# What Is Ontology Merging?

**– A Category-Theoretical Perspective Using Pushouts**

**Pascal Hitzler** and **Markus Krötzsch** and **Marc Ehrig** and **York Sure**

Institute AIFB, University of Karlsruhe, Germany;
{hitzler,mak,ehrig,sure}@aifb.uni-karlsruhe.de

### Abstract

Ontology merging describes the process of integrating two (or more) ontologies into a single one. How this is done best is a subject of ongoing research in the Semantic Web community. We propose a generic solution to the question, what the result of a merging should be in the ideal case. We will do this independent of a specific choice of ontology representation language, and thus provide a sort of *blueprint* for the development of algorithms applicable in practice. Our methods are taken from category theory. More precisely, we will argue that ontology merging is best captured by the notion of categorical *pushout*. Our paper is a first step towards the development of practically applicable algorithms. We will not assume any background in the abstract field of category theory and will explain in detail and from scratch what our perspective on ontology merging means and entails.

## Introduction

In this paper we explain how merging of ontologies is captured by the pushout construction from category theory, and argue that this is a very natural approach to the problem. For this purpose, we view category theory as a universal "meta specification language" that enables us to specify properties of ontological relationships and constructions in a way that does not depend on any particular implementation. This can be achieved since the basic objects of study in category theory are the *relationships* between multiple ontological specifications, not the internal structure of a single knowledge representation.

The problems that one can study with this categorical approach include those that relate to the *communication* between different communities with different specification formalisms, where one wants to obtain information from other ontologies to include it in own specifications. Indeed, users of such distributed ontologies would not be interested in implementation details of every available ontology, but in the *interfaces* that exist to own knowledge bases and in the *translation methods* that are available to accomplish exchange of data.

Intuitively, the relationships or translation methods available between ontologies would also entail constraints for combining the information of multiple specifications. For example, in order to merge the content of two ontologies into a bigger one, it must be possible to transfer all the data of the input ontologies to the resulting description. It is certainly not obvious how relationships between ontologies alone can suffice to define the result of such a merging operation. However, we shall see that this is indeed the case and explain the corresponding pushout construction.

Categorical pushouts are already considered in some approaches to ontology research (Jannink *et al.* 1998; Kent 2000; Schorlemmer, Potter, & Robertson 2002; Goguen 2005; Kent 2005) and we do not claim our treatment to be entirely original. Still we have the impression that the potential of category theoretic approaches is by far not exhausted in todays ontology research. Consequently, our goal is not only to demonstrate how a concrete problem can be captured with categorical formalisms, but also to give introduction and motivation for those who did not study the mathematical framework of category theory yet. In this respect our attempts to make categories more accessible follow the spirit of (Goguen 1991) and the current discussions on the *Information Flow Framework* (Kent 2000).

In contrast to the some of the works mentioned above, we do not try to give a comprehensive overview of even the most important categorical methods. Instead, our treatment will focus on the particular aspect of ontology merging, for which we will give both intuitive explanations and precise definitions. This reflects our belief that, at the current stage of research, it is not desirable to fade out the mathematical details of the categorical approach completely, since the interfaces to current techniques in ontology research are not yet available to their full extent. We will also keep this treatment rather general, not narrowing the discussion to specific formalisms – this added generality is one of the strengths of category theory.

We proceed as follows: In the next section, we provide an introduction to categories, together with some basic examples that we will consider throughout this text. Then we investigate how category theory deals with cartesian products and relations, which we will utilize to model "ontology mapping" and "ontology alignment" in categorical terms thus establishing the framework for the first part of any merging operation. Section  then forms the core of this note, explaining pushouts and their relevance to ontology merging. In Section  we will explain how our theoretical considerations can be used to obtain practical methods for ontology merging. The last section includes references to the literature,

pointing to sources of further information on categorical and ontological issues touched on herein.

## Categorical preliminaries

In order to approach the concept of a category, we view it as a system of ontological specifications that includes both ontologies and their interrelations. Informally, an ontology can be viewed as something which conveys a certain specification (e.g. of some data) based on a given classification system. Mathematically, this description allows for a number of realizations: tree structures, formal contexts, partially ordered sets, or deductive systems of some logic are only examples. These approaches vary widely in their expressive power and may appear rather diverse indeed.

On the other hand, any suitable notion of an ontology should feature certain properties. This derives from the fact that ontologies are conceived as a means of sharing and reusing knowledge. Hence a typical task is to compare several (specifications of) ontologies or to combine them into a more extensive one. The latter process, termed *ontology merging*, will be discussed below. But for the sake of motivation, we shall first look at the former problem:

Given two ontologies, it is for example possible that one is a sub-ontology of the other or that both of them represent the same information, i.e. that they are equivalent. Depending on the chosen representation of ontologies this can be recognized by looking at the internal structures of the given ontologies. But looking at the internal structures requires an individual treatment for each new approach to the mathematical modelling of ontologies, while a generic treatment which abstracts from the particular choice of ontology language would certainly be preferable for understanding what the result of a merging shall be.

Fortunately, there is another possibility to compare objects mathematically, which lends itself to such a generic treatment. For example, two partially ordered sets can be considered to be *equivalent*, if there exists a bijective function (i.e. one which is one-to-one and onto) between these sets which does also preserve the order (i.e. which is *monotonic*). In this case, being monotonic means that a function respects the internal structure of partially ordered sets, while bijectivity indicates the equivalence of two ordered sets. Structure-preserving functions are a typical implementation of what is called a *morphism* in category theory, and what we will recognize as a suitable substitute for the consideration of internal structures.

While monotonic functions are reasonable morphisms for comparing partial orders, other mathematical spaces may suggest different kinds of morphisms: Vector spaces are considered with linear functions, groups with group homomorphisms, geometries with movements (e.g. on the plane), topological spaces with continuous functions, etc. Considering plain sets (with no further internal structure), a morphism between two sets could be any function between them. This approach ignores most individual features of the elements of a set: functions do not distinguish whether the elements of some set are labeled *a*, *b*, *c*, or *dog*, *cat*, *house*. Labels are only needed to specify the function, but the essential feature of a set turns out to be its cardinality. Sets of the same size would therefore appear equivalent.

The idea that emerges from these observations is that the relationships between objects are basically captured by the morphisms that exist between them. By deciding for a particular type of morphisms, we determine which internal properties of the mathematical objects are considered "essential" (e.g. order structure or cardinality). This is the approach taken in category theory: a class of *objects* (e.g. order structures) is equipped with *morphisms* (e.g. monotonic functions), thus forming a large directed graph with objects as nodes and morphisms as arrows. Depending on the given situation, arrows can be identified with certain functions or relations between the entities that were chosen for objects, but no such concrete meaning is required. In order to constitute a category, a directed graph only has to include a *composition* operation for pairs of compatible arrows, satisfying some straightforward axioms that are typical for the composition of functions and the relational product. Let us now make this informal description precise.

**Definition 1** A category $\mathscr{C}$ consists of the following:

- A class[1] of *objects* $|\mathscr{C}|$,
- for any two objects $A, B \in |\mathscr{C}|$, a set $\mathscr{C}(A, B)$ of *morphisms* from $A$ to $B$,
- for any three objects $A, B, C \in |\mathscr{C}|$, a *composition function* $\circ : \mathscr{C}(B, C) \times \mathscr{C}(A, B) \to \mathscr{C}(A, C)$, that combines a morphism from $A$ to $B$ with one from $B$ to $C$ to obtain a morphism from $A$ to $C$,
- for any object $A \in |\mathscr{C}|$, an *identity morphism* $\mathrm{id}_A \in \mathscr{C}(A, A)$.

This data is required to satisfy the following additional axioms:

- For all $f \in \mathscr{C}(A, B)$, $f \circ \mathrm{id}_A = f = \mathrm{id}_B \circ f$, and
- for all $f \in \mathscr{C}(A, B)$, $g \in \mathscr{C}(B, C)$, and $h \in \mathscr{C}(C, D)$, $((h \circ g) \circ f) = (h \circ (g \circ f))$.

We will also write $f : A \to B$ for $f \in \mathscr{C}(A, B)$.

The additional requirements for a directed graph to become a category are few indeed, and in many cases the definition of morphisms already entails an obvious and well-behaved composition operation. Yet the results one can derive from the components of a category are surprisingly rich, and usually all the essential knowledge about a class of objects is captured by some suitable category. The defining axioms of a category provide an abstract interface to all kinds of structures, and category theory allows for a unified treatment of all of them, since internal features of objects are disregarded completely.

As a simple example, consider the category Set of all sets and functions between them, i.e. |Set| consists of all sets,

[1] *Class* should be understood as a kind of *collection*. Classes of objects are not always *sets* of objects for reasons which have to do with Russell's paradox from set theory, but we shall not inconvenience us with such matters here. The term *class* is certainly not supposed to mean classes as e.g. in Description Logics!

and given two sets $A, B \in |\mathsf{Set}|$ the collection $\mathsf{Set}(A, B)$ of all morphisms from $A$ to $B$ is just the collection of all functions from $A$ to $B$. The identity morphisms are given as the identity functions, i.e. those functions which map all elements onto themselves. Composition in Set is given by composition of functions. Another category which we will discuss in more detail later is the category Poset of all partially ordered sets together with monotone functions. We remark that categorical morphisms are often given by functions, but that this is by no means necessary. For example, we can view a single partially ordered set as a category, where we have a single morphism between two elements $p$ and $q$ if and only if $p \leq q$. Composition is provided by transitivity and identity morphisms exist by reflexivity. This last example might appear somewhat peculiar at first, but it shows how general the basic notions in category theory really are.

Above, we gave two examples of specific relationships between objects: being a subobject and being equivalent. In the given examples, these do still refer to the internal structure of the objects, so we have to consider defining both in purely categorical terms. Let us first explore the notion of equivalence (or, speaking categorically, *isomorphism*) for the categories Set and Poset. Restating our earlier insights, we find that two partially ordered sets $P$ and $Q$ are equivalent (isomorphic) whenever there is a monotone function $f : P \to Q$ that has a monotone *inverse*, i.e. for which there is a monotone function $g : Q \to P$ with $g \circ f = \mathrm{id}_P$ and $f \circ g = \mathrm{id}_Q$. Generalizing this to arbitrary categories, we call a morphism an *isomorphism* if it has a (necessarily unique) inverse morphism.

Application of this definition to Set reveals bijective functions as the isomorphisms of sets. Likewise, the categorical definition immediately provides us with a suitable notion of equivalence in any category we may wish to study. As mentioned in the introduction, the possible translations of information between ontologies are suggestive morphisms for ontology research. Indeed, no matter how ontologies and translations between them are implemented, composition of translations methods and the existence of identity translations should always be available. In consequence, isomorphic ontologies are intuitively described by the possibility of translating knowledge back and forth between them without loosing information. In a similar fashion, we will gain a general description of *ontology merging* later on, though it will be a bit more involved.

To obtain the notion of a subobject, let us consider the category Set and note that every subset $A \subseteq B$ of a given set $B$ can be obtained as the image (range) of some injective (i.e. one-to-one) function into $B$. By injectivity, the domain of any such function is in bijective correspondence with the subset $A$. Thus any subset can be given by some injective function and any such function defines a subset[2]. We remark that the subobjects are really given by the injective function, not by its domain: for instance, a function from the one-element set $\{a\}$ to the natural numbers can map $a$ to 0, to 1, or to any other number. In spite of the constant function domain, this always denotes different subsets ($\{0\}$, $\{1\}$, ...) of the natural numbers. Yet, due to injectivity, the set $\{a\}$ is surely isomorphic (thus essentially equivalent) to the indicated subsets; still the *position* of the isomorphic copy of $\{a\}$ as a subobjects of the codomain does make a difference.

Now in order to obtain a categorical definition of injectivity, we observe that an injective function $f : A \to B$ in Set has the following peculiar property: for every pair of functions $g : C \to A$ and $h : C \to A$, the equality $f \circ g = f \circ h$ implies $g = h$. Morphisms (of arbitrary categories) with this feature are called *monomorphisms* and it turns out to be appropriate to consider a monomorphism as the specification of a subobject of its codomain. Intuitively, the condition describes a monomorphism $f : A \to B$ as an embedding of $A$ into $B$ that does not obliterate any essential features of $A$. Two morphisms $C \to A$ that are distinct on some part of $A$ must also be distinct when extended via $f$ to $B$. Thus the monomorphism can be thought of as a pointer to the specified subobject, which in turn is isomorphic to the domain of the monomorphism.

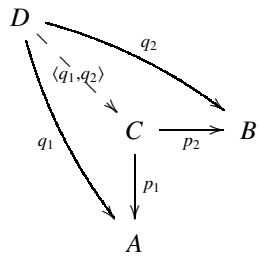
These examples give but a brief glimpse at the expressiveness of category theory. We continue next with discussing some constructions which will help in understanding pushouts.

## Products and Relations

In set theory, the cartesian product of two sets is defined as the set of all pairs of elements from two given sets. This is not a suitable description from the viewpoint of category theory, since we want to avoid to mention the internal (element-based) structure of our objects. In order to rephrase this in categorical language, we need to find alternative criteria that rely exclusively on properties of the morphisms. To this end, an important observation is that a product does in general also provide two *projection functions* to the first respectively second component of the product. Furthermore, the product is distinguished by a *universal property* given in the next definition.

**Definition 2** Consider a category $\mathscr{C}$ and objects $A, B \in |\mathscr{C}|$. Given an object $C \in |\mathscr{C}|$ and morphisms $p_1 : C \to A$ and $p_2 : C \to B$, we say that $(C, p_1, p_2)$ is the *product* of $A$ and $B$ if the following universal property holds:

For any object $D \in |\mathscr{C}|$ and morphisms $q_1 : C \to A$ and $q_2 : C \to B$, there is a unique morphism $\langle q_1, q_2 \rangle : D \to C$, such that $q_1 = p_1 \circ \langle q_1, q_2 \rangle$ and $q_2 = p_2 \circ \langle q_1, q_2 \rangle$. The latter situation is depicted in the following diagram:




[2]Note, however, that there are usually many injective functions with the same image. So for obtaining an exact definition of subobjects, one would still have to identify the equivalent injective functions.

For example, when considering the category Set and its usual cartesian product, we can define the function $\langle q_1, q_2 \rangle$ by setting $\langle q_1, q_2 \rangle(d) = (q_1(d), q_2(d))$. In spite of this, the above defines the cartesian product of sets only up to isomorphism (i.e. bijective correspondence) – *any* set with the cardinality of the cartesian product can be equipped with appropriate morphisms. This is a typical feature of category theory: isomorphic objects are not distinguished, since they behave similar in all practical situations. It is the choice of morphisms that determines what distinctions are considered relevant in the first place. Yet we will henceforth assume that we have fixed one representative for the product of any two elements $A$ and $B$ which we label $A \times B$. We also remark that products do not exist in every category, so the previous convention needs to be restricted to existing products.

We remark, nevertheless, that the notion of *product* of two objects depends solely on the chosen category, i.e. on the objects and their morphisms. Fixing, for example, a specific ontology language, and finding an agreement on which features of an ontology should be preserved by a corresponding morphism, we obtain a notion of *product* in a canonical way.

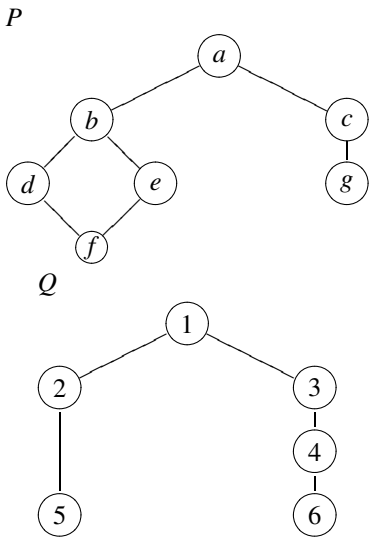
The categorical product definition also turns out to be suitable to model many well-known product constructions. As a product of two partially ordered sets one usually considers the product-order, i.e. the cartesian product of the two sets, ordered such that a pair $(a, b)$ is below a pair $(c, d)$ whenever $a$ is below $c$ and $b$ is below $d$. The partially ordered set obtained in this way corresponds to the categorical product in Poset, which arguably is the reason for the significance of this particular construction. To give another example: If we consider a single partially ordered set as a category, as discussed earlier, then the product of two of its elements is just the greatest lower bound. This is also an example where a product may fail to exist.

Combining the product construction with our earlier considerations on subobjects into practice, we can also introduce *binary relations* on objects. Indeed an ordinary set-theoretic binary relation is just a subset of the cartesian product of two objects. Hence it makes sense to consider a monomorphism $r : D \to (A \times B)$ from some object $D$ to the product of $A$ and $B$ as a binary relation between $A$ and $B$. Note that this does also give us two functions $p_1 \circ r : D \to A$ and $p_2 \circ r : D \to B$ to the two components of the product, for which the morphism $r$ is already the unique factorization that exists due to the definition of a product. Much generalized theory can be developed around this, but we shall be content at this point.

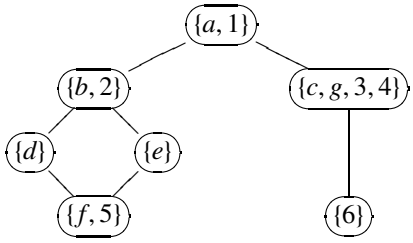
## Merging ontologies via pushouts

We will now return to our initial motivation. Our intuition is that the objects of our category represent ontologies and that the morphisms between them serve as meaningful transitions between these specifications. The categorical product construction is not suitable for the purpose of modelling ontology merging, since it does obviously not consider any relationship between two ontologies. Such a relationship – commonly referred to as an *ontology mapping* – however is the base of an ontology merging process, so we have to find a means of modelling it in our categorical setting. We are in fact more interested in a certain kind of *sum* than in a product. Indeed, if two ontologies were entirely unrelated, they could be combined by just taking their disjoint union (provided that this operation makes sense for the chosen ontology representation language). However, we are more interested in merging ontologies that do overlap (via some mapping), where some elements are related while others are not. Merging two such ontologies should lead to a new ontology that identifies equivalent elements but that tries to keep unrelated elements apart, as far as this is possible without violating the requirements that are imposed on the structure of an ontology.

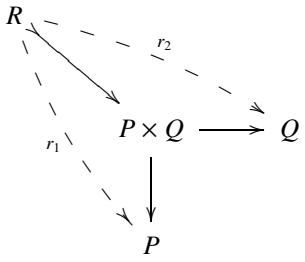
As an example, let us consider the following two partial orders:



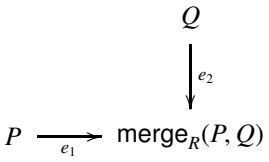

We assume that some elements of these structures are known to be equivalent. This is expressed by a relation $R \subseteq P \times Q$ (usually called an *ontology mapping*) that we define as $R = \{(a, 1), (b, 2), (c, 4), (f, 5), (g, 3)\}$. A reasonable result of merging the posets $P$ and $Q$ would then be the following structure:




Observe that all elements related by $R$ are indeed identified, but that some additional identifications are necessary to obtain a partially ordered set. Categorically, we can already specify the data that we have considered for such an operation. The given situation is depicted in the following diagram:




The shape of the arrow from $R$ to $P \times Q$ indicates that it defines a subobject (a monomorphism). The dotted arrows $r_1$ and $r_2$ are those that are obtained by composing the projections of the product with this monomorphism. They project every pair of elements of $R$ to its first and second component, respectively. Now the result of merging $P$ and $Q$ is not just some poset $\mathsf{merge}_R(P, Q)$, but also the two obvious embeddings of $P$ and $Q$ into $\mathsf{merge}_R(P, Q)$. As a diagram, we obtain:

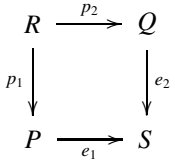



The property that $R$-related elements are identified can now be expressed in terms of functions: we find that, for any pair $(p, q) \in R$, $e_1(p) = e_2(q)$. Still a better way to express this for arbitrary morphisms is to say that $e_1 \circ r_1 = e_2 \circ r_2$.

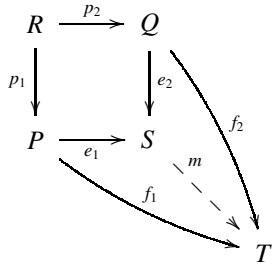
This condition alone, however, does not suffice. Usually, there are many objects for which $e_1 \circ r_1 = e_2 \circ r_2$ holds. Which of these is the one which we want to consider as the *merging* of $P$ and $Q$? Clearly, the merging shall not identify anything unnecessarily. This can be stated by means of another *universal property*, as follows.

**Definition 3** For a category $\mathscr{C}$, consider objects $R$, $P$, $Q$, and morphisms $p_1 : R \to P$ and $p_2 : R \to Q$. An object $S$ together with two morphisms $e_1 : P \to S$ and $e_2 : Q \to S$ is a *pushout* if it satisfies the following properties:

(i) $e_1 \circ p_1 = e_2 \circ p_2$, i.e. the following diagram *commutes*




(ii) For every other object $T$ and morphisms $f_1 : P \to T$ and $f_2 : Q \to T$, with $f_1 \circ p_1 = f_2 \circ p_2$, there is a unique morphism $m : S \to T$ such that $f_1 = e_1 \circ m$ and $f_2 = e_2 \circ m$. This situation is depicted in the following diagram.




Condition (ii) in this definition states the universal property of the pushout, requiring that it is in a sense the most general object that meets all requirements. Let us try to explain this a bit further. We have already understood that in this setting we can encode the ontology mapping (e.g. binary relation) $R$ conveniently, in that the resulting $S$ identifies (at least) all those elements which are related by $R$. But now we want to *avoid* the identification of other elements as much as possible. Intuitively, this means that a suitable pushout object needs to keep elements from both components as distinct as possible, while still implementing all necessary identifications, and without including irrelevant information. Enforcing the desired identifications was achieved by condition (i) in the above definition. Excessive identifications are prevented by requiring the *existence* of a factorization $m$: appending $m$ to $e_1$ and $e_2$ cannot make prior identifications undone, and hence a pair that was merged in $S$ can never be separated in an alternative solution $(T, f_1, f_2)$ if a suitable $m$ is known to exist. Finally, the possibility of including entirely unrelated information, like adding some elements not present in either $P$ or $Q$, is ruled out by assuring *uniqueness* of the factorization $m$: if $S$ would include elements that are neither in the image of $e_1$ nor in the image of $e_2$ then a valid factorization can assign these to arbitrary values in $T$ without loosing the factorization property – but this would result in many possible choices in place of $m$. In other words, having "unnecessary" elements in the $S$ would result in additional degrees of freedom in the choice of $m$, thus violating the required uniqueness.

Note also that we ignored our earlier restriction of $R$ being a subobject of the product $P \times Q$. However, by the universal property of the product, any object $R$ with functions to $P$ and $Q$ must have a unique factorization through the product, and hence does still capture part of the idea of a relation. Furthermore, such a generalized $R$ can also be viewed as a suitable background knowledge that both $P$ and $Q$ are based on. In spite of this generalization, $R$ is still an object of the considered category, i.e. it is itself an ontology with all necessary structure. We just dropped some side conditions on this object, such that some redundancy can be introduced into the ontology mapping if desired.

## How to put our approach into practice

Let us now see how our approach can be used as a guidance for ontology merging. We noted earlier that the notion of product hinges only on (1) a decision regarding the ontology representation language used, and (2) the structural properties which shall be preserved by a morphism. The situation for pushouts is similar, we just need a third decision,

namely (3) the fixing of an ontology mapping. Once these decisions have been made, the notion of pushout, and thus of the merging of two ontologies, is determined canonically, but one still needs to find a convenient concrete representation for the specified object (4). From there, conceptually sound algorithms for calculating both ontology mappings (5) and pushouts (6) can be devised.

Decisions need to be made step by step, and we propose the following workflow. Later steps, however, may indicate that earlier decisions need to be revised, and thus to retrace to earlier points.

1. *Decide on ontology representation language used.* This first step is probably the most unproblematic, since there are standard ontology languages around, and the specific application case will usually dictate the language. Potential candidates are e.g. F-Logic (Kifera, Lausen, & Wu 1995; Angele & Lausen 2004) and different variants of OWL (Antoniou & van Harmelen 2004).
2. *Determine what suitable morphisms are.* This step consists of describing the conditions which morphisms must satisfy. These conditions will primarily be dictated by the semantic interpretation of the ontology representation language chosen earlier, and by the specific requirements of the application case. Typical conditions could include the following.
   - The preservation of class hierarchies, i.e. functions shall be monotonic with respect to the *general class inclusion* orders on classes and/or roles.
   - The preservation of types (e.g. classes, roles, annotated objects).
   - The taking into account of model-theoretic logical properties, if featured by the underlying ontology representation language, like satisfiability, or the preservation of specific models.
   - The taking into account of proof-theoretic properties, i.e. such relating to particular inference methods chosen for reasoning with ontologies.
   - The preservation of language classes, e.g. by requiring that the merging of two OWL Lite ontologies shall not result in an OWL ontology which is not in OWL Lite.
3. *Determine what the ontology mapping is for this setting.* Usually, ontology mappings will be given by (binary) relations between elements of ontologies, indicating which elements shall be identified in the merging process. However, as the product of two ontologies may not always be described conveniently as a set of pairs of elements – as in the case of Set or Poset –, it needs to be understood at this stage, what the product really is, and thus what ontology mappings are in this setting.
4. *Determine what pushouts are for this setting.* While the characteristics of a pushout are fully determined by the previous steps, it is still necessary to find a particular instance of the pushout (both for the object and the embedding morphisms) in terms of the ontology language. This requires to define a possible result for arbitrary pushout operations and to show that it satisfies the formal requirements of a pushout. Difficulties at this stage arise from the fact that, like products, pushouts are not guaranteed to exist in general. Negative results may yield effective conditions for the existence of pushouts or even suggest a modification of the considered theory.
5. *Algorithmize how to obtain the mapping.* The issue of how to obtain suitable ontology mappings is a separate issue from the one discussed here, and will usually depend heavily on the application domain and on the ontology representation language chosen. Machine learning techniques may be used here together with linguistics-based approaches (see e.g. (Ehrig & Sure 2004)). Fuzzy relations usually obtained by such approaches may however have to be defuzzified at some stage, in order to obtain a precise ontology mapping which will be used for the merging.
6. *Algorithmize how to obtain the pushout.* At this stage, it is theoretically clear what the pushout – and thus the merged ontology – will be. Casting this insight into an algorithm may require a considerable amount of work. The practitioner may also choose at this step to forego an exact implementation of the merging, and settle for an approximate or heuristic approach for reasons of efficiency, while at the same time being guided by the exact merging result as the ontology to be approximated.

## Conclusion and further reading

We have argued that the problem of merging ontologies based on a given ontology mapping can be formulated conveniently in the language of category theory. This lead to the well-known definition of the categorical pushout construction, which describes ontological merging independently from the concrete implementation that was chosen. Since pushouts do not exist in all categories, this also yields general guidelines for devising systems of interrelated ontologies. Methods and insights from category theory could be used to assist in the development both of rigorous theoretical settings for ontology merging and of conceptually sound algorithms for practical implementations. Conversely, similar considerations can also be useful to validate merging constructions that have been conceived exclusively on practical grounds, since one may ask in which sense (in which category) a given merging process produces results of general validity.

In the remainder of this section, we mention some related work and point out connections to our approach. Despite the little formal work done so far on ontology merging, applications demanded tools for this task. Early work in aiding the user to create a merged ontology is included in the Chimaera Ontology Environment (McGuinness *et al.* 2000). The PROMPT Suite (Noy & Musen 2003) is a mapping and merging tool using both linguistical and structural similarity to propose merging candidates to the user. How to obtain ontology mappings by using machine learning techniques was studied in (Ehrig & Sure 2004). Finally we would like to mention a methodology for creating a merged ontology based on *Formal Concept Analysis* (Stumme & Maedche

2001). The ontologies are not explicitly merged, but rather a new ontology is created based on the underlying instances.

On the other hand, the importance of pushouts is certainly well-known among researchers working on theoretical grounds. Particular examples are (Jannink *et al.* 1998), where algebraic operations are illustrated by categorical considerations (though the authors keep the actual categories informal), or (Kent 2000; Schorlemmer, Potter, & Robertson 2002), where categories of local logics and logic infomorphisms are studied in a mathematically more rigorous way. The latter efforts are based on the theory of *information flow* (Barwise & Seligman 1997), which provides a formal model of how information can be conveyed between heterogeneous systems. Technically, this work builds on modelling the actual knowledge available in some (physical or logical) system as a so-called *classification*, a simple relation between the statements made in a system and the objects these statements may or may not refer to. This component is augmented with a collection of logical *constraints*, that describe what is actually believed about the system, and which can be regarded as a vehicle to control the loss of soundness and completeness that occurs when transforming data as suggested by this theory. Together with a classification one obtains a so-called *local logic* which yields the most complex ontological description considered in the information flow framework (IFF) (Kent 2005). Classifications and local logics in IFF are equipped with so-called *infomorphisms*[3] to obtain a category where constructions can be performed.

In spite of the simplicity of classifications, one can still capture a considerable amount of logical formalisms in such a relational scheme. This has been introduced earlier in the framework of *institution theory* (Goguen & Burstall 1992), where binary relations that capture the model theory $\models$ of a logic have been recognized as a convenient setting for unifying the treatment of different logics. Categorical pushouts were incorporated in this setting to implement modular program specification on a mathematical rigorous basis (see references in (Goguen & Burstall 1992)).

Though the study of the IFF is making considerable progress, we believe that it is necessary to study formal approaches to ontology construction for other settings as well. As explained, the IFF focuses on the use of local logics for ontological specification, while other formalisms such as F-Logic (Kifera, Lausen, & Wu 1995) and description logics (Baader *et al.* 2003) are widely used in implementations. Though IFF, like institution theory, could still subsume such logics, it may be asked whether the chosen infomorphisms are the adequate translation method for these concrete settings. Hence our suggestion is to search native translation mechanism for common ontological formalisms without loosing sight of the categorical modelling. The latter can provide guidance and reassurance for concrete implementations, but should not dictate what particular language is used by ontology engineers and end-users.

[3]Infomorphisms are also known as "Chu mappings." They are closely related to the notion of a "Galois connection" between complete lattices.

We stress the fact that this task is clearly distinguished from the problem of *ontology mapping*. The latter raises the question of how some particular ontologies are interrelated, while we are asking for general constraints that a well-behaved relationship between ontologies must satisfy. Ontology mapping then relates to the task of identifying concrete morphisms of this form, yielding a solution to the specific translation problem in which the user is interested. An overview of current approaches in ontology mapping is given in (Kalfoglou & Schorlemmer 2003). Much effort is put into obtaining sufficiently applicable mappings, possibly in an approximate or heuristic way. These attempts to actually compute something that can be put to immediate practical use are contrasted by the lack of *formal* definitions of *what* actually is to be computed. As in other areas (consider software-engineering), approaches that are not founded on a strong theoretical basis defy most mathematical reasoning on the resulting structures, eventually resulting in established engineering principles that, in spite of their practical convenience, can hardly provide features that require formal verification (like proving certain safety or reliability constraints[4]).

In addition, the use of the machinery of category theory depends on the formal (semantic) properties of the chosen way of representing ontologies. The high-level language of category theory provides simple and clearly specified interfaces for concrete implementations – whether concrete solutions will eventually be compatible with these interfaces remains to be seen.

Finally, for a more in-depth treatment of category theory, numerous textbooks and research monographs can be consulted. For an easy-paced introduction with a particular emphasis on set-theoretic constructions we strongly recommend (Lawvere & Rosebrugh 2003). Another very accessible textbook is (McLarty 1992). For a more extensive treatment of pushouts and related constructions, the reader may want to consult (Borceux 1994). Relations as subobjects of categorical products are discussed in much more detail in the first chapter of (Pedicchio & Tholen 2004), where order theory is developed in this general setting. General motivation for the application of category theory in computer science is given in (Goguen 1991), whereas mathematical details are omitted therein.

[4]As experience shows, the inability to prove these requirements formally does indeed result in most of todays software being neither safe nor reliable.

## References


Angele, J., and Lausen, G. 2004. Ontologies in F-logic. In Staab, S., and Studer, R., eds., *Handbook on Ontologies*. Springer. 29–50.

Antoniou, G., and van Harmelen, F. 2004. Web Ontology Language: OWL. In Staab, S., and Studer, R., eds., *Handbook on Ontologies*. Springer. 67–92.

Baader, F.; Calvanese, D.; McGuinness, D.; Nardi, D.; and Patel-Schneider, P., eds. 2003. *The description logic hand-*

*book: theory, implementations and applications*. Cambridge University Press.

Barwise, J., and Seligman, J. 1997. *Information flow: the logic of distributed systems*, volume 44 of *Cambridge tracts in theoretical computer science*. Cambridge University Press.

Borceux, F. 1994. *Handbook of categorical algebra 1: basic category theory*, volume 53 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press.

Ehrig, M., and Sure, Y. 2004. Ontology mapping — an integrated approach. In Bussler, C.; Davis, J.; Fensel, D.; and Studer, R., eds., *Proceedings of the First European Semantic Web Symposium*, volume 3053 of *Lecture Notes in Computer Science*, 76–91. Heraklion, Greece: Springer Verlag.

Goguen, J., and Burstall, R. 1992. Institutions: abstract model theory for specification and programming. *Journal of the ACM* 39.

Goguen, J. 1991. A categorical manifesto. *Mathematical Structures in Computer Science* 1:49–67.

Goguen, J. 2005. Three perspectives on information integration. In Kalfoglou, Y., and et al., eds., *Semantic Interoperability and Integration*, Dagstuhl Seminar Proceedings 04391.

Jannink, J.; Pichai, S.; Verheijen, D.; and Wiederhold, G. 1998. Encapsulation and composition of ontologies. In *Proceedings of the AAAI Workshop on AI & Information Integration*.

Kalfoglou, Y., and Schorlemmer, M. 2003. Ontology mapping: the state of the art. *The Knowledge Engineering Review* 18:1–31.

Kent, R. 2000. The information flow foundation for conceptual knowledge organization. In *Proceedings of the Sixth International Conference of the International Society for Knowledge Organization*.

Kent, R. E. 2005. Semantic integration in the Information Flow Framework. In Kalfoglou, Y., and et al., eds., *Semantic Interoperability and Integration*, Dagstuhl Seminar Proceedings 04391.

Kifera, M.; Lausen, G.; and Wu, J. 1995. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM* 42.

Lawvere, F. W., and Rosebrugh, R. 2003. *Sets for mathematics*. Cambridge University Press.

McGuinness, D.; Fikes, R.; Rice, J.; and Wilder, S. 2000. The Chimaera ontology environment. In *Proceedings of the 17th National Conference on Artificial Intelligence (AAAI)*, 1123–1124.

McLarty, C. 1992. *Elementary categories, elementary toposes*. Clarendon Press.

Noy, N. F., and Musen, M. A. 2003. The PROMPT suite: interactive tools for ontology merging and mapping. *International Journal of Human-Computer Studies* 59(6):983–1024.

Pedicchio, M. C., and Tholen, W., eds. 2004. *Categorical foundations: special topics in order, topology, algebra, and sheaf theory*, volume 97 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press.

Schorlemmer, M.; Potter, S.; and Robertson, D. 2002. Automated support for composition of transformtional components in knowledge engineering. Technical Report EDI-INF-RR-0137, Division of Informatics, University of Edinburgh.

Stumme, G., and Maedche, A. 2001. FCA-MERGE: Bottom-up merging of ontologies. In *IJCAI*, 225–234.