Title: Lifting problems and data

David I. Spivak

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

Presented on 2012/08/04
at MathFest

Research goal

- My goal is to develop a rigorous understanding of information:
  - what it is,
  - how it is used,
  - how it can be faithfully transmitted.
- Databases are information storage devices.
- They are suitable objects for mathematical investigation.

We can model databases as categories

- I'll show that databases and categories are fundamentally similar.
- I'll explain how functors connect different databases.
    - Data can *migrate* along a functor from one database to another.
    - Typical sheaf operations yield typical database operations.
- I'll discuss how lifting problems correspond to constraints and queries.

What is a database?

- A database consists of a collection of interconnected tables.
- The *schema* of a database lays out its connectivity structure (wiring).
  - A set of tables, connected together by their columns.
  - Each table has an *ID column*, giving a unique identifier to each record.
  - A column might indicate pure data, like "First name".
  - A column might be a *foreign key* that links its table to another table.
- An *instance* of the database is a collection of data conforming to the schema.
  - The rows of a table are called *records*.
  - A cell in a foreign key column points out to a record in another table.

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

- In each table, note three types of columns: ID, data, foreign key.
- Perhaps we should enforce certain 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

- Theoretically we can consider a data-type as a 1-column table.
- Examples:

String
ID
a
b
c
⋮
z
aa
⋮

First name
ID
Alan
Alice
Bertrand
⋮
David
Eugene
⋮

- So even data columns can be considered as foreign keys (to respective 1-column tables).
- Conclusion: each non-ID column in a table is a foreign key.

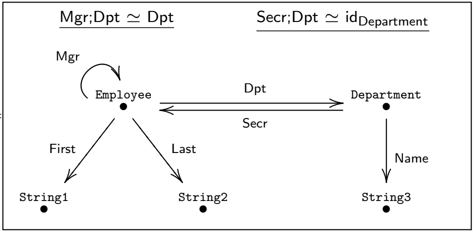
Example again

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

String1
ID
Alan
Alice
Bertrand
⋮
David
Eugene
⋮

$\mathcal{C} :=$



$\mathbf{Sch} \cong \mathbf{Cat}$

Definition

- A *schema* is a small category presentation; i.e. a graph with an appropriate equivalence relation on the set of paths.
- A *morphism of schemas* $F\colon \mathcal{C} \to \mathcal{D}$ sends objects to objects and arrows to paths, respecting source, target, and the path equivalence relation.
- We denote the *category of schemas* by **Sch**.

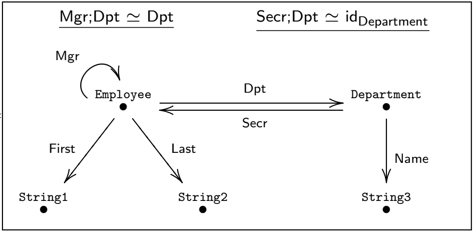
Proposition

There is an equivalence of categories,

$$\mathbf{Sch} \cong \mathbf{Cat}.$$

Schema=Category, Instance=Set-valued functor

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

$\mathcal{C} :=$ 

- A functor $I : \mathcal{C} \to \mathbf{Set}$ 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.

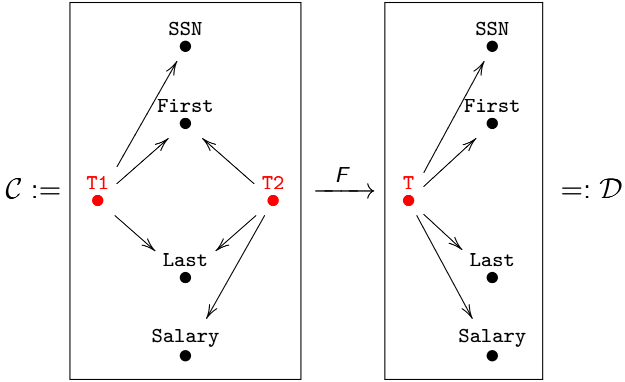
Functorial data migration

- 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 $F^*\colon \mathcal{D}\text{–}\mathbf{Set} \to \mathcal{C}\text{–}\mathbf{Set}$,

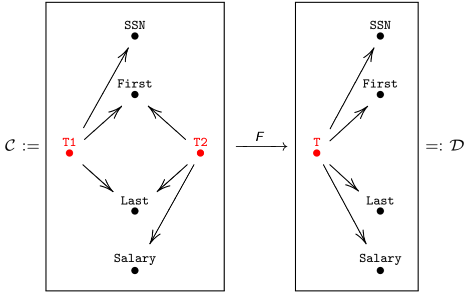
$$\mathcal{C} \xrightarrow{F} \mathcal{D} \xrightarrow{I} \mathbf{Set}.$$

- The functor F^* migrates data from $\mathcal{D}$ back to $\mathcal{C}$.
- It has two adjoints $F_!\colon \mathcal{C}\text{–}\mathbf{Set} \to \mathcal{D}\text{–}\mathbf{Set}$ and $F_*\colon \mathcal{C}\text{–}\mathbf{Set} \to \mathcal{D}\text{–}\mathbf{Set}$.

Uses of functorial data migration 1: Translation F



Uses of functorial data migration 2: Projection via F^*



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

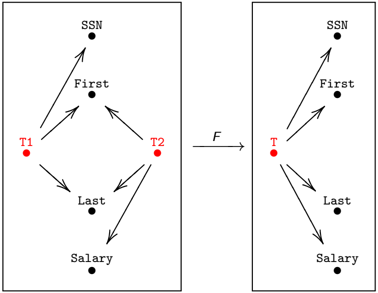
T					
ID	SSN	First	Last	Salary
XF667	115-234	Bob	Smith	$250
XF891	122-988	Sue	Smith	$300
XF221	198-877	Alice	Jones	$100

$F^*(J)\colon \mathcal{C} \to \textbf{Set}$:

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 3: Joins via F_*

$\mathcal{C} :=$  $=: \mathcal{D}$

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

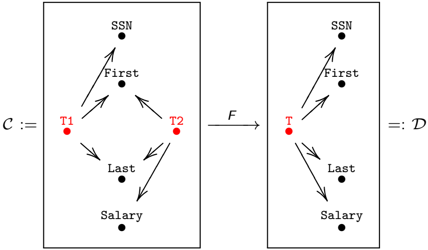
T1			
ID	**SSN**	**First**	**Last**
T1-001	115-234	Bob	Smith
T1-002	122-988	Sue	Smith
T1-003	198-877	Alice	Jones

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

$F_*(I): \mathcal{D} \to \mathbf{Set}$:

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 4: Unions via $F_!$



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

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

$F_!(I): \mathcal{D} \to \mathbf{Set}$:

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

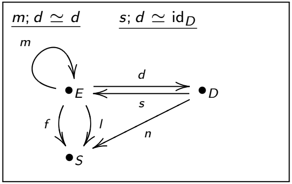
Category of elements of a database instance

- 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

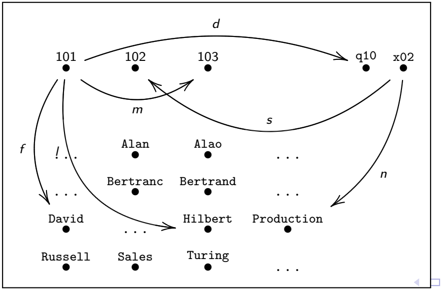
Department		
ID	**Name**	**Secr'y**
q10	Sales	101
x02	Production	102

$\mathcal{C} =$



Here is Elt(I), the category of elements of I:

Elt(I) =

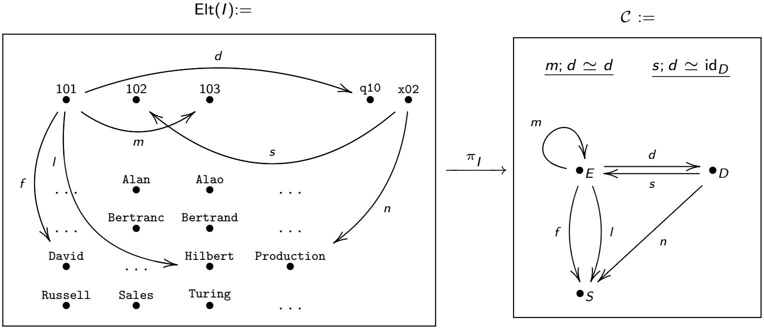


10 arrows left out.

A different perspective on data, "RDF"

In fact, the category of elements of an instance $I\colon \mathcal{C} \to \mathbf{Set}$ always yields not only a category $\mathrm{Elt}(I)$ but a functor

$$\pi_I\colon \mathrm{Elt}(I) \to \mathcal{C}.$$



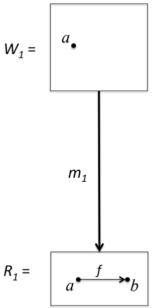
For each $X \in \mathcal{C}$, its inverse image is $\pi_I^{-1}(X) = I(X)$, the set of rows in table X.

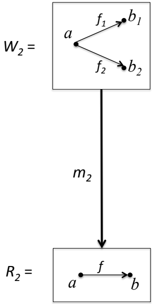
Relational fibration

$\mathrm{Elt}(I) \xrightarrow{\pi_I} \mathcal{C}$ will satisfy two *global lifting criteria*:

$$\begin{array}{ccc} W_1 & \xrightarrow{\forall} & \mathrm{Elt}(I) \\ {\scriptstyle m_1}\downarrow & \overset{\exists}{\nearrow} & \downarrow{\scriptstyle \pi_I} \\ R_1 & \xrightarrow[\forall]{} & \mathcal{C} \end{array} \qquad\qquad \begin{array}{ccc} W_2 & \xrightarrow{\forall} & \mathrm{Elt}(I) \\ {\scriptstyle m_2}\downarrow & \overset{\exists}{\nearrow} & \downarrow{\scriptstyle \pi_I} \\ R_2 & \xrightarrow[\forall]{} & \mathcal{C} \end{array}$$

where m_1 and m_2 are the following functors:





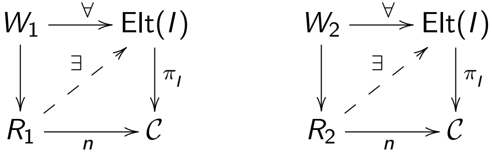
Global vs. local lifting criteria

Suppose we have a relational fibration $\text{Elt}(I) \xrightarrow{\pi_I} \mathcal{C}$.

- What if we want one table to be the product of two others?
- We have a diagram in $\mathcal{C}$,

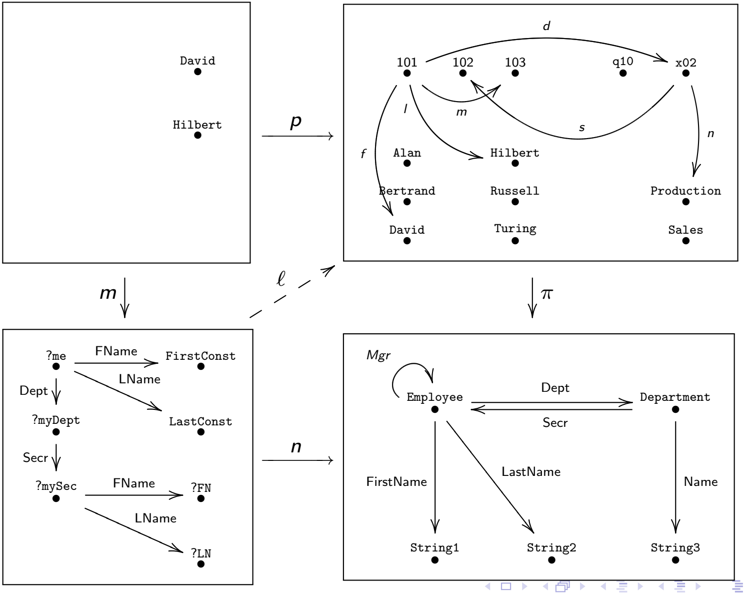
$$\boxed{\overset{S_1}{\bullet} \longleftarrow \overset{T}{\bullet} \longrightarrow \overset{S_2}{\bullet}} \xrightarrow{n} \mathcal{C}.$$

- The desired result can be achieved using two *local lifting criteria*



See boardwork.

Query as lifting problem



Homotopy of data? Database of homotopy?

- Can we learn things about data $F\colon \mathcal{C} \to \mathbf{Set}$ by studying it topologically?
    - We could take its homotopy colimit.
    - This can be computed as the nerve,

$$\text{hocolim}(F) \simeq \mathit{Nerve}(\text{Elt}(F)).$$

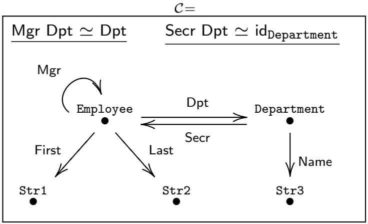
- When data is topological, this has the desired meaning.
    - If $F\colon \Delta^{\text{op}} \to \mathbf{Set}$ is a simplicial set, then $\text{Re}(F) \simeq \mathit{Nerve}(\text{Elt}(F))$.
- Given a compact Lie group G acting on a space X, taking connected components of fixed-point spaces yields a database whose tables are the subgroups of G.
    - This is the topic of Morava's recent paper.
    - The idea is that scientific databases may have a more mathematical (catastrophe-theoretic) foundation.

Conclusion

- I hope the connection between databases and categories is clear.

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



- This opens new opportunities for collaboration between math and CS.
  - Can topology be used to make sense of data?
  - Can databases be used to investigate mathematics?

Thanks for the invitation to speak!