Toward a mathematical science of informatics

David I. Spivak

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

Presented on 2013/09/18
at the Office of Naval Research Review

Introduction

Outline of the talk

1. Introduction.
   - The problem to address.
2. Information structures and categories.
   - 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 Duke.
  - 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 1522359927674611942 4 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?

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 structural reasoning.
- **Information** is governed by this kind of structural reasoning.
  - If that's true, 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 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, etc.
  - Amgen, Microsoft, Honeywell, NIST.
  - "Our gold standard for specifying anything now is that it must be categorical. We are beginning to trust nothing else. [snip] I now understand that knowledge representation can be rigorous and extendable."

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.

Information structures and categories

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.

In Formation

What is in common to information presentations?

- They are in formation.
  - Controlling formation is the same as enforcing order, dispelling chaos.
  - It creates the possibility for roles.
  - It obviates guessing and 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.

Foreign Keys and business rules

- Example:

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

- Note the ID (primary key) columns and the foreign key columns.
- Perhaps we should enforce certain integrity constraints (business rules):
  - The manager of an employee E must be in the same department as E,
  - The secretary of a department D must be in D.

Data columns as foreign keys

- We can consider data columns as foreign keys (to respective 1-column tables).

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

FNString
ID
Alan
Bertrand
Carla
David
⋮

LNString
ID
Ardon
Blithe
⋮
Hilbert
⋮

DNString
ID
Marketing
Production
Sales
Research
⋮

- Conclusion: each column in a table is a key – one primary, the rest foreign.

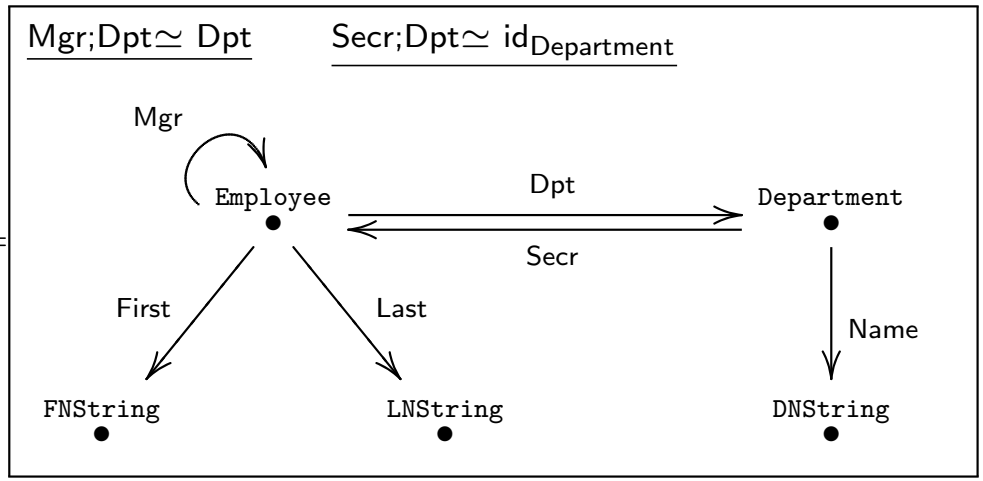
Example again

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

FNString
ID
Alan
Bertrand
Carla
David
.
.
.

$\mathcal{C} :=$



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.

What is a category?

- A category consists of objects, morphisms, and a composition law.
- It is an algebraic object, much like a group.
- Like a group, a category may be *presented* by generators and relations.
- Punchline: one can formulate any database schema as a category presentation.

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

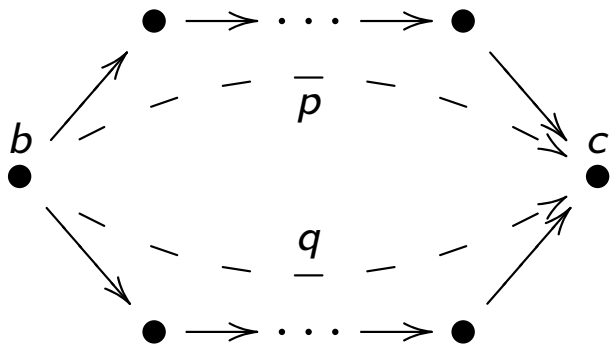
$$src, tgt : \mathbf{Arr}(\mathcal{C}) \to \mathbf{Ob}(\mathcal{C}),$$

   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 $\simeq$.

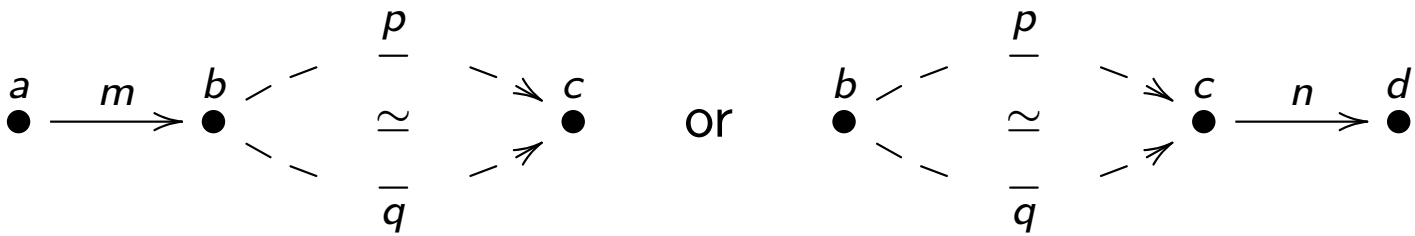
Definition of a category presentation. Part II: Rules

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) $b \to c$:



If $p \simeq q$ then for any extensions

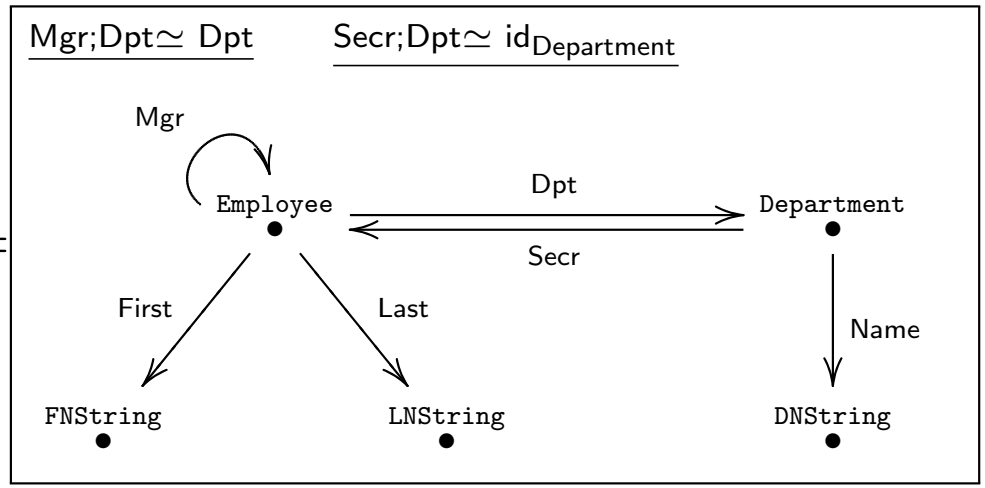


$m; p \simeq m; q$ and $p; n \simeq q; n$.

Our pictures have been category presentations

- Database schemas are category presentations.

$\mathcal{C} :=$



- Other examples of categories:
  - **Set**, the category of sets and functions,
  - **Vect**, the category of vector spaces and linear transformations,
  - **Type**, the category of types and programs in a functional programming language.

Linking disparate information structures using CT

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.

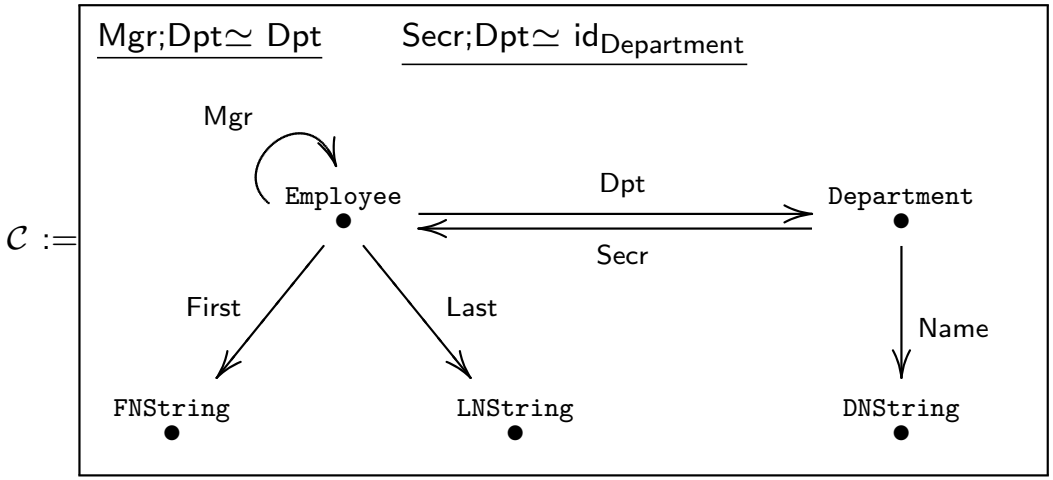
Functors: mappings between categories

- 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 $F : \mathcal{C} \to \mathcal{D}$ consists of
  - a function $\mathbf{Ob}(\mathcal{C}) \to \mathbf{Ob}(\mathcal{D})$ and
  - a function $\mathbf{Arr}(\mathcal{C}) \to \mathbf{Path}(\mathcal{D})$,

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

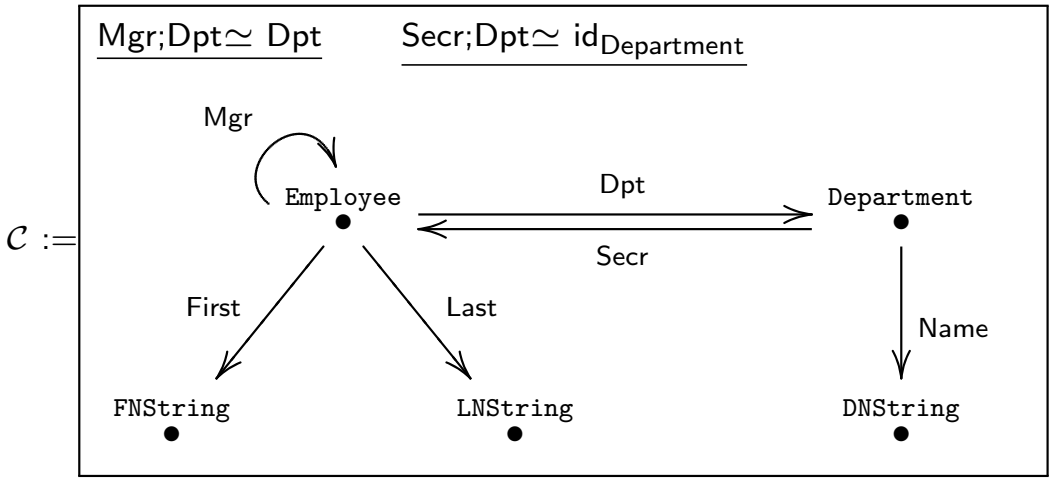
Backing up: a database instance is a functor!

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



- As we said, there is a category **Set** of sets and functions.
- A functor $I : \mathcal{C} \to \mathbf{Set}$ 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 $I : \mathcal{C} \to \mathbf{Set}$ is a database instance on $\mathcal{C}$; i.e. it is a way to fill $\mathcal{C}$ with compatible data.

Example again



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

FNString
ID
Alan
Bertrand
Carla
David
.
.
.

Changes in schema

- We may want to find a link between two schemas $\mathcal{C}$ and $\mathcal{D}$.
- We should find a functorial connection between them.
- Over time we may have something like

$$\mathcal{C} = \mathcal{C}_0 \xrightarrow{F_0} \mathcal{C}_1 \xleftarrow{F_1} \mathcal{C}_2 \xrightarrow{F_3} \cdots \xrightarrow{F_n} \mathcal{C}_n = \mathcal{D}$$

- 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 in predictable ways.

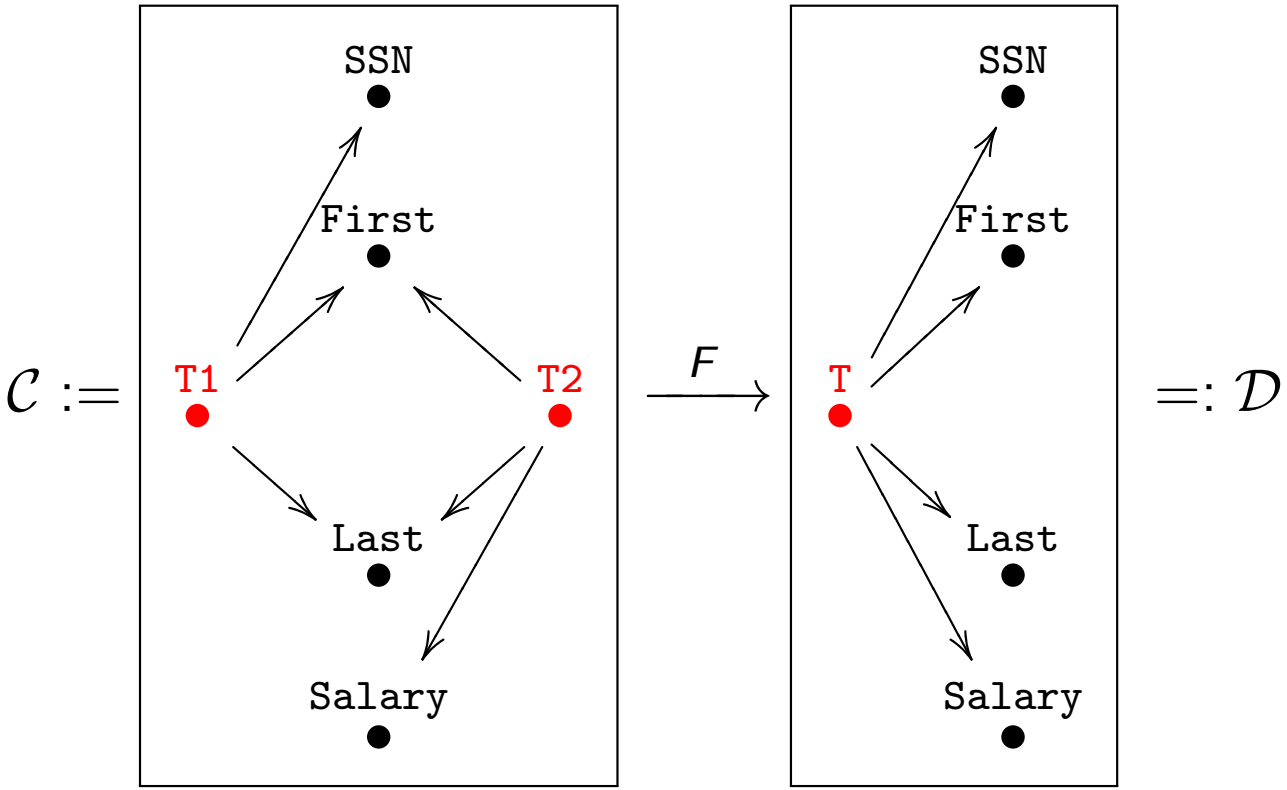
Functorial data migration for CT experts

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

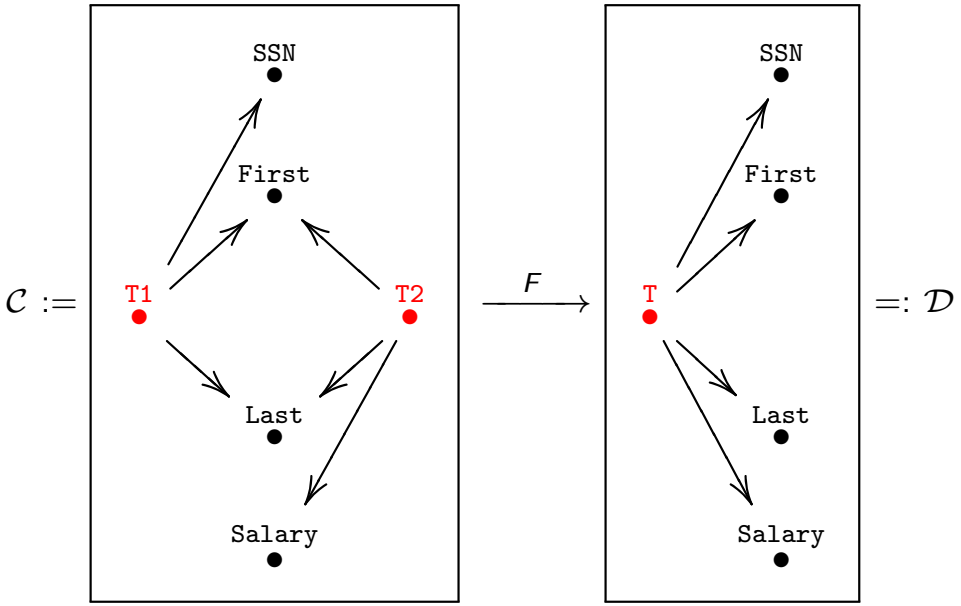
$$\mathcal{C} \xrightarrow{F} \mathcal{D} \xrightarrow{I} \mathbf{Set}.$$

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

Uses of functorial data migration 0: Translation F



Uses of functorial data migration 1: Projection via Δ_F



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

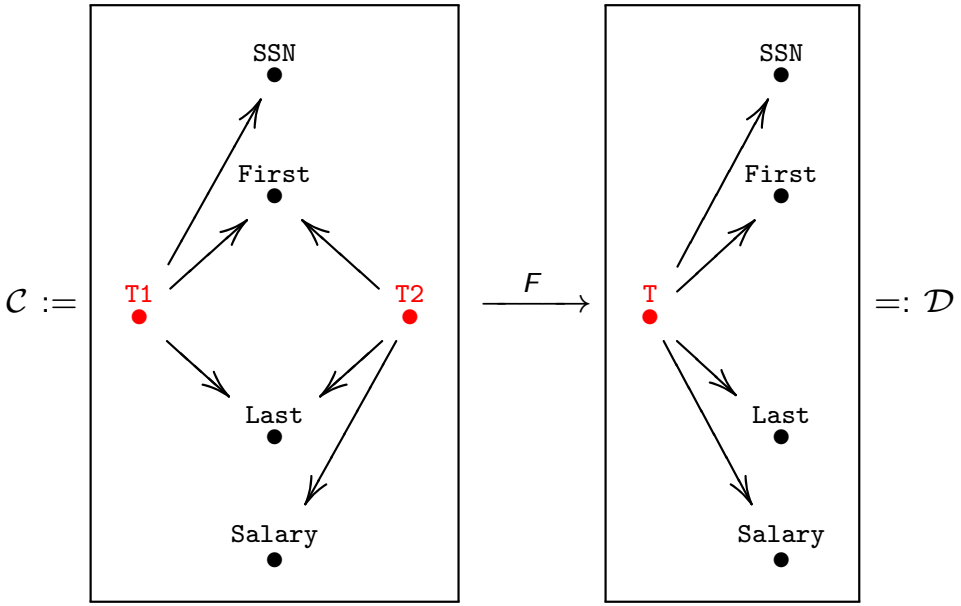
T				
ID	**SSN**	**First**	**Last**	**Salary**
XF667	115-234	Bob	Smith	$250
XF891	122-988	Sue	Smith	$300
XF221	198-877	Alice	Jones	$100

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

T1			
ID	**SSN**	**First**	**Last**
XF667	115-234	Bob	Smith
XF891	122-988	Sue	Smith
XF221	198-877	Alice	Jones

T2			
ID	**First**	**Last**	**Salary**
XF667	Bob	Smith	$250
XF891	Sue	Smith	$300
XF221	Alice	Jones	$100

Uses of functorial data migration 2: Joins via Π_F



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

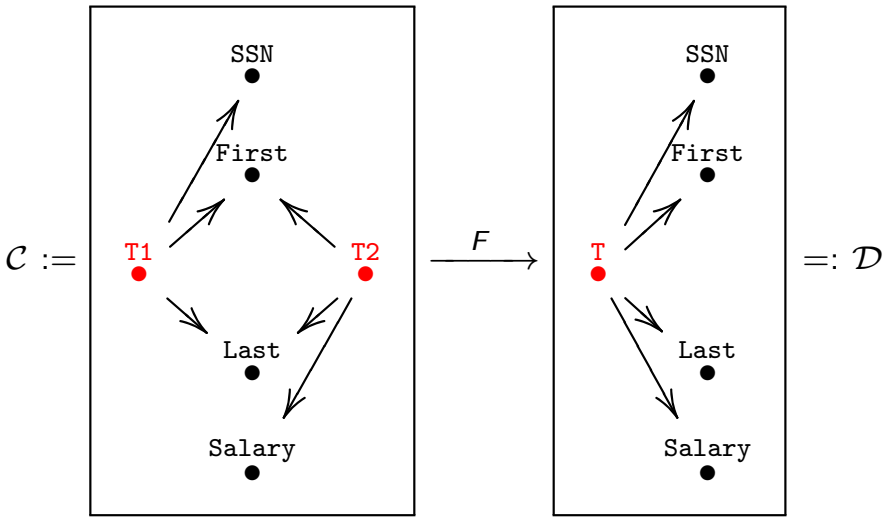
T1			
ID	**SSN**	**First**	**Last**
T1-001	115-234	Bob	Smith
T1-002	122-988	Sue	Smith
T1-003	198-877	Alice	Jones

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

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



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

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

The above ideas have been implemented.

- I'm working with a Harvard CS graduate student named Ryan Wisnesky.
- He has implemented the above data migration story.
  - It's called FQL (Functorial Query Language)
  - Create schemas (category presentations), functors, instances, queries.
  - FQL is available online, and it's open source.

Screenshot 1 of FQL

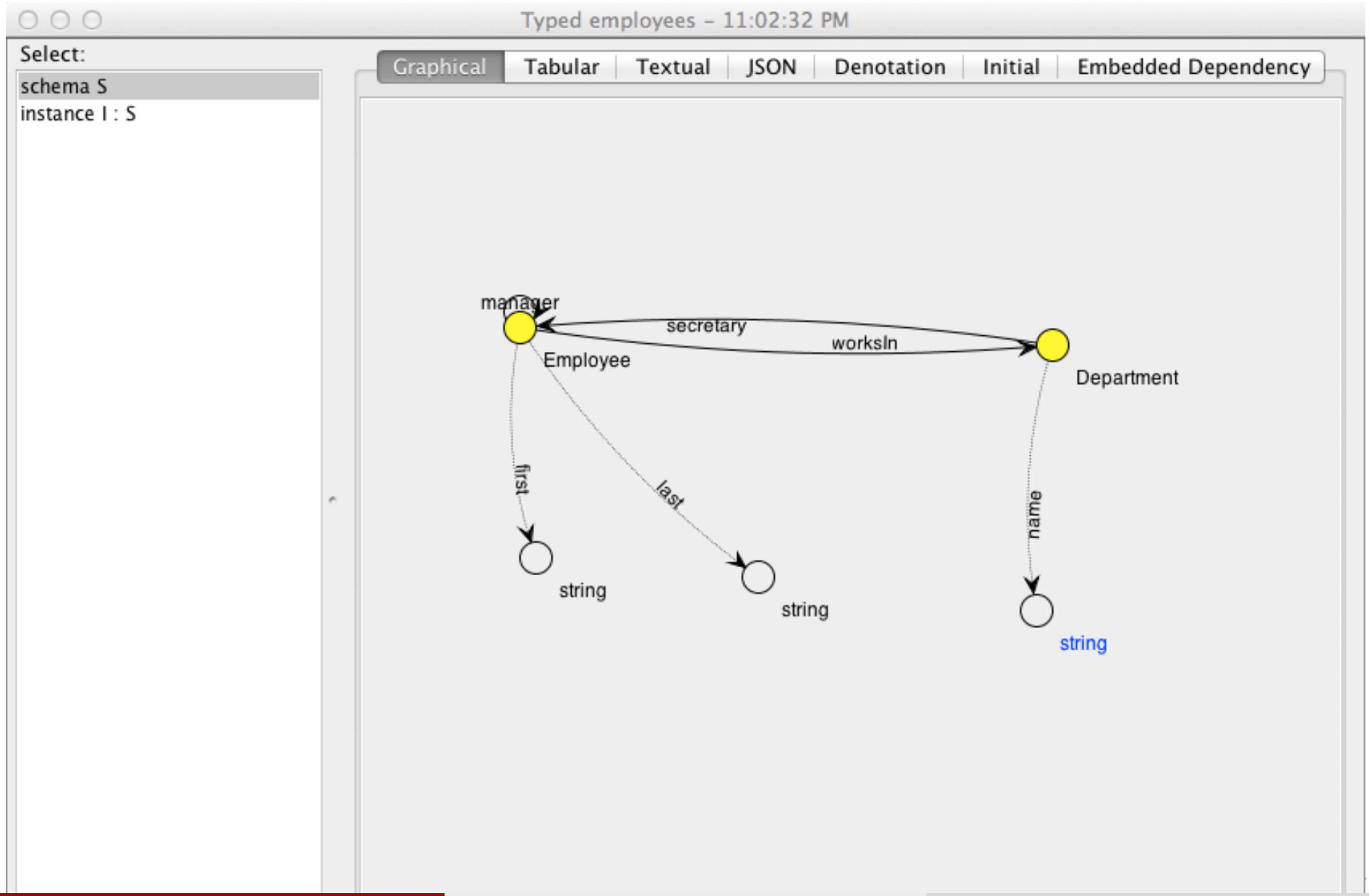
FQL IDE

Compile | New | Open | Save | Help | Options | Load Example: Typed ...

Untitled 1 × | Typed employees ×

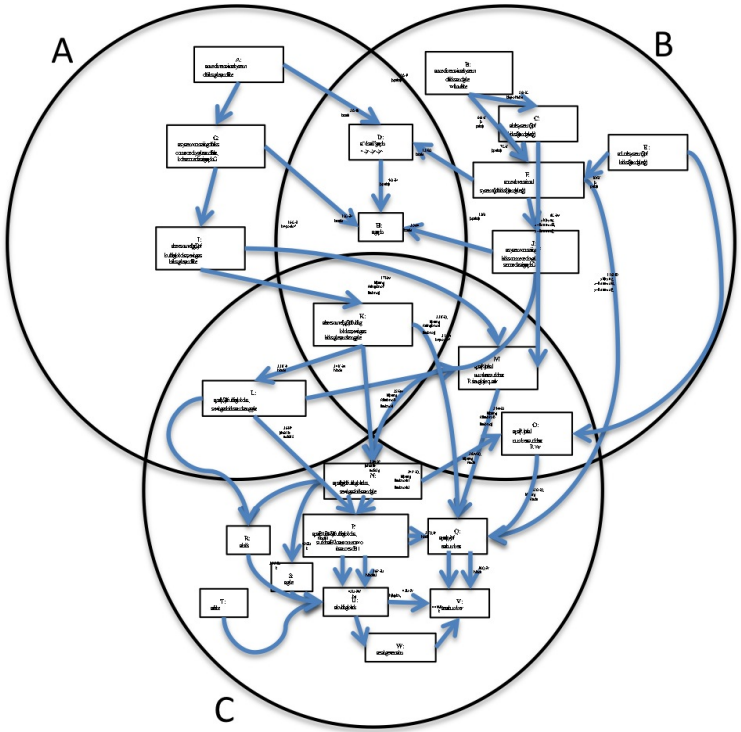
```
1  schema S = {
2   nodes
3       Employee, Department;
4   attributes
5       name  : Department -> string,
6       first : Employee -> string,
7       last  : Employee -> string;
8   arrows
9       manager   : Employee -> Employee,
10      worksIn   : Employee -> Department,
11      secretary : Department -> Employee;
12  equations
13      Employee.manager.worksIn = Employee.worksIn,
14      Department.secretary.worksIn = Department,
15      Employee.manager.manager = Employee.manager;
16  }
17
18 instance I : S = {
19  nodes
20      Employee -> { 101, 102, 103 },
21      Department -> { q10, x02 };
22  attributes
23      first -> { (101, Alan), (102, Camille), (103, Andrey) },
24      last  -> { (101, Turing), (102, Jordan), (103, Markov) },
25      name  -> { (q10, AppliedMath), (x02, PureMath) };
26  arrows
27      manager -> { (101, 103), (102, 102), (103, 103) },
28      worksIn -> { (101, q10), (102, x02), (103, q10) },
29      secretary -> { (q10, 101), (x02, 102) };
30  }
```

Screenshot 2 of FQL

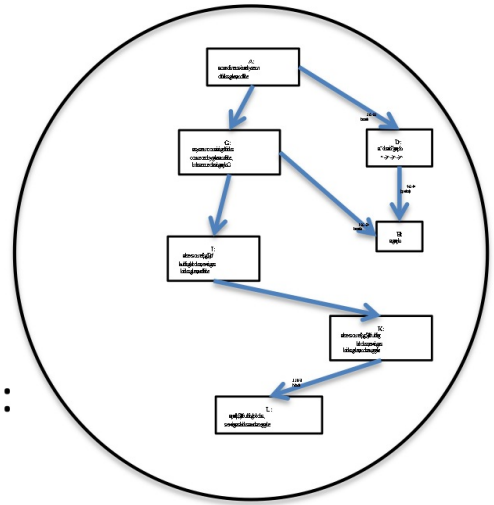


Forming a knowledge network

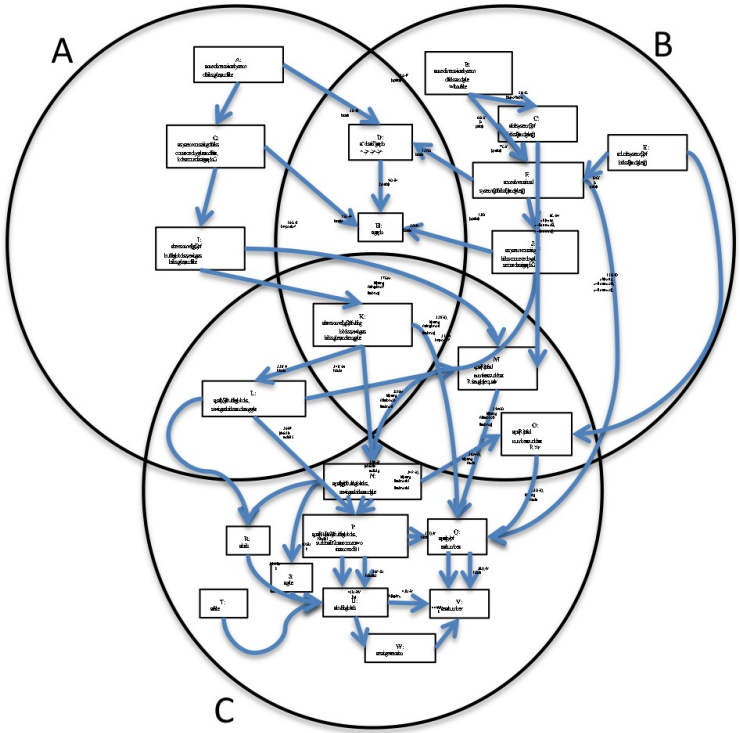
Network of scientists 1: overlapping understanding

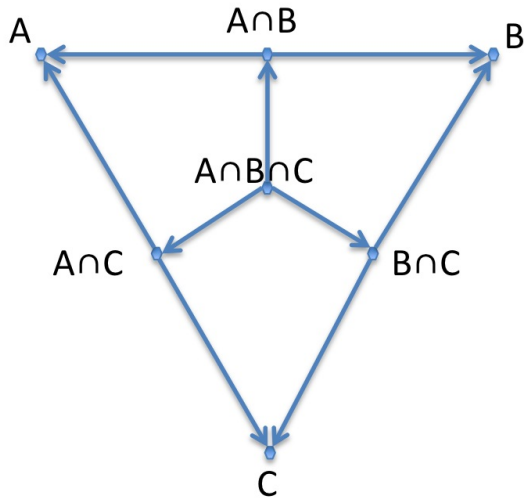


Scientist A's research topic:

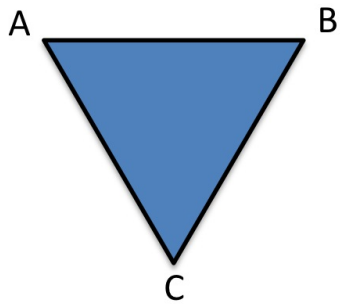


Network of scientists 2: encoding interaction groups



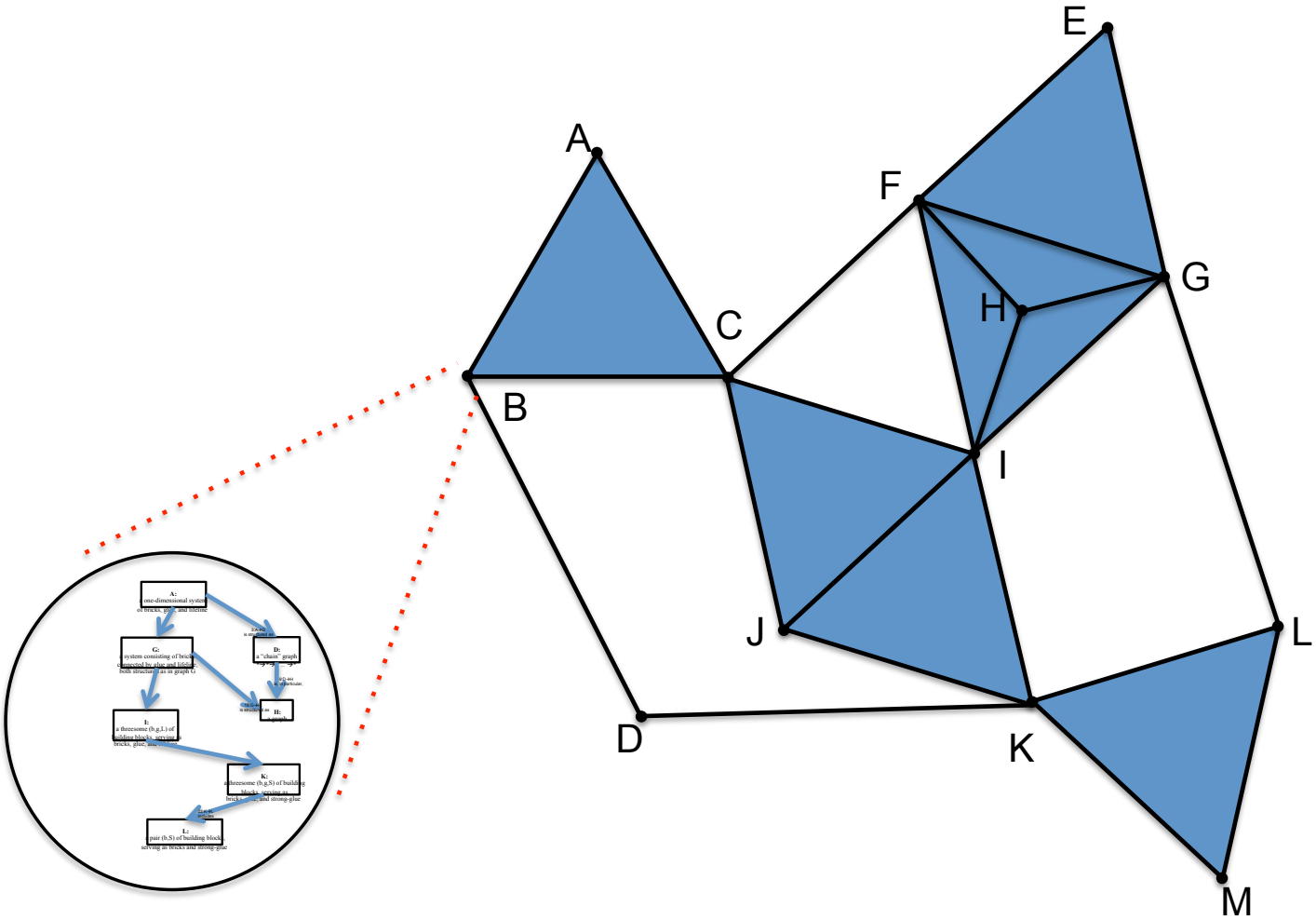


Abstraction:



Network of scientists 3: simplicial complex

A network of database schemas, a network of scientific understanding.



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

Conclusion

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.
- Category theory will 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.

Thank you

Thanks for listening!

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.

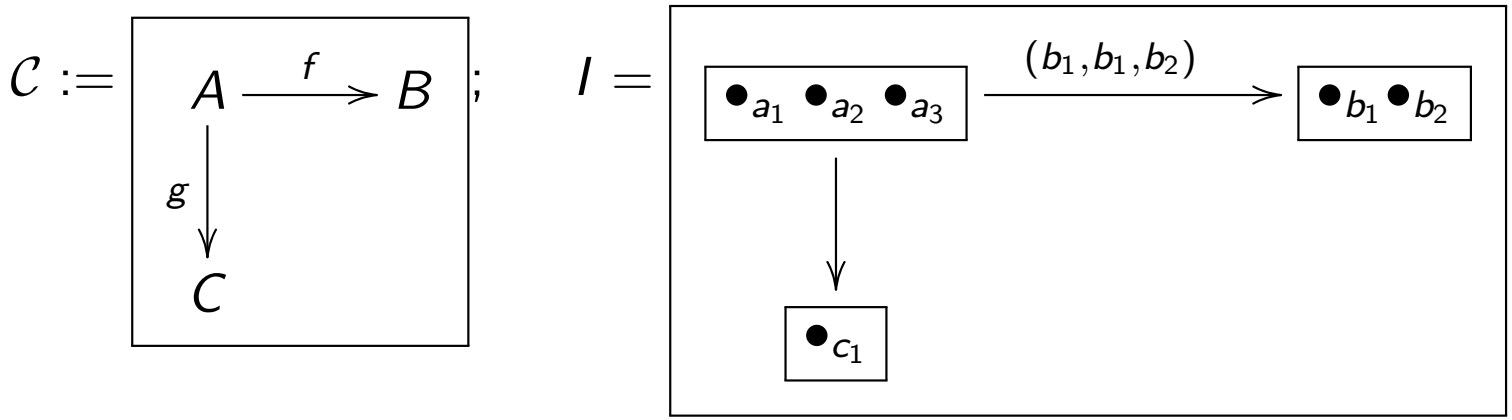
Appendix

Contents:

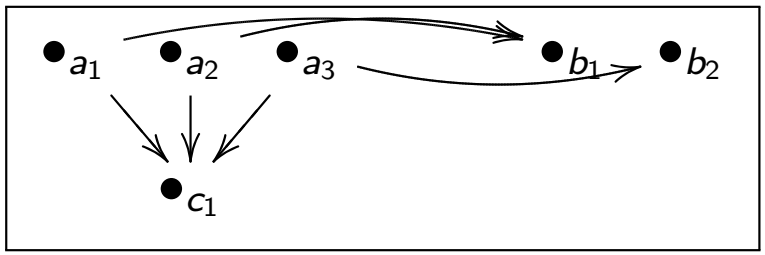
- RDF via the Grothendieck construction.
- A sample SQL query using data migration functors.

The Grothendieck construction

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



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



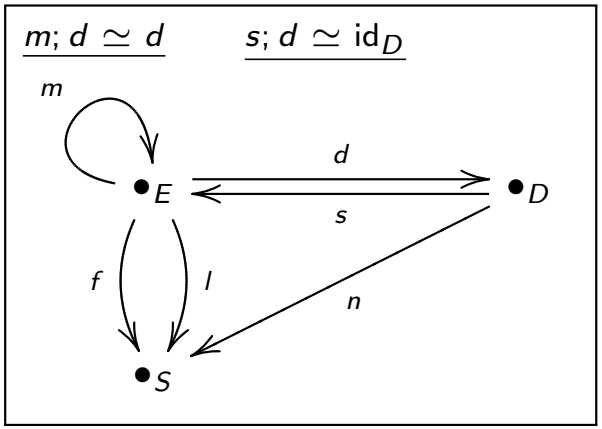
Grothendieck construction applied to database instances

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

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

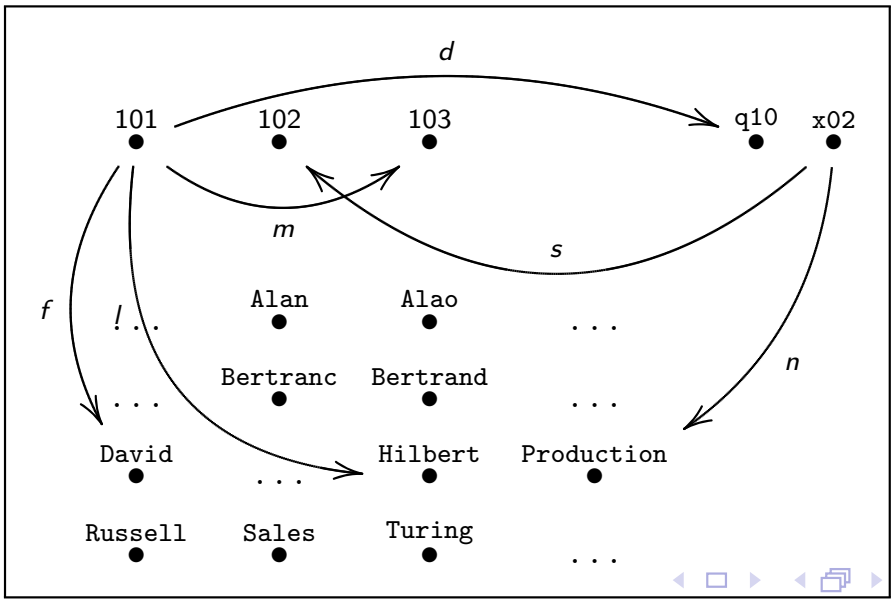
Department		
ID	**Name**	**Secr'y**
q10	Sales	101
x02	Production	102

$\mathcal{C} =$



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

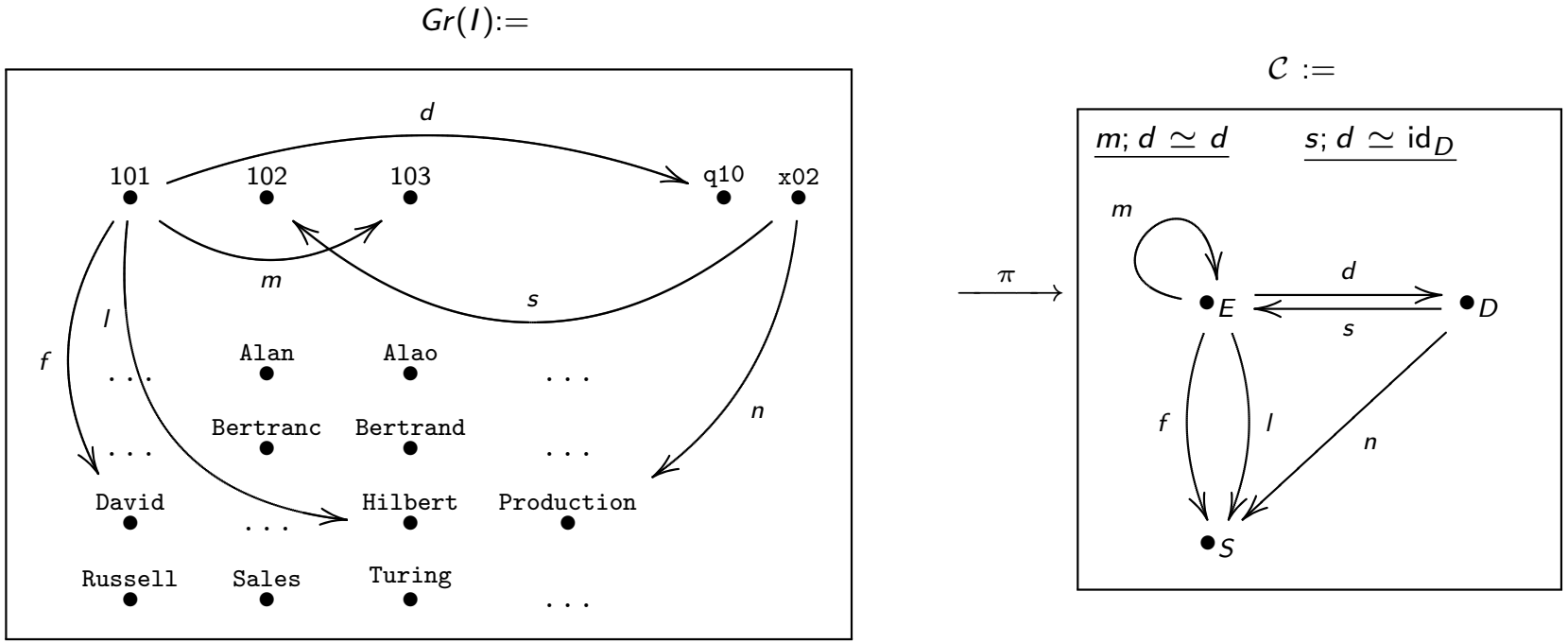
$Gr(I) =$



A different perspective on data

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

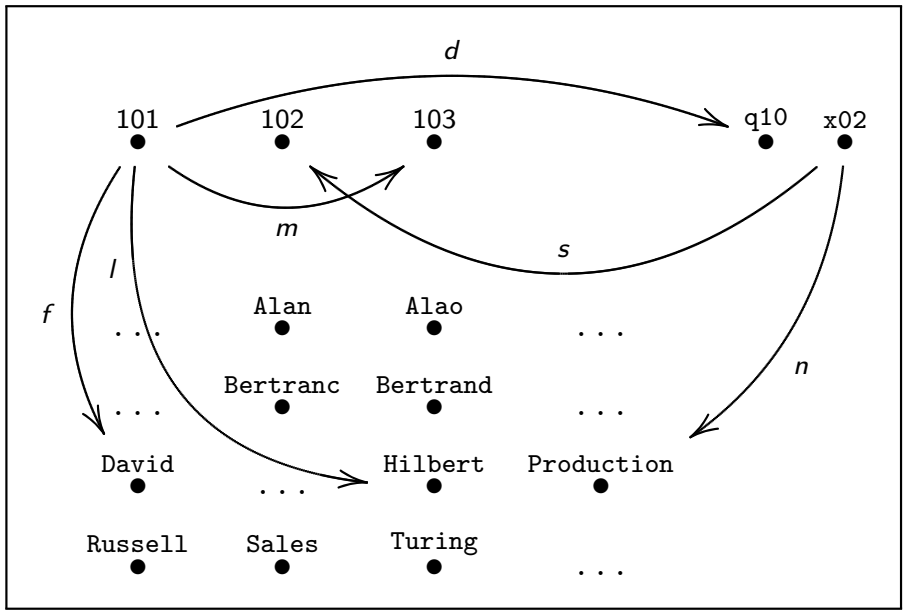
$$\pi : Gr(I) \to \mathcal{C}.$$



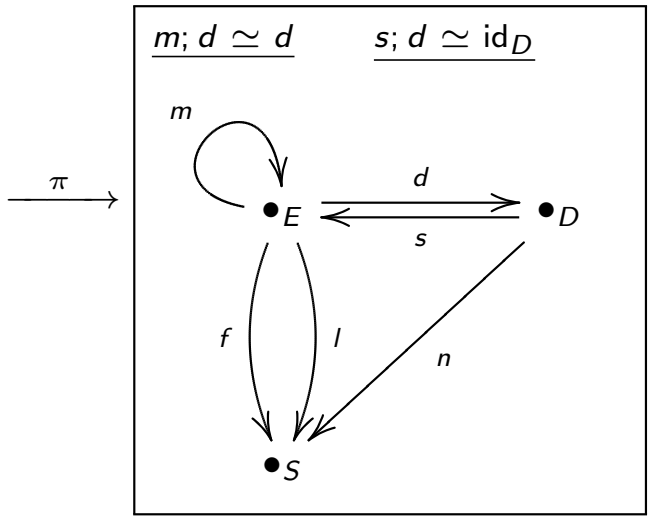
The fiber over (inverse image of) every object $X \in \mathcal{C}$ is a set of objects $\pi^{-1}(X) \subseteq Gr(I)$. That set is $I(X)$.

RDF schema and stores

$Gr(I)=$



$\mathcal{C} =$



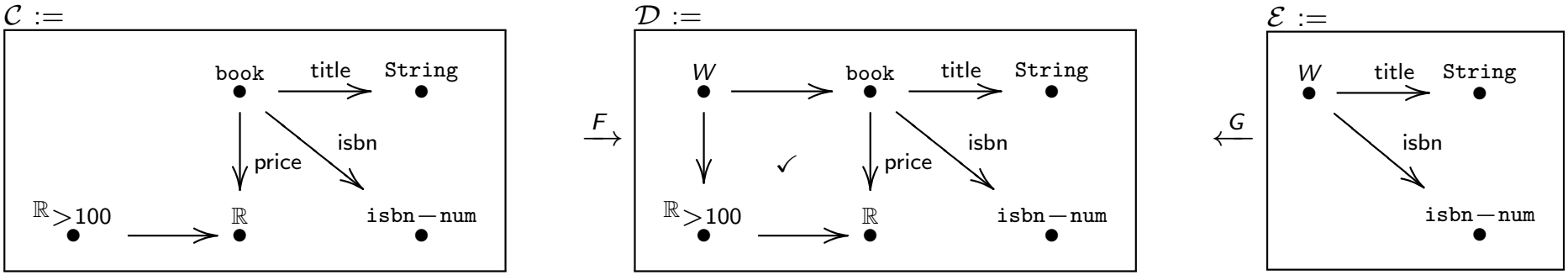
- The relation to RDF triples is clear: each arrow $f : x \to y$ in $Gr(I)$ is a triple with subject x, predicate f, and object y.
- For example (101 department q10), (x02 name Production), etc..
- $\mathcal{C}$ is the RDF schema and $Gr(I)$ is the triple store.
- SPARQL queries (graph patterns) are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

A best schema for data?

- Question: given RDF data, D, is there a "best schema" for it?
  - That is, a schema $\mathcal{C}$ and an instance I, such that $Gr(I) = D$?
- Technical rephrase: given a category D, does there exist a terminal object in the category of discrete op-fibrations $D \to X$?
- One can prove that the answer is no.

A simple "SELECT" query using functors

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



- $V := \Delta_G \circ \Pi_F$ 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.