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,kroetzsch,ehrig,sure}@aifb.uni-karlsruhe.de

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. We study 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. 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.

Categorical pushouts are already considered in some approaches to ontology research (Jannink *et al.* 1998; 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. For our particular case the treatment will focus on the 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.

A long version of this paper with a tutorial character is available from the first author's homepage.

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 is often termed *ontology merging*, and will be discussed herein. For the sake of simplicity, our examples will use is-a hierarchies in order to explain the abstract notions involved.

The mathematical structure of is-a hierarchies is simply that of partially ordered sets, posets in short. Two posets 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 posets, other mathematical spaces may suggest different kinds of morphisms: Vector spaces are considered with linear functions, groups with group homomorphisms, topological spaces with continuous functions, etc. 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 cate-

Copyright © 2005, American Association for Artificial Intelligence (www.aaai.org). All rights reserved.

gory, a directed graph only has to include a *composition* operation (denoted $\circ$) for pairs of compatible arrows, satisfying some simple axioms that are typical for the composition of functions and the relational product. A precise definition is found in the aforementioned full version of this paper.

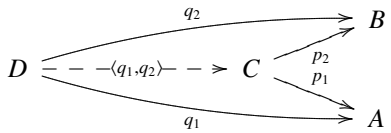
Specific relationships between objects can now be defined in purely categorical terms. Let us explore the notion of equivalence (or, speaking categorically, *isomorphism*) for the category of posets and monotone functions, where composition of morphisms is just usual composition of functions. Restating 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. 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 1 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:



For example, in set theory for the 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. We also remark that products do not necessarily exist in every category.

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.

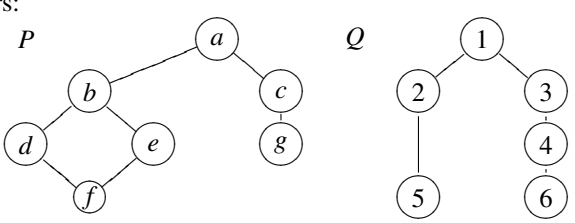
The categorical product definition also turns out to be suitable to model many well-known product constructions. For instance, when considering posets and monotone functions one obtains the usual product order, i.e. the cartesian product of the two sets, ordered such that a pair (a, b) is below a pair (c, d) whenever a is below c and b is below d.

By means of the product construction, 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 morphism $r : D \to (A \times B)$ 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.

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 $R \subseteq P \times Q$

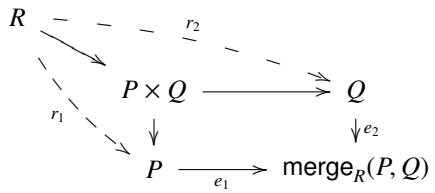


Figure 1: The pushout construction

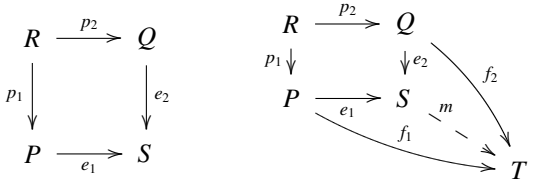
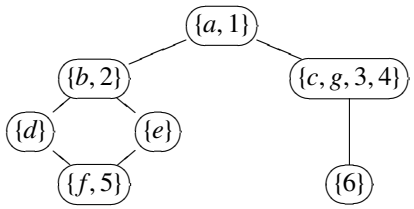


Figure 2: Diagrams for the pushout

(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 Figure 1. The dotted arrows r_1 and r_2 are those that are obtained by composing the projections of the product with the morphism from R to $P \times Q$. 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 $\mathsf{merge}_R(P,Q)$. 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 2 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 diagram in Figure 2 (left) *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 Figure 2 (right).

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.

How to put our approach into practice

Let us now see how our approach can be used as a guidance for ontology merging. 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 (Kifer, Lausen, & Wu 1995) and different variants of OWL (OWL 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 sets or posets –, 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.

Conclusions

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.

Acknowledgements We are grateful for helpful comments by the referees on the subject and purpose of this paper. We also acknowledge support by the by the European Commission under contracts IST-2003-506826 SEKT and FP6-507482 KnowledgeWeb, and by the German Federal Ministry of Education and Research (BMBF) under the SmartWeb project. The expressed content is the view of the authors but not necessarily the view of any of the projects as a whole.

References

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. 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*.

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.

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

2004. Web ontology language (OWL). `www.w3.org/2004/OWL/`.

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.