Music as a Composition of Cartesian Monad over a Topos

Nick Rossiter
Visiting Fellow
Computing Science and Digital Technologies
Northumbria University

ANPA 38
St John's College,
Rowlands Castle, Hampshire, UK
10 August 2017

Acknowledgements

- Michael Heather
- Michael Brockway
- Members of the Royal Northern Sinfonia,The Sage, Gateshead

Outline of Presentation

- The work to be presented builds on that presented at ANPA 37, taking up the challenge of a testing application for the Cartesian monad approach to universal design.
- The monad presents a musical performance as a composition over time signatures, such as barlines, with the monad looking forward/back and its associated comonad looking back/forward.

Outline of Presentation 2

- The physical characteristics of the notes in each time-frame are complex, so it is necessary to use a strong Cartesian monad, facilitating the representation of each time-frame as a product.
- The monad is process, handling dynamic aspects. The category upon which the monad operates will be a topos holding relatively static information such as the players, the score and the venue, together with the relationships between them.

Outline of Presentation 3

- The topos is far from totally static with its arrows facilitating flexibility in all information held, including relationships; the topos is also searchable through the subobject classifier.
- There is no assumption of any particular musical genre.
- Such a categorial framework could be implemented in the functional programming language Haskell in a similar way to the banking example.

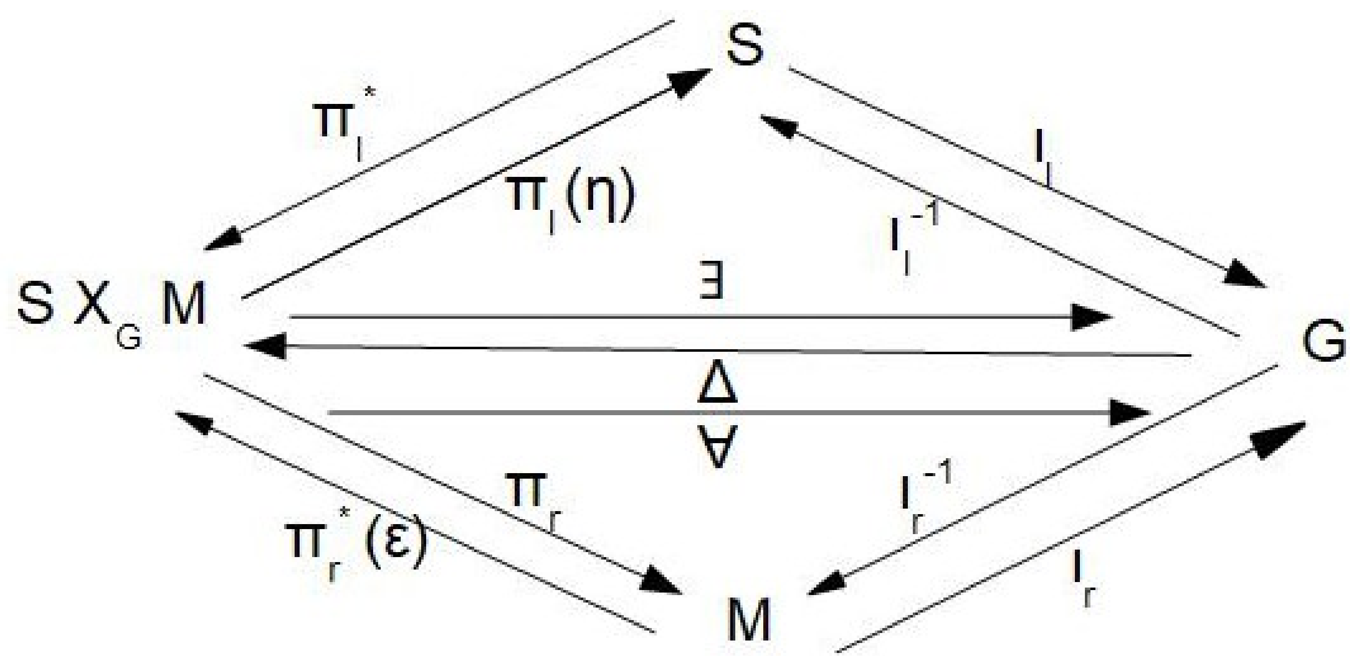
The Topos – Structural Data-type

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

Examples

- Student Marks
  - Simple pullback (1 square)
- Bank Transactions
  - Simple pullback (1 square)
  - Pasted pullback (2 pasted squares, 3 pullbacks)
  - Pasted pullback (4 pasted squares, 10 pullbacks)

Pullback - Single Relationship
Student Marks by Grade



Pullback - Single Relationship Constraints

- SX_G M (Student X_{Grade} Mark)
- Logic of adjointness: $\exists \dashv \Delta \dashv \forall$
  - Δ selects pairs of S and M in a relationship in context of G
  - Such that $\exists \dashv \Delta$ and $\Delta \dashv \forall$
- Projections π are from product, left and right (dual π^*)
- Inclusions ι are into sum S+M+G, left and right (dual ι^{-1})
- S, M, G are categories, with internal pullback structure, giving recursive pullbacks
- η is the unit of adjunction (creativity), ε is the counit of adjunction (qualia)

Recursive Pullbacks

A node of a
pullback may
itself
be a pullback

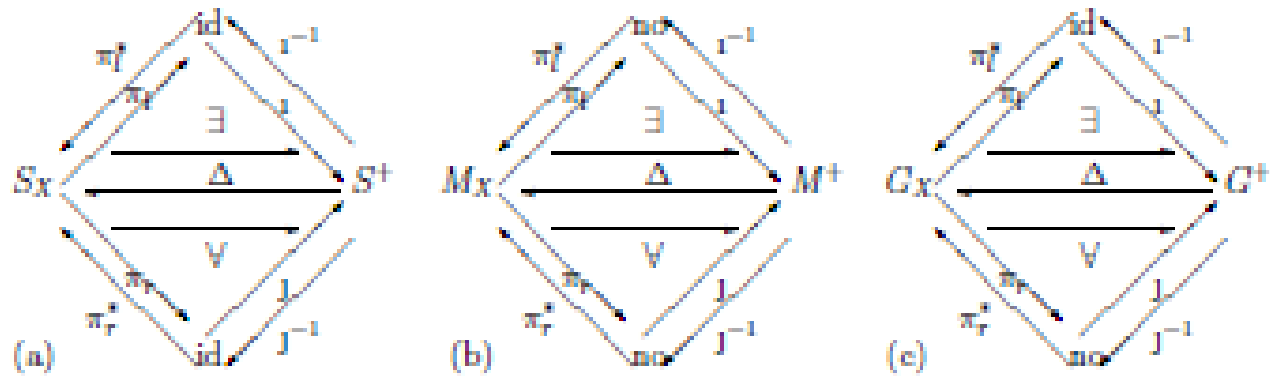
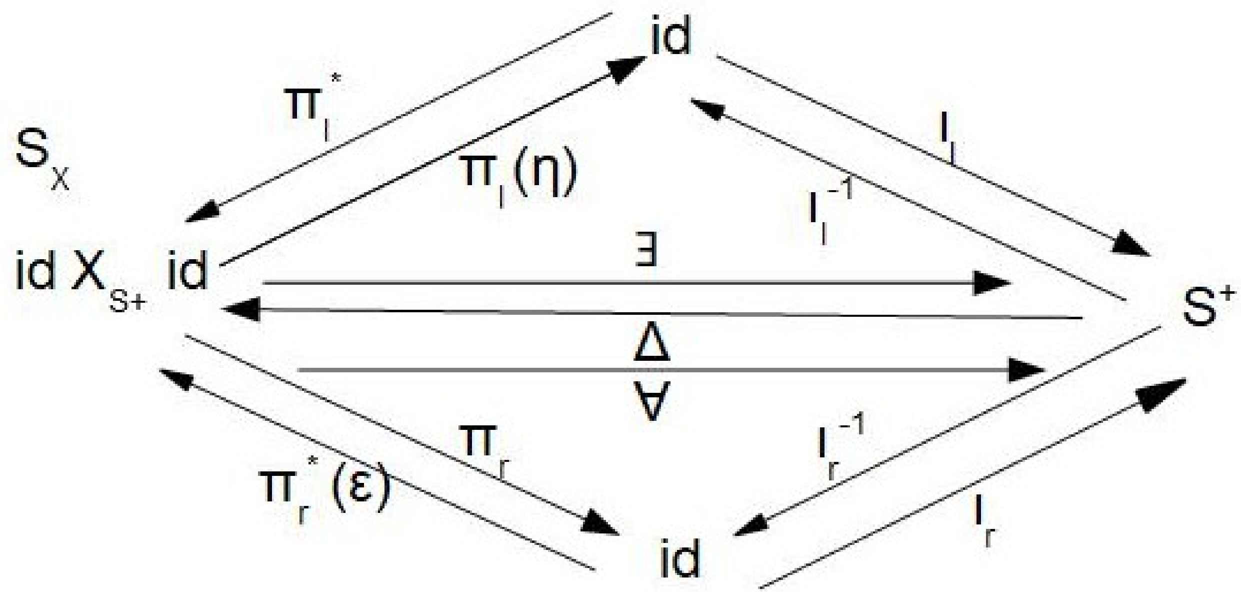


Figure 2: Internal Structure of Categories: a) The Pullback in **S**. S_X is id $\times_{S^+}$ id, S^+ is name $+_{\mathrm{id}}$ address. b) The Pullback in **M**. M_X is no $\times_{M^+}$ no, M^+ is title $+_{\mathrm{no}}$ grade, c) The Pullback in **G**. G_X is id $\times_{G^+}$ no, G^+ is mark $+_{\mathrm{id}\times\mathrm{no}}$ decision.

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

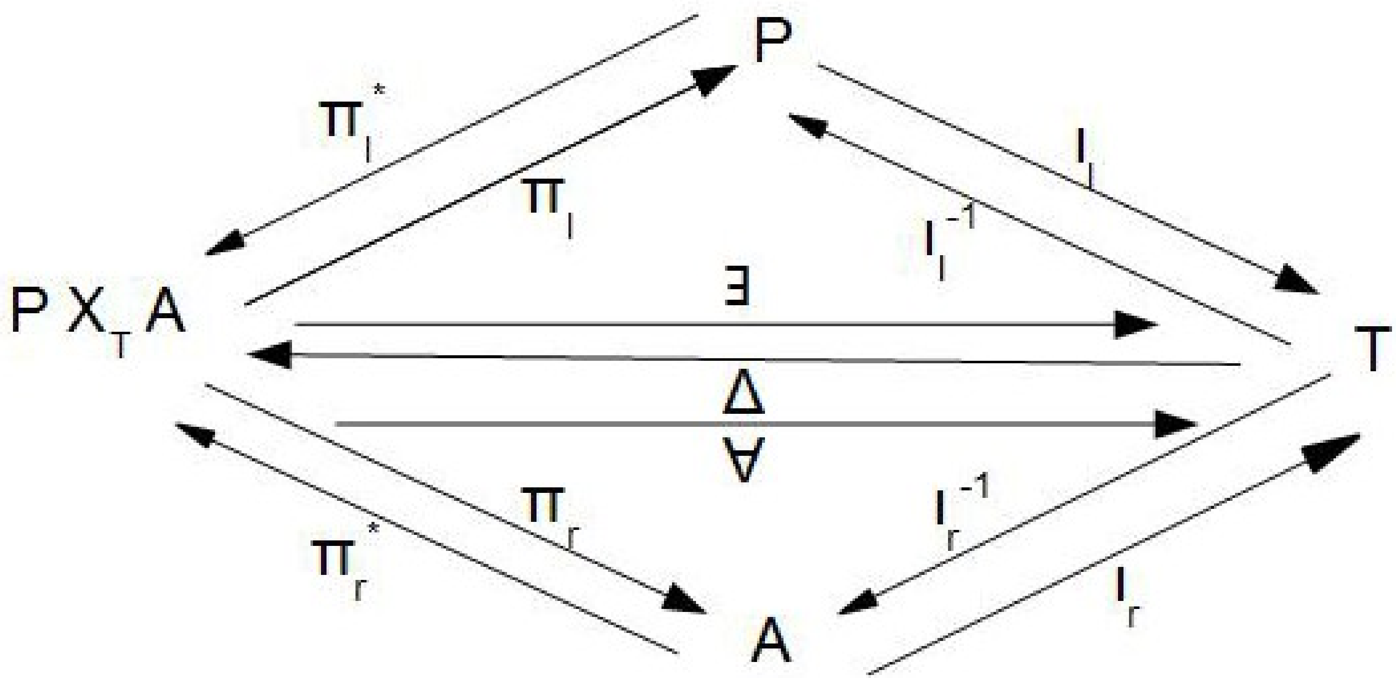
These are Dolittle diagrams (pulation squares). See Adámek, 1990 (p.205), Herrlich 2007, Freyd 1990.
The Story of Dr Dolittle by Hugh Lofting (1920), Pushmi-pullyu.
Endofunctors relate top (intension) to bottom (extension).
Each pullback node should decompose ultimately into a Dolittle diagram.

Dolittle Diagram for Category S



id is the key (identifier) for a student
S^+ is all information held on a student
S^+ is name $+_{id}$ address

Pullback - Single Relationship: Bank Transactions by Procedure and Account

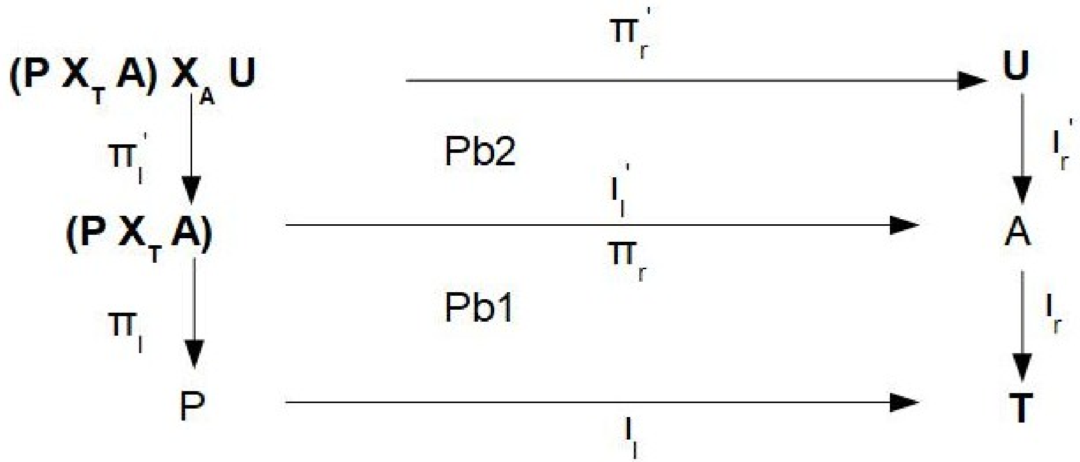


Pullback - Single Relationship Details

- P X_T A (Procedure $X_{Transaction}$ 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

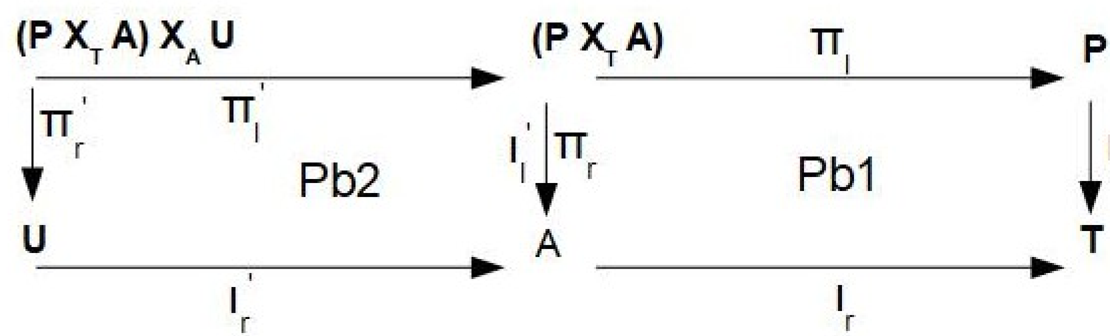
Pullback - Two Pasted Relationships: Bank Transactions bv User/Account

Three Pullbacks Pb1, Pb2, Pb2 X Pb1



U is user
A is account
T is transaction

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

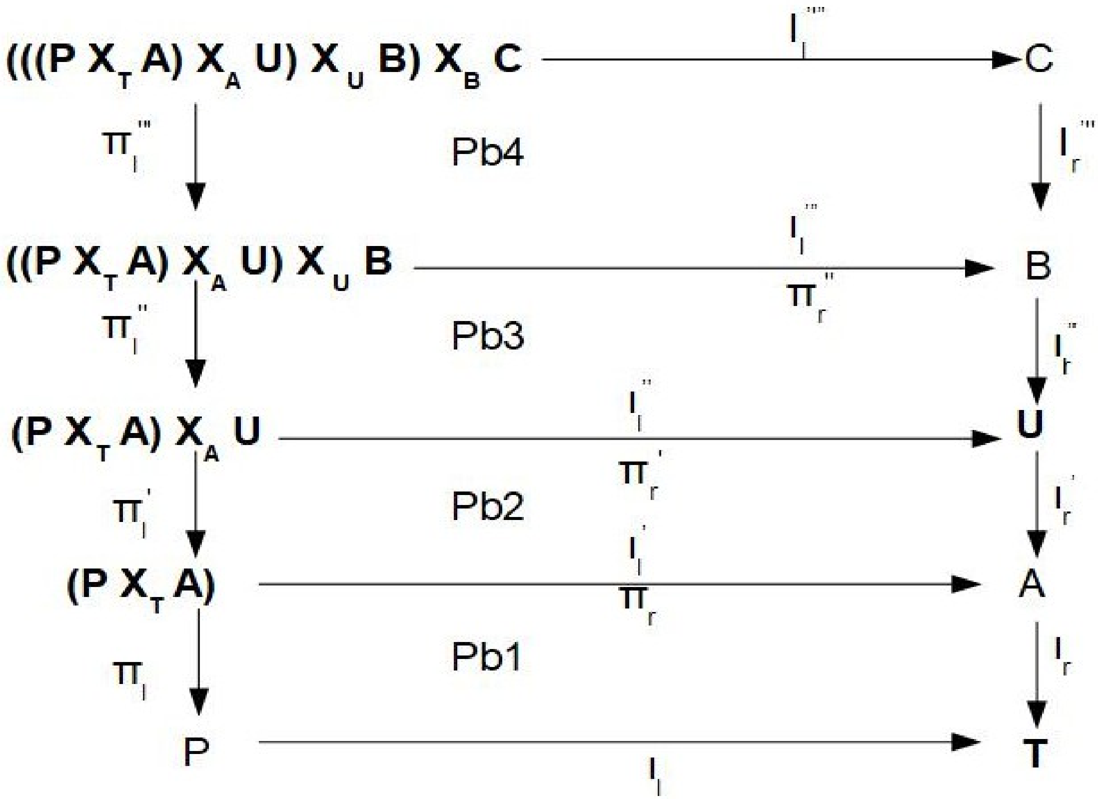


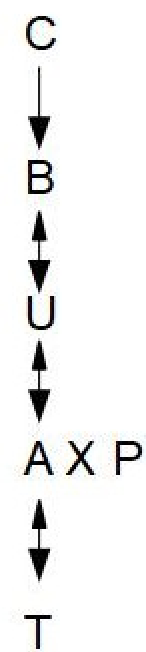
Pasting condition for Pb2 X Pb1: $\iota_l' = \pi_r$ after Freyd's Pasting Lemma

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

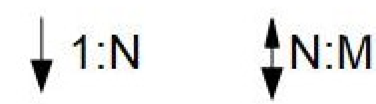
Pasting is associative (order of evaluation is immaterial) but not commutative (relationship A:B 1:N is not same as A:B N:1)

Pullback – x10 Natural Bank Account Transactions





N:M and 1:N are handled by same pullback structure



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

Data Structuring with Pullbacks

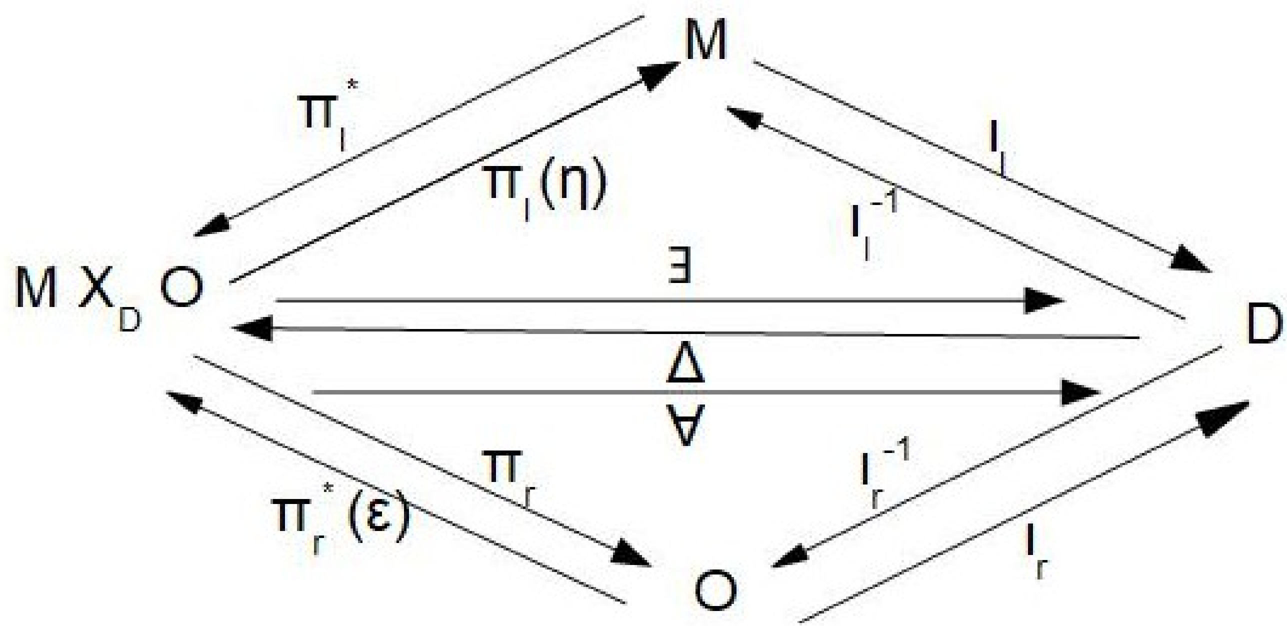
- Pasting of pullbacks is desirable when the related entities have stand-alone existence e.g. (Bank) Branch and User
- Expansion of information on an entity, as through hierarchies, may be best handled by nesting pullback structures recursively
- This is still an experimental area and another example will increase perspective

A Topos for Music

- Music is viewed as a communication of some manuscript by communicators
- The topos is relatively static (compared to the monad) but being arrow-based can readily handle change.
- Manuscript comprises scores and other intentions of composers and writers
    - Includes musical notation (typeset, handwritten or digital) or more spontaneous formats
- Communicators comprise performers and other aspects of performance
    - Includes an orchestra, group, recording company

Topos of Manuscript by Performers in Context of Delivery

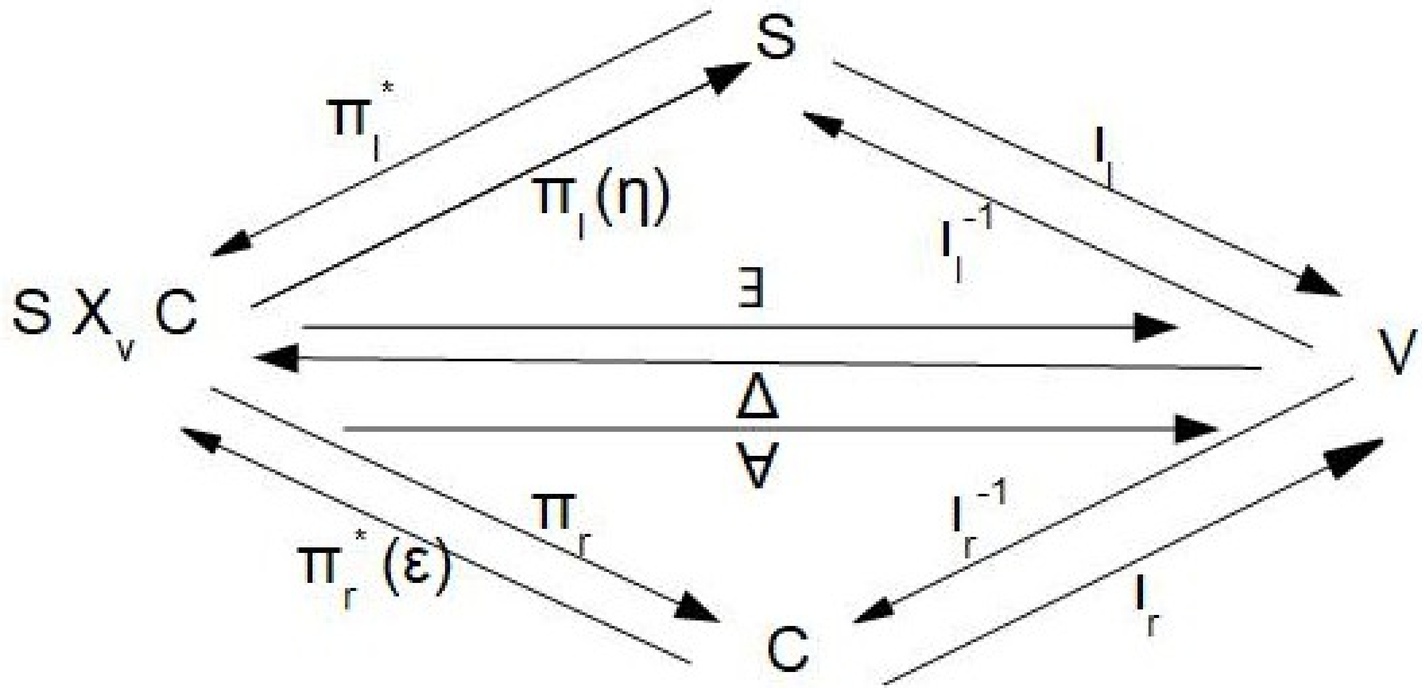
- Pullback top E



M is category for Manuscript, O for Orchestra, D for Delivery

Each of the nodes can be expanded 1

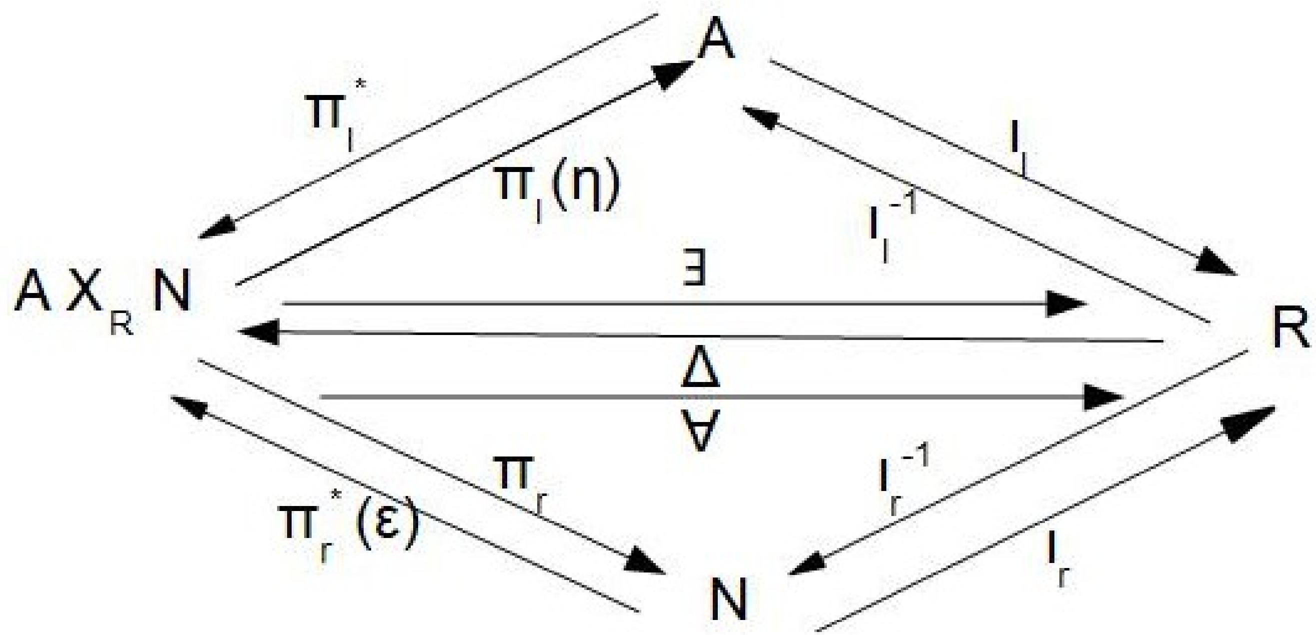
- M (Manuscript) could be



S is category for Score, C for Composer, V for Version (variant)

Each of the nodes can be expanded 2

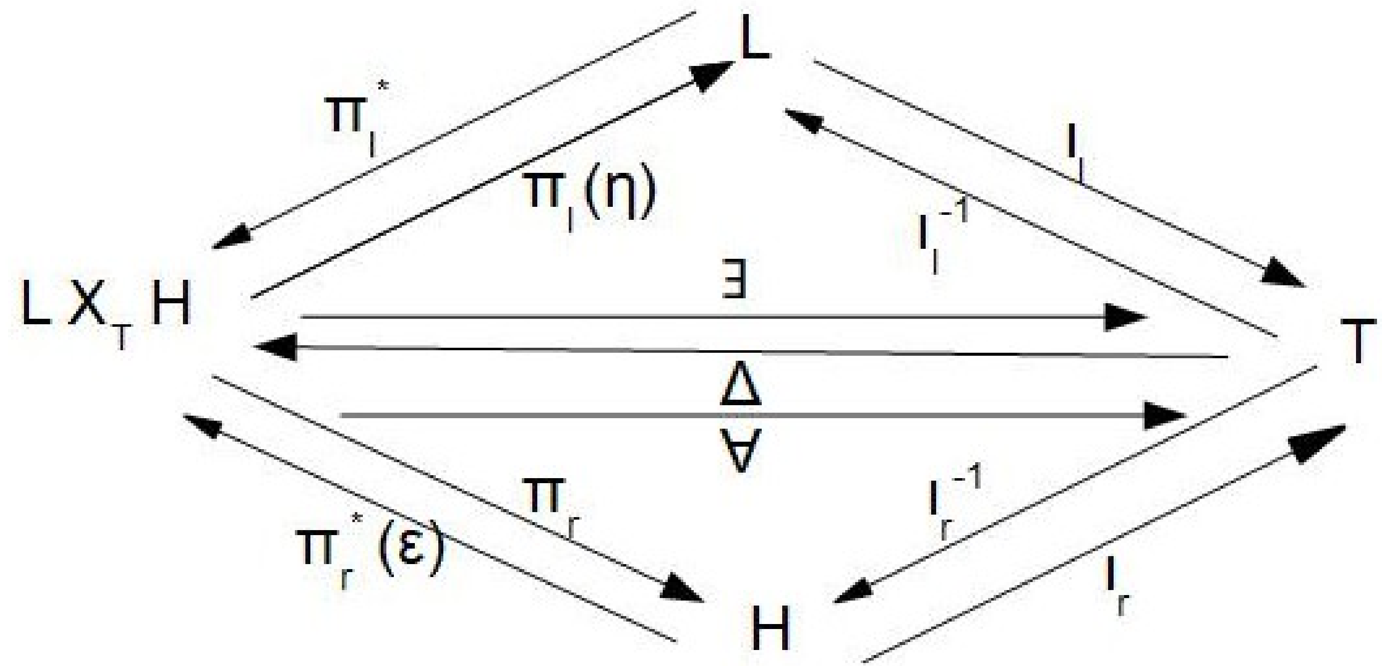
- O (Orchestra) could be



A is category for Assemblage, N for Named Musician,
R for Role

Each of the nodes can be expanded 3

- D (Delivery) could be:



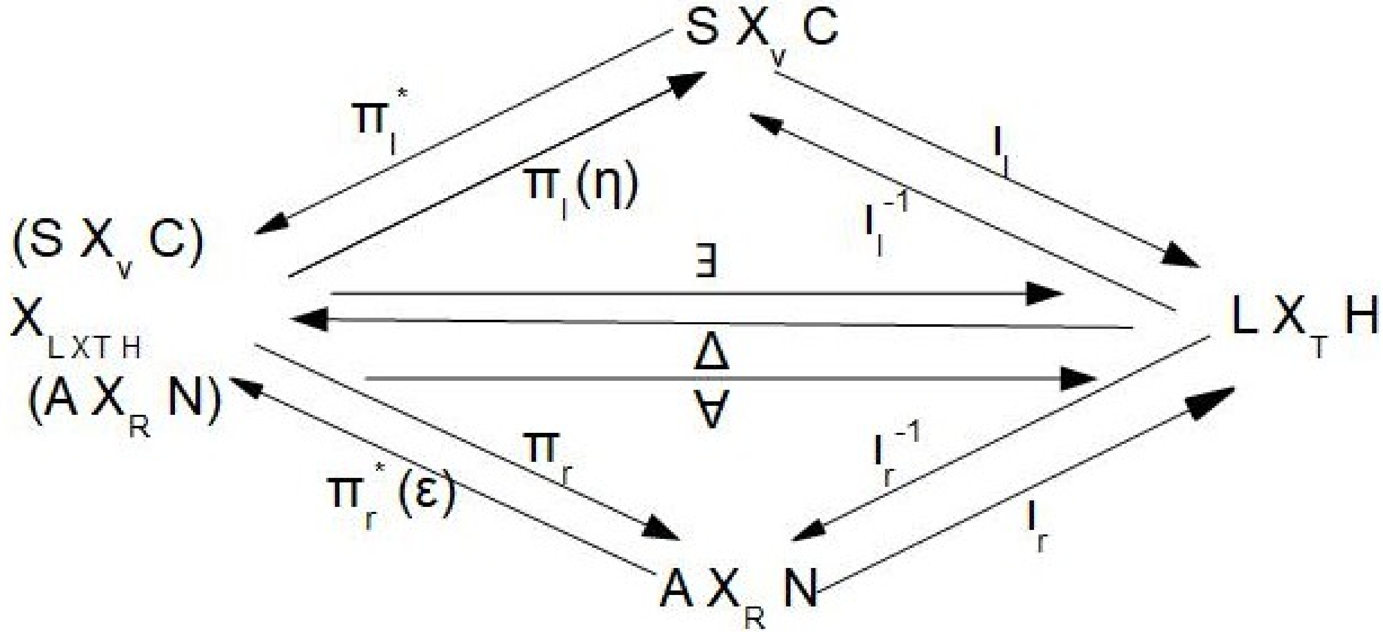
L is category for Location, H for Hall,
T for Time

Notes on Expansions

- The nodes in the top diagram E are pullbacks in their own right
- Need to match across the various nested levels with the logic
- The top digram is effectively a pullback of pullbacks as shown next

The Topos E

- Overall Pullback E



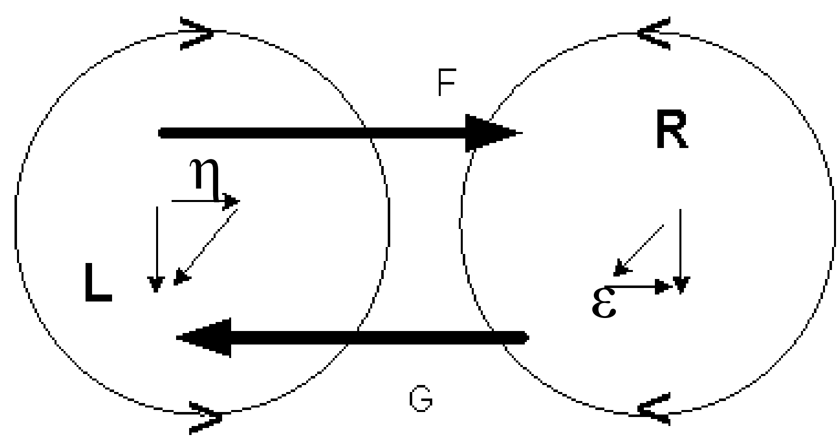
The nodes are pullback squares
Categories may be nested further

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

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)
  - Conceptually cycle 1 for execution, 2 for review, 3 for tidy-up



F -| G

$\eta: 1_L \rightarrow GF(L)$ $\varepsilon: FG(R) \rightarrow 1_R$

Failure in Adjointness

- Means transaction has failed
- Communication is suspended
- Restart is necessary at some convenient point (Rollback)
- In music need to distinguish between a wrong note and differences in expression

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: it is a simple, reliable and clean technique

Monad can be based on an adjunction

- The transaction involves GF, a pair of adjoint functors F -| G
  - F: X → Y
  - G: Y → 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)

Monad/Comonad Overview

- Functionality for free functor T, underlying functor S
  - Monad
    - $T^3 \rightarrow T^2 \rightarrow T$ (multiplication)
    - 3 'cycles' of T
    - In Bitcoin considered to be zooming-in
  - Comonad (dual of monad)
    - $S \rightarrow S^2 \rightarrow S^3$ (comultiplication)
    - 3 'cycles' of S
    - In Bitcoin considered to be zooming-out
- Objects:
  - An endofunctor on a category E (the topos)
- Note this multiple performance matches our transaction approach, outlined earlier with GF performed 3 times

Using the Monad Approach

- A monad is a 4-cell <1,2,3,4>
  - 1 is a category E
  - 2 is an endofunctor (T: E → E, functor with same source and target)
  - 3 is the unit of adjunction $\eta: 1_X \rightarrow T$ (change, looking forward)
  - 4 is the multiplication $\mu: T \times T \rightarrow T$ (change, looking back)
- A monad is therefore <E, T, η, μ>

The Comonad

- The dual of the monad
- A comonad is a 4-cell <1,2,3,4>
  - 1 is a category E
  - 2 is an endofunctor (S: E → E, functor with same source and target, S is dual of T)
  - 3 is the counit of adjunction ε: S → 1_X (change, looking back)
  - 4 is the comultiplication δ: S → S X S (change, looking forward)
- A comonad is therefore <E, S, ε, δ> or <S, ε, δ>
- Both monad and comonad are often defined by a 3-cell descriptor with the category omitted (as implicit)

3-cell descriptors with adjoints

- The 3-cell monad < T, η, μ>
  - is written <GF, η, GεF> (last up a level for multiplication)
- The 3-cell comonad <S, ε, δ>
  - is written <FG, ε, FηG> (last up a level for comultiplication)
- The monad structure looks forward with F and η and backwards with G and GεF
- The comonad structure looks backwards with G and ε and forward with F and FηG

Terminology

- A monad is often simply addressed by its endofunctor.
  - So < T, η, μ> is called the monad T
- Similarly for the comonad
  - <S, ε, δ> is called the comonad S
- It's a synecdoche

Operating on a Topos

- The operation is simple:
  - T: E → E
    - where T is the monad <GF, η, GεF> in 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

Process in Musical Performance

- The topos E defined earlier contains
  - The physical notation in the category V (for Variant) for the music as conventionally laid out in sheet music (or otherwise!)
  - The performers in the category R (for Role) for the actual musical event
- A single monad/comonad action (of 3 cycles T^3) will take the music forward one unit of performance (phrase or bar), say one step

Process in Musical Performance 2

- Moving from one barline to another is determined uniquely by the adjunction F -| G
  - F is the free functor (looking forward, creative)
  - G is the underlying functor (looking back, enforcing the rules, qualia)

Process in Musical Performance 3

- If adjointness holds over the 3 cycles
  - Then η the unit of adjunction measures the creativity of the step going forward (dialectic)
  - And ε the counit of adjunction measures the qualia of the step looking back (rhetoric)
- If adjointness does not hold over the 3 cycles
  - Then integrity has been lost and resynchronization is necessary

Experience

- Performers do comment that playing is an intensive experience:
  - at the same time both looking back as to what you have played and anticipating what is to come.
- Such experience is captured by the monad/comonad structure with its forward/backward nature and inherent adjointness

Composition

- A musical work is referred to as a composition.
- It is indeed a composition of steps
  - With the output from one step becoming the input to the next step
- The order is fixed in advance
- Composition is an inherent feature of category theory
- With one monad execution as a single step, it is necessary to compose monads to perform a full work

Therefore composability is the Key

- Compose many monads together to give the power of adjointness over a whole wide-ranging application
- In banking with Bitcoin the reliability obtained from composing processes over a wide-range of machines (distributed data recovery) justifies the move to Category Theory

Blockchain 1

- The categorial monadic approach is being used for the Blockchain [Meredith], a transaction system, adopted by Bitcoin, for keeping hundreds or even thousands of copies of each transaction record, using multiple transaction logs.
- The monadic design pattern provides a broad range of transactional semantics with composition the key to scaling any system.

Blockchain 2

- The blockchain approach is drawing interest from the established banking industry, where a blockchain is viewed as a shared, encrypted `ledger' that cannot be manipulated, offering promise for secure transactions.
- Meredith indicates that compositionality is the key to reliability but offers few details on how this is achieved in the monad.

Monad Composition needs Care

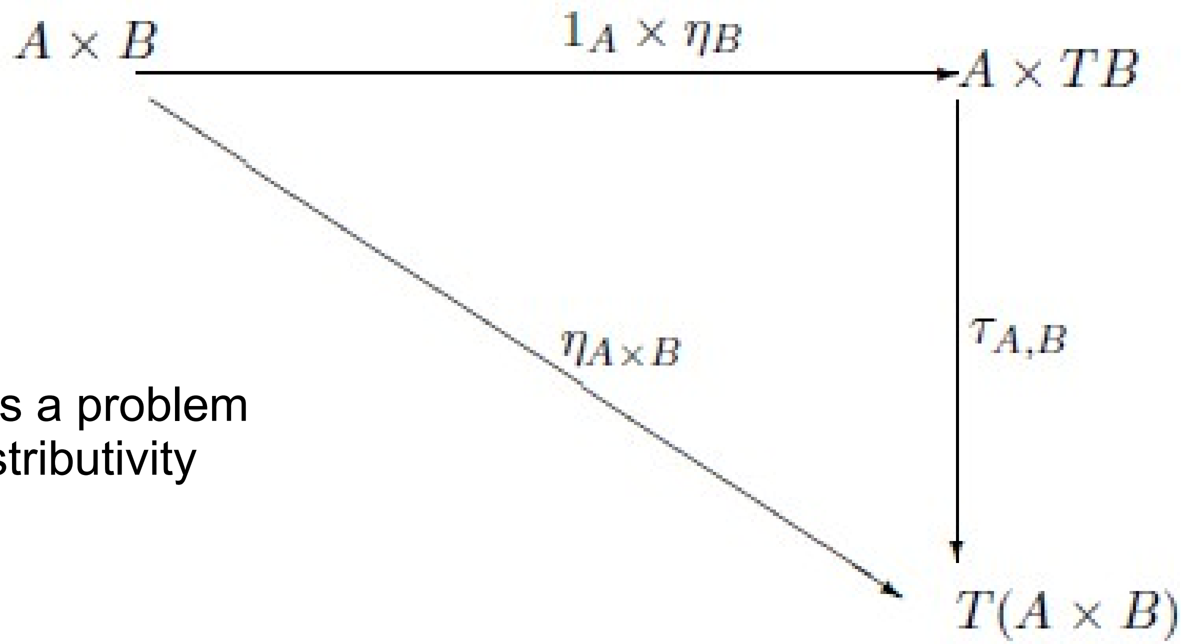
- There is a problem though in EML (Eilenberg/ Mac Lane) Category Theory:
  - Monads do not compose naturally

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 X TB → T (A X 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

Cartesian Monads in Music

- Take each barline, or some other time signature, as a unit of process
  - Such a barline will be Cartesian, representing the potentially complex physics of the music
    - Combinations of notes, including chords
- Therefore Cartesian Monads as strengthened by the Kleisli Lift are essential for composition purposes

Summary of Progress

- Topos has been established as data-type of choice
  - Design with pasted pullbacks and recursive pullbacks is being explored
  - Dolittle diagrams at bottom level provide intension/extension mapping
- Monad can process the topos
  - Readily as a single step
  - A Cartesian Monad requires the Kleisli lift for multiple composition
- Advent of Monads in Haskell gives an experimental test-bed

Look Forward

- Music application to be developed further
  - More contact with real musicians
  - Topos should be elaborated
    - As general as possible
    - Construction of Dolittle diagrams for intension/extension
  - Clarification of monad/comonad role
    - Describing process in more detail
    - Recognition of time jitter
    - Understanding of dialectic/rhetoric balance
- Knowledge gained to be fed into general advance in utilising category theory