in search(O)$

if $\iota : O \hookrightarrow O_2 \in Morphisms(Ont)$

$Results := Results \cup (pushout(O_1 \leftarrow O \hookrightarrow O_2))$

Return $Results$

A web consult would be implemented in the following way: $search(O)$ is a given algorithm, that performs the search for an ontology O resulting in a set

of related ontologies. Let us name O_2 an ontology of this set. If it is possible to define an injective mapping from O to O_2 , the diagram $O_1 \leftarrow O \hookrightarrow O_2$ can be constructed and given as input to a pushout algorithm. The set of diagrams resulting of these pushouts gives the semantic search result.

As usual, $search(O)$ may result in a set of possibilities, each of which is considered as target of the injection. Being a morphism, the injection ensures the coherence of O_2 with respect to O , that is, hierarchy and relations of O would be preserved in the resulting ontology. Being an injection, it ensures that concepts of O are not collapsed in O_2 . Pushing out the diagram $O_1 \leftarrow O \hookrightarrow O_2$ would result in consistent integration of concepts in O_1 and O_2 that is, an ontology (O') expressing the resulting of the consult, and morphism ($O_1 \rightarrow O'$) that gives semantic connections between the result and the original ontology. Many possibilities can result of this process, considering that the search results in a set of ontologies. Each possibility represents a semantic interpretation of concepts of the query.

By exploring other kinds of limits and colimits it is possible to construct a complete algebra to manipulate queries without losing meaning: the morphisms ensure the coherence with respect to the knowledge domain expressed by the context.

6 Related Work

Several researchers have been investigating how to adapt to the Semantic Web the benefits of the use of Category Theory (Schorlemmer05 (2005), Bench-Capon & Malcolm. (1999)). This is justified by the large contribution obtained by the use of Category Theory in areas where interoperability is a crucial question (see Ehrig & Mahr (1986) for the use of Category Theory in algebraic specifications, Haeusler & Meseguer (2000) in systems architecture). But in most of the researches, limit is not used. This fact can be justified on reasons that vary from difficulties of defining this concept in some categories to the small applicability of the concept in many fields. Zimmermann (2006) presents a categorical approach to formalize alignment, and uses limit to formalize intersection of alignments. But defines merge, composition, union and intersection as operations on alignments, and not on ontologies, as we present in this paper.

7 Conclusion

Focusing the mapping between local ontology, we formalize the concept of ontology mapping as a coherent mapping between ontologies. By *coherent*, we mean that mapping always preserves hierarchy and relations, and thus, preserves the integrity of the domain ontology. In this sense, any way of combining ontologies defined over the concept of ontology mapping is also coherent, and operations can be defined without the need of constraints to verify consistence. We use the concept of categorical morphism - a mapping that preserves structure of objects - to formalize ontology mapping. As categorical operations are defined over the concept of morphism, the categorical definitions of limit and colimit can be used to formalize ontology integration. Limit formalizes decomposition of ontologies and colimit formalizes composition. In this way, we have a complete useful set of operations to manipulate ontologies in a modular way, improving reuse of ontologies. The formalization presented here adopts only the basic concepts of Category Theory, which facilitates the understanding for the general public. The operations cited above are defined at the structural level of ontologies. In Cafezeiro & Haeusler

(2007), Lucanu2004 (2004) Description Logics is suggested to formalize the logical counterpart.

This paper contributes to the literature of Ontology Systems presenting a simple categorical framework consisting of a set of operations to compose and decompose ontologies. This paper also sketches an algebra for operating contextualized web queries, focusing the achievement of semantically satisfactory results in web search.

References

- Bench-Capon, T. & Malcolm, G. (1999), Formalizing Ontologies and their Relations, *in* K. V. Andersen, J. K. Debenham, R. Wagner, eds., 'Proceedings of the 16th DEXA'.
- Bruijn, J., Martin-Recuerda, F., Manov, D. & Ehrig, M. D4.2.1: State of the ArtSurvey on Ontology Merging and Aligning. (2007), Available: <http://www.inf.unibz.it/~jdebruijn/>
- Cafezeiro, I. & Haeusler, E. H. (2007), Description Logics as Institutions, Universidade Federal Fluminense, Niteroi, Brasil.
- Cafezeiro, I. & Haeusler, E. H. (2007), Categorical Formalization of Ontologies: the terminology, Universidade Federal Fluminense, Niteroi, Brasil.
- Ehrig, H. & Mahr, B. (2007) *Fundamentals of Algebraic Specifications. Equations and Initial Semantics*, SpringerVerlag.
- Haeusler, E. H. & Meseguer, J. (2000), Protem/NSF. PUCRio, S.R.I International.
- Jannink, J., Pichai, S., Verheijen, D. & Wiederhold, G. (1998), Encapsulation and Composition of Ontologies, *in* 'Proceedings of the AAAI98'.
- Kalfoglou, Y. & Schorlemmer, M. (2004), 'Ontology Mapping: the state of the art', *The Knowledge Engineering Review*, **18**(1) 1-31.
- Klein, M., Ding, Y., Fensel, D. & Omelayenko, B. (2003), Ontology Management: Storing, Aligning and Maintaining Ontologies, *in* J. Davies, D. Fensel, F. Van Harmelen, eds., 'Towards the Semantic Web'.
- Lucanu, D., (2004), A logical foundation for the OWL languages, *in* 'Symposium on Leveraging Applications of Formal Method (IsoLA)', Cyprus.
- MacLane, S. (1997), *Categories for the Working Mathematician*, Springer Verlag.
- Maedche, A. & Staab, S. (2001), 'Ontology learning for the Semantic Web', *IEEE Intelligent Systems* **16**, no. 2 (2001) 72-7.
- Schorlemmer, M. & Kalfoglou, Y. (2005), Progressive Ontology Alignment for Meaning Coordination: An Information-Theoretic Foundation, *in* 'Proceedings of the 4th AAMAS'05', Utrecht, Holland.
- Zimmermann, A., Krotzsch, M., Euzenat, J. & Hitzler, P. (2006), Formalizing Ontology Alignment and its Operations with Category Theory, *in* FOIS'06', Baltimore.