Relational Foundations for Functorial Data Migration

David Spivak, **Ryan Wisnesky**

Department of Mathematics
Massachusetts Institute of Technology

{dspivak, **wisnesky**}@math.mit.edu

DBPL
October 27, 2015

Introduction

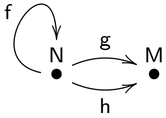
- In this talk I will describe an equivalence between a fragment of the relational data model (SPCU queries) and a fragment of the (extended) functorial data model (FQL queries):

$$SPCU \cong FQL$$

- The functorial data model (my name) originated with Rosebrugh et al. in the late 1990s.
  - Schemas are categories, instances are set-valued functors.
  - Spivak extends it to solve information integration problems.
- Sponsored by:
  - ONR grant N000141310260
  - AFOSR grant FA9550-14-1-0031

Category Theory

- A **presentation** of a **category** is a *reflexive*, *directed*, *labelled*, *multi-graph* and a set of *path equations*:



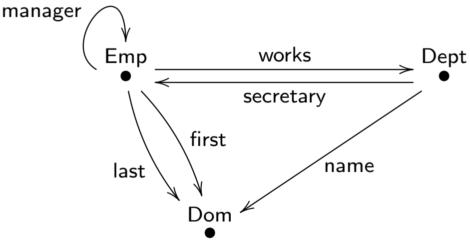
$$\mathsf{N.f.f.g} = \mathsf{N.f.h}$$

- A **set-valued functor** assigns a set to each node and a function to each edge, such that the equations holds.

$$N = \mathbf{N} \quad M = \{\text{bill}\} \quad f(x) = x + 1 \quad g(x) = h(x) = \text{bill} \quad \forall x \in \mathbf{N}$$

- Category theory was instrumental in the development of two extensions to the relational model, both of which inform work on language-integrated query (LINQ):
  - **The nested relational model** generalizes sets to nested collections and is inspired by *monads*.
  - **Algebraic datatypes** implement nested collections using recursion and are inspired by *algebras*.

The Functorial Data Model



$$\text{Emp.manager.works} = \text{Emp.works}$$

$$\text{Dept.secretary.works} = \text{Dept}$$

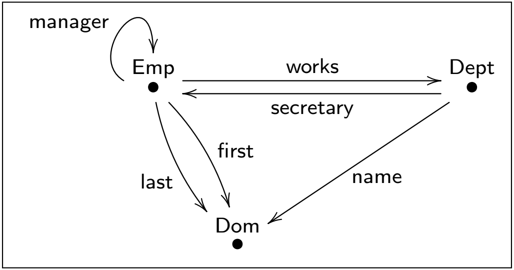
Emp				
ID	**mgr**	**works**	**first**	**last**
101	103	q10	Al	Akin
102	102	x02	Bob	Bo
103	103	q10	Carl	Cork

Dept		
ID	**sec**	**name**
q10	102	CS
x02	101	Math

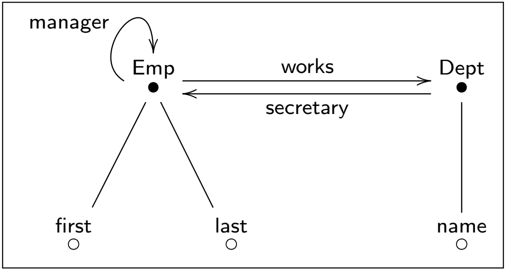
Dom
ID
Al
Akin
Bob
Bo
Carl
Cork
CS
Math

Convention

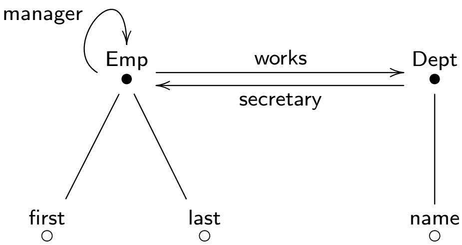
- Omit Dom table, and draw edges $\bullet \to_f \bullet_{\mathrm{Dom}}$ as $\bullet - \circ_f$:



=



The Functorial Data Model (abbreviated)



Emp.manager.works = Emp.works   Dept.secretary.works = Dept

Emp				
ID	**mgr**	**works**	**first**	**last**
101	103	q10	Al	Akin
102	102	x02	Bob	Bo
103	103	q10	Carl	Cork

Dept		
ID	**sec**	**name**
q10	102	CS
x02	101	Math

Functorial Data Migration

- A functor $F \colon S \to T$ is a constraint-respecting mapping:

$$nodes(S) \to nodes(T) \qquad edges(S) \to paths(T)$$

  and it induces three adjoint data migration functors:

  - $\Delta_F \colon T$-inst $\to S$-inst (like project)

$$\begin{tikzcd} S \arrow[r, "F"] \arrow[rr, bend right, "\Delta_F(I) \ := \ I \circ F"'] & T \arrow[r, "I"] & \mathbf{Set} \end{tikzcd}$$

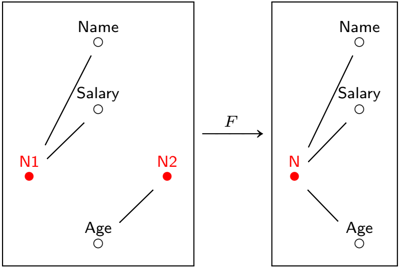
  - $\Pi_F \colon S$-inst $\to T$-inst (like join)

$$\Delta_F \dashv \Pi_F$$

  - $\Sigma_F \colon S$-inst $\to T$-inst (like outer disjoint union then quotient)

$$\Sigma_F \dashv \Delta_F$$

Δ (Project)



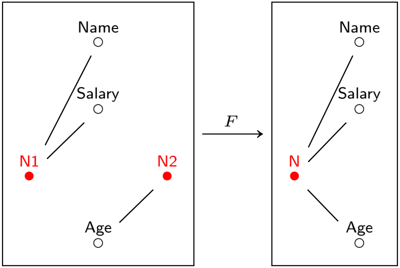
N1		
ID	**Name**	**Salary**
1	Alice	$100
2	Bob	$250
3	Sue	$300

N2	
ID	**Age**
4	20
5	20
6	30

$\xleftarrow{\Delta_F}$

N			
ID	**Name**	**Salary**	**Age**
a	Alice	$100	20
b	Bob	$250	20
c	Sue	$300	30

Π (Join)



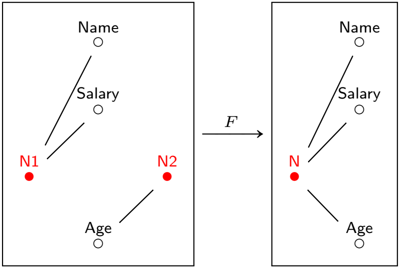
N1		
ID	**Name**	**Salary**
1	Alice	$100
2	Bob	$250
3	Sue	$300

N2	
ID	**Age**
4	20
5	20
6	30

$\xrightarrow{\Pi_F}$

N			
ID	**Name**	**Salary**	**Age**
a	Alice	$100	20
b	Alice	$100	20
c	Alice	$100	30
d	Bob	$250	20
e	Bob	$250	20
f	Bob	$250	30
g	Sue	$300	20
h	Sue	$300	20
i	Sue	$300	30

Σ (Union)



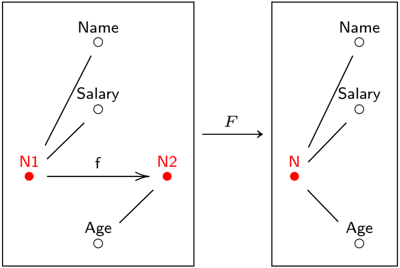
N1		
ID	**Name**	**Salary**
1	Alice	$100
2	Bob	$250
3	Sue	$300

N2	
ID	**Age**
4	20
5	20
6	30

$\xrightarrow{\Sigma_F}$

N			
ID	**Name**	**Salary**	**Age**
a	Alice	$100	$null_1$
b	Bob	$250	$null_2$
c	Sue	$300	$null_3$
d	$null_4$	$null_5$	20
e	$null_6$	$null_7$	20
f	$null_8$	$null_9$	30

Foreign keys



N1				N2	
ID	**Name**	**Salary**	**f**	**ID**	**Age**
1	Alice	$100	4	4	20
2	Bob	$250	5	5	20
3	Sue	$300	6	6	30

$$\xleftarrow{\Delta_F} \quad \xrightarrow{\Pi_F, \Sigma_F}$$

N			
ID	**Name**	**Salary**	**Age**
a	Alice	$100	20
b	Bob	$250	20
c	Sue	$300	30

Evaluation of the functorial data model

- Positives:
  - The category of categories is bi-cartesian closed (model of the STLC).
  - For each category C, the category C-inst is a topos (model of HOL).
  - Data integrity constraints (path equations) are built-in to schemas.
  - Data migration functors transform entire instances.
  - The FDM is expressive enough for many information integration tasks.
  - Easy to pivot.
- Negatives:
  - Data integrity constraints (in schemas) are limited to path equalities.
  - Data migrations lack analog of set-difference.
  - No aggregation.
  - Data migration functors are hard to program directly.
  - Instance isomorphism is too coarse for many integration tasks.
  - Many problems about finitely-presented categories are semi-computable:
    - Path equivalence
    - Generating a category from a presentation

The Attribute Problem

N			
ID	**Name**	**Age**	**Salary**
1	Alice	20	$100
2	Bob	20	$250
3	Sue	30	$300

$\cong$ (good)

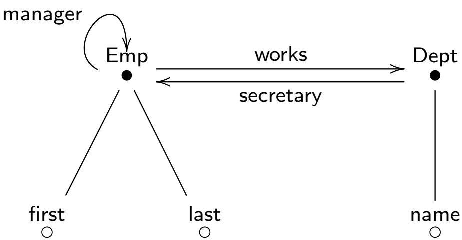
N			
ID	**Name**	**Age**	**Salary**
4	Alice	20	$100
5	Bob	20	$250
6	Sue	30	$300

$\cong$ (bad)

N			
ID	**Name**	**Age**	**Salary**
1	Amy	20	$100
2	Bill	20	$250
3	Susan	30	$300

Solving the Attribute Problem

- Mark certain edges to leaf nodes as “attributes”.
  - In this extension, a schema is a category C, a discrete category C_0, and a functor $C_0 \to C$. Instances and migrations also generalize.
  - Schemas become special ER (entity-relationship) diagrams.
  - The FDM takes C_0 to be empty.
  - The example schema below, which was an abbreviation in the FDM, is a bona-fide schema in this extension: attributes are first, last, and name.



Solved Attribute Problem

N			
ID	**Name**	**Age**	**Salary**
1	Alice	20	$100
2	Bob	20	$250
3	Sue	30	$300

$\cong$ (good)

N			
ID	**Name**	**Age**	**Salary**
4	Alice	20	$100
5	Bob	20	$250
6	Sue	30	$300

$\ncong$ (good)

N			
ID	**Name**	**Age**	**Salary**
1	Amy	20	$100
2	Bill	20	$250
3	Susan	30	$300

Functorial Data Migration as SPCU

- Theorem: migrations of the form

$$\Sigma_F \circ \Pi_G \circ \Delta_H$$

  - F is a discrete op-fibration (ensures union compatibility).
  - G is a surjection on attributes (ensures domain independence).
  - all categories are finite (ensures computability).
- can be implemented using SPCU (select, project, cartesian product, union) and keygen, under set semantics.
- are closed under composition.

Δ using SPCU

Given $F : S \to T$ and $I \in T\text{–}\mathbf{Inst}$, define $\Delta_F(I) \in S\text{–}\mathbf{Inst}$ as:

- for each node N in S, the table $\Delta_F(N)$ is $I(F(N))$.
- for each attribute A in S, the table $\Delta_F(A)$ is $I(F(A))$.
- for each edge $e : X \to Y$ in S mapping to a path $F(e) : F(X) \to F(Y)$ in T, compose $I(F(e))$ to obtain $\Delta_F(e)$.

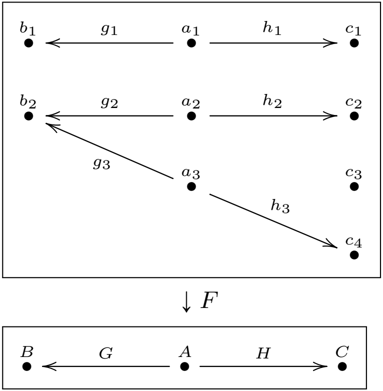
$$\begin{tikzcd} S \arrow[r, "F"] \arrow[rr, bend right, "\Delta_F(I) \ := \ I \circ F"'] & T \arrow[r, "I"] & \mathbf{Set} \end{tikzcd}$$

Σ using SPCU

Gven $F : S \to T$ a discrete op-fibration, a S-instance I, we define $\Sigma_F(I) \in T$–$\mathbf{Inst}$ as

- for each node N in T, the table $\Sigma_F(N)$ is the union of the node tables in I that F maps to N.
- for each attribute A in T, the table $\Sigma_F(A)$ is the union of the attribute tables in I that F maps to A.
- Let $e : X \to Y$ be an edge in T. We know that for each $c \in F^{-1}(X)$ there is at least one path p_c in S such that $F(p_c) \cong e$. Compose p_c to a single binary table, and define $\Sigma_F(e)$ to be the union over all such c. The choice of p_c will not matter.

Discrete Op-Fibrations / Union Compatibility



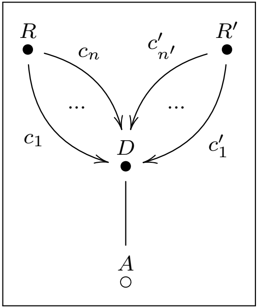
Π using SPCU

Given $F : S \to T$ with S finite and a S-instance I, we define $\Pi_F(I) \in T\text{–}\mathbf{Inst}$ as:

- too difficult to describe in a presentation.
- Intuitively, Π is a "join all"

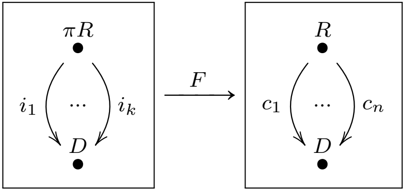
SPCU as Functorial Data Migration

- Theorem : SPCU (bags) can be implemented using Δ, Σ, Π.
- Theorem : SPCU (sets) can be implemented using $\Delta, \Sigma, \Pi, dedup$, where $dedup_T : T\text{–}\mathbf{Inst} \to T\text{–}\mathbf{Inst}$ equates IDs which cannot be distinguished along any attribute path.
- We must encode relational schemas, for example, $R(c_1, \ldots, c_n)$ and $R'(c'_1, \ldots, c'_{n'})$ becomes:



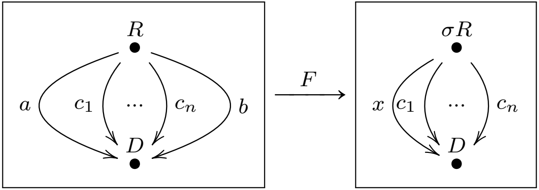
Project using Δ

We express $\pi_{i_1,\ldots,i_k} R$ as Δ_F:



Select using Δ, Π

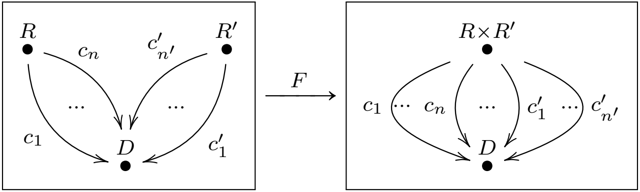
We express $\sigma_{a=b}R$ as $\Delta_F \circ \Pi_F$:



Here $F(a) = F(b) = x$ and $F(c_i) = c_i$ for $1 \leqslant i \leqslant n$.

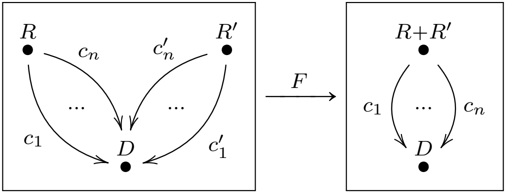
Product using Π

We express $R \times R'$ as Π_F:



Union using Σ

We express $R + R'$ as Σ_F:



FQL - A Functorial Query Language

- The open-source, graphical FQL IDE available at categoricaldata.net/fql.html implements functorial data migration (with attributes) in software. FQL translates migrations of the form

  $$\Sigma_F \circ \Pi_G \circ \Delta_H$$

  into SQL and vice versa.

- Demo

FQL evaluation

- Positives:
  - Attributes.
  - Running on SQL enables interoperability and execution speed.
  - Better Σ semantics than TGD-only systems (e.g., Clio).
- Negatives:
  - No selection by constants.
  - Relies on fresh ID generation.
  - Cannot change type of data during migration.
  - Attributes not nullable.
- See our follow-up work for solutions to these problems.

Conclusion

- I described the functorial data model and data migration functors,
- how to extend the functorial data model to have attributes,
- an equivalence

$$SPCU \cong FQL$$

  where FQL is a fragment of the data migration functors
- a tool, FQL (categoricaldata.net/fql.html) based on this equivalence.