Technical Proposal

Office of Naval Research
2013 – 2016

BAA number: #13-001

Title of Proposal: Categorical Informatics

Identity of prime Offeror:

Massachusetts Institute of Technology
77 Massachusetts Avenue
Cambridge, MA 02139

Technical contact:

David I. Spivak
MIT Math Department
77 Massachusetts Ave, Bldg 2, Rm 236
Cambridge MA 02139
dspivak@gmail.com
(510) 684-6425

Administrative/business contact:

Ian C. Cariolo
Office of Sponsored Programs
Massachusetts Institute of Technology
Building NE18, Room 901
77 Massachusetts Avenue
Cambridge, MA 02139-4307
ph: 617-253-7260
fx: 617-253-4734
icariolo@mit.edu

Duration of Effort: 33 months: From 2013/02/01 to 2015/10/31.

Contents

Abstract

I am pursuing a mathematical framework for capturing the structure and dynamics of information. Throughout science and commerce, people are having tremendous difficulty making databases behave intelligently. The relational model is not sufficient for handling information as it is being generated and used today. Billions of dollars are being spent on translating between databases that cannot talk to each other, or on undoing human "cleverness" that arises when a data model does not conform to the phenomena being observed. We need to better understand how information should work.

For example, hierarchical structures are ubiquitous in organizations (such as businesses, universities, governments, militaries, etc.) and they are similarly ubiquitous in the phenomena we observe (such as atoms-molecules-cells-tissues-organs, or letter-word-sentence-paragraph-document). However, information management tools have not been produced that make this work in accordance with the dynamics of such phenomena. Similarly, while databases are constantly being updated as new information is found, we do not have a sufficient understanding for how this should be conceptualized. Finding an adequate model would greatly facilitate issues of data provenance and communication between databases. Finally, while relational databases currently require atomic fields, document databases (such as MongoDB) offer their users enormous flexibility by allowing non-atomic values.

Mathematics, and category theory in particular, has the power to formalize such notions and weave together a general framework that encompasses all the different approaches. For example, a simple extension of the relational database model, using *monads*, will allow for the inclusion of non-atomic data such as lists, sets, assurance estimates, data provenance, and so on. So-called hierarchical categories offer a rigorous approach to hierarchies, data aggregation, and parameter estimation, as used in science. And correspondences are well-suited for making sense of database updating.

For this grant I propose to formalize such commonly-needed information management issues. I will also supply the necessary theorems to relate different models, so that one can translate between them as seamlessly as possible. All this will serve as a theoretical basis on which real advances in computer science may be built.

Part I
Technical Approach and Justification

1 Topic of study: Mathematical structures for transparent, flexible, and reliable information exchange

It is becoming increasingly clear that human language, e.g. English prose, is not sufficiently robust to fully articulate an understanding of highly complex systems, nor therefore to tackle the associated set of large-scale problems, that exist today. Translating ideas—whether from lab to lab, from scientist to engineer, from company to auditor, from supervisor to employee, or from outgoing practitioner to incoming hire—is fraught with difficulties. Errors in translation can cause major disruptions in the workings of a company and affect large portions of the supply chain downstream.

Many such problems can be traced to a few common sources. First of all, the rules that govern complex systems are often opaque. This leads to fuzzy agreements, i.e. situations in which parties believe they are in agreement, but in fact are not, because a failure in the subtleties of their language hides differences in their assumptions. Secondly, the available tools for translating information from one domain-specific language (or *schema*) to another are difficult to use. This can lead a company to force all its members to adopt a single global schema that is not optimized for anyone. If one had available sufficient flexibility for translating information from one schema to another, each subgroup could view the information in a way that was optimized for the workings of that particular subgroup, and then different subgroups could interface through an established translation system. Third and finally, when translations do exist between domains, the translations are heuristically generated and their correctness is merely believed by humans whereas it should be checked by machines. Often the translations are performed by code that is difficult to reason about and almost impossible to reliably evolve through time.

The three themes in the above paragraph can be summarized as failures of transparency, flexibility, and reliability, and it is in these areas that mathematics has historically been and will continue to be useful. Mathematics is by far humanity's most reliable language. While it takes time to learn a given subject area (such as calculus), there is value in it: the community of those that do understand such a language can speak together in a way that allows very complex and subtle information to reliably pass between its members. For example, imagine trying to express that the graph of some economic indicator (say GDP) with respect to time currently has a high value, a negative slope, and a positive second derivative. Without calculus this might sound like "the economy is good but getting worse; however the rate at which it is getting worse is decreasing." Mathematics allows us to pass complex but structured information reliably, quantifiably, and as easily as possible.

While clarity and rigor have always been a part of the mathematical aesthetic, flexi-

bility in terms of comparing disparate fields has not. Historically, different fields within mathematics—geometry, algebra, analysis, logic—were separated into silos. Even though a single mathematician could often work within many different such silos, there was not sufficient language to integrate them into a coherent whole. Even within one discipline, like group theory, one would study the properties of individual groups rather than studying the interactions between groups.

As researchers like Galois and Poincaré began to see and apply the untapped potential that existed in the connections between fields, the need for a more universal language became apparent. In the 1940s Samuel Eilenberg and Saunders Mac Lane developed *category theory* for this purpose. Researchers from a variety of mathematical disciplines could phrase their issues in the common language of categories, forcing them to pay attention to the connections and interplay between objects rather than to single objects in isolation. Moreover, category theory offered a rigorous type of connection, called a functor, between different categories, by which insights and theorems from one field could be imported to another. For example, a functor connecting topology (shapes) and algebra (equations) allowed old theorems from algebra to be applied in topology, resulting in new theorems there. Category theory can similarly be used to connect different schematic representations of knowledge so that data and insights can be translated from one such representation scheme to another.

Since its inception, category theory has been applied to increasingly diverse disciplines, not just within mathematics but also in physics, computer science, and linguistics. Its organizational power is unmatched. Though sometimes denigrated as "just a convenient language", it has proven its worth countless times, and is arguably on par with Newton's calculus in terms of its potential to influence human society long-term. Furthermore, it is becoming increasingly apparent just how important and transformative a convenient language can be. In today's world of handheld computing devices, we see that convenience has a huge affect on life. The world of mathematics is not different in this regard: convenience is effective. Category theory is a discipline that promotes adherence to the principles that govern a given subject. The practitioner is led to clearly expose and articulate the ideas in the categorical framework, and the payoff is interoperability with every other field for which such work has been done.

For the past four years, benefiting immensely from an ONR grant (N000141010841), I have worked on bringing the categorical perspective to information science. My work has focused on databases because it is generally considered that databases are semantically omnipotent: if something can be known and expressed then it can be stored in a database. But there is a problem: if two different databases ostensibly contain information on the same subject (say two bank databases both contain information about people and their investments), that does not imply that it is easy to compare their held knowledge. My work has focused on this problem of communication, providing means by which information can be faithfully transferred from one world-view (database) to another.

While communication between different systems or subsystems has been a driving force

for my research, I have followed many paths in my pursuit of a category-theoretic understanding of information. In particular, I have formulated a database architecture framework that is both simple and powerful. As I explore the ramifications of using this architecture, I continue to find a proliferation of theoretical extensions that are both well-grounded in established mathematical theory and useful in practice. I will discuss some of these in Section 2.

My work has been met with appreciation both from within academia and without. In Section 2.2, I will discuss some of the collaborations that have been most mutually beneficial. I will also discuss a *transition* that occurred, whereby Microsoft has begun implementing some of my work for use in their semantic storage system. I have learned a great deal from my interactions with both academia and industry, and I plan to continue pursuing them.

Broadly speaking, my research attempts to mathematically express the fundamental dynamics of information, including how it is stored, updated, processed, and transferred. In Section 3, I will discuss three approaches to continuing this work, and in the remainder of this introduction I will summarize these three approaches.

In Section 3.1, I will discuss work in progress on hierarchical categories and aggregate properties. Hierarchical categories encode the idea that one object is composed of a structured system of other objects. As a biological example, a muscle is composed of a structured system of proteins, each of which is composed of a structured system of amino acids. As a mathematical example, a graph is composed of a structured system of vertices and edges. Once an object E is designated as a structured system of other objects, one can ask about the aggregate properties of E. For example, a function that assigns to each muscle its strength is an aggregate property—it is derived from the particular assembly of proteins that compose the muscle—and in the same way, a function that assigns to each graph its diameter is an aggregate property of the graph.

Hierarchical categories allow users of different levels (i.e. low-level workers and high-level managers) to interact with the same information at the scale appropriate to them. For example, details that are necessary for the operations of a lower-level employee are often simply distracting for his or her manager. For a different kind of example, hierarchical categories make it possible for an image to be catalogued at multiple levels, from pixels, to faces, to scenes. Hierarchical categories can also give a better theoretical underpinning for indexing and sorting, as I will describe in Section 3.1.3.

Next, in Section 3.2, I will discuss the category-theoretic notion of correspondences, which may be useful in better describing database updates. An update to a database is a change in the information content held there. Current systems do not adequately hold a record of such changes. A correspondence is a category-theoretic structure that records how something changes over the course of a discrete time step. Using correspondences we can keep track of the way data flows and changes over time, as well as how the schema evolves. Category theory allows us to articulate all this information coherently so that we can view how current data would look in the old schema, how old data would look in the

current schema, or how queries against the old database differ in their results from queries against the new one.

Finally in Section 3.3, I will discuss monads in databases. Monads have been quite successful in the theory of programming languages, allowing for increased expressivity in functional languages. Since the categorical model of databases is a close analogue and kindred spirit to the functional programming paradigm, it is no surprise that monads can greatly improve the expressivity of databases as well. In turn this will allow for significant compression in schematic representations of data. What takes four or more tables to describe without monads can be described in a single column using the monad formalism.

2 Previous work

Since 2008 I have working on similar goals. My work has generally centered around finding category-theoretic or algebro-topological formulations for the structure and operation of databases. The work has generated a healthy amount of interest, both from within academia and from without. Below I will describe the papers I have published and then the contacts I have made and maintain.

2.1 Papers

A list of papers I have published and talks I have given on the subject of categorical information theory can be found in my CV. Below I will outline a few broad topics.

2.1.1 Simplicial databases

A database, as formulated by Codd, is a set of relations. This mathematical formalism provided a unified language and toolset that together led to major advances in the field. However, his model has serious weaknesses. For example, databases involve interacting tables, and Codd's notion treats this fact as an afterthought. Thus the model is not fit for understanding schemas holistically, which is necessary when comparing them. My intention is to improve communication between disparate entities, and hence it is imperative that schemas be nicely comparable. To that end I developed simplicial databases,[1] which are geometric objects that capture the interrelationship between various tables in a schema.

The geometric nature of these databases has valuable consequences. For example, a simplicial schema often looks like what it is intended to model: one can model one-way airplane tickets using a line segment (1-simplex) between two points, and then simply glue this object to itself, going backwards, to model round-trip tickets. The mathematics follows suit. Also, one can draw paths through a simplicial database and information will be carried from the start point to the end point of the path. More precisely, any such path

[1] D.I. Spivak. (2009) "Simplicial databases". ePrint available http://arxiv.org/abs/0904.2012. (Supported by ONR grant N000140910466.)

entails a query against the database in which data is entered into the table at the source of the path and data is output from the table at the target of the path. This was discussed in a second paper.[2]

Several mathematicians saw the value in this approach and invited me to speak at various universities and conferences. Some high-level people in industry (including a vice president at Amgen and a director at Johnson & Johnson) also appreciated the work and I have worked closely with members of both companies.

2.1.2 Databases as categories

At some point I began to recognize that foreign keys (functional connections between tables) were essential, and I worked to make them more prominent in the theory. Realizing that even data columns could be considered as foreign key columns allowed me to say that a database schema *is* a category $\mathcal{C}$, and that an instance *is* a functor $I\colon \mathcal{C} \to \mathbf{Set}$. This greatly simplified matters, and the idea has been very fruitful.

For one thing, it allowed a good definition of schema integration and data migration. I published a paper on this in 2012.[3] For another, taking the Grothendieck construction of an instance converts any relational database (collection of tables) into an RDF triple store, a common format for storing semi-structured data. Formulating functorial connections between the viewpoints of multiple users in "the cloud" will create an atlas of knowledge, so that information can be more coherently shared between them. Alignment of concepts can be guaranteed without requiring a unified naming convention. As an added benefit, this perspective allows one to view queries and constraints in terms of lifting problems, in the sense of homotopy theory, so that theorems from algebraic topology can be applied to make sense of data. I submitted a paper on this subject in 2012 as well.[4]

The simplicity of the categorical approach leads to flexibility in the sense that it is easy to extend the theory. For example, I was able to use monads in much the same way they are used in programming language theory to relax the atomicity requirement for data in a database. Using the Kleisli construction allows lists, sets, probability distributions, accuracy estimates, and other types of additional information to occur in each cell of a database. A paper on this subject is complete and will be submitted to the PODS database conference this year.[5]

[2] D.I. Spivak. (2010) "Table manipulation in simplicial databases". ePrint available http://arxiv.org/abs/1003.2682. (Supported by ONR grant N000140910466.)

[3] D.I. Spivak. (2012) "Functorial data migration". *Information and Computation.* (Supported by ONR grant N000141010841.)

[4] D.I. Spivak (2012) "Database queries and constraints via lifting problems". Submitted to *Mathematical structures in computer science.* (Supported by ONR grant N000141010841.)

[5] D.I. Spivak. (2012) "Kleisli database instances". ePrint available, http://arxiv.org/abs/1209.1011. (Supported by ONR grant N000141010841.)

2.1.3 Ologs as outreach

Improving communication between related entities is a core motivation of my research, and as such I felt it was important to reach out to other disciplines in order to improve communication between our respective academic fields. To that end I am teaching a course at MIT in Spring 2013 called *Category Theory for Scientists*, where I will discuss the basic tenants of category theory, including functors, natural transformations, adjoints, monads, and operads. But I will do so in a very down-to-earth way that includes real-world examples.

The same motivation led me to put a linguistic twist on categories, so that they could be drawn as human readable box-and-arrow diagrams. I called these structures *ologs* and wrote a paper on the subject that was published in 2012.[6] I have received a good deal of appreciation for that paper.

The olog concept also led to an unexpected collaboration with a distinguished MIT professor in materials science, named Markus Buehler. Together we have published four papers in the last 2 years.[7][8][9][10]

2.2 Contacts and collaborations

2.2.1 Math faculty and postdocs

I spoke about my work on categorical databases at Johns Hopkins University in late 2011. This talk was enthusiastically received by professor Jack Morava, a well-known algebraic topologist. He wrote a paper called *Theories of anything* that connects my work to classical concepts in representation theory and Thom's catastrophe theory. Morava explains that symmetry breaking in physics, chemistry, and throughout science, is on one hand a common generator of scientific data (e.g. the melting point of various substances comes from symmetry breaking). On another hand, it induces a category, i.e. a database schema, on which such scientific data naturally lives. This is a profound and potentially very powerful idea that should be pursued.

My work on ologs and functorial data migration also inspired Scott Morrison (an ARC research fellow and senior lecturer at the Mathematical Sciences Institute of the Australian

[6] D.I. Spivak, R.E. Kent. (2012) "Ologs: a category-theoretic foundation for knowledge representation". *PLoS ONE*. (Supported by ONR grant N000141010841.)

[7] D.I. Spivak, T. Giesa, E. Wood, M.J. Buehler. (2011) "category-theoretic analysis of hierarchical protein materials and social networks". *PLoS ONE*. (Supported by ONR grant N000141010841.)

[8] T. Giesa, D.I. Spivak, M.J. Buehler. (2011) "Reoccurring patterns in hierarchical protein materials and music: the power of analogies". *BioNanoScience*.

[9] T. Giesa, D.I. Spivak, M.J. Buehler. (2012) "Category theory based solution for the building block replacement problem in materials design." *Advanced Engineering Materials*. (Supported by ONR grant N000141010841.)

[10] J.Y. Wong, J. McDonald, M. Taylor-Pinney, D.I. Spivak, D.L. Kaplan, M.J. Buehler. (2012) "Materials by design: merging proteins and music." *Nano Today*.

National University) to write a code base for a category-theoretic database engine. The current incarnation of this work can be found at http://www.categoricaldata.net.

I am involved in collaborations (at various stages) with various postdocs and research fellows in mathematics, including David Gepner, Nat Stapleton, Henrik Forssell, and Mathieu Anel.

2.2.2 Non-math professors

I am working with Adam Chlipala, an expert in the mathematical proof assistant called Coq. We are co-advising an excellent MIT undergraduate student named Jason Gross, who is implementing my categorical database ideas in the Coq proof assistant. A code base is available online: https://bitbucket.org/JasonGross/catdb.

I have also had a productive collaboration (four papers) with Markus Buehler's lab. Buehler works on materials science, for example investigating the structural relationships that give hierarchical protein materials such as spider silk their astounding functional properties. He was interested in collaborating with me because he needs to accurately describe materials, both their structure and their function, and this cannot be adequately done in English prose. Ologs allow for extremely precise descriptions. In an effort to capture the hierarchies his lab was seeing, I invented hierarchical ologs, which I hope to write about in mathematical detail in the near future.

2.2.3 Industry

It is important to me that my work relate to real informatic issues in the world. If I am studying information and communication, then my success is measured by whether people with experience in those fields find my work useful. Much of my research has grown out of conversations with Peter Gates (Johnson & Johnson) and Dave Balaban (Amgen, AMS Fellow). For example, they have gotten across to me the importance of aggregate functions. I have also been invited three times for 2-day sessions at Amgen, during which we discussed my research. These meetings made clear to me the importance of ETL, which later became a major selling point for my work on functorial data migration.

2.3 Transition to industry

In each meeting with the informatics group at Amgen, we were joined by Allen Brown of Microsoft. He found the olog concept to be intuitive and useful, and in fact has begun to implement ologs in the upcoming release of the Microsoft Semantic Storage System (SSS), which is expected to be used by Amgen. He also considers the monadic database instances work to be "dynamite" and is employing it also in the SSS.

I am quite pleased with this transition from academic research to industry-level software. I take it as evidence that my research is grounded in reality, as opposed to being stuck in the ivory tower. Interestingly, many of my colleagues at MIT still consider what

I am doing to be pure math because it adheres to the rigor and aesthetic that pure math demands. My expertise is in listening to what is being done and what is needed in the real world and in turn, offering appropriate structures and tools that are well-known to be effective in mathematical research. It is gratifying that these same tools are found to be useful by industry.

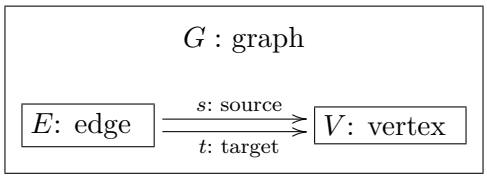
I hope and believe that this will be the first of many instances in which my work is transitioned from academia into practice.

3 Proposed project: 2013 – 2016

This proposal includes only one task: to do basic mathematical research on the fundamental structure and dynamics of information and communication. In what follows I will outline three proposed themes for this work: hierarchical categories (Section 3.1), correspondences for updates (Section 3.2), and monadic databases (Section 3.3).

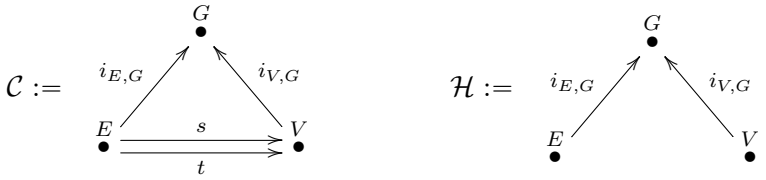
3.1 Hierarchical categories and aggregate properties

Hierarchies are ubiquitous in our current scientific understanding of reality. It seems that objects are often comprised of more basic objects that fit together in some specified way. We will call the larger object an *assembly* and the basic objects that comprise it *the constituency set* of the assembly. In this language, a graph is an assembly of vertices and edges (each edge having a source and target vertex) and these vertices and edges are the constituency set of the graph. We can express all this as follows:



(1)

The notation of hierarchical categories is richer than it first appears. Diagram (1) secretly depicts a category with three objects, four arrows, and two commutative diagrams, isomorphic to the category $\mathcal{C}$ on the left



(2)

with $i_{V,G} \circ s = i_{V,G} \circ t = i_{E,G}$. However, Diagram (1) encodes more structure than just its underlying category as on the left of Diagram (2); it also encodes constituency information

as on the right of (2). In other words, the fact that boxes E and V are drawn inside of box G implies the existence of (invisible) *structure maps* $i_{E,G}$ and $i_{V,G}$, and the fact that the source and target arrows are drawn inside of box G imply that s and t commute with the structure maps.

The point is that Diagram (1) is a pictorial representation of the information $(\mathcal{C}, \mathcal{H})$ found in Diagram (2); these satisfy the following definition:

Definition 3.1.1. A *hierarchical category* is a pair $(\mathcal{C}, \mathcal{H})$ where $\mathcal{C}$ is a category and $\mathcal{H} \subseteq \mathcal{C}$ is a subcategory such that

- $\mathrm{Ob}(\mathcal{H}) = \mathrm{Ob}(\mathcal{C})$, and
- $\mathcal{H}$ has the structure of a forest (a disjoint union of trees).

If there is a morphism $c \to d$ in $\mathcal{H}$ we denote it $i_{c,d}$ and call it a *structure morphism*; we sometimes write $c \leqslant d$ or $i_{c,d} \colon c \leqslant d$.

Let $f \colon c \to c'$ be a morphism in $\mathcal{C}$ and let $d \in \mathrm{Ob}(\mathcal{C})$ be an object. We say that f is *a constituent morphism of* d if $c, c' \leqslant d$ and $i_{c,d} = i_{c',d} \circ f$. We say that c is a *constituent object* of d if $c \leqslant d$ (or equivalently if id_c is a constituent morphism of d). For any object $d \in \mathrm{Ob}(\mathcal{C})$ the *constituent category of* d, denoted $\mathcal{C}_{\leqslant d}$, is the category with constituent objects and constituent morphisms as defined above.

Example 3.1.2. In the graph example (2) above, the constituent category $\mathcal{C}_{\leqslant E}$ of E is the one-object category $\{E\}$, the constituent category $\mathcal{C}_{\leqslant V}$ of V is just $\{V\}$, and the constituent category $\mathcal{C}_{\leqslant G}$ of G is all of $\mathcal{C}$. This aligns with the drawing in (1): the boxes surrounding E and V contain nothing but themselves, whereas the box surrounding G contains everything in sight.

Use of hierarchical categories Definition 3.1.1 emerged as a result of two separate forces upon my research: the need to model hierarchical objects in materials science and the need to understand aggregation functions in database theory. A preliminary understanding of how hierarchical categories should work was sufficient to produce a published paper in materials science.[11] Understanding aggregation functions took a good deal of additional thought, but I now have a fairly settled account. The latter was also applied in a published paper,[12] but a more detailed mathematical description will be the subject of a paper I plan to write in the upcoming year.

I will now give a brief review of what aggregate properties are supposed to model in the real world. The pressure of a gas on its container is an aggregate property: the gas is actually a collection of smaller particles, and the principles of statistical thermodynamics

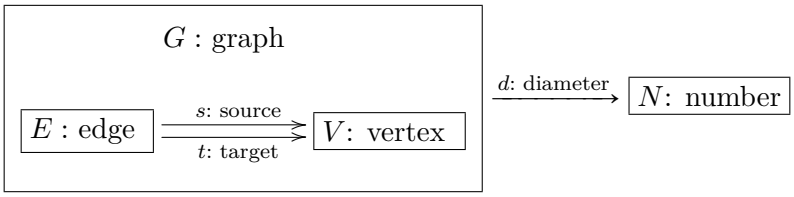
[11] Giesa, Spivak, Buehler. (2011) "Reoccurring patterns in hierarchical protein materials and music: The power of analogies." *BioNanoScience*.

[12] Giesa, Spivak, Buehler. (2012) "Category theory based solution for the building block replacement problem in materials design." *Advanced Engineering Materials*

tell us that pressure in a region can be considered as a count of the number of smaller particles that hit that region in unit time. Pressure does not relate to the individual particles in the gas, but to the whole collection. Similarly the diameter of a graph (maximum distance from one vertex to another) is an aggregate property of the graph.

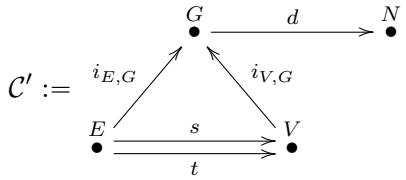
This notion of aggregates is ubiquitous in science. For example fitting a curve to existing data or estimating parameters of the data involves considering the data as a whole; i.e. these are aggregate properties. In informatics, one commonly wants reports, such as counts, averages, sums, etc., and these are again aggregate properties. One should make the distinction between aggregates and simple queries. The difference is that in response to a simple query (such as "tell me all the paths of length 4 in the graph"), results may be put forth before all the data is amassed and considered. That is, the query engine can return a result after a few steps and then continue its search. In contrast, aggregate properties are holistic: one cannot count the number of balls in a canister or determine the diameter of a graph without considering all the data.

We extend our hierarchical category from Diagram (1) by adding an aggregate property:



(3)

Here a graph is an assembly of edges and vertices, and each such assembly can be assigned a number as its diameter. Note that as a (non-hierarchical) category, Diagram (3) would look like this



and it would be unclear how the diameter d of a graph was related to its pattern of edges and vertices. But with the additional information held by the hierarchical structure, we provide ourselves the language to discuss assemblies and constituents, i.e. that a graph is an assembly of vertices and edges. Now we can encode the formula for diameter *in terms of* the constituent category of G using an underlying type system T on $(\mathcal{C}, \mathcal{H})$.

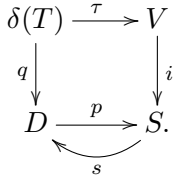
Part of the proposed grant will be dedicated to carefully writing down the foundations of hierarchical categories and using them in various applications, both mathematical and scientific.

3.1.3 Indexing via hierarchical inheritance

Database management systems make extensive use of indexes to speed up query performance. Before I explain how hierarchical categories may be useful in indexing, I will roughly explain what indexing is. The idea is familiar from every day life. For example, suppose a customer of a hardware store is looking for a particular kind of item (e.g. a hammer). One method of finding it is to search up and down every aisle in the store. But given an index, the customer can first determine the aisle that this kind of item is in and then search only that aisle. Similarly in databases we may need to find a record with a particular value in some table, and an index will speed up that search. We formalize the indexing concept as follows.

Suppose that D is a linearly ordered set, representing the ordered set of blocks on a disk. A disk index is a projection $p\colon D \to S$ and section $s\colon S \to D$ such that $p \circ s = \mathrm{id}_S$ and, for all $d \in D$, we have $s(p(d)) \leqslant d$). In other words the blocks are partitioned into regions (one for each index value) and s sends an index value to the first block in its region. We call the triple (S, p, s) an index of disk D.

Given a database instance δ and a table T in the schema, we have a set $\delta(T)$ of records in T and a function $\tau\colon \delta(T) \to V$ that assigns a relevant value to the records. It is these values that will be indexed. The data is stored on disk as $q\colon \delta(T) \to D$. An (S, p, s)-index of table T is a function $i\colon V \to S$ such that $i \circ \tau = p \circ q$ and $p \circ s = \mathrm{id}_S$ in following diagram:



In the hardware store analogy above, $\delta(T)$ represents a set of physical items (e.g. Hammer 1033) in the store, V represents the set of item-kinds (e.g. hammer), D represents the set of physical locations in the store, and S represents the set of aisles (each of which has a front end in D). One is searching for an element of kind $v \in V$ within the set $\delta(T)$, i.e. he is searching for some $x \in \delta(T)$ with $\tau(x) = v$. The element $i(v) \in S$ is the index value for v (i.e. the hammer aisle), so we know the region $p^{-1}(i(v))$ of the store in which any successful x will be. This region is linearly ordered and $s(i(v))$ is its minimal position. We check each item $x \in (p \circ q)^{-1}(i(v))$ to see if $\tau(x) = v$.

The goal is to represent this situation category-theoretically. The idea of ordering rows in each table may lead one to the idea that instead of a functor $\delta\colon \mathcal{C} \to \mathbf{Set}$, we should consider an instance of $\mathcal{C}$ to be a functor $\mathcal{C} \to \mathbf{Lin}$, where $\mathbf{Lin}$ is the category of linear orders. However this will rarely work because it forces every column in the database to preserve ordering. A database instance with this property would be very strange; e.g. if T were a table with first and last name columns, then we would be requiring that both columns must be in simultaneous alphabetical order.

That idea obviously fails. The question then is how one can specify which columns of a table should preserve the choice of linear order. Hierarchical categories allow one to specify grouping relationships among objects. If an aisle is a set of locations, then it is natural to order locations by their aisle, so it would not be strange if the map $i\colon V \to S$ were enforced as preserving linear order.

Thus we can make the following definition. Let $(\mathcal{C}, \mathcal{H})$ be a hierarchical category and $\delta\colon \mathcal{C} \to \mathbf{Set}$ an instance on $\mathcal{C}$. An *indexing of δ subordinate to the hierarchy* is a linear ordering of $\delta(c)$ for each $c \in \mathrm{Ob}(\mathcal{C})$ such that for each structure map $i_{c,c'} \in \mathcal{H}$ the map $\delta(i_{c,c'})$ preserves the order.

3.1.4 Other applications of hierarchical categories

Consider again the olog given in (3). A higher-level employee may only want to see the diameter of graphs and not care about the actual set of edges and vertices in each graph, whereas a lower-level employee needs to see these details. The hierarchical construction provides a ready-made abstraction barrier: higher-level employees can be shown only higher-level objects.

Hierarchical categories may also be useful for content-based image retrieval. An image contains information at multiple hierarchical levels, and different applications and searches demand access to information at different levels. Hierarchical categories are designed to encode and keep straight these structural relationships.

3.2 Correspondences for updates

Databases are constantly being updated as time progresses; for example rows may be added or deleted, split or merged, and values (such as a person's address) can be changed. In older databases, these actions simply changed the state of the world, as known by the database, without keeping track of the history. However, many factors, such as the need to undo human error, the accountability demands of internal and external auditing requirements, and the desire to analyze trends in time made that simplistic approach untenable. Data provenance is the demand to keep track of the old data, how it connects to the new data, and what caused the change.

A good theory of updates, therefore, must keep track of how data persists and changes across time. The category-theoretic notion of correspondences will model the database as a time-series of instances, akin to frames of a movie except with much more structure. For example, the pixels in a movie at frame t are not connected in any way to the pixels at frame $t + 1$. Similarly the fact that there is a human in the picture at frame t and the same human at frame $t + 1$ is not important when simply projecting images on a screen. But informationally, it is crucial to understand that the two frames include a depiction of the same human. Correspondences allow one to track individual entities as they progress through the time-series.

3.2.1 Updates and data provenance

Let $\mathcal{C}$ be a database schema, i.e. a category. Then in [Spivak, *Functorial Data Migration*] I define the category of instances on $\mathcal{C}$ to be $\mathcal{C}$–**Set**, the category of functors $\mathcal{C} \to$ **Set**. This setup is nice in the following ways:

- $\mathcal{C}$–**Set** is a topos for any schema $\mathcal{C}$, i.e. it is as nice a category as one could want;
- given a functor $F\colon \mathcal{C} \to \mathcal{D}$, the pullback and pushforward functors capture classical database concepts such as Project, Union, Join;
- the Grothendieck construction takes a $\mathcal{C}$-set and returns the category of RDF triples, and morphisms of instances on $\mathcal{C}$ return functors between corresponding RDF triple stores.

Thus the set of morphisms in $\mathcal{C}$–**Set**, i.e. natural transformations $\delta \to \epsilon$, is clearly a good choice in certain respects. And yet, these morphisms do not seem to fully capture the semantic meaning of database updates. Natural transformations do nicely capture insertions and merges (also known as deduplication). But in order to capture deletions one needs to turn the natural transformations around: if $\delta \to \epsilon$ is injective and corresponds to ϵ being an insertion of some data into δ, then we can consider δ as being a *deletion* of some data from ϵ.

A typical update could be considered as a sequence of deletions and insertions, and hence as a zigzag of natural transformations. However this leaves open the question of when two updates (zigzags) are considered equivalent. For example, if we delete a record and then re-insert it, should the result be considered isomorphic to the original? If so, our category of updates becomes a contractible groupoid, i.e. it loses all structure. Moreover, changing one field (such as first name) in a record would be understood as deleting the entire record and then re-inserting a nearly identical record with only one field changed.

There is something inadequate about this model for updates. Updates are more complex than just deletes and inserts. Data provenance demands that we see where and when records were changed: if I change my name to Eugene, the database should not delete my whole record and insert a new one with almost identical information because all provenance will be lost. Instead, it should connect my old identifier with my new identifier and connect almost all my old data with my new data (the exception being of course my name).

After much thought it seems that what is needed is quite obvious: a mathematical model for the structure of time series. At each time t we have a database state δ_t, and the state at time t should be connected to the state at time $t+1$. What kind of connection is appropriate? As above, we need to connect objects from time t to objects at time $t+1$; in typical categorical fashion we do so with arrows. We assume nothing goes backwards in time, so all of our arrows emanate from an object existing at some time t and hit some object at time $t+1$. An object at time t may die, so not everything at time t needs to have an arrow emanating from it. An object at time $t+1$ may be born, so not everything

at time $t+1$ needs have an arrow hitting it. And an object at time t may split (like cell mitosis) to become many things at time $t+1$.

Together, this suggests a correspondence, i.e. a category over [1] with δ_t and δ_{t+1} as the fibers over 0 and 1 respectively. The history of a database would thus be a category over

$$[n] = \boxed{\overset{0}{\bullet} \longrightarrow \overset{1}{\bullet} \longrightarrow \cdots \longrightarrow \overset{n}{\bullet}}$$

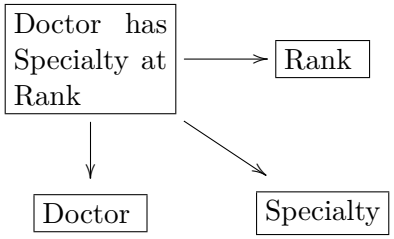
for some $n \in \mathbb{N}$. The classical requirement of *serializability* for database transactions simply states that any update that occurs between time t and time $t+k$ factors through the k intermediate time steps. In the language of [Lurie, HTT] this seems to line up precisely with the condition that $G\colon \mathcal{A} \to [n]$ be a coCartesian fibration.

For the proposed grant I will study how updates should work and attempt to formulate a category (probably in terms of coCartesian fibrations) that is would serve as a useful model for database updates. I will investigate basic properties of this category and attempt to prove theorems that would be useful and interesting in the database community. This work may also have applications to using database formalisms to study physical phenomena that depend on time. Such research may give a new mathematical foundation for temporal databases.

3.3 Monads in databases

Recently I have been working on using monads in databases and the results have been quite interesting. One can greatly increase the expressive power of a database by allowing a field to contain non-atomic data such as lists, sets, or probability distributions. Relational databases, as defined by Codd, are mere "sets of relations". From a category-theoretic standpoint this model appears quite rigid and it is no surprise that it cannot handle non-atomic data. The categorical framework is far more flexible and works well with the monadic concept.

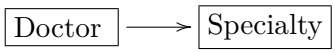
By incorporating monads within databases, the amount of ancillary data needed to describe an observation is drastically reduced. For example, suppose we want to associate to each doctor a ranked list of specialties. The "specialties" column of the doctor table should have *non-atomic data*, but this is not allowed in relational databases. In Codd's model the database architect is forced to use a somewhat arcane work-around, often involving multiple tables and columns, as below:



Doctor has Specialty at Rank			
ID	**Doctor**	**Specialty**	**Rank**
q101	Elaine Wu	Pediatrics	1
q102	Elaine Wu	Neurology	2
q103	Elaine Wu	Vision	3
q104	Phil Block	Internal	1
q105	Phil Block	GI	2

Further problems emerge in that the database has to check that the ranks are in order; i.e. if row q102 is deleted then the rank of Elaine Wu's vision specialization becomes 2, and the database has to manage that update.

A more intuitive, accurate, and schematically smaller approach is to simply use lists. We replace the above table with



Doctor	
ID	**Specialty**
Elaine Wu	[Pediatrics,Neurology,Vision]
Phil Block	[Internal, GI]

The non-atomicity of the data is handled by the List monad. Other monads can handle probability distributions on data, as well as other kinds of interesting non-atomicity. This topic is discussed in a recent paper (see footnote 5).

Monadic databases are worthy of further study. We need to develop intuition for the kinds of real-world queries one would want to pose to a monadic database and then express them mathematically. For example, one may want to combine (take the union of) the set of specialties associated to each doctor in a group. Even though this may feel like a simple task, it must be understood wholly within the categorical framework in order for it to work well with other concepts.

Another important direction for research is to understand how monadic databases operate under change of schema. The category **Monad** of finitary monads on **Set** is complete and cocomplete. This means that to any functor $F\colon \mathcal{C} \to \mathbf{Set}$ there are monad migration functors $F^*\colon \mathcal{D}\text{–}\mathbf{Monad} \to \mathcal{C}\text{–}\mathbf{Monad}$ and $F_!, F_*\colon \mathcal{C}\text{–}\mathbf{Monad} \to \mathcal{D}\text{–}\mathbf{Monad}$. It would be valuable to make real-world sense of what these functors are doing. I have not even approached this yet. It would also be useful to understand how the associated Kleisli states would interact, because Kleisli categories are generally neither complete nor cocomplete. On the other hand, the category of algebras on $\top \in \mathbf{Monad}$ is complete and cocomplete. Since the Kleisli category $\mathbf{Kls}(\top)$ is equivalent to the category of free algebras on $\top$, this suggests a mild generalization that could have nice formal properties.

During work on using monads in databases, a surprising coincidence arose. Namely, if $\mathcal{L}oop$ is the category depicted below

$$\mathcal{L}oop := \boxed{\begin{array}{c} f \\ \circlearrowright \\ s \bullet \end{array}}$$

then for different choices of monads $\top$, the set of Kleisli $\top$-instances on $\mathcal{L}oop$ can be

interpreted in terms of classical mathematical subject areas.

Classical mathematical subject area	**monad** $\top$
Discrete dynamical systems	$\mathrm{id}_{\mathbf{Set}}$
Graphs	Multiset
Markov chains	Dist
Finite state automata	$X \mapsto X^U$
Turing machines	$\mathrm{Tur}^{\{0,1\}}_{\{L,R,W_0,W_1\}}$
"Jordan Canonical Form" (vector spaces with endomorphism)	Vect
Multigraphs	$\mathbb{N}[-]$

(4)

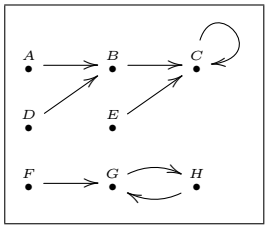
Example 3.3.1. Without monads, the $\mathcal{L}oop$ schema encodes discrete dynamical systems (DDSs). A DDS consists of a set of states and a function that takes each state to its position at the next moment. A functor $\delta\colon \mathcal{L}oop \to \mathbf{Set}$ can be written as a table with an ID column and an f-column, as below.

$\delta :=$

s	
ID	**f**
A	B
B	C
C	C
D	B
E	C
F	G
G	H
H	G

pictured

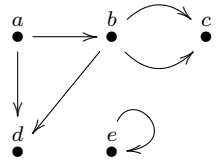
$\int \delta =$



Example 3.3.2. With monads, we can relax the atomicity requirement for cells in the f column. For example, using the multi-set monad, we can put a whole multi-set (unordered list) of values in that cell. This allows a state to point to point to more than one "next" state.

$\delta :=$

s	
ID	**f**
a	{b,d}
b	{c,c,d}
c	{}
d	{}
e	{e}



(5)

The use of monads in databases remains to be adequately explored. For example, in [Leinster, *Higher Operads, Higher Categories*], Leinster uses monads to construct multicategories of various types, and both graphs and multigraphs play a role there. I will explore possible connections between my work and his. I will also explore whether the above coincidence (4) is part of a larger pattern. For example, perhaps fixed point cycles (these

are precisely the eigenvectors when $\top = \mathbf{Vect}$) are important invariants in all of the above examples.

Finally, using monads in databases brings up an interesting extension that has not, until this point, seemed necessary: the use of higher categories. I have hesitated to use higher categories in my work on databases because I did not want to make the theory unnecessarily abstruse. However, once one begins working with values that are collections (e.g. sets, multi-sets, lists) of other values, an ordering relation immediately arises. For example if we are using the power set monad, then instead of imposing a commutative diagram as to the left in (6),

(6)

one could impose that for every element of A, the set of elements in C obtained by $g \circ f$ is a subset of that obtained by h, i.e. a natural transformation diagram as to the right in (6).

The topic of monads in databases is quite new and I anticipate that continuing to pursue it will lead to interesting and probably unexpected results.

Part II
Future Naval Relevance

Complex systems have a pressing need to process information efficiently, and the Navy is no exception. A group only functions as a unit when all the parts are in good communication. Data from one part of the structure needs to be transferable to and understandable by other parts. Short-term solutions, such as creating links between data sets in an ad hoc manner, will inevitably fail as the system evolves and becomes more complex.

The above work on hierarchical categories and correspondences may have applications to content-based image retrieval (CBIR). Hierarchical categories organizes different levels of classification, and correspondences track individual entities as their attributes change over time. The work on monads will have value in compressing databases, as well as in simplifying and clarifying the architecture.

Category theory is the language of structure, and is a powerful tool for organizing information so that it is flexible, transferable, and scalable. Disparate research efforts by a variety of labs can be meaningfully compared and integrated if they are expressed in the language of category theory.

Part III
Project Schedule and Milestones

I will continue to produce papers on information and communication from a category-theoretic perspective. I will work with other academic researchers and technicians from industry to hone the approaches. I will work with a programmer to create a user interface by which these ideas may be implemented, which will lead to quicker transition of academic research into working software.

Part IV
Reports

The following data deliverables will be provided:

- Annual Technical and Financial Progress Reports
- Final report

Part V
Management approach

I will be working directly with a programmer to implement the categorical database concept. This will include frequent emailing back and forth as well as biweekly meetings for progress updates.

MIT provides basic office space and library services. The campus at the Massachusetts Institute of Technology is networked by both a wired 100/1000Mbps Ethernet LAN and campus-wide 802.11b/g wireless networks.

Part VI
Current Project and Pending Proposals

I have an existing grant from the ONR, detailed below; it is set to end 2013/06/01. I am currently applying for a 5-year grant from AFOSR.

1 Existing grant: N000141010841

Title of Proposal: Categorical information theory.

Summary: Using category theory to understand the basic dynamics of information via databases and ontologies.

Source and amount of funding: Office of Naval Research. Annual total cost: $120,000. Percentage effort devoted to this project: 100%.

Identity of prime Offeror: David I. Spivak

List of subcontractors: None.

Technical contact:
David I. Spivak
MIT Math Department
77 Massachusetts Ave, Bldg 2, Rm 236
Cambridge MA 02139
dspivak@gmail.com
(510) 684-6425

Administrative/business contact :
Ian C. Cariolo

Office of Sponsored Programs
Massachusetts Institute of Technology
Building NE18, Room 901
77 Massachusetts Avenue
Cambridge, MA 02139-4307
ph: 617-253-7260
fx: 617-253-4734
icariolo@mit.edu

Period of performance: Period 1: 2010/06/01 – 2010/09/30.
Period 2: 2010/10/01 – 2011/09/30.
Period 3: 2011/10/01 – 2012/09/30.
Period 4: 2012/10/01 – 2013/05/31

Total award amount and person-months : $360,000 for 36 person months.

Relation to present proposal: The present proposal is a renewal of the existing grant; it sets out from the new and unforeseen results obtained while working on the existing grant.

2 Concurrent grant application: AFOSR

Title of Proposal: Categorical approach to entity interaction

Summary: Using category theory and algebraic topology to understand interaction between agents.

Source and amount of funding: AFOSR. Annual total cost: $180,000. Percentage effort devoted to this project: 74%.

Identity of prime Offeror: David I. Spivak

List of subcontractors: None.

Technical contact:
David I. Spivak
MIT Math Department
77 Massachusetts Ave, Bldg 2, Rm 236
Cambridge MA 02139
dspivak@gmail.com
(510) 684-6425

Administrative/business contact :
Ian C. Cariolo

Office of Sponsored Programs
Massachusetts Institute of Technology
Building NE18, Room 901
77 Massachusetts Avenue
Cambridge, MA 02139-4307
ph: 617-253-7260
fx: 617-253-4734
icariolo@mit.edu

Period of performance: Period 1: 2013/06/01 – 2014/05/31.
Period 2: 2014/06/01 – 2015/05/31.
Period 3: 2015/06/01 – 2016/05/31.
Period 4: 2016/06/01 – 2017/05/31.
Period 5: 2017/06/01 – 2018/05/31.

Total award amount and person-months : $920,000 and 44.4 months.

Relation to present proposal: The present ONR proposal is significantly different from the concurrent AFOSR proposal. The present ONR proposal is dedicated to the investigation of basic properties of information, including hierarchies, compression, updates. The AFOSR proposal is dedicated to interaction between entities in biology, neuroscience, and programming languages.

3 Concurrent grant application: NSF

Title of Proposal: EFRI-ODISSEI: Biomaterial design using hierarchical origami.

Summary: Multi-scale modeling (e.g. using category theory) of hierarchical materials.

Source and amount of funding: NSF. Annual total cost: $164,013. Percentage effort devoted to this project: 15%

Identity of prime Offeror: Boston University

List of subcontractors: MIT

Technical contact:
Markus J. Buehler
MIT Civil and Environmental Engineering Department
77 Massachusetts Avenue
Cambridge, MA 02139

Administrative/business contact :
Michele Hudak

Office of Sponsored Programs
Massachusetts Institute of Technology
Building NE18, Room 901
77 Massachusetts Avenue
Cambridge, MA 02139-4307
ph: 617-324-5382
fx: 617-253-4734
hudak@mit.edu

Period of performance: 2013/09/01 – 2017/08/31

Total award amount and person-months : $48,000 over 7.2 months.

Relation to present proposal: The NSF proposal is an application of work on ologs which was performed under a previous ONR grant. It may also use hierarchical ologs, but will be geared toward modeling and application, in contrast with mathematical research which will be performed for the present proposal.

Part VII
SETA statement and Data Rights

We also wish to note that MIT is not providing scientific, engineering, and technical assistance (SETA) or any similar support to any ONR technical office through an active contract or subcontract. In addition, MIT places no restrictions on the GovernmentÕs use of Intellectual Property contemplated under any subsequent award.

Part VIII
Qualifications

The PI received a Ph.D. in Mathematics from the University of California, Berkeley in 2007. The PI was then employed as a Paul Olum Visiting Assistant Professor by the University of Oregon from 2007 to 2010 and as a Postdoctoral associate at MIT from 2010 to present. He is well-trained in category theory and has studied information and communication for several years.

The PI's curriculum vitae is attached.