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 paper has three main sections, corresponding to three different kinds of contribution to Dagstuhl Seminar 04391, called Semantic Interoperability and Integration, and held from 20 to 24 September 2004. These sections respectively concern: (1) a brief sociology of a science lab, exploring 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 gives some conclusions, and the appendix contains some mathematical details for the second topic.

One of the more striking observations a sociologist might make about this seminar, and the field it addressed, is that a great diversity of disciplines are involved. On this occasion, perhaps the most prominent were artificial intelligence, databases, and semantic theory, all from within computer science, although perspectives from 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. The 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 in its infancy. The thoughts that follow are attempts to understand such problems, more explorations of unexpected connections than final reports. Nevertheless, I hope they may be of value as orientation for future exploration.

Acknowledgements: Thanks to participants in Dagstuhl Seminar 04391 for their enthusiasm and 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 long

term ecological research community. Thanks to Grigore Roşu and Răzvan Diaconescu for collaboration on results in Appendix A.

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, in view of their goals and methods, as a way to address 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 different groups are likely to yield different results.

Despite my reputation as a formalist, I am also very interested in social aspects of information technology, and my brief panel presentation was conceived in the spirit of 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, whose book [6] I highly recommend for its fascinating 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 this experience. 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. 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 a 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 quickly as possible, since they are often in competition with others.

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 alluded to in my presentation that I would have liked more 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 Bruno Latour [36] based on field work from 1975 to 1977 in the Guillemin lab of the Salk Institute. Similar points about ontologies are raised in [23], drawing on phenomenology, ethnomethodology, cognitive linguistics, and psychology, as well as sociology of science; [18] gives a more theoretical perspective.

2 Foundations of Information Integration

This section summarizes an effort 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 particular representations, and institutions to abstract away from particular logics. The main reference for this project is [22], from which much of the material below is taken. The approach is motivated by the multitude of representation schemes and logics used for computer-based 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, but in practice this is often missing.

Metadata generally consists of syntactic declarations plus axioms over those declarations, forming what we call a **theory**. There is a growing tendency to express metadata in a formal logic, but the variety of different logics can greatly complicate integration, since we must deal not only with the structure of representation, assertions about what is represented, and the data itself, but also with the logics in which the metadata is expressed. Category theory provides a language that supports the necessary level of abstraction, and the following assumes some familiarity with its basics; there are many places to learn such material, such as [42, 30, 16, 17]; also, the appendix to this paper includes a very condensed summary of some main definitions.

Unfortunately, there is currently no easy introduction to institutions, although a brief intuitive introduction is given in Section 2 of [22], and

an exposition without category theory is given in [29][1]. Institutions axiomatize the notion of logic, by abstracting and generalizing Tarski's "semantic definition of truth" [45]. An institution consists of a category of *signatures* (for syntax declarations), sets (or categories) of *sentences* over those signatures, categories (or sets) of *models* over those signatures, and a relation of *satisfaction* between sentences and models. Parameterization by signature is functorial, and allows examples where part of a situation is fixed while another part can vary, e.g., the function symbols used in equational logic vary, while the logic remains fixed[2]. The basic reference for institutions is [25], and the latest version is in [28], 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. An impressive number of deep model theoretic results have been extended from first order logic to arbitrary institutions by Diaconescu, e.g., [8]. Till Mossakowski [38] 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 on Diaconescu's Grothendieck institution construction [8], which is also applied to ontologies in [21, 22].

The greater generality of institutions over classifications, local logics, concept lattices, concept graphs, etc. allows doing information integration over arbitrary logics. For any institution, a set of sentences over a common signature is a theory. As discussed in [25, 15], 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 [11] and [44]; 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 [15] and implemented in the OBJ languages under the name "parameterized programming"; these ideas also influenced the module systems of C++, Ada, and LOTOS.

[1] The aim of this paper was to give semantics for a powerful extension of the Ada module system; however, the technical work of that paper is significantly more complex than it would have been if categorical language had been used.

[2] This can be considered a instance of the triadic notion of meaning advocated by Charles Sanders Peirce in his semiotics [41].

Applications to ontologies, e.g., in [3, 31, 35, 34], were a major motivation for [22], which generalizes and extends the information flow and channel theories of [2] using the language of institutions, and follows [33] in combining this with formal conceptual analysis [11] and the lattice of theories approach [44], based on the Galois connection that exists between theories and models in any institution. In addition, [22] draws on the categorical general systems theory of [12, 13], and provides several new formalizations of database systems as institutions.

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 $p_1 : R \to A$ and $p_2 : R \to B$. 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 $\mathbb{C}$, by defining a composition of relations using 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 a special case. Binary relations generalize to polyadic relations, which are families $p_i : R \to A_i$ 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, $f_1 : A \to C$ and $f_2 : B \to C$. These also generalize to the polyadic case, as families $f_i : A_i \to C$, 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 [2], the **channel**, is a co-relation, $f_i : A_i \to C$ 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 [2] 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 [2] are **colimits**, although [2] uses the term "limit", perhaps because the tokens are more concrete than types[3]. Category theorists do not use the term "core," but often use the term **apex** for both relations and co-relations.

Co-relations (perhaps with shared subobjects, as discussed below) appear in local-as-view integration in database theory [7], blending in cogni-

[3] The categorical notion of limit is also of interest; for example, joins in relational databases are limits.

tive linguistics [10], channels in information flow [2], module composition[4] in programming [29], user interface composition [20], and merging different versions of software. The dual global-as-view approach to database integration can also be formulated as a co-relation; it is less general, but more efficient for query answering. 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 [10]. Computer scientists have used the terms "alignment" and "articulation" for this process, but the relation to more general notions of integration (for which terms like "fusion," "merging" and "reconciliation" have been used) has remained somewhat mysterious.

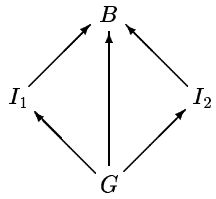

Fig. 1. Integration over a Shared Subobject

Many information integration problems have material which is known to be shared among some objects to be integrated, and which should therefore not appear more than once in the integrated object; Figure 1 shows a co-relation with a subobject G of shared material. In cognitive linguistics, such a diagram (though upside down from Figure 1) is called a "blend diagrams" and its objects are *conceptual spaces*, which consist of individual elements, and instances of binary relations between such elements [9]. For such a diagram, the special colimit called a *pushout* gives a blend that is optimal in a certain precise mathematical sense (for which see Appendix A). This is consistent with the view that co-relations, co-cones, and co-limits can integrate any kind of object, as suggested in [12, 17] and later for databases in [1]. However, in many practical situations, it is unrealistic to expect this kind of optimality; factors that complicate real integration problems include incomplete information, inconsistent information, different levels of certainty and granularity of information, and different goals for the result of integration. Moreover, the objects that result from pushouts are unique up to isomorphism, whereas integration can often be done correctly in more than one way.

[4] Especially the instantiation of generic or parameterized modules, the best known examples of which are the "functors" of the SML language [46].

For this reason, pragmatic optimality criteria like those used in cognitive linguistics [10] may be more appropriate than the so-called universal properties that characterize colimits. However, the optimality principles in chapter 16 of [10] are too informal to be implementable, which is why the poetry generation system developed by Fox Harrell with the author [27] uses purely structural optimality principles that formalize only some aspects of those in [10]. In addition, analysis in [27] of poetry of Pablo Neruda found some particularly creative blends for the generation of which the conceptual blending algorithm of [27] would need optimality criteria actually opposite to those of [10]; one such is the phrase "a water of beginnings and ashes" at the end of the first stanza of *Walking Around*, which blends elements having very different types, and thus required (what in programming language theory is called) "type casting." Similarly, the *Duino Elegies* by Rainer Maria Rilke contains phrases that blend elements from very distant domains, such as "the cheap winter hats of fate" in the fifth elegy.

Another problem with [10] is that it does not provide principles for selecting domains and their conceptual subspaces for blending during the process of understanding (or generating) some piece of language or experience. A different approach from that of [10] calls for assembling a network of spaces such that the network and its blend are as simple as possible according to some (socially determined but) precise criteria. This approach has been successfully applied to music in work at UCSD with David Borgo [4, 5], using a hierarchical minimum complexity information theory [14] to measure the simplicity of the network of selected spaces and its blend.

The systems described in [27] use the formalism of algebraic semiotics [19], which combines ideas from algebraic specification and social semiotics, and which generalizes the conceptual spaces of cognitive linguistics to *semiotic spaces*, by adding types, functions and axioms. A key idea in this approach is to formalize the intuitive notion of a representation of one system of signs by another system of signs as a morphism of semiotic spaces; combined with the methodological principles of viewing user interfaces as such representations, and viewing design as involving blending, this gives a powerful set of tools for the design and analysis of user interfaces. The algebraic semiotics notion of blending generalizes colimits by including ordering relations on the semiotic morphisms between two given spaces, based on their quality as representations [19]; see the appendix of this paper for formal details, which include $\frac{3}{2}$-colimits and $\frac{3}{2}$-categories. This approach supports a great variety of optimality principles, by choos-

ing different ordering relations on morphisms, and it has been applied to a number of problems in user interface design [19, 20, 24, 26].

3 Tools for Information Integration

A great deal of sophisticated research has been done on integration in the database community, including semantic theory, tool building, and experiments with real data and users (see [32] for a survey). The local-as-view approach is a co-relation in a category of schemas and views, where 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 translated into local queries, the results of which are integrated to form an answer to the original query. Since this requires views from the global to the local schemas, much effort has been devoted to schema mapping tools, which help to construct these views (see [43] for a survey). One such tool, called SCIA, has been designed and built at UCSD [21, 39, 40, 47]. Because fully automatic schema mapping generation is infeasible, this tool tries to minimize total user effort by identifying **critical points**, where a small user input can yield the largest reduction of future matching effort. A major result is that this approach can significantly reduce total user effort. Other tools only try to find the easiest 1-to-1 matches, leaving the most difficult matches for the user to do by hand.

An unusual but important feature of SCIA is its ability to handle semantic functions and conditions, where "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 systems do not treat functions and conditions at all, or else leave them for a different tool, e.g., view generation.

4 Conclusions

This Dagstuhl seminar, and papers in the related literature, show 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. It will surely be interesting to see what happens over the next few years, as further connections emerge, perhaps as unexpected as those with institutions and blending that are suggested in this paper.

It is not coincidental that this paper is highly interdisciplinary. Experience shows that purely technical solutions, based on what it is feasible and/or 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, and for technology to reach real users, the results of social analyses and mathematical foundations should be embedded in tools. Such an approach can ignite cycles of improvements to a technology, its tools, its scientific basis, and the understanding of its social context; if done sensitively, it can also improve the life quality of the individuals involved.

A Some Mathematical Details

This appendix gives details of some mathematics mentioned in the paper, following Appendix B of [19], which was developed with Grigore Roşu and Răzvan Diaconescu; the proofs in [19], due to Roşu, are omitted here. Although self-contained, this material may be difficult for readers not already familiar with category theory. The main intuition is that categories capture mathematical structures; for example, sets, groups, vector spaces, and automata, along with their structure preserving morphisms, each form a category, with the morphisms playing the central role. For example, theories and their morphisms over any institution form a category [25], as do the sign systems and semiotic morphisms of [19, 20]; theory morphisms are translations of conceptual systems, and semiotic morphisms are representations of signs by other signs.

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*.* □

The following reviews the notions of pushout, cone and colimit, relates them to information integration and blending, and then generalizes this setup to $\frac{3}{2}$-categories, which capture more of the phenomenology of blending and other complex information integration problems. The intuition for colimits is that they integrate components, identifying as little as possible, with nothing left over, and with nothing essentially new added [17]. Thus colimits are a kind of optimal blend; although this kind of optimality is not appropriate for conceptual blending, it is nevertheless a good place to begin our journey.

Definition 2. *Given a category $\mathbb{C}$, a* **V** *in $\mathbb{C}$ is a pair $a_i : G \to I_i$ ($i = 1, 2$) of morphisms, and a* **cone** *with* **apex** *B over a V a_1, a_2 is a pair $b_i : I_i \to B$ ($i = 1, 2$) of morphisms; then a_1, a_2 and b_1, b_2 together are said to form a* **diamond** *(or* **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 $c_i : I_i \to C$ over a_1, a_2, there is a unique arrow $u : B \to C$ 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 $f : D_i \to D_j$ labels an edge $e : i \to j$, 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 $b_i : D_i \to B$ for each node i, called an* **injection***, from each object of D to B, and is* **commutative** *iff for each $f : D_i \to D_j$ in D, we have[5] $b_i = f; b_j$. A* **colimit** *of D is a commutative cone $b_i : D_i \to B$ over D such that if $c_i : D_i \to C$ is any other commutative cone over D, then there is a unique $u : B \to C$ such that[6] $b_i; u = c_i$ for all nodes i of D.* □

Pushouts are the special case of colimits where the diagram is a V, as in Figure 1. (There might seem to be a discrepancy in the definitions, in that 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 $a_1; b_1 = a_2; b_2$.) The following discussion temporarily uses the term **blend** for commutative diamonds, in order to test its adequacy as an hypothesis.

There is a short proof that any two colimits of a diagram D are isomorphic. Let the cones be $b_i : D_i \to B$ and $b'_i : D_i \to B'$. Then there are unique arrows $u : B \to B'$ and $v : B' \to B$ satisfying the appropriate

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

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

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 $v;u = 1_{B'}$.

But conceptual blends are not unique up to isomorphism; for example, the conceptual spaces for "house" and "boat" have 48 blends [27], two of the more familiar being "houseboat" and "boathouse" [19]. It is shown in [19] that the "houseboat" blend is actually a colimit, but "boathouse" is not. Another problem with defining blends to be commutative cones is that the cones of many blends *do not* fully commute; for example, the "boathouse" blend has only one triangle commutative. The following is one solution this problem: Let the **auxiliary morphisms** of a diagram D be a subset the triangles over which need not commute. Removing these morphisms from D yields another diagram D' having the same nodes as D, such that commutative cones over D' are cones over D that commute except possibly over auxiliary morphisms; moreover, a colimit of D' is an optimal such cone over D. This suggests defining a blend to be a commutative cone over a diagram with its auxiliary morphisms removed; however, this notion still has the uniqueness property.

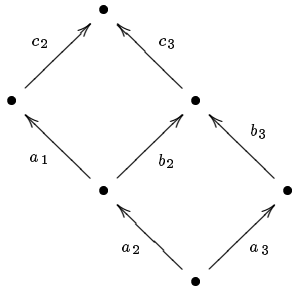

Fig. 2. Composing Pushouts

An advantage of formalization is that it makes possible stating and proving general laws, such as that "the composition of blends is a blends." The meaning of this compositionality result may be clarified by reference to Figure 2, in which we assume 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 elegantly as

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

using the convention that $a_1 \diamond (a_2 \diamond a_3)$ indicates blending a_1 over the left injection of $(a_2 \diamond a_3)$ (the top left edge of its diamond).

A pushout composition result (proved e.g. in [30, 37]) 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 can also be written elegantly, 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[7]. A related 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 the cone and colimit notions of integration.

We now broaden our framework, motivated by the category of sign systems with semiotic morphisms [19], which has additional structure over that of a category: it is an *ordered category*, because algebraic semiotics provides orderings on its morphisms by their quality as representations. This provides a richer framework for considering information integration in general, and conceptual blending in particular, in which optimality principles, analoguous to those of [10] but precisely formalized as the ordering on morphisms, play a central role. Moreover, categorical compositionality results about pushouts and colimits extend to $\frac{3}{2}$-categories.

Definition 3. *A* **$\frac{3}{2}$-category**[8] *is a category* $\mathbb{C}$ *such that each set* $\mathbb{C}(A, B)$ *is partially ordered, composition preserves the orderings, and identities are maximal.* □

In this context, a somewhat different notion of pushout is appropriate, and for this notion, the uniqueness property is (fortunately!) lost:

[7] A special case of this (or more precisely, of its dual) is that the most general unifier of any finite set of terms can be computed by taking most general unifiers pairwise, in any order.

[8] 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.

Definition 4. *Given $a_i\colon G \to I_i$ $(i = 1, 2)$ and a V in a $\frac{3}{2}$-category $\mathbb{C}$, then a cone $b_i\colon I_i \to B$ $(i = 1, 2)$ 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* $\frac{3}{2}$**-pushout** *iff for every consistent cone $c_i\colon I_i \to C$ over a_1, a_2, the subset*

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

of $\mathbb{C}(B, C)$ has a maximum element. □

Notice that if each set $\mathbb{C}(A, B)$ is finite (as is usual in computer science applications) and linearly ordered (which is not unusual), then any non-empty subset has a maximum element. Figure 2 again may be helpful for the following:

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 typical compositionality result:

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

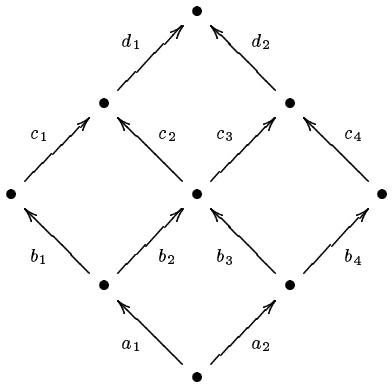

Fig. 3. Composing Four $\frac{3}{2}$-Pushouts

Passing from V's to arbitrary diagrams of morphisms generalizes $\frac{3}{2}$-pushouts to $\frac{3}{2}$-colimits, and provides a natural way to integrate complexly interconnected information. The notion of consistent diamond extends to arbitrary diagrams as follows:

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

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 $\{b_i\}_{i \in |D|}$ is a $\frac{3}{2}$-***colimit** *of D iff it is a cone and for any D-consistent family $\{c_i\}_{i \in |D|}$, the set $\{h \mid b_i; h \leq c_i,$ for each $i \in |D|\}$ has a maximum element.* □

The following is another generalization from ordinary 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 3 (but without a_1, a_2).* □

Extending our pushout notation $\bowtie$ to $\frac{3}{2}$-categories, the above result can be elegantly written as

$$(b_1 \bowtie b_2) \bowtie (b_3 \bowtie b_4) = Colim(W) \ ,$$

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. It is also worth noting that the notion of auxiliary morphism carries over to the framework of $\frac{3}{2}$-categories without any change.

In the theory of semiotic spaces [19], morphisms that preserve more structure are considered better; in particular, they should be as defined as possible, should preserve as many axioms as possible, and should identify as few elements as possible. These three conditions define an 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, preserves all axioms that f does, and is as inclusive as f (see [19] for details). Applying this ordering to the house and boat example, the best blends are $\frac{3}{2}$-pushouts, and those that fail to preserve as much structure of their input spaces as they could are not $\frac{3}{2}$-pushouts. We can now connect with the optimality principles of [10] through the observation that an ordering on morphisms induces an ordering on potential blends b by measuring how close they are to being a $\frac{3}{2}$-colimit under the given ordering on morphisms, e.g., by comparing how many maximal morphisms are in each $\mathbb{C}one_V(b, c)$, where $\mathbb{C}one_V$ is the category of all cones in $\mathbb{C}$ over V, a fixed V in $\mathbb{C}$; for a fixed b, each $\mathbb{C}one_V(b, c)$ has exactly one maximal element iff b is a $\frac{3}{2}$-colimit of V.

References

1. Suad Alagic and Philip Bernstein. A model theory for generic model management. In Giorgio Ghelli and Gösta Grahne, editors, *Proc. Database Programming Languages 2001*, pages 228–246. Springer, 2002.

2. Jon Barwise and Jerry Seligman. *Information Flow: Logic of Distributed Systems.* Cambridge, 1997. Tracts in Theoretical Computer Science, vol. 44.
3. 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.
4. David Borgo and Joseph Goguen. Sync or swarm: Group dynamics in musical free improvisation. In Richard Parncutt, Annekatrin Kessler, and Frank Zimmer, editors, *Proceedings, Conference on Interdisciplinary Musicology*, pages 52–53. Dept. Musicology, University of Graz, 2004. Held in Graz, Austria, 15–18 April 2004.
5. David Borgo and Joseph Goguen. Rivers of consciousness: The nonlinear dynamics of free jazz, 2005. To appear, *Proceedings, Annual Meeting of International Association of Jazz Educators.*
6. Geoffrey Bowker and Susan Leigh Star. *Sorting Things Out.* MIT, 1999.
7. Andrea Cali, Diego Calvanese, Biuseppe De Giacomo, and Maurizio Lenzerini. Data integration under integrity constraints. *Information Systems*, 29(2):147–163, 2004.
8. Răzvan Diaconescu. Grothendieck institutions. *Applied Categorical Structures*, 10:383–402, 2002.
9. Gilles Fauconnier. *Mental Spaces: Aspects of Meaning Construction in Natural Language.* Bradford: MIT, 1985.
10. Gilles Fauconnier and Mark Turner. *The Way We Think.* Basic, 2002.
11. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer, 1997.
12. Joseph Goguen. Mathematical representation of hierarchically organized systems. In E. Attinger, editor, *Global Systems Dynamics*, pages 112–128. S. Karger, 1971.
13. 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.
14. Joseph Goguen. Complexity of hierarchically organized systems and the structure of musical experiences. *International Journal of General Systems*, 3(4):237–251, 1977.
15. 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.
16. 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.
17. Joseph Goguen. A categorical manifesto. *Mathematical Structures in Computer Science*, 1(1):49–67, March 1991.
18. 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.
19. 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.

20. Joseph Goguen. Semiotic morphisms, representations, and blending for interface design. In *Proceedings, AMAST Workshop on Algebraic Methods in Language Processing*, pages 1–15. AMAST Press, 2003. Conference held in Verona, Italy, 25–27 August, 2003.
21. 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.
22. Joseph Goguen. Information integration in instutions, 2004. To appear in memorial volume for Jon Barwise, edited by Lawrence Moss, `www.cs.ucsd.edu/~goguen/papers/IF.html`.
23. Joseph Goguen. Ontology, society, and ontotheology. In Achille Varzi and Laure Vieu, editors, *Formal Ontology in Information Systems*, pages 95–103. IOS Press, 2004. Proceedings of FOIS'04, Torino, Italy.
24. Joseph Goguen. Steps towards a design theory for virtual worlds. In Maria-Isabel Sánchez-Segura, editor, *Developing Future Interactive Systems*, pages 116–152. Idea Publishing Group, 2004.
25. 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.
26. Joseph Goguen and Fox Harrell. Information visualization and semiotic morphisms. In Grant Malcolm, editor, *Multidisciplinary Studies of Visual Representations and Interpretations*, pages 93–106. Elsevier, 2004. Proceedings of a workshop held in Liverpool, UK.
27. Joseph Goguen and Fox Harrell. Style as a choice of blending principles. In Shlomo Argamon, Shlomo Dubnov, and Julie Jupp, editors, *Style and Meaning in Language, Art Music and Design*, pages 49–56. AAAI Press, 2004.
28. Joseph Goguen and Grigore Roşu. Institution morphisms. *Formal Aspects of Computing*, 13:274–307, 2002.
29. 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.
30. Robert Goldblatt. *Topoi, the Categorial Analysis of Logic*. North-Holland, 1979.
31. 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.
32. Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *Knowledge Engineering Review*, 18(1):1–31, 2003.
33. 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.
34. Robert Kent. Formal or axiomatic semantics in the IFF, 2003. Available at `suo.ieee.org/IFF/work-in-progress/`.
35. 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.
36. Bruno Latour and Steve Woolgar. *Laboratory Life*. Sage, 1979.
37. Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, 1971.

38. Till Mossakowski. Heterogeneous specification and the heterogeneous tool set, 2004. Habilitation thesis, University of Bremen, to appear.
39. Young-Kwang Nam, Joseph Goguen, and Guilian Wang. A metadata integration assistant generator for heterogeneous distributed databases. 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 1332–1344. Springer, 2002.
40. 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.
41. Charles Saunders Peirce. *Collected Papers*. Harvard, 1965. In 6 volumes; see especially Volume 2: Elements of Logic.
42. Benjamin C. Pierce. *Basic Category Theory for Computer Scientists*. MIT, 1991.
43. Erhard Rahm and Philip Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
44. John Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks/Coles, 2000.
45. Alfred Tarski. The semantic conception of truth. *Philos. Phenomenological Research*, 4:13–47, 1944.
46. Jeffrey Ullman. *Elements of ML Programming*. Prentice Hall, 1998.
47. 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.