A Functorial Query Language

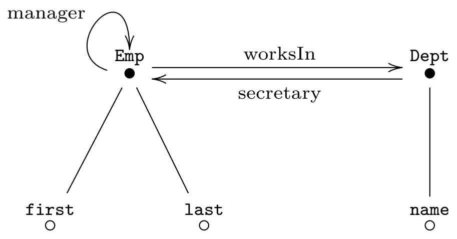
David I. Spivak
Massachusetts Institute of Technology
dspivak@math.mit.edu

Ryan Wisnesky
Harvard University
ryan@cs.harvard.edu

January 2, 2014

In this talk we present FQL, a **F**unctorial **Q**uery **L**anguage. FQL is the natural query language of the functorial data model (FDM) [8], which is a proposed alternative to the relational data model based on category-theoretic foundations. The mathematics of the FDM are developed in [8], but few specific connections are made to computer science. In this talk we describe recent work [9] to connect the FDM to the relational model, culminating in a definition of the FQL language, a compiler for FQL that targets SQL, and a visual IDE. The project webpage at wisnesky.net/fql.html contains the IDE and other papers and slides about the project.

In the FDM, a schema is a category and a schema mapping is a functor; schemas and mappings form a category, **Cat**. A database instance on a schema $\mathcal{C}$ is a functor $\mathcal{C}$ to **Set**, the category of sets, and a database homomorphism between two $\mathcal{C}$-instances I and J is a natural transformations $I \Rightarrow J$. The instances and homomorphisms on a schema $\mathcal{C}$ constitute a category, denoted $\mathcal{C}$–**Inst**. A mapping M between schemas S and T, which can be generated from a visual correspondence between graphs, induces three adjoint data migration functors, $\Sigma_M \colon S\text{–}\mathbf{Inst} \to T\text{–}\mathbf{Inst}$, $\Pi_M \colon S\text{–}\mathbf{Inst} \to T\text{–}\mathbf{Inst}$, and $\Delta_M \colon T\text{–}\mathbf{Inst} \to S\text{–}\mathbf{Inst}$. In recent work [9], we added "attributes" to FDM schemas and instances and proved that migrations of the form $\Sigma_F \circ \Pi_G \circ \Delta_H$, where F is a discrete op-fibration, are closed under composition. Moreover, we proved that such migrations can be implemented using the SPCU (select, project, product, union) relational algebra extended with a unique ID generating operation, and proved that SPCU can be implemented as such migrations. The following is an example FQL schema:



Emp.manager.worksIn = Emp.worksIn

Dept.secretary.worksIn = Dept

The two black nodes, three directed edges, and two equational constraints determine a category: the category freely generated by the graph, modulo the equations. An instance on this schema consists of, for each black node ("entity set"), a set of meaningless IDs; for each directed edge, a function from IDs to IDs, and for each white node ("attribute"), a function from IDs to the domain (e.g., strings). An instance can be represented as tables:

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

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

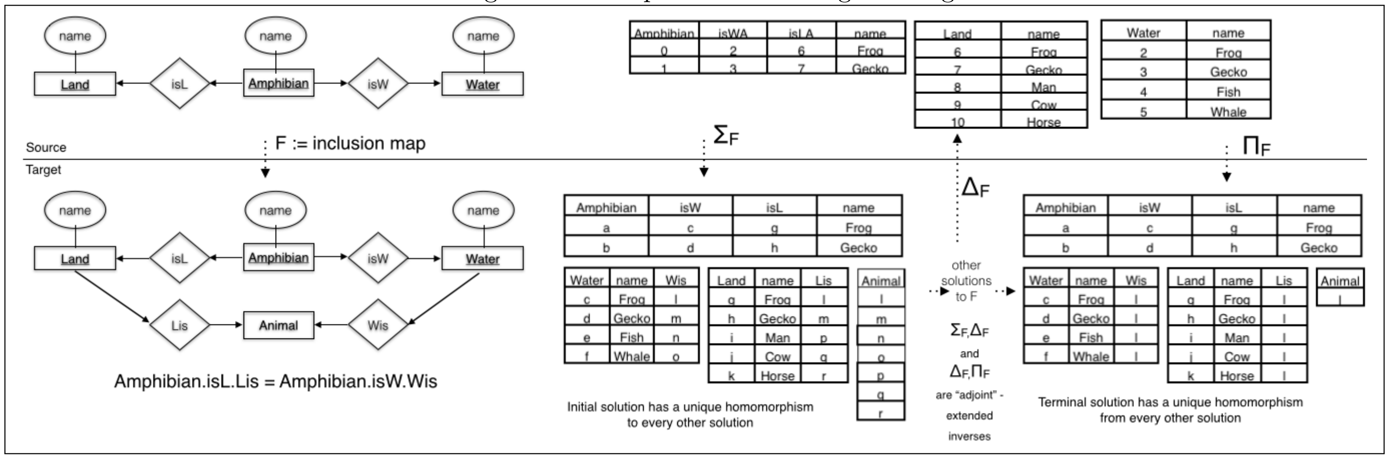
A schema mapping $F : \mathcal{C} \to \mathcal{D}$ is a functor from $\mathcal{C}$ to $\mathcal{D}$: a constraint-respecting function $nodes(\mathcal{C}) \to nodes(\mathcal{D})$ and $edges(\mathcal{C}) \to paths(\mathcal{D})$. We define $\Delta_F \colon \mathcal{D}\text{–}\mathbf{Inst} \to \mathcal{C}\text{–}\mathbf{Inst}$ as composition:

$$\Delta_F(I) := I \circ F \colon \mathcal{C} \to \mathbf{Set} \qquad \begin{tikzcd} \mathcal{C} \arrow[r, "F"] \arrow[rr, bend right, "\Delta_F I"'] & \mathcal{D} \arrow[r, "I"] & \mathbf{Set} \end{tikzcd}$$

Then, $\Sigma_F \colon \mathcal{C}\text{–}\mathbf{Inst} \to \mathcal{D}\text{–}\mathbf{Inst}$ is defined as the left adjoint to Δ_F, and $\Pi_F \colon \mathcal{C}\text{–}\mathbf{Inst} \to \mathcal{D}\text{–}\mathbf{Inst}$ is defined as the right adjoint to Δ_F. Intuitively, Δ_F can be thought of as projecting along F, Σ_F as taking unions along F, and Π_F as taking joins along F. The semantics of Σ is similar to that of the Clio data migration system [5], and Δ is the crucial reverse data migration operation identified by Bernstein and Alagic in their categorical model theory [1]. An example FQL data exchange setting using entity-relationship (ER) notation is shown in Figure 1.

Schemas, mappings, instances, and homomorphisms are the constants of FQL. To combine them, FQL uses the internal language of **Cat** as a bi-cartesian closed category (BCCC) as its programming language for schemas and schema mappings (see Figure 2). This language can equivalently be thought of as the simply-typed λ-calculus with strong $0, 1, \times, +, \to$ types and finitely presented categories and functors as constants. For each schema T, the category T–**Inst** is a topos – a BCCC with a sub object classifier [2]. Hence FQL T-instances and T-homomorphisms are programmed using the internal language of a topos, which can equivalently be thought of as intuitionistic higher-order logic, extended with data migration operations Δ, Σ, Π.

Figure 1: Example Data Exchange Setting



Amphibian	isWA	isLA	name
0	2	6	Frog
1	3	7	Gecko

Land	name
6	Frog
7	Gecko
8	Man
9	Cow
10	Horse

Water	name
2	Frog
3	Gecko
4	Fish
5	Whale

Amphibian	isW	isL	name
a	c	g	Frog
b	d	h	Gecko

Water	name	Wis
c	Frog	l
d	Gecko	m
e	Fish	n
f	Whale	o

Land	name	Lis
g	Frog	l
h	Gecko	m
i	Man	p
j	Cow	q
k	Horse	r

Animal
l
m
n
o
p
q
r

Initial solution has a unique homomorphism to every other solution

Amphibian	isW	isL	name
a	c	g	Frog
b	d	h	Gecko

Water	name	Wis
c	Frog	l
d	Gecko	l
e	Fish	l
f	Whale	l

Land	name	Lis
g	Frog	l
h	Gecko	l
i	Man	l
j	Cow	l
k	Horse	l

Animal
l

Terminal solution has a unique homomorphism from every other solution

Figure 2: Syntax of FQL Schemas T and Schema Mappings F

$$T \quad ::= \quad 0 \mid 1 \mid T + T \mid T \times T \mid T^T \mid \mathcal{T} (= \text{finitely presented categories})$$

$$\begin{aligned} F \quad ::= \quad & id_T \mid F \circ F \mid proj^1_{T,T} \mid proj^2_{T,T} \mid inj^1_{T,T} \mid inj^2_{T,T} \mid tt_T \mid ff_T \mid \\ & F \otimes F \mid F \oplus F \mid ev_{T,T} \mid \Lambda F \mid \mathcal{F}_{T,T} (= \text{finitely presented functors}) \end{aligned}$$

Related Work

Although category presentations are a common notation for schemas [3], most work treats such schemas as abbreviations for relational schemas. For example, in Clio [5], users draw lines connecting related elements between two schemas-as-graphs. The user's input correspondence is translated into the relational language of tuple generating dependencies, from which a query that implements the user's intended data transformation is generated. We believe that it is advantageous to treat such schemas directly as categories, and indeed Clio cannot perform the Σ data migration in Figure 1 because it cannot make sense of the path equality constraint.

In many ways, our work is an extension and improvement of Rosebrugh et al's initial work on understanding caterories as database schemas [4]. In that work, the authors identify the Σ and Π data migration functors, but they do not identify Δ as their adjoint. Moreover, they were unable to implement Σ and Π using relational algebra, and they do not formalize a query language or investigate the composition behavior of Σ and Π. Our development diverges from their subsequent work on "sketches" [7].

Category-theoretic techniques were instrumental in the development of nested relational data model [10]. However, the FDM and the nested relational model do not appear to be connected. The FDM is more closely related to work on categorical logic and type theory, where operations similar to Σ, Π, and Δ often appear under the slogan that "quantification is adjoint to substitution" [6].

References

[1] Suad Alagic and Philip A. Bernstein. A model theory for generic schema management. In *DBPL*, 2001.

[2] Michael Barr and Charles Wells, editors. *Category theory for computing science, 2nd ed.* 1995.

[3] P. Buneman, S. Davidson, and A. Kosky. Theoretical aspects of schema merging. In *EDBT*, 1992.

[4] Michael Fleming, Ryan Gunther, and Robert Rosebrugh. A database of categories. *J. SYMBOLIC COMPUT*, 35:127–135, 2002.

[5] Laura M. Haas, Mauricio A. Hernández, Howard Ho, Lucian Popa, and Mary Roth. Clio grows up: from research prototype to industrial tool. In *SIGMOD '05:*.

[6] Bart Jacobs. *Categorical logic and type theory.* PhD thesis, Mathematics, Amsterdam, Lausanne, New York, 1999.

[7] Michael Johnson and Robert Rosebrugh. Sketch data models, relational schema and data specifications. *Electronic Notes in Theoretical Computer Science*, 61(0):51 – 63, 2002.

[8] David I. Spivak. Functorial data migration. *Inf. Comput.*, 217:31–51, August 2012.

[9] David I Spivak and Ryan. Wisnesky. Relational foundations for functorial data migration. http://arxiv.org/abs/1212.5303, 2013.

[10] Limsoon Wong. *Querying nested collections.* PhD thesis, Philadelphia, PA, USA, 1994. AAI9503855.