Toward a mathematical science of informatics

David I. Spivak

dspivak@math.mit.edu
Mathematics Department
Massachusetts Institute of Technology

Presented on 2013/06/13
at the National Institute of Standards and Technology

Outline of the talk

1. Introduction.
   - The problem to address.
2. Information, categories, and ologs.
   - What is information, and how do we work with it currently?
   - Basic category theory.
   - The similarity between information structures and categories.
3. Linking disparate information structures using CT.
   - Schema evolution.
   - Translation systems.
   - Data migration.
4. Forming a knowledge network.
5. Conclusion.

The goal is clarity and coherence

- The same issue is arising all over the world.
    - Increased complexity of multi-disciplinary systems.
    - The need to share information between parts of an emerging whole.
- We need to integrate multiple perspectives into an effective whole.
- This depends on *quality communication* between individuals and domains.

What creates quality communication?

- Communicating is inherently difficult.
    - The connection pattern of our brain is far more individualized than our finger print.
    - It follows that my structure of thinking is very different from yours.
    - How do I communicate to you if we each organize our information idiosyncratically?
- Quality communication is designed by the participants.
    - We work together to make communication occur.
    - E.g.: I speak, you give me feedback, I alter my approach to align with you.

What makes a good language?

- A good language should:
  - Be broad-stroke or fine-point as necessary.
  - Be rigorizable: google can tell me exactly how to get to NIST.
  - Be able to capture all the relevant distinctions.
  - Be able to hide the irrelevant distinctions.
  - Be efficient, not bogged down.
- Is atomic physics a good language for a soccer match?
  - I want to know who has the ball and whether they score.
  - I don't care where atom #15223599276746119424 is right now.
  - All the wrong things are being described.

The language problem in mathematics

- Mathematics is a network of understanding.
  - Until Frege, math's language was the result of happenstance.
  - There was no standard, no solid foundation.
  - Inconsistencies, paradoxes, anomalies emerged.
- Logic and set theory were proposed as a solid foundation.
  - The math community had been shaken by these paradoxes.
  - While set theory-as-foundation was strange, at least it settled things.
- The foundation was solid, but not scalable.
  - In the early 20th century, different math fields were growing apart.
  - Each subfield was siloed in its own language and ways of thinking.
  - Each had grown up separately and was focused on solving its own problems.
  - But they didn't understand each other, so their power was limited.
- Sound familiar?

Searching for an interlinking language

Interlinking language: Category theory

Logic/Set theory describes (?)

Topology	Algebra	Calculus

Interlinking language: Category Theory?

Quantum physics describes (?)

Computer program operation	Human psychology	Social movements

The language problem in science and society

We face the same issues today in the real world that mathematics faced in the early 20th century.

- In the sciences:
  - We have siloed approaches to different scientific disciplines.
  - In computer science, database (DB) theory is siloed apart from programming language (PL) theory.
- In society:
  - People are required to obey laws whose language they cannot understand.
  - Science is not communicated effectively to officials, other scientists, or society at large.
  - Local experts communicate in prose rather than in structured language.
- What is needed to make good decisions as a species?
  - We need a coherent understanding of our world.
  - For this we need to organize and network our knowledge.
  - For this we need a well-structured language.

What can category theory do for us?

- Category theory was invented to connect disparate mathematical fields.
  - The idea was to connect topology (the study of shapes) to algebra (the study of equations).
  - But the result was a language system that captures the essence of mathematical reasoning.
- I'll argue that **information** is governed by mathematical reasoning.
  - If so, then category theory should be useful as a language of information.
  - This talk will be an attempt to show that categories and information structures are quite similar.

Category theory in mathematics

- How category theory (CT) works in math.
  - Each mathematical subfield can be framed as a category.
  - Links between subfields can be framed as functors.
  - Functors are rigorous connections between mathematical fields.
    - What is the measure of this "rigorous connection"?
    - Theorems from one category, when passed through a functor, will remain true in the other category.
- Category theory: Not a language but a language system.
  - Each category $\mathcal{C}$ is a domain-specific language.
  - Each functor $\mathcal{C} \to \mathcal{D}$ is a translation system.
  - Category theory collects the most important features of languages and translations.
  - By knowing the essential "shapes" that a category can take, one can comprehend and tackle new situations quickly, like in Go or Chess.

Category theory interlinks subfields of mathematics

- The reception at first to CT was mixed.
    - Some used it to prove important conjectures.
    - Others dismissed it as “abstract nonsense”.
- By now the fight is basically over.
    - Like with climate change, there are a few hold-outs.
    - However, in domains that have a CT formulation, mathematicians who use CT easily outpace and overpower those who don’t.
    - In algebra, geometry, and topology, CT is now ubiquitous and greatly appreciated.

Category theory as a language of science

- Can CT be useful for creating quality communication in ordinary life?
  - My internal language is domain specific, fit to myself and my needs.
  - A company's database (think of this as its language) is fit to its needs.
  - A standard is fit to the needs of the individual group of stakeholders.
  - Can CT capture such domain specific languages?
  - Can CT help us translate between different languages?
- In this talk, I propose that:
  - CT can be useful for organizing information.
  - CT can be useful for translating information between entities.
  - Therefore, CT can help us form a knowledge network.

What is information?

- There is plenty of information being produced and used.
- But it is hard to say exactly what information *is*.
- Some sources of information:
  - Dictionaries.
  - Digitial circuit diagrams.
  - Architect's floor plans.
  - Databases.
- In contrast to the thing itself:
  - A leaf.
  - A novel.
  - A soccer match.
- The difference:
  - Information is *presented* in the former.
  - It must be *extracted* from the latter.

What is in common to information presentations?

- They are in formation.
  - Controlling formation is the same as enforcing order, dispelling chaos.
  - It obviates guessing.
  - It promotes effective reasoning.
  - Information is always in formation.
- Information presentations again:
  - Dictionaries.
  - Digital circuit diagrams.
  - Architect's floor plans.
  - Databases.
- What is common to these information presentations?
  - A certain structure / vocabulary / syntax to which the presentation conforms.
  - Let's call this structure the *language* of the presentation.
  - By conforming to a single language, the presentation becomes consistent and comprehensible – informative.

We will concentrate on databases

- Easiest information source to understand categorically: databases.
  - Part of specifying a database is specifying what its structure will be.
  - The information structure of a database is called its *schema*.
  - For databases to communicate, we link their schemas.
- We will see a tight connection between:
  - Categories (which we called "domain specific languages" on slide 10)
  - Database schemas (which we called "presentation languages" above.)

$$\mathbf{Cat} \simeq \mathbf{Sch}$$

- I will concentrate on (relational) databases throughout this talk.

What is a database?

- A database consists of a schema and conforming data.
- Database schema (conceptual layout).
  - A schema consists of a collection of tables.
  - Each table will house observations about a type of thing T.
  - Each table has some number of columns.
  - Each column corresponds to an observable of the type T.
- Database instance (on-the-ground facts).
  - A database instance is a collection of data.
  - Each table is filled with rows of data, one for each thing of type T.
  - All the data is in accordance with the schema.

Example database instance

A family of linked tables:

dog			
ID	**name**	**owner**	**address**
D101	Wally	P34	15 Ash St.
D102	Fido	P46	201 5th Ave.
D104	Buster	P17	27 Spring Ln.

person		
ID	**lastName**	**address**
P17	Jones	27 Spring Ln.
P19	Smith	201 Gladys Ave.
P34	Smith	15 Ash St.
P46	D'Angelo	201 5th Ave.

dogName
ID
Barkie
Buster
Fido
Puppers
Rosie
Samson
Wally

address
ID
15 Ash St.
27 Spring Ln.
201 5th Ave.
201 Gladys Ave.

lastName
ID
Bennet
D'Angelo
Jimenez
Jones
Moran
Smith
Vickers

A database schema, from the CT viewpoint

dog address $\simeq$ dog owner address

$$\begin{tikzcd}
\texttt{dog} \bullet \arrow[r, "\texttt{owner}"] \arrow[d] \arrow[dr, "\simeq"] & \texttt{person} \bullet \arrow[r] \arrow[d] & \texttt{lastName} \bullet \\
\texttt{name} \bullet & \texttt{address} \bullet &
\end{tikzcd}$$

dog			
ID	**name**	**owner**	**address**
D101	Wally	P34	15 Ash St.
D102	Fido	P46	201 5th Ave.
D104	Buster	P17	27 Spring Ln.

person		
ID	**lastName**	**address**
P17	Jones	27 Spring Ln.
P19	Smith	201 Gladys Ave.
P34	Smith	15 Ash St.
P46	D'Angelo	201 5th Ave.

name
ID
Buster
⋮

address
ID
15 Ash St.
⋮

lastName
ID
D'Angelo
⋮

Goal: a mathematical foundation for information structures

- The world's information is stored in databases.
- I wanted to find a mathematical basis for databases which:
    - Completely describes schemas, instances, and the relationship between them.
    - Formalizes all typical database operations and querying.
    - Simplifies schema evolution, data migration, and database merging.
    - Links with other information paradigms (RDF and programming languages).
    - Offers new insights and tools.
- How I judge success of the mathematical formulation.
    - Good if: it is simple.
    - Good if: it aligns database practice align.
    - Good if: it connects with well-oiled mathematical machinery.
    - Unimportant if: it agrees with current database theory.

My history with category theory

- My background is in algebraic topology and algebraic geometry.
- I first fell in love with category theory when it explained in two sentences what had been for me a very difficult concept: sheaves.
- CT was indispensable to my PhD research on derived manifolds.
- As a postdoc, I got interested in the human knowledge-network idea.
  - I thought "Category theory should be used in computer science".
  - I naively assumed that this was an original thought.
  - I knocked on doors in the CS department, to see if anyone wanted to talk CT.
  - Only one guy wanted to talk, and he was doing the most boring thing: databases.
- Now I make a living connecting CT and information structures.

What is a category?

- It's time for the formal definition of category.
- It's quite simple, once you get past the notation.
    - Nodes (we call them "objects").
    - Arrows.
    - Paths.

Definition of a category presentation. Part I: Constituents

A *category presentation* $\mathcal{C}$ consists of the following constituents:

1. A set $\mathbf{Ob}(\mathcal{C})$, called *the set of objects of* $\mathcal{C}$.
   - I'll denote each object $x \in \mathbf{Ob}(\mathcal{C})$ by $\overset{x}{\bullet}$.
2. A set $\mathbf{Arr}(\mathcal{C})$, called *the set of arrows of* $\mathcal{C}$, and two functions

$$src, tgt \colon \mathbf{Arr}(\mathcal{C}) \to \mathbf{Ob}(\mathcal{C}),$$

   assigning to each arrow its *source* and its *target* object, respectively.
   - An arrow $f \in \mathbf{Arr}(\mathcal{C})$ is often written $\overset{x}{\bullet}\xrightarrow{f}\overset{y}{\bullet}$, where $x = src(f), y = tgt(f)$.
   - We define a *path in* $\mathcal{C}$ to be a finite "head-to-tail" sequence of arrows in $\mathcal{C}$, e.g. $\overset{x}{\bullet}\xrightarrow{f}\overset{y}{\bullet}\xrightarrow{g}\overset{z}{\bullet}$.
   - Paths can have length n for any $n \in \mathbb{N}$, including $n = 0$ and $n = 1$.
3. An notion of equivalence for paths, denoted $\simeq$.

Definition of a category presentation. Part II: Rules

These constituents must satisfy the following requirements:

1. If $p \simeq q$ are equivalent paths then the sources agree: $src(p) = src(q)$.
2. If $p \simeq q$ are equivalent paths then the targets agree: $tgt(p) = tgt(q)$.
3. Suppose we have two paths (of any lengths) $b \to c$:

$$\begin{tikzcd}
& \bullet \arrow[r] & \cdots \arrow[r] & \bullet \arrow[dr] & \\
b \arrow[ur] \arrow[dr] \arrow[rrrr, dashed, bend left, "p"] \arrow[rrrr, dashed, bend right, "q"'] & & & & c \\
& \bullet \arrow[r] & \cdots \arrow[r] & \bullet \arrow[ur] &
\end{tikzcd}$$

   If $p \simeq q$ then for any extensions

$$\begin{tikzcd}
a \arrow[r, "m"] & b \arrow[rr, dashed, bend left, "p"] \arrow[rr, dashed, bend right, "q"'] \arrow[rr, phantom, "\simeq"] & & c
\end{tikzcd}
\quad \text{or} \quad
\begin{tikzcd}
b \arrow[rr, dashed, bend left, "p"] \arrow[rr, dashed, bend right, "q"'] \arrow[rr, phantom, "\simeq"] & & c \arrow[r, "n"] & d
\end{tikzcd}$$

   $m; p \simeq m; q$ and $p; n \simeq q; n$.

Our pictures have been categories

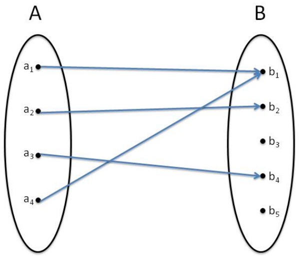
- These visual representations have been drawings of categories.

dog address $\simeq$ dog owner address

$$\mathcal{C} := \begin{tikzcd} \texttt{dog} \bullet \arrow[r, "\text{owner}"] \arrow[d] \arrow[dr] & \texttt{person} \bullet \arrow[r] \arrow[d] \arrow[dl, phantom, "\simeq"] & \texttt{lastName} \bullet \\ \texttt{name} \bullet & \texttt{address} \bullet & \end{tikzcd}$$

- Next, we'll see an example from pure mathematics.

Mathematical example: the category of Sets



- Above we see two sets and a function between them. We would denote this categorically by $\overset{A}{\bullet}\xrightarrow{f}\overset{B}{\bullet}$
  - The objects of **Set** represent sets.
  - The arrows in **Set** represent functions.
  - A path represents a sequence of composable functions.
  - Two paths are equivalent if their compositions are the same.

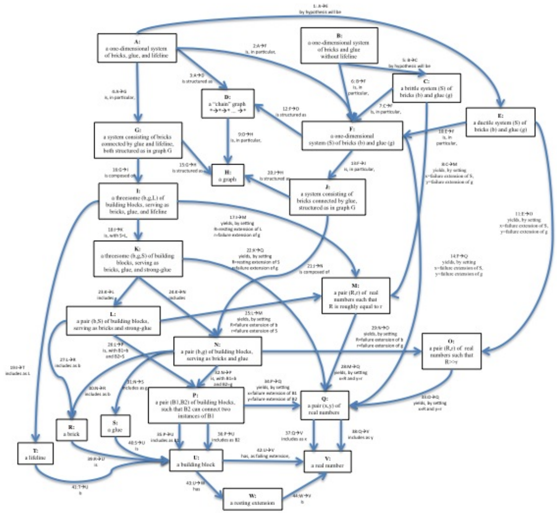
Ologs connect natural language, databases, and categories

- It turns out that categories and database schemas have the same structure!
- A third idea with the same structure is something I call *ologs*.

$$\begin{tikzcd}
\boxed{\text{a dog}} \arrow[r, "\text{has as owner}"] \arrow[d, "\text{has}"] \arrow[dr, "\text{has}"'] \arrow[r, phantom, "\simeq"', shift right=4ex] & \boxed{\text{a person}} \arrow[r, "\text{has}"] \arrow[d, "\text{has}"] & \boxed{\text{a last name}} \\
\boxed{\text{a dog name}} & \boxed{\text{an address}} &
\end{tikzcd}$$

- An olog is a natural language version of both a database schema and a category.

Example: an olog describing relation of structure and function of two materials



What is an olog?

- An olog is a conceptual description of a subject.
- Olog stands for "ontology log"
    - Ontology is the study of what something *is*.
    - "Log" because the study is never complete—always expanding.
- Components of an olog:
    - Labeled boxes,
    - Labeled arrows,
    - Path equivalences.
- Ologs are human readable.
- A student of mine made a simple English-to-Olog translator.
    - Enter an input file of sentences (e.g. "a dog has as owner a person")
    - It outputs an olog (e.g. $\boxed{\text{a dog}} \xrightarrow{\text{has as owner}} \boxed{\text{a person}}$).

Ologs are database schemas 1: an example olog

$$\begin{tikzcd}
 & \boxed{\text{a woman}} \arrow[d, "\text{is}"] & \\
\boxed{\text{a dog}} \arrow[r, "\text{has as owner}"] \arrow[d, "\text{has}"] & \boxed{\text{a person}} \arrow[r, "\text{has}"] \arrow[d, "\text{has}"] & \boxed{\text{a last name}} \\
\boxed{\text{a dog name}} & \boxed{\text{a first name}} &
\end{tikzcd}$$

- Olog boxes become database tables
- Olog arrows become database columns.
  - We can predict how many columns the $\boxed{\text{a dog}}$ table will have.

Ologs are database schemas 2: database

$$\begin{tikzcd}
 & \boxed{\text{a woman}} \arrow[d, "\text{is}"] & \\
\boxed{\text{a dog}} \arrow[r, "\text{has as owner}"] \arrow[d, "\text{has}"] & \boxed{\text{a person}} \arrow[r, "\text{has}"] \arrow[d, "\text{has}"] & \boxed{\text{a last name}} \\
\boxed{\text{a dog name}} & \boxed{\text{a first name}} &
\end{tikzcd}$$

a woman

ID	is a person
W17	P17
W34	P34
W38	P38
W51	P51

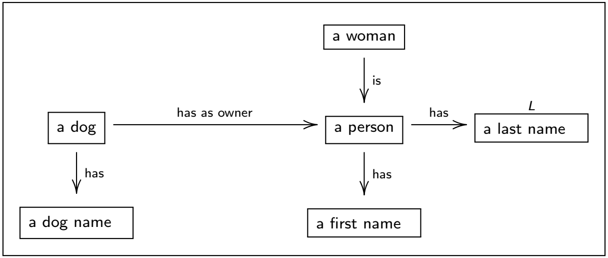
a dog

ID	has as owner a person	has a dog name
D101	P34	Wally
D102	P46	Fido
D103	P34	Samson
D104	P17	Buster
D106	P19	Rosie

a person

ID	has a first name	has a last name
P17	Alice	Jones
P19	Bob	Smith
P34	Barbara	Smith
P38	Sandra	Moran
P46	Jeremy	D'Angelo
P51	Luisa	Jimenez

Leaf tables

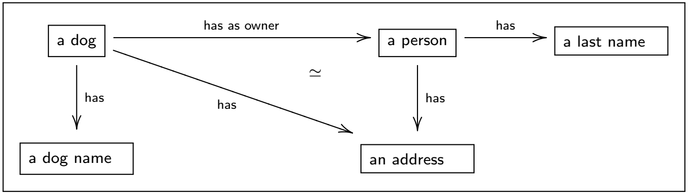


a dog name
ID
Barkie
Buster
Fido
Puppers
Rosie
Samson
Wally

a first name
ID
Alice
Bob
Barbara
Carl
Jeremy
Luisa
Sandra
Thomas

a last name
ID
Bennet
D'Angelo
Jimenez
Jones
Moran
Smith
Vickers

Equivalent paths require equivalent data



$$\begin{tikzcd}
\boxed{\text{a dog}} \arrow[r, "\text{has as owner}"] \arrow[d, "\text{has}"] \arrow[dr, "\text{has}"'] \arrow[dr, phantom, "\simeq", shift left=4] & \boxed{\text{a person}} \arrow[r, "\text{has}"] \arrow[d, "\text{has}"] & \boxed{\text{a last name}} \\
\boxed{\text{a dog name}} & \boxed{\text{an address}} &
\end{tikzcd}$$

dog			
ID	**name**	**owner**	**address**
D101	Wally	P34	15 Ash St.
D102	Fido	P46	201 5th Ave.
D104	Buster	P17	27 Spring Ln.

person		
ID	**lastName**	**address**
P17	Jones	27 Spring Ln.
P19	Smith	201 Gladys Ave.
P34	Smith	15 Ash St.
P46	D'Angelo	201 5th Ave.

Another example of path equivalences

$$\mathcal{C} := \boxed{\begin{array}{l} \underline{\text{Mgr;Dpt} \simeq \text{Dpt}} \qquad \underline{\text{Secr;Dpt} \simeq \text{id}_{\text{Department}}} \\ \begin{tikzcd} & \texttt{Employee} \arrow[loop above, "\text{Mgr}"] \arrow[dl, "\text{First}"'] \arrow[dr, "\text{Last}"] \arrow[rr, shift left, "\text{Dpt}"] & & \texttt{Department} \arrow[ll, shift left, "\text{Secr}"] \arrow[d, "\text{Name}"] \\ \texttt{String1} & & \texttt{String2} & \texttt{String3} \end{tikzcd} \end{array}}$$

Employee				
Id	**First**	**Last**	**Mgr**	**Dpt**
101	David	Hilbert	103	q10
102	Bertrand	Russell	102	x02
103	Alan	Turing	103	q10

Department		
Id	**Name**	**Secr**
q10	Sales	101
x02	Production	102

String
Id
a
b
⋮
z
aa
ab
⋮

Ologs bridge the divide

- Each olog is authored by an individual or group, about a subject.
  - The olog idea can be understood by ordinary people.
  - No database theory or category theory background necessary.
  - I can teach almost anyone to make DB schemas (as ologs) in an hour.
- Ologs are both databases and categories, in disguise.
  - Ologs are database schemas; we can fill them with relevant data.
  - Ologs are categories; mathematics can be brought to bear.
- I will use the following words interchangeably:
  - Olog,
  - Database schema,
  - Category.

Lowering the barrier to using information structures

- Why does wikipedia work?
    - No one would have imagined people would take the time.
    - But the barrier to making small contributions was lowered.
    - As a result, wikipedia is wildly successful.
- We need to lower the barrier to information structures.
    - Easily put our information online in a structured way.
    - Easily combine and query existing data from multiple sources.
- Quality communication means sharing information easily.
    - I hope that ologs and other ideas can help.

This talk: where we are and where we're going

- We've discussed what information is, specifically focusing on databases.
- We've shown how categories capture database schemas.
- We want to talk about *linking* information structures.
    - This will bring us to functors.
    - Functors connect categories, hence they connect database schemas.
    - But we'll also see that functors connect schemas to data.

Linking information structures together

- Forming a coherent whole.
    - Different scientists or different banks may structure their data differently.
    - If they are studying the same subject, links should exist.
    - We want to stitch differently-structured schemas together.
    - Connecting different schemas is the same as connecting different categories.
- Category theory was designed specifically for this.
- Next we will discuss the links between categories, called *functors*.

Functors: mappings between categories

- One way to think of a category is as a directed graph, where certain paths have been declared equivalent.
- A functor is a graph-mapping that is required to respect equivalence of paths.
- **Definition**: A functor $F\colon \mathcal{C} \to \mathcal{D}$ consists of
  - a function $\textbf{Ob}(\mathcal{C}) \to \textbf{Ob}(\mathcal{D})$ and
  - a function $\textbf{Arr}(\mathcal{C}) \to \textbf{Path}(\mathcal{D})$,

  such that F
  - respects sources and targets,
  - respects equivalences of paths.

Backing up: a database instance is a functor!

- A database schema (layout of tables) is simply a category $\mathcal{C}$.

dog address $\simeq$ dog owner address

$$\mathcal{C} := \begin{tikzcd}
\texttt{dog} \arrow[r, "\text{owner}"] \arrow[d] \arrow[dr] & \texttt{person} \arrow[r] \arrow[d] \arrow[dl, phantom, "\simeq"] & \texttt{lastName} \\
\texttt{name} & \texttt{address} &
\end{tikzcd}$$

- As we said, there is a category **Set** of sets and functions.
- A functor $I\colon \mathcal{C} \to \mathbf{Set}$ assigns:
  - to each object $c \in \mathbf{Ob}(\mathcal{C})$ a set $I(c)$,
  - to each arrow $h\colon c \to d$ in $\mathcal{C}$ a function $I(h)\colon I(c) \to I(d)$,
  - such that all path equivalences are respected.
- In other words, a functor $I\colon \mathcal{C} \to \mathbf{Set}$ is a database instance on $\mathcal{C}$; i.e. it is a way to fill $\mathcal{C}$ with compatible data.

Example

$\mathcal{C} :=$

dog address $\simeq$ dog owner address

$$\begin{tikzcd}
\overset{\texttt{dog}}{\bullet} \arrow[r, "\texttt{owner}"] \arrow[d] \arrow[dr, ""{name=D}] & \overset{\texttt{person}}{\bullet} \arrow[r] \arrow[d] & \overset{\texttt{lastName}}{\bullet} \\
\overset{\texttt{name}}{\bullet} & \overset{\texttt{address}}{\bullet} &
\end{tikzcd}$$

$\simeq$

We can represent a functor

$$I : \mathcal{C} \to \textbf{Set}$$

as follows:

dog			
ID	**name**	**owner**	**address**
D101	Wally	P34	15 Ash St.
D102	Fido	P46	201 5th Ave.
D104	Buster	P17	27 Spring Ln.

person		
ID	**lastName**	**address**
P17	Jones	27 Spring Ln.
P19	Smith	201 Gladys Ave.
P34	Smith	15 Ash St.
P46	D'Angelo	201 5th Ave.

name
ID
Buster
⋮

address
ID
15 Ash St.
⋮

lastName
ID
D'Angelo
⋮

Changes in schema

- Suppose in our modeling of a given subject, we evolve from schema $\mathcal{C}$ to schema $\mathcal{D}$.
- We should find a functorial connection between them.
- Over time we may have something like

$$\mathcal{C} = \mathcal{C}_0 \xrightarrow{F_0} \mathcal{C}_1 \xrightarrow{F_1} \cdots \xrightarrow{F_n} \mathcal{C}_n = \mathcal{D}$$

- We want to be able to migrate data from $\mathcal{C}$ to $\mathcal{D}$ and vice versa.
- We want to be able to migrate queries against $\mathcal{C}$ to queries against $\mathcal{D}$ and vice versa.
- And we want this all to work as expected.

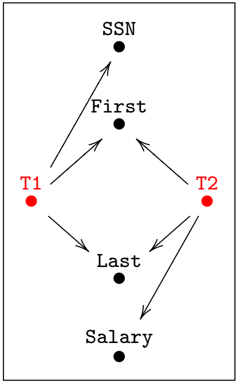
Functorial data migration for CT experts

- For any schema (category) $\mathcal{C}$, we have the category $\mathcal{C}$–**Set** of set-valued functors $I\colon \mathcal{C} \to \mathbf{Set}$ and natural transformations. These are the instances of the database.
- A functor $F\colon \mathcal{C} \to \mathcal{D}$ serves as a translation between schemas.
- Composition with F induces a functor $\Delta_F\colon \mathcal{D}\text{–}\mathbf{Set} \to \mathcal{C}\text{–}\mathbf{Set}$,

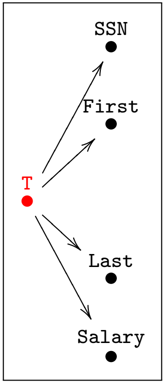
$$\mathcal{C} \xrightarrow{F} \mathcal{D} \xrightarrow{I} \mathbf{Set}.$$

- The functor Δ_F migrates data from $\mathcal{D}$ back to $\mathcal{C}$.
- It has two adjoints $\Sigma_F\colon \mathcal{C}\text{–}\mathbf{Set} \to \mathcal{D}\text{–}\mathbf{Set}$ and $\Pi_F\colon \mathcal{C}\text{–}\mathbf{Set} \to \mathcal{D}\text{–}\mathbf{Set}$.

Uses of functorial data migration 0: Translation F

$\mathcal{C} :=$ 

$\xrightarrow{F}$



$=: \mathcal{D}$

Uses of functorial data migration 1: Projection via Δ_F

$$\begin{tikzcd}
 & & \texttt{SSN} & & & & & \texttt{SSN} & \\
 & & \texttt{First} & & & & & \texttt{First} & \\
\mathcal{C} := & \texttt{T1} \arrow[uur] \arrow[ur] \arrow[dr] & & \texttt{T2} \arrow[ul] \arrow[dl] \arrow[ddl] & {} \arrow[r, "F"] & {} & \texttt{T} \arrow[uur] \arrow[ur] \arrow[dr] \arrow[ddr] & & =: \mathcal{D} \\
 & & \texttt{Last} & & & & & \texttt{Last} & \\
 & & \texttt{Salary} & & & & & \texttt{Salary} &
\end{tikzcd}$$

$J \colon \mathcal{D} \to$ **Set**:

T				
ID	**SSN**	**First**	**Last**	**Salary**
XF667	115-234	Bob	Smith	\$250
XF891	122-988	Sue	Smith	\$300
XF221	198-877	Alice	Jones	\$100

$\Delta_F(J) \colon \mathcal{C} \to$ **Set**:

T1			
ID	**SSN**	**First**	**Last**
XF667T1	115-234	Bob	Smith
XF891T1	122-988	Sue	Smith
XF221T1	198-877	Alice	Jones

T2			
ID	**First**	**Last**	**Salary**
XF667T2	Bob	Smith	\$250
XF891T2	Sue	Smith	\$300
XF221T2	Alice	Jones	\$100

Uses of functorial data migration 2: Joins via Π_F

$$\mathcal{C} := \begin{tikzcd} & \texttt{SSN} & \\ & \texttt{First} & \\ \texttt{T1} \arrow[uur] \arrow[ur] \arrow[dr] & & \texttt{T2} \arrow[ul] \arrow[dl] \arrow[ddl] \\ & \texttt{Last} & \\ & \texttt{Salary} & \end{tikzcd} \xrightarrow{F} \begin{tikzcd} & \texttt{SSN} \\ & \texttt{First} \\ \texttt{T} \arrow[uur] \arrow[ur] \arrow[dr] \arrow[ddr] & \\ & \texttt{Last} \\ & \texttt{Salary} \end{tikzcd} =: \mathcal{D}$$

$I\colon \mathcal{C} \to$ **Set**:

T1

ID	**SSN**	**First**	**Last**
T1-001	115-234	Bob	Smith
T1-002	122-988	Sue	Smith
T1-003	198-877	Alice	Jones

T2

ID	**First**	**Last**	**Salary**
T2-A101	Alice	Jones	\$100
T2-A102	Sam	Miller	\$150
T2-A104	Sue	Smith	\$300
T2-A110	Carl	Pratt	\$200

$\Pi_F(I)\colon \mathcal{D} \to$ **Set**:

T

ID	**SSN**	**First**	**Last**	**Salary**
T1-002T2-A104	122-988	Sue	Smith	\$300
T1-003T2-A101	198-877	Alice	Jones	\$100

Uses of functorial data migration 3: Unions via Σ_F

$$\mathcal{C} := \begin{tikzcd} & \texttt{SSN} & \\ & \texttt{First} & \\ \texttt{T1} \arrow[uur] \arrow[ur] \arrow[dr] & & \texttt{T2} \arrow[ul] \arrow[dl] \arrow[ddl] \\ & \texttt{Last} & \\ & \texttt{Salary} & \end{tikzcd} \xrightarrow{F} \begin{tikzcd} & \texttt{SSN} \\ & \texttt{First} \\ \texttt{T} \arrow[uur] \arrow[ur] \arrow[dr] \arrow[ddr] & \\ & \texttt{Last} \\ & \texttt{Salary} \end{tikzcd} =: \mathcal{D}$$

$I\colon \mathcal{C} \to$ **Set**:

T1			
ID	**SSN**	**First**	**Last**
T1-001	115-234	Bob	Smith
T1-002	122-988	Sue	Smith
T1-003	198-877	Alice	Jones

T2			
ID	**First**	**Last**	**Salary**
T2-A101	Alice	Jones	\$100
T2-A102	Sam	Miller	\$150
T2-A104	Sue	Smith	\$300
T2-A110	Carl	Pratt	\$200

$\Sigma_F(I)\colon \mathcal{D} \to$ **Set**:

T				
ID	**SSN**	**First**	**Last**	**Salary**
T1-001	115-234	Bob	Smith	T1-001.Salary
T1-002	122-988	Sue	Smith	T1-002.Salary
T1-003	198-877	Alice	Jones	T1-003.Salary
T2-A101	T2-A101.SSN	Alice	Jones	\$100
T2-A102	T2-A102.SSN	Sam	Miller	\$150
T2-A104	T2-A104.SSN	Sue	Smith	\$300
T2-A110	T2-A110.SSN	Carl	Pratt	\$200

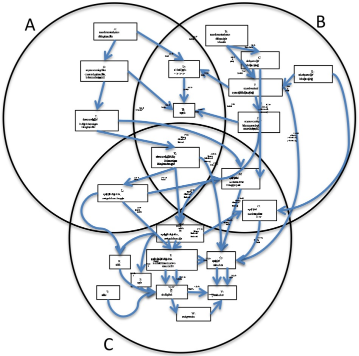
Ryan Wisnesky's FQL program

- Most of this has been implemented.
    - I'm working with a Harvard CS graduate student named Ryan Wisnesky.
    - He has implemented almost all of the above, and more.
        - It's called FQL (Functorial Query Language)
        - Create category-theoretic schemas, mappings, instances, queries.
    - FQL is available online, and it's open source.

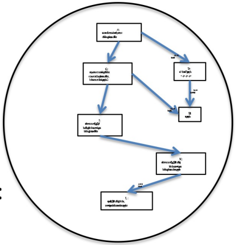
Category theory in academia and industry

- Category theory naturally fosters connections between disparate fields.
- It has branched out of math and into physics, linguistics, materials science, and biology.
- It has had much success in computer science.
    - Specifically important in the theory of programming languages.
    - The category-theoretic concept of *monads* has vastly extended the reach of functional programming.
- It is a language for formalizing analogies.
    - I collaborate with a material science professor at MIT (M. Buehler).
    - E.g., we articulated a formal analogy between spider silk and western music.
- Collaboration with industry.
    - Amgen, Microsoft, etc.

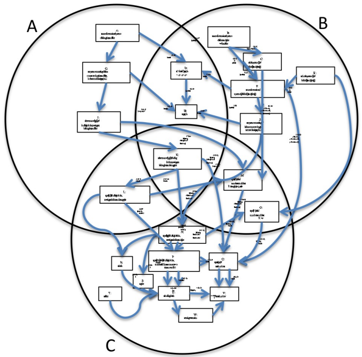
Network of scientists 1: overlapping understanding

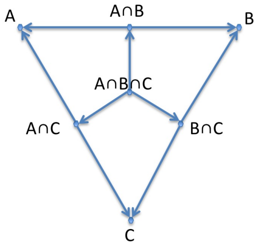


Scientist A's research topic:

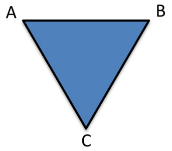


Network of scientists 2: encoding interaction groups



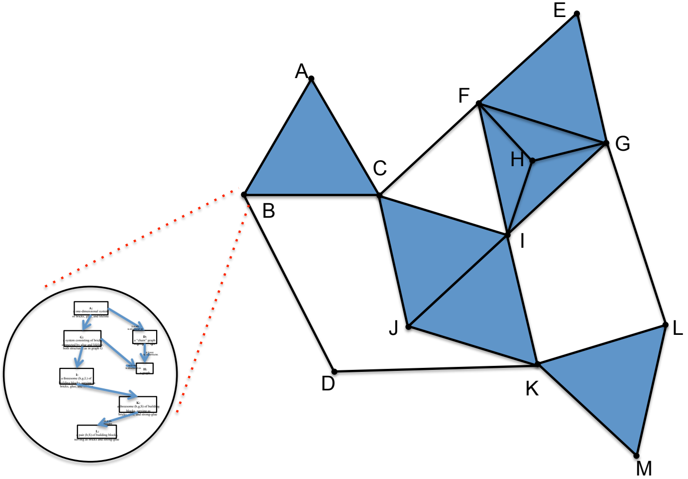


Abstraction:



Network of scientists 3: simplicial complex

A network of ologs, a network of scientific understanding.


A
B
C
D
E
F
G
H
I
J
K
L
M
A: a one-dimensional system of bricks, glue, and lifeline
D: a "chain" graph
G: a system consisting of bricks connected by glue and lifeline, both structured as in graph G
H: a graph
I: a threesome (b,g,L) of building blocks, serving as bricks, glue, and lifeline
K: a threesome (b,g,S) of building blocks, serving as bricks, glue, and strong-glue
L: a pair (b,S) of building blocks, serving as bricks and strong-glue
is structured as
is structured as
includes

This whole network can be queried, with provenance plainly evident.

Summary of the talk

- We need to improve our ability to communicate rigorously about complex subjects.
  - Transferring knowledge from one group to another is difficult.
  - It cannot be left to human guessing and ad-hoc interpretation.
  - We need to have available a high-assurance framework for communication.
- Ologs and category theory provide such a framework.
  - Categories and databases are quite similar.
  - Functors link schemas holistically.
    - Each functor $\mathcal{C} \to \mathcal{D}$ establishes various data migration functors.
    - These can act as queries (project, join, select, union).
  - A network of linked databases can serve as an atlas of knowledge.

Future work

- Connect to existing information frameworks
  - We have connected databases to RDF and PL.
  - To do: UML, OWL, XML, etc.
- Build tools that actually serve our needs using CT ideas.
  - Translators between olog/CT representations and these existing models.
  - Lower the bar to using information structures.
- Apply CT ideas to existing interoperability problems.
  - Medical health records.
  - Supply chain.

Thank you

Thanks for inviting me to speak!

Reference links:

- Category Theory for Scientists (book).
- Databases:
  - Functorial Data Migration (paper).
  - Relational foundations of Functorial Data Migration (paper, joint with R. Wisnesky).
  - Download Wisnesky's FQL (program)
- Ologs (paper, joint with R. Kent).
- CT for RDF and SPARQL (paper)
- Materials science papers (joint with M. Buehler, et al.):
  - Formal analogy: Spider silk and western music.
  - Ductility in materials and social networks.
  - Building block replacement problem.

Appendix

Contents:

- RDF via the Grothendieck construction.
- A sample SQL query using data migration functors.

The Grothendieck construction

- Let $\mathcal{C}$ be a category and let $I\colon \mathcal{C} \to \mathbf{Set}$ be a functor.
- We can convert I into a category $Gr(I)$ in a canonical way:
  - Example:

$$\mathcal{C} := \boxed{\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "g"'] & B \\ C \end{tikzcd}}\,; \qquad I = \boxed{\begin{tikzcd} \boxed{\bullet a_1 \ \bullet a_2 \ \bullet a_3} \arrow[r, "{(b_1,b_1,b_2)}"] \arrow[d] & \boxed{\bullet b_1 \ \bullet b_2} \\ \boxed{\bullet c_1} \end{tikzcd}}$$

  - $Gr(I)$ is also known as *the category of elements of* I:

$$\boxed{\begin{tikzcd} \bullet a_1 \arrow[rrr, bend left] \arrow[dr] & \bullet a_2 \arrow[rr, bend left] \arrow[d] & \bullet a_3 \arrow[rr, bend right] \arrow[dl] & \bullet b_1 & \bullet b_2 \\ & \bullet c_1 \end{tikzcd}}$$

Grothendieck construction applied to database instances

- Suppose given the following instance, considered as $I\colon \mathcal{C} \to \mathbf{Set}$

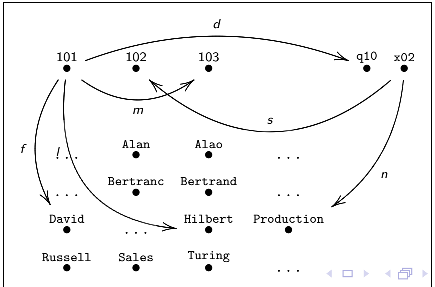
Employee				
Id	**First**	**Last**	**Mgr**	**Dpt**
101	David	Hilbert	103	q10
102	Bertrand	Russell	102	x02
103	Alan	Turing	103	q10

Department		
Id	**Name**	**Secr'y**
q10	Sales	101
x02	Production	102

$$\mathcal{C} = \boxed{\begin{array}{l} \underline{m;d \simeq d} \qquad \underline{s;d \simeq \mathrm{id}_D} \\ \begin{tikzcd} \bullet E \arrow[loop left, "m"] \arrow[rr, shift left, "d"] \arrow[d, bend right, "f"'] \arrow[d, bend left, "l"] & & \bullet D \arrow[ll, shift left, "s"] \arrow[dll, "n"] \\ \bullet S & & \end{tikzcd} \end{array}}$$

Here is $Gr(I)$, the category of elements of I:

$Gr(I) =$

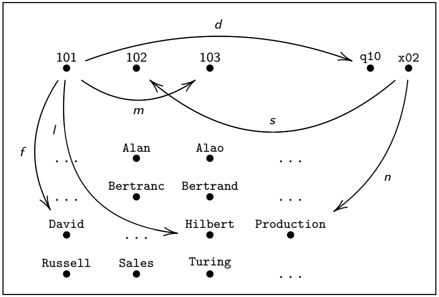


A different perspective on data

In fact, the Grothendieck construction of $I\colon \mathcal{C} \to \mathbf{Set}$ always yields not only a category $Gr(I)$ but a functor

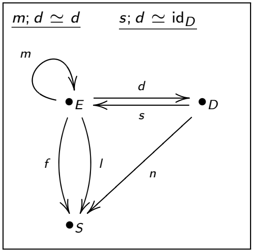
$$\pi\colon Gr(I) \to \mathcal{C}.$$

$Gr(I)$:=



$\xrightarrow{\pi}$

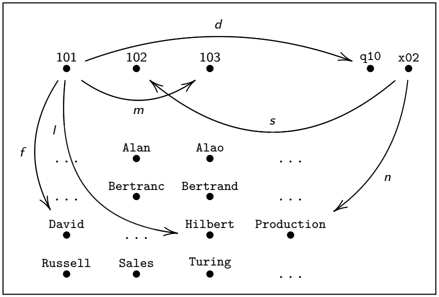
$\mathcal{C}$:=


$\underline{m; d \simeq d}$
$\underline{s; d \simeq \mathrm{id}_D}$
m
E
d
s
D
f
l
n
S

The fiber over (inverse image of) every object $X \in \mathcal{C}$ is a set of objects $\pi^{-1}(X) \subseteq Gr(I)$. That set is $I(X)$.

RDF schema and stores

$Gr(I)=$



$\xrightarrow{\pi}$

$\mathcal{C} =$

$\underline{m; d \simeq d}$ $\underline{s; d \simeq \mathrm{id}_D}$

$$\begin{tikzcd} \bullet_E \arrow[loop left, "m"] \arrow[r, shift left, "d"] \arrow[d, bend right, "f"'] \arrow[d, bend left, "l"] & \bullet_D \arrow[l, shift left, "s"] \arrow[dl, "n"] \\ \bullet_S & \end{tikzcd}$$

- The relation to RDF triples is clear: each arrow $f: x \to y$ in $Gr(I)$ is a triple with subject x, predicate f, and object y.
- For example (101 department q10), (x02 name Production), etc..
- $\mathcal{C}$ is the RDF schema and $Gr(I)$ is the triple store.
- SPARQL queries (graph patterns) are easily expressible in this model.

A simple “SELECT” query using functors

SELECT title, isbn
FROM book
WHERE price $>$ 100

$$\mathcal{C} := \begin{tikzcd} & \text{book} \arrow[r, "\text{title}"] \arrow[d, "\text{price}"] \arrow[dr, "\text{isbn}"] & \texttt{String} \\ \mathbb{R}_{>100} \arrow[r] & \mathbb{R} & \texttt{isbn-num} \end{tikzcd} \xrightarrow{F} \mathcal{D} := \begin{tikzcd} W \arrow[r] \arrow[d] \arrow[dr, phantom, "\checkmark"] & \text{book} \arrow[r, "\text{title}"] \arrow[d, "\text{price}"] \arrow[dr, "\text{isbn}"] & \texttt{String} \\ \mathbb{R}_{>100} \arrow[r] & \mathbb{R} & \texttt{isbn-num} \end{tikzcd} \xleftarrow{G} \mathcal{E} := \begin{tikzcd} W \arrow[r, "\text{title}"] \arrow[dr, "\text{isbn}"] & \texttt{String} \\ & \texttt{isbn-num} \end{tikzcd}$$

- $V := \Delta_G \circ \Pi_F$ is the appropriate sequence of functors.
- For any $I : \mathcal{C} \to \mathbf{Set}$, we materialize the query as $V(I)$.
- Views with foreign keys are easy.