Abstract Relations as Allegorical Categories

Nick Rossiter
Visiting Fellow
Computing Science and Digital Technologies
Northumbria University

ANPA 36 (August 2015)

Outline of Presentation

- Relationships versus Process
- Set theoretic approaches to relations
- Categorial translation from sets
- Allegories as enhanced categorial relations
- Findings
- Discussion on way forward

Relationships are Key

- Representing relationships is a key activity in the physical sciences
  - Entanglement in quantum theory
  - Interactions between particles
  - Connections between entities
- Intertwined with process
  - Relationships are often an abstract (partial) view of a process
- Capturing relationships in detail
  - Suitability for implementation in computer system increases credibility of approach

Relation versus Process 1

- Process:
  - Registration: a student registers for a module on a particular course
    - Physical activity
    - Written contract
    - Usually not stand-alone
      - Linked to other processes
      - Part of another process
      - Comprises other processes
  - Registration linked to other processes:
    - monitoring activity, determining outcome
  - Each process has rules – transaction

Relation versus Process 2

- Relation is often the information side of process
- Recording the facts
- Relatively static versus dynamism of process

In set theoretic terms 1

- Relation is the data structure
- sMt where
  - s, t are sets; M is relationship between sets
  - s is male partner, t is female partner, M is marriage
- But note that this is a surrogate for a process, the act of marriage

In set theoretic terms 2

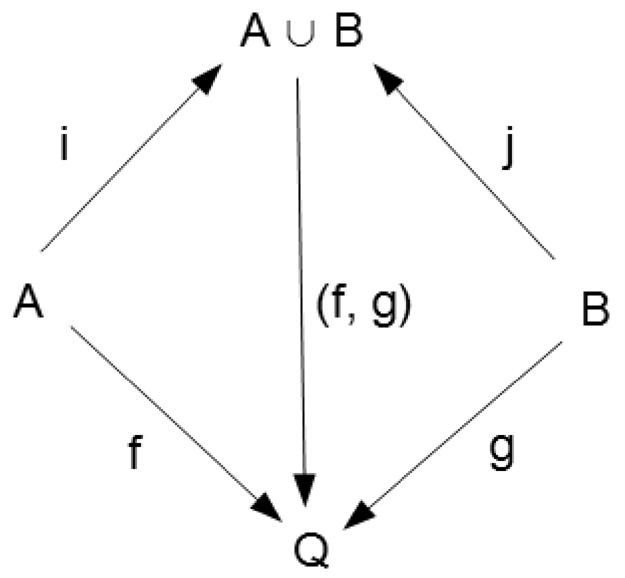
- General form for relation R is: sRt
  - s, t are sets; R is mapping between them
- R has various interpretations, either:
  - $R \subseteq S \times T$ (subset of product)
  - $R \in (S, T)$ (member of ordered pairs)
  - $R = \bigoplus(S \times T)$ (disjoint union of product members)
- Last is revealing, structure of R is disjoint union, not a product, although may be expressed as one

What does Category Theory say?

- Category **Rel** is:
  - Either a mapping (a functor) between sets:
    - Rel: **Set** → **Set**
      - where Rel ⊆ (**Set** X **Set**)
  - Or a disjoint union, that is a coproduct category
    - **Rel**: $\oplus$(A X B)
      - where **Rel** is a coproduct diagram over objects A, B

Rel as a Coproduct

- The category **Rel** with A ∪ B as relation over objects A, B



A ∪ B disjoint union
i,j inclusions
(f,g): A ∪ B → Q is unique
morphism such that diagram
commutes:
(f,g) o I = f
(f,g) o j = g
Q quotient with (f,g) as coequaliser

We like Cartesian Closed Category (CCC)

- Why?
- Vital properties
  - Cartesian for *products* (basis for relationships)
  - *Closed* at terminal object (closure at top)
  - Exponentiation for *connectivity* (eval property)
  - *Internal logic* from adjointness ($\exists \dashv \Delta \dashv \forall$)
  - *Identity* functor
  - Categories and objects *interchangeable*
- Implementable
  - CCC can be implemented on λ-calculus machines

Is **Rel** a CCC?

- Far from it!
- No terminal object if take Rel: **Set** → **Set**
  - initial and terminal objects are the same
- No product if take basis of **Rel** as coproduct
- So **Rel** is not, in our view, a viable construction for relationships or process
- **Rel** is categorification (translation) of the set theoretic concept

Way forward

- Set theoretic concept of relation is inadequate as basis for representing relations in category theory
- What about Allegories? (Freyd & Scedrov 1990)

  “Allegories are to binary relationships between sets as categories are to functions between sets.” (p.195, section 2.1)
- We next explore allegories and this claim

Allegories Defined (Freyd 2.1 p.195)

- An allegory is a category with unary operation R^0 and binary product operation $R \cap S$
  - R^0 reciprocation
    - $R : X \rightarrow Y$
    - xR^0y iff yRx
  - $R \cap S$ intersection
    - $R, S : X \rightarrow Y$
    - $xR \cap Sy$ iff xRy and xSy
    - Intersections are idempotent, commutative, associative
    - Composing intersections composes relations
    - Necessity for 2 relations in category theory to provide mapping; one relation could be the universal relation U

Operations on an Allegory

- Constant 1
  - x1y iff x=y
- Reciprocation unary R^0
  - xR^0y iff yRx
- Composition binary RS (relational join)
  - xRSy iff there exists z such that xRz and zSy
- Intersection binary $R \cap S$
  - $xR \cap Sy$ iff xRy and xSy

Underlying Regular Category

- Allegories are 'best' based on regular categories
- A regular category is Cartesian: a pullback with some 'nice' properties
  - stable factorization, with preservation of
    - epimorphisms (onto, all objects in colimit assigned)
    - coequalisers (pairs of parallel arrows converge onto one arrow as a sum)
- As a pullback, regular categories are CCC (Locally CCC in fact)

Logic of Pullback

From our ANPA 2014 paper

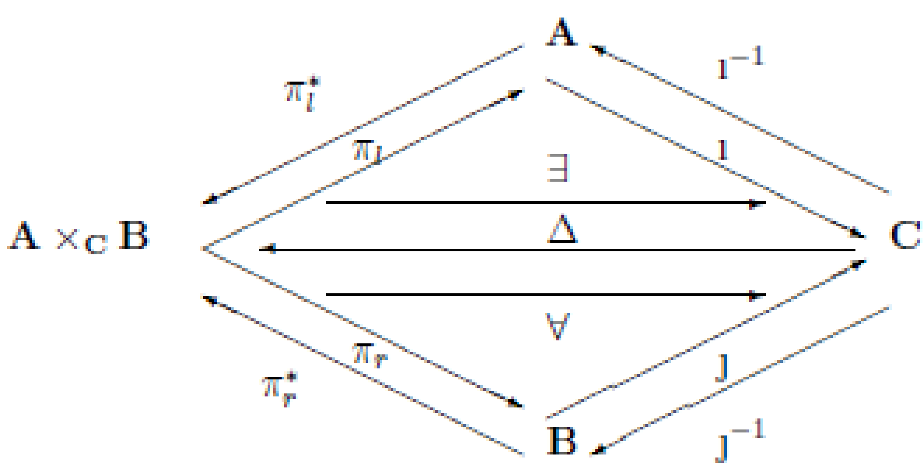


Figure 10: The Logic of a Cartesian Closed Category: the Pullback and its Dual

In regular category context, the classical relational calculus

Use of Allegories 1

- Past work developing the categorial concept:
  - Freyd & Scedrov (1990)
  - Johnstone in Elephant (2002-)
  - Freyd is main worker
  - Very little development since the early 1990s

Use of Allegories 2

- Theory of logic
  - First-order unification using variable-free relational algebra, Arias et al, Logic Journal of IGPL (2010)
  - Logic Programming in Tabular Allegories, Arias et al, Leibniz International Proceedings in Informatics (2009)
  - Logic programming in tau categories, Finkelstein, Freyd & Lipton, Computer Science Logic (1995)
  - Partial Horn logic and cartesian categories, Palmgren & Vickers, Annals of Pure and Applied Logic (2007)
  - Categories, allegories and circuit design, Brown & Hutton, Logic in Computer Science (1994)
  - Structural induction and coinduction in a fibrational setting, Hermida & Jacobs, Information and Computation (1998)

Use of Allegories 3

- Since 2014
  - Modalities for an Allegorical Conceptual Data Model, Zieliński et al, Axioms (2014)
  - Declarative Compilation for Constraint Logic Programming, Arias et al, Logic-Based Program Synthesis and Transformation (2014)
  - Unifying exact completions, Maietti & Rosolini, Applied Categorical Structures (2015)
  - Weak n-Ary Relational Products in Allegories, Zieliński & Maślanka (2014)

Use of Allegories 4

- Quotes:
  - “the theory of allegories is a generalization of relation algebra to relations between different sorts” (Wikipedia)
  - “an allegory is a category with properties meant to reflect properties that hold in a category Rel of relations” (nLab)
  - “Freyd and Scedrov's work on Allegories (replace functions in categories with relations) would be more suitable for relational databases” (Hacker News)
  - “With the definition of category, it is easy to have an idea of what is a category, but with allegories I'm totally lost” (Maths Stack Exchange)

Usage Suggests

- That
  - Allegories have been used mainly for relational systems with 1st order logic
  - Take up of the concept is far from spectacular and is not increasing in rate
  - Maybe the concept has not been found to be readily comprehensible
- Look at further facility before producing pros and cons
- Freyd's use of term allegory is more as a transformation than as an abstraction (correct!)

Allegories – the Table Category

- Allegories have a tabulation view
- Some correspondence here to the relational database model which is defined popularly in terms of tables
- Hints of categorification

Table Defined

- T: x_1, x_2, x_3, x_4, … is a table name with column names
  - T is the name of the table
  - x_i is a column (name)
- A_1, A_2, A_3, A_4, … are the feet of the table
  - A_i correspond to the values for a particular column
  - FEET = collection of A
- Table is x_i: T → FEET

Mapping between tables gives closure

- Another table (universal table?):
  - X'_i: T' → FEET'
- Natural closure:
  - Θ: T ≈ T'
  - Θ is a RELATION
    - REL(A_1, A_2, A_3, A_4, …)
  - "The usual extensional notion of relations on sets coincides with the categorical notion as applied to this case [of a table]" (Freyd 1.415 p.39)

Allegories: further views

- Further constructions possibly as allegories:
  - Hyperdoctrine
  - Bicategory
- Need to satisfy definition and properties given earlier
- Reduce cohesion of approach

Allegories: Pros

- More in spirit of category theory than **Rel**
  - Based 'best' on regular category (pullbacks)
    - Unital property with terminal object (identity for CCC)
- Not categorification with regular category basis
- Internal logic of 1st order relational calculus
- Can represent relational databases (>90% of commercial data)
- Has tabular, hyperdoctrine, bicategory views

Allegories: Cons

- Closed world assumption, Boolean logic
  - But division allegories are claimed to be Heyting
- No higher-order logic
- Not natural, no basis for metaphysics
- Number of views reduces cohesion
  - Tabular view is categorification of relational databases
    - other views may not be unital (not CCC)
- Not suited to new generation of object-bases
  - Will not form part of our work going forward on natural information systems

But Allegories could still be significant

- Allegories and topos have some commonality
  - Same underlying data structure (pullbacks)
  - Both can be viewed as regular categories
  - Both have an internal logic
- Difference is natural topos vs allegories as sets
- Potential for interoperability, major problem in information systems today
  - Relational database as allegory (**A**)
  - Natural database as topos (**T**)
  - Adjointness: F: A → T; G: T → A; F -| G

Challenge to the Sketch Workers

- Very difficult to justify the elaborate work on Entity-Relationship database models with sketches
- Allegories 'off-the-shelf' can do everything they want functionally in a relational database
- Simply add a graphical interface to an allegory-based system to complete the work
  - Regular category structure with pullback diagrams makes this readily achievable

Topos: further work identified

- Data Process
  - Queries – use of subobject classifier, particularly with power objects
  - Examples of Heyting intuitionistic logic
- Database design
  - Co-cartesian approach
  - Pasting of pullbacks
  - Recursive pullbacks
  - Allegories

Progress

Data Process	
Queries	Subobject classifier extended to power-objects for generality
Heyting examples	Stalled, as group at unn has fewer meetings
Database Design	
Co-cartesian	In progress
Pasting of pullbacks	Expressed in complex, more realistic design
Recursive pullbacks	In progress
Allegories	Explored, not useful in natural IS but significant for interoperability

The Topos going forward

- CCC
  - Products; Closure at top; Connectivity; Internal Logic; Identity; Interchangeability of levels
- Plus:
  - Subobject classifier
  - Internal logic of Heyting (intuitionistic)
  - Reflective subtopos (query closure)
- Gives
  - A Topos

More Complex Examples

- Last year's paper dealt with a single pullback as a topos
- Developing more complex pullback structures to show can handle realistic examples
- Here we extend the Student-Module example to include Departments, Universities, Lecturers.
- Pullbacks are pasted together, following laws of composition on paths
  - not intersection of paths, which would be set-based

Single Pullback: S X$_O$ M

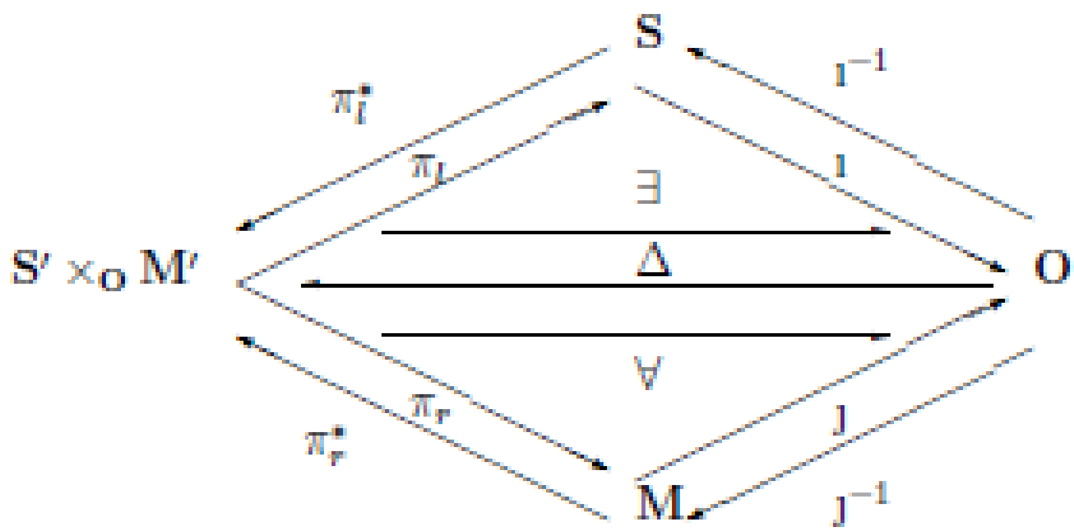


Figure 12: The Logic of the Database Example, as a Cartesian Closed Category **CCC** and its dual, with categories $\mathbf{S}, \mathbf{S}'$ for Student, $\mathbf{M}, \mathbf{M}'$ for Module, $\mathbf{O}$ for Outcome

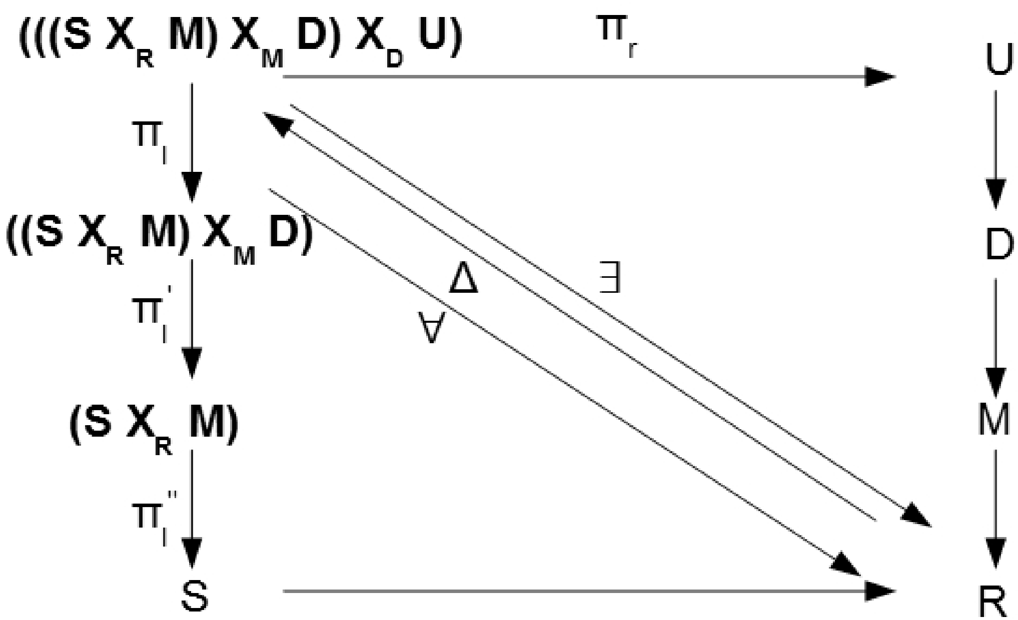
Topos – Pasted Pullbacks x6

$$
\begin{tikzcd}
(((S \ X_R \ M) \ X_M \ D) \ X_D \ U) \arrow[r, "\pi_r"] \arrow[d, "\pi_l"'] & U \arrow[d] \\
((S \ X_R \ M) \ X_M \ D) \arrow[r, "\pi_r'"] \arrow[d, "\pi_l'"'] & D \arrow[d] \\
(S \ X_R \ M) \arrow[r, "\pi_r''"] \arrow[d, "\pi_l''"'] & M \arrow[d] \\
S \arrow[r] & R
\end{tikzcd}
$$

R result, S student, M module, D department, U university

Pasted Pullback: relationships between categories R, S, M, D, U

Pasted Pullback with CCC Logic



Hierarchical Structure

R result, S student, M module, D department, U university

Pasted Pullback: relationships between categories R, S, M, D, U
and calculus $\exists \dashv \Delta \dashv \forall$
Analogous diagrams can be drawn for other commuting squares

There are potentially 6 commuting squares

Topos – Pasted Pullbacks x7

$$
\begin{tikzcd}
(((S \; X_R \; M) \; X_M \; D) \; X_D \; U) \arrow[r, "\pi_r"] \arrow[d, "\pi_l"'] & U \arrow[d] & \\
((S \; X_R \; M) \; X_M \; D) \arrow[r, "\pi_r'"] \arrow[d, "\pi_l'"'] & D \arrow[d] & (L \; X_m \; D) \arrow[l, "\pi_r'''"'] \arrow[d, "\pi_l'''"] \\
(S \; X_R \; M) \arrow[r, "\pi_r''"] \arrow[d, "\pi_l''"'] & M \arrow[d] & L \arrow[l] \\
S \arrow[r] & R &
\end{tikzcd}
$$

Network
Structure

R result, S student, M module, D department, U university,
L lecturer

Pasted Pullback: relationships between categories
R, S, M, D, U, L

Subobject Classifier – Boolean example

$$\begin{tikzcd} U \arrow[r] \arrow[d, "j"'] & 1_{topos} \arrow[d, "true"] \\ S \arrow[r, "X_j"'] & \Omega\ \{0,1\} \end{tikzcd}$$

Simple database query in category theory style

Ω {0,1} is subobject classifier; subobjects classified as either 0 or 1
X_j characteristic function is query mapping from object S to {0,1), false or true
1_{topos} is terminal object of topos (handle on topos)
j is mapping from subobject U (result of query) to object S

Diagram is actually a pullback of *true* along $X_{j.}$
U is $1_{topos}\ X_{\Omega\{0,1\}}\ S$
U is the identity of the subtopos, giving query closure

Subobject Classifier as Power-object

Defined by commuting pullback square:

$$\begin{tikzcd}
U \arrow[r] \arrow[d, "j"] & 1_{topos} \arrow[d, "true"'] \\
S(\Omega X \Omega) \arrow[r, "X_j"'] & \Omega \text{ (power-object)}
\end{tikzcd}$$

Subobject classifier Ω is non-Boolean, a power-object of some objects
Characteristic function X_j defines subobject U of object S from topos
represented by 1_{topos}
S is of type AND (intersection)
Diagram is again a pullback of *true* along $X_{j.}$
U is $1_{topos}\ X_{\Omega \text{ (power-object)}}\ S(\Omega X \Omega)$
U is the identity of the subtopos, giving query closure

Advantages of General Subobject Classifier

- Power-object represents all possible combinations of all objects
  - Basis for general search capability
- Object with type $\Omega x \Omega$
  - Facilitates comparison of all power-objects with each other
- So X_j is a general database query
- Subtopos U is result of a general query over a general object

Summary of Progress

- Topos 'data model' established as optimum way forward for information systems
- Recent work has confirmed the suitability of the model for large-scale design and general interrogation
- Next stage: provide a demonstrator project to show how system would work from design to implementation with a reasonably complex application. At same time work on remaining issues: Heyting logic, design alternatives.