Functorial Data Migration

Categorical Informatics, Inc.

info@catinf.com

December 1, 2015

$$\mathcal{C}_i$$

Outline

- The Functorial Data Model (FDM)
  - The FDM is based on **category theory**, which was designed to migrate **theorems** from one **area of mathematics** to another.
- Functorial Data Migration
  - Now, researchers at MIT use category theory to migrate **data** from one **computer system** to another.
- The Functorial Query Language (FQL) tool
  - The FDM research has culminated in a prototype **ETL** software tool, FQL, available at catinf.com.
- Categorical Informatics ($\mathcal{C}_i$)
  - Because the FQL tool is based on a principled theoretical foundation, it gives **the best possible answer** to many ETL problems, and is being commercialized by $\mathcal{C}_i$.

Category theory

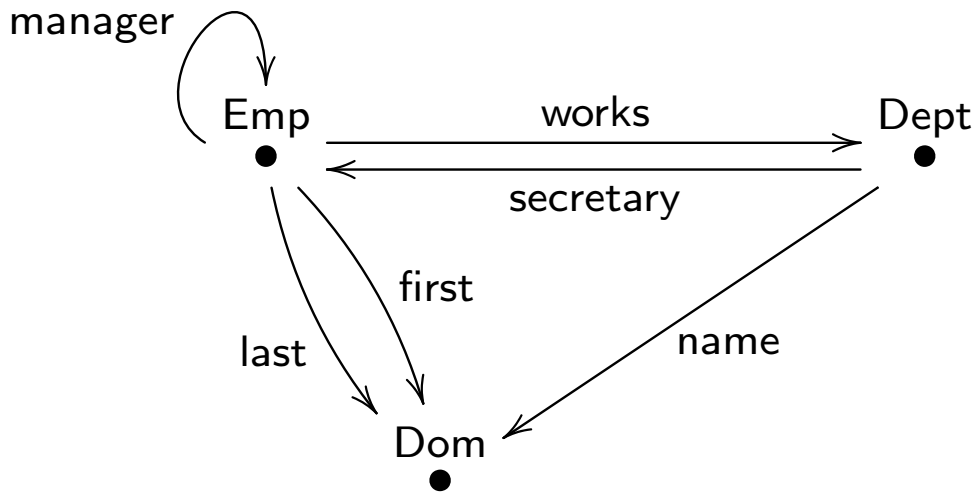
- Knowledge of category theory is **not required** to understand this talk, functorial data migration, or use the FQL tool.
  - But for completeness we include a brief description of category theory.
- A **category** is an algebraic structure similar to a group, ring, or field.
  - A category is a "multi-sorted monoid" and an "algebra of functions".
- A **functor** is a homomorphism of categories.
  - In the FDM, schemas, instances, schema mappings, and even data migration operations are expressed as categories and functors.
- Category theory has informed the design of:
  - Functional programming languages (Haskell, ML, Scala)
  - Query languages (nested relational calculus, LINQ)
  - Proof assistants (Coq, Agda, HoTT)
  - Mathematics itself (algebraic topology, set theory, model theory)

Category theory

- A category $\mathcal{C}$ consists of
  - a set of *objects*
  - for all objects X, Y a set $\mathcal{C}(X, Y)$ of *arrows*
  - for all objects X an arrow $id \in \mathcal{C}(X, X)$
  - for all objects X, Y, Z a function $\circ\colon \mathcal{C}(Y, Z) \times \mathcal{C}(X, Y) \to \mathcal{C}(X, Z)$
  - such that $f \circ id = id$ and $id \circ f = f$ and $(f \circ g) \circ h = f \circ (g \circ h)$
- A functor $F : \mathcal{C} \to \mathcal{D}$ is a function taking objects in $\mathcal{C}$ to objects in $\mathcal{D}$ and arrows $f\colon X \to Y$ in $\mathcal{C}$ to arrows $F(f)\colon F(X) \to F(Y)$ in $\mathcal{D}$ such that $F(id) = id$ and $F(f \circ g) = F(f) \circ F(g)$.

- A category presentation $\mathcal{C}$ consists of
  - a set of *nodes*
  - for all nodes X, Y a set $\mathcal{C}(X, Y)$ of *edges*
  - a set of path equations
- A functor presentation $F\colon \mathcal{C} \to \mathcal{D}$ is a function taking nodes in $\mathcal{C}$ to nodes in $\mathcal{D}$ and edges $f\colon X \to Y$ in $\mathcal{C}$ to paths $F(f)\colon F(X) \to F(Y)$ in $\mathcal{D}$ such that $\mathcal{C} \vdash p = q$ implies $\mathcal{D} \vdash F(p) = F(q)$.

The Functorial Data Model



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

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

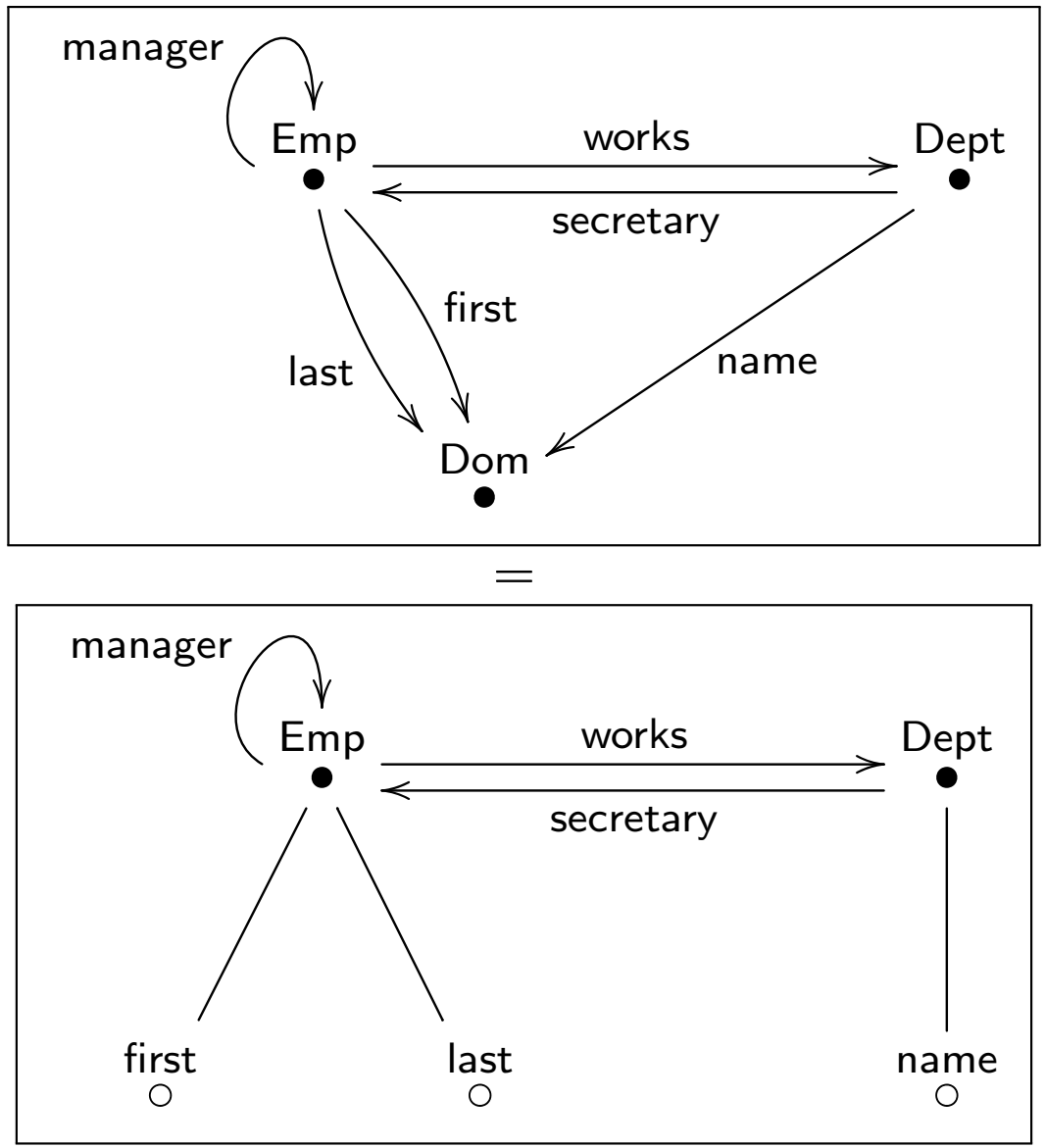
Emp				
ID	**mgr**	**works**	**first**	**last**
101	103	q10	Al	Akin
102	102	x02	Bob	Bo
103	103	q10	Carl	Cork

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

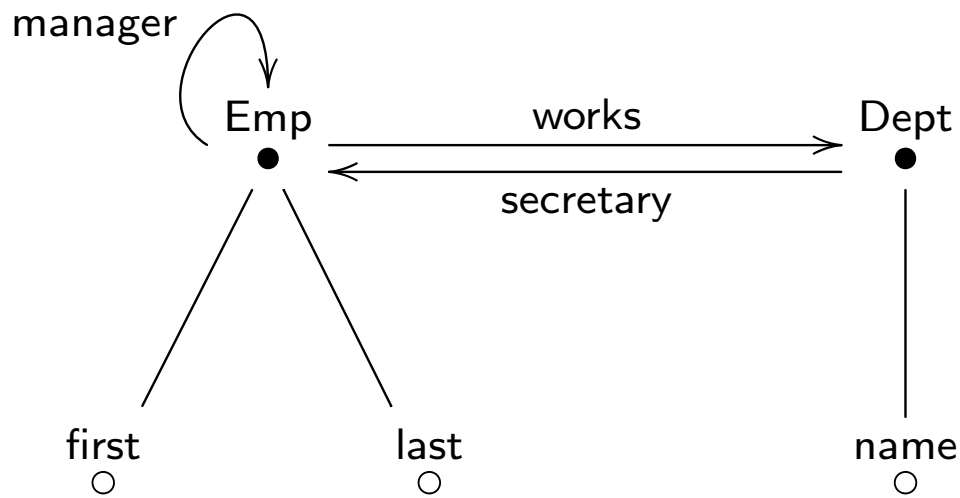
Dom
ID
Al
Akin
Bob
Bo
Carl
Cork
CS
Math

Attributes

- Omit Dom table, and draw edges $\bullet \rightarrow_{\mathsf{f}} \bullet_{\mathsf{Dom}}$ as $\bullet - \circ_{\mathsf{f}}$:



The Functorial Data Model



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

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

and it induces three adjoint data migration functors:

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

$$S \xrightarrow{F} T \xrightarrow{I} \mathbf{Set}$$
$$\Delta_F(I) := I \circ F$$

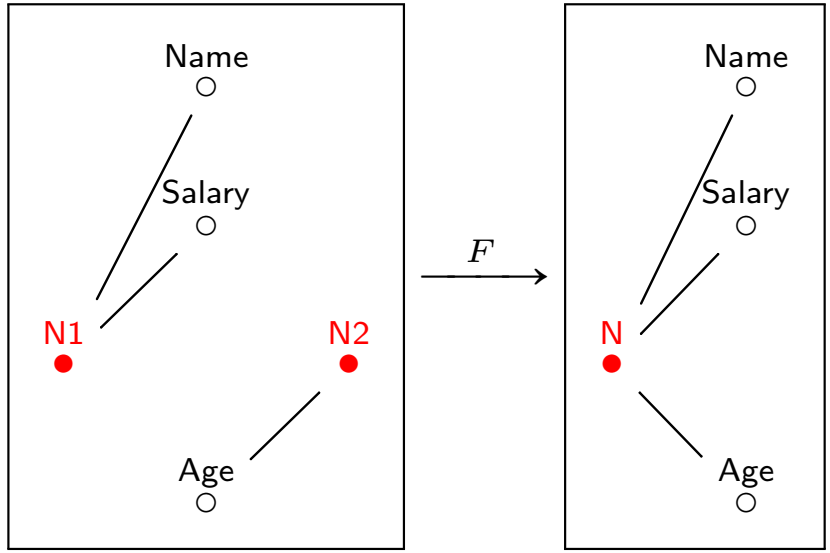
- $\Pi_F : S$-inst $\to T$-inst (like join)

$$\Delta_F \dashv \Pi_F$$

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

$$\Sigma_F \dashv \Delta_F$$

Δ (Project)



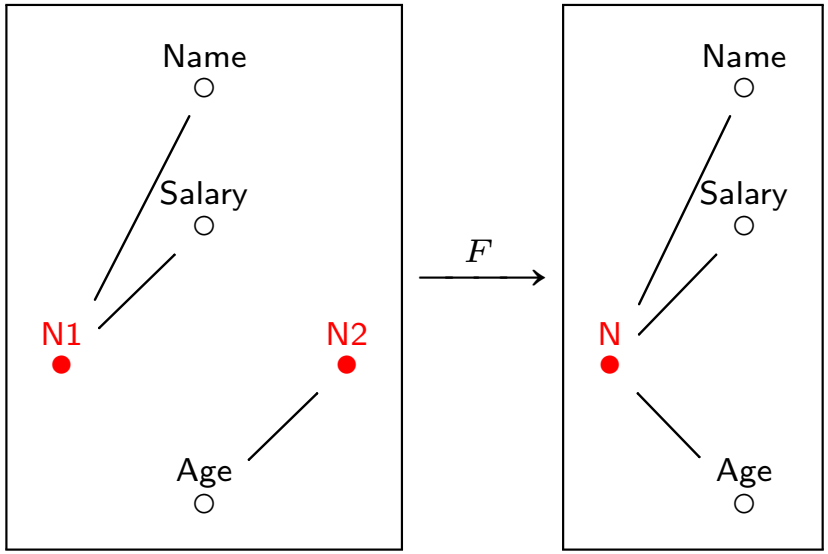
N1		
ID	**Name**	**Salary**
1	Alice	$100
2	Bob	$250
3	Sue	$300

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)



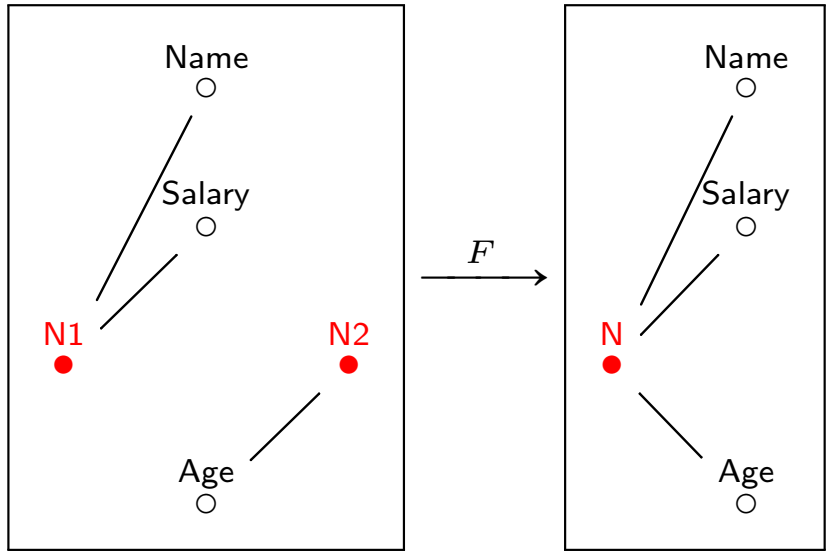
N1		
ID	**Name**	**Salary**
1	Alice	$100
2	Bob	$250
3	Sue	$300

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

Π_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)



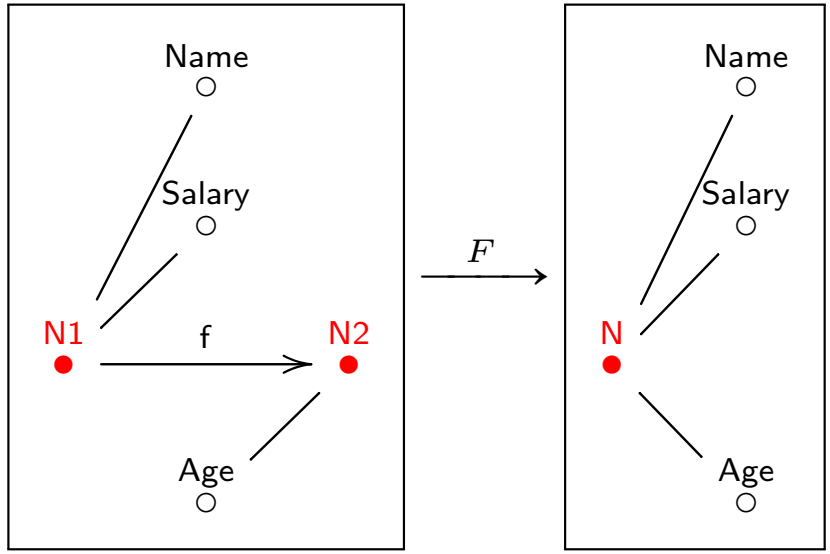
N1		
ID	**Name**	**Salary**
1	Alice	$100
2	Bob	$250
3	Sue	$300

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



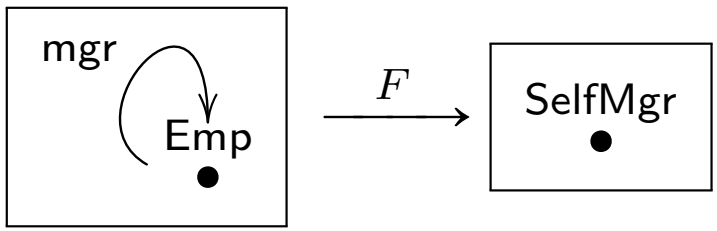
N1			
ID	**Name**	**Salary**	**f**
1	Alice	$100	4
2	Bob	$250	5
3	Sue	$300	6

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

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

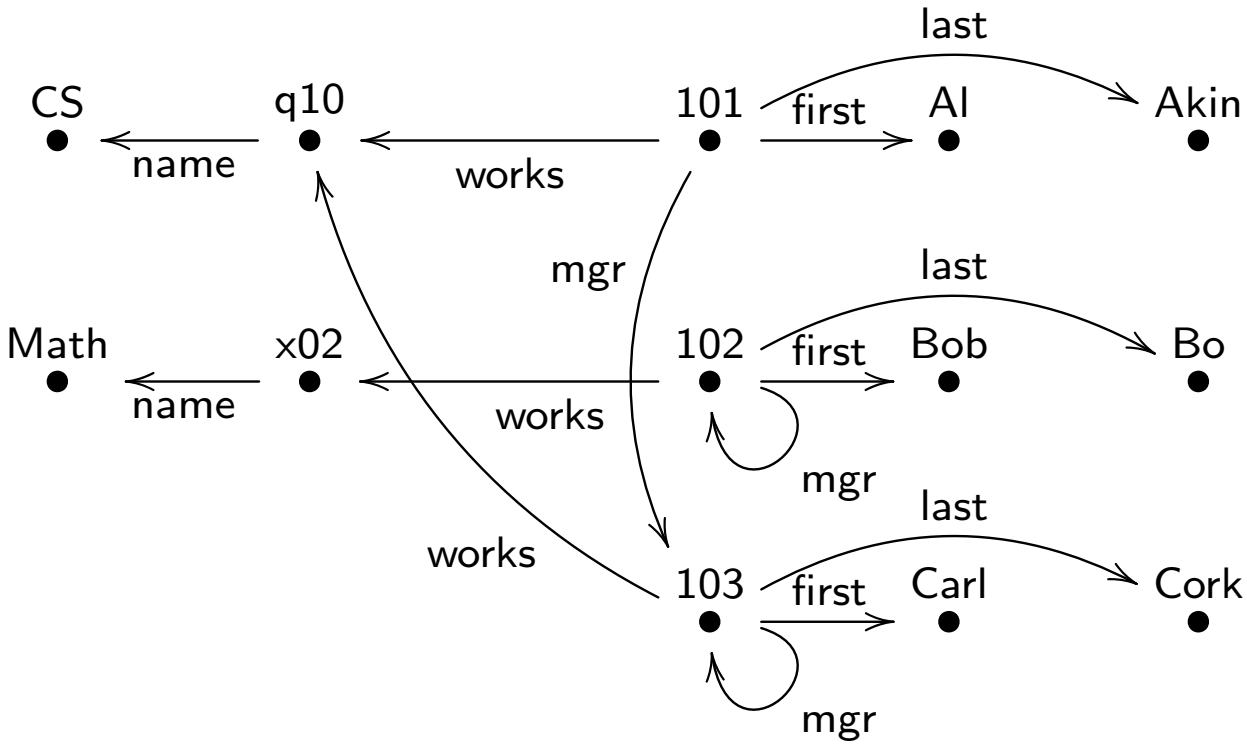
N			
ID	**Name**	**Salary**	**Age**
a	Alice	$100	20
b	Bob	$250	20
c	Sue	$300	30

Expressivity of Functorial Data Migration - Example



- Δ_F will copy SelfMgr into Emp, and put the identity into mgr.
- Π_F will migrate into SelfMgr those Emps who are their own mgr.
- Σ_F will migrate into SelfMgr representatives of the "management groups" of Emp, i.e. equivalence classes of Emps modulo the equivalence relation generated by mgr.

Pivot (Instance ⇔ Schema)



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

Dept	
ID	**name**
q10	CS
x02	Math

Positives of the functorial data model

- The category of categories is bi-cartesian closed (model of STLC).
  - Schemas support $0, 1, +, \times, \widehat{\ }$.
- For each category C, the category C-inst is a topos (model of HOL).
  - Instances support $0, 1, +, \times, \widehat{\ }$ and $\forall, \exists, \wedge, \vee, \neg, \to, \top, \bot$.
- Data integrity constraints (path equations) are built-in to schemas.
  - In progress: more expressive constraints ("EDs") in schemas.
- Data migrations transform entire instances.
- Easy to pivot.
- Σ has better semantics than TGD-only systems (e.g., Clio).

FQL - A Functorial Query Language

- FQL is an open-source, graphical IDE available at catinf.com. It translates data migrations of the form

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

  into SQL and executes via JDBC whenever possible. Otherwise, FQL executes the migration directly.

- FQL also includes a (partial) translator SQL $\rightarrow$ FQL, as well as RDF/JSON input/output.

- Some FQL queries can be written using built-in SELECT-FROM-WHERE syntax.

- Demo

Conclusion

- There are deep connections between the FDM and other data models, including relational, RDF, and XML.
- MIT and $\mathcal{C}_i$ have had initial success using FQL on a data integration scenario identified by the National Institute of Standards and Technology (NIST) and are looking for academic collaborators, customers, test cases, and employees.
- Visit catinf.com for more information.
- See categoricaldata.net and appliedcategorytheory.org for other interesting categorical projects.