Categorical Databases

David I. Spivak

Department of Mathematics
Massachusetts Institute of Technology
Categorical Informatics Inc.
www.catinf.com

Presented at BAE
January 9, 2019

Outline

An invitation to engage with us, and solve real-world problems.

A road to true interdisciplinarity

- Scientific disciplines are conceptual analogies of the world.
  - Science: a schematic, conceptual account of phenomena.
  - Engineering is using these accounts to channel world events.
  - But how do different disciplines and accounts cohere?
  - To solve big problems, we need to connect different approaches.

A road to true interdisciplinarity

- Scientific disciplines are conceptual analogies of the world.
  - Science: a schematic, conceptual account of phenomena.
  - Engineering is using these accounts to channel world events.
  - But how do different disciplines and accounts cohere?
  - To solve big problems, we need to connect different approaches.
- We need a shared fabric, a substrate for interdiscipinarity.
  - Interdisciplinarity consists of effective analogy-making.
  - To go further, we need to formalize the analogies themselves.

A road to true interdisciplinarity

- Scientific disciplines are conceptual analogies of the world.
  - Science: a schematic, conceptual account of phenomena.
  - Engineering is using these accounts to channel world events.
  - But how do different disciplines and accounts cohere?
  - To solve big problems, we need to connect different approaches.
- We need a shared fabric, a substrate for interdiscipinarity.
  - Interdisciplinarity consists of effective analogy-making.
  - To go further, we need to formalize the analogies themselves.
- Better yet: we need a conceptual stem-cell.
  - Something that can differentiate into huge variety of forms.
  - Find the analogies between forms as aspects within the stem cell.

Category theory as conceptual stem-cell

Category theory (CT) can differentiate into many forms:

- All forms of pure math... (we'll briefly discuss this)

Category theory as conceptual stem-cell

Category theory (CT) can differentiate into many forms:

- All forms of pure math... (we'll briefly discuss this)
- Databases and knowledge representation (categories and functors)
- Functional programming languages (cartesian closed categories)
- Universal algebra (finite-product categories)
- Dynamical systems and fractals (operad-algebras, co-algebras)
- Shannon Entropy (operad of simplices)
- Partially-ordered sets and metric spaces (enriched categories)
- Higher order logic (toposes = categories of sheaves)
- Measurements of diversity in populations (magnitude of categories)
- Collaborative design (enriched categories and profunctors)
- Petri nets and chemical reaction networks (monoidal categories)
- Quantum processes and NLP (compact closed categories)

Popper's objection

"A theory that explains everything explains nothing." – Karl Popper

Popper's objection

"A theory that explains everything explains nothing." – Karl Popper

We counter this objection in two ways:

- Couldn't the same objection be made about mathematics?
  - Mathematics is the basis of hard science, used everywhere.
  - CT—like math—explains, models, formalizes many many things.
  - Conclude that math/CT explains everything and hence nothing?

Popper's objection

"A theory that explains everything explains nothing." – Karl Popper

We counter this objection in two ways:

- Couldn't the same objection be made about mathematics?
  - Mathematics is the basis of hard science, used everywhere.
  - CT—like math—explains, models, formalizes many many things.
  - Conclude that math/CT explains everything and hence nothing?
- Stem cells don't do work until they differentiate.
  - "Adult-level" work requires differentiation and optimization.
  - But the unified origins lead to impressive interoperability.

CT is the mathematics of mathematics.

You could also say: CT is mathematics, self-aware.

- Designed to transport theorems from one area of math to another.
  - From topology (shapes) to algebra (equations).
  - This isn't mere analogy, it's analogy made rigorous.

CT is the mathematics of mathematics.

You could also say: CT is mathematics, self-aware.

- Designed to transport theorems from one area of math to another.
  - From topology (shapes) to algebra (equations).
  - This isn't mere analogy, it's analogy made rigorous.
- It's revolutionized pure math since its inception in 1940s.
  - Most modern pure math research is written cat.-theoretically.
  - It's become a gateway to pure mathematics.

CT is the mathematics of mathematics.

You could also say: CT is mathematics, self-aware.

- Designed to transport theorems from one area of math to another.
    - From topology (shapes) to algebra (equations).
    - This isn't mere analogy, it's analogy made rigorous.
- It's revolutionized pure math since its inception in 1940s.
    - Most modern pure math research is written cat.-theoretically.
    - It's become a gateway to pure mathematics.
- And it's branched out from math in a big way.
    - Databases and knowledge representation (categories and functors)
    - Functional programming languages (cartesian closed categories)
    - Dynamical systems and fractals (operad-algebras, co-algebras)
    - Shannon Entropy (operad of simplices)
    - Measurements of diversity in populations (magnitude of categories)
    - Collaborative design (enriched categories and profunctors)
    - Petri nets and chemical reaction networks (monoidal categories)
    - Quantum processes and NLP (compact closed categories)

Our historical moment

Compare the information revolution to the industrial revolution:

Our historical moment

Compare the information revolution to the industrial revolution:

- Industrial revolution
  - Sewage in streets, runoff in rivers, smog in skies.
  - Uncontrolled flows of stuff.

Our historical moment

Compare the information revolution to the industrial revolution:

- Industrial revolution
  - Sewage in streets, runoff in rivers, smog in skies.
  - Uncontrolled flows of stuff.
- Information revolution
  - Big data is messy: it's gleaned, not channeled.
  - Break Humpty Dumpty into a thousand pieces, then reconstruct?
  - Uncontrolled flows of information.

Our historical moment

Compare the information revolution to the industrial revolution:

- Industrial revolution
  - Sewage in streets, runoff in rivers, smog in skies.
  - Uncontrolled flows of stuff.
- Information revolution
  - Big data is messy: it's gleaned, not channeled.
  - Break Humpty Dumpty into a thousand pieces, then reconstruct?
  - Uncontrolled flows of information.

If you care about information hygiene, CT needs to be on your radar.

Getting to specifics

The category-theoretic stem cell is about compositional design patterns.

Getting to specifics

The category-theoretic stem cell is about compositional design patterns.

Let's focus on one: Data frameworks and data transformations.

- The problem: multiple models of similar information
- What is "model-space"?

Getting to specifics

The category-theoretic stem cell is about compositional design patterns.

Let's focus on one: Data frameworks and data transformations.

- The problem: multiple models of similar information
- What is "model-space"?
- Category theory offers a mathematical notion of model-space.
- The kinematics of data: how it moves and rests.

Plan of the talk

- The problem: pervasive and insidious.
- The math: Category theory describes model space.
- The tool: Open-source implementation and commercialization.

Outline

An invitation to engage with us, and solve real-world problems.

The Copernican revolution continues

The earth was the center; then the sun; then no need for center.

The Copernican revolution continues

The earth was the center; then the sun; then no need for center.

- Having a world-center provides an origin; good for coordinating.
- But there isn't just one best coordinate system.
- Each coordinate system is a perspective, a basis for calculation.

(Read the above in terms of algebra and in terms of information.)

The Copernican revolution continues

The earth was the center; then the sun; then no need for center.

- Having a world-center provides an origin; good for coordinating.
- But there isn't just one best coordinate system.
- Each coordinate system is a perspective, a basis for calculation.

(Read the above in terms of algebra and in terms of information.)

- Linear Algebra studies coordinate systems *and transformations*.
- But people still search for the "best" information model.
  - E.g. OMOP in EMRs
  - BFO, CIDOC, SUMO, etc., etc. in upper ontologies
- Let's change focus to transformations.

The Copernican revolution continues

The earth was the center; then the sun; then no need for center.

- Having a world-center provides an origin; good for coordinating.
- But there isn't just one best coordinate system.
- Each coordinate system is a perspective, a basis for calculation.

(Read the above in terms of algebra and in terms of information.)

- Linear Algebra studies coordinate systems *and transformations*.
- But people still search for the "best" information model.
  - E.g. OMOP in EMRs
  - BFO, CIDOC, SUMO, etc., etc. in upper ontologies
- Let's change focus to transformations.

Multiplicity of perspectives is not going away. Let's learn to integrate.

Information integration

- Information:
  - It's constantly being generated;
  - It arises from multiple sources and perspectives;
  - It must be integrated to solve larger problems.

Information integration

- Information:
  - It's constantly being generated;
  - It arises from multiple sources and perspectives;
  - It must be integrated to solve larger problems.
- Information integration:
  - Putting things together.
  - Making connections, drawing analogies.
  - Finding common structures.

Information integration

- Information:
  - It's constantly being generated;
  - It arises from multiple sources and perspectives;
  - It must be integrated to solve larger problems.
- Information integration:
  - Putting things together.
  - Making connections, drawing analogies.
  - Finding common structures.
- Information kinematics:
  - Information rests in databases.
  - Information moves by data transformations.
  - Let's dig in.

Information kinematics

Information rests primarily in databases.

- Domain knowledge informs the structure of the database.
  - The structure is called the database *schema*.
  - It consists of a collection of interlocking tables.
- The data itself is structured according to the schema.

Information kinematics

Information rests primarily in databases.

- Domain knowledge informs the structure of the database.
  - The structure is called the database *schema*.
  - It consists of a collection of interlocking tables.
- The data itself is structured according to the schema.

Information moves through transformations.

- Different situations require different data structures.
- Most common transformation: *querying*:
  - A query transforms data from one structure (schema) to another.
  - The result of a query is a schema with only one table.

Information kinematics

Information rests primarily in databases.

- Domain knowledge informs the structure of the database.
  - The structure is called the database *schema*.
  - It consists of a collection of interlocking tables.
- The data itself is structured according to the schema.

Information moves through transformations.

- Different situations require different data structures.
- Most common transformation: *querying*:
  - A query transforms data from one structure (schema) to another.
  - The result of a query is a schema with only one table.
- Other transformations: ETL, schema evolution, warehousing

Think vector spaces and linear transformations.

What is a database?

Information rests primarily in databases.

What is a database?

Information rests primarily in databases.

- A database consists of a bunch of interlocking tables.
- Each table represents some sort of *entity*:
  - Its rows represent examples of that entity.
  - Its columns represents aspects of that entity.

What is a database?

Information rests primarily in databases.

- A database consists of a bunch of interlocking tables.
- Each table represents some sort of *entity*:
  - Its rows represent examples of that entity.
  - Its columns represents aspects of that entity.
- Example: name and owner are aspects of a house-cat.
  - The house-cat is an entity.
  - The house-cat table has a name column.
  - The house-cat table has an owner column.
  - A house-cat owner is a person, an entity of type person.

House-cat	Name	Owner
C101	Prince Charming	P52
C241	Patches	P52
C468	Mittens	P81

Person	Name
P17	Alice
P52	Bob
P81	Carl

The house-cat schema

Domain knowledge:

- House-cats have names and owners;
- owners are people; and
- people have names.

The house-cat schema

Domain knowledge:

- House-cats have names and owners;
- owners are people; and
- people have names.

The database collects worldly examples of this knowledge:

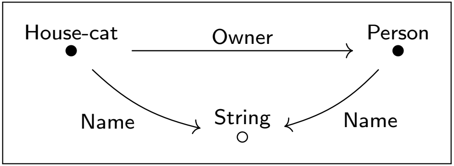
House-cat	Name	Owner
C101	Prince Charming	P52
C241	Patches	P52
C468	Mittens	P81

Person	Name
P17	Alice
P52	Bob
P81	Carl

String
Mittens
Patches
⋮

The schema for this knowledge can be drawn as a graph:

$S :=$

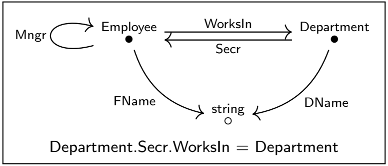


Each column connects its table to another "foreign" table.

A bit more interesting

Let's add loops and integrity constraints:

mySchema :=



Employee	FName	WorksIn	Mngr
1	Alan	101	2
2	Ruth	101	2
3	Kris	102	3

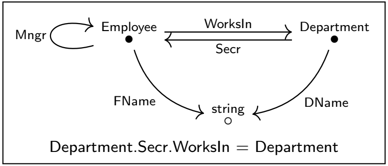
Department	DName	Secr
101	Sales	1
102	IT	3

String
Alan
IT
⋮

A bit more interesting

Let's add loops and integrity constraints:

mySchema :=



Employee	FName	WorksIn	Mngr
1	Alan	101	2
2	Ruth	101	2
3	Kris	102	3

Department	DName	Secr
101	Sales	1
102	IT	3

String
Alan
IT
⋮

Stats:

- Three dots, three tables, three ID columns.
- Five arrows, five non-ID columns.

What is data transformation?

Data transformation is changing the form of data.

- When data is resting in one form, but you want it in another.

What is data transformation?

Data transformation is changing the form of data.

- When data is resting in one form, but you want it in another.
- Examples: querying!

What is data transformation?

Data transformation is changing the form of data.

- When data is resting in one form, but you want it in another.
- Examples: querying, ETL, warehousing, converting, evolving, ...

Alice might want to know who her department secretary is.

What is data transformation?

Data transformation is changing the form of data.

- When data is resting in one form, but you want it in another.
- Examples: querying, ETL, warehousing, converting, evolving, ...

Alice might want to know who her department secretary is.

- That's a query.

```
FOR    e:Employee
WHERE  e.FName = Alice
RETURN e.WorksIn.Secr.FName
```

What is data transformation?

Data transformation is changing the form of data.

- When data is resting in one form, but you want it in another.
- Examples: querying, ETL, warehousing, converting, evolving, …

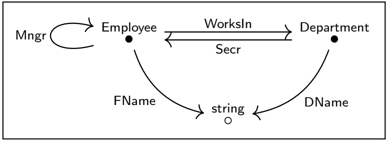
Alice might want to know who her department secretary is.

- That's a query.

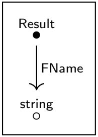
```
FOR    e:Employee
WHERE  e.FName = Alice
RETURN e.WorksIn.Secr.FName
```

- It's a way of transforming data: form A to form B:

$A :=$



$B :=$



What's the problem?

Data transformation is poorly understood in IT culture.

What's the problem?

Data transformation is poorly understood in IT culture.

- Linear algebra analogy:
  - Imagine using vector spaces but not linear transformations.
  - No idea about matrices, eigenvalues, PCA, etc.
  - There's a lot of room for improvement.

What's the problem?

Data transformation is poorly understood in IT culture.

- Linear algebra analogy:
  - Imagine using vector spaces but not linear transformations.
  - No idea about matrices, eigenvalues, PCA, etc.
  - There's a lot of room for improvement.
- Information management is perhaps the biggest problem today.
  - Calculus and diff. eq.? We can hire people to do that.
  - But 40% of IT budgets are spent on information integration.
  - We're constantly breaking and reviving Humpty Dumpty.
  - IT culture has a poor understanding of data transformations.
  - IT culture doesn't even seem to name this problem explicitly.

What's the problem?

Data transformation is poorly understood in IT culture.

- Linear algebra analogy:
  - Imagine using vector spaces but not linear transformations.
  - No idea about matrices, eigenvalues, PCA, etc.
  - There's a lot of room for improvement.
- Information management is perhaps the biggest problem today.
  - Calculus and diff. eq.? We can hire people to do that.
  - But 40% of IT budgets are spent on information integration.
  - We're constantly breaking and reviving Humpty Dumpty.
  - IT culture has a poor understanding of data transformations.
  - Not knowing it has this problem makes the problem insidious.

What's the problem?

Data transformation is poorly understood in IT culture.

- Linear algebra analogy:
  - Imagine using vector spaces but not linear transformations.
  - No idea about matrices, eigenvalues, PCA, etc.
  - There's a lot of room for improvement.
- Information management is perhaps the biggest problem today.
  - Calculus and diff. eq.? We can hire people to do that.
  - But 40% of IT budgets are spent on information integration.
  - We're constantly breaking and reviving Humpty Dumpty.
  - IT culture has a poor understanding of data transformations.
  - Not knowing it has this problem makes the problem insidious.
- If science needs math, what math underlies data science?
  - A huge opportunity to clean up our information Dumpty problem.
  - To properly handle information, we must understand it formally.
  - AI beyond ML requires information agility.

What's the problem?

Data transformation is poorly understood in IT culture.

- Linear algebra analogy:
  - Imagine using vector spaces but not linear transformations.
  - No idea about matrices, eigenvalues, PCA, etc.
  - There's a lot of room for improvement.
- Information management is perhaps the biggest problem today.
  - Calculus and diff. eq.? We can hire people to do that.
  - But 40% of IT budgets are spent on information integration.
  - We're constantly breaking and reviving Humpty Dumpty.
  - IT culture has a poor understanding of data transformations.
  -
- If science needs math, what math underlies data science?
  - A huge opportunity to clean up our information Dumpty problem.
  - To properly handle information, we must understand it formally.
  - AI beyond ML requires information agility.

Let's talk math.

Outline

An invitation to engage with us, and solve real-world problems.

Definition of a category I: Constituents

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

Definition of a category I: Constituents

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

1. A class $\text{Ob}(\mathcal{C})$, called *the objects of* $\mathcal{C}$.
   - Objects $x \in \text{Ob}(\mathcal{C})$ are written as $\bullet^x$.

Definition of a category I: Constituents

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

1. A class $\mathrm{Ob}(\mathcal{C})$, called *the objects of* $\mathcal{C}$.
   - Objects $x \in \mathrm{Ob}(\mathcal{C})$ are written as $\bullet^x$.
2. A set $\mathrm{Arr}(\mathcal{C})$, called *the set of arrows of* $\mathcal{C}$, and two functions

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

   assigning to each arrow its *source* and its *target* object, respectively.
   - An arrow $f \in \mathrm{Arr}(\mathcal{C})$ is written $\bullet^x \xrightarrow{f} \bullet^y$, where $x = src(f)$, $y = tgt(f)$.

Definition of a category I: Constituents

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

1. A class $\mathrm{Ob}(\mathcal{C})$, called *the objects of* $\mathcal{C}$.
   - Objects $x \in \mathrm{Ob}(\mathcal{C})$ are written as $\bullet^x$.
2. A set $\mathrm{Arr}(\mathcal{C})$, called *the set of arrows of* $\mathcal{C}$, and two functions

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

   assigning to each arrow its *source* and its *target* object, respectively.
   - An arrow $f \in \mathrm{Arr}(\mathcal{C})$ is written $\bullet^x \xrightarrow{f} \bullet^y$, where $x = src(f)$, $y = tgt(f)$.
   - A *path in* $\mathcal{C}$ is a finite "head-to-tail" sequence $f_1 \fatsemi \cdots \fatsemi f_n$ of arrows

$$\bullet^{x_0} \xrightarrow{f_1} \bullet^{x_1} \xrightarrow{f_2} \cdots \xrightarrow{f_n} \bullet^{x_n}.$$

   - Can have $n = 1$ (one arrow), or $n = 0$ (just a node).

Definition of a category I: Constituents

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

1. A class $\mathsf{Ob}(\mathcal{C})$, called *the objects of* $\mathcal{C}$.
   - Objects $x \in \mathsf{Ob}(\mathcal{C})$ are written as $\bullet^{x}$.
2. A set $\mathsf{Arr}(\mathcal{C})$, called *the set of arrows of* $\mathcal{C}$, and two functions

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

   assigning to each arrow its *source* and its *target* object, respectively.
   - An arrow $f \in \mathsf{Arr}(\mathcal{C})$ is written $\bullet^{x} \xrightarrow{f} \bullet^{y}$, where $x = src(f)$, $y = tgt(f)$.
   - A *path in* $\mathcal{C}$ is a finite "head-to-tail" sequence $f_1 \fatsemi \cdots \fatsemi f_n$ of arrows

   $$\bullet^{x_0} \xrightarrow{f_1} \bullet^{x_1} \xrightarrow{f_2} \cdots \xrightarrow{f_n} \bullet^{x_n}.$$

   - Can have $n = 1$ (one arrow), or $n = 0$ (just a node).
3. An notion of equivalence for paths, denoted $\simeq$.

Definition of a category II: Rules

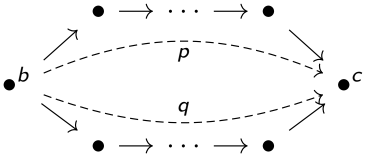
These constituents must satisfy the following requirements:

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: $tgt(p) = tgt(q)$.

Definition of a category II: Rules

These constituents must satisfy the following requirements:

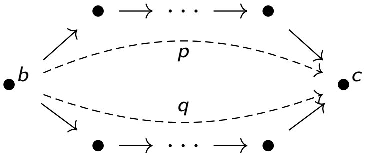
1. If $p \simeq q$ are equivalent paths then the sources agree: $src(p) = src(q)$.
2. If $p \simeq q$ are equivalent paths then the targets agree: $tgt(p) = tgt(q)$.
3. Suppose we have two paths (of any lengths) $b \to c$:



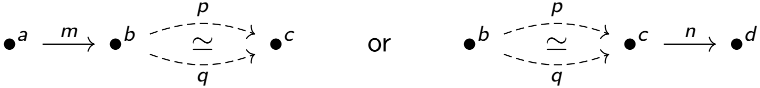
Definition of a category II: Rules

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) $b \to c$:



If $p \simeq q$ then for any extensions



We have equivalences: $m \mathbin{⨟} p \simeq m \mathbin{⨟} q$ and $p \mathbin{⨟} n \simeq q \mathbin{⨟} n$.

Categories = database schemas

- Database schemas are categories!
  - The objects of the category $\mathcal{C}$ are tables.
  - The arrows of $\mathcal{C}$ are columns, connecting one table to another.
  - The integrity constraints are path equations.
  - We brush some details under the rug (distinction between $\bullet$ and $\circ$).

Categories = database schemas

- Database schemas are categories!
  - The objects of the category $\mathcal{C}$ are tables.
  - The arrows of $\mathcal{C}$ are columns, connecting one table to another.
  - The integrity constraints are path equations.
  - We brush some details under the rug.
- But there are also categories that are well-known in math.

Categories from mathematics

Categories are everywhere in mathematics.

- The category **Set** of sets:

Categories from mathematics

Categories are everywhere in mathematics.

- The category **Set** of sets:
  - Objects = all sets
  - arrows $S \to T$ = all functions from S to T
  - paths = composable functions $S_0 \to S_1 \to \cdots \to S_n$
  - paths equivalent $\iff$ same composite function

Categories from mathematics

Categories are everywhere in mathematics.

- The category **Set** of sets:
  - Objects = all sets
  - arrows $S \to T$ = all functions from S to T
  - paths = composable functions $S_0 \to S_1 \to \cdots \to S_n$
  - paths equivalent $\iff$ same composite function
- The category **Vect** of vector spaces
  - Objects = vector spaces
  - arrows = linear transformations....

Categories from mathematics

Categories are everywhere in mathematics.

- The category **Set** of sets:
  - Objects = all sets
  - arrows $S \to T$ = all functions from S to T
  - paths = composable functions $S_0 \to S_1 \to \cdots \to S_n$
  - paths equivalent $\iff$ same composite function
- The category **Vect** of vector spaces
  - Objects = vector spaces
  - arrows = linear transformations....
- The category of measurable spaces
- The category of metric or topological spaces,
- The category **Cat** of categories.

Categories from mathematics

Categories are everywhere in mathematics.

- The category **Set** of sets:
  - Objects = all sets
  - arrows $S \to T$ = all functions from S to T
  - paths = composable functions $S_0 \to S_1 \to \cdots \to S_n$
  - paths equivalent $\iff$ same composite function
- The category **Vect** of vector spaces
  - Objects = vector spaces
  - arrows = linear transformations....
- The category of measurable spaces
- The category of metric or topological spaces,
- The category **Cat** of categories.

There's also a notion of *mapping* between categories: functors.

Functors: mappings between categories

What's a functor?

Functors: mappings between categories

What's a functor?

- Recall: a category is a directed graph with path equivalences.
- A functor is a graph mapping that respects path equivalence.

Functors: mappings between categories

What's a functor?

- Recall: a category is a directed graph with path equivalences.
- A functor is a graph mapping that respects path equivalence.

- **Definition**: A functor $F\colon \mathcal{C} \to \mathcal{D}$ consists of
  - a function $\text{Ob}(\mathcal{C}) \to \text{Ob}(\mathcal{D})$ and
  - a function $\text{Arr}(\mathcal{C}) \to \mathrm{Path}(\mathcal{D})$,

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

Functors and databases

Recall:

- A category is a directed graph with path equivalences.
- A functor $\mathcal{C} \to \mathcal{D}$ is a mapping that preserves these structures.

Functors and databases

Recall:

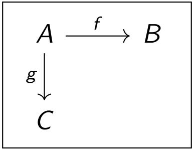
- A category is a directed graph with path equivalences.
- A functor $\mathcal{C} \to \mathcal{D}$ is a mapping that preserves these structures.
- **Set** is a category; recall
  - its objects are all sets
  - its arrows $S \to T$ are functions, and
  - two paths are equivalent if they compose to the same function.

Functors and databases

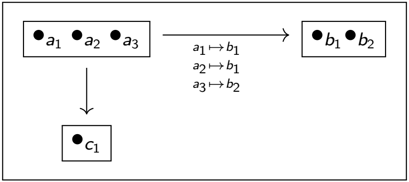
Recall:

- A category is a directed graph with path equivalences.
- A functor $\mathcal{C} \to \mathcal{D}$ is a mapping that preserves these structures.
- **Set** is a category; recall
  - its objects are all sets
  - its arrows $S \to T$ are functions, and
  - two paths are equivalent if they compose to the same function.
- A functor $\mathcal{C} \to$ **Set** fills schema $\mathcal{C}$ with data.
  - Example: Let $\mathcal{C}$ be the category on the left.
  - Then here's an example functor $I : \mathcal{C} \to$ **Set** :

$\mathcal{C} :=$

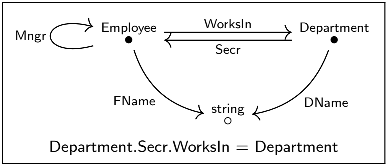


$I :=$



Schema=Category, Instance=Set-valued functor

- Let $\mathcal{C}$ be the following category

$\mathcal{C} :=$ 

- A functor $I \colon \mathcal{C} \to \mathbf{Set}$ consists of
  - A set for each object of $\mathcal{C}$ and
  - a function for each arrow of $\mathcal{C}$, such that
  - the declared equations hold.
- In other words, I fills the schema with compatible data.

Summary of the connection

- The connection between categories and databases is simple.
- A database schema is a custom category.
- Functors $I: \mathcal{C} \to \mathbf{Set}$ are database instances.
- What about functors $F: \mathcal{C} \to \mathcal{D}$ between schemas?

Data transformations

We want to move data between different frameworks.

- Data is resting in schema $\mathcal{C}$.
- We want to move it in a specific way to schema $\mathcal{D}$.
- We can specify this transformation using functors.

Functorial data migration

We can do all sorts of data transformations using functors.

- Queries, ETL processes, warehousing, schema evolution, etc.

Functorial data migration

We can do all sorts of data transformations using functors.

- Queries, ETL processes, warehousing, schema evolution, etc.
- A functor $\mathcal{C} \to \mathcal{D}$
  - sends nodes to nodes,
  - sends arrows to paths, and
  - respects path equivalence.

Functorial data migration

We can do all sorts of data transformations using functors.

- Queries, ETL processes, warehousing, schema evolution, etc.
- A functor $\mathcal{C} \to \mathcal{D}$
  - sends nodes to nodes,
  - sends arrows to paths, and
  - respects path equivalence.
- But functors play two roles here:
  - They connect schemas to schemas, $\mathcal{C} \to \mathcal{D}$
  - They connect schemas to data, $\mathcal{D} \to \mathbf{Set}$.
  - Upshot: one can compose and get a functor $\mathcal{C} \to \mathbf{Set}$.

The category of instances

- Given a schema $\mathcal{C}$, the *category of instances* on $\mathcal{C}$ is denoted $\mathcal{C}$-**Inst**.
  - The objects of $\mathcal{C}$-**Inst** are functors (instances) $I: \mathcal{C} \to \mathbf{Set}$.
  - The arrows are insertions and deduplications of data.

The category of instances

- Given a schema $\mathcal{C}$, the *category of instances* on $\mathcal{C}$ is denoted $\mathcal{C}$-**Inst**.
  - The objects of $\mathcal{C}$-**Inst** are functors (instances) $I: \mathcal{C} \to \mathbf{Set}$.
  - The arrows are insertions and deduplications of data.
- Functors between schemas allow us to move data between them.
  - Given a functor $F: \mathcal{C} \to \mathcal{D}$,
  - There are automatically three data transformation functors

$$\mathcal{C}\text{-}\mathbf{Inst} \; \underset{\Pi_F}{\overset{\Sigma_F}{\underset{\longrightarrow}{\overset{\longrightarrow}{\longleftarrow \Delta_F \longrightarrow}}}} \; \mathcal{D}\text{-}\mathbf{Inst}$$

  - Δ, Σ, Π were known by mathematicians in 1960. "Kan extensions".
  - But they had no idea Δ, Σ, Π correspond to database transformations.

The category of instances

- Given a schema $\mathcal{C}$, the *category of instances* on $\mathcal{C}$ is denoted $\mathcal{C}$-**Inst**.
  - The objects of $\mathcal{C}$-**Inst** are functors (instances) $I\colon \mathcal{C} \to \mathbf{Set}$.
  - The arrows are insertions and deduplications of data.
- Functors between schemas allow us to move data between them.
  - Given a functor $F\colon \mathcal{C} \to \mathcal{D}$,
  - There are automatically three data transformation functors

$$\mathcal{C}\text{-}\mathbf{Inst} \; \underset{\Pi_F}{\overset{\Sigma_F}{\underset{\longrightarrow}{\overset{\longrightarrow}{\longleftarrow \Delta_F \longrightarrow}}}} \; \mathcal{D}\text{-}\mathbf{Inst}$$

  - Δ, Σ, Π were known by mathematicians in 1960. "Kan extensions".
  - But they had no idea Δ, Σ, Π correspond to database transformations.

There's a lot of mathematics ready made for moving data.

Outline

An invitation to engage with us, and solve real-world problems.

The history of AQL

- The mathematical foundations of this story are old.
  - The basic idea was known to mathematicians 60 years ago.
  - More recently we've learned a lot about how to calculate them fast.
  - Getting data types ($\circ^{\text{string}}$) into the picture is a little more delicate.

The history of AQL

- The mathematical foundations of this story are old.
  - The basic idea was known to mathematicians 60 years ago.
  - More recently we've learned a lot about how to calculate them fast.
  - Getting data types ($\circ^{\text{string}}$) into the picture is a little more delicate.
- I realized the connection to data transformations in around 2008.
  - I hired Ryan Wisnesky, a Harvard CS grad student, to code it.
  - Ryan joined me as a postdoc at MIT.
  - The math was finally completed in 2015.
  - I learned a lot about the difference between math and code.

The history of AQL

- The mathematical foundations of this story are old.
  - The basic idea was known to mathematicians 60 years ago.
  - More recently we've learned a lot about how to calculate them fast.
  - Getting data types ($\circ^{\text{string}}$) into the picture is a little more delicate.
- I realized the connection to data transformations in around 2008.
  - I hired Ryan Wisnesky, a Harvard CS grad student, to code it.
  - Ryan joined me as a postdoc at MIT.
  - The math was finally completed in 2015.
  - I learned a lot about the difference between math and code.

"Beware of bugs in the above code; I have only proved it correct, not tried it." – Donald Knuth

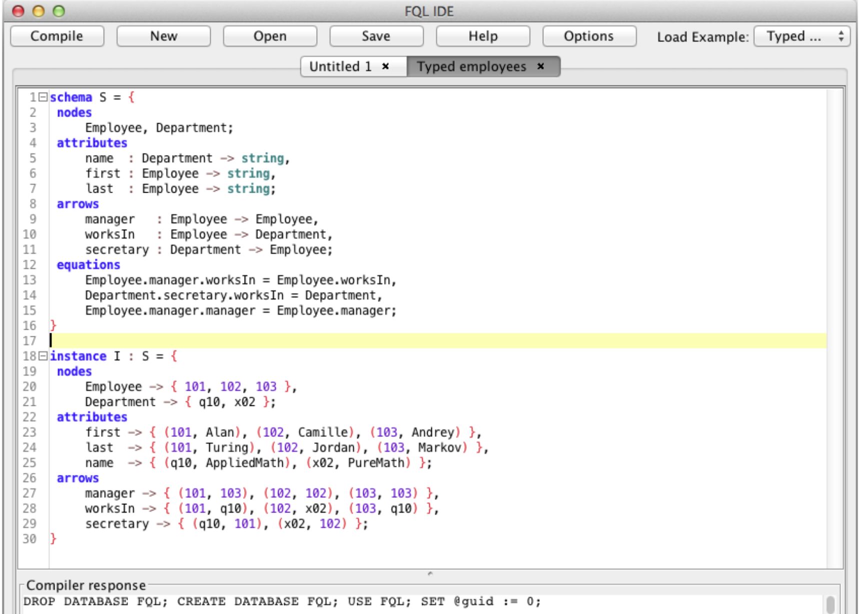
The history of AQL

- The mathematical foundations of this story are old.
  - The basic idea was known to mathematicians 60 years ago.
  - More recently we've learned a lot about how to calculate them fast.
  - Getting data types ($\circ^{\text{string}}$) into the picture is a little more delicate.
- I realized the connection to data transformations in around 2008.
  - I hired Ryan Wisnesky, a Harvard CS grad student, to code it.
  - Ryan joined me as a postdoc at MIT.
  - The math was finally completed in 2015.
  - I learned a lot about the difference between math and code.

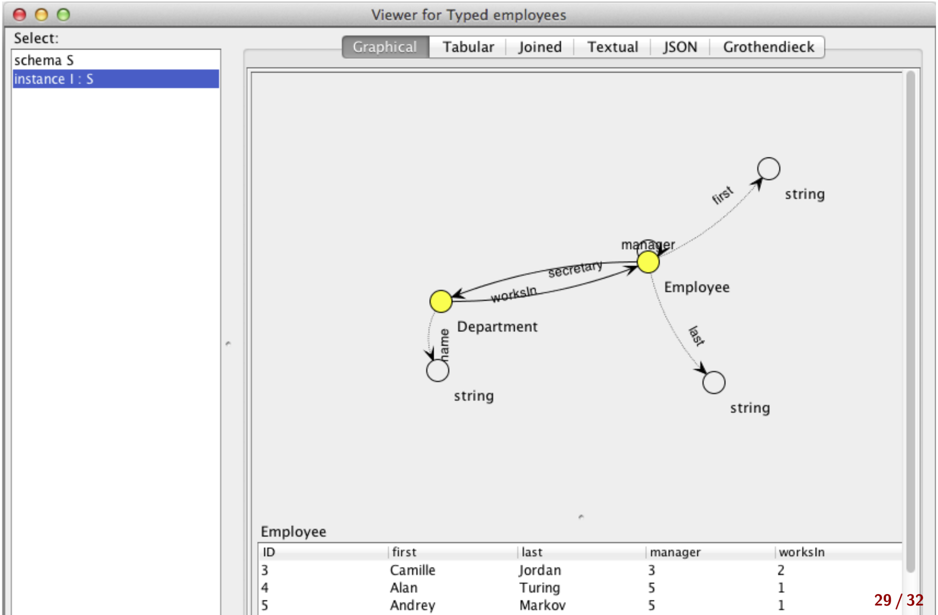
"Beware of bugs in the above code; I have only proved it correct, not tried it." – Donald Knuth

- We received funding from various government agencies.
  - ONR, AFOSR, NIST, NSF.
- A company spun out of MIT in 2015.
  - Categorical Informatics Inc.
  - All MIT IP is open source, all Catinf IP is not.

Screenshot 1



Screenshot 2



ID	first	last	manager	worksIn
3	Camille	Jordan	3	2
4	Alan	Turing	5	1
5	Andrey	Markov	5	1

Outline

An invitation to engage with us, and solve real-world problems.

The bigger picture, again

The point of all that was to give a glimpse into category theory.

- A simple principle—data transformations—formalized mathematically.

The bigger picture, again

The point of all that was to give a glimpse into category theory.

- A simple principle—data transformations—formalized mathematically.
- And this database stuff is just one part of category theory.
  - CT has formalized the principles of mathematics, in mathematics.
  - Space, measure, operation, data, symmetry, equivalence, syntax.
  - There is a web of interconnection between all these principles.

The bigger picture, again

The point of all that was to give a glimpse into category theory.

- A simple principle—data transformations—formalized mathematically.
- And this database stuff is just one part of category theory.
    - CT has formalized the principles of mathematics, in mathematics.
    - Space, measure, operation, data, symmetry, equivalence, syntax.
    - There is a web of interconnection between all these principles.
- CT been recently highlighted by agencies such as NIST and DARPA.
    - Interdisciplinarity in science and engineering is a big problem.
    - Solving it requires understanding the kinematics of information.

Summary of the talk

Once again, CT is *way bigger* than databases.

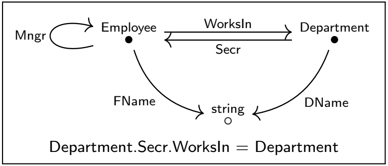
- Tip of iceberg: the connection between databases and categories.

Summary of the talk

Once again, CT is *way bigger* than databases.

- Tip of iceberg: the connection between databases and categories.

mySchema :=



Employee	FName	WorksIn	Mngr
1	Alan	101	2
2	Ruth	101	2
3	Kris	102	3

Department	DName	Secr
101	Sales	1
102	IT	3

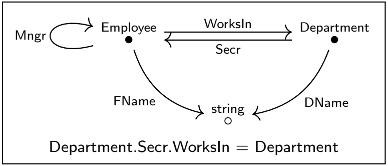
String
Alan
IT
⋮

Summary of the talk

Once again, CT is *way bigger* than databases.

- Tip of iceberg: the connection between databases and categories.

mySchema :=



Employee	FName	WorksIn	Mngr
1	Alan	101	2
2	Ruth	101	2
3	Kris	102	3

Department	DName	Secr
101	Sales	1
102	IT	3

String
Alan
IT
⋮

- Information kinematics—how data moves—is well-modeled by CT.
- Without a good understanding, we waste a lot of time and effort.

Thanks for the invitation to speak!

For more information

Book: *An Invitation to Applied Category Theory: Seven Sketches in Compositionality*. Cambridge University Press, July 2019.
https://arxiv.org/abs/1803.05316

MIT course: Jan 14 – Feb 1, 14:00–15:00, Room 2-143

Company: Categorical Informatics. Website: http://catinf.com