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

Abstract. This paper unifies and/or generalizes several approaches to information, including the information flow of Barwise and Seligman, the formal conceptual analysis of Wille, the lattice of theories of Sowa, the categorical general systems theory of Goguen, and the cognitive semantic theories of Fauconnier, Turner, Gärdenfors, and others. Its rigorous approach uses category theory to achieve independence from any particular choice of representation, and institutions to achieve independence from any particular choice of logic. Corelations and colimits provide a general formalization of information integration, and Grothendieck constructions extend this to several kinds of heterogeneity. Applications include modular programming, Curry-Howard isomorphism, database semantics, ontology alignment, cognitive semantics, and more.

1 Introduction

The world wide web has made unprecedented volumes of information available, and despite the fact that this volume continues to grow, improvements in browser technology are gradually making it easier to find what you want. On the other hand, the problem of finding several collections of relevant data and integrating them[1] has received relatively little attention, except within the well-behaved realm of conventional database systems. Ontologies have been suggested as an approach to such problems, but there are many difficulties, including the connection of ontologies to data, the creation and maintenance of ontologies, existence of multiple ontologies for single domains, the use of multiple languages to express ontologies, and of multiple logics on which those languages are based, as well as some deep philosophical problems [32].

This paper unifies and generalizes several approaches to information[2], providing a rigorous foundation for integrating heterogeneous information that uses category theory to achieve independence from any particular

[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.

[2] Although we often use the terms "information" and "concept," we do not claim or believe that the various formal entities that we study are identical with, or are even fully adequate representations of, information or concepts as actually experienced and used by humans; a survey of some reasons for this appears in [30].

choice of representation, and institutions to achieve independence from any particular choice of logic[3]. Corelations, cocones, and colimits over arbitrary diagrams, provide a very general formalization of information integration that includes many examples from databases, cognitive linguistics, information flow, ontologies, and other areas. In addition, we argue that "triads," which are 3-ary relations of "satisfaction," "denotation," or "classification," connecting concepts and percepts, or logical and geometrical entities, are particularly appropriate models for many situations in cognitive science[4].

Barwise and Seligman have developed a philosophical theory called information flow or channel theory [3], to describe how information about one thing (or several things) can convey information about something else, i.e., how information can "flow" through "channels" to convey new information. Kent [51, 53, 52] uses this for integrating distributed ontologies. This paper generalizes information flow to any logic by using institutions, and following the lead of Kent [51], also combines it with the formal concept analysis of Wille [75, 19] and the lattice of theories approach of Sowa [70]. In addition, it draws on the categorical general systems theory of Goguen [21, 22, 26] for further generalization.

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 [19] (originating with Wille circa 1980 [75]), we let "IFF" abbreviate "Information Flow Framework" as in the IEEE Standard Upper Ontology project of Kent [53, 52], we let "LOT" abbreviate the lattice of theories approach of Sowa, we let "CGST" abbreviate the categorical general systems theory of Goguen, and we let "BT" abbreviate the blending theory of Fauconnier and Turner [16]. Applications to ontologies, as in [4, 50, 53], and to databases, as in [33], were a major inspiration for much of this paper, and are discussed in Section 3.6 and Appendix C.

The greater generality of institutions over classifications, local logics, concept lattices, etc. not only allows 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 a

[3] Readers who are not very familiar with category theory and institutions will find a definition of institution that uses no category theory in Appendix A, followed by a brief exposition of some basic notions on category theory.

[4] This name was chosen to honor Charles Sanders Peirce.

basic judgement of satisfaction, and all the theory that rests upon it, by this kind of variation. The paper takes a philosophical stance that is unusual in computer science: inspired by Peirce's pragmatics and semiotics, it rejects reductionist and realist views of information, claiming that mathematical models are just models, useful to varying extents for various applications. A philosophical motivation for parameterizing satisfaction (i.e., for using triads) also 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; thus institutions formalize a key insight of Peirce's semiotics, as discussed further in Section 3.6 and Appendix C.

Unfortunately, no easy introduction to institutions as yet exists, but a short summary is given in Appendix A, including institution morphisms and comorphisms, triads, the category of theories, and some motivation. Institutions not only have great generality and elegant theory, but they also support a powerful framework for modularization and maintenance of complex theories (such as ontologies), as shown in Example 1. Triads make it easier to explain institutions, especially their variant and multivalued generalizations, and also support integration in situations that involve both syntactic and semantic information, such as Gurevich's evolving algebra [46] (Example 2 in Section 2), and the use of triad blending to align populated ontologies in Unified Concept Theory (abbreviated UCT) [30] (Example 7 in Section 3.7).

The use of categories in this paper goes well beyond the pervasive but implicit use of category theoretic ideas in IF [3]; for those readers who are less familiar with category theory, Appendix A provides a condensed introduction, but of course there are many other sources, such as [18, 44]. We show not only that many important IF structures are the objects of categories with the relevant morphisms, but also 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 from category, institution, or CGST theories; some are new even for the special cases of traditional IF, including Propositions 3 and 5, and Theorems 5, 6 and 7. The Grothendieck flattening construction supports heterogeneity at several different levels, including the lifting of triads to form institutions.

IF, FCA, IFF, LOT, and BT are unified and generalized in Section 3; the Galois connections of institutions play 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 aspects of FCA to arbitrary institutions, particularly the concept lattice construction, and Example 3 applies it to the Curry-Howard isomorphism. Section 3.3 discusses IF channels and distributed systems; corelations are used to formalize information integration, with IF channels, database local-as-view systems, cognitive linguistic blends, and software architectures all as special cases. Section 3.4 discusses CGST, including the Interconnection Theorem, a result that is both new to and relevant for IF and FCA. Local logics are discussed in Section 3.5. Data, schema, and ontology integration are discussed in Section 3.6, including a very general way to integrate ontologies with databases in Example 4, several ways to view database systems as institutions, and a new approach to mereology[5] in Example 5, unifying inheritance (`is-a`) and constituency (`part-of`) relations in databases and ontologies, based on order sorted algebra. Applications to cognitive semantics are discussed in Section 3.7, especially the use of triads to unify conceptual spaces in the senses of Fauconnier and Gärdenfors, and to generalize both (see Examples 6 and 7). Some conclusions are given in Section 4. Appendix B explains the Grothendieck construction. Appendix C illustrates the flexibility of institutions by formalizing databases so that satisfaction says whether an answer satisfies a query in the context of a given database, inspired by the triadic semiosis of Peirce [65].

Familiarity is assumed with basics of category theory, including category, opposite category, functor, natural transformation, limit, colimit, and the categories **Set** of sets and **Cat** of categories; places to learn this material include [18, 66, 24, 56]. We use ";" for the composition of morphisms. Some mathematicians have expressed aversion to category theory, presumably due to their professional orientation towards difficult proofs of 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 results in new ways, to help solve important practical problems. Although most proofs are omitted, they can be found in the references and/or reconstructed as exercises to test understanding of the material.

[5] Sometimes also called "partology," this refers to the formal study of the "`part-of`" or "constituency" relation, also called the `has-a` 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 exceptional in both the depth and breadth of his vision of logic as far more than foundations for mathematics, and he is much missed.

Thanks to Robert Kent, Till Mossakowski, and Marco Schorlemmer for valuable comments on 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 of 24 sites doing long term ecological research.

2 Institutions and their Morphisms

Institutions[6] arose at the end of the 1970s in response to the enormous expansion of logical systems used in computer science, and the need for a uniform way to structure theories in them. Early applications included module systems for specification and knowledge representation languages; the approach was later extended to programming languages (see [42] for a general discussion), and is here further extended to ontologies and databases. The basic reference for institutions is [35], and the latest version of basic definitions appears in [40], 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 versions of higher order and modal logic, and much more; it seems that essentially any logical system has a corresponding institution, as explored in detail in [60]. There are now hundreds of papers that study and/or apply institutions. A recent contribution is Mossakowski's Heterogeneous

[6] The name "institution" was inspired by the solidly institutional status of first order logic, and to a lesser extent, of other well established logics, such as modal, intuitionistic, (higher order) type theory, and classical second order logic. Rod Burstall and I thought of the Bank of England, or the US Treasury, as models of such stability, and considered the name somewhat ironic, an implicit criticism of the tendency to downplay the importance of developing and using specialized logics (e.g., in computer science) for specialized tasks.

Tool Set [59], an extension of CASL [62] which supports verification over multiple logics, based in part on the approach to heterogeneity described in Appendix B.

Institutions abstract and generalize Tarski's "semantic definition of truth" [72], 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. Institutions generalize by parameterizing satisfaction by "context," which in many applications is a vocabulary, since many applications of logic 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 the induced changes of notation. It may be illuminating to think of this in terms of Peirce's semiotics, which extended meaning from dyadic to triadic relations [65].

$$\begin{tikzcd}
 & \mathbf{Set} \\
\mathbf{Sign} \arrow[ur, "Sen"] \arrow[dr, "\mathbf{Mod}"'] & \models \\
 & \mathbf{Cat}^{op}
\end{tikzcd}$$

$$\begin{tikzcd}
\Sigma \arrow[d, "\phi"'] & \mathbf{Mod}(\Sigma) \arrow[r, phantom, "\models_{\Sigma}"] & Sen(\Sigma) \arrow[d, "Sen(\phi)"] \\
\Sigma' & \mathbf{Mod}(\Sigma') \arrow[u, "\mathbf{Mod}(\phi)"] \arrow[r, phantom, "\models_{\Sigma'}"] & Sen(\Sigma')
\end{tikzcd}$$

Fig. 1. Structure of an Institution

Institutions use category theory to formalize the notions of context (e.g., vocabulary) and translation among contexts, as well as the effects of such translations on sentences and on models, each of which is parameterized by abstract objects called **signatures** or **contexts**, and by abstract mappings called **signature** (or **context**) **morphisms**, which form a category denoted **Sign**. Parameterization of sentences is then given by an 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 $f\colon \Sigma \to \Sigma'$, 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 $f\colon \Sigma \to \Sigma'$; both Sen and Mod are functors, but notice that Mod is contravariant; see Figure 1.

Definition 1: An **institution** consists of an abstract category **Sign**, the objects of which are called signatures, a functor $Sen\colon \mathbf{Sign} \to \mathbf{Set}$, and a functor $Mod\colon \mathbf{Sign}^{op} \to \mathbf{Set}$ (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 $f\colon \Sigma \to \Sigma'$, 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 expresses the invariance of truth under change of notation[7]. □

General concepts from model theory generalize, e.g., we can define, for T is a set of Σ-sentences, $M \models_\Sigma T$ iff $M \models_\Sigma \varphi$ for all $\varphi \in T$, and $T \models_\Sigma \varphi$ iff $M \models_\Sigma T$ implies $M \models_\Sigma \varphi$, for all Σ-models M. Similarly, if V is a class of models, let $V \models_\Sigma T$ iff $M \models_\Sigma T$ for all M in V.

Some of the earliest and most useful results about 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 [35]. The following result is less trivial, but still well known for

[7] This includes adding new symbols and identifying old symbols, as well as just renaming.

Galois connections; Section 3.1 shows how it helps to greatly generalize some results in [19].

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 all T_i. Existence of arbitrary least upper bounds implies a complete lattice. A similar argument works for the closed model classes. The two isomorphisms are the two * maps, which 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 [70], when the institution is first order logic (**FOL**, but Sowa uses an elegant Peircean syntax). Note that LOT and also IFF, use the opposite ordering relation, which seems more intuitive for many purposes; in addition, [70] also proposes a number of set theoretic operations for navigating the lattice of theories. However, I believe it is more useful to consider all the theories over all the signatures of an institution as a single *category*, as in the following from [35], because this makes several more general operations available, such as renaming operation symbols, and slicing theories.

Definition 2: Let the category $Th(\mathbf{I})$ 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 theories with different signatures are needed, for example, for integrating data from domains that have different ontologies (since ontologies are theories); see [49] for a useful survey of ontology mappings. Category theory also supports many useful operations on theories (see the discussion following Theorem 1 below).

Definition 2 is a special case of the Grothendieck construction described in Appendix B, which the reader may consult for a general 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 [52], using the fibration formulation of the Grothendieck construction, though without benefit of the unifying and generalizing notions and results of institution theory; [35] also proves the following (actually, a stronger result):

Theorem 1: $Th(\mathbf{I})$ has whatever colimits and limits the signature category of the institution $\mathbf{I}$ has. □

Colimits can be used to integrate theories over different signatures, and of course such theories could in particular be ontologies[8]. As discussed in [35, 23], and other publications beginning with [6], 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 [58], though ML does not provide all the functionality defined in [23] and implemented under the name "parameterized programming" in the OBJ language family, which includes CafeOBJ [13], Maude [8, 7], BOBJ [38], OBJ3 [43], and CASL [62]; these ideas also influenced the module systems of C++, Ada, and LOTOS, and are the basis for new approaches to software architecture, such as the CommUnity of Fiadeiro [18].

Example 1: Parameterized Programming: The category $Th(\mathbf{I})$ was invented to support flexible modularity for knowledge representation and software specification for the Clear specification language [35]. The most interesting constructions are for parameterized theories and their instantiation. An example from numerical software is `CPX[X :: RING]`, which constructs the complex integers, `CPX[INT]`, the ordinary complexes, `CPX[REAL]`, etc., where `RING` is the theory of rings, which serves as an *interface theory*, in effect a "type" for modules[9], declaring that any theory with a ring structure can serve as an argument to `CPX`. For another example, if `TRIV` is the trivial one sort theory, an interface that allows any sort of any theory as an argument, then `LIST[X :: TRIV]` denotes a parameterized theory of lists, which can be instantiated with a theory of natural numbers, `LIST[NAT]`, of Booleans, `LIST[BOOL]`, etc. Instantiation is given by the categorical pushout construction (a special case of colim-

[8] According to [24], in many practical cases, colimits capture the notion of "putting together" objects to form larger objects, in a way that takes proper account of shared substructures.

[9] This idea is elaborated in [25].

its) in $Th(\mathbf{I})$, where $\mathbf{I}$ is an institution suitable for specification theories, such as order sorted algebra, or else for programs as in [42].

Other operations on theories include renaming (e.g., renaming sorts, relations, functions) and sum; thus, compound "module expressions," such as `NAT + LIST[LIST[CPX[INT]]]` are also supported, and can be used to describe, and when "executed," to actually construct, complex software systems [74, 42]; this is the basis for the most important practical applications of parameterized programming. Colimits give the semantics of module expressions such as $E =$ `REAL + LIST[CPX[INT]]`. Figure 2 illustrates this, with B the module expression E above, $I_1 =$ `REAL`, $I_2 =$ `LIST[CPX[INT]]`, and $G =$ `INT` as a common subobject, noting that `INT` is a subtheory of `REAL`. This approach also applies to programming languages, viewed as providing programs as models for institutions with specifications as sentences [42]. □

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

Fig. 2. Information Integration over a Shared Subobject

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 less 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 $Th(\mathbf{I})$.

A related construction is given in the following:

Definition 3: The **derived institution** $\mathbf{Der}(\mathbf{I})$ of an institution $\mathbf{I}$ has as its signatures and its models, those of $\mathbf{I}$, but its Σ-sentences are Σ-theories, with satisfaction as just after Definition 1. In essence, the Σ-sentences of $\mathbf{Der}(\mathbf{I})$ are arbitrary conjunctions of Σ-sentences of $\mathbf{I}$. □

While the material above provides a good foundation[10] 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 [40], there are actually many different kinds of logic translation, useful for different purposes. The following (from [35]) is perhaps the most basic of these:

$$\begin{tikzcd}
Sen'(F(\Sigma)) \arrow[r, "a_\Sigma"] \arrow[d, "Sen'(F(\phi))"'] & Sen(\Sigma) \arrow[d, "Sen(\phi)"] \\
Sen'(F(\Sigma')) \arrow[r, "a_{\Sigma'}"'] & Sen(\Sigma')
\end{tikzcd}$$

Fig. 3. The Sentence Natural Transformation

$$\begin{tikzcd}
\mathbf{Mod}(\Sigma') \arrow[r, "b_{\Sigma'}"] \arrow[d, "\mathbf{Mod}(\phi)"'] & \mathbf{Mod}'(F(\Sigma')) \arrow[d, "\mathbf{Mod}'(F(\phi))"] \\
\mathbf{Mod}(\Sigma) \arrow[r, "b_\Sigma"'] & \mathbf{Mod}'(F(\Sigma))
\end{tikzcd}$$

Fig. 4. The Model Natural Transformation

Definition 4: An **institution morphism** from an institution $\mathbf{I}$ to another institution $\mathbf{I}'$ consists of a functor $F\colon$ **Sign** $\to$ **Sign**$'$ and two natural transformations[11] $a\colon F;Sen' \to Sen$ 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 as morphisms. □

Intuitively, institution morphisms are truth preserving translations from one logical system to another. More technically, for any signature Σ, then a_Σ maps $F(\Sigma)$-sentences to Σ-sentences, and b_Σ maps Σ-models to $F(\Sigma)$-models, in a way that is *consistent* with respect to the satisfaction relation. This consistency is enforced by the vertical maps in Figures 3 and 4 induced by $\phi\colon \Sigma \to \Sigma'$, which prevent arbitrary mappings.

For example, there is an institution morphism from first order logic with equality to equational logic, where F forgets predicates in signatures,

[10] Although ignoring many practical issues, some of which are discussed, for example, in [1, 63, 32].

[11] This notion is briefly explained in Appendix A.

where each a_Σ represents equations using the new equality predicate, and where each b_Σ forgets predicates in models. Among other variants of the institution morphism notion, perhaps the most natural is the following:

Definition 5: An **institution comorphism** from an institution $\mathbf{I}$ to another $\mathbf{I}'$ consists of a functor $F\colon \mathbf{Sign} \to \mathbf{Sign}'$, a natural transformation $a\colon \mathit{Sen} \to F;\mathit{Sen}'$, and a natural transformation $b\colon F;\mathit{Mod}' \to \mathit{Mod}$, 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. $\Box$

One class of examples arises from subinstitutions, for which F and a_Σ are inclusions, and b_Σ is forgetful. For example, there is such a morphism from **HCL** to **FOL**. A more general notion allows F and the a_Σ 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.

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 institutions with comorphisms, thus formalizing this important duality, and giving many theorems for free.

Institution (co-)morphisms formalize logic translations, and can be used to provide a very general notion of what integration means in many different contexts; they also provide many useful results, such as that logic translations preserve the modular structure of an ontology under certain mild assumptions [35]. Many results from logic have been generalized to institutions, including the Craig interpolation, Robinson consistency, Beth definability, ultrafilter, and Herbrand universe theorems; the institutional versions apply to logics where the results were already known, and yield new results for many other logics [14, 10, 11].

An alternative definition of institution in [35] supports useful generalizations and easier proofs, based on the intuition that institutions provide particular ways to interpret things in particular contexts. Formally, a triad is a relation between expressions (sentences) and entities (models), and is therefore similar to the IF notion of classification and the FCA notion of formal context, except for allowing internal structure for sentences and models[12], and allowing variation over a "signature" parameter that represents context. These generalizations are made explicit in the category of triads and the definition of institution as a functor into that category. The first step is the category **Rel** of relations, which has sets as objects and relations as morphisms, seen as triples $\langle A, R, B\rangle$ where $R \subseteq A \times B$, with the usual composition and with identity functions (i.e., diagonal relations) as identities. Next, the category of triads is the category **Trel** of "twisted relations," having triads $\langle A, R, B\rangle$ as objects, and having as morphisms $\langle A, R, B\rangle \to \langle A', R', B'\rangle$ pairs $\langle f, g\rangle$ of functions where $f\colon A' \to A$ and $g\colon B \to B'$ such that the following commutes in **Rel**,

$$\begin{tikzcd}
A \arrow[r, "R"] & B \\
A' \arrow[u, "f"] \arrow[r, "R'"'] & B' \arrow[u, "g^{\sim}"']
\end{tikzcd}$$

i.e., such that $f; R = R'; g^{\sim}$, where $g^{\sim}$ denotes the converse of g. Then an **institution** is a functor $\mathbf{I}\colon \mathbf{Sign} \to \mathbf{Trel}$.

The above covers the basic case where additional structure on models and sentences is not made explicit. It is generalized by allowing the sets and functions (but not relations) in **Trel** to be the images under forgetful functors, e.g., from **Cat** to **Set**. Let $\mathbf{A}$ and $\mathbf{B}$ be categories for model classes and sentences, respectively, with forgetful functors $U\colon \mathbf{A} \to \mathbf{Set}$ and $V\colon \mathbf{B} \to \mathbf{Set}$, and let $(U\!\downarrow / V\!\uparrow)$ denote the category[13] with objects $\langle A, R, B\rangle$ where A is an object in $\mathbf{A}$, B is an object in $\mathbf{B}$, and $R \subseteq U(A) \times V(B)$, and with morphisms $\langle A, R, B\rangle \to \langle A', R', B'\rangle$ pairs $\langle f, g\rangle$ of functions where $f\colon A' \to A$ in $\mathbf{A}$ and $g\colon B \to B'$ in $\mathbf{B}$ such that $U(f); R = R'; V(g)^{\sim}$ in **Rel**. Then

Definition 6: A **variant institution** is a functor $\mathbf{I}\colon \mathbf{Sign} \to (U\!\downarrow / V\!\uparrow)$, and is a **close variant** if $\mathbf{A}, \mathbf{B}$ are either **Set** or **Cat**. We may also use the term (U, V)-**triad** for the objects in $(U\!\downarrow / V\!\uparrow)$. $\Box$

The construction in [35] is more general, because relations are in a third category $\mathbf{C}$, which allows e.g., the "generalized institutions" of [34] where satisfaction is "multi-valued," using V-valued relations where V is a structure such as a complete lattice.

[12] Some interesting models have a geometrical structure, such as that of a sheaf [26] or of a manifold [30] (see Example 6).

[13] The notation indicates that it is a comma category, but the details of this are not necessary here, though general properties of comma categories are what eases proofs, e.g., of limits and colimits for institution categories [35].

Example 2: Evolving Algebras: The abstract state machines of Gurevich [46] (formerly called evolving algebras) can be seen as functors from $\mathbf{N}$, the natural numbers with the usual order viewed as a category, to a category of triads consisting of algebras and their specifications, both covariant rather than contravariant; hence an evolving algebra is a variant institution. (We omit many details.) $\Box$

This view of institutions as functors also gives rise to institution morphisms in a very natural way: for a fixed signature category, the institutions over that category form a functor category, $[\mathbf{Sign}, (U\downarrow / V\uparrow)]$ and allowing signatures to vary gives a functor $\mathbf{Cat} \to \mathbf{Cat}$, which we can view as an indexed category and then flatten with the Grothendieck construction of Appendix B; this gives rise to the morphisms in Definition 4 and a category that we denote $\mathbf{INS}(U, V)$. General results (e.g., Proposition E.4 of [40]) about comma categories and the Grothendieck construction now give the following:

Theorem 2: If $\mathbf{A}$, $\mathbf{B}$ have (co-)limits and if U, V preserve (co-)limits, then $(U\downarrow / V\uparrow)$ and $\mathbf{INS}(U, V)$ also have (co-)limits. $\Box$

It is perhaps surprising that institutions with the simplest possible (non-void) signatures play an important role:

Definition 7: A **1-institution** is an institution where $\mathbf{Sign} = \mathbf{1}$, the category having just one object $\bullet$, and just one morphism. $\Box$

The morphism $\bullet \to \bullet$ in **Sign** is necessarily the identity on $\bullet$, so the transformations $Sen(\mathbf{1}_\bullet)$ and $Mod(\mathbf{1}_\bullet)$ are both necessarily identity morphisms on a set, and so can be identified with that set. Thus **1**-institutions are just triads, which therefore form the category **Trel**, with morphisms the same as their morphisms as institutions.

3 Information Flow in Institutions

"Information" is central to contemporary computer science, cognitive science and philosophy, but no definition is widely accepted, and no existing definition is adequate for all the intended applications. Both IF and FCA capture key aspects of information in their formalizations, though neither provides an explicit definition of information as such[14]. This section shows that many basic IF and FCA concepts are **1**-institution concepts, and also shows that they generalize to arbitrary institutions.

[14] My definition of information in [27] is broadly consistent with the Peircean pragmatism espoused by [70], though considerably more social.

3.1 Basic Information Flow

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] builds on the intuition that information flow is only 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 [19] 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 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, it immediately gives the following as a corollary of Theorem 2:

Theorem 3: **IFC** has limits and colimits of all (small) diagrams. □

A more detailed discussion of this result is given in paragraphs just above Section 3.4 below. Note that IF-descriptions are also triads, and that **IFC** is a category of triads. The following gives still another formulation, along with some important additional IF concepts:

Definition 8: 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 $C \colon P \to 2^K$ to subsets of K. By convention we write p for

$C(p)$, and $k \models p$ instead of $p(k)$ or $(C(p))(k)$. A P-**sequent** is a pair of lists from 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 $p \in \Delta$. $\Box$

Thus, P-classifications are IF-classifications with type set P; they are also models of P in the usual sense of first order logic. 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 [35], which extends Definition 1 by allowing $Mod\colon \mathbf{Sign}^{op} \to \mathbf{Cat}$, i.e., we get a close variant institution with informorphisms as model homomorphisms. Here is a formal statement of the simpler form:

Definition 9: 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 $Sen(P) = 2^P \times 2^P$ (the set of all pairs of subsets of P), written as sequents, and with $\models_P$ the usual satisfaction relation. $\Box$

The above suggests generalizing IF to allow tokens that are terms built using arbitrary constructors[15], 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 10: The **theory of** an IF-classification C, denoted $Th(C)$, has as its constraints all sequents that satisfy C. An IF-theory is **regular**

[15] 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 many applications, including mereology for ontologies, as discussed in Section 3.6.

iff it satisfies the identity, weakening, and global cut axioms given in [3], page 119. The **classification of** a regular 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 4: $Th(Cla(T)) = T$ for any regular IF-theory T. □

The above result is extended to a categorical equivalence in [51]. Using Theorem 4 and definitions in Section 2 gives a nice (and apparently new) result, for which we first extend Cla to non-regular theories by replacing E by E^{**} in Definition 10:

Proposition 3: $Th(C) = C^*$ for any P-classification C, and an IF-theory is closed iff it is regular. Moreover, $Cla(T) \in T^*$ for any P-theory T. □

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 4 is a special case of the general institutional result that $T^{**} = T$ iff T is closed; what then becomes interesting is how the particular constructions Th and Cla relate to the Galois duality in this special case.

3.2 Formal Concept Analysis

A **formal context** in FCA [19] is the same as an IF classification[16], except that instances are called **objects**, types are called **attributes**, and classification relations are called **incidence relations**; thus formal contexts are also **1**-institutions, as well as triads. It follows from Theorem 3 that, with the natural morphisms, formal contexts form a category **FCA** that has all limits and colimits.

The main notion of FCA is that of a **formal concept** in a formal context, defined to be a pair (T, V) such that $T = V^*$ and $V = T^*$, for which FCA uses the terms **intent** and **extent**, respectively; note that T is a set of attributes, and V is a set of objects. Given a formal concept (T, V) of a formal context $(K, P, \models)$, it follows that both T and V are closed, and that they correspond under the dual isomorphism of Proposition 1. Thus the collection of all formal contexts forms a concrete complete lattice

[16] Although for consistency we mainly use IF terminology, it should be noted that FCA originated around 1980 [75], and thus predates IF. Moreover, institutions go back to the late 1970s, and the notions of polarity (which is the same as FCA formal context and IF classification) can be found in the 1948 first edition of Birkhoff's classic book on lattice theory [5].

under inclusion, of pairs of subsets; it is called the **concept lattice**, and we denote it $CL(K, P, \models)$. Conversely, given a lattice of formal concepts over sets K, P, we can define a classification by $k \models p$ iff $k \in \{p\}^*$. It is now straightforward to verify the following version of the Basic Theorem of Concept Lattices [19]:

Proposition 4: Given sets P, K, there is a bijection between classifications over P, K and their concept lattices. $\Box$

Intuitively, a concept lattice identifies tokens and types that cannot be distinguished by how they are used in a formal context. One appeal of FCA is the nice pictures that can be drawn of (sufficiently small) concept lattices; these pictures are called Hasse diagrams in lattice theory. It is natural to define concept lattice morphisms to be complete lattice homomorphisms, and then show that the above bijection is an equivalence of categories, as in [51]; many other nice categorical extensions of FCA can also be found in [51].

Formal concepts are easily defined for an arbitrary institution **I**, as pairs (T, V) such that $T = V^*$ is a set of Σ-axioms (i.e., a Σ-theory), and $V = T^*$ is a a set (or class) of Σ-models; as before, these form a complete lattice under inclusion, which we may again call a concept lattice, although for the logical systems usually considered in the literature on institutions, their Hasse diagrams are not finite. 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 extract the meaningful "concepts" for that that signature over that institution.

Proposition 4 also extends to arbitrary institutions. Given an institution **I**, define a functor $CL(\mathbf{I})$ on the signature category of **I**, with $CL(\mathbf{I})(\Sigma)$ the complete lattice of Σ-concepts (T, V), and with $CL(\mathbf{I})(f)$ the map that sends (T, V) to $(Sen(f)(T),\ Sen(f)(T)^*)$. Let **CLA** denote the category of complete lattices. Then

Proposition 5: CL is an injective functor from **INS** to the comma category $(\mathbf{Cat}/\mathbf{CLA})$, and hence it induces an equivalence of its source and image categories. $\Box$

Definition 11: Given an institution **I**, its **concept lattice institution**[17] **CL(I)** has signatures those of **I**, Σ-sentences the closed Σ-theories, Σ-models the closed Σ-model classes, and Σ-satisfaction the bijection rela-

[17] [33] calls **CL(I)** the **Galoisification** of **I**; this notion was stimulated by persistent questions in conversations with Erick and Linda von Schweber, and Liane Gabora.

tion between these given in Proposition 5. □

Proposition 5 also implies that **CL** can be seen as an endofunctor on **INS**, which we may call the **concept lattice functor**.

In many cases, formal concepts can be considered the "natural concepts" for a context; for example, given sentences over an ontology and models that populate that ontology, the formal concepts help understand the meaning of these objects in this context; this can be useful, e.g., in constructing ontologies, as illustrated in [50, 30]. Unfortunately, real data is often too dirty for this to work well, which suggests that new ideas are needed to deal with the complex uncertainties of real world data.

A number of extensions of FCA have been proposed to improve its expressive power, several of which are discussed in [19]; these include power contexts, which support n-ary relations for a fixed n, and multi-valued contexts, which allow non-binary truth values for the incidence relation. Various "logic contexts" have also been proposed to extend attributes with syntax, e.g., [17]. In my view, none of these extensions are necessary, because institutions allow concept lattices over arbitrary logics (e.g., modal logics, description logics, higher order logics), generalized institutions allow arbitrary truth values, and the usual FCA results can all be obtained using the categorical machinery of variant institutions. In my opinion, the major advantage of the IF formalism over that of FCA is its emphasis on informorphisms and information "flow" via (co)limits.

Example 3: Curry-Howard Isomorphism: A perhaps surprising application of the concept lattice construction is an institutional formulation of the Curry-Howard isomorphism, one of the most beautiful results in logic that is connected to computer science (e.g., see [73]). Its basic form asserts an isomorphism between types "inhabited" by λ-terms (which can be thought of as programs) and intuitionistically valid Boolean formulae. We outline the simplest non-trivial case, which uses only the binary function type constructor $\rightarrow$, but the approach extends to more complex types; no proofs are given, the goal is just to get an elegant formulation that makes the semantic aspects of this result more explicit than usual.

The signature categories of all institutions we build are the full subcategory of finite subsets of some fixed countable set of propositional variable symbols (though any full subcategory of **Set** would do as well); we will use Greek letters for these variable symbols. Given a variable set V, let $T(V)$ be the free type algebra on V, with elements propositions (built from just $\rightarrow$ in this case) such as $((\alpha \rightarrow \beta) \rightarrow (\beta \rightarrow \gamma)) \rightarrow (\alpha \rightarrow \gamma)$, let $B(V)$ be the free term algebra on V with elements (built from just $\Rightarrow$ in this case) such as $((\alpha \Rightarrow \beta) \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow (\alpha \Rightarrow \gamma)$, let $\Lambda(X)$ be the free algebra of lambda terms over X with elements such as $\lambda x\, y\, z.\, x(y\, z)$, and let $\Lambda^{\bullet}(X)$ denote the closed λ-terms over X; whereas types use variables from V, λ-terms use some other fixed countable set X of variable symbols. Next, for $b \in B(V)$ let $b^{\diamond}$ be the corresponding type in $T(V)$, obtained by replacing $\Rightarrow$ by $\rightarrow$, and for $t \in T(V)$ let $t^{\diamond}$ be the corresponding term in $B(V)$; note that $^{\diamond}$ is an isomorphism.

The first institution, for a signature V has $T(V)$ for its V-sentences, and has typed closed λ-terms as its V-models, i.e., pairs $\langle f,\, a\colon\ X \rightarrow T(V)\rangle$ with $f \in \Lambda^{\bullet}(X)$. Satisfaction $\langle f, a\rangle \models_V t$ holds iff f inhabits t with assignment a, i.e., iff $\overline{a}(f) = t$ where $\overline{a}\colon\ \Lambda^{\bullet}(X) \rightarrow T(V)$ is the free extension of a. Signature morphisms do nothing to λ-terms, but transform types by variable renaming. To obtain an institution, we take the opposite of the category of models, so that they transform contravariantly. Let $\mathbf{\Lambda}$ denote this institution.

The second institution, denoted $\mathbf{B}$, for a signature V has $B(V)$ for its V-sentences, and for its V-models has assignments $h\colon\ V \rightarrow H$ where H is an arbitrary Heyting algebra[18]. Satisfaction $h \models_V b$ holds iff $\overline{h}(b) = \top$ where $\overline{h}\colon\ B(V) \rightarrow H$ is the free extension of h to H, i.e., iff the formula b is intuitionistically valid. Signature morphisms transform terms covariantly and models contravariantly in the obvious way.

These two institutions are not isomorphic or even equivalent, nor are their concept lattice institutions, but their concept lattices provide key structures for the Curry-Howard result. Concepts in $\mathbf{CL}(\mathbf{\Lambda})$ for a fixed V collect all λ-terms that inhabit a given type, since the extension $\{t\}^*$ of a type t is $\{\langle f, a\rangle \mid \overline{a}(f) = t\}$, which we denote $\Lambda(t)$, and $\{\langle f, a\rangle\}^* = \{\overline{a}(f)\}$, so that $\{t\}^{**} = \{t\}$ and $\{\langle f, a\rangle\}^{**} = \{\langle f', a'\rangle \mid \overline{a'}(f') = \overline{a}(f)\}$; this includes λ-terms that differ by only a bijective renaming $\varphi\colon\ X \rightarrow X$ of their variables. The concept generated by a type $t \in T(V)$ is $\langle \{t\}^*, \{t\}\rangle$, which is $\langle \emptyset, \{t\}\rangle$ if t is uninhabited.

Concepts in $\mathbf{CL}(\mathbf{B})$ are terms with all the assignments into Heyting algebras that make them true, because $\{b\}^{**} = \{b\}$, and $\{h\}^* = \{b \mid \overline{h}(b) = \top\}$. Let $tt(b)$ denote the assignments $h\colon\ V \rightarrow H$ that makes b true; these can be thought of as generalized truth tables. We now express the Curry-Howard isomorphism for a fixed V as a bijection between the inhabited concepts $\langle \Lambda(t), \{t\}\rangle$ in $\mathbf{CL}(\mathbf{\Lambda})$ and the elements of the concept $\langle tt(T), \{T\}\rangle$ in $\mathbf{CL}(\mathbf{B})$ where T is any tautology, i.e., the intuitionistically

[18] Heyting algebras are needed instead of Boolean algebras in order to capture intuitionistic validity; Peirce's formula is a well known example b involving only $\Rightarrow$ that is not intuitionistically valid.

valid formulae, i.e., those b where $tt(b)$ includes all assignments to all Heyting algebras.

This institutional formulation makes the semantic aspects of the Curry-Howard isomorphism explicit, whereas the usual formulations are purely syntactic. The approach extends to richer types and more complex λ-expressions. Some recent work in this direction appears in [61]; one relatively straightforward approach takes a syntactic view of models as proofs of habitability and validity in appropriate formal systems. $\Box$

3.3 Channel Theory and Information Integration

There is a very general notion of a **relation** in a category $\mathbf{C}$, as 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 $\mathbf{Set}$, then relations can be considered to consist of pairs $\langle a, b\rangle$ with $a \in A$ and $b \in B$, and with p_1, p_2 the **projection** maps. Most of the usual theory of relations can be done in this very general setting (see below). There is also a dual notion of **corelation**, consisting of three objects A, B, C and two **injection** morphisms, $f_1\colon\ A \to C$ and $f_2\colon\ B \to C$. These binary relations (and corelations) generalize to families $p_i\colon\ R \to A_i$ (or $f_i\colon\ A_i \to C$), which are n-ary relations (and corelations). We now give the categorical form of another basic concept of IF:

Definition 12: A **channel** is a corelation $f_i\colon\ A_i \to C$ for $i \in I$, in the category **IFC**; C is called the **core** of the channel. $\Box$

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 general categorical notion of a **cocone** of a diagram, for the special case of a diagram in **IFC**; similarly, the "minimal covers" of [3] are the **colimits** in **IFC**, although [3] uses 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). Extending the geometrical analogy, category theorists often use the term **apex** instead of "core" for both cones and cocones.

It is encouraging that corelations arise in many interesting examples, not just channel theory, but also local-as-view[19] data integration in

[19] The dual global-as-view approach corresponds to relations in a category, but can also be formulated using corelations.

database theory, blending in cognitive linguistics [16], module interconnection [42] and concurrent process interconnection [26, 18] in software engineering, and (following algebraic semiotics [29, 28]) the composition of subsigns to form interfaces in user interface design. It is therefore reasonable to suggest that corelations, cocones, and colimits provide one very reasonable notion for integration of the constituent objects [21, 24]. Note that in the case of channels, the integrated objects are classifications, or more generally, triads, not just theories.

Many information integration problems involve a subtheory of shared material. Diagrams with morphisms can describe complex sharing among objects, and colimits of such diagrams combine those objects in a way that optimally respects that sharing, in the usual universal sense. Given input theories I_1, I_2 with a shared subobject G (as in Figure 2), the colimit (which is a pushout) gives such an optimal corelation B. However, in some practical situations, it is not realistic to expect this kind of optimality, because there are multiple feasible solutions, among which trade-offs must be negotiated; in such cases, one should consider pragmatic optimality criteria like those used in the cognitive linguistics notion of blending [16] discussed in Section 3.7 below. In [28], it is suggested that so called **$\frac{3}{2}$-colimits** in **$\frac{3}{2}$-categories**, which have an ordering relation on the morphisms between any two objects, have the necessary flexibility for applications to areas like cognitive linguistics and user interface design.

Relations in a category with pullbacks satisfy the usual calculus of set theoretic relations, i.e., composition, converse, union, intersection, etc., have their usual properties, e.g., composition is associative. The composition $R;R'$ of binary relations R, R' is given by letting $R;R'$ with its projection arrows be the pushout of the lower "M" shaped subdiagram of the following:

$$\begin{tikzcd}
 & & R;R' \arrow[dl] \arrow[dr] & & \\
 & R \arrow[dl] \arrow[dr] & & R' \arrow[dl] \arrow[dr] & \\
A & & B & & C
\end{tikzcd}$$

The dual calculus of corelations is less well known but is just as rich, and everything generalizes to n-ary relations and corelations. For example, the join of relations in database theory is a special case of composing two n-ary relations over projections to a common subrelation.

Theorem 3 implies 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 of [24] and of CGST described in Section 3.4). What IF calls the **sum** of a set of classifications is the special case where the system diagram is discrete, i.e., has no infomorphisms; here the token set is the product of the token sets of the components. Theorem 3 can be considered a generalization of the "Dretske Xerox Principle" discussed in [3], which asserts the "transitivity of information flow"[20]. In addition, it makes available the powerful structuring and integration mechanisms of parameterized programming that are discussed in Example 1 of Section 2.

The assertion in Theorem 3 that **IFC** has limits is not in [3]. 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. In the special case where the diagram is discrete, its type set is the product of the type sets of its components, and its token set is the disjoint union of the token sets of its components.

Barwise and Seligman [3] cite Chu spaces (introduced in the appendix of [2]) for their proof that **IFC** has colimits; 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**[21] of [34], in which satisfaction takes values in a suitable structure V; the so called Chu transformations are of course the comorphisms of such institutions.

3.4 Categorical General Systems Theory

In 1971, the categorical general system theory (CGST) of [21] proposed several 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; the latter can be considered an even further generalization of the Dretske Xerox Principle. CGST amounts to doing information flow and logical architecture in an arbitrary category. An important idea from [21] not in [3] is that the colimit of a diagram of systems computes the system that results from the

[20] 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 [22, 56]).

[21] This can be accomplished by allowing V-valued relations in the category of twisted relations; thus Chu spaces are variant institutions. See also the end of Section 2 of [71], which gives a better exposition of this than that in [34].

interconnection. A main result, called the **Interconnection Theorem** [22], 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 [26] provides a better exposition than that in [22]):

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

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

□

Specialized to **IFC**, 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 formalism of [21] is applied to various distributed systems in [26], which also treats 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 could 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 noting the logic for sheaves given by the internal logic of a topos in [26].

3.5 Local Logics

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

Definition 13: 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 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 [40]; as before, it enables the structuring and integration mechanisms of parameterized programming.

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; this 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 define 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; this is a Grothendieck construction, as described in Appendix B, which the reader may wish to consult at this time. 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 automatically, 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 [40]. And again, it makes available the structuring and integration mechanisms of parameterized programming. Restricting to the closed theories and model classes of **IFL** for a fixed signature, we can construct concept lattices for local classifications just as above for **IFCL**; we conjecture that the resulting category of local concept lattices is equivalent to the category of local classifications.

It is natural to consider further examples in the style of **IFC**, **IFS**, etc., such as 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, **IFE**, for which the same results follow in the same uniform way; we can even add subsets of normal tokens. Horn clause logic is an alternative for which inference is easier, and description logics go even further in this direction.

3.6 Data, Schema, and Ontology Integration

Information integration over distributed databases, such as the world wide web, is a significant application for ideas discussed in this paper, but it is very challenging, requiring integration of both schemas and ontologies with data. General categorical principles [24] say that because schemas and ontologies are theories, they should be combined using colimits. But because different signatures are involved, the category of theories over a fixed signature is inadequate; therefore we extend it using the Grothendieck construction of Appendix B. (Of course, this only addresses a few of the many practical problems involved.)

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 with heteromorphisms. Next, applying the Grothendieck flattening 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 restricted to institutions and morphisms in $\mathbf{O}$; let the result be denoted **GTh(O)**. (**GTh** and **GTh(O)** can also be obtained from the Grothendieck institution construction of [9].) It is not difficult to show that limits and colimits exist in these categories under reasonable conditions; see [33] for details.

An ontology is a different kind of theory for databases than a schema, because its purpose is to give language for describing entities represented by elements that can appear in databases, rather than to describe the structure of the database. Practical ontologies are often restricted to constants that denote individuals, unary predicates that classify individuals, and binary predicates that relate individuals, while sentences are often restricted to special kinds of Horn clause. However, such restrictions are by no means necessary, and are imposed mainly to ensure efficient decidability. The illustrative examples below use Horn clause logic with equality over order sorted algebra, but in general, an **ontology** O consists of an arbitrary theory extending a theory D of a model Ω, over an arbitrary institution $\mathbf{I}$ such that its model category is concrete; O is the ontology and Ω is the "population" of values that can be used to fill slots in databases. It is convenient to assume that all elements of Ω are also constants in D.

A **schema** is a theory T over $\mathbf{I}$ that contains D, but in general only initial models of T are of interest[22]. A **database state** over the schema T is an element of an initial model B of T such that $B|_D = \Omega$, and a **query** over T is an existential sentence over the signature of T. When both a schema and an ontology are available, existential sentences over the pushout of T and O over D (which we denote TO) can also be used as queries. More concrete definitions of schema, as well as of schema species, schema morphism, database, query, etc. are given in [33].

[22] A model is **initial** if there is a unique homomorphism from it to any other model.

Now we have everything needed for our database institution. Its signatures are schemas over **I** as defined above, with theory morphisms as signature morphisms; the models of a schema T are the database states for a fixed database B for 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 a 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 **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 $f \& f' \models B$ iff $f \models B$ and $f' \models B$, in which case conjunction is associative, commutative, and idempotent. Similar tricks work for other logical operations, including existential quantification in case it is missing from **I**, although this requires further assumptions on **I** (see Definition 5.17 of [60]). The following gives another way to institutionalize databases:

Example 4: A Course Database: Let **I** be Horn clause logic with order sorted equality; let D be an order sorted theory having 4 sorts, for naturals, reals, booleans, and strings of characters, abbreviated respectively N, R, B, C, where N is a subsort of R; and let D also have all the usual operations on these sorts. Let Ω be the D-algebra where all sorts and operations are interpreted in the usual way.

We define next the theory O. Let $CSCourse$, $MathCourse$, $ReqCourse$, $OptCourse$ be unary predicates in O, and let

$$\begin{array}{ll} ReqCourse(X) & \text{implies } CSCourse(X) \\ MathCourse(X) & \text{implies } OptCourse(X) \end{array}$$

be sentences in O, where X has sort N. Now let O classify items with axioms such as

$$\begin{array}{l} MathCourse(329) \\ ReqCourse(130) \\ OptCourse(171) \end{array}$$

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 following 2-tuples:

$(329, \mathit{Category\ Theory})$,
$(171, \mathit{User\ Interface\ Design})$,
$(130, \mathit{Programming\ Language\ Concepts})$,
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 $\mathit{CourseId}\colon R2 \to \Omega_N$ and $\mathit{CourseName}\colon R2 \to \Omega_C$. Then the following is a key constraint

$$\mathit{CourseId}(r_1) = \mathit{CourseId}(r_1') \text{ implies } r_1 = r_1'$$

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 N, and Y, Y' range over C. The following is a data integrity constraint

$$0 < \mathit{CourseId}(R1(X,Y)) < 400$$

which is again an abbreviation, for

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

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

Key constraints support a weak form of unique identity for entities, a topic much discussed 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 $X_1, ..., X_n$). Tool support for inference over a combined database and ontology could convert database entries to Horn clauses. $\square$

Schema heteromorphisms can be defined (as in 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 $f : \Phi; (f) \to S'$ 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 with heterogeneous databases as models and queries as sentences, over $\mathbf{I}$.

The above treatment of queries is not fully satisfactory because we can only know whether or not a query has an answer, but not what that 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 Mayoh [57], and requires the generalized V-institutions of [34]. However generalized institutions can be avoided by defining an institution $\mathbf{DB}'(\mathbf{I}, O, \Omega)$ like $\mathbf{DB}(\mathbf{I}, O, \Omega)$ except that sentences have the form $q \vdash \theta$, and are satisfied by B iff $\theta(q)$ satisfies B in $\mathbf{DB}(\mathbf{I}, O, \Omega)$ with the substitution θ, which 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 still more sophisticated institutionalization of database systems is given in Appendix C.

On the other hand, a less sophisticated approach ignores queries, as in [1], which applies institutions to databases, using the term "schema translation framework" for "institution," "database" for "model," "constraint" for "sentence" and "schema" for "theory"; neither queries nor heteromorphisms are considered, but integration is conceptualized as a corelation over a shared part arising from an initial signature, with colimit as the ideal result if it is defined.

Although Horn clauses are the most common sentences for ontologies and schemas in the literature, 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 subsumptions 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 Ω are required to respect that ordering, in that if $s \leq s'$ then $\Omega_s \subseteq \Omega_{s'}$ so that classification and class subsumption are in signatures rather than theories, an approach which has been found more convenient for many purposes (e.g., see [43]). In addition, order sorted algebra allows overloaded operation symbols, and handles their subtle interactions with subsorting. See [39] for the formal details of order sorted algebra.

It is even more interesting to consider the `has-a` or `part-of` relation, because it is so often problematic in real ontologies, for reasons nicely discussed in [45]. 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; this is symptomatic of the difficulties involved. In order sorted algebra, as in algebra generally, elements are represented as terms, which are built from constants and other terms, using operations called **constructors**. For example, if c is a binary constructor and if $a = c(b1, b2)$, then we can 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. (Although not all real world constituent hierarchies can be formalized using constructors, 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 seems more useful than those based on formal mereology. We now illustrate the order sorted algebra approach on the example given earlier.

Example 5: Course Database Revisited: Let D have sorts $CourseID$, $StudentId$, $CourseName$, $StudentName$, $CourseGrade$, GPA, represented in Ω as integers for the first two, strings for the second two, and reals for the final two, each with the usual operations. Let D also have complex data sorts $Course$, $Student$, $Grade$, with the constructors:

$$\begin{array}{lll}
course: & CourseId\ \ CourseName & \to\ Course \\
student: & StudentId\ \ StudentName\ \ GPA & \to\ Student \\
grade: & CourseId\ \ StudentId\ \ CourseGrade & \to\ Grade
\end{array}$$

Let O add the following subsorts to D

$$\begin{array}{l}
ReqCourse\ \ < CSCourse < CourseId \\
MathCourse < OptCourse < CourseId
\end{array}$$

plus some "population" classifications, such as

$$\begin{array}{l}
329: MathCourse \\
130: ReqCourse \\
171: CSCourse
\end{array}$$

and finally, let T declare further sorts $DataBase$, $R1$, $R2$, $R3$, and further constructors:

$$\begin{array}{lll}
database\colon & R1\ R2\ R3 & \to Database \\
R1add\colon & R1\ Course & \to R1 \\
R2add\colon & R2\ Student & \to R2 \\
R3add\colon & R3\ Grade & \to R3
\end{array}$$

with equations such that $R1, R2, R3$ are sets (or bags) of elements of sorts *Course*, *Student*, *Grade* respectively, with also an empty set (or bag) for each. Then under initial algebra semantics [41], elements of sort *DataBase* will consists of three relations, and the initial algebra itself will contain all such possible states of this database. Queries can be posed over the combined theory TO, such as, find all students with an average grade more than 3.5 in their required courses. See [33] for a more detailed exposition of ideas related to this algebraic style of database semantics.

We can now see how the subsort relation express an `is-a` relation, e.g., $ReqCourse < CSCourse$ says that all required courses are CS courses. Also, the subterm relation expresses a `part-of` relation, e.g., that the *course* constructor has arguments of sort *CourseId* and *CourseName* implies that course Ids and course names are the parts of the individual "records" in the *Course* relation. As in Example 4, it is interesting to consider tools that answer queries over such systems. For queries involving `part-of`, unique identifiers for basic parts and contexts of superordinate terms will be important. □

3.7 Cognitive Semantics

Fauconnier's *conceptual spaces* [16], originally called *mental spaces* [15], formalize the important idea that concepts are used in clusters of related concepts, represented in a very simple logic consisting of individual constants and assertions that certain relations (mostly binary) hold among certain of those constants[23]; it is remarkable how much natural language semantics can be encoded in this framework [15, 16]. Figure 5 shows two simple conceptual spaces, the first for "house" and the second for "boat." These do not give all possible information about these concepts, but only the minimal amount needed for a particular application. The "dots" represent the individual constants, and the lines represent true instances of relations among those individuals. Thus, the leftmost line asserts `own(owner, house)`, which means that the relation `own` holds between these two constants.

[23] Thus the word "space" is used metaphorically in these phrases "conceptual space" and "mental space." Our later discussion of triads will make it clearer why it is an appropriate metaphor.

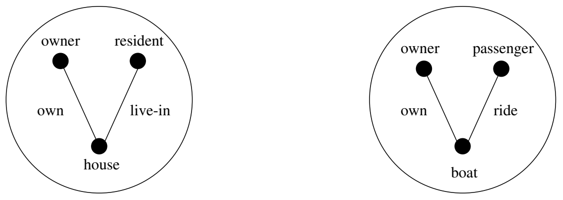

Fig. 5. Two Simple Conceptual Spaces

Algebraic semiotics [28] uses algebraic theories to provide features beyond those of conceptual spaces, that are important for user interface design. These extensions are needed because many signs are complex, i.e., they have parts, and these parts can only be put together in certain ways, e.g., the icons and windows of a PC interface, the words in a sentence, or the visual constituents of the diagram in Figure 5. *Semiotic spaces* further extend algebraic theories by adding priority relations on sorts and constructors, which express the relative importance of (sub)signs; this information is vital for optimization in user interface design applications [28, 29]. The intended denotation of a semiotic space is not a single sign, but (in the spirit of Saussure [68]) a family of signs, such as all possible displays on a particular digital clock, or on a particular cell phone.

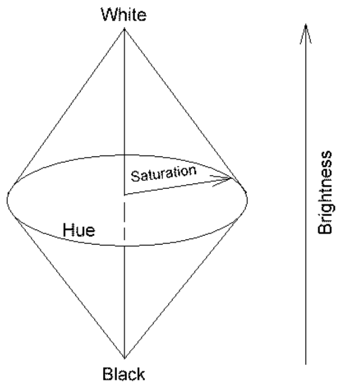

Fig. 6. The Human Color Manifold

Gärdenfors [20] proposes a notion of conceptual space very different from that of Fauconnier, since it is based on geometry rather than logic. For example, a sort `color` might be interpreted as the set of points in a fixed 3D manifold representing human color space, coordinatized by hue, saturation and brightness values, as shown in Figure 6; this is a precise model that can serve as a basis for reasoning about color properties.

Although [20] says it aims to reconcile its geometric conceptual spaces with symbolic representations like those of Fauconnier, it does not in fact provide a unified framework. The Unified Concept Theory (**UCT**) of [30] uses triads to combine a symbolic space (or theory), a geometrical space (or model), and a relation of satisfaction between them for the given context. The model provides a set of instances for each sort, a function for each function symbol, and a relation for each relation symbol. Triads give a solution to the *symbol grounding problem*[24], because symbols in conceptual spaces can refer to entities in geometrical spaces, which in turn can be identified with real world entities. This way of combining formal models with concrete interpretations works over any *concrete institution*, where symbols have sorts and models are many-sorted sets (see e.g. [60] for details of this notion). We should also require that geometrical space morphisms preserve the relevant geometrical structure. One of the most intriguing ideas in [20] is that all geometrical conceptual spaces are convex[25]; it is nicely used in the following:

Example 6: Human Color Manifold: A suggestive example in [20] concerns the (English) terms used to describe human skin tones (red, black, white, yellow, etc.), which have very different meanings in that context than e.g., in a context for describing fabrics. Gärdenfors explains this shift of meaning by embedding the space of human skin tones within the larger color manifold (see Figure 6), and showing that the standard regions for the given color names are the closest fits to the corresponding skin tone names. Gärdenfors does not give a formal treatment of the color terms themselves, but we can view them as belonging to a (logic) *theory* of color, and view their semantics as a *triad* between a color theory and a color model [30].

It is technically convenient to view the color manifold and its skin tone submanifold as related by a canonical projection from the color manifold to the submanifold; in fact, the convexity assumption guarantees that such a canonical projection exists. Then the embedding and projection morphisms of the color manifolds give rise to triad morphisms. A suitable variant institution for this example has convex smooth manifolds[26] as models, has sentences built from constants and binary relations, and has satisfaction defined by interpretions of constants and relations as smooth

[24] This is the problem in classic logic-based AI, of how abstract computational symbols can be made to refer to entities in the real world [47]. Our approach is also related to the "material anchors" of Hutchins [48].

[25] A subset of Euclidean space is **convex** if the straight line between any two points inside the subset also lies inside the subset; this generalizes to non-Euclidean manifolds by using geodesics instead of straight lines.

[26] Smooth is usually taken to mean that the space has continuous derivatives of all orders at every point.

convex submanifolds and relations; satisfaction even can be multi-valued if desired. □

An important recent development in current cognitive semantics is *conceptual blending*, claimed in [16] to be a (previously unrecognized, and unconsiously operating) fundamental cognitive operation, which combines different conceptual spaces into a unified whole. The simplest case is as in Figure 2, where I_1, I_2 might be mental spaces for "house" and "boat" (as in Figure 5), with G containing so-called "generic" elements such that the maps $G \to I_i$ indicate which individuals should be identified. In contrast to the categorical notion of colimit, blends are *not* determined uniquely up to isomorphism; for example, B could be "houseboat," or "boathouse," or some other combination of the two input spaces I_i (see [37] for a detailed discussion of this example[27]). The $\frac{3}{2}$-categories discussed in Section 3.3 address this problem.

Several "optimality principles" are given in Chapter 16 of [16] for judging the quality of blends, and hence determining which blends are most suitable, although these distillations of common sense are far from being formal. Unfortunately, it does not seem, as suggested in [16], that any single set of optimality criteria can be appropriate for all situations, but rather that different criteria are needed for different situations. For example, [36] examined metaphors in the poetry of Neruda and Rilke, and found that especially creative blends often used criteria opposite to the common sense criteria of [16]; one such blend is the phrase "water of beginnings and ashes" in the first stanza of Neruda's "Walking Around" [64]. Thus, much recent poetry requires what we call "disoptimality principles" to explain such metaphors; one such principle is casting a type to another very distant type [36].

Blending theory [16] refines the metaphor theory of Lakoff, by proposing that a metaphorical mapping from I_1 to I_2 is a kind of "side effect" of a blend B of I_2 and I_2, because a metaphor really constructs a new space in which only certain parts of I_1 and I_2 appear, and in which some "emergent" structure found in neither I_1 nor I_2 may also appear; the usual formulation of metaphor as a "cross space mapping" $m\colon I_1 \to I_2$ reflects the identifications made in B, i.e., if i_1, i_2 are constants in I_1, I_2 respectively, that map to the same constant in B, then we set $m(i_1) = i_2$. For example, in the metaphor "the sun is a king," the constants "sun" and "king" from the input spaces are identified in the blend, so "sun"

[27] The discussion in [37] can be extended to triads and truad blending by providing prototypes and examplars for the concepts and relations in the logical theories shown in Figure 5.

maps to "king," but the hydrogen in the sun is (probably) not mapped up or across; similarly, the fact that kings may collect taxes is probably not mapped up to B. But if we add the clause "the corona is his crown," then another element is added to the cross space map.

Our formalization of blending theory is tested against some simple examples in [28], and has been implemented in a blending algorithm called Alloy [37, 36], which generates novel metaphors on the fly as part of a poetry generation system (called Griot) implemented by Fox Harrell. The optimality principles used are purely formal (since they could not otherwise be implemented); they measure the degree to which the injection morphisms $I_i \to B$ in Figure 2 preserve structure, including constants, relation instances, and types. We are currently extending the system to blend and display multimedia objects.

Example 7: Ontology Alignment: A new idea from [30] is to integrate triads instead of just theories. We illustrate this using a simple example from Sowa [70], elaborated in [69] to illustrate "ontology alignment" and "ontology merging," where the former refers to how concepts relate, and the latter refers to creating a joint ontology. The present exposition extends [30] by explicitly using a pushout of triads to integrate populated ontologies; these triads have models on one side, and theories with their signatures on the other. Besides its clarity and rigor, the approach has the advantage of making it clear how to generalize to examples that use less trivial models and logics. The informal semantics behind this example is explained in the following from [70]:

> In English, size is the feature that distinguishes "river" from "stream"; in French, a "fleuve" is a river that flows into the sea, and a "riviére" is a river or a stream that runs into another river.

This example has 2 triads, each with 3 objects and 2 attributes:

English	river	stream
Mississippi	1	0
Ohio	1	0
Captina	0	1

French	fleuve	riviére
Rhône	1	0
Saône	0	1
Roubion	0	1

This information is not enough to recover the informal relations between concepts in the quotation above, but if we can align the objects, a surprising amount of insight can be obtained. So we further suppose it is known that the corresponding rows of the two tables have the same conceptual properties, e.g., that the Mississippi is a fleuve but not a riviére, that the Saône is a river but not a stream, etc.

The underlying logic for this example is very simple: Its signatures are sets of unary predicates. Its sentences over a signature Σ are the nullary predicate *false*, and the unary predicates in Σ. Signature morphisms are maps of unary predicate names. A model for Σ is a one point set, with some subset of predicates in Σ designated as being satisfied; the satisfaction relation for model is then given by this designation. The resulting institution is really **IFC** in disguise, for which $Th(\Sigma)$ has as its objects the set of all sets of Σ-sentences. But since this example uses triads where satisfaction relates models to theories instead of just sentences, we use the derived institution **Der**(**IFC**), where the Σ-sentence for triads are sets of predicates, interpreted as their conjunction.

The English signature Σ^E in this example is {river, stream} and the French signature is Σ^F = {fleuve, riviére}. The tables above define two triads, one over each signature. We will use a pushout to blend them, as in Figure 2. Since the entities corresponding to the three rows of the two tables have the same properties, we can merge them in the blend; on the other hand, none of the attributes should be merged. Therefore the base triad for the blend has three generic objects that map to the models to be identified in the blend, and the logic component is the empty theory. Denoting the merged entities by MR, OS and CR, the pushout triad is given by the following:

	river	stream	fleuve	riviére
MR	1	0	1	0
OS	1	0	0	1
CR	0	1	0	1

Note that triad pushouts compute the limit (pullback) of models and the colimit of theories with their signatures. In particular, the signature for the blend theory is $\Sigma^B = \Sigma^E \cup \Sigma^F$.

Figure 7 shows the formal concept lattice of this merged triad over the merged signature Σ^B, with its nodes labeled by the model sets and theories to which they correspond. A minimal set of generators for this lattice gives a canonical formal vocabulary for classifying the given models. Although in this case there are $2^3 = 8$ possible sets of models, only 7 appear in the concept lattice and a subset of 3 is sufficient to generate the lattice, namely fleuve, stream, and river&riviére. (The reason there are only 7 elements is that there is no model that both is small and flows into the sea.) □

The Buddhist monk example in [30], which formalizes and analyzes the version in [16], can also be seen as a blend of triads.

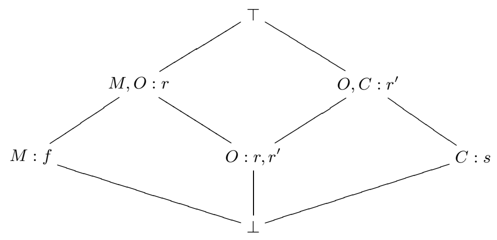

Fig. 7. A formal concept lattice

Most schema and ontology mapping tools seek to construct a correspondence between two input spaces[28]; see [50, 67]. The above example shows that this process can be non-trivial, and it also suggests a way, extending prior work of Shorlemmer and Kalfoglou [49, 69], to achieve partial automation, based on our very general theory of information integration using colimits of triads over arbitrary institutions.

4 Conclusions

The main contribution of this paper is to unify and greatly generalize IF, FCA, IFF, LOT and BT to arbitrary logical systems, including the use of heterogeneous triads for integration in situations that involve multiple logical systems, such as populated ontologies in different description logics, with no restriction to unary predicates. Triads also unify and generalize conceptual spaces in the senses of Fauconnier and of Gärdenfors. The Grothendieck construction uses heteromorphisms to support heterogeneity at multiple levels, including the integration of multiple ontologies, and even multiple ontologies over multiple logics; it also supports the integration of databases over multiple schemas. Triads also make it easier to explain institutions, especially their variant and multivalued generalizations, and triads. The flexability of using abstract categories allows exotic models that can handle change of types and tokens over time. These are all issues that standard IF has left open. Some novel formalizations of databases as institutions provide very general ways to connect databases and ontologies. Other new ideas include using order sorted algebra to unify inheritance (`is-a`) and constituency (`part-of`) relations in databases and ontologies, generalizing FCA to the concept lattice institution, and using this construction in an institutional version of the Curry-Howard isomorphism.

Rising above details, some interesting patterns emerge. Each classification logic (**IFC**, **IFS**, **IFCL**, **IFL**) gives rise to institutions at two

[28] Among the many terms used for this are alignment, articulation, fusion, coordination, merge, and reconciliation.

levels: (1) classifications are institutions of the simplest non-trivial kind, **1**-institutions, i.e., triads; and (2) these simple institutions form the model categories of other institutions with non-trivial sentences. The institutional formulations yield very general results in a uniform way using established methods, and many are new for at least one of IF, FCA and LOT, including the Galois connection between models and theories, several (co-)completeness results, the Interconnection Theorem, and the concept lattice institution.

In addition, over any of these institutions, channels are corelations (or cocones), distributed systems are diagrams, and behaviors are colimits, and the same applies for many other formalisms for information integration, including databases, conceptual blending, and areas of user interface design and software engineering. Therefore ideas from IF, FCA, IFF, LOT and BT can be applied in these aspects, supported by a precise and very general theoretical framework.

There are many other directions that seem worth exploring, including parts of [3], [19] and [70] that are not yet institutionalized; the categorical formulations of Kent [51] are very relevant here. It would be useful to extend the Galois connection between theories and model classes to an indexed family of adjunctions for variant institutions, e.g., where the sentence and model functors yield categories (with sentence morphisms for deduction in the style of categorical logic [55] as in [60]). It would also be interesting to further explore the use of sheafs as space- and time-varying tokens and types in classification systems, following [26]. Tools based on methods in this paper to support query answering, inference, and ontology alignment over a combined ontology and database would be useful; triads should be important here. Finally, it seems necessary for theory and tools to cope with the multiple forms of uncertainty inherent in the real world (e.g., see [27] 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 help.

Our generalization and unification of IF, FCA, LOT, IFF, and BT does not detract from the practical and philosophical achievements of these works, nor does it detract from the expository simplicity of the originals, part of which is due to avoiding category theory. In particular, it is fascinating to consider material systems from the viewpoint of information flow, e.g., what events in one component tell about events in another. Similarly, even though many ideas about systems may be more general and elegant in CGST, this does not detract from the particular application of [3] to information flow; in particular, [21, 22] 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 [19], 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 be traced to these aspects. There also seems to be an implicit Aristotelean view that concepts are given as conjunctions of attribute, although this is inconsistent with the experimental results of Rosch and others in cognitive semantics (e.g., see [54, 30]).

My own views on the nature of information [27] are consistent with the Peircean pragmatism advocated by Sowa [70], but are more explicitly social (though not essentially more social) than Pierce. But philosophical debates often have little effect on applicability, and I have no doubt that ideas of IF, FCA, LOT, IFF and BT can have significant applications in many areas of information technology and cognitive science. Nevertheless, the enormous variety of data formats in use on the web, and the range of different logics used for ontologies, suggest that institutional formulations have advantages over less general approaches.

Finally, I wish to express my 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 schema management. In Giorgio Ghelli and Gosta Grahne, editors, *Proc. Database Programming Languages 2001*, pages 228–246. Springer, 2002. Lecture Notes in Computer Science, volume 2397.
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. Rod Burstall and Joseph Goguen. The semantics of Clear, a specification language. In Dines Bjorner, editor, *Proceedings, 1979 Copenhagen Winter School on Abstract Software Specification*, pages 292–332. Springer, 1980. Lecture Notes in Computer Science, Volume 86; based on unpublished notes handed out at the Symposium on Algebra and Applications, Stefan Banach Center, Warsaw, Poland, 1978.
7. 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.
8. Manuel Clavel, Steven Eker, Patrick Lincoln, and José Meseguer. Principles of Maude. In José Meseguer, editor, *Proceedings, First International Workshop on Rewriting Logic and its Applications*. Elsevier Science, 1996. Volume 4, *Electronic Notes in Theoretical Computer Science*.
9. Răzvan Diaconescu. Grothendieck institutions. *Applied Categorical Structures*, 10:383–402, 2002.
10. Răzvan Diaconescu. Institution-independent ultraproducts. *Fundamenta Informaticae*, 55(3–4):321–348, 2003.
11. Răzvan Diaconescu. Herbrand theorems in arbitrary institutions. *Information Processing Letters*, 90:29–37, 2004.
12. Răzvan Diaconescu. Interpolation in Grothendieck institutions. *Theoretical Computer Science*, 311:439–461, 2004.
13. 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.
14. Răzvan Diaconescu. An institution-independent proof of Craig interpolation theorem. *Studia Logica*, 77:59–79, 2002.
15. Gilles Fauconnier. *Mental Spaces: Aspects of Meaning Construction in Natural Language*. Bradford: MIT, 1985.
16. Gilles Fauconnier and Mark Turner. *The Way We Think*. Basic, 2002.
17. Sebastien Ferré and Olivier Ridoux. A logical generalization of formal concept analysis. In Guy Mineau and Bernhard Ganter, editors, *Proceedings, Int. Conf. Conceptual Structures*, pages 371–384. Springer, 2000. LNCS 1867.
18. José Luiz Fiadeiro. *Categories for Software Engineering*. Springer, 2004.
19. Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis: Mathematical Foundations*. Springer, 1997.
20. Peter Gärdenfors. *Conceptual Spaces: The Geometry of Thought*. Bradford, 2000.
21. Joseph Goguen. Mathematical representation of hierarchically organized systems. In E. Attinger, editor, *Global Systems Dynamics*, pages 112–128. S. Karger, 1971.
22. Joseph Goguen. Categorical foundations for general systems theory. In Franz Pichler and Robert Trappl, editors, *Advances in Cybernetics and Systems Research*, pages 121–130. Transcripta Books, 1973.
23. 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.
24. Joseph Goguen. A categorical manifesto. *Mathematical Structures in Computer Science*, 1(1):49–67, March 1991.
25. Joseph Goguen. Types as theories. In George Michael Reed, Andrew William Roscoe, and Ralph Wachter, editors, *Topology and Category Theory in Computer Science*, pages 357–390. Oxford, 1991.
26. Joseph Goguen. Sheaf semantics for concurrent interacting objects. *Mathematical Structures in Computer Science*, 11:159–191, 1992.

27. 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.
28. 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.
29. 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.
30. Joseph Goguen. What is a concept? In Frithjof Dau and Marie-Laure Mungier, editors, *Proceedings, 13th Conference on Conceptual Structures*, volume 3596 of *Lecture Notes in Artificial Intelligence*, pages 52–77. Springer, 2005. Kassel, Germany.
31. Joseph Goguen. Mathematical models of cognitive space and time. In Mitsu Okada and Daniel Andler, editors, *Reasoning and Cognition*. To appear, 2006.
32. Joseph Goguen. Ontotheology, ontology, and society. *Int. J. Human Computer Studies*, page to appear, 2006. Special issue on ontology, ed. by Christopher Brewster and Kieron O'Hara.
33. Joseph Goguen. Data, schema, ontology and logic integration. In Paulo Mateus and Walter Carnielli, editors, *Combinination of Logics*. Kluwer, to appear 2006.
34. 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.
35. 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.
36. 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.
37. Joseph Goguen and Fox Harrell. Foundations for active multimedia narrative: Semiotic spaces and structural blending, to appear. *Interaction Studies: Social Behaviour and Communication in Biological and Artificial Systems*.
38. 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.
39. 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.
40. Joseph Goguen and Grigore Roşu. Institution morphisms. *Formal Aspects of Computing*, 13:274–307, 2002.
41. Joseph Goguen, James Thatcher, Eric Wagner, and Jesse Wright. Initial algebra semantics and continuous algebras. *Journal of the Association for Computing Machinery*, 24(1):68–95, January 1977.
42. 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.

43. 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.
44. Robert Goldblatt. *Topoi, the Categorial Analysis of Logic.* North-Holland, 1979.
45. Nicolo Guarino and Christopher Welty. An overview of OntoClean. In Steffan Staab and Rudi Studer, editors, *Handbook on Ontologies*, pages 151–159. Springer, 2004.
46. Yuri Gurevich. Evolving algebra 1993: Lipari guide. In Egon Börger, editor, *Specification and Validation Methods*, pages 9–36. Oxford, 1995.
47. Stevan Harnad. The symbol grounding problem. *Physica D*, 42:335–346, 1990.
48. Edwin Hutchins. *Cognition in the Wild.* MIT, 1995.
49. 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.
50. Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *Knowledge Engineering Review*, 18(1):1–31, 2003.
51. 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.
52. Robert Kent. Formal or axiomatic semantics in the IFF, 2003. Available at `suo.ieee.org/IFF/work-in-progress/`.
53. 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.
54. George Lakoff and Mark Johnson. *Philosophy in the Flesh: The Embodied Mind and its Challenge to Western Thought.* Basic, 1999.
55. Joachim Lambek and Phil Scott. *Introduction to Higher Order Categorical Logic.* Cambridge, 1986. Cambridge Studies in Advanced Mathematics, Volume 7.
56. Saunders Mac Lane. *Categories for the Working Mathematician.* Springer, 1998. Second Edition.
57. Brian Mayoh. Galleries and institutions. Technical Report DAIMI PB-191, Aarhus University, 1985.
58. Robin Milner, Mads Tofte, Robert Harper, and David MacQueen. *The Definition of Standard ML (Revised).* MIT, 1997.
59. Till Mossakowski. Heterogeneous specification and the heterogeneous tool set, 2005. Habilitation thesis, University of Bremen.
60. Till Mossakowski, Joseph Goguen, Razvan Diaconescu, and Andrzej Tarlecki. What is a logic? In Jean-Yves Beziau, editor, *Logica Universalis*, pages 113–133. Birkhauser, 2005. Selected papers from the First World Conference on Universal Logic.
61. Till Mossakowski, Florian Rabe, Valeria De Paiva, Lutz Schröder, and Joseph Goguen. An institutional view on categorical logic and the Curry-Howard-Tait isomorphism, 2005. Submitted for publication.
62. Peter Mosses, editor. *CASL Reference Manual.* Springer, 2004. Lecture Notes in Computer Science, Volume 2960.
63. 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.
64. Pablo Neruda. *Windows that Open Inward (Images of Chile)*. White Pine, 1999. With photographs by Milton Rogovin.
65. Charles Saunders Peirce. *Collected Papers*. Harvard, 1965. In 6 volumes; see especially Volume 2: Elements of Logic.
66. Benjamin C. Pierce. *Basic Category Theory for Computer Scientists*. MIT, 1991.
67. Erhard Rahm and Philip Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.
68. Ferdinand de Saussure. *Course in General Linguistics*. Duckworth, 1976. Translated by Roy Harris.
69. Marco Schorlemmer and Yannis Kalfoglou. A channel-theoretic foundation for ontology coordination. In *Second Eruopean Workshop on Multi-Agent Systems*. 2004. Barcelona, Spain.
70. John Sowa. *Knowledge Representation: Logical, Philosophical and Computational Foundations*. Brooks/Coles, 2000.
71. 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.
72. Alfred Tarski. The semantic conception of truth. *Philos. Phenomenological Research*, 4:13–47, 1944.
73. Simon Thompson. *Type Theory and Functional Programming*. Addison-Wesley, 1991. Available online in postscript.
74. William Tracz. *Formal Specification of Parameterized Programs in* LILLEANNA. PhD thesis, Stanford University, 1997.
75. Rudolf Wille. Restructuring lattice theory: an approach based on hierarchies of concepts. In Ivan Rival, editor, *Ordered Sets*, pages 445–470. Reidel, 1982.

A Institutions and Categories

For those who are less than fully comfortable with category theory, we first explicate the notion of institution [35], without any explicit use category theory; this is followed by a brief exposition of some basic categorical concepts. Institutions axiomatize the possible kinds of theory and model that can be used for UCT. This is important because the examples we have worked out (e.g., here and in [31]) use several different kinds of logic and model, and other applications will no doubt require still other logics and models. First, we assume there are *contexts*[29], and *context morphisms*, which have a (partially defined) composition operation (if $\phi\colon\ C \to C'$ and $\psi\colon\ C' \to C''$ then $\phi;\psi\colon\ C \to C''$ denotes their composition) that is associative (i.e., $(\phi;\psi);\xi = \phi;(\psi;\xi)$ whenever these compositions are defined)

[29] Most institution literature uses the term "signature," but "context" is more appropriate for the wider range of applications now being explored for institutions, many of which exhibit similarity to Peirce's triadic semiotics [65], in which the context of a sign is important for its interpretation.

and has identities (i.e., given a context C, there is a context morphism 1_C such that $1_C; \phi = \phi$ and $\psi; 1_C = \psi$ whenever these compositions are defined)[30].

Next, we describe how context changes affect theories and models; this also is done axiomatically[31]: We assume that for each context C, there is a given set $Sen(C)$ of axioms (or sentences) for that context, and a given set[32] **$\mathbf{Mod}(C)$ of models for that context. If a is an axiom for context C and if $\phi\colon C \to C'$, then we let $\phi(x)$ or $Sen(\phi)(x)$ denote the result of translating x to the context C', and we assume that $1_C(x) = x$ and $\phi; \psi(x) = \psi(\phi(x))$ for $\psi\colon C' \to C''$. Similarly, given a model M' for C', let $\phi(M')$ or $\mathbf{Mod}(\phi)(\mathbf{M}')$ denote the C-model that results from the change of context (notice that this translation is "contravariant" rather than "covariant"), and we assume that $1_C(M') = M'$ and that $\psi; \phi(M') = \psi(\phi(M'))$. Finally, given $\phi\colon C \to C'$, we assume the *satisfaction condition*, that**

$$M' \models_{C'} \phi(x) \quad \text{iff} \quad \phi(M') \models_C x$$

for all C'-models M' and all C-axioms x. It seems that any logic (that has a notion of model) is an institution in this sense; a detailed argument for this is given in [60], which also shows that a great deal of metamathematics can be done at the level of institutions. (We mention that institutions can be generalized in various ways: one is to let the satisfaction relation take values other than true and false, e.g., in a lattice; another is to let model and sentence classes have more structure, such as a category; see [40] for details.)

A *C-theory* is a set of C-axioms. We now define a *triad* over an institution $\mathcal{I}$ to be a triple $(\mathcal{M}, C, T)$ where $\mathcal{M}$ is a set[33] **of C-models and T is a C-theory such that $\mathcal{M} \models_C T$. Then a *triad morphism* from $(\mathcal{M}, C, T)$ to $(\mathcal{M}', C', T')$ is a pair of maps $\Phi\colon T \to T'$ and $\Psi\colon \mathcal{M}' \to \mathcal{M}$ such that**

$$M' \models_{C'} \Phi(x) \quad \text{iff} \quad \Psi(M') \models_C x$$

for all models $M' \in \mathcal{M}'$ and all axioms $x \in T$; this condition is similar to the satisfaction condition. An equivalent form is

$$M' \models_{C'} \Phi(T) \quad \text{iff} \quad \Psi(M') \models_C T$$

for all M' in $\mathcal{M}'$, where $\Phi(T) = \{\Phi(x) \mid x \in T\}$. It is easy to see that every context morphism induces a triad morphism; but

[30] More technically, we are assuming a given *category* of contexts.
[31] More technically, we assume two *functors* on the category of contexts.
[32] More technically, a class, in the sense of Gödel-Bernays set theory.
[33] This is more general than the definition in the body of the paper.

triad morphisms are more general, as is shown by the skin color example. In the special of just unary predicates and just one context, triad morphisms degenerate to the *infomorphisms* of [**3**].

Categories represent structures, such as automata (with their homomorphisms), groups (with their homomorphisms), and vector spaces (with linear transformation). The description of contexts and their morphisms im the first paragraph actually constitutes a precise definiton of the category notion. For example, in Definition 1, Set denotes the category of set, and Cat denotes the the category of (small) categories. Similarly, functors represent constructions on structures, such as the formal language accepted by an automaton, or the lattice of normal subgroups of a group; and the description of how context changes sentences and models constitutes precise definitions for covariant and contravariant functors. Finally, natural transformations represent relations between functors, such as that one constructs less structured objects than another. The natural transformation a in Definition 4, express the relation between $Sen'(F(\Sigma))$ and $Sen(\Sigma)$, and similarly the natural transformation b expresses the relation between the two model functors; the vertical arrows in the diagram enforce the mutual consistency of the relationships a_Σ and the relationship b_Σ. Many other intuitions and examples can be found in [**24**], and many other places.

B 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 $F\colon \mathbf{I}^{op} \to \mathbf{Cat}$ (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 [33] 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[34] $F\colon \mathbf{I}^{op} \to \mathbf{Cat}$ 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 $f\colon\ i \to i'$ in $\mathbf{I}$ (i.e., $f\colon\ i' \to i$ in $\mathbf{I}^{op}$), and $h\colon\ A \to F(f)(A')$ in the category $F(i)$, noting that $F(f)\colon\ F(i') \to F(i)$; such morphisms have also been called "heteromorphisms," "cryptomorphisms" and several other things. Given also $(f', h')\colon\ (i', A') \to (i'', A'')$, define the composition $(f, h); (f', h')\colon\ (i, A) \to (i'', A'')$ 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 [71], but a better exposition appears in Section 2.1 of [40]. 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 for our applications.

The Grothendieck institution construction of [9] 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 flattened Grothendieck institution under suitable assumptions, e.g., Craig interpolation [12].

C A Novel Database Institution

This section illustrates the power of the triadic satisfaction relation of institutions with a novel database formalization. As

[34] Here, $\mathbf{I}^{op}$ is the **opposite category** of $\mathbf{I}$, obtained by reversing the direction of all arrows in $\mathbf{I}$.

in Section 3.6, we let I be many sorted Horn clause logic, so that relation symbols really denote relations, which are represented in models by appropriate sets of tuples, rather than by Boolean-valued binary functions. The novel twist is to take these database states as the objects of the category of signatures; it may seem strange to call them "signatures," but it makes good sense if we think of them as contexts that relate queries and answers. Although the term "context" is less suggestive of the original examples from logic, it has the advantage of better reflecting Peirce's semiotics. Let morphisms of database states be 3-tuples of inclusions of the three sets of tuples.

Example 8: 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 present 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 $i\colon \Sigma \to \Sigma'$, let $Mod(i)\colon Mod(\Sigma') \to Mod(\Sigma)$ 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, in which case 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 gives 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 [33], which provides a GUI to help users construct schema mappings, and which is currently being extended to ontology mappings.**