Three Perspectives on Information Integration

Joseph Goguen
University of California at San Diego, Dept. Computer Science & Engineering
9500 Gilman Drive, La Jolla CA 92093–0114 USA

0 Introduction

This note has three main sections, corresponding to three different kinds of contribution to Dagstuhl Seminar 04391, Semantic Interoperability and Integration, held from 20 to 24 September 2004. These sections respectively concern: (1) a brief sociology of a science lab, elucidating the goals and methods of a particular group of potential users of technology for integration and interoperability; (2) mathematical foundations for information integration, based on ideas from category and institution theories; and (3) tool support for information integration, with an emphasis on mapping tools. The final section of the paper gives some conclusions, and the appendix contains some mathematical details for the second topic.

One of the more striking observations that a sociologist might make about this seminar, as well as the field it addresses, is that a great diversity of disciplines (and individuals) are involved. On this particular occasion, perhaps the most prominent were artificial intelligence, databases, and semantic theory, all from within computer science, although the perspectives of philosophy, mathematics, engineering, and commerce were also represented. This diversity is natural, given the broad and pervasive nature of the subject addressed, namely the integration of information. This topic is timely, given the recent explosion of the world wide web, as well as important, given the ever increasing dependence of society on information technology, and the ever rising expectations for what it can accomplish. On the other hand, the systematic study of the issues involved is clearly in its infancy, and we have relatively little idea of what is and is not in principle possible, of what directions are most worth pursuing, and of what methods are most likely to be productive. The thoughts that follow represent my own attempts to come to grips with such problems, and should be understood as having a preliminary nature, more in the nature of explorations of some unexpected connections, rather than final reports. Nevertheless, I hope that they may be of some value in providing an orientation for future explorations.

Acknowledgements: I thank the participants in Dagstuhl Seminar 04391, Semantic Interoperability and Integration, for their enthusiasm and many

Dagstuhl Seminar Proceedings 04391
Semantic Interoperability and Integration
http://drops.dagstuhl.de/opus/volltexte/2005/38

comments; of course, any remaining bugs are my own fault. This work was partially supported by the National Science Foundation under Grant No. ITR 0225676, the Science Environment for Ecological Knowledge (SEEK) project, the goal of which is to provide an integrated information infrastructure to support a distributed community doing long term ecological research. The new concepts and results in Appendix A were developed in collaboration with Grigore Roşu and Răzvan Diaconescu, and the proofs (which are omitted here) were provided by Grigore Roşu.

1 Sociology of a Science Lab

This section is a digest of my contribution to a panel on light weight versus heavy weight approaches to information integration. I thought it would be valuable to see what "light" and "heavy" might mean to a particular user group, as well as what their goals and methods are, as a way of addressing what technologies we should be trying to develop. An important caveat is that user goals and methods, as well as the technologies that they currently use, vary greatly from one group to another; what follows is just one case study among many that are possible, and that should be expected to yield rather different results.

Despite my reputation as a formalist, I am very interested in social aspects of information technology. This brief penal presentation was a report on a certain class of potential users of ontologies, web services, etc.; it can be seen as the context of the sociology of science and technology. Although I am not a professional in this area, I have had the benefit of working with Geoff Bowker and Leigh Star at UCSD, who are true experts, and whose fine book [3], I recommend very highly for its treatment of classifications and standards.

I have spent time talking with and observing ecologists in the SEEK project, with which I am affiliated, and these remarks are based on these interviews and observations. Scientists in general want to control the data they use, to get it all in one place, clean it, get consistent formats, units, annotations, etc. They do not want to use a distributed workflow system unless they have to do so. Much information (probably most) is in spreadsheets and structured files, not in XML or relational databases; metadata is informal, it it exists at all. Processing is done in the simplest possible way to achieve the given goal, e.g., using Perl scripts, standard statistical tools, and specific models developed (often in the same lab) for the particular experimental situation at hand. There is much more interest in workflows than in ontologies, but only in an informal, quick and dirty, sense. Scientists want to do science, to establish and publish their results with as little delay as possible, and are often in competition with others seeking to solve similar problems.

Scientists doing long term ecological research face particularly difficult problems in data integration, since the horizon for meaningful results is at least 30 years. My panel presentation described two examples of data integration, in colorful stories about difficulties arising in reconciling old data with current data, one about soil samples, and the other about taxonomic systems. The latter turn out to be much more complex than I had imagined, subject to debates among specialists that appear both arcane and intense to outsiders.

A topic that I only alluded to in my presentation, but would have liked to have had time to develop, is the nature of information. Computer scientists often assume that information is stable, objective, and determinate. But studies of what happens in real science labs suggest a very different view, of work that might be called “information manufacture,” which is process oriented, highly social, and often indeterminate. These points are well supported in a famous study by [27] based on field work from 1975 to 1977 in the Guillemin lab of the Salk Institute. Similar points about ontologies are raised in [16], drawing not only on sociology of science, but also on phenomenology, ethnomethodology, cognitive linguistics, and psychology; a compatible social theory of information is given in [12].

2 Foundations of Information Integration

This section summarizes and expands informal remarks made in working groups on my efforts to provide a rigorous foundation for information integration that is not tied to any specific representational or logical formalism, by using category theory to abstract away from representations, and institutions to axiomatize the notion of logic. The main reference for this approach is [15], from which much of the material below has been taken. The approach is motivated by the plethora of representation schemes and logics used for information. For example, data may be in a spreadsheet, ascii file, relational database, XML file, etc.; with luck, there may be some “metadata” describing the structure of the data, e.g., an XML Schema for XML data, although in practice, this is often missing. Metadata in general consists of syntactic declarations, called a **signature**, plus axioms over those declarations, together giving what is called a **theory**. Due to the variety of different incompatible formalisms, it can be very difficult to integrate theories. However, it helps if they are based on a formal logic, which in fact many ontologies and schemas are today. This explains why we must deal not just with the structure of representation, but also with the logic in which this structure is expressed. Category theory provides a language that can achieve the required level of generality, and the following assumes some familiarity with its basics; there are many places to learn such material, including [32, 21, 10, 11]; also, the appendix to this paper includes a brief summary with some of the main definitions.

Applications to ontologies, e.g., [2, 22, 26, 25], were a major motivation for [14], which generalizes and extends the information flow and channel theories of [1], using the language of institutions; it also follows the lead of [24] by combining this with the formal conceptual analysis of [6] and the lattice of theories approach of [34]; in addition, it draws on the categorical general systems theory of [7, 8].

The greater generality of institutions over classifications, local logics, concept lattices, concept graphs, etc. allows doing information integration over arbitrary logics. Institutions abstract and generalize Tarski's "semantic definition of truth" [35], the most significant ingredient of which is a relation of *satisfaction* between models and sentences. An institution consists of a category of signatures, categories (or sets) of sentences, categories (or sets) of models, and a relation of satisfaction between them, each of which is parameterized by (i.e., functorial over) signatures. This parameterization allows an elegant treatment of examples where part of a situation is fixed while another part is allowed to vary, e.g., the function symbols used in equational logic vary, while the logic remains fixed. The basic reference is [17], and the latest version is in [19], which focuses on variants of the institution morphism notion. Many logical systems have been shown to be institutions, including first order logic, many sorted equational logic, Horn clause logic, many variants of higher order and of modal logic, and much more; it seems that essentially any logical system has a corresponding institution. A number of deep model theoretic results have been extended from first order logic to arbitrary institutions by Diaconescu, e.g., [4].

Unfortunately, there is currently no easy introduction to institutions, although a brief intuitive introduction is given in Section 2 of [15]. Alternatively, an exposition of institutions without any category theory is given in [20], the aim of which is to give semantics for a powerful extension of the Ada module system; however, the technical work of the paper is quite a bit more complex than it would have been if categorical language had been used.

Every institution has notions of signature and sentence, and a set of sentences over a common signature is a theory. As discussed in [17, 9], , and other publications, colimits enable powerful methods for structuring and combining theories, including inheritance, sums over shared subtheories, renaming parts of theories, and (best of all) parameterizing and instantiating theories. This goes far beyond the (generalized) Boolean operations of [6] and [34]; moreover, it provided the basis for the powerful module system of the ML programming language, though ML does not have all the functionality that is defined in [9] and implemented in the OBJ languages under the name "parameterized programming"; these ideas also influenced the module systems of C++, Ada, and LOTOS. Till Mossakowski [29] has built a theorem proving system that works over a variety of institutions, and so can be used for proving properties of heterogeneous theories, building also on Diaconescu's Grothendieck institution construction [4], which is extended in [14, 15] to ontologies.

A useful notion from category theory is that of a **relation** in a category $\mathbb{C}$: it consists of three objects, say A, B, R, and two morphisms, say $p_1 \colon R \to A$ and $p_2 \colon R \to B$. One can think of R as containing pairs (a, b) with $a \in A, b \in B$, and of p_1, p_2 as **projection** maps. The usual calculus of relations lifts to this very general setting, with modest assumptions on that $\mathbb{C}$ by defining composition with pullbacks. Then associativity of composition follows, and converse, union, intersection, etc. can be defined, and have their usual properties. The join of relations in database theory is also a special case. Binary relations generalize to polyadic relations, which are families $p_i \colon R \to A_i$ where i ranges over some set I, and one can again prove soundness of laws given in various axiom systems for polyadic relational calculus.

There is a dual notion of **co-relation**, consisting of three objects A, B, C and two **injection** maps, $f_1 \colon A \to C$ and $f_2 \colon B \to C$. These also generalize to the polyadic case, as families $f_i \colon A_i \to C$, giving rise to a calculus of co-relations that is dual to but less well known than that for relations. One of the most basic concepts in [1], the **channel**, is a co-relation, $f_i \colon A_i \to C$ for $i \in I$, in the category of classifications and infomorphisms, with $\mathbb{C}$ called its **core**. Looking at only the tokens, a channel yields a relation that "connects" each token c in its core to the tokens $f_i(c)$ in its components A_i. A "cover" of a diagram is defined in [1] to be a channel over that diagram such that every triangle formed by an infomorphism in the diagram and the two injections, from its source and target, commutes. This is exactly the categorical notion of a **co-cone**, for their special case; similarly, the "minimal covers" of [1] are **colimits**, although [1] uses the term “limit” for this concept, perhaps because the tokens are more concrete than types. Category theorists often use the term **apex** instead of “core,” for both relations and co-relations.

It is very encouraging that the notion of co-relation is essentially the same as that of local-as-view in database theory (this notion is defined in Section 4), and blending in cognitive linguistics [5], as well as channel in information flow. It is therefore reasonable to suggest that co-relations, co-cones, and co-limits are the obvious way to integrate any collection objects, as already suggested in [7, 11]. In the context of databases, the dual global-as-view approach corresponds to relations; it is less general, but more efficient for query answering. It is interesting to notice that in concrete cases, a co-cone gives rise to a partial map between the input spaces, connecting those elements mapping to the same element under the injections; this “emergent” partial map is what most schema and ontology mapping tools seek to construct, and it is also an important aspect of the theory of metaphor developed in cognitive linguistics by [5]. Computer scientists have used the terms “alignment” and “articulation” for this process, but its relation to more general notions of integration (for which terms like “fusion,” “merging” and “reconciliation” have been used) has remained somewhat mysterious. As in the set case, total functions are a special case of relations, which one can feel fortunate to find in practice.

It is natural to consider information integration over a subtheory of shared material, as in the blends of cognitive linguistics. For co-relations with a shared subtheory (denoted G in Figure 1), pushouts (which are a special kind of colimit) give an optimal blend, and dually, pullbacks are optimal in the relational case.

$$\begin{tikzcd}
 & B & \\
I_1 \arrow[ur] & & I_2 \arrow[ul] \\
 & G \arrow[ul] \arrow[ur] \arrow[uu] &
\end{tikzcd}$$

Fig. 1. A Blend Diagram

But in many practical situations, it seems too optimistic to expect this kind of optimality; instead, pragmatic optimality criteria like those used in cognitive linguistics [5] may be more appropriate. Factors that complicate integration in real problems include incomplete information, inconsistent information, different levels of certainty and granularity of information, and different goals for the result of integration. However, the optimality principles in chapter 16 of [5] seem to be most appropriate for "common sense" blends, and so are unlikely to be appropriate for situations with more specialized goals. For example, the poetry of Pablo Neruda was found [18] to have some particularly creative blends that used criteria opposite to those given in [5]; one such is the phrase "a water of beginnings and ashes" at the end of the first stanza of *Walking Around*; also, the *Duino Elegies* by Rainer Maria Rilke contains phrases that blend parts from very different domains, such as "the cheap winter hats of fate" in the fifth elegy. A conceptual blending algorithm is described in [18], and some poetry generated using it is given there. The implementation is based on a formalization of blending given in [13], using ideas from algebraic specification, semiotics, and $\frac{3}{2}$-categories. The latter is described in the appendix of this paper, which shows how it supports a great variety of optimality principles.

3 Tools for Information Integration

A great deal of sophisticated research has been done on integration in the database community, including logical semantics, building tools, and experimenting with real data and real users; [23] also surveys some of this research. A common database integration scenario, called **local-as-view**, is exactly a co-relation in a category of schemas and views. Here, the apex is the schema of a global (virtual) database, which integrates the information in the local databases which map to it. Queries are posed over the global schema, and then translated into local queries, the results of which are integrated to form an answer to the original query. A good deal of sophisticated research has been devoted the improving the efficiency of this for various classes of schemas and queries; this subarea is known as query optimization.

The main prerequisite for making such an approach work is to have the views from the global to the local schemas. For this reason, considerable effort has been devoted to schema mapping tools; [33] surveys some of this work. One of these tools, called SCIA, has been designed and built by the UCSD Meaning and Computation Laboratory; it is an interactive tool for constructing maps between DTDs and/or XML Schemas, and is packaged with software for integrating and querying XML databases that have such schemas [14, 30, 31, 36]. Because fully automatic schema mapping generation is infeasible, this tool attempts to minimize total user effort by identifying the critical decision points, where a small user input can yield the largest reduction of future matching effort. A **critical point** is where a core context has either no good matches, or else has more than one good 1-to-1 match, where **core contexts** are the most important contextualizing elements for tags within their subtrees. Core contexts typically have a large subtree, and are identified by heuristics and/or by user input. In interactive mode, the tool solicits user input at critical points, and iterates until both the user and the tool are satisfied; in automatic mode, it does just one pass using default strategies. Each pass has four steps: linguistic and data type matching; structural matching; context checking; and combining match results. Figure 2 below depicts the matching process for one source and one target schema. Other tools only try to find the easiest 1-to-1 matches, leaving all other difficult matches for the user to do by hand. A major result of our research is that this approach can significantly reduce total user effort.

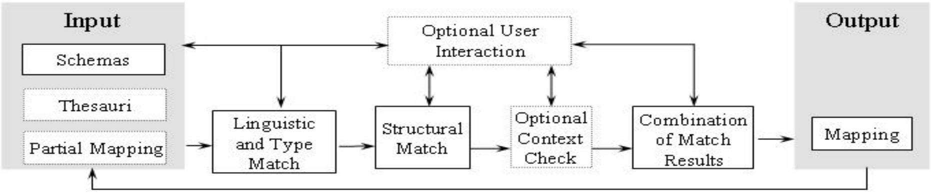

Fig. 2. Architecture of the Matching Process

An unusual, perhaps unique, feature of our tool is its ability to handle semantic functions and conditions. In this context, the term “semantic function” refers to operations on basic data types, such as arithmetic operations on numbers, or operations on lists such as head, tail, and append. For example, mappings between a schema that has first-name and last-name to one that has full-name will require such functions to pull apart and put together names. The term “condition” refers to mappings that must do different things under different circumstances; for example, if one student database puts majors and non-majors in the same relation, while another puts them in different relations, then a condition will be needed to map the first to the second. Most other tools do not treat functions and conditions at all, or else leave them for a different tool, e.g., view generation.

4 Conclusions

It is not coincidental that this note is highly interdisciplinary; I believe an interdisciplinary approach is called for by the problems addressed. Experience shows that purely technical solutions, based on what it is feasible and fun to do, rarely deliver what users really need. Similarly, explorations based on mathematical aesthetics often fail to connect closely with applications; for mathematics to be useful, it should address practical questions that need to be answered. Finally, for technology to reach real users, it helps if the results of social analyses and mathematical foundations are embedded in tools. Doing so in general ignites cycles of improvements to the tools, the foundations, and the social understanding.

One conclusion that can be drawn from interactions at the seminar and from papers in the literature, is that information integration is an important problem, solutions to which are being actively pursued from very diverse directions, with considerable excitement and promise, but with relatively little coordination. Perhaps further connections will emerge, as unexpected as those with institutions and blending that are proposed in this paper. It will surely be interesting to see what happens over the next few years.

A Some Mathematical Details

This appendix gives some details of mathematics mentioned in the body of the paper, closely following Appendix B of [13]; the concepts were developed in collaboration with Grigore Roşu and Răzvan Diaconescu, and the proofs (which are omitted here) are due to Grigore Roşu, and can be found in [13]. Although self-contained, this material may be difficult for readers who are not already familiar with category theory. The essential intuition behind categories is that they capture mathematical structures; for example, sets, groups, vector spaces, and automata, along with their structure preserving morphisms, each form a category, and their morphisms are an essential part of the picture. The theories and their morphisms over any institution also form a category [17], and so do the sign systems and semiotic morphisms of [13].

Definition 1. *A* **category** $\mathbb{C}$ *consists of: a collection, denoted* $|\mathbb{C}|$*, of* **objects***; for each pair* A, B *of objects, a set* $\mathbb{C}(A, B)$ *of* **morphisms** *(also called* **arrows** *or* **maps***) from* A *to* B*; for each object* A*, a morphism* 1_A *from* A *to* A *called the* **identity** *at* A*; and for each three objects* A, B, C*, an operation called* **composition**, $\mathbb{C}(A, B) \times \mathbb{C}(B, C) \to \mathbb{C}(A, C)$ *denoted*

";" such that $f;(g;h) = (f;g);h$ and $f;1_A = f$ and $1_A;g = g$ whenever these compositions are defined. We write $f\colon A \to B$ when $f \in \mathbb{C}(A,B)$, and call A the **source** *and B the* **target** *of f.* □

We now review the notions of pushout, cone and colimit for ordinary categories, relate this to blending, and then generize to $\frac{3}{2}$-categories, which better capture what blending is supposed to do. The intuition for colimits is that they put some components together, identifying as little as possible, with nothing left over, and with nothing essentially new added [11]. This suggests that colimits should give some kind of optimal blend. Although this is true, we will see that this kind of optimality has problems, which imply that the traditional categorical notions are not appropriate for blending. Nevertheless, they provide a good place to begin our journey of formalization.

Definition 2. *Given a category $\mathbb{C}$, a* **V** *in $\mathbb{C}$ is a pair $a_i\colon G \to I_i$ ($i = 1,2$) of morphisms, and a* **cone** *with* **apex** *B over a V a_1, a_2 is a pair $b_i\colon I_i \to B$ ($i = 1,2$) of morphisms; then a_1, a_2 and b_1, b_2 together are said to form a* **diamond** *(or a* **square***). The cone (or its diamond)* **commutes** *iff $a_1;b_1 = a_2;b_2$, and is a* **pushout** *iff given any other commutative cone $c_i\colon I_i \to C$ over a_1, a_2, there is a unique arrow $u\colon B \to C$ such that $b_i;u = c_i$ for $i = 1,2$.*

A **diagram** *D in a category $\mathbb{C}$ is a directed graph with its nodes labeled by objects from $\mathbb{C}$ and its edges labeled by arrows from $\mathbb{C}$, such that if an arrow $f\colon D_i \to D_j$ labels an edge $e\colon i \to j$, then the source node i of e is labeled by D_i and the target node j of e is labeled by D_j. A* **cone over** *D is an object B, called its* **apex***, together with an arrow $b_i\colon D_i \to B$, called an* **injection***, from each object of D to B, and is* **commutative** *iff for each $f\colon D_i \to D_j$ in D, we have*[1] *$b_i = f;b_j$. A* **colimit** *of D is a commutative cone $b_i\colon D_i \to B$ over D such that if $c_i\colon D_i \to C$ is any other commutative cone over D, then there is a unique $u\colon B \to C$ such that*[2] *$b_i;u = c_i$ for all nodes i of D.* □

Pushouts are the special case of colimits where the diagram is a V; see Figure 1. (It might appear that there is a discrepancy in the definitions, because pushouts are not required to have an arrow $G \to B$. But when the diagram is a V, this missing arrow is automatically provided by the morphism $a_1;b_1 = a_2;b_2$.)

There is a short proof that any two colimits of a diagram D are isomorphic. Let the cones be $b_i\colon\ D_i \to B$ and $b'_i\colon\ D_i \to B'$. Then there

[1] These equations are called **triangles** below, after the corresponding three node commutative diagrams.

[2] These equations may also be called "triangles" below.

are unique arrows $u\colon\ B \to B'$ and $v\colon\ B' \to B$ satisfying the appropriate triangles, and there are also unique arrows $B \to B$ and $B' \to B'$ satisfying their appropriate triangles, namely the respective identities 1_B and $1_{B'}$; but $u;v$ and $v;u$ also satisfy the same triangles; so by uniqueness, $u;v = 1_B$ and $v;u = 1_{B'}$.

If blends are commutative cones, it follows that colimits should be some kind of optimal blend. For example, the "houseboat" blend of "house" and "boat" is a colimit. But the fact that colimits are only determined up to isomorphism seems inconsistent with this, because the names attached to the elements in a blend are important; that is, isomorphic cones do *not* represent the same blend. This differs from the situation in group theory or topology, where it is enough to characterize an object up to isomorphism. However the requirement (also motivated by the examples in [13]) that the injections should be inclusions to as great an extent as possible, causes the actual names of elements to be captured by blends, and thus eliminates this problem.

Another problem with defining blends to be commutative cones is that not all blends actually have fully commutative cones; for "house" and "boat", only the "houseboat" blend has all its triangles commutative. However, the following notion solves this problem: The **auxiliary morphisms** of D are a subset whose triangles are not required to commute; these morphisms can be removed from D, to yield another diagram D' having the same nodes as D. Commutative cones over D' are then cones over D that commute except possibly over the auxiliary morphisms. Now we can also form a colimit of D', to get a "best possible" such cone over D. It therefore makes sense to define a blend to be a commutative cone over a diagram with the auxiliary morphisms removed.

One advantage of formalization is that it makes it possible to prove general laws, in this case, laws about blends based on general results from category theory, such as that "the pushout of a pushout is a pushout." This result suggests proving that "the blend of a blend is a blend," so that compositionality of the kind of optimal blends given by pushouts follows from the above quoted result about pushouts. The meaning of these assertions will be clearer if we refer to Figure 3.

Here we assume that b_2, b_3 is a blend of a_2, a_3, and c_2, c_3 is a blend of a_1, b_2, i.e., that $a_2;b_2 = a_3;b_3$ and $a_1;c_2 = b_2;c_3$; then the claim is that $c_2,\ b_3;c_3$ is a blend of $a_2;a_1,\ a_3$, which follows because $a_2;a_1;c_2 = a_3;b_3;c_3$. Using the notation $a_2 \diamond a_3$ for an arbitrary blend of a_2, a_3, we can write this result rather nicely in the form

$$a_1 \diamond (a_2 \diamond a_3) = (a_2;a_1) \diamond a_3\ ,$$

$$\begin{tikzcd}
 & \bullet & & \\
\bullet \arrow[ur, "c_2"] & & \bullet \arrow[ul, "c_3"'] & \\
 & \bullet \arrow[ul, "a_1"] \arrow[ur, "b_2"'] & & \bullet \arrow[ul, "b_3"'] \\
 & & \bullet \arrow[ul, "a_2"] \arrow[ur, "a_3"'] &
\end{tikzcd}$$

Fig. 3. Composing Pushouts

taking advantage of a convention that $a_1 \Diamond (a_2 \Diamond a_3)$ indicates blending a_1 with the left injection of $(a_2 \Diamond a_3)$ (the top left edge of its diamond).

The pushout composition result (proved e.g. in [21, 28]) states that if b_2, b_3 is a pushout of a_2, a_3, and c_2, c_3 is a pushout of a_1, b_2, then $c_2,\ b_3; c_3$ is a pushout of $a_2; a_1,\ a_3$. If we write $a_2 \bowtie a_3$ for the pushout of a_2, a_3, then this result can also be written neatly, as

$$a_1 \ \bowtie\ (a_2 \ \bowtie\ a_3) = (a_2; a_1) \ \bowtie\ a_3 \ .$$

We can also place a second blend (or pushout) on top of b_3 instead of b_2; corresponding results then follow by symmetry, and after some renaming of arrows can be written as follows:

$$(a_1 \Diamond a_2) \Diamond a_3 = a_1 \Diamond (a_2; a_3) \ .$$
$$(a_1 \ \bowtie\ a_2) \ \bowtie\ a_3 = a_1 \ \bowtie\ (a_2; a_3) \ .$$

We can further generalize to any pattern of diamonds: if they all commute, then so does the outside figure; and if they are all pushouts, then so is the outside figure. Another very general result from category theory says that the colimit of any connected diagram can be built from pushouts of its parts. Taken all together, these results give a good deal of calculational power for blending.

We now broaden our framework. The category of sign systems with semiotic morphisms has some additional structure over that of a category: it is an *ordered category*, because of the orderings by quality of representation that can be put on its morphisms. This extra structure gives a richer framework for considering blends; this approach captures at least some of what Fauconnier and Turner have called "emergent" structure, without needing any other machinery. Moreover, all the usual categorical compositionality results about pushouts and colimits extend to $\frac{3}{2}$-categories.

Definition 3. *A* $\mathbf{\frac{3}{2}}$**-category**[3] *is a category* $\mathbb{C}$ *such that each set* $\mathbb{C}(A, B)$ *is partially ordered, composition preserves the orderings, and identities are maximal.* □

Because we are concerned here with ordered categories, a somewhat different notion of pushout is appropriate, and for this notion, the uniqueness property is (fortunately!) lost:

Definition 4. *Given a V,* $a_i \colon G \to I_i$ $(i = 1, 2)$ *in a* $\frac{3}{2}$*-category* $\mathbb{C}$*, a cone* b_1, b_2 *over* a_1, a_2 *is* **consistent** *iff there exists some* $d \colon G \to B$ *such that* $a_1; b_1 \leq d$ *and* $a_2; b_2 \leq d$*, and is a* $\mathbf{\frac{3}{2}}$**-pushout** *iff given any consistent cone* $c_i \colon I_i \to C$ *over* a_1, a_2*, the set*

$$\{h \colon B \to C \mid b_1; h \leq c_1 \quad \textit{and} \quad b_2; h \leq c_2\}$$

has a maximum element. □

Proposition 1. *The composition of two* $\frac{3}{2}$*-pushouts is a* $\frac{3}{2}$*-pushout.* □

However, unlike the situation for ordinary pushouts, the composition of consistent diamonds need not be consistent, and two different $\frac{3}{2}$-pushouts need not be isomorphic; this means that ambiguity is natural in this setting. The following is another compositionality result for $\frac{3}{2}$-pushouts:

Proposition 2. *If the four small squares in Figure 4 are* $\frac{3}{2}$*-pushouts, then so is the large outside square.* □

$$\begin{tikzcd}
& & \bullet & & \\
& \bullet \arrow[ur, "d_1"] & & \bullet \arrow[ul, "d_2"'] & \\
\bullet \arrow[ur, "c_1"] & & \bullet \arrow[ul, "c_2"'] \arrow[ur, "c_3"] & & \bullet \arrow[ul, "c_4"'] \\
& \bullet \arrow[ul, "b_1"] \arrow[ur, "b_2"'] & & \bullet \arrow[ul, "b_3"] \arrow[ur, "b_4"'] & \\
& & \bullet \arrow[ul, "a_1"] \arrow[ur, "a_2"'] & &
\end{tikzcd}$$

Fig. 4. Composing Four $\frac{3}{2}$-Pushouts

[3] In the literature, similar structures have been called "one and a half" categories, because they are half way between ordinary ("one dimensional") categories and the more general "two (dimensional)" categories.

Passing from V's to arbitrary diagrams of morphisms generalizes $\frac{3}{2}$-pushouts to $\mathbf{\frac{3}{2}}$-colimits, and provides what seems a natural way to blend complex interconnections of meanings. The notion of consistent diamond extends naturally to arbitrary diagrams, as follows:

Definition 5. *Let D be a diagram. Then a family $\{\alpha_i\}_{i\in|D|}$ of morphisms is D-**consistent** iff $a;\alpha_j \leq \alpha_i$ whenever there is a morphism $a : i \to j$ in D. Similarly, given $J \subseteq |D|$, we say a family of morphisms $\{\alpha_i\}_{i\in J}$ is D-consistent iff $\{\alpha_i\}_{i\in J}$ extends to a D-consistent family $\{\alpha_i\}_{i\in|D|}$.* □

Fact 1: A diamond a_1, a_2, b_1, b_2 is consistent if and only if $\{b_1, b_2\}$ is $\{a_1, a_2\}$-consistent. □

Definition 6. *Let D be a diagram. Then a family $\{\alpha_i\}_{i\in|D|}$ is a $\frac{3}{2}$-**colimit** of D iff it is a cone and for any D-consistent family $\{\beta_i\}_{i\in|D|}$, the set $\{h \mid \alpha_i; h \leq \beta_i,$ for each $i \in |D|\}$ has a* maximum *element.* □

The following is another typical result that extends from ordinary colimits to $\frac{3}{2}$-colimits:

Theorem 1. *Let a* **W diagram** *consist of two V's connected at the middle top. If D is a W diagram, then a $\frac{3}{2}$-colimit of D is obtained by taking a $\frac{3}{2}$-pushout of each V, and then taking a pushout those two pushouts, as shown in Figure 4.* □

Extending our pushout notation $\bowtie$ to $\frac{3}{2}$-categories, the above result can be rather neatly written in the form

$$(a_1 \bowtie a_2) \bowtie (a_3 \bowtie a_4) = \mathit{Colim}(W) \ ,$$

where W is the bottom part of the diagram, with edges labeled a_1, a_2, a_3, a_4. A generalization of the above result implies that $\frac{3}{2}$-pushouts can be used to compute the $\frac{3}{2}$-colimit of any connected diagram. Observe that the notion of auxiliary morphism carries over to the framework of $\frac{3}{2}$-categories without any change.

It is very natural to wish that morphisms should be as defined as possible, should preserve as many axioms as possible, and should be as inclusive as possible, and these conditions define a quality ordering: given morphisms $f, g\colon A \to B$ between conceptual spaces A, B, define $f \leq g$ iff g preserves as much content as f, and preserves all axioms that f does, and is as inclusive as f (see [13] for details). More work is needed to determine whether this approach of designing orderings always works well, but for example, it works well for the house and boat example, since the most natural blend is an ordinary pushout, all the other good blends are $\frac{3}{2}$-pushouts, and blends that fail to preserve as much structure as they could are not any kind of pushout. The intuition here is that the ordering on morphisms induces an ordering on the $\frac{3}{2}$-colimit objects, such that they satisfy the corresponding optimality principle.

References

1. Jon Barwise and Jerry Seligman. *Information Flow: Logic of Distributed Systems.* Cambridge, 1997. Tracts in Theoretical Computer Science 44.
2. Trevor Bench-Capon and Grant Malcolm. Formalising ontologies and their relations. In *Proceedings of the 16th International Conference on Database and Expert Systems Applications (DEXA '99)*, pages 250–259. Springer, 1999. Lecture Notes in Computer Science, volume 1677.
3. Geoffrey Bowker and Susan Leigh Star. *Sorting Things Out.* MIT, 1999.
4. Răzvan Diaconescu. Grothendieck institutions. *Applied Categorical Structures*, 10:383–402, 2002.
5. Gilles Fauconnier and Mark Turner. *The Way We Think.* Basic, 2002.
6. Bernhard Ganter and Rudolf Wille. *Formal Concept: Mathematical Foundations.* Springer, 1997.
7. Joseph Goguen. Mathematical representation of hierarchically organized systems. In E. Attinger, editor, *Global Systems Dynamics*, pages 112–128. S. Karger, 1971.
8. Joseph Goguen. Categorical foundations for general systems theory. In F. Pichler and Robert Trappl, editors, *Advances in Cybernetics and Systems Research*, pages 121–130. Transcripta Books, 1973.
9. Joseph Goguen. Principles of parameterized programming. In Ted Biggerstaff and Alan Perlis, editors, *Software Reusability, Volume I: Concepts and Models*, pages 159–225. Addison Wesley, 1989.
10. Joseph Goguen. What is unification? A categorical view of substitution, equation and solution. In Maurice Nivat and Hassan Aït-Kaci, editors, *Resolution of Equations in Algebraic Structures, Volume 1: Algebraic Techniques*, pages 217–261. Academic, 1989.
11. Joseph Goguen. A categorical manifesto. *Mathematical Structures in Computer Science*, 1(1):49–67, March 1991.
12. Joseph Goguen. Towards a social, ethical theory of information. In Geoffrey Bowker, Leigh Star, William Turner, and Les Gasser, editors, *Social Science, Technical Systems and Cooperative Work: Beyond the Great Divide*, pages 27–56. Erlbaum, 1997.
13. Joseph Goguen. An introduction to algebraic semiotics, with applications to user interface design. In Chrystopher Nehaniv, editor, *Computation for Metaphors, Analogy and Agents*, pages 242–291. Springer, 1999. Lecture Notes in Artificial Intelligence, Volume 1562.
14. Joseph Goguen. Data, schema and ontology integration. In *Proceedings, Workshop on Combinination of Logics*, pages 21–31. Center for Logic and Computation, Instituto Superior Tecnico, Lisbon, Portugal, 2004.
15. Joseph Goguen. Information integration in instutions, 2004. To appear in a volume dedicated to Jon Barwise, edited by Larry Moss.

16. Joseph Goguen. Ontology, society, and ontotheology, 2004. To appear, *Proceedings, Conference on Formal Ontology in Information Systems (FOIS'04)*. Available at http://www.cs.ucsd.edu/ goguen/pps/fois04.ps.
17. Joseph Goguen and Rod Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, January 1992.
18. Joseph Goguen and Fox Harrell. Style as a choice of blending principles, 2004. To appear, *Proceedings, Workshop on Style and Meaning in Language, Art Music and Design.*
19. Joseph Goguen and Grigore Roşu. Institution morphisms. *Formal Aspects of Computing*, 13:274–307, 2002.
20. Joseph Goguen and William Tracz. An implementation-oriented semantics for module composition. In Gary Leavens and Murali Sitaraman, editors, *Foundations of Component-based Systems*, pages 231–263. Cambridge, 2000.
21. Robert Goldblatt. *Topoi, the Categorial Analysis of Logic.* North-Holland, 1979.
22. Yannis Kalfoglou and Marco Schorlemmer. Information-flow-based ontology mapping. In Robert Meersman and Zahir Tari, editors, *Proc. Intl. Conf. on Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems*, volume 2519 of *Lecture Notes in Computer Science*, pages 1132–1151. Springer, 2002.
23. Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *Knowledge Engineering Review*, 18(1):1–31, 2003.
24. Robert Kent. Distributed conceptual structures. In Harre de Swart, editor, *Sixth International Workshop on Relational Methods in Computer Science*, pages 104–123. Springer, 2002. Lecture Notes in Computer Science, volume 2561.
25. Robert Kent. Formal or axiomatic semantics in the IFF, 2003. Available at suo.ieee.org/IFF/work-in-progress/.
26. Robert Kent. The IFF foundation for ontological knowledge organization. In Giorgio Ghelli and Gosta Grahne, editors, *Knowledge Organization and Classification in International Information Retrieval.* Haworth, 2003.
27. Bruno Latour and Steve Woolgar. *Laboratory Life.* Sage, 1979.
28. Saunders Mac Lane. *Categories for the Working Mathematician.* Springer, 1971.
29. Till Mossakowski. Heterogeneous specification and the heterogeneous tool set, 2004. Habilitation thesis, University of Bremen, to appear.
30. Young-Kwang Nam, Joseph Goguen, and Guilian Wang. A metadata integration assistant generator for heterogeneous distributed databases. In *Ontologies, DataBases, and Applications of Semantics for Large Scale Information Systems*, pages 1332–1344. Springer, 2002. Lecture Notes in Computer Science, volume 2519.
31. Young-Kwang Nam, Joseph Goguen, and Guilian Wang. A metadata tool for retrieval from heterogeneous distributed XML documents. In P.M.A. Sloot et al., editors, *Proceedings, International Conference on Computational Science*, pages 1020–1029. Springer, 2003. Lecture Notes in Computer Science, volume 2660.
32. Benjamin C. Pierce. A taste of category theory for computer scientists. Technical Report CMU-CS-90-113, Carnegie-Mellon University, 1990.
33. Erhard Rahm and Philip Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
34. John Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations.* Brooks/Coles, 2000.
35. Alfred Tarski. The semantic conception of truth. *Philos. Phenomenological Research*, 4:13–47, 1944.

36. Guilian Wang, Joseph Goguen, Young-Kwang Nam, and Kai Lin. Critical points for interactive schema matching. In Jeffrey Xu Yu, Xuemin Lin, Hongjun Lu, and YanChun Zhang, editors, *Advanced Web Technologies and Applications*, pages 654–664. Springer, 2004.