

The Monad in Process-Relational Systems

Nick Rossiter

Visiting Fellow

Computing Science and Digital Technologies
Northumbria University

Whitehead 11, Azores (July 2017)

Acknowledgements

- Michael Heather
- Michael Brockway

Process-Relational Philosophy 1

- Whitehead's *Process and Reality* introduces many of the concepts of metaphysics.
- Later workers, including Robert Mesle, Margaret Stout and Mary Follett, have used the ideas of Whitehead to formulate the process-relational philosophy.
- Such a philosophy has been applied in a social context to handle creativity, Becoming, imagination and experience.
- In a language context, the same philosophy has been applied to ontology or Being.

Process-Relational Philosophy 2

- The process-relational philosophy considers that the world can be thought of a collection of interrelated processes,
 - rejecting the Cartesian dualism of Descartes, and
 - favouring the dynamic process (flux) of Heraclitus.
- Such a philosophy satisfies current requirements in computer science and information systems but has often been difficult to achieve.

Problems in Computing Science

- The basis of much of computer science is set theory,
 - provides adequately the static (Being)
 - but is restricted to process as function.
- Further, handling the logical types across the static and process components in an integrated manner is very difficult in practice, a problem encountered by Russell and Whitehead in their series on set theory, Principia Mathematica.
- A single-level approach is inadequate for the complexities of information systems.

Process and Reality

- Much of Whitehead's Process and Reality can be considered as informal category theory
 - preceding the later developments in pure mathematics, starting in the 1940s by such workers as Eilenberg and Mac Lane (EML Category Theory)
- For instance Whitehead's category of prehension, or grasping, corresponds to the categorial adjunction.
- Other examples are that Whitehead's category of the ultimate corresponds to the topos and his category of existence to the Cartesian Closed category.

Process-relational and Category Theory

- In this paper we consider how the process-relational philosophy, naturally arising from Process and Reality, can be considered formally in category theory by the monad, which relates inputs and outputs through adjointness.
- The monad operates on a category, such as a topos, over three-levels, providing the necessary closure of being defined as unique up to natural isomorphism.
- The term monad is very 'old' but was made better known by Leibniz. We have made a comparison of the various usages of the term, including its use today in mathematics and computer science.

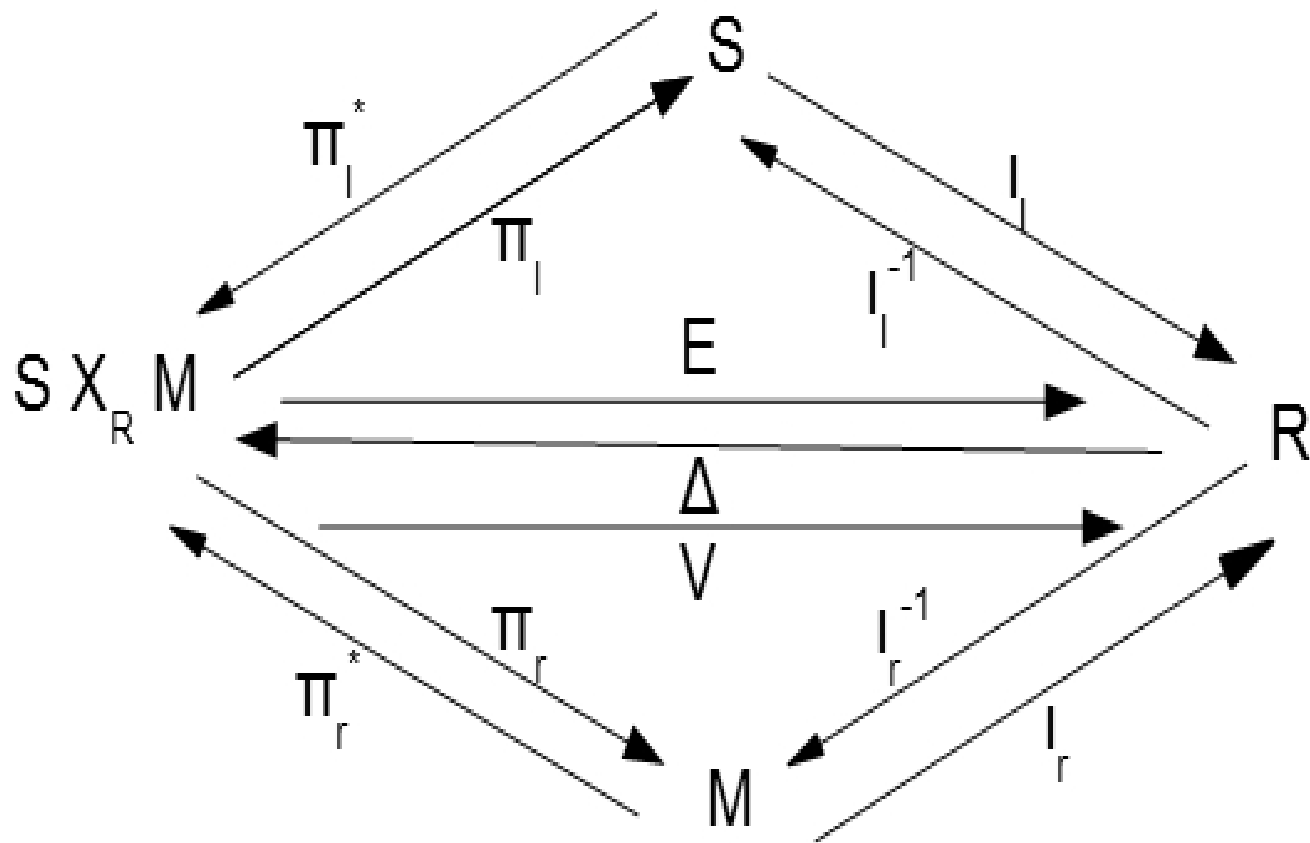
The Topos – Structural Data-type

- Is a Cartesian Closed Category (CCC)
 - Products; Closure at top; Connectivity (exponentials); Internal Logic of λ calculus; Identity; Interchangeability of levels
- If we add:
 - Subobject classifier
 - Internal logic of Heyting (intuitionistic)
 - Reflective subtopos (query closure)
- We get a Topos

Structural Examples

- Student Marks
 - Simple (single pullback)
- Bank Transactions
 - Simple (single pullback)
 - Simple pasted (2 pasted squares, 3 pullbacks)
 - Complex (5 pasted squares, 10 pullbacks)
 - Complex structure (5 pasted squares, not valid pullback)

Pullback - Single Relationship Student Marks



Pullback - Single Relationship Constraints

- $S \times_R M$ (Student X_{Result} Mark)
- Logic of adjointness: $E \dashv \Delta \dashv V$
 - Δ selects pairs of S and M in a relationship in context of R
 - Such that $E \dashv \Delta$ and $\Delta \dashv V$
 - Termed by Lawvere as a hyperdoctrine
- Projections π are from product, left and right (dual π^*)
- Inclusions ι are into sum $S+M+R$, left and right (dual ι^{-1})
- S, M, R are categories, with internal pullback structure, giving recursive pullbacks

Recursive Pullbacks

A node of a pullback may itself be a pullback

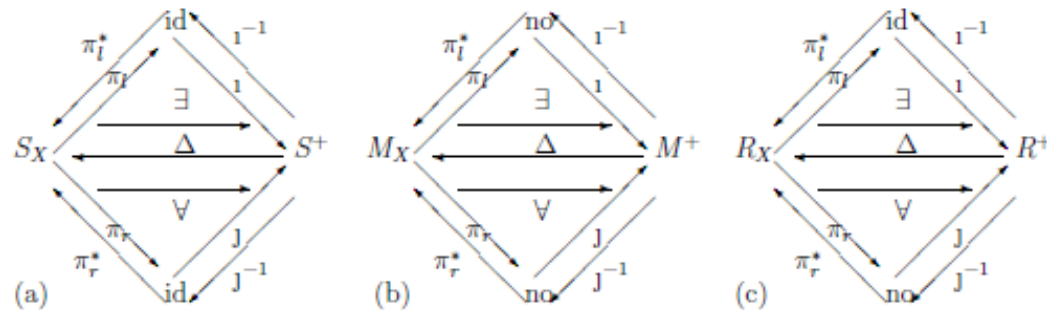
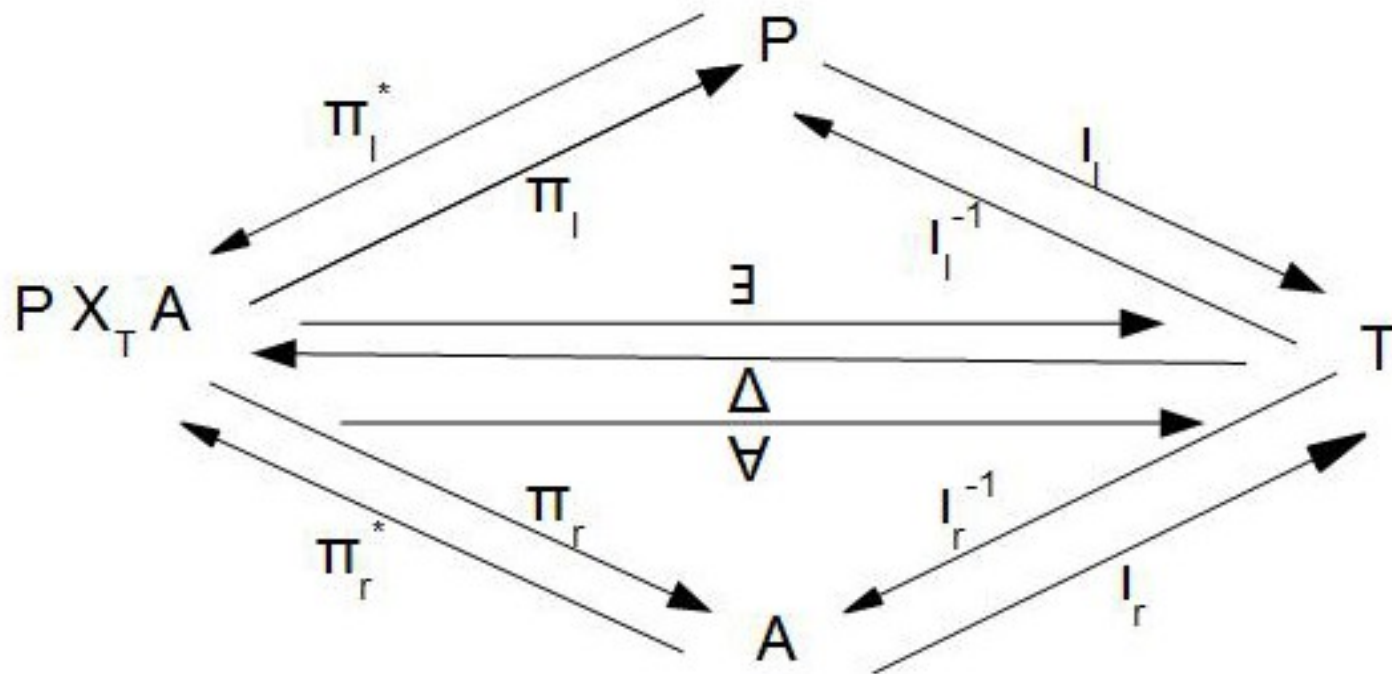


Figure 13: Internal Structure of Categories: a) The Pullback in \mathbf{S} . S_X is $\text{id} \times_{S^+} \text{id}$, S^+ is name +_{id} address. b) The Pullback in \mathbf{M} . M_X is $\text{no} \times_{M^+} \text{no}$, M^+ is title +_{no} grade, c) The Pullback in \mathbf{R} . R_X is $\text{id} \times_{R^+} \text{no}$, R^+ is mark +_{id+no} decision.

Each node in the pullback for Student over Marks in context of Result is itself a pullback, giving a recursive structure

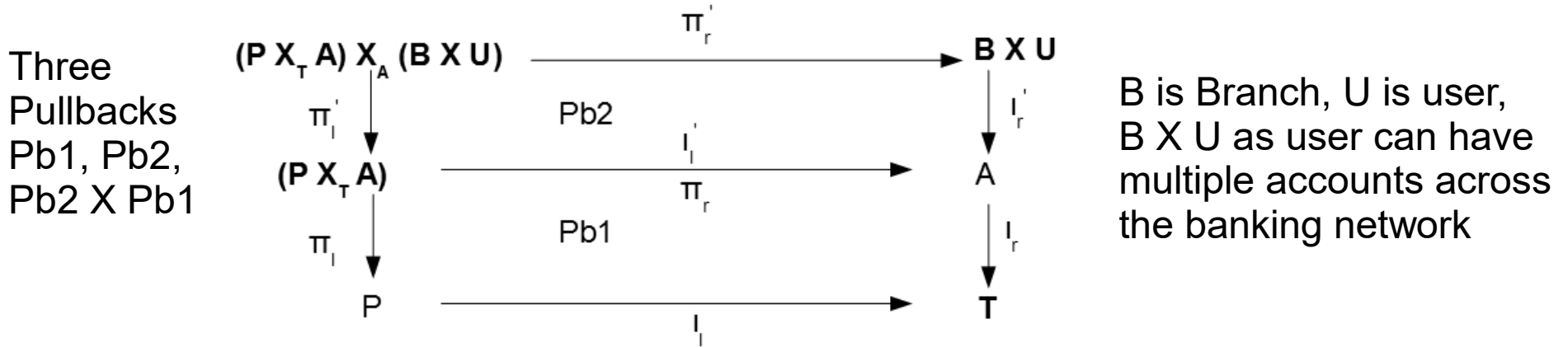
Pullback - Single Relationship: Bank Transactions by Procedure and Account



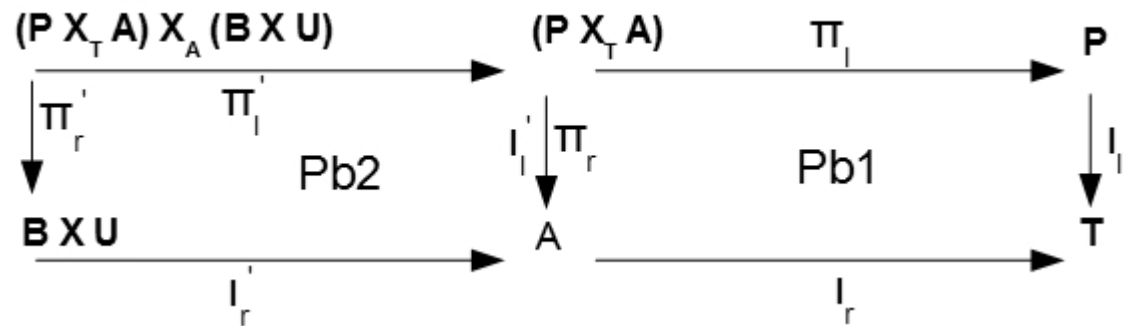
Pullback - Single Relationship Details

- $P \times_T A$ (Procedure $X_{\text{Transaction}}$ Account)
 - Procedure is type of transaction: e.g. standing order, direct debit, ATM cash withdrawal
 - Account can belong to many users
 - Transaction is item for transfer of funds according to ACID requirements
- P, A, T are categories, with internal pullback structure, giving recursive pullbacks

Pullback - Two Pasted Relationships: Bank Transactions by Branch/User



Usually written in horizontal (landscape) form. Vertical layout enables deep nested structures to be represented more readily

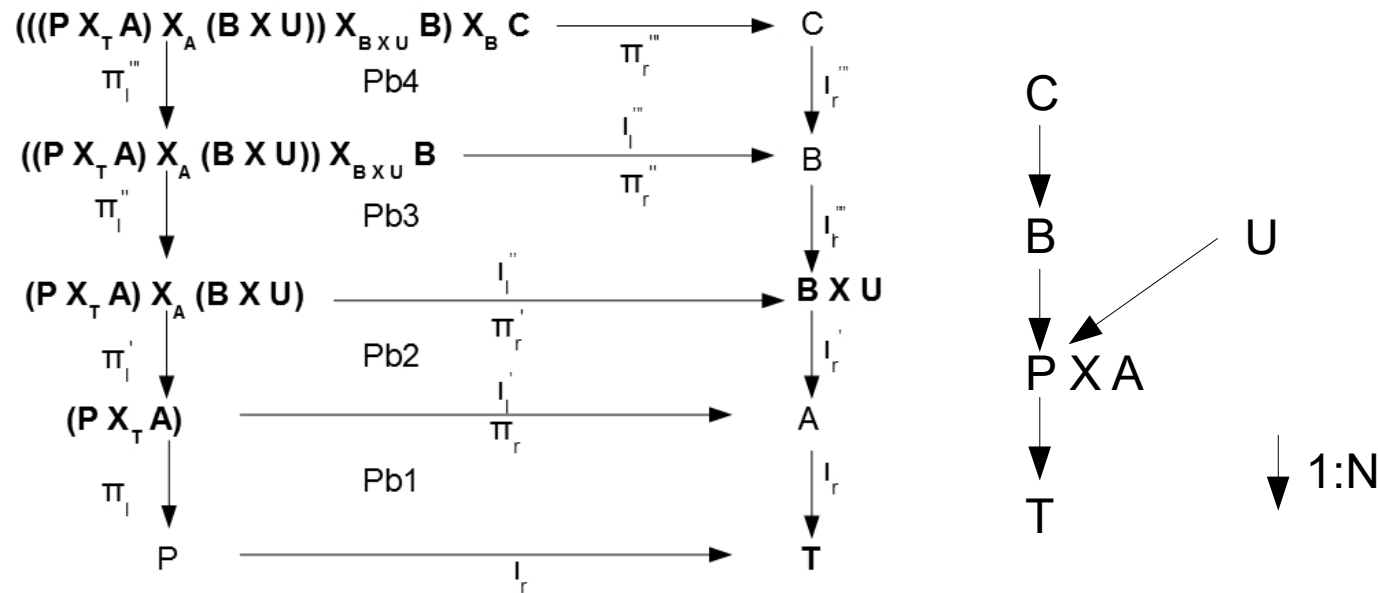


Pasting condition for Pb2 X Pb1: $i'_l = \pi_r$ after Freyd's Pasting Lemma

For our purposes, a pasted pullback is only a valid pullback if all inner and outer diagrams are pullbacks

Pasting is associative (order of evaluation is immaterial) but not commutative (relationship A:B 1:N is not same as A:B N:1)

Pullback – x10 Natural Bank Account Transactions



C company, B branch, U user, A account, P procedure, T transaction

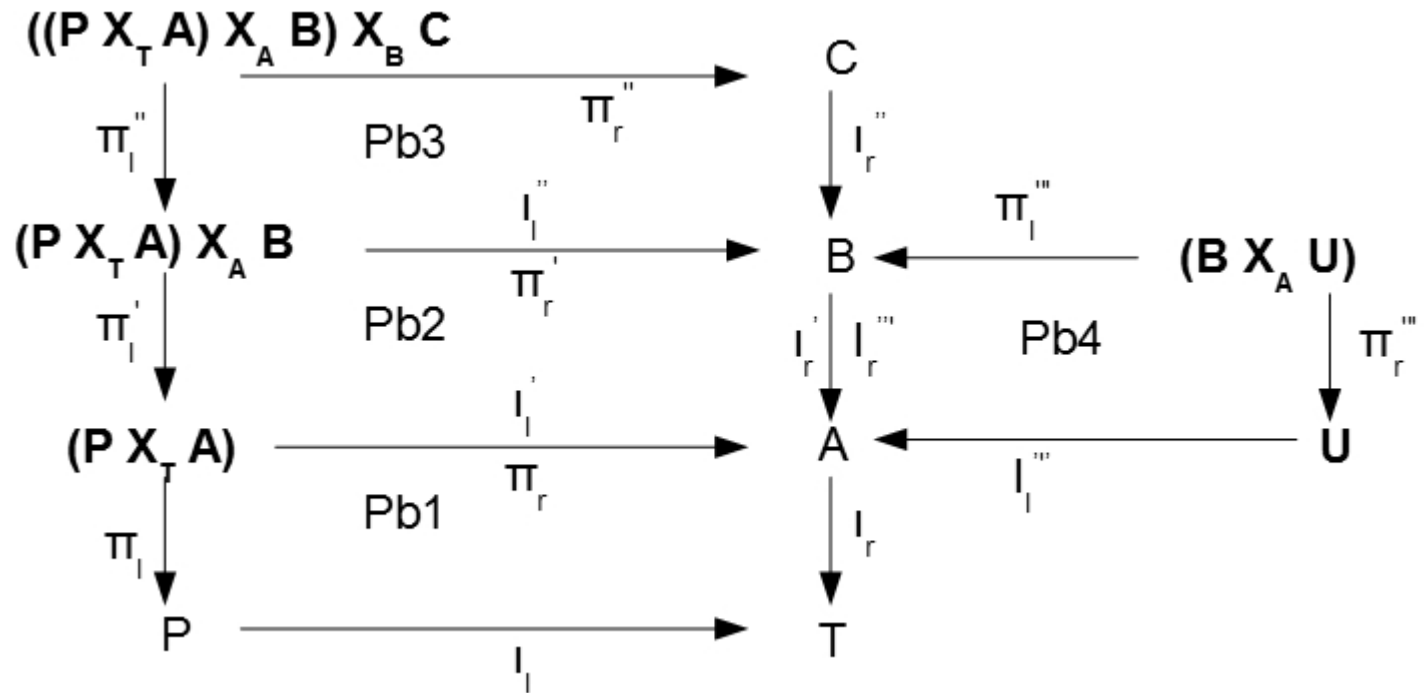
- 10 pullbacks: Pb1, Pb2, Pb3, Pb4
 Pb2 X Pb1, Pb3 X Pb2, Pb4 X Pb3
 Pb3 X Pb2 X Pb1, Pb4 X Pb3 X Pb2
 Pb4 X Pb3 X Pb2 X Pb1

For our purposes, a pasted pullback is only a valid pullback if all inner and outer diagrams are pullbacks

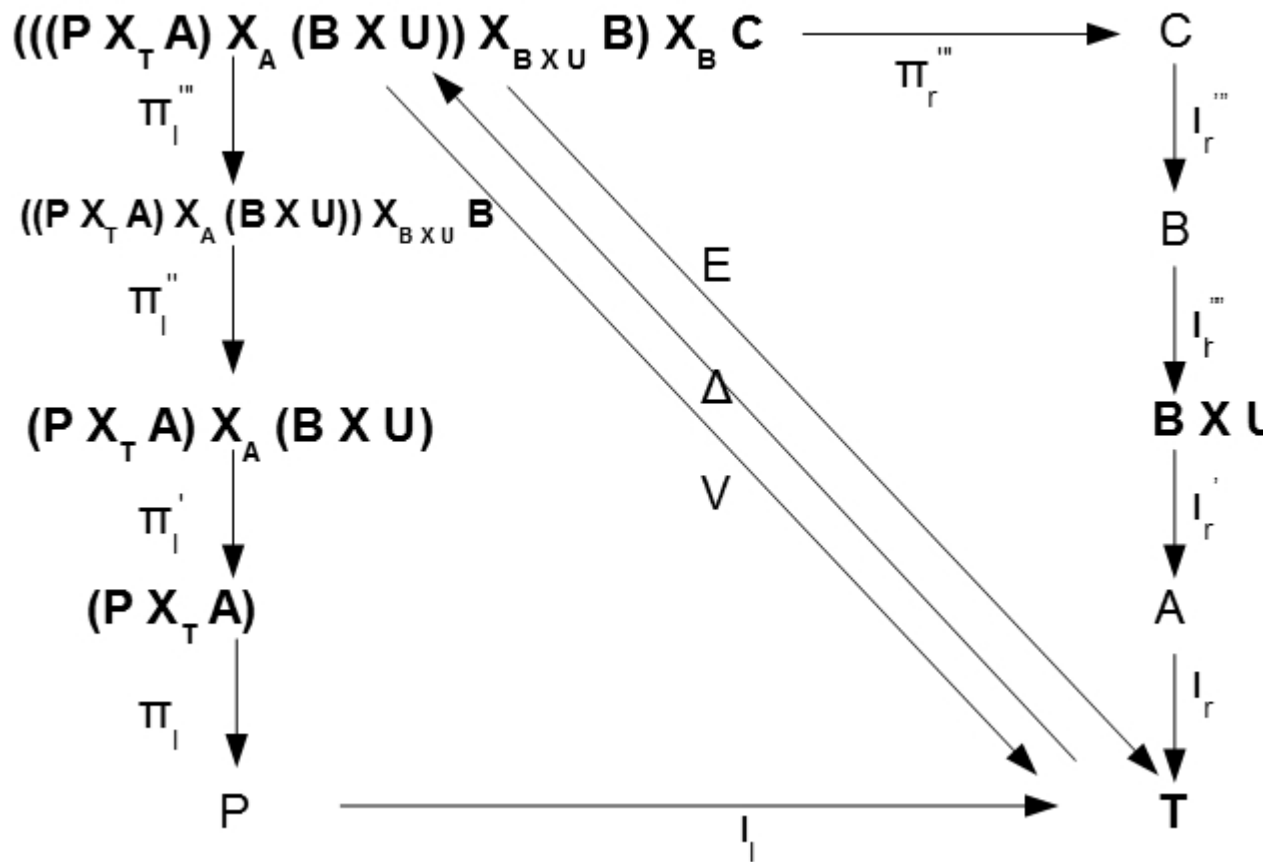
Invalid Pullback

Invalid as
not all squares
are pullbacks

For instance
Pb4 X Pb2 is not
a pullback



Adjointness Holds for all Pullbacks



$E \dashv \Delta \dashv V$ for this outer pullback and all other 9 pullbacks

Pasting Pullbacks – Discussion 1

- All diagrams commute
- All diagrams, inner or outer, are pullbacks
 - In pure maths, the condition is relaxed a little
 - Not appropriate for applied
- Structure is recursive
 - A pullback node may be a pullback structure in its own right
 - No limit to recursion

Pasting Pullbacks – Discussion 2

- Pasting condition appears to be:
 - $i'_l = \pi_r$ (left-inclusion of outer square = right-projection of inner square)
 - Discussed further later

Pasting Pullbacks – Discussion 3

- Pasted structure
 - is a Cartesian Closed Category (CCC) with products, terminal object and exponentials
 - is a topos as a CCC with subobject classifier and internal Heyting Logic
- The subobject classifier provides an internal query language

The Pasting Condition 1

- $\iota_l' = \pi_r$ (left-inclusion of outer square = right-projection of inner square
 - Looks rather set theoretic
- But any pullback can be represented as an equalizer (ncatlab)

Equalizer for Pullback

$$\begin{array}{ccc} \mathbf{P} \times_{\mathbf{T}} \mathbf{A} & \longrightarrow & \mathbf{P} \times \mathbf{A} \\ & & \xrightarrow{\quad l_l \circ \pi_l \quad} \mathbf{T} \\ & & \xrightarrow{\quad l_r \circ \pi_r \quad} \mathbf{T} \end{array}$$

The Pasting Condition 2

Similarly for a pasted pullback, the equaliser is

$$\begin{array}{ccc}
 (P \times_T A) \times_A (B \times U) & \longrightarrow & (P \times_T A) \times (B \times U) \\
 & & \begin{array}{c} \xrightarrow{\quad \pi_l \pi_l' \quad} \\ \xrightarrow{\quad \pi_r \pi_r' \quad} \end{array} \\
 & & A
 \end{array}$$

Equals in sets is undefined as context is not defined

Equaliser in categories, as a limit, is fully defined up to natural isomorphism

External Process

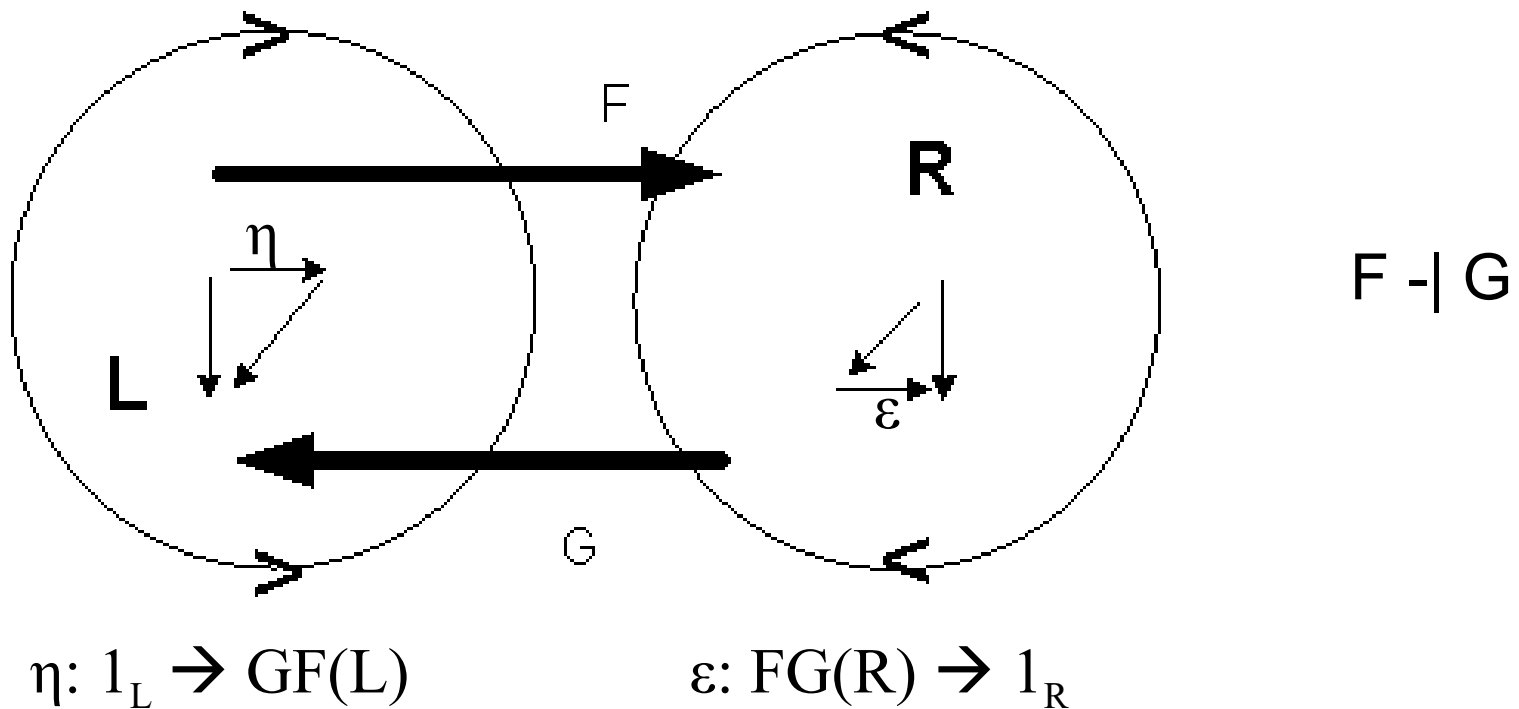
- Metaphysics (Whitehead)
- Transaction (universe, information system)
- Activity
 - Can be very complex but the whole is viewed as atomic – binary outcome – succeed or fail
 - Before and after states must be consistent in terms of rules
 - Intermediate results are not revealed to others
 - Results persist after end

Transaction in Category Theory

- In earlier work (ANPA 2010) we used adjointness to represent a transaction
 - Employing multiple cycles to capture ACID
- The aim now is to abstract this work using the monad, which we earlier described as the way forward
- The monad is an extension of the monoid to multiple levels
 - Monoid: $M \times M \rightarrow M$, $1 \rightarrow M$ (binary multiplication, unit)

Multiple 'Cycles' to represent adjointness

- Three 'cycles' GFGFGF:
 - Assessing unit η in L and counit ε in R to ensure overall consistency
 - 'Cycles' are performed simultaneously (a snap, not each cycle in turn)



Promising Technique - Monad

- The monad is used in pure mathematics for representing process
 - Has 3 'cycles' of iteration to give consistency
- The monad is also used in functional programming to formulate the process in an abstract data-type
 - In the Haskell language the monad is a first-class construction
 - Haskell B Curry transformed functions through currying in the λ -calculus
 - The Blockchain transaction system for Bitcoin and more recently other finance houses uses monads via Haskell
 - Reason quoted is it's a simple, reliable and clean technique

Monad/Comonad Overview

- Functionality:
 - Monad
 - $T^3 \rightarrow T^2 \rightarrow T$ (multiplication)
 - 3 'cycles' of T , looking back
 - Comonad (dual of monad)
 - $S \rightarrow S^2 \rightarrow S^3$ (comultiplication)
 - 3 'cycles' of S , looking forward
- Objects:
 - An endofunctor on a category X

Using the Monad Approach

- A monad is a 4-cell $\langle 1, 2, 3, 4 \rangle$
 - 1 is a category X
 - 2 is an endofunctor ($T: X \rightarrow X$, functor with same source and target)
 - 3 is the unit of adjunction $\eta: 1_x \rightarrow T$ (change, looking forward)
 - 4 is the multiplication $\mu: T X T \rightarrow T$ (change, looking back)
- A monad is therefore $\langle X, T, \eta, \mu \rangle$ (or $\langle T, \eta, \mu \rangle$ or $\langle T, \eta, G \& F \rangle$ or in usage T)

The Comonad

- The dual of the monad
- A comonad is a 4-cell $\langle 1, 2, 3, 4 \rangle$
 - 1 is a category X
 - 2 is an endofunctor ($S: X \rightarrow X$, functor with same source and target, S is dual of T)
 - 3 is the counit of adjunction $\varepsilon: S \rightarrow 1_X$ (change, looking back)
 - 4 is the comultiplication $\delta: S \rightarrow S X S$ (change, looking forward)
- A comonad is therefore $\langle X, S, \varepsilon, \delta \rangle$ (or $\langle S, \varepsilon, \delta \rangle$ or $\langle S, \varepsilon, F \eta G \rangle$ or in usage S)

Monad is often based on an adjunction

- The transaction involves GF, a pair of adjoint functors $F \dashv G$
 - $F: X \rightarrow Y$
 - $G: Y \rightarrow X$
- GF is an endofunctor as category X is both source and target
- So T is GF (for monad)
- And S is FG (for comonad)

Process: Operating on a Topos

- The operation is simple:
 - $T: E \rightarrow E'$
 - where T is the monad $\langle GF, \eta, G\varepsilon F \rangle$ in E, E' , the topos, with input and output types the same
- The extension (data values) will vary but the intension (definition of type) remains the same
- Closure is achieved as the type is preserved

Composability is the Key

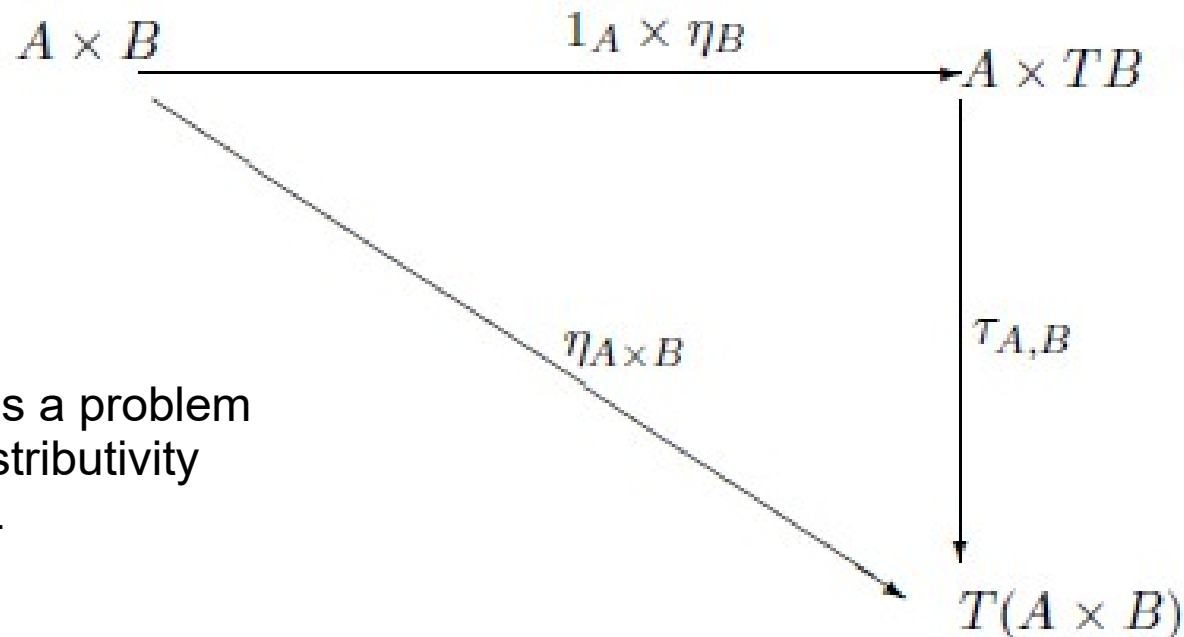
- Compose many monads together to give the power of adjointness over a whole wide-ranging application
- In banking (Bitcoin) the reliability obtained from composing processes over a wide-range of machines (distributed data recovery) justifies the move to Category Theory
- There is a problem though in EML Category Theory:
 - Monads do not compose naturally

Haskell and Monads

- Kleisli Category of a Monad
 - Transforms a monad into a monadic form more suitable for implementation in a functional language
 - Used in Haskell rather than the pure mathematics form of Mac Lane
- Strengthens the monad for composability
 - As in the Cartesian Monad, with products
- A practical application of the pure maths has exposed problems in the maths
- Solution has come from another pure mathematician Kleisli

Kleisli Lift

- Define a natural transformation:
 - $\tau_{A,B} : A \times TB \rightarrow T(A \times B)$ where A, B are objects in X and T is the monad such that the following diagram commutes



There is a problem
with distributivity
In EML

Summary of Progress/Look forward

- Topos has been established as data-type of choice
- Monad shows potential for processing the topos
- Advent of Haskell gives an experimental test-bed
- Next application area is music (Music as a Composition of Cartesian Monad over a Topos, ANPA 38, Hampshire, UK, August 2017)