Abstract Relations and Allegorical Categories

Nick Rossiter & Michael Heather
Department of Computer Science and Digital Technologies
Northumbria University, NE1 8ST, UK
nick.rossiter1@btinternet.com
http://www.nickrossiter.org/process/

Abstract

The World is concerned with relationships between entities whether living or inanimate. Representing relationships is therefore a key activity in the physical sciences, life sciences and the social sciences. These may all be modelled formally within information systems. Mathematically relations can be represented as a generalisation of the function in sets but there are a number of problems with such treatment: the fundamental basis of set theory is discrete elements rather than morphisms between sets. There is no inherent natural way in set theory of employing higher-level mappings as is often required for a full solution to real-world problems.

The topos within category theory with its emphasis on morphisms (instead of sets of elements) and with a multilevel architecture is a promising candidate as a structure for representing relations. At recent ANPA meetings the authors have explored the topos with its potential of the Cartesian closed category as the leading contender in meeting requirements for handling relations. Pure mathematics on the other hand has developed the use of the category REL as a categorification of the relation in set theory. However REL is not Cartesian closed which severely limits its usability. Recognising the limitations of REL Freyd and Scedrov, working in set based category theory have developed the 'category of allegories'.

The purpose of the current paper is to understand limitations in REL, to describe allegorical categories and to explore and evaluate their use in comparison with the topos approach in the context of information systems.

1 Introduction

The concept of relation underpins the Universe. It works coherently through the interactions between its particles. To model the Universe it is not surprising therefore that every branch of mathematics attempts to capture the relation as connections between one entity and another: from quantum entanglement through graphs to simplified set representations. Models that are suitable for implementation in information systems have recently found increased credibility, for example the Relational Data Model in computer science

It is usually not possible to separate relation from process. Process is the more fundamental concept as it covers the functionality and the inherent atomicity, consistency, isolation and durability, of any transaction, as well as the necessary, but limited, description in the relation of the static properties alone. In this sense a relation can be viewed as a static view of process as opposed to its dynamic aspects.

2 Process and Categories

Aristotle recognized that no two entities in the Universe are alike but exist at various levels of equivalence. In order to explore in detail the internal logical relations of this empirical phenomenon he introduced into his *Organon* the concept of type by coining the word 'category'. Its literal meaning was 'down at the market' and it had there evolved as a technical legal term. For the market place in Athens was where justice was administered in an early form of court proceedings. The word category was used to describe an indictment as a precise legal statement or charge. Exact descriptions are needed at law for there is often an adversary to challenge any statement with some rational argument. That need for high level of precision continues today into the scientific use of the word 'category'. Type identifies differences within equivalent concepts. Thus no two persons are absolutely alike (even so-called identical twins) but humans form a category with many different subcategories. No two computers are identical but form a category and so on for any real world entity.

There are two significant ontological features of categories: how they relate one to the other internally and how they relate one to the other externally. Two important movements of the twentieth century have advanced these. Category Theory as first conceived by Eilenberg and Saunders Mac Lane [19] provides a formal calculus of relationships in mathematics with three levels represented by arrows, functors and natural transformations. Thus a category of computers could represent the physical machinery as objects and their comparison by the arrows between the objects.

The other movement posits 'process' as Universal. The notion of process originally harks back to the pre-Socratic philosophers of ancient Greece but has more recently been revived by Bergson and others culminating in the version of Whitehead's Process & Reality [27]. Existence is 'becoming'. Thus Whitehead even interprets the equals signs in operations on Natural Numbers as 'becoming'. For example $2 \times 3 = 6$ means 'twice three becomes six' ([28] p 91 *et seq*). This recognises that the two sides of the equation are not the same.

3 The Predominance of Process

In order to understand the concept of process further consider the registration of students on a university course as an example of process. The process as a whole can be broken down into a number of stages:

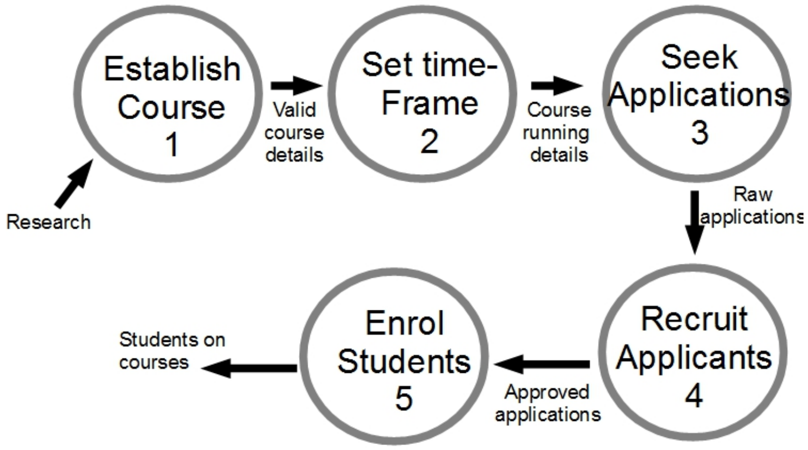

Figure 1: Activity Diagram for Student Registration System

1. Establishing the course with credentials and requirements
2. Setting a time frame for each delivery of the course
3. Seeking applicants to the course
4. Recruiting appropriate applicants
5. Enrolling students on the course

All these stages are dynamic, involving transformations from one state to another, as in Figure 1 where each stage is shown in a circle, together with its name and sequence number as in the enumerated list above. The arrows indicate a flow of data between each stage. Such data might be:

- $1 \longrightarrow 2$ valid course details
- $2 \longrightarrow 3$ course running details
- $3 \longrightarrow 4$ raw applications
- $4 \longrightarrow 5$ approved applications

These flows are also shown in Figure 1. It should be noted that initial and terminal flows of data are needed, the former is represented by *Research* to start the sequence of process stages, and the latter by *Students on courses* to indicate the eventual outputs from the stages. Such a presentation is popular from the perspective of systems analysis and design perspective, where it is termed an activity diagram in modelling languages such as the Universal Modelling Language UML [8]. For converting such diagrams into category theory representations we need precisely to distinguish each action by identifying its morphism, with its source and target. We abbreviate the concepts to avoid obscuring the mathematics. The morphisms are:

- $est : R \longrightarrow V$ (est is establishing the course, R is research, V is valid course details)
- $set : V \longrightarrow C$ (set is setting the time-frame, C is course running details)
- $seek : C \longrightarrow RAW$ (seek is seeking applicants to the course, RAW is raw applications)
- $rec : RAW \longrightarrow A$ (rec is recruiting suitable applicants, A is approved applications)
- $enr : A \longrightarrow S$ (enr is enrolling students on the course, S is students on courses)

As the output of one process is the input for another, we can compose the morphisms to form a further arrow $course\_make$:

$$course\_activity = enr \circ rec \circ seek \circ set \circ est$$

where $\circ$ gives a product with an ordering according to the convention explained below.

The result can then be viewed as the category **Course_Create** shown in Figure 2 with the five morphisms $enr, rec, seek, set, est$ and their composition $course\_activity$. For **Course_Create** to be a category, there will be identity arrows for each object, for example $1_V : V \longrightarrow V$, and identity arrows are composable with the arrows forming the process.

As it stands **Course_Create** is very limited, lacking any rules and constraints, which are part of the operational environment. Indeed **Course_Create** is really an example of categorification where a simple translation has been made between systems analysis

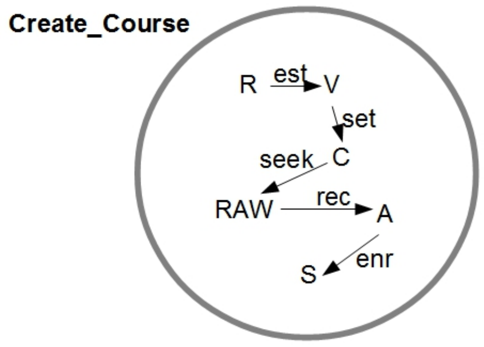

Figure 2: The Category Create_Course for the Student Registration System

properties and categorial ones. Besides the typing of the individual arrows and objects, the process should also satisfy the basic properties of a transaction, often summarised as ACID (Atomicity, Consistency, Isolation, Durability). One way to exploit the power of category theory is to raise the levels of the application so that, within a category, objects become categories in their own right and the arrows between the objects become functors. Raising the levels for the application above yields the categorial structure shown in Figure 3.

$$\begin{tikzcd}\mathbf{R} \arrow[r, "EST"] & \mathbf{V} \arrow[r, "SET"] & \mathbf{C} \arrow[r, "SEEK"] & \mathbf{RAW} \arrow[r, "REC"] & \mathbf{A} \arrow[r, "ENR"] & \mathbf{S}\end{tikzcd}$$

Figure 3: Student Registration System as categories with functors

In Figure 3 **R**, **V**, **C**, **RAW**, **A**, **S** are categories and EST, SET, $SEEK$, REC, ENR are functors, mapping again from left to right. One of the powers of the functorial level is that we can define inverse functors, mapping from right to left, representing the mapping in the opposite direction. These are defined as:

- $CHEC : \mathbf{V} \longrightarrow \mathbf{R}$ (checking course details against the research)
- $VER : \mathbf{C} \longrightarrow \mathbf{V}$ (verifying course running details against the course details)
- $VAL : \mathbf{RAW} \longrightarrow \mathbf{C}$ (validating raw applications against the course running details)

- $ENS : \mathbf{A} \longrightarrow \mathbf{RAW}$ (ensuring application has been made)
- $QUAL : \mathbf{S} \longrightarrow \mathbf{A}$ (ensuring student is qualified)

$$\begin{tikzcd}
\mathbf{R} \arrow[r, shift left, "EST"] \arrow[r, leftarrow, shift right, "CHEC"'] & \mathbf{V} \arrow[r, shift left, "SET"] \arrow[r, leftarrow, shift right, "VER"'] & \mathbf{C} \arrow[r, shift left, "SEEK"] \arrow[r, leftarrow, shift right, "VAL"'] & \mathbf{RAW} \arrow[r, shift left, "REC"] \arrow[r, leftarrow, shift right, "ENS"'] & \mathbf{A} \arrow[r, shift left, "ENR"] \arrow[r, leftarrow, shift right, "QUAL"'] & \mathbf{S}
\end{tikzcd}$$

Figure 4: Student Registration System as categories with dual functors

The pairs of arrows are shown in Figure 4. In the right-hand direction, we have creativity through the free functors and, in the left-hand-direction, we have a series of validation checks ensuring consistency through the underlying functors. This representation is very powerful in handling transactions. If the dual pairs of composed functors are adjoint then the structure in Figure 4 satisfies the ACID requirements [13], through the monad (T^3) and comonad (S^3) constructions, where F is the composition of free functors, G is the composition of underlying functors, T^3 is the cycle GF repeated three times and S^3 is the cycle FG repeated three times:

$$F : ENR \circ REC \circ SEEK \circ SET \circ EST$$

$$G : CHEC \circ VER \circ VAL \circ ENS \circ QUAL$$

$$T^3 : GF \circ GF \circ GF$$

$$S^3 : FG \circ FG \circ FG$$

The monad and comonad structures can be applied to a topos, holding the structure of the data, to define the transaction design for process handling. The category of algebras over a monad is traditionally called its Eilenberg-Moore category ([19] at pp. 139-142). Dually, the Eilenberg-Moore category of a comonad is its category of coalgebras. The subcategory of free algebras is traditionally called the Kleisli category of the monad, as is its dual the subcategory of co-free co-algebras of the comonad ([19] at pp. 147-148). Further work is planned to expound on implementation from the free algebra perspective.

The categorial monadic approach is being used for the Blockchain [21], 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. 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 [23].

What is apparent from the development of this transaction example is that the application is predominantly dynamic based on process, with the static data structures as data relations, a relatively small part of the whole. Yet the static side cannot be ignored as it is through the relations that data is stored with persistence to enable organizations to function over time. We now look at the static side to examine the choice of the theoretical structures available.

4 Theories for Relations

In set theoretic terms a *Relation* is for example the data structure sMt where s, t are sets and M is a relationship between the sets. In marriage s is male partner, t is female partner and M is the marriage between s and t. It is important to note that even for this simple example the relation is a surrogate for a process, the act of marriage [1].

In general set theoretic terms, the form for a relation R is sRt where s, t are sets and R the mapping between them. There may be more than one relation between the same two sets; for example sUt is another relation U between the two sets s, t.

R has various interpretations, being expressed formaically as either:

1. $R \subseteq S \times T$ (subset of product)
2. $R \in \{< S, T >\}$ (member of ordered pairs)
3. $R = \oplus < S \times T >$ (disjoint union of product members)

In case 1 the relation is expressed as the subset of the cartesian product of the two sets. This is a frequently used representation, particularly in relational databases. Case 2 expresses the relation as being in the collection of ordered pairs. Case 3 is interesting, leading more into the categorial representation, with R as a disjoint union of the ordered pairs.

In terms of Category Theory on the other hand the Category **Rel** is either :

1. $Rel : Set \longrightarrow Set$. That is a functor mapping between sets where $Rel \subseteq (Set \times Set)$ or

[1]This example shows the limitation of the fixed structure of sets as it cannot easily cope with an extension to same-sex marriages.

2. *Rel* : $\oplus(A \times B)$. That is a disjoint union category where Rel is a co-product diagram over objects A, B

The coproduct diagram over objects A, B is shown in Figure 5 where $A \cup B$ is the disjoint union of A and B, i, j are inclusions, $(f, g) : A \cup B \longrightarrow Q$ is a unique morphism such that the diagram commutes with $(f, g) \circ i = f$ and $(f, g) \circ j = g$. Q is the quotient with $< (f, g) >$ as the coequaliser.

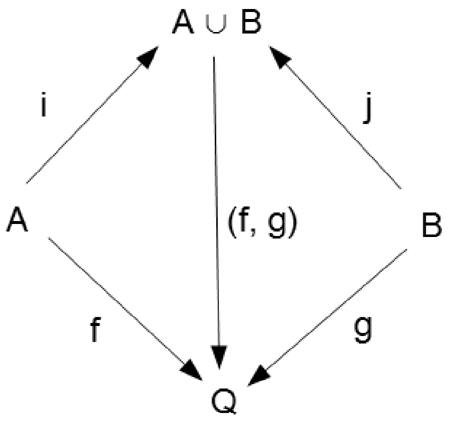
$$\begin{tikzcd} & A \cup B \arrow[dd, "(f, g)"] & \\ A \arrow[ur, "i"] \arrow[dr, "f"'] & & B \arrow[ul, "j"'] \arrow[dl, "g"] \\ & Q & \end{tikzcd}$$

Figure 5: The category **Rel** with $A \cup B$ as relation over objects A, B

5 The Cartesian Closed Category

Both the set theoretic *Relation* and the category **Rel** need to be contrasted with the vital properties of the Cartesian Closed Category (CCC):

1. as cartesian the CCC provides for products which is the basis for relationships
2. the CCC has closure at the top with the terminal object
3. the CCC has a natural inherent internal intuitionistic logic by way of adjointness of the logic quantifiers with $\exists \dashv \Delta \dashv \forall$
4. the terminal object of a CCC can perform the role of an identity functor
5. categories and objects are interchangeable in the CCC

6. the CCC has exponentiation which provides for connectivity and evaluation with the operation *eval*
7. the CCC is unconstrained by the axioms of sets or natural numbers in process metaphysics
8. the CCC is implementable

The last three properties are particularly relevant for process studies. Sets and numbers require independence of fundamental elements which offend against Whitehead's principle of non-separability. That is also the sense here of *exponentiation* where there is no need for elaborate constructions like Cantor's sets. Implementation is an intriguing concept for the Universe is an implementation of a CCC but no example of a human implementation has been reported for it has to satisfy both the principles of Einstein's Theory of Relativity and those of Quantum Theory. A true quantum computer would be such an implementation but the difficulties in achieving this by finite means may well be associated with the difficulties we are discussing here. The CCC may be modelled on a digital computer based on the von Neuman architecture including advanced versions such as λ-calculus machines but it must be emphasised that these are only models and therefore only reliable to first order.

So it is an important question as to whether **Rel** is a CCC. **Rel** fails to be a CCC whichever definition is accepted. There is no terminal object for if **Rel** is taken as $Set \longrightarrow Set$ then both the start and end objects are the same. There is no product if the basis for **Rel** is taken as a coproduct then**Rel** is not a CCC and, in our view, does not therefore provide a viable construction for relationships or process. **Rel** is indeed categorification as a direct translation from the set theoretic concept.

6 Theory for Allegories

The set theoretic concept of relation and its categorification is inadequate as a basis for representing relations in category theory. Is there a better way forward?

Allegories were proposed by Freyd & Scedrov 1990 as a more categorial way forward: "Allegories are to binary relationships between sets as categories are to functions between sets." ([11] p.195, section 2.1). They claim that the category of allegories is decidable but not axiomatisable and defined it as a category with the properties listed in Panel 1 from ([11], section 2.1, p.195). Operations on an Allegory are given in Panel 2 from the same source of Freyd & Scedrov.

Allegories are not inherently Cartesian closed categories: their structure and operations does not automatically confer them with a terminal object for instance. However,

Panel 1 : The Definition of the Category of Allegories

- An allegory is a category with unary operation R^0 and binary product operation $R \cap S$
  - R^0, reciprocation, where for $R : X \longrightarrow Y : xR^0y$ iff yRx
  - $R \cap S$, intersection, where for $R, S : X \longrightarrow Y : xR \cap Sy$ iff xRy and xSy
- Introducing a third relation $T : X \longrightarrow Y$
  - The main axiom is the modular law: $(RS \cap T) \subset (R \cap TS^o)S$ [25]
- Intersections are idempotent, commutative, associative
- Composing intersections composes relations

the intersection can be defined in terms of products and the concept has emerged of an allegory requiring an *Underlying Regular Category* as good practice. Such a categorical basis provides the facilities of the classical relational calculus. There are two conditions for a category to be regular:

(a)
$$\begin{tikzcd} Z \arrow[r, "\pi_l"] \arrow[d, "\pi_r"'] & X \arrow[d, "f"] \\ X \arrow[r, "f"'] & Y \end{tikzcd}$$

(b)
$$\begin{tikzcd} W \arrow[r] \arrow[d, "g"'] & X \arrow[d, "f"] \\ Z \arrow[r] & Y \end{tikzcd}$$

Figure 6: Regular Category: (a) Co-equaliser Diagram; (b) Epimorphism Diagram

- if the diagram in Figure 6(a) is a pullback and there is a co-equaliser for π_l and π_r, that is $f : X \longrightarrow Y$

Panel 2 : Operations on the Category of Allegories

- 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

Panel 3 : Properties of the Underlying Regular Category

- A regular category is Cartesian closed (CCC), typically a pullback with some 'nice' properties giving stable factorization:
  - regular epimorphisms (onto, all objects in coproduct assigned)
  - coequalisers (pairs of parallel arrows converge onto one arrow in coproduct)

- if the diagram in Figure 6(b) is a pullback and f (and hence g) are regular epimorphisms; a regular epimorphism is an epimorphism (surjection) which coequalises some parallel pair of morphisms, in this case π_l and π_r in Figure 6(a).

The two diagrams in Figure 6 can be combined into a single diagram as in Figure 7 to show the regular category **C**. This category has two products: 1) W which is the product $X \times Z$ in the context of Y, that is $X \times_Y Z$; 2) Z which is the product $X \times X$ in the context of Y, that is $X \times_Y X$. The regular category is therefore the relationship of W and Z in the context of Y, or a factorisation of W and Z through Y. The coequaliser and epimorphism requirements ensure that all maps in the category terminate with $f : X \longrightarrow Y$ and that all Y participate in the factorisation. Y is the coproduct of the pullback. The requirements for a regular category are summarised in Panel 3.

$$\begin{tikzcd}
 & W \arrow[dl, "g(\pi'_r)"'] \arrow[dr, "\pi'_l"] & \\
Z \arrow[rr, "\pi_l"] \arrow[d, "\pi_r"'] \arrow[drr] & & X \arrow[d, "f"] \\
X \arrow[rr, "f"'] & & Y
\end{tikzcd}$$

Figure 7: Regular Category **C**: the two Pullback Diagrams in Figures 6(a) and 6(b) combined

6.1 The Table Category

Perhaps reflecting their close association with first-order logic, allegories do have a tabulation view, bearing some correspondence to the relational database model which, although set-based, is defined popularly in terms of tables. There is a hint of categorification here with the allegorical table being a translation of the relational data model. The Table category is defined in Panel 4.

Freyd acknowledges the link between the table construction and relational intension/extension mappings as in relational databases: "The usual extensional notion of relations on sets coincides with the categorical notion as applied to this case of a table" ([11] section 1.415 p.39). Besides regular categories, hyperdoctrines and tables, other constructions such as the bicategory can be used in an allegory. While this may offer increased flexibility, it nevertheless reduces the cohesion of the approach.

6.2 The Implementation of Allegories

Allegories have received a rather uncertain reception since their original detailed introduction in 1990 by Freyd & Scedrov's *Categories, Allegories* [11]. The allegorical concept has not been developed further in any significant way by Freyd or other workers in pure mathematics though it is discussed by Johnstone in *Sketches of an Elephant* **1** *section A3.2* [16] under *Allegories and Tabulations*. Bob Walters [25] considered that the modular law of an allegory was bizarre and that the objectives of an allegory would be better achieved with the bicategory concept.

In reviewing the literature on the application of allegories since the work of Freyd and

Panel 4 : The Table Category

- For a table with name T and n columns
  - $T : x_1, x_2, x_3, x_4, \ldots$ with column names $x_i (1 \leq i \leq n)$
- The feet of the table are $FEET : A_1, A_2, A_3, A_4, \ldots$
  - with values for a particular column $A_i (1 \leq i \leq n)$
- Each column of a table is addressed through the mapping: $x_i : T \longrightarrow FEET$
- Mapping between tables, including the universal table, gives closure.
  - Natural closure between two tables is represented by $\Theta : T \approx T'$
  - Θ is a relation $REL(A_1, A_2, A_3, A_4, \ldots)$

Scedrov, it appears that the take-up of the idea has been limited to allegories selected for the development of *typed* first-order logic, particularly of Prolog. The theory of logic has been dominant with the use of tabular allegories explored by Arias *et al* [1, 2] in first-order unification and logic programming with Prolog. Arias completed his Doctoral programme and published his thesis in 2012 *Relational and Allegorical Semantics for Constraint Logic Programming* [3]. His work had employed Σ-Allegories as the basis for the Prolog-style constraints; such allegories are described as pretabular. An interesting comment at p.138 is on the choice of Σ-Allegories:

> Instead of requiring tabular distributive allegories for the semantics of CLP programs, we built the weakest categorical structure needed for an adequacy theorem. The partial-tabular nature of Σ-allegories constitutes an interesting notion from the logical point of view and is an example of 'just the required structure' approach'.

However, much of the further work in Chapter 10 *Extensions to Logic Programming in Tabular Allegories* at pp. 127-136, deals with more ambitious logic objectives such as the use of tabular allegories themselves and monads. This may suggest that the allegory logic had been difficult to implement in the time available.

Finkelstein *et al* [9] consider logic programming with Prolog in tau categories. One of the co-authors of this paper was Peter Freyd himself. Brown & Hutton [5] apply first-order relational algebra to circuits designed as pretabular allegories. Palmgren & Vickers [22] mention allegories in the context of Horn logic but do not actually employ Freyd's ideas. Hermida & Jacobs [14] consider the algebraic compactness of Freyd's structures without actually applying allegories.

Looking at current research, since 2014, work has been continued by Arias *et al* on constraint logic programming [4]. The proposals by Zieliński *et al* [29] strike a resonance with the authors of this current paper:

> Because of the deep relationship between relational databases and relational algebra and also a lively research program of using category theory for conceptual modeling ..., it is surprising that allegories were hardly ever used for database modeling. The use of allegories for this purpose was suggested in [9], and recently, the new allegorical data model was introduced by the authors ...; barring that, the authors were unable to secure more references (at pp.260-261).

The reference '[9]' quoted is to Diskin [7]. The theoretical allegorical model developed by Zieliński *et al* [29] also extends the standard relational model to include fuzzy and modal logic. In a subsequent paper Zieliński *et al* [30] demonstrate that their earlier results for binary relations also apply to n-ary relations with appropriate construction of products. Finally Maietti & Rosolin [20] develop their theory of existential elementary doctrines with extensive use of allegories.

There have been a number of more informal comments about Allegories, which do give some insight into how the concept is viewed. From Wikipedia a view consistent with the aims of Freyd: "the theory of allegories is a generalization of relation algebra to relations between different sort". From ncatlab a comment on the aspirations of Freyd: "an allegory is a category with properties meant to reflect properties that hold in a category Rel of relations". From a computing practitioner viewpoint in *Hacker News*: "Freyd and Scedrov's work on Allegories (replace functions in categories with relations) would be more suitable for relational databases". A mathematics blog (Maths Stack Exchange) comments: "With the definition of category, it is easy to have an idea of what is a category, but with allegories I'm totally lost".

The above suggests that allegories have been used mainly for relational systems with first order logic, particularly Prolog and that the take-up of the concept is far from spectacular and is not increasing at any rate. Maybe the concept has not been found to be readily comprehensible.

6.3 Advantages and Disadvantages of Allegories

An advantage of the allegory approach to relations is that it is more in the spirit of category theory than the set approach, because of its arrow basis. Further as the preferred basis is the regular category, for example a pullback, the allegory is cartesian closed including a terminal object and identity (unital property). Unlike with the category *Rel* the allegory approach is not categorification. The allegory has the internal logic of first order relational calculus, making it ideal for formalising relational databases, which hold more than 90% of commercial data, and logic languages such as Prolog. The tabular view of an allegory is very close to the relational database design. There is potential for greater capability in the allegory approach with hyperdoctrine and bicategory views but the extensions to power allegories and division allegories have still to be shown to be useful in the application context. Indeed category theory has properties on the wish list of Codd, the relational database originator ([6] at p477): "The major problem in the entity-relationship approach is that one person's entity is another person's relationship. There is no general precisely defined distinction between these two concepts ..." Arrows and objects in category theory are interchangeable but in a closely defined manner.

A major disadvantages of the allegory approach is the closed world assumption with its inherent Boolean logic, although some extensions to the basic concept such as division allegories are claimed to be Heyting. The logic is first order, not higher order, and with the lack of naturality, allegories do not form a basis for metaphysics or for the new generation of object databases. Although the number of views add diversity, they also reduce the cohesion of the approach and may lead to inconsistencies in properties such as unitality.

The choice by Freyd of the term *allegory* for describing the relational concept is interesting. In literature an allegory is usually a transformation. This appears to be the motive of Freyd for his choice of language as set-based relational structures are transformed into allegorical structures including the underlying regular category.

Beyond first order it is always necessary to be mindful of Gödel uncertainty namely that any system relying on axioms and sets is undecidable and any open system is logically incomplete. In pure mathematics this uncertainty is often just ignored but in work on real world systems any such omission may be dangerous. Freyd and Scedrov [11] were clearly aware of the difficulties and declare (at p 195) that the entire equational theory on the operations of allegories "is decidable but not finitely axiomatizable [2.158]." However there is nothing about this in their paragraph 2.158 and it can be speculated that the reference to 2.158 may well be a misprint for paragraph 2.438, which relies on the 'hypothesized existence of a simulated consistency proof'. That may be mathematically sound but is of no use for applications in physics or in computer science. There is some treatment of these issues of uncertainty by Freyd and Scedrov elsewhere in their book in paragraph [1.182] within the context of logoi and in the alternative formulation of the work in their Appendix

B but there is no rigorous or comprehensive treatment of uncertainty in connection with allegories which should therefore carry a Gödel health warning.

6.4 Prospects for Allegories

As asserted in the quote above by Zieliński *et al* [29], it is very surprising that workers in relational data modelling with categories have ignored allegories. For example Johnson and Rosebrugh have done much elaborate work on entity-relationship database models and relational databases with sketches; see for example [15]. It would have seemed simpler to use 'off-the-shelf' allegories for the categorial modelling, adding a graphical interface to the allegory-based system to complete the work.

Allegories will not form part of our work going forward on natural information systems because of the disadvantages outlined above. However, allegories could still be significant for another aspect of our work on interoperability: the ability of heterogeneous information systems to work together are a major problem for information systems today. Allegories, based on regular categories, are closer to our preferred structure, the topos, than other approaches to relational systems such as the sketch. For instance the topos and the allegory have the same underlying structure of the pullback, both can be viewed as regular categories and both have an internal logic. This commonality provides much potential for interoperability. With a relational database as an allegorical category **A** and a natural database as a topos **T**, we can compare the two categories through the adjointness: $F : \mathbf{A} \longrightarrow \mathbf{T}; G : \mathbf{T} \longrightarrow \mathbf{A}; F \dashv G$.

To summarise: the allegory seems to have unrealised potential for the construction of first-order relational systems as categories enabling the full power of category theory to be applied, such as with functors, natural transformations and adjoints.

7 The Topos Revisited

Aristotle not only introduced the concept of the category but also used the term *topos* as a metonym for a metaphysical space that gives rise to different aspects of reality. Alexander Grothendieck needing an abstract space to embed Riemann's algebraic geometry picked up on the term and formally defined the topos within axiomatic category theory as an abstract of topology. As the later is but an enriched theory of open sets the Grothendieck topos can be variously defined according to the assumptions applied in the enrichment. Freyd & Scedrov seem to prefer the topological theoretic definition for topos from Grothendieck's student Giraud using sheaf theory. This is inadequate for process studies. Although it was Whitehead who with his student Russell wrote the definitive treatise on sets [26], nevertheless both authors later repudiated the work. Whitehead even extended his denunciation very widely:

We can misconceive the very meaning of number and the interconnections of number. The great mathematicians of the seventeenth and eighteenth centuries misconceived the subject matter of their studies. For example, in respect to the notions of infinitesimals of the necessary precautions in the use of infinite series and the doctrine of complex numbers, their discoveries were suffused with error ([28] at p68).

But Whitehead has initiated a revolution which has yet to take off:

I revolt against this concentration upon the multiplication table and the regular solids: in other words against the notion that topology, based upon numerical relations, contains in itself the one fundamental key to the understanding of the nature of things.

The regular solids are fundamental to Euclidean geometry but to illustrate allegories Freyd & Scedrov digress ([11] p205 *et seq*) beyond Euclid into Desargues' theorem of projective geometry with its invariance under projective transformations and a duality between points and lines. It is to be noted that projective geometry gives rise to a non-metric space which therefore transcends measurement. This is to be expected in higher order space and it is a characteristic of the topos that it has no inherent natural number system or a Euclidean geometric structure. For Whitehead has demonstrated that process operates top down. The category theory of Freyd & Scedrov is bottom up and relies on the axioms of set theory to ensure the uniqueness 'up to natural isomorphism' that defines mathematical naturality. Process on the other hand provides the uniqueness 'down from natural isomorphism' that defines physical naturality. That is the simple uniqueness earlier observed that no two entities in the World are identical and in this context of relationships gives rise to Whitehead's empirical fact of non-separability as an alternative to a Euclidean structure. The latter has proved a very useful first order model for space but for work on higher order relationships the process topos is more natural.

8 The Topos: further work identified

In earlier work we established the topos as the categorial structure of choice for handling information systems. We developed a table, Table 5, in [24], which compared capability of an approach with the facilities required. This showed that the Cartesian Closed Category approach, described earlier in Section 5, matches the relational model in all respects and additionally handles the interoperability requirement. The topos approach improves further on CCC approach by providing additional features for searching and by facilitating query closure. Indeed the topos approach alone handles all of the listed requirements from

Table 5 cited above. That earlier work demonstrated that process is the pervasive theme of an information system and the examples here in Section 3 reinforce how crucial is the handling of activity in developing an effective information system, even where a superficial glance might suggest that static data structures suffice.

$$\begin{tikzcd}
 & \mathbf{S} \arrow[dl, shift right, "\pi_l^*"'] \arrow[dr, shift left, "\imath"] & \\
\mathbf{S} \times_{\mathbf{R}} \mathbf{M} \arrow[ur, shift right, "\pi_l"'] \arrow[dr, shift left, "\pi_r"] \arrow[rr, shift left=2, "\exists"] \arrow[rr, shift right=2, "\forall"'] & & \mathbf{R} \arrow[ul, shift left, "\imath^{-1}"] \arrow[ll, "\Delta"'] \arrow[dl, shift left, "\jmath^{-1}"] \\
 & \mathbf{M} \arrow[ul, shift left, "\pi_r^*"] \arrow[ur, shift left, "\jmath"] &
\end{tikzcd}$$

Figure 8: Single Pullback: $\mathbf{S} \times_{\mathbf{R}} \mathbf{M}$; $\mathbf{S}$ student, $\mathbf{M}$ module, $\mathbf{R}$ result

In concluding [24] a number of areas were identified for further work. On the data process side, it was thought that queries, including the use of the subobject classifier and power objects, should be better illustrated. Examples of intuitionistic logic, the Heyting internal logic for the topos, should also be developed. On the database design side, areas for study included the cocartesian dual of the topos, which may aid normalisation, the pasting of pullbacks to handle more complex (and realistic) examples, recursive pullbacks for nested structures and allegories, for handling relations.

We have made progress in a number of areas, all of which strengthen the case for the topos as the preferred structure for information systems. Allegories have been explored in depth, as described in Section 6, with their potential suggested for handling the relational part of a broader-based system. With a more complex data description, the pasting of pullbacks has now been explored. The use of the subobject classifier with both Boolean- and Heyting-type queries has also now been demonstrated in more detailed examples.

8.1 More Complex Pullback Structures

Our paper [24] dealt with a single pullback as a topos, as here in Figure 8, where the diagram is $\mathbf{S} \times_{\mathbf{R}} \mathbf{M}$, the product of $\mathbf{S}$ and $\mathbf{R}$ in the context of $\mathbf{M}$, where $\mathbf{S}$ is a student category, $\mathbf{M}$ is a module category and $\mathbf{R}$ is a result category. The operators $\exists$ and $\forall$ have the standard meanings of the existential and universal quantifiers respectively. Δ is the diagonal operator, right adjoint to $\exists$ and left adjoint to $\forall$: $\exists \dashv \Delta \dashv \forall$. Other arrows in the pullback diagram provide projection π and inclusion $\imath, \jmath$ relationships. Figure 8, with the adjoints, is a hyperdoctrine, as defined by Lawvere [17].

$$\begin{tikzcd}
(((S\ X_R\ M)\ X_M\ D)\ X_D\ U) \arrow[rr, "\pi_r"] \arrow[d, "\pi_l"'] \arrow[drr, phantom, "P3"] & & U \arrow[d] \\
((S\ X_R\ M)\ X_M\ D) \arrow[rr, "\pi_r'"] \arrow[d, "\pi_l'"'] \arrow[drr, phantom, "P2"] & & D \arrow[d] \\
(S\ X_R\ M) \arrow[rr, "\pi_r''"] \arrow[d, "\pi_l''"'] \arrow[drr, phantom, "P1"] & & M \arrow[d] \\
S \arrow[rr] & & R
\end{tikzcd}$$

Figure 9: Pasted Pullback Category: $(((\mathbf{S} \times_{\mathbf{R}} \mathbf{M}) \times_{\mathbf{M}} \mathbf{D}) \times_{\mathbf{D}} \mathbf{U})$; $\mathbf{U}$ university, $\mathbf{D}$ department, $\mathbf{S}$ student, $\mathbf{M}$ module, $\mathbf{R}$ result; $P1, P2, P3$ are labels for diagrams for purpose of discussion

Realistic examples will involve more categories. For instance we can extend the Student-Module example to include categories for Departments $\mathbf{D}$, Universities $\mathbf{U}$ and Lecturers $\mathbf{L}$. Pullbacks are pasted together following laws of composition on paths. In general given a category $\mathbf{C}$, a pasting diagram in $\mathbf{C}$ is a sequence of composable morphisms in $\mathbf{C}$. In the pullback context the composable morphisms are the individual pullbacks within an outer pullback structure. The pastings therefore follow the normal rules of composition, within a category, and are not intersections of paths, which would be set-based.

Figure 9 shows the pullback category $(((\mathbf{S} \times_{\mathbf{R}} \mathbf{M}) \times_{\mathbf{M}} \mathbf{D}) \times_{\mathbf{D}} \mathbf{U})$ where the bottom square vertically P1 is the pullback diagram $\mathbf{S} \times_{\mathbf{R}} \mathbf{M}$ (relation between student and module in the context of result) as in Figure 8. The middle square vertically P2 is the pullback diagram $((\mathbf{S} \times_{\mathbf{R}} \mathbf{M}) \times_{\mathbf{M}} \mathbf{D})$ (relation between pullback diagram P1 and department in the context of module) and the top square vertically P3 is the pullback diagram $(((\mathbf{S} \times_{\mathbf{R}} \mathbf{M}) \times_{\mathbf{M}} \mathbf{D}) \times_{\mathbf{D}} \mathbf{U})$ (relation between pullback diagram P2 and university in the context of department).

The pastings indicate a number of compositions: the path $\mathbf{U} \longrightarrow \mathbf{D} \longrightarrow \mathbf{M} \longrightarrow \mathbf{R}$ shows a hierarchical path from university to result, giving successive right projections from the product. The path of left projections: $\pi_l'' \circ \pi_l' \circ \pi_l$, composes the left projections from the products. There are also three further pullback diagrams: the combination of P1 and P2, the combination of P2 and P3, and the outer diagram, the combination of P1, P2 and P3, giving a total of six pullback diagrams in the category if naturality is

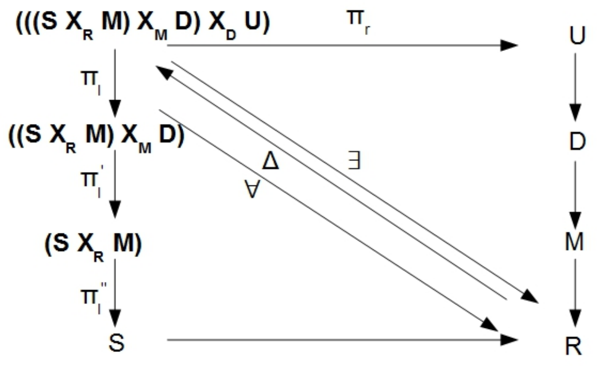

Figure 10: Pasted Pullback Category $(((\mathbf{S} \times_{\mathbf{R}} \mathbf{M}) \times_{\mathbf{M}} \mathbf{D}) \times_{\mathbf{D}} \mathbf{U})$; **U** university, **D** department, **S** student, **M** module, **R** result; intuitionistic logic $\exists \dashv \Delta \dashv \forall$ for the outer pullback diagram

satisfied. There are rules for handling cases where only some of the diagrams compose; however, it is probably safer for the integrity of information systems to ensure that all inner and outer diagrams do commute.

Figure 10 shows the intuitionistic logic for the pullback category of Figure 9 with the adjoints $\exists \dashv \Delta \dashv \forall$ for the whole (outer) pullback. Analogous diagrams can be drawn for the other commuting squares in the pullback category. The whole category forms a topos, which can be queried with Heyting logic as described later in Subsection 8.2.

We have also experimented with more complex structures such as the network structure where the category **L** for lecturers is introduced with the application rule that a lecturer may be assigned to more than one department. Figure 11 shows the pasting of the square P4, representing the lecturer-department relationship. This destroys the category as a pullback with for instance diagrams P2 and P4 not having the appropriate composition.

The solution to this lack of naturality is to introduce $\mathbf{D} \times \mathbf{L}$ into the hierarchy on the right-hand side between **D** and **M** as shown in Figure 12. This enables the relationship between department and lecturer to be represented in the pullback diagram of P2 as the product of $\mathbf{S} \times_{\mathbf{R}} \mathbf{M}$ and $\mathbf{D} \times \mathbf{L}$ in the context of **M**. With all diagrams commuting we now have:

- four individual pullback diagrams P1, P2, P3, P4
- three paired pullback diagrams P1, P2; P2, P3; P3,P4
- two triple pullback diagrams P1, P2, P3; P2, P3, P4
- the outer pullback diagram P1, P2, P3, P4

$$\begin{tikzcd}
(((S\ X_R\ M)\ X_M\ D)\ X_D\ U) \arrow[r, "\pi_r"] \arrow[d, "\pi_l"'] \arrow[dr, phantom, "P3"] & U \arrow[d] & \\
((S\ X_R\ M)\ X_M\ D) \arrow[r, "\pi_r'"] \arrow[d, "\pi_l'"'] \arrow[dr, phantom, "P2"] & D \arrow[d] \arrow[dr, phantom, "P4"] & (D\ X_m\ L) \arrow[l, "\pi_l''"'] \arrow[d, "\pi_r'''"] \\
(S\ X_R\ M) \arrow[r, "\pi_r''"] \arrow[d, "\pi_l''"'] \arrow[dr, phantom, "P1"] & M \arrow[d] & L \arrow[l] \\
S \arrow[r] & R &
\end{tikzcd}$$

Figure 11: Pasted Construction $(((\mathbf{S} \times_{\mathbf{R}} \mathbf{M}) \times_{\mathbf{M}} \mathbf{D}) \times_{\mathbf{D}} \mathbf{U})$; **U** university, **D** department, **L** lecturer, **S** student, **M** module, **R** result; $P1, P2, P3, P4$ are labels for diagrams for purpose of discussion; not a valid pullback category as not natural

giving a total of ten commuting pullback diagrams. Figure 12 is valid design as it is natural and 11 is invalid design as it is not natural. Valid design is therefore focused on pasting together pullbacks in such a way that the inner and outer pullbacks all commute. The significance of this is quite fundamental for the design of information systems. Normalisation techniques are seen as an effort to capture naturality in the context of artificial set-based systems. Such techniques are redundant in the design of topos structures such as the pullbacks dealt with here, where naturality alone is the design criteria. The naturality is linked to the metaphysics, which is the ultimate determinant of the relationships. Normalisation is consequently regarded as a 'hangover' from traditional design methods with no relevance to topos-based system design.

8.2 Subobject Classifier

The subobject classifier plays an important role in the facilities of a topos, providing a query language and the return of the result as a subtopos of known type, giving query closure. However, the use of the subobject classifier in an applied sense is not straight-forward as there are a number of important issues which are glossed over by pure mathematicians. For instance in the Universe there is a single intension, the one-object structure, which defines the extension, the many instances, in a preorder. In information systems the concepts of intension and extension are also paramount. But in the topos of pure mathematics, the concepts are not clearly separated and indeed their distinction is blurred so that it is not an easy task to transfer the results from pure mathematics to the applied

$$\begin{tikzcd}
(((S\ X_R\ M)\ X_M\ (D\ X\ L))\ X_{D\,X\,L}\ D)\ X_D\ U \arrow[r, "\pi_r"] \arrow[d, "\pi_l"'] \arrow[dr, phantom, "P4"] & U \arrow[d] \\
((S\ X_R\ M)\ X_M\ (D\ X\ L))\ X_{D\,X\,L}\ D \arrow[r, "\pi'_r"] \arrow[d, "\pi'_l"'] \arrow[dr, phantom, "P3"] & D \arrow[d] \\
(S\ X_R\ M)\ X_M\ (D\ X\ L) \arrow[r, "\pi''_r"] \arrow[d, "\pi''_l"'] \arrow[dr, phantom, "P2"] & \mathbf{D\ X\ L} \arrow[d] \\
(S\ X_R\ M) \arrow[r, "\pi'''_r"] \arrow[d, "\pi'''_l"'] \arrow[dr, phantom, "P1"] & M \arrow[d] \\
S \arrow[r] & R
\end{tikzcd}$$

Figure 12: Pasted Pullback Category $(((\mathbf{S} \times_{\mathbf{R}} \mathbf{M}) \times_{\mathbf{M}} (\mathbf{D} \times \mathbf{L})) \times_{\mathbf{D}\times\mathbf{L}} \mathbf{D})\mathbf{X_D U}$; **U** university, **D** department, **L** lecturer, **S** student, **M** module, **R** result; $P1, P2, P3, P4$ are labels for diagrams for purpose of discussion

arena. For instance at the intensional level there may be no subtopos as the intension cannot be changed through a query. However, the extension can be changed by a query with the subtopos as an embedding onto the topos through a full and faithful functor [18].

We will give a Boolean example first as while it is not the best for the real-world, it is the easiest to explain. Figure 13 shows a pullback diagram of the product of $\mathbf{1_{CCC}}$ and **S** in the context of Ω, that is $\mathbf{1_{CCC}} \times_{\mathbf{\Omega\{0,1\}}} \mathbf{S}$. $\mathbf{1_{CCC}}$ is the identity for the topos, **S** is an object in the topos and Ω is the subobject classifier. In this simplest case the subobject classifier, or truth object, takes a value from the pair $\{0, 1\}$ and is written $\Omega\{0, 1\}$ to indicate the Boolean logic. χ_j is the characteristic function, that is the query mapping from object **S** to $\{0, 1\}$, false or true respectively. The arrow j is the mapping between the result and the subobject. The result of the query is **U**, the subtopos, derived from the pullback of *true* along χ_j. In typical information systems language χ_j is the query, in say SQL (Structured Query Language), $\mathbf{1_{CCC}}$ is the information base, **S** is the scope of the search, $\Omega\{0, 1\}$ is the type of logic and **U** is the result as a typed structure to give closure.

The arrow *sub* indicates that **U** is embedded in $\mathbf{1_{CCC}}$ so, for the query to be natural, the square should commute, that is $\text{true} \circ \text{sub} = \chi_j \circ j$. However, pure mathematicians

$$\begin{tikzcd}
\mathbf{U} \arrow[r, "\mathrm{sub}"] \arrow[d, "\mathrm{J}"'] & \mathbf{1_{CCC}} \arrow[d, "\mathrm{true}"] \\
\mathbf{S} \arrow[r, "\chi_j"'] & \Omega\{0,1\}
\end{tikzcd}$$

Figure 13: Pullback Diagram: product of $\mathbf{1_{CCC}}$ and $\mathbf{S}$ in the context of $\Omega\{0, 1\}$; subobject classifier is Boolean

$$\begin{tikzcd}
\mathbf{U} \arrow[r] \arrow[d, "\mathrm{J}"'] & \mathbf{1_{CCC}} \arrow[d, "\mathrm{true}"] \\
\mathbf{S} \arrow[r, "\chi_j"'] & \Omega(U)
\end{tikzcd}$$

Figure 14: Pullback Diagram: product of $\mathbf{1_{CCC}}$ and $\mathbf{S}$ in the context of $\Omega(U)$; subobject classifier is Heyting

generally do not attempt this composition because of typing problems, arising from the intension/extension conflagration. Instead they reason about the arrow $sub : \mathbf{U} \longrightarrow \mathbf{1_{CCC}}$ in isolation from the commuting square for the query, using theories such as Lawvere-Tierney [18], which is a generalisation of the Grothendieck topology [12].

In the Heyting example shown in Figure 14 the Boolean subobject classifier $\Omega\{0, 1\}$ is replaced by $\Omega(U)$, the collection of all open subobjects $\mathbf{S}$ of an open object $\mathbf{U}$ in $\mathbf{X}$. The truth value from the viewpoint of an open object $\mathbf{U}$ is the open subobject $\mathbf{S}$ of $\mathbf{U}$ where the assertion is true. The work here on the subobject classifier needs to be developed further, in particular with respect to handling both the intension and the extension and for exploring the related use of the Lawvere-Tierney theories in the topos-subtopos mapping.

9 Closing Remarks

From the perspective of the Universe, the topos approach is confirmed as the optimum way forward for information systems (Section 7), meeting all objectives outlined in our paper at ANPA in 2014 [24], including transaction design for process handling (Section 3), realistic system design (Subsection 8.1) and general interrogation (Subsection 8.2). The concept of normalisation, so important in artificial set-based database techniques, has been revealed as a non-issue in natural information systems. Designing a composition of pasted pullbacks, which commute across all inner and outer diagrams, is the simple principle to be applied. The subobject classifier, in its Heyting form, does appear to provide query closure but more work is required on the intension-extension and topos-subtopos relationships to determine the full potential.

We also looked at the handling of relations generally to confirm that our approach with the topos was optimal (Section 4). Neither set-based approaches nor the category **Rel**, a categorification of the relation as a disjoint union, are Cartesian closed (Section 5). The most promising candidate for a Cartesian approach to relations is the allegory (Section 6), defined in detail by Freyd and Scedrov [11]. From an examination of the usage of allegories, it appears they are suited to relational databases and first order logic languages such as Prolog, with the inevitable weaknesses: the closed world assumption and the Boolean and first order logic. Allegories are based on the regular category, an extension of the pullback, which being Cartesian offers an approach more in harmony with the topos. Indeed we see the scope for improved interoperability between relational systems as Cartesian allegories and other systems, as topos, based on different paradigms. Further work is planned on developing the Kleisli category of the monad/comonad to integrate the topos and the process design and on the subobject classifier in its Heyting context (Section 3).

From the perspective of mathematics, Alfred North Whitehead and Peter Freyd are both twentieth century pioneers. Whitehead, with Russell, developed Frege's set theory [10]. Within that framework Freyd, with Scedrov, studied relationships and extended Eilenberg's categories into allegories. We have attempted to evaluate the category of allegories as a useful structure to implement information systems but find that allegories as a category of sets are inadequate for this purpose because of their restriction to first order Boolean logic. On the other hand Whitehead's later work on process in metaphysics suggests that the topos as a cartesian closed category, without natural numbers but with an internal inherent intuitionistic Heyting logic, as a subobject classifier, will satisfy the requirements to interrogate real world information systems.

References

[1] Arias, Emilio Jesús Gallego, James Lipton, Julio Mariño, and Pablo Nogueira. First-order unification using variable-free relational algebra, Logic Journal

of IGPL **19**(6):790-820 (2011). https://www.cri.ensmp.fr/people/gallego/papers/GLMN11.pdf

[2] Arias, Emilio Jesús Gallego, & Lipton, James B, Logic Programming in Tabular Allegories, Technical Communications of the 28th International Conference on Logic Programming (ICLP'12, A. Dovier and V. Santos Costa (edd.), pp. 334-347 Leibniz International Proceedings in Informatics (2012). http://drops.dagstuhl.de/opus/volltexte/2012/3634/pdf/32.pdf

[3] Arias, Emilio Jesús Gallego, Relational and Allegorical Semantics for Constraint Logic Programming, PhD Thesis, July (2012). http://oa.upm.es/12705/1/EMILIO_JESUS_GALLEGO_ARIAS.pdf

[4] Arias, Emilio Jesús Gallego, James Lipton & Julio Mariño, Declarative Compilation for Constraint Logic Programming, in: Logic-Based Program Synthesis and Transformation LNCS **8981** pp 299-316 (2015). http://link.springer.com/chapter/10.1007%2F978-3-319-17822-6_17

[5] Brown, Carolyn, & Graham Hutton, Categories, allegories and circuit design, Logic in Computer Science, LICS'94 Proceedings, IEEE (1994). http://www.cs.nott.ac.uk/~pszgmh/allegories.pdf

[6] Codd, E F, The Relational Model for Database Management, 2nd edition, Addison Wesley (1990).

[7] Diskin, Z. Generalised Sketches as an Algebraic Graph-Based Framework for Semantic Modeling and Database Design; University of Latvia: Riga, Latvia (1997).

[8] Dumas, Marlon, & Arthur H M Ter Hofstede, UML activity diagrams as a Workflow Specification Language, UML 2001 - The Unified Modeling Language: Modeling Languages, Concepts, and Tools, Springer Berlin Heidelberg, pp. 76-90 (2001).

[9] Finkelstein, Stacy E., Peter Freyd, & James Lipton, Logic programming in tau categories, Computer Science Logic '94 , LNCS 933 (1995).http://jlipton.web.wesleyan.edu/jftp/1990-2000/csl-camera5.pdf

[10] Frege, Gottlob, Grundgesetze der Arithmetik **1** Jena (1893).

[11] Freyd, Peter, & Scedrov, Andre, Categories, Allegories. Mathematical Library **39** North-Holland (1990).

[12] Goldblatt, Robert, Grothendieck topology as geometric modality, Mathematical Logic Quarterly **27**(31-35) 495-529 (1981).

[13] Heather, Michael, & Rossiter, Nick, The Process Category of Reality, ANPA 31, Cambridge 224-262 (2011). http://nickrossiter.org.uk/process/anpa0911.pdf

[14] Hermida, Claudio, & Jacobs, Bart , Structural induction and coinduction in a fibrational setting, **145**(2) 107-152 Information and Computation (1998). http://citeseer.ist.psu.edu/viewdoc/download?doi=10.1.1.36.7400&rep=rep1&type=pdf

[15] Johnson, M & Rosebrugh, R, Sketch data models, relational schema and data specifications. Electron Notes Theor Comput Sci **61** 51-63 (2002). http://www.mta.ca/~rrosebru/articles/sdmrsds.pdf

[16] Johnstone, Peter, Sketches of an Elephant: A Topos Theory Compendium. I, II, Oxford University Press (2002-).

[17] Lawvere, F W, Adjointness in Foundations, Dialectica **23** 281-296 (1969).

[18] Mac Lane, Saunders, & Moerdijk, Ieke, Sheaves in Geometry and Logic (1992),

[19] Mac Lane, Saunders, Categories for the Working Mathematician, 2nd ed, Springer (1998).

[20] Maietti , Maria Emilia & Rosolin, Giuseppe, Unifying exact completions, Applied Categorical Structures **23**(1) pp 43-52 (2015). http://www.math.unipd.it/~maietti/papers/uec.pdf.

[21] Meredith, Lucius Greg, Monadic design patterns for the Blockchain, DEVCON1, Ethereum Developer Conference, Gibson Hall, London, 9-13 Nov (2015). https://www.youtube.com/watch?v=uzahKc_ukfM&feature=youtu.be

[22] Palmgren, E & Vickers, Steven, Partial Horn logic and cartesian categories. Annals of Pure and Applied Logic, **143**(3) pp. 314-353 (2006). http://eprints.bham.ac.uk/198/

[23] Phys Org, Bitcoin's 'blockchain' tech may transform banking, http://phys.org/news/2015-12-bitcoin-blockchain-tech-banking.html

[24] Rossiter, Nick, & Heather, Michael, Formal Natural Philosophy: Top-down Design for Information & Communication Technologies with Category Theory, ANPA 35, Explorations, Grenville J Croll, Nicky Graves Gregory (edd.), 155-193 (2015). http://nickrossiter.org.uk/process/anpa-2015-a5-Latex.pdf

[25] Walters, RFC, Categorical Algebras of Relations http://rfcwalters.blogspot.co.uk/2009/10/categorical-algebras-of-relations.html

[26] Whitehead, Alfred North, & Russell, Bertrand, Principia Mathematica **1-3** 1st edition, Cambridge University Press (1910-13).

[27] Whitehead, Alfred North, Process and Reality: An Essay in Cosmology, Macmillan, New York (1929).

[28] Whitehead, Alfred North, Modes of Thought, Macmillan (1938)

[29] Zieliński, Bartosz, Paweł Maślanka & Ścibor Sobieski, Modalities for an Allegorical Conceptual Data Model, Axioms **3** 260-279 doi:10.3390/axioms3020260 (2014). http://www.mdpi.com/2075-1680/3/2/260

[30] Zieliński, Bartosz, & Paweł Maślanka, Weak n-Ary Relational Products in Allegories, Axioms **3**(4) pp.342-359 doi:10.3390/axioms3040342 (2014). http://www.mdpi.com/2075-1680/3/4/342