Information Integration in Institutions

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

1 Introduction

The world wide web has made an unprecedented volume of information available, and despite the fact that this volume continues to grow, it is also becoming easier to find what you want. On the other hand, the problem of finding several collections of relevant data and integrating them[1] has hardly been tackled outside the well-behaved realm of conventional database systems. Ontologies have been suggested as an approach to this problem, but there are many difficulties, including the creation and maintenance of ontologies, existence of multiple ontologies for single domains, the use of multiple languages to express ontologies, and the multiple logics on which those languages are based.

The motivation for this paper is to unify and generalize a number of approaches to information, in order to provide a rigorous foundation for integrating heterogeneous information, not tied to any specific representational or logical formalism, by using category theory to achieve independence from any particular choice of representation, and using institutions to achieve independence from any particular choice of logic. Co-relations are proposed as a very general formalization of information integration, that includes numerous special cases, from databases, cognitive linguistics, information flow, and other areas (see Section 4).

Barwise and Seligman have developed a theory of information flow and channels [3], to solve the problem of how information about one thing (or several different things) can convey information about something else, i.e., how information can "flow" through "channels" to activate new information. Kent [33, 35, 34] has proposed to use this as a model for distributed ontologies. This paper generalizes information flow to any logic by using institutions, and following the lead of Kent [33], also combines it with the formal conceptual analysis of Ganter and Wille [13] and the lattice of theories approach of Sowa [44]. In addition, it draws on the categorical general systems theory of Goguen [14, 15, 18] as a further generalization of information flow.

[1] Such as, find all computer science faculty in the European Union who teach a graduate course on logic and have published in a major database conference.

Hereafter, we let "IF" abbreviate "information flow" and also use it as a prefix designating concepts in the IF approach; similarly, we let "FCA" abbreviate "formal concept analysis" as in [13] (originating with Wille circa 1982), we let "IFF" abbreviate "Information Flow Framework" as in the IEEE Standard Upper Ontology project of Kent [35, 34], we let "LOT" abbreviate the lattice of theories approach of Sowa, and we let "CGST" abbreviate "categorical general systems theory" of Goguen. Applications to ontologies, as in [4, 32, 35], were a major inspiration for much of this paper, and are discussed in Section 5 and Appendix B.

The greater generality of institutions over classifications, local logics, concept lattices, etc. not only allows doing information integration over arbitrary logics, but also allows an elegant treatment of many interesting examples in which part of a situation is fixed while another part is allowed to vary, e.g., non-logical symbols can vary, while the logical symbols and rules of deduction are fixed. Intuitively, institutions parameterize the basic judgement of satisfaction, and all of the theory that rests upon it, by this kind of variation. A philosophical motivation for parameterizing satisfaction follows arguments of Peirce against dyadic theories of meaning, and in favor of meaning being triadic, where the third ingredient between signs (such as formulae) and meanings is an "*interpretant*" that supplies a context in which interpretation is done; in this sense, institutions formalize a key insight of Peirce's semiotics; see Appendix B and Appendix C for further discussion. Unfortunately, there is currently no easy introduction to institutions, but a brief intuitive introduction is given in Section 2, including institution morphisms and comorphisms, with motivation for their use in ontology integration; some aspects of Sowa's LOT approach are also generalized here.

The translation and generalization of IF and FCA begins in Section 3; the Galois connection associated to every institution plays a key role here. Section 3.1 introduces IF classifications, constraints and theories, showing that these are special cases of institutional concepts, and then generalizing them. Section 3.2 generalizes a related basic result of FCA to arbitrary institutions. Section 4 discusses channels, distributed systems, and the Interconnection Theorem, a CGST result that is both new for and relevant to IF. Also, co-relations are suggested here to formalize information integration, noting that IF channels, database local-as-view systems, and cognitive linguistic blends are all special cases. Local logics are discussed in Section 4.2; data, schema, and ontology integration in Section 5; and some conclusions are given in Section 6. Appendix A explains the Grothendieck "flattening" construction, a powerful category

theoretic tool that we use to handle heterogeneity, including ontologies and schemas over different signatures and/or logics. Appendix B gives several ways to view database systems with ontologies as institutions, including a new approach to mereology[2]. An important theme of this paper is that institutions generalize the dyadic classifications of IF and FCA to a triadic relation that parameterizes judgements (such as that a type classifies a token, or a model satisfies a sentence) by a context in which the judgement is made, in a way that has many advantages, including not just greater generality, but also an elegant mathematical theory, and a powerful framework of the modularization and maintenance of complex theories (e.g., ontologies). Appendix C illustrates the power of the triadic institutional satisfaction relation, using an institution that formalizes databases in a novel way inspired by the triadic semiosis of C.S. Peirce; here the relation says whether an answer satisfies a query in the context of a given database.

The use of categories and institutions in this paper goes well beyond the pervasive but implicit use of category theoretic ideas in IF [3]. In particular, we show not only that many important IF structures are the objects of categories with the relevant morphisms, but we also show that they are the models or theories of interesting institutions, and moreover that some of the most important IF structures actually *are* very special kinds of institutions. In addition, we prove many new results using known methods and results about categories, institutions, or CGST; although some details are omitted from this note, they can be found in the references and/or reconstructed by readers as exercises to test their understanding of the material. Some mathematicians have expressed aversion to category theory, presumably due to their professional orientation towards difficult proofs for specific results in traditional areas, whereas category theory is oriented towards easy proofs of very general results; but the latter is precisely what is needed for the goals of this paper. There are no difficult new theorems here; instead, this paper integrates previously disconnected areas by applying old theorems in new ways, with the intention of helping to solve some important practical problems.

Familiarity is assumed with the basics of category theory, including category, opposite category, functor, natural transformation, limit, colimit, and the categories **Set** of sets and **Cat** of categories; there are many places to learn this material, such as [43, 17, 36]. We use ";" for the composition of morphisms.

[2] Sometimes also called "partology," this refers to the formal study of the "`part-of`" of "constituency" relation.

This paper is dedicated to the memory of Jon Barwise, with whom I had the pleasure of working for several years at the Center for the Study of Language and Information at Stanford University. Jon was an exceptional thinker in both the depth and breadth of his vision, and he is much missed.

Thanks to Robert Kent and Marco Schorlemmer for valuable comments on earlier drafts, and to the participants in Dagstuhl Seminar 04391, Semantic Interoperability and Integration, for their enthusiasm and comments; of course, any remaining bugs are my own fault. This work is 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.

2 Institutions and their Morphisms

The basic reference for institutions is [24], 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 (with first order structures as models, denoted **FOL**), many sorted equational logic (with abstract algebras as models, denoted **EQL**), Horn clause logic (denoted **HCL**), many variants of higher order and of modal logic, and much more; it seems that essentially any logical system has a corresponding institution.

Institutions abstract and generalize Tarski's "semantic definition of truth" [46], the most significant ingredient of which is a relation of *satisfaction* between models and sentences, denoted $\models$. For example, Tarski defined the semantics of conjunction with a formula like

$$M \models (P \text{ and } Q) \text{ iff } (M \models P) \;\&\; (M \models Q)$$

where (just for the moment) "and" is syntactic conjunction, "&" is semantic conjunction, M is a model, and P, Q are formulae.

Many applications of logic, for example in computer science, require the vocabulary from which sentences are constructed (such as predicate and function symbols) to vary from one situation to another in such a way that truth is invariant under translations among such vocabularies. This requires formalizing the notions of vocabulary and of translations among vocabularies, as well as the effects of such translations on sentences and on models, each of which must be taken as parameterized by the vocabularies that they use.

Institutions accomplish this formalization by generalizing from "vocabularies" (the word "lexicon" has also been used) to **signatures**, which are abstract objects, and from "translations among vocabularies" to abstract mappings between objects, called **signature morphisms**; then the parameterization of sentences by signatures is given by as assignment of a set $Sen(\Sigma)$ of sentences to each signature Σ, and a translation $Sen(f)$ from $Sen(\Sigma)$ to $Sen(\Sigma')$ for each signature morphism $f: \Sigma \to \Sigma'$, while the parameterization of models by signatures is given by an assignment of a class $Mod(\Sigma)$ of models for each signature Σ, and a translation $Mod(\Sigma') \to Mod(\Sigma)$ for each $f: \Sigma \to \Sigma'$.

Definition 1: An **institution** consists of an abstract category **Sign**, the objects of which are signatures, a functor Sen: **Sign** $\to$ **Set**, and a functor Mod: **Sign**^{op} $\to$ **Set** (technically, we might uses classes instead of sets here). **Satisfaction** is then a parameterized relation $\models_\Sigma$ between $Mod(\Sigma)$ and $Sen(\Sigma)$, such that the following **Satisfaction Condition** holds, for any signature morphism $f: \Sigma \to \Sigma'$, any Σ'-model M', and any Σ-sentence e

$$M' \models_{\Sigma'} f(e) \text{ iff } f(M') \models_\Sigma e$$

where $f(e)$ abbreviates $Sen(f)(e)$ and $f(M')$ abbreviates $Mod(f) < M' >$. This condition expresses the invariance of truth under change of notation. □

Some of the earliest (circa 1980), and most useful, results in the theory of institutions concern a duality between theories and model classes. A **Σ-theory** is a set of Σ-sentences, and a **Σ-model class** is a class of Σ-models. Every Σ-theory T determines a Σ-model class T^*, consisting of all Σ-models that satisfy all its sentences, and every Σ-model class V determines an Σ-theory V^*, consisting of all Σ-sentences satisfied by all the models in V. These two operations define the kind of duality between model classes and theories known as a **Galois connection** [5] (for those who know the concept, it is also an adjoint functor situation). Many simple results are known to hold in such situations, some of which we now discuss.

A theory is **closed** iff it equals its closure; intuitively, a closed theory already contains all the consequences of its sentences; the **closure** of a theory T is the theory T^{**}, while the **closure** of a model class V is V^{**}. The closure of a closed theory (or model class) equals itself (in fact, $T = T^{**}$ iff T is closed), and $(T_1 \cup T_2)^* = T_1^* \cap T_2^*$; many more such results for arbitrary institutions are given in Propositions 3 and 4, and Lemmas 6 and 7, of [24]. The following result is less trivial, but still well known for

Galois connections; Section 3.1 shows how it helps to greatly generalize some results in [13].

Proposition 1: For any institution $\mathbf{I}$ and signature Σ, the closed Σ-theories, and the closed Σ-model classes, are complete lattices under inclusion. Moreover, there is a dual isomorphism between these complete lattices.

Proof Sketch: Given a family T_i for $i \in I$ of closed theories, $(\bigcup_i T_i)^{**}$ is clearly the least closed that contains each T_i. Existence of arbitrary least upper bounds implies a complete lattice, this case ordered by inclusion. A similar argument works for the closed model classes. The two isomorphism maps are the two * maps, and they are dual because $T \leq T'$ iff $T^* \leq T'^*$. Functorial adjointness gives continuity of the isomorphisms. □

The collections of all theories and of all model classes over a given signature are also lattices, with $T \leq T'$ iff $T^* \subseteq T'^*$. This is exactly the "lattice of theories" of Sowa [44], when the institution is first order logic (**FOL**, but using an elegant Peircean syntax), though LOT and also IFF, use the opposite ordering relation, which seems more intuitive for many purposes; in addition, [44] also proposes a number of set theoretic operations for navigating the lattice of theories. However, I believe it is actually more useful to consider all the theories over all the signatures of an institution as a single category, as in the following from [24]:

Definition 2: Let $Th(\mathbf{I})$ have as its objects pairs (Σ, T) where Σ is a signature of $\mathbf{I}$ and T is an Σ-theory of $\mathbf{I}$, and as its morphisms semantic consequence preserving signature morphisms, i.e., $f\colon (\Sigma, T) \to (\Sigma', T')$ is $f\colon \Sigma \to \Sigma'$ such that if t is in T then $T' \models f(t)$, in the sense that $M' \models T'$ implies $M' \models f(t)$ for all M', or equivalently, $t \in T$ implies $f(t) \in T^{**}$. □

Such "heteromorphisms" between ontologies with different signatures are needed if ontologies are to be used for integrating data from domains with different ontologies (see [31] for a use survey of work on ontology mappings). Category theory also supports a wide variety of useful operations on theories (see the discussion following Theorem 1 below).

Definition 2 is a special case of the Grothendieck construction described in Appendix A, which the reader may consult for a definition of the composition of heteromorphisms. Some results similar to the above are proved for the special case of many sorted first order logic in [34], using the fibration variant of the Grothendieck construction, though without benefit of the unifying and generalizing notions and results of theory institution theory; [24] also proves the following (or actually, the stronger

result, that the forgetful functor from theories to signatures creates colimits):

Theorem 1: If the signature category of an institution **I** has colimits, then so does $Th(\mathbf{I})$. □

These colimits can be used integrate theories over different signatures, and of course such theories could in particular be ontologies. As discussed in [24, 16], 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 FCA and LOT; moreover, it provided the basis for the powerful module system of the ML programming language, with its so called signatures, structures and functors [38], though ML does not provide all the functionality defined in [16] and implemented in the OBJ language family, which include CafeOBJ [9], Maude [6], BOBJ [26], OBJ3 [30], and to some extent CASL [41], under the name "parameterized programming"; these ideas also influenced the module systems of C++, Ada, and LOTOS.

If we restrict to a fixed signature Σ and also restrict to theory morphisms where f is the identity on Σ, then there is at most one morphism from any Σ-theory to any other, and the resulting category becomes a quasi-ordered set of Σ-theories, in which theories T, T' are **equivalent** iff $T \leq T'$ and $T' \leq T$. If we identify equivalent theories, we again get the complete lattice of closed Σ-theories. The set of (not necessarily closed) Σ-theories is also a complete lattice under inclusion, but this is not very interesting. Entirely similar constructions apply to model classes. However, as shown by ML and the other languages mentioned above, it is more useful to work with $Th(\mathbf{I})$.

While the material above provides a good foundation[3] for integrating theories (such as ontologies) over a fixed logic, it is not adequate for integrating theories over different logics, which is often needed in practice. For this, we need to be able to translate between logics, i.e., institutions. As discussed in [28], there are actually many different kinds of logic translation. The following (from [24]) is perhaps the most basic of these:

Definition 3: An **institution morphism** from an institution **I** to another institution $\mathbf{I}'$ consists of a functor F: **Sign** $\rightarrow$ **Sign**$'$ and two nat-

[3] Although ignoring many practical issues, some of which are discussed, for example, in [1, 42].

ural transformations $a\colon\ F; Sen' \to Sen$ and $b\colon\ Mod \to F; Mod'$, such that, for any Σ-model M and $F(\Sigma)$-sentence e'

$$M \models_\Sigma a_\Sigma(e') \text{ iff } b_\Sigma(M) \models'_{F(\Sigma)} e' \ .$$

Let **INS** denote the category with institutions as objects, and with these morphisms. □

Intuitively, institution morphisms are truth preserving translations from one logical system to another. More technically, for any signature Σ, letting $\Sigma' = F(\Sigma)$, then a_Σ maps Σ'-sentences to Σ-sentences, and b_Σ maps Σ-models to Σ'-models, in a way that is consistent with respect to the satisfaction relation. One class of examples consists of institution inclusion morphisms, for which F, a_Σ and b_Σ are each an inclusion. For example, there is an inclusion morphism from **HCL** to **FOL**. A more general notion of **subinstitution** allows these functions to be injective. An example is the translation from (many sorted) **EQL** into (unsorted) **FOL**, in which equality is treated as a special relation. Of course, there are many other examples of subinstitutions, some of which are given below; and there are also many institution morphisms that are not subinstitution morphisms, some of which we will also see below. Among the variants of the institution morphism notion, perhaps the most natural is the following:

Definition 4: An **institution comorphism** from an institution **I** to another $\mathbf{I}'$ consists of a functor $F\colon\ \mathbf{Sign} \to \mathbf{Sign}'$, a natural transformation $a\colon\ Sen \to F; Sen'$, and a natural transformation $b\colon\ F; Mod' \to Mod$, such that, for any $F(\Sigma)$-model M' and Σ-sentence e

$$b_\Sigma(M') \models_\Sigma e \text{ iff } M' \models'_{F(\Sigma)} a_\Sigma(e) \ .$$

Let co**INS** denote the category with institutions as objects, and with institution comorphisms as morphisms. □

It can be helpful to think of an institution comorphism as an institution morphism between the source and target co-institutions, where the **co-institution** of an institution replaces its signature category by its opposite, and then swaps its sentence and model functors. This transformation gives two functors that define an isomorphism between the categories of institutions with morphisms and with comorphisms, thus formalizing this important duality.

There are now hundreds of papers that study and/or apply institutions. The original application was to provide an expressive module system for knowledge representation in the Clear language. The approach was later extended to provide module systems for formal specification languages, and later still for programming languages, as mentioned above;

see [29] for a general discussion. A recent related contribution is Mossakowski's Heterogeneous Tool Set [39] extension of CASL, which supports verification over a number of different logics, based on ideas described in Appendix A.

Institution morphisms formalize logic translations, and provide a very general notion of what integration can mean in many different contexts; they also provide many useful results about logic translations, such as that they preserve the modular structure of an ontology under certain mild assumptions (see [24]). Many results from logic have been generalized to institutions, including the Craig interpolation, Robinson consistency, ultrafilter, and Herbrand universe theorems; the institutional versions apply to all logics where the results were already known, and they yield new results for many other logics, e.g., [10].

3 Information Flow in Institutions

"Information" is central to contemporary computer science, cognitive science, and philosophy, but there is no widely accepted definition for this concept, and no definition that is adequate to all its intended applications. Both IF and FCA capture certain key aspects of information in their formal mathematical theories, though neither provides an explicit definition of information as such[4]. This section shows that many basic IF and FCA concepts are very special cases of institution concepts, based on those institutions that have the simplest possible signatures:

Definition 5: A **1-institution** is an institution where **Sign** $= \mathbf{1}$, the category having just one object $\bullet$, and just one morphism. $\square$

Then the morphism $\bullet \to \bullet$ in **Sign** is necessarily the identity on $\bullet$, so that the transformations $Sen(\mathbf{1}_\bullet)$ and $Mod(\mathbf{1}_\bullet)$ are both necessarily identities, and so can be ignored.

3.1 IF Classifications, Infomorphisms, Constraints and Theories

Barwise and Seligman [3] give an account of what it means for one sign (they use the word "token") to "carry information" about another, or in a different metaphor, of how "information flows." The rigorous mathematical theory in [3] is based on the intuition that information flow is only

[4] My definition of information is given in [19]; it is broadly consistent with the Peircean pragmatism espoused by [44], though considerably more social.

possible in a stable distributed system, consisting of "channels" through which "information flows." This subsection gives a brief self-contained exposition of some basics of this theory.

A classification in the sense of [3] consists of a set K of **tokens**, a set P of **types**, and a relation $\models$ between K and P, that tells whether a given token has a given type; we may call this an **IF-classification** for clarity. Classifications have often appeared in various literatures, e.g., [5] calls them "polarities," and [13] calls them "contexts;" they can be considered a primitive kind of ontology. According to [3], tokens carry information, and types represent the information that is carried. Given classifications $(K, P, \models)$ and $(K', P', \models')$, an **infomorphism** $(K, P, \models) \to (K', P', \models')$ consists of functions $f^\vee\colon K' \to K$ and $f^\wedge\colon P \to P'$ such that $f^\vee(k) \models t$ iff $k \models' f^\wedge(t)$, for all $k \in K'$ and $t \in P$. Infomorphisms are the constituents of channels, which express the regular structure of distributed systems in IF theory. It is now easy to check the following, noting that the condition on $f^\vee$ and $f^\wedge$ is just the institutional Satisfaction Condition:

Proposition 2: A classification is a **1**-institution where $Mod(\bullet)$ is the set of tokens and $Sen(\bullet)$ is the set of types. Moreover, infomorphisms are comorphisms of these institutions. Let **IFC** denote the resulting category of IF-classifications, which is the subcategory of co**INS** having the **1**-institutions as its objects. □

Alternatively, if we consider tokens as sentences and types as models (i.e., take the co-institution), then we can also view **IFC** as a subcategory of **INS**, though this may seem rather counter-intuitive. We now give another formulation of IF-classifications along with some additional important IF concepts:

Definition 6: Given a set P of type symbols, a P-**classification** C is a set K of tokens plus a unary relation $C(p)$ on K for each $p \in P$, i.e., a function $P \to 2^K$ to subsets of K. By convention we write p for $C(p)$, and $k \models p$ instead of $p(k)$. A P-**sequent** is a pair of subsets of P, written in the form $\Gamma \vdash \Delta$. A P-**theory** is a set E of P-sequents, called P-**constraints** in this context; an **IF-constraint** is a P-constraint for some P, an **IF-theory** is a P-theory for some P, and a P-classification C **satisfies** a P-constraint $\Gamma \vdash \Delta$ iff for all tokens k of C, $k \models q$ for every $q \in \Gamma$ implies $k \models p$ for some $p \in \Delta$. □

Thus, P-classifications are IF-classifications with type set P. From here on, we feel free to omit prefixes IF- and P- if they are clear from the context. A pretty little institution lurks in the above definitions: its signature category is **Set**, the objects of which are considered sets P of

predicate symbols; its P-models are P-classifications; its P-sentences are P-sequents; and its satisfaction is as above. We let the reader define the translations of models and sentences under signature translations, and check the satisfaction condition. Let us denote this institution by **IFS**; it is easy to see that this "institution of **1**-institutions" is a subinstitution of **FOL**. Moreover, by adding infomorphisms, the model classes become categories (of classifications), yielding an institution in the original sense of [24], which extends Definition 1 by allowing Mod: $\mathbf{Sign}^{op} \to \mathbf{Cat}$.

Definition 7: The institution **IFS** has **Set** as its category of signatures, with $Mod(P)$ consisting of functions $P \to 2^K$ from P to subsets of some set K, with $Sen(P) = P^* \times P^*$ (the set of all pairs of subsets from P), written as sequents, and with $\models_P$ the usual satisfaction relation. □

The above suggests generalizing IF to allow tokens that are terms built using arbitrary constructors[5], and to allow first order sentences with non-unary predicates, instead of just sequents with unary predicates. The result is actually the familiar institution of first order logic with terms. But we can go further, and consider information flow theory over an *arbitrary* institution; also we can obtain a deeper understanding of the relation between classifications and theories by viewing it as a special case of the Galois connection between model classes and theories that holds in any institution. For this, we first review more material from [3]:

Definition 8: The **theory of** an IF-classification C, denoted $Th(C)$, has as its constraints all sequents that satisfy C. An IF-theory is **regular** iff it satisfies the identity, weakening, and global cut axioms given in [3], page 119. The **classification of** a theory $T = (P, E)$, denoted $Cla(T)$, is $(P, K, \models)$, where K is the set of all partitions (Γ, Δ) of P that are not in E, and where $(\Gamma, \Delta) \models p$ iff $p \in \Gamma$. □

The following is proved in [3]:

Theorem 2: $Th(Cla(T)) = T$ for any regular IF-theory T. □

The above result is extended to a categorical equivalence in [33]. Comparing Theorem 2 with definitions in Section 2 gives a nice little result, which appears to be new:

Proposition 3: $Th(C) = C^*$ and $Cla(T) = T^*$; moreover, an IF-theory is closed iff it is regular. □

[5] Although [3] says "... any suggestion that tokens must have something like 'syntax' is quite unwelcome," we claim that such structuring is precisely what is needed for some applications, including mereology for ontologies, as discussed in Appendix B.

Although this formulation is particular to **IFS**, the Galois connection and its many consequences hold for every institution. In particular, we can now see that Theorem 2 is a special case of the general institutional result that $T^{**} = T$ iff T is closed; what then becomes interesting is the particular form of the constructions, Th and Cla, for the Galois duality in this special case.

3.2 Formal Concept Analysis

A **formal context** in FCA [13] is the same as an IF classification, except that instances are called "objects" and types are called "attributes"; thus formal contexts are **1**-institutions. The main notion of FCA is that of a **formal concept** for a formal context, which is a pair (T, V) where $T = V^*$ is a theory and $V = T^*$ is a model class (for which FCA uses the terms "intent" and "extent," respectively). In many concrete cases, these may be considered the natural "concepts" for that situation; for example, sentences might be formed over an ontology, and models might "populate" that ontology with certain instances. Then the formal concepts should be exactly those that are useful in understanding the meanings of the attributes of objects; this can be useful in practice, e.g., in constructing ontologies. Unfortunately, real data is often too dirty for such an idealization to work well in practice, which suggests that some new ideas are needed to deal effectively with the uncertainties of the real world.

It follows that both T and V are closed in formal concepts, which are thus the pairs of closed elements that correspond under the dual isomorphism of Proposition 1. Note that it is not necessary to keep both T and V, since they determine each other; note also that these are concrete lattices of sets of elements from P and K, respectively. The lattice of formal concepts associated with a classification is called its **formal concept lattice**. Conversely, given a lattice of closed theories over given sets K and P, we can define $k \models p$ iff $k \in \{p\}^*$. Intuitively, the concept lattice of a classification identifies tokens and types that cannot be distinguished by how the classification uses them. It is now easy to verify the following version of the Basic Theorem of Formal Concept Lattices of [13]:

Proposition 4: There is a bijective correspondence between classifications and their formal concept lattices. □

It is natural to define formal concept morphisms to be complete lattice homomorphisms, and then show that the above bijection is an equivalence of categories, as in [33]; many other categorical extensions of FCA can also be found in [33]. One appeal of FCA is that nice pictures can be

drawn for the concept lattices of (sufficiently small) classifications; these pictures are called **Hasse diagrams** in lattice theory.

Proposition 4 extends easily to arbitrary institutions, where $\models$ is parameterized by signatures, calling the lattice of closed theories associated with a given signature of a given institution its formal concept lattice. However, the Hasse diagram is not finite for the logical systems usually considered in the literature on institutions. Intuitively, the closed theories (or model classes) of an institution at a given signature identify those theories (and model classes) that are indistinguishable with respect to satisfaction, and thus (as in the special case of classifications) extract the meaningful "concepts" for that institution.

4 Channel Theory and Information Integration

A very general and often useful notion is that of a **relation** in a category $\mathbf{C}$, which consists of three objects, say A, B, R, and two morphisms, say $p_1 \colon R \to A$ and $p_2 \colon R \to B$. If $\mathbf{C}$ is **Set**, then a relation R consists of pairs (a, b) with $a \in A$ and $b \in B$, and p_1, p_2 are **projection** maps. Most of the usual theory of relations between sets can be done in this very general setting, with some modest assumptions on $\mathbf{C}$, such as that $\mathbf{C}$ has pullbacks. There is also a dual notion of **co-relation**, consisting of three objects A, B, C and two **injection** morphisms, $f_1 \colon A \to C$ and $f_2 \colon B \to C$. Relations (and co-relations) generalize to families $p_i \colon R \to A_i$ (or $f_i \colon A_i \to C$). We now give the categorical form of another basic concept in [3]:

Definition 9: A **channel** is a co-relation $f_i \colon A_i \to C$ for $i \in I$, in the category **IFC**; C is called the **core** of the channel. □

Looking at only the tokens, a channel yields a relation that "connects" each token c in its core to the tokens $f_i^\vee(c)$ in its components A_i. A **cover** of a diagram is defined in [3] to be a channel over that diagram such that every triangle formed by an infomorphism in the diagram and the two injections, from the source and target of that infomorphism, commutes. This is just the categorical notion of a **co-cone**, for the special case of a diagram in **IFC**; similarly, the "minimal covers" of [3] are **colimits** in **IFC**, although [3] use the term "limit" for this concept, perhaps because the tokens are more concrete than types (more technically, we might attribute it to the duality between morphisms and comorphisms). Category theorists often use the term **apex** instead of "core" for both relations and co-relations.

It is highly encouraging that co-relations are the essence of a number of interesting special cases, including not only channel theory, but also local-as-view data integration in database theory, blending in cognitive linguistics [11], module interconnection [29] and concurrent process interconnection [18, 12] in software engineering, and (following algebraic semiotics [21, 20]), the composition of subsigns to form interfaces in user interface design. It is therefore reasonable to suggest that co-relations, co-cones, and co-limits are the obvious way to integrate the constituent objects, as already suggested in [14, 17]. Although the dual global-as-view approach corresponds to relations in a category, it can also be formulated using co-relations; it is less general, but more efficient for query answering. In many cases, it is natural to consider a subtheory of shared material, as is already standard for blending in cognitive linguistics. For co-relations with a shared subtheory (denoted G in the figure below), *pushouts* give an optimal blend (and dually, pullbacks are optimal in the relational case).

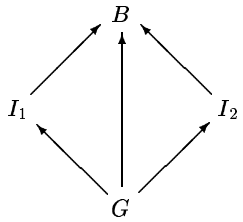

Fig. 1. Information Integration over a Shared Subobject

But it is too optimistic to expect this kind of optimality in many practical situations, where there are many possible solutions; instead, one should consider pragmatic optimality criteria like those used in cognitive linguistics. However, it does not seem, as suggested in [11], that any single set of optimality criteria is appropriate for all situations, but rather that different criteria are needed for different situations. For example, [25] examined metaphors in the poetry of Pablo Neruda, and found that some especially creative blends there used criteria opposite to the common sense criteria given in chapter 16 of [11]; one such is the phrase "water of beginnings and ashes" in the first stanza of the poem "Walking Around." In [20], it is suggested that categories with an ordering relation on the morphisms between any two objects, called **$\frac{3}{2}$-categories**, and be used as a basis for **$\frac{3}{2}$-colimits**, which seem to have the flexibility necessary for user interface design and cognitive linguistics.

It is well known that categorical relations with pullbacks yield the usual calculus of set theoretic relations, e.g., composition of relations and its associativity, as well as converse, union, intersection, etc., with their

usual properties. The join of relations in database theory is a special case of this. The dual calculus of co-relations is less well known but similarly rich, and moreover, everything generalizes to n-ary relations and co-relations, with and without shared material. It is also 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 [11]. Computer scientists have also used the terms "alignment" and "articulation" for this process, but its relation to more general notions of integration (under names like "fusion", "merging" and "reconciliation") have until now remained somewhat mysterious; although I do not wish to suggest that the connections described here suffice to resolve all the practical problems that arise, I do believe that they provide the basis of a general theory of information integration.

The important result below follows from general methods for proving properties of categories of **close variant institutions** given in [28] (to be more precise, **IFC** is $(1_{\mathbf{Set}}\downarrow / 1_{\mathbf{Set}}\uparrow)$, a comma category of so called "twisted relations," which Proposition E.4 of [28] implies is complete and cocomplete; note also that classifications are twisted relations). **IFC** is a **1**-institution because the category of functors from **1** to it is isomorphic to it; see [28, 24] for details of the twisted relation formulation of institutions.

Theorem 3: Every (small) diagram in **IFC** has a colimit. □

This means that every IF-distributed system has a channel that best describes its flow of information. For the special case of **IFC**, the "core" of the colimit has as its types the colimit of the corresponding diagram of type sets, and as its tokens the limit of the corresponding diagram of token sets, just as one might expect (e.g., from principles in [17]). The **sum** of a set of classifications is the special case where the system diagram is discrete, i.e., contains no infomorphisms; here the token set is the product of the token sets of the components. Theorem 3 can be considered a very general version of the Dretske Xerox Principle discussed in [3], which asserts the "transitivity of information flow"[6].

[6] This can be seen from the construction of limits (for tokens) in concrete categories such as **Set**, or more abstractly, from their construction using equalizers of products (e.g., see [15, 36]).

In addition, Theorem 3 makes available the powerful structuring and integration mechanisms of parameterized programming [16]. Proof methods in [28] also give the following result, which is not in [3]:

Theorem 4: Every (small) diagram in **IFC** has a limit. □

The construction is dual to that for colimits: the type set of the limit object is the limit of the corresponding diagram of type sets, and its token set is the colimit of the corresponding diagram of token sets. The special case where the diagram is discrete is the product of the components; its token set is the disjoint union of the token sets of its components.

Barwise and Seligman [3] cite Chu spaces (for which see the appendix of [2]) for their proof of Theorem 3; this is consistent with our approach, because Chu spaces with $Z = \{0, 1\}$ are institutions with $\mathbf{Sign} = \mathbf{1}$; moreover, general Chu spaces are subsumed (still using the trivial signature category) by the **generalized institutions** of [23], in which satisfaction takes values in an arbitrary category $\mathbf{V}$ (the end of Section 2 of [45] gives a better exposition of this than that in [23]); so called Chu transformations are of course comorphisms of such institutions.

4.1 Categorical General Systems Theory

In 1971, the categorical general system theory (CGST) of [14] proposed many of the same ideas as [3], including representing distributed systems as diagrams in a category, using relations to describe connections among components, and using *limits* to compute behavior; this can be considered a very general form of the Dretske Xerox Principle (but note the dualization). CGST amounts to doing information flow and logical architecture in an arbitrary category. An important idea from [14] not in [3] is that the colimit of a diagram of systems computes the system that results from the interconnection. A main result, called the **Interconnection Theorem** [15], says that the behavior of an interconnection of systems is the limit of the diagram of behaviors of the component systems (but Theorem 3.11 of [18] provides a better exposition than that in [15]):

Theorem 5: For D a diagram of diagrams over a complete category $\mathbf{C}$, and for Lim the limit computing functor on the category of diagrams,

$$Lim(colim D) = lim(D; Lim) \ .$$

□

Specialized to **IFC** and suitably dualized, this result is a useful addition to IF theory: it tells how to compute the core of an interconnection of distributed systems from the cores of its component systems. The CGST for-

malism of [14] is applied to various distributed systems in [18], which also includes treatments of object oriented concepts like inheritance, concurrency concepts like deadlock, and security concepts like non-interference. The use of sheaves in this work to capture the time varying behavior of components might also be useful for IF, FCA, and LOT, since their approaches to dynamics seem rather weak, and in particular seem unable to handle continuous time; it is also worth remarking that an appropriate logic for sheaves is given by the so called internal logic of a topos, as noted in [18].

4.2 Local Logics

We begin with an extension of classifications from [3]:

Definition 10: A **local classification** is a classification $(P, K, \models)$ together with a subset N of K, called its **normal tokens**. □

Local classifications are not quite **1**-institutions, but they are close variant **1**-institutions, $(1_{\mathbf{Set}}{\downarrow} \,/\, p_1{\uparrow})$ where $p_1 \colon \mathbf{SubSet} \to \mathbf{Set}$ is the subset extracting functor, where **SubSet** is the category of subset inclusions with commutative squares as morphisms. Let us denote the category of local classifications by **IFCL**. The new result below again follows from Proposition E.4 of [28]; as before, it enables the structuring and integration mechanisms of parameterized programming [16].

Theorem 6: IFCL has both limits and colimits. □

We now construct a category the objects of which are the **local logics** of [3], consisting of pairs (T, M) of a regular P-theory T and a local classification M that satisfies T when restricted to N; the construction will also yield the so called **logic infomorphisms** of [3]. Given an institution $\mathbf{I}$, we first define the functor $Modth \colon Th(\mathbf{I})^{op} \to \mathbf{Cat}$ to map each theory to the category of all models that satisfy that theory, where $Th(\mathbf{I})$ is the category of all theories of $\mathbf{I}$, as in Definition 2. Next, we apply the Grothendieck construction Appendix A to that functor, the result of which is a category $\mathbf{Gr}(Modth)$ the objects of which are pairs (T, M) where T is a theory of $\mathbf{I}$ and M is a model of $\mathbf{I}$ that satisfies T, with morphisms $(T, M) \to (T', M')$ pairs (h, f) where $h \colon T \to T'$ is a theory morphism and $f \colon M \to h(M')$ is a T-model morphism. Finally, if we let $\mathbf{I} = \mathbf{IFS}$, then we get exactly the category of local logics in the sense of [3], which we denote by **IFL**. The Galois connection, concept lattice bijection, Interconnection Theorem, and many other results now follow easily, including the following, which also seems new:

Theorem 7: IFL has both limits and colimits. □

The proof applies general completeness results for the Grothendieck construction, e.g., in [28]. And again, it makes available the structuring and integration mechanisms of parameterized programming [16].

Restricting to the closed theories and model classes of **IFL** for a fixed signature, we can construct concept lattices for local classifications exactly as previously for ordinary classifications; we conjecture that the resulting category of local concept lattices is equivalent to the category of local classifications.

5 Data, Schema, and Ontology Integration

Information integration over distributed databases, such as the world wide web, is a significant potential application for ideas discussed in this paper, but it is also very challenging, and accomplishing it will require integrating both schemas and ontologies that are associated with data. Categorical principles (e.g., as in [17]) say that since schemas and (so called "unpopulated") ontologies are theories, they should be combined with colimits. However, the fact that, in general, different signatures are involved means that the category of Σ-theories for a fixed signature Σ is inadequate, and so we extend it using the Grothendieck construction of in Appendix A. (Of course, there are also many practical problems to be addressed.)

We first note that the construction of $Th(\mathbf{I})$ in Definition 2 extends to a functor $Th\colon \mathbf{INS} \to \mathbf{Cat}$, sending each institution to its category of theories. Next, applying the Grothendieck construction to this functor yields a category **GTh** with objects $(\mathbf{I}, \Sigma, E)$ where $\mathbf{I}$ is an institution, and (Σ, E) is a theory of $\mathbf{I}$, and with morphisms $(\mathbf{I}, \Sigma, E) \to (\mathbf{I}', \Sigma', E')$ consisting of an institution morphism $(F, a, b)\colon \mathbf{I} \to \mathbf{I}'$ plus a signature morphism $f\colon \Sigma' \to F(\Sigma)$ such that $E \subseteq a_S(f(E'))^{**}$. However, **GTh** is an enormous beast, nearly all of which is useless for any particular application. This motivates considering a (small) category $\mathbf{O}$ of institutions, on which as before we define a functor $Th\colon \mathbf{O} \to \mathbf{Cat}$, and then flatten it, just as for **GTh**, but restricting to institutions and morphisms in $\mathbf{O}$; let the result be denoted **GTh(O)**. (**GTh** and **GTh(O)** also result from applying the remarkable Grothendieck institution construction of [7] to the theory functor.) It is not difficult to show that limits and colimits exist in these categories under reasonable conditions. See [22] for details.

6 Conclusions

Rising above details, some beautiful patterns emerge that are not visible in more specialized theories such as IF, FCA, and LOT. First, each of the logics for classification, **IFC**, **IFS**, **IFCL**, and **IFL**, gives rise to an institution. Secondly, institutions occur at two different levels: classifications are institutions, which in turn form the model categories of other institutions with non-trivial sentences. Although one can view the satisfaction of these sentences as another classification, the resulting structure (via Tarski's semantic definition of truth) is not particularly helpful. By contrast, our third pattern is that the institutional formulations give rise to many interesting results in a uniform way, using established methods; moreover, many of these results are new for IF, FCA and LOT, including the Galois connection between models and theories, several (co-)completeness results, and the Interconnection Theorem. It is therefore natural to make further generalizations having the same mathematical structure. For example, consider institutions with non-trivial term constructors and constants (so that terms serve as descriptors for complex tokens), and with non-unary predicates, including a binary equality predicate that is interpreted as identity in models; these form an institution, say **IFE**, for which the same results follow in the same uniform way; we can even add subsets of normal tokens. Horn clause logic provides an alternative notion of sentence for which inference is easier; moreover, this logic features prominently in many currently popular ontology languages.

In addition to all this, channels are co-relations (or co-cones), distributed systems are diagrams, and behaviors are colimits, over any of these institutions, and the same applies for quite a number of other formalisms for information integration, including database theory, blending theory, and areas of user interface design and software engineering. It seems evident that ideas from IF, FCA and LOT could therefore be applied in these other areas, once the proper conceptual infrastructure is established.

There are many other directions that also seem well worth exploring, including parts of [3], [13] and [44] that we have not yet tried to institutionalize. The categorical formulations of [33] are very relevant to this effort, and inspired the attention paid to [13] in the present paper. It would seem worthwhile to extend the Galois connection between theories and model classes to an indexed family of adjunctions between theories with deduction and categories of models, noting that standard methods of categorical logic allow viewing deductions as morphisms between sentences,

so that both theories and models with their morphisms form categories (see [40] for some details). It would also be interesting to further explore the use of sheafs as space- and time- varying tokens and types in classification systems, as in [18]. Finally, it seems important to develop theory and tools that can cope with the multiple forms of uncertainty that are inherent in the real world (e.g., see [19] for a discussion of some sources of such uncertainty). Further development of $\frac{3}{2}$-categories seems promising in this respect, and perhaps some ideas from fuzzy logic could also be useful.

Even if one accepts that much of the mathematical content of IF, FCA and LOT can be done more generally and more elegantly using institutions, this does not detract from the practical and philosophical achievements of these works, nor does it detract from their expository ease, part of which is due to avoiding category theory. In particular, it is indeed fascinating to consider physical systems from the viewpoint of information flow, what events in one component tell about events in another. Similarly, even if one accepts that many ideas about systems are done more generally in the CGST of [14, 15], this does not detract from the particular application of [3] to information flow; in particular, [14, 15] do not treat inference.

But my admiration for these works does not mean that I always agree with their philosophical views. In particular, I do not accept the implicit philosophical realism of [3] and [13], which seems to me to take too little account of the social and cognitive aspects of information, and of the many practical difficulties that can arise from this. My own views on the nature of information [19] are consistent with the Peircian pragmatism advocated by Sowa [44], but I think are more explicitly social, but not essentially more social, than Pierce. However, philosophical debates often have little effect on applicability, and I have no doubt that ideas IF, FCA, LOT and IFF can have significant applications in information technology, e.g., see [32] and [35] for some interesting applications to ontologies, e.g., for the web. However, I expect, for reasons that include the enormous variety of data formats in use on the web, and the range of different logics used for ontologies, that institutional formulations may have some significant advantages over less general approaches. I also hope that some day, computer scientists (and philosophers) will be sufficiently familiar with concepts like category, colimit, and even institution, that these can be freely used without alienating large parts of the potential audience, and that authors will no longer feel it necessary to camouflage their use of such concepts with idiosyncratic "user friendly" terminology.

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. Michael Barr. $\star$-autonomous categories and linear logic. *Mathematical Structures in Computer Science*, 1:159–178, 1991.
3. Jon Barwise and Jerry Seligman. *Information Flow: Logic of Distributed Systems.* Cambridge, 1997. Tracts in Theoretical Computer Science, vol. 44.
4. 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.
5. Garrett Birkhoff. *Lattice Theory.* American Mathematical Society, 1948. Colloquium Publications, Volume XXV (revised edition, 1960).
6. Manuel Clavel, Francisco Durán, Steven Eker, Patrick Lincoln, Narciso Martí-Oliet, José Meseguer, and José F. Quesada. Maude: Specification and programming in rewriting logic. *Theoretical Computer Science*, 2001.
7. Răzvan Diaconescu. Grothendieck institutions. *Applied Categorical Structures*, 10:383–402, 2002.
8. Răzvan Diaconescu. Interpolation in Grothendieck institutions. *Theoretical Computer Science*, 311:439–461, 2004.
9. Răzvan Diaconescu and Kokichi Futatsugi. *CafeOBJ Report: The Language, Proof Techniques, and Methodologies for Object-Oriented Algebraic Specification.* World Scientific, 1998. AMAST Series in Computing, Volume 6.
10. Răzvan Diaconescu. An institution-independent proof of Craig interpolation theorem. *Studia Logica*, 77:59–79, 2002.
11. Gilles Fauconnier and Mark Turner. *The Way We Think.* Basic, 2002.
12. José Luiz Fiadeiro. *Categories for Software Engineering.* Springer, 2004.
13. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations.* Springer, 1997.
14. Joseph Goguen. Mathematical representation of hierarchically organized systems. In E. Attinger, editor, *Global Systems Dynamics*, pages 112–128. S. Karger, 1971.
15. 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.
16. 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.
17. Joseph Goguen. A categorical manifesto. *Mathematical Structures in Computer Science*, 1(1):49–67, March 1991.
18. Joseph Goguen. Sheaf semantics for concurrent interacting objects. *Mathematical Structures in Computer Science*, 11:159–191, 1992.
19. 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.
20. 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.

21. 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.
22. 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.
23. Joseph Goguen and Rod Burstall. A study in the foundations of programming methodology: Specifications, institutions, charters and parchments. In David Pitt, Samson Abramsky, Axel Poigné, and David Rydeheard, editors, *Proceedings, Conference on Category Theory and Computer Programming*, pages 313–333. Springer, 1986. Lecture Notes in Computer Science, Volume 240.
24. 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.
25. 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.
26. Joseph Goguen and Kai Lin. Behavioral verification of distributed concurrent systems with BOBJ. In Hans-Dieter Ehrich and T.H. Tse, editors, *Proceedings, Conference on Quality Software*, pages 216–235. IEEE Press, 2003.
27. Joseph Goguen and José Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105(2):217–273, 1992. Drafts exist from as early as 1985.
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. Joseph Goguen, Timothy Winkler, José Meseguer, Kokichi Futatsugi, and Jean-Pierre Jouannaud. Introducing OBJ. In Joseph Goguen and Grant Malcolm, editors, *Software Engineering with OBJ: Algebraic Specification in Action*, pages 3–167. Kluwer, 2000.
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. Saunders Mac Lane. *Categories for the Working Mathematician*. Springer, 1998. Second Edition.

37. Brian Mayoh. Galleries and institutions. Technical Report DAIMI PB-191, Aarhus University, 1985.
38. Robin Milner, Mads Tofte, Robert Harper, and David MacQueen. *The Definition of Standard ML (Revised)*. MIT, 1997.
39. Till Mossakowski. Heterogeneous specification and the heterogeneous tool set, 2004. Habilitation thesis, University of Bremen, to appear.
40. Till Mossakowski, Joseph Goguen, Razvan Diaconescu, and Andrzej Tarlecki. What is a logic?, 2004. To appear in Proceedings, First World Conference on Universal Logic.
41. Peter Mosses, editor. *CASL Reference Manual*. Springer, 2004. Lecture Notes in Computer Science, Volume 2960.
42. 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.
43. Benjamin C. Pierce. *Basic Category Theory for Computer Scientists*. MIT, 1991.
44. John Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks/Coles, 2000.
45. Andrzej Tarlecki, Rod Burstall, and Joseph Goguen. Some fundamental algebraic tools for the semantics of computation, part 3: Indexed categories. *Theoretical Computer Science*, 91:239–264, 1991.
46. Alfred Tarski. The semantic conception of truth. *Philos. Phenomenological Research*, 4:13–47, 1944.

A Grothendieck Constructions

Situations in which one kind of structure is indexed by another are rather common in mathematics, computer science, and their applications, and are the essence of many information integration problems. Alexander Grothendieck developed a very general way of dealing with this kind of structural heterogeneity, as part of his brilliant reformulation of algebraic geometry into the language of category theory, in order to solve a number of then outstanding problems. The word "structure" in the first sentence of this paragraph is formalized as the mathematical concept of category, in which the structure-preserving morphisms play a central role, and the indexing is then given by a functor $F\colon I^{op} \to \mathbf{Cat}$ (the original formulation of Grothendieck assumes a weaker coherence than that given by functoriality, but this complex extra generality is not needed for our applications). The Grothendieck construction "flattens" this indexed family of categories into a single category. This is important because it supports combining objects from different categories in the single flattened category using colimits, in which morphisms describe sharing and translations among objects. Typical objects of practical interest are ontologies and schemas (see [22] for some detail), both of which are special cases of the Grothendieck construction on theories given in Definition 2.

Now some details: The Grothendieck category $\mathbf{Gr(F)}$ of an indexed family $F : \mathbf{I}^{op} \to \mathbf{Cat}$ has as its objects pairs (i, A) where i is an object in $\mathbf{I}$ and A is an object in $F(i)$, and has as its morphisms $(i, A) \to (i', A')$ pairs (f, h) where $f : i \to i'$ in $\mathbf{I}$ and $h : A \to F(f)(A')$ in $F(i)$; such morphisms have also been called "cryptomorphisms" and "heteromorphisms" in the computer science literature. Given also $(f', h') : (i', A') \to (i'', A'')$, define the composition $(f, h); (f', h') : (i, A) \to (i'', A'')$ to be $(f; f', h; F(f)(h'))$. It is easy to check that this gives a category.

Several useful results about colimits and limits in Grothendieck categories are given in [45], but a better exposition appears in Section 2.1 of [28]. It is worth mentioning an alternative approach to the same phenomena based on fibered categories, though in our opinion, its greater technical complexity does not yield corresponding benefit.

The Grothendieck institution construction of [7] applies the same idea to indexed families of institutions. The result is not just a single category, but a single institution, the signature category of which is the Grothendieck flattening of the indexed family of signature categories, and similarly for the sentences and the models. Moreover, logical properties of the individual institutions tend to lift to the whole under suitable assumptions, e.g., Craig interpolation [8].

B Ontologies and Database Institutions

An ontology is a different kind of theory for a database than its schema, because its purpose is to give vocabulary and sentences for describing the elements that can appear in the database. In practice, the vocabulary is often restricted to constants that denote individuals, unary predicates that classify individuals, and binary predicates that relate individuals, while sentences are often restricted to Horn clauses. However, such restrictions are by no means necessary, and are imposed mainly to ensure efficient decidability. Here, we work with an arbitrary theory O and an arbitrary model Ω of O, over an arbitrary institution $\mathbf{I}$ such that its model category is concrete; O is the **ontology**, and Ω is the **population** of individuals and values that can be used in fill slots in the database. We give examples later, but note in passing that two good choices for $\mathbf{I}$ are Horn clause logic and order sorted algebra, and that Ω is often a free extension of O.

A **schema** is a theory T over $\mathbf{I}$ that contains O. Let $T(\Omega)$ denote T with Ω adjoined as additional constants, and let Σ be the signature of Ω. Then a **database state** over the schema T is a model B of T such

that $B|_S = \Omega$, and a **query** over T is an existential sentence over the signature of T (noting that if $\mathbf{I}$ does not have existential quantifiers, there is a standard trick for defining new sentences that are $\mathbf{I}$-sentences bound by existential quantifiers). Defining databases morphisms is optional.

Now we have everything needed for our database institution. Its signatures are schemas over $\mathbf{I}$ as defined above, with theory morphisms as signature morphisms; the models of a schema T are the database states over T; the sentences for a schema T are queries over T; and satisfaction of a query q by a model B over T holds iff there is an valid instantiation of q in B, i.e., an assignment θ of values in B to the existential variables in q such that $\theta(f) \models B$ in $\mathbf{I}$. It is not difficult to check that this setup forms an institution, denoted $\mathbf{DB}(\mathbf{I}, O, \Omega)$. It is natural to have logic for queries, e.g., a conjunction of queries should satisfy $f \models B$ and $f' \models B$ iff $f \& f' \models B$, in which case conjunction is associative, commutative, and idempotent; similar things can be done for the other connectives, including existential quantification in case it is missing from $\mathbf{I}$, although this requires further assumptions on $\mathbf{I}$. Thus, database systems in a sense are logics; see [40] for discussion of the close connection between institutions and logics. Some other ways to institutionalize databases are discussed below.

Although the most common sentences for ontologies and schemas are Horn clauses, it is also interesting to consider the alternative of order sorted algebra for $\mathbf{I}$, since it handles the hierarchical classification of elements in a quite different way from Horn clause logic, where class subsumption may be encoded as implications between unary predicates that represent classes, and may also be encoded using a binary `is-a` relation. In order sorted algebra, signatures have a set of types, called sorts, on which a partial subsort order is defined, and models are required to respect that ordering, in that if $s \leq s'$ then $\Omega_s \subseteq \Omega_{s'}$ thus classification and class subsumption are in signatures rather than theories, an approach which has been found more convenient for many purposes. In addition, order sorted algebra allows overloaded operation symbols, and handles their subtle interactions with subsorting.

It is even more interesting to consider the `has-a` or `part-of` relation, because it is so often problematic in real ontologies. Formal mereology is a branch of philosophy that tries to axiomatize this relation; it is a controversial area, with little agreement on what the axioms should be, or even what intuitions should be axiomatized, which is symptomatic of the difficulties with this relation. In order sorted algebra, as in algebra generally, elements are represented as terms, which are built from con-

stants and other terms, using certain operations called **constructors**. For example, if c is a binary constructor and if $a = c(b1, b2)$, then we could say that $b1$ and $b2$ are "parts of" a, but the formula is actually a more precise and informative statement that avoids the problems of the `part-of` relation, by explicitly saying how the parts are put together to form the whole. (Of course, not all real world constituent hierarchies can be formalized using constructors, but it seems those that typically arise in ontologies can be.) Moreover, order sorted algebra also nicely axiomatizes the somewhat subtle relations between subsumption and constituency. To me, this approach is more useful than those based on mereology; see [27] for the formal details of order sorted algebra.

For a simple example, let O be an order sorted theory having 6 sorts, for naturals, reals, booleans, and strings of characters, abbreviated respectively N, R, B, C. Let `cs-course`, `math-course`, `req-course`, `opt-course` be unary predicates in O, and let

`req-course`(X) implies `cs-course`(X)
`math-course`(X) implies `opt-course`(X)

be constraints in O. Finally, let O classify items with the axioms

`math-course`(329)
`req-course`(130)
`opt-course`(171)

Next, define a relational schema T having three relations, $R1, R2, R3$ as Boolean-valued functions, with respectively 3, 2, 3 fields, with arities (i.e., argument sorts) (N, C, R), (N, C), (N, N, R), respectively. Intuitively, the fields of $R1$ might be for student Id, student name, and GPA; the fields of $R2$ for course Id and course name; and the fields of $R3$ for course Id, student Id, and grade. Then a database state over this schema can be represented as three sets of tuples, where each tuple is a ground instance of the corresponding relation; e.g., $R2$ might be true of the 2-tuples (329, Category Theory), (171, User Interface Design), (130, Programming Language Concepts), and no others.

It is common for databases to include constraints in their schemas. These may be key constraints, which assert that distinct tuples in a relation must have distinct values in certain of their fields, and data integrity constraints, which assert that the values in certain fields must have certain properties. A convenient way to present such assertions is to use names assigned to the fields of relations as "selectors," or projection operations from tuples to data values; more formally, these are inverses to the corresponding constructor operations. For example, $R2$ might have

selectors `course-no`: $R2 \to \Omega_N$ and `course-title`: $R2 \to \Omega_C$. Then the following is a key constraint

$$\texttt{course-no}(r_1) = \texttt{course-no}(r_1') \text{ implies } r_1 = r_1'$$

where r_1, r_1' range over the rows (i.e., tuples) of $R1$; however, this should be considered an abbreviation for the conditional equational form

$$Y = Y' \text{ if } R1(X,Y) = \mathit{true} \;\&\; R1(X',Y') = \mathit{true} \;\&\; X = X'$$

where X, X' range over course Ids, and Y, Y' range over course names. The following is a data integrity constraint

$$0 < \texttt{course-no}(R1(X,Y)) < 400$$

which is again an abbreviation, for

$$0 < X = \mathit{true} \;\&\; X < 400 = \mathit{true} \text{ if } R1(X,Y) = \mathit{true}.$$

Key constraints support a weak form of unique identity for entities, a topic that has been much discussed in philosophical logic under various names.

A **query** (i.e., sentence) for this institution is an expression of the form $(\exists X_1, ..., X_n)\varphi$, where φ is a Boolean combination of Boolean-valued terms over T and Ω, and such a query is **satisfied** by a database B iff there are values $x_1, ..., x_n$ for $X_1, ..., X_n$ such that $\varphi(X_1 \leftarrow x_1, ..., X_n \leftarrow x_n) = \mathit{true}$. For example,

$$(\exists X_1, X_2, X_3, X_4)\ R1(X_1, X_2, X_3) \;\&\; R3(329, X_1, X_4) \;\&\; X_3 > 3.6$$

asks whether there is a student taking course 329 who has a GPA greater than 3.6 (strictly speaking, we should give sorts for $X_1, ..., X_n$).

Schema heteromorphisms can be defined similarly to the Grothendieck construction, as pairs $(\Phi, f)\colon (S,T) \to (S',T')$ where $\Phi\colon \mathbf{Sign} \to \mathbf{Sign}'$ is a functor and $f\colon \Phi;(f) \to S'$ is a schema morphism. These compose in the natural way, and we get a category of schemas with heteromorphisms over an arbitrary institution $\mathbf{I}$. With a little more work, this can be made the category of signatures of an institution of heterogeneous databases as models with queries as sentences, over $\mathbf{I}$.

The above treatment of queries is less than satisfactory because we only know whether or not a query has an answer, but not what the answer is. A more sophisticated approach indexes queries by the type of their answer, i.e., by the string of sorts of their existential quantifiers, and then allows the "truth values" of satisfaction to be the answers to the queries; such an application of institutions to databases was first suggested by Brian Mayoh [37], and requires the generalized V-institutions of [23]. However generalized institutions can be avoided by defining an institution $\mathbf{DB}'(I, O, \Omega)$ like $\mathbf{DB}(\mathbf{I}, O, \Omega)$ except that sentences have the form $q \vdash$

θ, and are satisfied by B iff $\theta(q)$ satisfies B in $\mathbf{DB}(\mathbf{I}, O, \Omega)$ with the substitution θ; here θ serves as the answer. It is natural to define $\theta \& \theta' \vdash B$ iff $\theta \vdash B$ and $\theta' \vdash B$, to extend this to finite sets of substitutions, perhaps even using set notation; similar things can be done for other logical connectives. A more sophisticated institutionalization of database systems is given in Appendix C.

On the other hand, a more naive approach is to ignore queries, as in [1], which applies institutions to databases by translating "institution" to "schema translation framework", "model" to "database", "sentence" to "constraint", and "theory" to "schema"; queries are not considered, nor are heteromorphisms, although integration is nicely conceptualized as a co-relation over a shared part arising from an initial signature, with colimit as the ideal result if it is defined.

C A Novel Database Institution

This section illustrates the power of the triadicity of satisfaction in institutions, by giving a novel database formalization; however, for many practical purposes, the institutionalizations in Appendix B may be better. We modify the example of Appendix B by letting $\mathbf{I}$ be many sorted Horn clause logic, so that relation symbols really denote relations, represented in models by appropriate sets of tuples, rather than being functions. Then we let morphisms of database states be 3-tuples of inclusions of the three sets of tuples. However, the twist is that we take these to be the objects and morphisms of our category of signatures; it is confusing to call these objects "signatures," but this is precisely why this example can be considered "novel". Because we will let sentences be queries, and models be answers, the term "context" might be a better than "signature," although it is less suggestive of the original examples from logic.

Now fix a set $X_1, X_2, ..., X_n$ of variable symbols, each having a fixed sort from among those of Ω, and let the queries in $Sen(\Sigma)$ be formulae of the form $P_1 \& ... \& P_q$, where each P_j is of the form $R(t_1, ..., t_m)$ for some relation R in the schema, where each t_k is either some X_i or else some constant from Ω, and where $(t_1, ..., t_m)$ has the arity required by R. Since $Sen(\Sigma)$ does not actually depend on Σ, we may as well write just Sen for this set of formulae, and given $i : \Sigma \to \Sigma'$, we can let $Sen(i)$ be the identity on Sen. These queries should be thought of as implicitly existentially quantified by the variables $X_1, ..., X_n$. We could certainly consider more complex queries, but these are sufficient for our illustrative purpose.

The models in $Mod(\Sigma)$ are possible answers to queries over Σ, by which we mean sets of tuples $(d_1, ..., d_n)$ where each d_i is an element of Ω having the same sort as X_i, where each tuple has an associated "witness" $(r_1, ..., r_p)$, where each r_j is a tuple in the set of instances belonging to Σ. Given an inclusion $i : \Sigma \to \Sigma'$, let $Mod(i) : Mod(\Sigma') \to Mod(\Sigma)$ be the map that sends M' to the set of pairs $((d_1, ..., d_n), (r_1, ..., r_p))$ in M' such that each r_j is among the instances belonging to Σ. It is not hard to check that this is a functor, which restricts M' to Σ. Finally, given a query Q and an answer A over a state Σ, let $A \models_\Sigma Q$ hold iff for each $((d_1, ..., d_n), (r_1, ..., r_p)) \in A$, substituting d_i for X_i in P_j yields the tuple r_j, for each j. We can now check the satisfaction condition.

Of course, it is also desirable to allow different sets of variables, but this can be accomplished with a Grothendieck construction; and another Grothendieck construction gives a database institution that allows queries over arbitrary relational schemas as well as over arbitrary finite variable sets. These flattenings are like those done previously for signatures, but they provide suggestive illustrations of the potential of institutions for situating judgements (such as satisfaction) within the contexts where they are made, in a way that is quite different from that of the usual examples of logics as institutions.

A somewhat more sophisticated version packages the schema with the database to form signatures, so that the Sen functor will vary with the schema component. Another extension is to enrich the query language with predicates and functions form O. Ideally this would give users a familiar and convenient vocabulary that abstracts away from details of database structure, with which users may not be familiar. The query language could even be GUI-based, e.g., as an extension of our SCIA tool [22], which provides a GUI to help users construct schema mappings, and which is currently being extended to ontology mappings.