FORMAL NATURAL PHILOSOPHY: TOP–DOWN DESIGN FOR INFORMATION & COMMUNICATION TECHNOLOGIES WITH CATEGORY THEORY

NICK ROSSITER & MICHAEL HEATHER

Department of Computer Science and Digital Technologies
Northumbria University, NE1 8ST, UK

<nick.rossiter1@btinternet.com>

Abstract: Information and Communication Systems are part of process in the natural world. Natural as formally defined in category theory needs to be satisfied to provide a full and faithful representation of communication in information systems. Current approaches of translating Information and Communication Technologies into objects and arrows do not compose naturally as categories. Such categorification that loses the naturality of the real world information systems is a major case in point.

The early attempts by Ehresmann to devise types of Sketches and Diskin's later development of his Unified Modelling Language both relax the rigour of category theory. Categorification of the entity-relationship model by Rosebrugh and more recently of Codd's relational model by Spivak show that the real world does not fit a category of sets: rather the identification should be within the well-established natural category of the topos. The topos of the Cartesian Closed Category with naturality provides a formal representation without loss of rigour for the necessary component of a modern information system: formal structuring capability, searching, query symmetry, transaction processing, query closure, transactions and interoperability. The approach from metaphysics includes the future potential for the quantum processing of data.

1 BACKGROUND

About a quarter of large projects in information technology fail. The cost of failure is difficult to assess but exceeds many billions of euros each year over Europe and possibly even more in the United States as discussed in the BCS Report: A Study in Project Failure [4] where the design stage was identified as the greatest contributory to the failure of the whole project. These figures might well be much greater had not the science of databases been developed quite early on with the recognition that the logical structure was more important than even the data itself. So that a database is not just a bank of data but a logical structure with the capability to hold data appropriately typed and with their intra-relations. The development of generalised structures according to some formal model – the Database – was an early application of reusable software with its attendant efficiency. The use of a Database off the shelf is not only much cheaper and more reliable than an *ad hoc* approach but can provide more powerful features with methods and procedures that a bespoke designer might not even foresee as needed.

Most formal information systems today rely on standard database systems involving models based on a mathematical structure as described later in Section 5. The effectiveness of the mathematical model and how faithful it is to represent the logical structure of real world events and data are constrained in two directions. First the limitations of the mathematics used and secondly the confines of the current von Neumann architecture of the machine on which the model is to be mapped. Unfortunately both of these, relying heavily today on number and set theory, are very poor at representing the naturality of the real world. For they have no inherent concept of 'naturality' because they are developed bottom-up *ex nihilo* which cannot provide coherence overall. The philosopher and mathematician Alfred North Whitehead sought to remedy this fundamental defect of modelling in the last century by ascending two levels from the 'sub-naturality' of reductionist models up through the natural reality of every-day perceptions to the highest level of 'supernaturality' in metaphysics as the top-down starter for operations in the concept of process [45]. Whitehead's enlightened concept of process has yet to be subsumed into main stream science. However in the meantime Category Theory has been developed in mathematics based on a set theoretic model of process as represented by 'the arrow'.

In category theory 'natural' has a technical meaning, relating to the definition of natural isomorphism. An object/arrow is defined uniquely over 3 levels up to natural isomorphism. Each level is a collection of arrows as follows:

1. Categories $\mathbf{C}, \mathbf{D}$ comprising objects A, B respectively with identities $1_A : A \longrightarrow A$ and $1_B : B \longrightarrow B$ and inter-object arrows $f : A \longrightarrow A'$ and $f' : B \longrightarrow B'$

2. Functors F, G mapping category $\mathbf{C}$ to $\mathbf{D}$ giving the arrows $F : \mathbf{C} \longrightarrow \mathbf{D}$, $G : \mathbf{C} \longrightarrow \mathbf{D}$

3. Natural Transformation α comparing functors F and G with the arrow $\alpha : F \longrightarrow G$

Objects A, B in categories $\mathbf{C}, \mathbf{D}$ are defined uniquely up to natural isomorphism if the diagram in Figure 1 commutes.

$$\begin{tikzcd}
F(A) \arrow[r, "\alpha(A)"] \arrow[d, "F(f)"'] & G(A) \arrow[d, "G(f)"] \\
F(B) \arrow[r, "\alpha(B)"'] & G(B)
\end{tikzcd}$$

Figure 1: α is natural for object A of $\mathbf{C}$

However that version of Category Theory from pure mathematics is but a half-way house as its concept of naturality is only 'up to the natural isomorphism' as defined by its axiomatic frame from set theory. This is where Natural Philosophy needs to kick in to mediate between reality and its formal representation. In natural category theory 'up to natural isomorphism' is to be interpreted as 'up to the laws of physics'. A database seeks to represent relations and types of data from the real world but a reductionist model based on a classical framework of mathematics cannot guarantee that these are faithfully maintained in the formalism. Hence the failures in the design of information systems. However the naturality of physics cannot be enforced in any formalism just by fiat, by definition or by any other positive assertion: it has to be found empirically from the real world where it inherently resides as a natural property. This is a very important principle that applies to any formal representation in science beyond first order. The natural philosophy needs to be explored further as it is crucial to understanding the process of representing real world relationships in information and communication technology. Indeed it suggests that every scientific result beyond first order needs to be validated in metaphysics.

This pertains to the relationship between category theory and reality and therefore the fundamentals of category theory itself. From what we have learned from the last century it is important for any formalism to satisfy quantum phenomena and the requirements of special and general relativity. This is all explained informally in some detail by Whitehead [45]. Category Theory itself is quite simple although the current manifestation from its provenance in pure mathematics is unnecessarily complicated but it can make the metaphysics of Whitehead a lot less obscure. Natural philosophy provides a 'natural' view of Reality as an adjointness between (using Whitehead's terms) the 'concepts' of quantum phenomena and our 'percepts' of the classical world around us that operate according to the laws of physics. In Figure 2 the left adjoint is the contingent free functor composed of the devolving actual events of the Universe while the deterministic right adjoint provides the unique laws of physics from Archimedes Principle to the Heisenberg Uncertainty Principle, in their archetypal form.

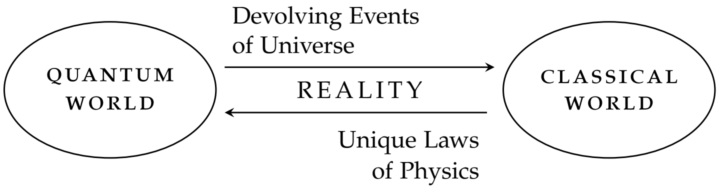

Figure 2: Natural philosophy: Natural reality is an equilibrium between the concepts of quantum process and the 'percepts' of the classical world as we perceive it.

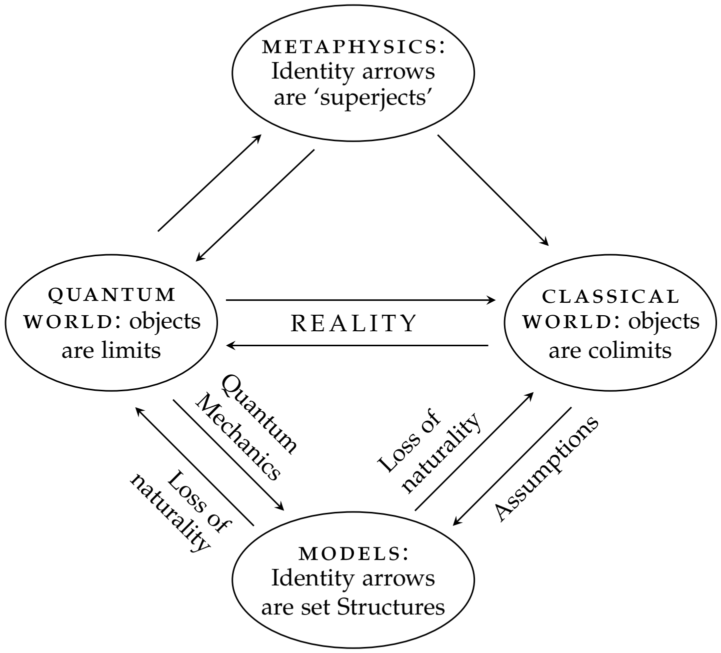
METAPHYSICS: Identity arrows are 'superjects'
QUANTUM WORLD: objects are limits
REALITY
CLASSICAL WORLD: objects are colimits
Quantum Mechanics
Loss of naturality
Loss of naturality
Assumptions
MODELS: Identity arrows are set Structures

Figure 3: Reality is an Instantiation of Metaphysics and Models of Reality.

If we zoom out from Figure 2 to Figure 3 we see how reality sits between metaphysics and models, up to the former, down to the latter. Natural category theory as outlined above operates then at the highest level and provides a formal representation of metaphysics. Reality is an instantiation of metaphysics and the arrow of category theory is defined by the physics constituting the pair of contravariant functors in the adjointness of Figure 2. Objects in category theory are an alternative label for an identity arrow. The Universe as process is just such an identity arrow – the highest possible in the Universe and therefore its closure. Conventionally it is given the label 'topos' following Aristotle's concept. This paper is concerned with exploring the properties of that topos. Models on the other hand both exist in the real world and and are also an instantiation of it. The object of a model is typically a 'subobject' – a general object that has the same name in category theory. Ordinary objects at the reality level exist as 'objects' in the sense used in category theory. Objects at the level of metaphysics might well be termed 'superjects', the term coined by Whitehead for components of metaphysics.

Returning to databases we can see from Figure 4 that, like other models, database models are reductionist.

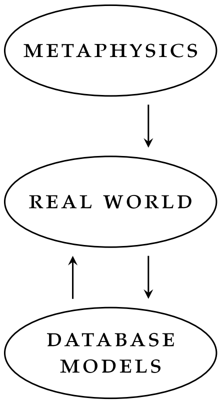

Figure 4: Database Models are Reductionist.

2 METAPHYSICS VERSUS MATHEMATICAL MODELS

Science has come a very long way on the back of arithmetic as first order mathematics and on Euclidean space as first order geometry. However these are not constructive as shown informally by Brouwer and more formally by Heyting but only hold approximately in the real world. Paeno's postulates have been promoted to axioms and hold in pure mathematics but not in physics where they are no more than a working model. The axioms assume the existence of zero, equality and a unique successor concept whereas none of these exist in physics. Absolute zero is a mathematical concept and is unattainable in the real Universe. No two entities in the Universe are exactly equal, every entity there is related to every other and any concept of 'successor' is arbitrary. Concepts at a higher level as found in metaphysics are needed and outlined at some length by Whitehead. Physics is a higher order process but to be distinguished from higher order in mathematics that normally refers only to higher order arithmetic.

A prime example is Einstein who found that he first needed to resort to non-Euclidean geometry to represent his theories of relativity. However it is not everywhere appreciated that he was compelled to go much further and made the switch from mathematics in 1911 [16] to physics in 1916 [17] with his equivalence principle in order to make his *General Theory of Relativity* conform with experimental results. Lo puts it succinctly:

> Theorists have incorrectly assumed that an accelerated frame must be related to a Euclidean subspace. To apply Einstein's equivalence principle, it is crucial that the space-time under consideration must be a physical space. Theorists, both for and against general relativity, have made mistakes by ignoring this [30].

Lo's list of such theorists to have made this mistake comes as quite a shock: Synge, Fock, Pauli, Bergmann, Tolman, Landau & Lifschitz, Zel'dovich & Novikov, Dirac, Wheeler, Thome, Hawking, Hong, Landau & Lifschitz and even Schwarzschild. Nevertheless the conclusion of Lo is that 'general relativity is not a product of just pure thought. Nature is the ultimate authority of science'.

Databases must have the same ultimate authority if they are to represent faithfully relationships in the real world. Existing databases are models built bottom-up and therefore cannot guarantee naturality. A top-down design however is able to begin with naturality and preserve it as it processes downwards. Another important advantage is that metaphysics can reflect the naturality of the quantum world. To preserve naturality requires therefore close attention to fundamentals usually termed 'foundations' in a bottom up approach. This top-down requires close and strict attention to initial definitions. For historic reasons Category Theory has been developed in pure mathematics where an axiomatic approach is favoured. Although any theory may be soundly developed from the axioms of set theory, the difficulty is that a 'set' is nowhere to be found in the physical Universe and cannot claim to be natural. Current textbooks on Category Theory are derived from set theory and cannot be relied on to represent relationships in the real world. However, where Category theory in mathematics is governed only by pure thought, it can be a useful guide to understand how the rigour of nature operates.

3 METAPHYSICAL CATEGORIES VERSUS CATEGORIFICATION

A ‘category’ in pure mathematics is defined bottom up as formed of objects of a given type related by arrows that satisfy axioms of identity, composition and associativity. These axioms are to be found in most current textbooks but the problems inherent in set theory are glossed over. Russell’s paradox is particularly relevant to any higher order operation as necessary for physics or information systems. For instance the tuple[1] is a higher order concept basic to database theory. The other defect is the independence of elements in a set. As Whitehead has also pointed out the world is connected and non-separable. Non-separability has to be externally applied in set theory whereas a category has an internal connectivity through its natural structure.

Representations of set theory often resort to semantics to give a set an implied order such as the set of natural numbers by writing the set $\{1, 2, 3, \ldots\}$ or likewise for an alphabet $\{a, b, c\}$ which is strictly equivalent to the set $\{b, c, a\}$ or $\{b, a, c\}$. An exception from pure mathematics is Adámek’s *Abstract & Concrete Categories* [1], which admits in section 2.3 the need for a foundation requiring three levels: sets, classes and conglomerates, and draws attention to earlier texts which recognised the difficulties such as the Appendix to Herrlich & Strecker’s *Category Theory* [24] in 1979 where a tuple needs to be defined by a closed world assumption.

An object may be defined as an identity arrow but the arrow and the operation of composition are primitives, that is derived from assumptions. The position is analogous to sets and number defined from some arbitrarily defined origin that is not identifiable in physics. Such Category Theory is therefore subject to the same type of error as Einstein’s theory of 1911 [16] when employed to express any features of the real world. Categorification is a further step down that road. Real world features expressed as categories even if proper categories in the sense that they satisfy the axioms are still subject to the same error. Categorification is a further example of modelling and does not take on board Whitehead’s point on the need for metaphysics.

1 In set theory an n-tuple is an ordered collection of n elements. For instance for the Student example shown later in Table 5.1 the first tuple listed is the 3-tuple $<$ ‘1001′, ‘Smith′, ‘2 The Cuttings, Hexham′ $>$, giving the values for id, name and address respectively for a particular student.

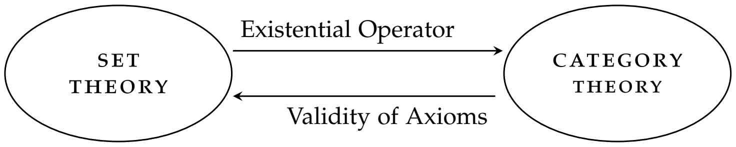

Figure 5: Categorification: Mapping Set Theory on to Category Theory

The simplest example of categorification is for set theory itself. The practice is described in Figure 5. Mapping set theory on to category theory is the left adjoint existential operation of a free functor with a right adjoint defining the qualifications, namely validity up to the natural isomorphism of the axioms of set theory which are insufficient to represent the real world. Categorification of databases would be to take the procedures of classical database theory, for example operations with Codd's Relational Model or rules of SQL, and express them in category theory as shown in Figure 6.

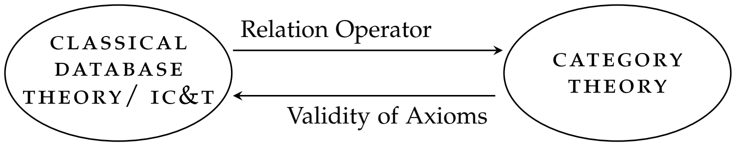

Figure 6: Categorification: Current Theory and Practice of Information & Communication Technology expressed as Category Theory

The last twenty years has seen a series of attempts to categorify physics led by Crane [9] in the early 1990s. Attention has been mainly directed at current problems in theoretical physics. Baez has initiated a series of projects, such as quantum gravity, quantum field, knots, gauge theory and black holes, employing categorification; more details can be found elsewhere, for instance on his approach to quantum physics [3]. In the light of Einstein's comments on the ultimate judge being nature, it is difficult to see how such categorification can produce elementary results beyond that available using the classical methods of set theory. For categorification is always limited by being founded in graph theory. The alternative is to use the naturality of category theory at the metaphysics level, operating in both top-down and bottom-up modes through adjointness, as opposed to the bottom-up approach of set theory.

4 PROCESS IN OPEN SYSTEMS

A further reason to invoke Category Theory and Process for information systems is because of advances in globalisation information systems, which need to be interoperable with the capability to communicate seamlessly with one another. For this to come about there needs to exist a coherence between information systems. Previously communications was a separate subject as in Weaver & Shannon's *Theory of Communication* [43] where the communication channel was restricted to information within a Boolean syntax. However the work advanced the understanding of information with the use of methods borrowed from statistical mechanics. While Weaver and Shannon advanced our understanding they dealt only with syntax in one channel. Now category theory raises the levels to integrate the semantics of communication in the communication process. This is important for databases where the semantics of relationships is critical.

5 CURRENT INFORMATION SYSTEM

Existing approaches to information systems are illustrated formally by database systems. Such systems are employed for the persistent storage of data, according to the definition and rules of a schema. The data can be retrieved and updated through a query language. The form of the data definition and query language is governed by the data model employed. Such models are based on a standard mathematical structure for reliability and efficiency. Each model separates the intension of the data, the definition, from the extension, the data values themselves.

Early approaches include the hierarchical data model, based on trees, and the network data model, based on graphs. The network data model was extended with the object-oriented paradigm to include objects, methods and a range of data abstractions; this paradigm has proven to be most useful for handling complex data where the methods can be integrated with an object-based query language. By far the most popular in data processing today is the relational data model, based on sets [10], where the intension describes the relationships and the extension is a set of tuples, represented as tables. Attempts to use functions as the basis have been attempted but have not gained practical acceptance. Details of all the various database approaches can be found in textbooks, including that by Connolly & Begg [8].

A graphical front-end to the relational model, for design purposes, is the entity-relationship model [6], with various enhancements to extend the semantics covered. It is considered that the graphical approach, with entities as nodes, connected by relationships as mappings, facilitates the construction of complex design structures. The design structures created can then be automatically mapped into a database design, for example for a relational data base. However, the entity-relationship graph can be considered as an incomplete data model as it has no inherent facilities for a query language. It might be better to view it instead as a design notation.

5.1 *Example Databases*

An example relational database is shown below in Table 1 for an application involving students' module choices and outcomes. For each table the top row holds the data definition, the intension, and the remaining rows the data values, the extension. The heading of each column is termed an attribute, each of which will be assigned a data type as described later. The $*$ against an attribute indicates that it is part of the primary key, the identifier of the table. There are links between the tables. Each Outcome.id value references Student.id and each Outcome.no value references Module.no. The attributes referencing other attributes as primary keys, termed foreign keys, are marked by a †.

Queries can be made on the tables using the SQL standard language. For instance the query "give full details of the students who have passed the module Programming" is expressed as:

STUDENT

id*	name	address
1001	Smith	2 The Cuttings, Hexham
1002	Jones	7 Crescent Way, Newcastle
1003	Roberts	The Grange, Corbridge

MODULE

no*	title	level
CS001	Programming	4
CS057	Architecture	5
CS124	Databases	4

OUTCOME

id*†	no*†	mark	decision
1001	CS057	65	P
1001	CS001	55	P
1002	CS057	35	F

Table 1: Relational Database Example: Module Outcomes for Students

```
SELECT STUDENT.*
FROM STUDENT, OUTCOME, MODULE
WHERE OUTCOME.id = STUDENT.id
AND OUTCOME.no = MODULE.no
AND MODULE.title = 'Programming'
                    AND OUTCOME.decision = 'P';    (1)
```

Figure 7(a) shows an entity-relationship diagram for the application. STUDENT and MODULE are entity-types in an N:M cardinality (many–to–many) relationship Outcome; each student entity is linked with between 0 and M module entries; each module entity is linked with between 0 and N student entries. The zeros (0) are significant indicating that neither a student entity nor a module entity has to be associated with each other through the Outcome relationship; this is optional participation. The general form of the cardinality property is $l, u : l', u'(E : E')$ where l is a lower-bound (some minimum[2]), u is an upper-bound (some maximum) and E, E' are entity-types. For example the cardinality for the above relationship is 0,N:0,M (Student:Module) meaning that each student entity is connected to between 0 and M modules and each module entity is connected to between 0 and N students. The entity-relationship model is often converted into the relational model for implementation purposes. As relationships with N:M cardinalities cannot be directly handled in the relational model, the diagram is converted to replace each N:M relationship by two one–to–many relationships as shown in Figure 7(b) where OUTCOME becomes an entity-type, associated with STUDENT by the relationship Has and with MODULE by the relationship Gives. The lower cardinality values for STUDENT and MODULE are now 1, indicating mandatory participation in the relationships Has and Gives from OUTCOME. The entity-relationship model has been extended with data abstractions such as inheritance and aggregation, the former representing subclasses such as the various types of student and the latter constructed collections of classes.

6 USE OF CATEGORIES

Category theory has been applied to existing database methods by a number of workers. The applications have attempted to replicate models including the relational and entity-relationship in categorial structures. Such categorification provides useful support for current database models but does not realise the full potential of category theory for advancing database techniques.

In categorification existing techniques are transformed on a 1:1 basis from application to categories. Examples of attempted applications are the work by Rosebrugh and co-workers [25, 26], Diskin [11, 12]

2 the lower bound is not necessarily 0, e.g. a child has a lower bound of 2 parents naturally but the upper bound may not be 2 with current genetic engineering and artificial insemination.

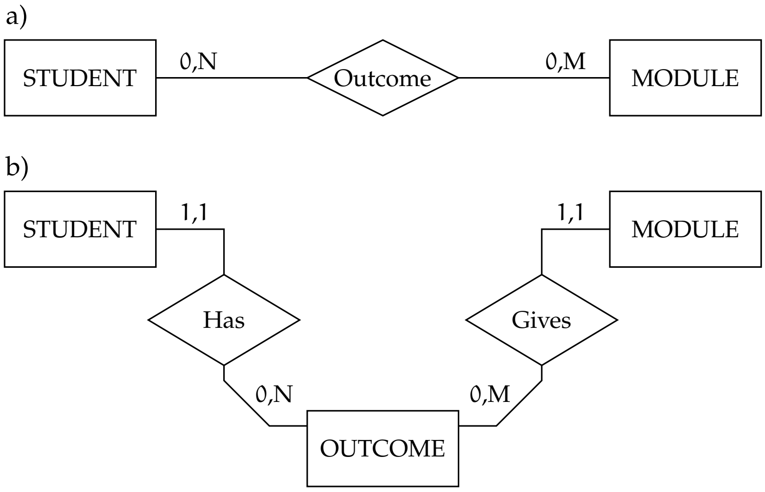

Figure 7: Entity-Relationship Example: Module Outcomes for Students. a) Student and Module in direct N:M relationship; b) Student and Module in two one-to-many relationships with Outcome. Rectangles indicate entity-types and diamonds indicate relationships. 0 . . . M, N are cardinalities.

and Spivak [39, 40, 41]. The models subject to categorification are the entity-relationship model with Rosebrugh and his co-workers, the Unified Modelling Language (UML) with Diskin and the relational model with Spivak.

The entity-relationship model is a graphical technique for designing databases as described in Figure 7. The graphical nature has encouraged the use of categorical sketches as a means of representation. Sketches were developed by Charles Ehresmann [13, 14, 15] as a flexible technique for relaxing certain of the criteria needed for a category to be robustly designed. There are many types of sketches and a taxonomical analysis has been made by Wells [44]. The preferred type of sketch for information systems has been the finite discrete sketch, the fifth in Well's list in subsection 3.1.5:

> 3.1.5 A finite discrete sketch has only discrete cones and cocones. It is usually required that the models of a finite discrete sketch (and a finite sum sketch (see 3.1.7) be in

> a category with finite disjoint sums (see [Barr and Wells, 1990], page 219, or any book on topos theory). This is discussed in Section 5.3. The category of fields is the category of models of a finite discrete sketch.

Traditionally a finite discrete sketch S is viewed as a graph G with:

- a set of diagrams D
- a set of discrete cones L (limits)
- a set of discrete cocones R (colimits)

A sketch S may be written as a 4-tuple set $< G, D, L, R >$ with G representing the data structure, D the constraints, L the relationships, R the attributes (properties). A model graph homomorphism M maps the graph G as an intension to an extension category C (a database state), taking associated diagrams in D to commutative diagrams, cones in L to limit cones and cocones in R to colimit cocones. M preserves products. In entity-relationship terms, the graph G, with associated constructions in the 4-tuple, is the class structure and the model M is the objects. This general definition of a finite discrete sketch is simplified by Johnson and Rosebrugh [26] to the EAS (Entity-Attribute Sketch), defined as $E =< E, L, R >$ where E is any entity in a database and L and R are as previously defined. Note that the 4-tuple is reduced to a 3-tuple. The model M subsumes its constraints D and maps E to C. The model M is embedded in a left exact category, a Cartesian category with finite coproducts wherein coproducts are preserved by pullbacks, providing facilities to define classes for database interrogation. The EAS has been implemented as a Java application, Easik, for database design, database implementation and data manipulation in a graphical environment [35]. The guide to Easik claims:

> Within the Easik graphical interface users create a database design of entities, attributes and constraints. The design can be exported to a database schema in SQL that enforces the graphical constraints. Easik is compatible with connectivity to some common database management systems [such as MySQL]. With a connection available, data entry and manipulation can be done via the graphical interface [35].

This quote shows that Easik is not a categorial database but is a front-end design tool for a relational database upon which all subsequent manipulation will be done in the usual way through SQL. An example of an Easik sketch is shown in Figure 8. This is taken from a more detailed description of the computing science behind the project [27]. On the graph, entities are indicated by rectangles and relationships by directed edges ($\longrightarrow$). Constraints may be of several types, including sum for subclasses, commutative diagrams, product and pullback. In the process of categorification, the sketch is then converted to a relational database design by a java program.

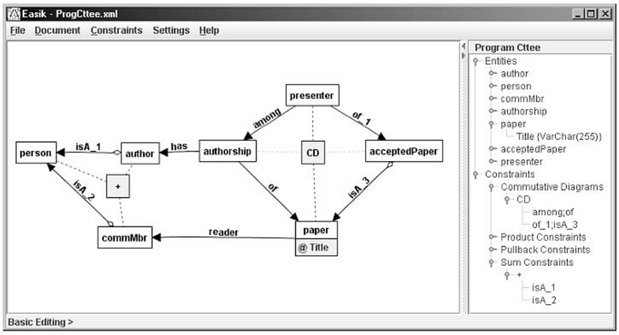

Figure 8: Example of an Easik Sketch: Publications at a Conference, from [27].

UML has a class structure close to that of the entity-relationship model and indeed Diskin [11] emphasises the similarity. The motivation behind Diskin's approach is "a general thesis that any diagram with precise semantics (to be described in mathematical terms) actually hides a sketch in a suitable signature of markers". Diskin shows that the sketch is a more precise definition than the design models, arguing that "the sketch view we suggest gives rise to a whole program of refining the vocabulary of visual modeling, making it precise and consistent, and unified". At the same time, he indicates that "by suggesting sketches we do not wish to force everyone to use the same universal graphical language".

Sketches are attractive at first sight in that they provide a readily visualisable graphical structure, popular in information system design. They also enable imprecise features of an application to be handled through relaxing the strict constraints of commutativity in category theory. However the structures developed may not satisfy the axioms of category theory as they are not using categories in a rigorous way. Wells considered that sketches "were invented by Ehresmann to provide a mathematical way to specify a species of mathematical structure" [44], such as groups and sets, providing a possible route for handling interoperability. Diskin has recently discussed the possibility of using sketches for metamodelling in information engineering [12]. Wells [44] thought that sketches were particularly useful for multisorted structures and for models in categories other than sets. It therefore appears that sketches may appeal to some as a convenient graphical language, with the potential for capturing semantics and for comparing different models, but is this only superficial? Work with sketches to date is more in the nature of a manifesto or a mission statement than as a proven tool. In earlier work [36] we briefly investigated the use of such sketches and came to similar conclusions.

Turning to Spivak's approach, he says "A database schema is a system of tables linked by foreign keys. This is just a category". But this may confuse intension and extension. This categorification of the relational model involves treating tables as objects within a category with cross-references between the tables, from foreign key to primary key, as functions. The effect is to construct a network through graphs. There seems to be an immediate problem with this approach. The table has complex internal structure of its own, which would need to be captured at the intra-object level. Further the correspondence of a table to a relation is only approximate with a relation referencing domains for typing purposes and containing functionality such as methods (see definition) in object-based extensions. The table also contains much more than just the schema: it also contains the data, as shown in our example of a relational database with the intension, the extension and the various types of key, given in Table 1.

Spivak captures the set-based nature of the relational model by introducing a set-valued functor to map from the schema to the category **Set**; the set-valued functor represents the instances of data conforming to the schema. This mapping is analogous to the model M model functor in the sketch approach. Typing is introduced by mapping values on to type objects e.g. String, Real, as in the functional model. Adjoints are considered between one set-valued functor and another for views and queries and for the purpose of data migration. It appears that Spivak is using the higher-order logic of Lawvere [29], with the adjoints $\exists \dashv \Delta$, $\Delta \dashv \forall$. Spivak appears to interpret Σ ($\exists$) as union, Δ as projection and Π for join. This seems at odds with Lawvere's treatment of $\exists$ as the existential quantifier, Δ as the diagonal re-indexing functor and $\forall$ as the universal quantifier, which is also the interpretation taken in our own work described later.

All the current approaches are clearly 'categorification' with a translation of current information system approaches to category theory rather than the more fundamental approach of determining which categorial structures would be the ideal basis for an information system. Thus the sketch approaches are a translation of the entity-relationship model into categories and Spivak's approach is a translation of the relational model into categories. The approaches are therefore backward looking and *ad hoc*. The flexibility of the sketch approach is claimed to be an asset but the ability to relax the commutativity requirement in some cones and cocones means the sketch is not a natural structure and therefore is an inappropriate formalism upon which to base an information system. The terminal object requirement is not enforced in sketches so there is no closure in such structures at the natural isomorphism level. The approaches in general fail as Whitehead reductionist models, not meeting the requirements of metaphysics.

More fundamental approaches to representing databases in category theory were followed in the 1990s by Rosebrugh [34] and Baclawski [2] with the database being embedded in a topos. However, such ideas were not further pursued, maybe because the full potential of the topos was not appreciated at the time or because the exact world of the topos could not handle the required relaxations of the theory.

7 FUNDAMENTAL APPROACH TO INFORMATION SYSTEMS WITH CATEGORIES

Starting from basics, we need to decide on requirements for an information system, identify features of category theory that help to meet requirements and produce a framework which satisfies the software engineering principles of high cohesion and low coupling.

7.1 *Requirements*

The requirements are for a database system, the basic properties of which have been described in Section 5. The requirements can be itemized as follows:

1. MATHEMATICAL STRUCTURE. For behaviour to be predictable with certainty, it is essential that underlying data structures have a mathematical rather than an ad hoc basis e.g. set operations would be used in a relational database where the set is the basic unit of data.

2. STRUCTURING CAPABILITY WITH HIGH COHESION, LOW COUPLING AND STRONG TYPING. The data should be able to be structured according to some formalism, which enables closely related data to be grouped together, giving high cohesion, but also permits relationships to be explored on demand, without such relationships always being pre-specified in a hard-wired manner, giving low coupling e.g. in a relational database, a set can be regarded as a class defining a number of closely related properties but in principle any property can be related to any another property, wherever defined, through the query language. Strong typing means data is to be checked against a pre-defined type system. Typing includes data model constraints such as participation rules and cardinalities for relationships.

3. SEARCHING AND MANIPULATION. The data should be searchable, updateable and deletable by a query language, employing operations suitable for the mode of data structuring e.g. a tree traversal query language would be used in a hierarchical database. Internal methods for classes are an integral part of the query system.

4. QUERY SYMMETRY. The way in which users express queries should reflect the complexity of the request and not be unduly affected by quirks in the data model.

5. QUERY CLOSURE. The result of a query can be held as a data structure, which ranks *pari passu* with other data structures in

the database. The situation desired is as with SVG (Scalable Vector Graphics) where images can be manipulated and saved as first class objects for further transformations.

6. TRANSACTIONS. Update operations can be grouped together into a transaction, whereby the collection of operations is regarded as an atomic unit with rules governing the success or failure of the process; the process executes completely or is rolled back so no changes are recorded.

7. INTEROPERABILITY. The ability for information systems to talk to one another is a critical requirement, needed today even if the systems are based on different paradigms.

7.2 *Potential of the Topos*

In previous work we have looked in depth at the Cartesian Closed Category (CCC) as the basis for natural systems. With its cartesian product for representing relationships and its terminal object for identity, the CCC is the obvious starting point [37, 38]. However, in our more recent work the topos has been given more prominence [22, 23]. The topos is a CCC but it has a number of additional features, useful for our work:

1. the subobject classifier for membership criteria

2. the internal logic of Heyting for query and rule processing

3. the ability to define subtopos, such as the reflective subtopos

4. the co-cartesian (dual) of the topos for exploration of database design issues

7.3 *The Topos: Definition and Properties*

The archetype of the natural world is the topos, in its early days formally defined as a Cartesian Closed Category with subobject classifiers and informally as a generalised set. Johnstone in his preface to *Sketches of an Elephant* [28] lists thirteen alternative descriptions that have been applied to the topos (pp. viii &sq). Many of them like for instance “A topos is a generalised space” still carry hangovers from sets. We would recommend as an informal definition: “The category of categories of categories”, where categories describes a structure of classes, after Aristotle’s *Organon*. There is a unique arrow from the source of the World to every object in it and a unique limiting arrow directly between any pair of objects as well as a repletion of indirect co-limiting arrows between them. These relationships satisfy our empirical perception of ‘the laws of physics’. As a categorical structure the topos has attracted much attention in standard texts, for example Mac Lane’s *Categories for the Working Mathematician* [32], Goldblatt’s *Topoi: The Categorial Analysis of Logic* [20], Johnstone’s *Sketches of an Elephant: A Topos Theory Compendium* [28]. There are also a number of reports, which make the material more accessible, for example Pettigrew’s *An introduction to toposes* [33]. The popularity of the topos approach appears to be because the topos captures properties of sets, based on the Cartesian Closed Category (CCC), which will be considered first.

Cartesian Closed Category

A building block for the topos is the Cartesian Closed Category, for which the pullback is a well-known example. A CCC is a category with:

1. All products, with all objects A, B related by products $A \times B$. This enables relationships to be expressed between any two objects.

2. Closure with a terminal object 1, where there is exactly one arrow from every object in the category to the terminal object. The terminal object is the least upper bound.

3. Exponentiation (connectivity), with the collection of arrows from the object $A \times B$ to the object C being equivalent to the collection of arrows from the object A to the exponential object C^B, that is $\hom(A \times B, C) \equiv \hom(A, C^B)$. For a topos ξ the following expression holds:

$$F : \_ \times B : \xi \longrightarrow \xi ; G : \_^B : \xi \longrightarrow \xi ; F \vdash G \quad (2)$$

$$\begin{tikzcd}
& A \arrow[dr, "i"] & \\
A \times_C B \arrow[ur, "\pi_l"] \arrow[dr, "\pi_r"'] & & C \\
& B \arrow[ur, "j"'] &
\end{tikzcd}$$

Figure 9: The Pullback, an example of a Cartesian Closed Category

A simple pullback is given in Figure 9 where C is the colimit, $A \times_C B$ the product of A and B in the context of C is the limit and $\hom(A \times B, C) \equiv \hom(A, C^B)$ is the expression for the exponential. A is the independent variable and B the dependent variable. Arrows $\imath$ and $\jmath$ are inclusions and π_l and π_r projections. The inclusions indicate that although the colimit is simply written as C, it must also contain A and B. In the simplest case the colimit might be written $A + B$ as the disjoint union of A and B but amalgamated sums of various types are possible depending on the semantics. An important special case is where π_l is monic (one path from $A \times_C B$ to A): the diagram is then both a pullback and a pushout with the colimit being $A +_C B$, the sum of A and B in the context of C. Such a diagram is called a Dolittle diagram[3], which we employed in the development of a universal logic [21] as it provides natural closure. In an analogous manner, the limit can be either a product, a product in context or an intersection, the nature of which depends on the semantics.

A more complete pullback diagram, additionally showing its dual, reveals the full power of the logic as shown in Figure 10. The information significance attached to each projection and inclusion arrow is summarised in Table 2. The table shows that the semantics captured by the full pullback diagram are very rich including typing features

3 The concept is also known as a pulation diagram [1]; Freyd introduced the concept [18] but unfortunately called it incorrectly a Doolittle diagram.

arrow	source	target	operation	type implications
π_l	$A \times_C B$	A	projection	if epic A has mandatory participation in $A \times_C B$: 1,*: *1,1* (C:A) if not epic A has optional participation in $A \times_C B$: 0,*: *1,1* (C:A)
π_r	$A \times_C B$	B	projection	if epic B has mandatory participation in $A \times_C B$: 1,*: *1,1* (C:B) if not epic B has optional participation in $A \times_C B$: 0,*: *1,1* (C:B)

π_l^*	A	$A \times_C B$	dual of projection	if monic $A \times_C B$ is *1,1*:*,1 (A:C)
				if not monic $A \times_C B$ is *1,1*:*,N (A:C)

π_r^*	B	$A \times_C B$	dual of projection	if monic $A \times_C B$ is *1,1*:*,1 (B:C)
				if not monic $A \times_C B$ is *1,1*:*,M (B:C)

arrows above				derive, from above, minimum and maximum cardinalities for participation of A, B in $A \times_C B$
$\imath$	A	C	inclusion	$A \in C$
$\jmath$	B	C	inclusion	$B \in C$
$\imath^{-1}$	C	A	superobject	$C \supset A$
$\jmath^{-1}$	C	B	superobject	$C \supset B$

Table 2: Information System Semantics: Types of Projection and Inclusion Arrows in the Pullback in Figure 10. Cardinalities in form $l, u : l', u'(E : E')$ are as described in Section 5.1. Cardinalities in italics are trivial, from the definition of a pullback. Cardinality of * is a wild-card indicating the value is unknown from this test alone,

such as cardinalities and participation for relationships as well as the more obvious projection and inclusion of Figure 9. The diagonal arrows $\exists, \Delta, \forall$ carry additional significance, associated with query language functionality, as shown in Table 3. All of the features of the relational calculus and its commercial interface SQL are to be found in the full Cartesian Closed Category of Figure 10. With respect to the relational database algebra, each operator may act as follows:

- π: projection from the limit $A \times_C B$ with $\imath \circ \pi_l$ or $\jmath \circ \pi_r$ picking out objects in C

$$\begin{tikzcd}
& A \arrow[dl, shift right=2, "\pi_1^*"'] \arrow[dr, "i"] & \\
A \times_C B \arrow[ur, "\pi_1"] \arrow[rr, shift left=3, "\exists"] \arrow[rr, shift right=3, "\forall"'] \arrow[dr, "\pi_r"'] & & C \arrow[ll, "\Delta"'] \arrow[ul, shift right=2, "i^{-1}"'] \arrow[dl, shift left=2, "j^{-1}"] \\
& B \arrow[ul, shift left=2, "\pi_r^*"] \arrow[ur, "j"'] &
\end{tikzcd}$$

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

- $\exists$: restrict where the relationship between source $A \times_C B$ and target C is queried according to some predicate defined by σ
- $\exists$: predefined join where the relationship between A and B in the context of C is explored from the limit $A \times_C B$
- $\exists$: intersection with source as $A \times B$ and target as the object C/B, the slice of C connected to B
- $\exists$: unrestricted product with target as the universal object $\{*\}$
- $\exists$: amated sum with target as $A +_C B$, when π_l is monic
- $\exists$: unrestricted disjoint union with source as the universal object $\{*\}$ and target as $A + B$
- $\forall$: divide as an example of a universal property, defined by the predicate σ
- Δ: query closure as non-contingent truth operator, right-adjoint to $\exists$ and left-adjoint to $\forall$

arrow	source	target	operation	query facilities
$\exists$	$A \times_C B$	C	existential quantifier, left adjoint to Δ	application of predicate
$\exists$	$A \times_C B$	$\{*\}$	existential quantifier, left adjoint to Δ	product operation
$\exists$	$A \times B$	C/B	existential quantifier, left adjoint to Δ	an intersection operation
$\exists$	$\{*\}$	$A + B$	existential quantifier, left adjoint to Δ	disjoint union operation
$\exists$	$A \times_C B$	$A +_C B$	existential quantifier, left adjoint to Δ (if π_1 is monic)	sum operation in context
Δ	C	$A \times_C B$	diagonal, re-indexing, right adjoint to $\exists$, left adjoint to $\forall$	query closure
$\forall$	$A \times_C B$	C	universal quantifier, right adjoint to Δ	application of predicate
$\exists \dashv \Delta$			adjoint	validates existential query results $< \exists, \Delta, \eta, \epsilon >$ where $\eta : 1_{A \times_C B} \longrightarrow \Delta\exists(A \times_C B)$, $\epsilon : \exists\Delta(C) \longrightarrow 1_C$
$\Delta \dashv \forall$			adjoint	validates universal query results $< \Delta, \forall, \eta, \epsilon >$ where $\eta : 1_C \longrightarrow \forall\Delta(C)$, $\epsilon : \Delta\forall(A \times_C B) \longrightarrow 1_{A \times_C B}$

Table 3: Information System Querying: Types of Diagonal Arrows in the Pullback in Figure 10

.

So the logic provided by the pullback diagram in Figure 10 is relationally complete, according to relational database theory [7]. The general form of an existential query is:

$$\pi((\sigma_{\text{predicate}}\exists) \dashv \Delta) \tag{3}$$

and of a universal query:

$$\pi(\Delta \dashv (\sigma_{\text{predicate}}\forall)) \tag{4}$$

The form of Expression 3 is analogous to that of the relational algebra, where the general form of query is:

$$\pi(\sigma_{\text{predicate}}R) \tag{5}$$

In the categorial form it is arrows that are searched such as $\exists$, rather than relations R as sets. The categorial queries must also satisfy the property of adjointness, readily providing the closure property. For existential queries the base of the query is the adjoint $\exists \dashv \Delta$ and for universal queries the base is $\Delta \dashv \forall$.

As stated earlier query symmetry is an important requirement for an information system. This is naturally achieved in CCC through the commutativity of the diagrams. The changing of the order of the operands (arrows in this case) does not affect the result.

7.4 *Additional Properties of the Topos*

From the information system perspective, products are the basis for relationships, closure the basis for identity and exponentiation the basis for connectivity. This is a useful starting point but not sufficient for complete specification of an information system. If we call the Cartesian Closed Category of Figure 10 **CCC** then a topos is the category **CCC** with the four additional properties given in Section 7.2. We consider firstly the two properties which enhance searching ability: the subobject classifier and the internal logic, Heyting. The subobject classifier is for membership criteria, the internal logic Heyting for query and rule processing. The key notion of a subtopos is next considered to explore the typing of the result of a query. The co-cartesian dual of the topos is not considered in this paper; it will be dealt with in a further paper on database design including normalisation. Finally we look briefly at whether there are any disadvantages to the topos approach.

Subobject Classifier

A subobject classifier lives within the topos. The classifier is defined by the pullback square in Figure 11 where Ω is a collection of truth values, true is the subobject classifier, 1 is the terminal object of the category **CCC**, $j : U \longrightarrow X$ is an inclusion arrow mapping from the subobject U to the object X and χ_j is the characteristic function. The subobject of X defined by the characteristic function χ_j is U.

$$\begin{tikzcd}
U \arrow[r] \arrow[d, "j"'] & 1_{CCC} \arrow[d, "\text{true}"] \\
X \arrow[r, "\chi_j"'] & \Omega
\end{tikzcd}$$

Figure 11: Pullback Square for Subobject Classifier: Derivation of j mapping from subobject U to Object X

The Heyting Internal Logic

The internal logic of a topos is analogous to methods in an object-oriented database. The logic augments the quantifiers and associated relational algebra operations defined in Section 7.3. An example language is the Mitchell-Bénabou Language of a Topos, defined in various texts [5], ([28] Volume II), [31]. In this language types and variables are defined. Expressions are built from formulae and predicates are constructed for membership tests. Logical operations available include intersection and union. The internal logic is intuitionistic (Heyting) which is more general than Boolean. As an example the handling of negation is more sophisticated in that a double negative may not result in a truth value. Some examples of the internal logic will be given in a future paper.

The Reflective Subtopos

After a database query has been executed, it is sometimes required to store the result as a database instance in its own right. This requires the type of the result to be known. If this can be done, the database is said to possess query closure. For the topos it is simple to hold the search result as a subtopos, whose type is defined by the functorial relationship between the topos and the subtopos. A construction that appears particularly promising is the Lawvere-Tierney topology, which is a closure operator on the subobject classifier of a topos [31].

Freyd & Scedrov [19] introduce the reflective subtopos, where the subtopos maps on to a topos through a full and faithful functor. The reflective subtopos can be a member of itself, giving a distinct advantage over set theory where a set may not be a member of itself. Reflective subtopos can be defined recursively as monads. From the perspective of querying the universe, the reflective subtopos is identified as a fragment either as a partial order subobject or as a pullback where the context is the universe and the product is of the two related objects. Adjointness between the topos and subtopos means the relationship is exact.

7.5 *Arguments against Topos*

While we see many formal advantages to a topos approach in achieving a natural information system, it has to be observed that there are currently no readily-accessible examples of usage in practical applications. In addition, as mentioned earlier in Section 1, an implementation on current computer architecture would be constrained by the von Neumann architecture currently in use, based on number and therefore on set.

8 WORKED EXAMPLE OF DATABASE APPLICATION AS A TOPOS

The database application in Section 5.1 is now to be represented as a topos. The starting point is to represent the example as the Cartesian Closed Category **CCC**, with its dual, shown in Figure 12, following the earlier pullback example given in Figure 10. The diagram features both **M** and **M**′ for Module and **S** and **S**′ for Student. **M** is the total collection of modules and **S** is the total collection of students. $\mathbf{S'} \times_{\mathbf{O}} \mathbf{M'}$ involves only those students and modules with an outcome, that is those students and modules that participate in the relationship **O**. Building on Table 2 we can now examine the typing of the various arrows in Figure 12 to capture the semantics input to the entity-relationship diagram of Figure 7(b). Table 4 shows the typing assigned, in particular that the cardinality of the relationship $S' \times_O M'$ is 1,1:0,N (S:O) and 1,1:0,M (M:O). In the entity-relationship model such typing is included as labels, which is much weaker than the inherent typing of the categorial formalism.

$$\begin{tikzcd}
& \mathbf{S} \arrow[dl, shift right=2, "\pi_1^*"'] \arrow[dr, "i"] & \\
\mathbf{S'} \times_{\mathbf{O}} \mathbf{M'} \arrow[ur, "\pi_1"] \arrow[rr, shift left=3, "\exists"] \arrow[rr, shift right=3, "\forall"'] \arrow[dr, "\pi_r"'] & & \mathbf{O} \arrow[ll, "\Delta"'] \arrow[ul, shift right=2, "i^{-1}"'] \arrow[dl, shift left=2, "j^{-1}"] \\
& \mathbf{M} \arrow[ur, "j"'] \arrow[ul, shift left=2, "\pi_r^*"] &
\end{tikzcd}$$

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

There is internal structure to all the categories. This is needed to represent the attributes in the application described in Figure 1. To represent this detail, the categories **S**, **M** and **O** in Figure 12 are represented as the pullbacks in Figure 13. Figures 13(a) and (b) show pull-

arrow	source	target	operation	type
π_l	$S' \times_O M'$	S	projection	not epic: S optional participation, $S' \times_O M'$ is 0,*:*1,1* (O:S)
π_r	$S' \times_O M'$	M	projection	not epic: M optional participation, $S' \times_O M'$ is 0,*:*1,1* (O:M)
π_l^*	S	$S' \times_O M'$	dual of projection	not monic: $S' \times_O M'$ is *,N:*1,1* (O:S)
π_r^*	M	$S' \times_O M'$	dual of projection	not monic: $S' \times_O M'$ is *,M:*1,1* (O:M)
arrows above				$S' \times_O M'$ is 1,1:0,N (S:O) and 1,1:0,M (M:O)
$\imath$	S	O	inclusion	$S \in O$
$\jmath$	M	O	inclusion	$M \in O$
$\imath^{-1}$	O	M	superobject	$O \supset A$
$\jmath^{-1}$	O	M	superobject	$O \supset M$

Table 4: Information System Semantics: Types of Projection and Inclusion Arrows in the Pullback for the Worked Example. Cardinalities in form $l, u : l', u'(E : E')$ as described in Section 5.1. Cardinalities in italics are trivial, from the definition of a pullback. Cardinality of * is a wild-card indicating the value is unknown from this test alone,

backs for categories **S** and **M** respectively where the pullback is of the arrow π_l with itself; these are termed kernel pairs. Figure 13(c) shows the pullback where the projection arrows differ: π_l mapping to id and π_r mapping to no. The colimits of these final level categories correspond to the relational tables of Table 1. An alternative representation considered was to use categories of type **Rel**, representing binary relationship between objects, for the underlying types. However, as **Rel** is not cartesian closed it has no identity functor and is not suitable for the representation of the categories as objects in Figure 12. Since the use of **Rel** is in effect categorification, this is another example of the problems of such an approach. The use of Freyd & Scedrov's generalisation of relations, namely allegories [19], should be considered in this context.

(a)
$$\begin{tikzcd}
& \mathrm{id} \arrow[dl, shift right=2, "\pi_1^*"'] \arrow[dr, "i"] & \\
S_X \arrow[ur, "\pi_1"] \arrow[dr, "\pi_r"'] \arrow[rr, shift left=2, "\exists"] \arrow[rr, shift right=2, "\forall"'] & & S^+ \arrow[ul, shift right=2, "i^{-1}"'] \arrow[dl, shift left=2, "j^{-1}"] \arrow[ll, "\Delta"'] \\
& \mathrm{id} \arrow[ul, shift left=2, "\pi_r^*"] \arrow[ur, "j"'] &
\end{tikzcd}$$

(b)
$$\begin{tikzcd}
& \mathrm{no} \arrow[dl, shift right=2, "\pi_1^*"'] \arrow[dr, "i"] & \\
M_X \arrow[ur, "\pi_1"] \arrow[dr, "\pi_r"'] \arrow[rr, shift left=2, "\exists"] \arrow[rr, shift right=2, "\forall"'] & & M^+ \arrow[ul, shift right=2, "i^{-1}"'] \arrow[dl, shift left=2, "j^{-1}"] \arrow[ll, "\Delta"'] \\
& \mathrm{no} \arrow[ul, shift left=2, "\pi_r^*"] \arrow[ur, "j"'] &
\end{tikzcd}$$

(c)
$$\begin{tikzcd}
& \mathrm{id} \arrow[dl, shift right=2, "\pi_1^*"'] \arrow[dr, "i"] & \\
O_X \arrow[ur, "\pi_1"] \arrow[dr, "\pi_r"'] \arrow[rr, shift left=2, "\exists"] \arrow[rr, shift right=2, "\forall"'] & & O^+ \arrow[ul, shift right=2, "i^{-1}"'] \arrow[dl, shift left=2, "j^{-1}"] \arrow[ll, "\Delta"'] \\
& \mathrm{no} \arrow[ul, shift left=2, "\pi_r^*"] \arrow[ur, "j"'] &
\end{tikzcd}$$

Figure 13: 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 **O**. O_X is id $\times_{O^+}$ no, O^+ is mark $+_{\mathrm{id}+\mathrm{no}}$ decision.

Querying can be done in at least three ways, either a) as a predicate on the $\exists$ arrow in category **CCC** of Figure 12, b) as a predicate on the arrow χ_J of Figure 11, or c) as an expression in the Heyting logic. For case a) the SQL query as Expression 1 in Section 5.1, building on the general form in Expression 3, can be expressed in terms of adjointness as:

$$\pi_l((\sigma_{\text{target.}\mathbf{O}\text{.decision} = \text{'P'} \wedge \text{source.}\mathbf{M}\text{.title} = \text{'Programming'}}{}^{\exists}) \dashv \Delta) \tag{6}$$

For case b) the terminal object of the category $\mathbf{1}_{\mathrm{CCC}}$ is an object in the subobject classifier. The arrow χ_J mapping from the object X to the truth value Ω is the logic of the SQL expression and the subobject U is the result of the query. As the query involves more than one object, it is necessary to extend the objects to powerobjects in Figure 11. This is the subject of further work as is case c), the exploration of the Heyting logic.

9 HANDLING REQUIREMENTS WITH THE VARIOUS APPROACHES: DISCUSSION

Figure 5 shows how the various approaches handle a particular information systems feature. For the standard models, the information is gleaned from standard texts [8, 7, 10]. For the CCC approach, the positive information is derived as follows:

- Structuring capability from Section 7.3
- Searching from Section 7.3
- Query symmetry from Section 7.3
- Transactions from earlier work by the authors [22]. Transactions are considered as processes, defined by the application of monads and comonads to the topos: the adjoint three-levels match the cyclic nature of database transaction structures where a natural closure is sought over three-levels of activity.
- Interoperability from earlier work by the authors [38]. Each CCC represents an intension-extension pair with a contravariant mapping from one component to another. Such pairs can be defined at varying levels of abstraction from the highest level representing a philosophical viewpoint of the data to the lowest representing the data values themselves. The integrity of an approach is preserved by functors linking the levels. Interoperability is achieved through natural transformations comparing the functors in one approach with those in another.

The additional positive information for the topos approach, beyond that for the CCC approach, is derived as follows:

- Searching from Sections 7.4–7.4
- Query closure from Section 7.4

Looking at Table 5 the relational model handles the requirements most fully of the standard data models. The Cartesian Closed Category approach matches the relational model in all respects and additionally handles the interoperability requirement. The topos approach improves on the CCC approach by providing additional features for searching and by facilitating query closure. Indeed the topos approach alone handles all of the listed requirements.

Approach (Standard)	Maths structure	Structuring Capability	Searching	Query symmetry	Query Closure	Transactions	Interoperability	Commercial Examples
Hierarchical	Trees	1:N, constraints	Tree traversal	No, bias downwards	No, tabular display	Yes, CICS	No	IMS
Network (ODMG)	Graphs	N:M, constraints	Graph traversal	No, bias to set paths	No, tabular display	Yes	No	IDMS
Relational (SQL4)	Relations of sets	N:M, constraints	Set operations ((SQL)	Yes	Yes	Yes	No	Oracle, DB2, Access
Extended Entity Relationship	Graphs	N:M, constraints	None	N/A	N/A	No	No	None
Object oriented (ODBT, XMI, XQUERY)	Graphs	N:M, constraints	Object query OQL)	Yes	No	Yes	No	Object DB
Functional	Functions	1:N, constraints	Function composition	No, bias to set paths	No	No	No	None
Sketch	Graphs	N:M, constraints	Function composition	No, bias to set paths	No	No	No	None
Spivak's approach	Graphs	N:M, constraints	Function composition	No, bias to set paths	No	No	No	None
Cartesian Closed Category	Pullbacks	N:M, constraints	Quantifiers	Yes	No	Yes, monad	Yes, natural	None yet
Topos	Categorical Topos	N:M, constraints	Quantifiers, subobject, Heyting	Yes	Yes, reflective subtopos	Yes, monad	Yes, natural	None yet

Table 5: Capability of Data Modelling Techniques for Handling Database System Requirements.

However, further work has been identified in this paper. More detail is sought on queries involving more than one object, where the extension in the subobject classifier from objects to powerobjects is required. Some examples of the Heyting internal logic are required both to show its scope and any limitations.

Database design, where complex networks of related objects are constructed, also requires further work. One idea is to employ the co-cartesian dual of the topos for this purpose. Nesting of pullbacks is another possible solution, giving a recursive approach. Pasting of pullbacks is adopted in many category theory textbooks but beyond very simple examples, this approach offers limited merits in visualisation and the re-use of names gives complex name hierarchies. Normalisation, a technique employed in relational database design, is considered to be a patch necessary for the set-based world, whose mimicry in topos design would be categorification. The use of Freyd & Scedrov's generalisation of relations, namely allegories, will be considered as a possible supplement to the pullback approach.

A database design method based on metaphysics is sought; for instance design rules such as a lecturer cannot teach two topics simultaneously is based on the physical restriction that a lecturer cannot be in two places at the same time. Ultimately database design rules should hold up to the natural isomorphism of the laws of metaphysics.

The further work identified above is more an indication of the obvious channels for additional study to work out the potential and limitations in more detail, rather than any fundamental weaknesses in the topos approach. Indeed the topos solution, alone of all the approaches, satisfies all of the requirements for a database system. A paramount consideration is that the adjointness of the topos solution provides a unique solution, critical in information systems.

10 CONCLUDING REMARKS

Process is a pervasive theme of ANPA and we have seen here that information systems within Information & Communication Technologies is a process. The software engineering in computer science requires a structural design process – the database – which is much more than a mere databank. The key feature of a database as a mediation process between the real world and a computer architecture is naturalness. Natural Philosophy suggests that naturalness requires a formal metaphysics of the calibre of category theory. For while mathematics is a formal language, computer science is a formal process.

11 ACKNOWLEDGEMENTS

We are grateful for encouragement from Michael Brockway and Paul Vickers, Department of Computer Science and Digital Technologies, Northumbria University, in the use of the topos in visualisation [42].

12 BIBLIOGRAPHY

[1] Adámek, Jiří, Herrlich, Horst; Strecker, George E, Abstract and concrete categories, John Wiley (1990). Recent edition at `http://katmat.math.uni-bremen.de/acc` (2005).

[2] Baclawski, K, & Simovici, D, An algebraic approach to databases with complex objects, Information Systems **17**(1):33-47. Pergamon Press. (1992).

[3] Baez, John C, & Stay, Mike, Physics, Topology, Logic and Computation: A Rosetta Stone, arXiv:0903.0340 [quant-ph] (2009).

[4] BCS Report: A Study in Project Failure, McManus, John, & Wood-Harper, Trevor, `http://www.bcs.org/content/ConWebDoc/19584`.

[5] Borceux, F, Handbook of Categorical Algebra. 3. Categories of Sheaves. Volume 52 of Encyclopedia of Mathematics and its Applications, Cambridge (1994).

[6] Chen, Peter, Entity-Relationship Modeling: Historical Events, Future Trends, and Lessons Learned, in: Software Pioneers, Springer-Verlag 296-310 (2002).

[7] Codd, E F, The Relational Model for Database Management, Addison-Wesley (1990).

[8] Connolly, Thomas M, & Begg, Carolyn E, Database Systems: A Practical Approach to Design, Implementation and Management, 6th edition (2014).

[9] Crane, L, & Frenkel, I, Four-dimensional topological quantum field theory, Hopf categories, and the canonical bases, J Math Phys **35** 5136-5154, arXiv: hep-th/9405183 (1994).

[10] Date, C J, An Introduction to Database Systems (8th Edition), Addison-Wesley (2004).

[11] Diskin, Zinovy, Kadish, Boris, Piessens, Frank & Michael Johnson: Universal Arrow Foundations for Visual Modeling. In Michael Anderson, Peter Cheng & Volker Haarslev, editors: Diagrams, Lecture Notes in Computer Science 1889, Springer, pp. 345-360 (2000). `http://link.springer.de/link/service/series/0558/bibs/1889/18890345.htm`.

[12] Diskin, Zinovy, & Maibaum, Tom, Category Theory and Model-Driven Engineering: From Formal Semantics to Design Patterns and Beyond, Proceedings of ACCAT 2012, EPTCS 93, U. Golas, T. Soboll (Eds.), 1-21 (2012). `http://arxiv.org/pdf/1209.1433.pdf%3Forigin%3Dpublication_detail`

[13] Ehresmann, Charles, Introduction to the theory of structured categories, Technical Report 10, University of Kansas (1966)

[14] Ehresmann, Charles, Catégories structurées généralisés, Cahiers de Topologie et Géométrie Différentielle, **X** 139-168 (1968).

[15] Ehresmann, Charles, Esquisses et types des structures algébriques, Buletinul Institutului Politehnic Iaşi XIV(18) (fasc. 1-2): 1-14 (1968).

[16] Einstein, Albert, Einfluss der Schwerkraft auf die Ausbreitung des Lichtes (On the Influence of Gravity on the Propagation of Light), Annalen der Physik **35** 898-908 (1911). `http://www.relativitybook.com/resources/Einstein_gravity.html`, `http://www.physik.uni-augsburg.de/annalen/history/einstein-papers/1911_35_898-908.pdf`.

[17] Einstein, Albert, Grundlage der allgemeinen Relativitätstheorie (The Foundation of the General Theory of Relativity) Annalen der Physik **49** 769-822 (1916). `http://www.physik.uni-augsburg.de/annalen/history/einstein-papers/1916_49_769-822.pdf`,

https://www.marxists.org/reference/archive/einstein/works/1910s/relative/relativity.pdf.

[18] Freyd, P, Abelian Categories: an introduction to the theory of functors, Harper & Row (1964). http://www.emis.de/journals/TAC/reprints/articles/3/tr3.pdf.

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

[20] Goldblatt, Robert, Topoi: The Categorial Analysis of Logic. Dover Books on Mathematics (1984).

[21] Heather, Michael, & Rossiter, Nick, Universal Logic applied to a Defeasible World, Perspectives on Universal Logic, Jean-Yves Béziau & Alexandre Costa-Leite (edd), Polimetrica 279-295 (2007). http://nickrossiter.org.uk/process/15_Rossiter.pdf

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

[23] Heather, M, & Rossiter, N, The Topos of Category Theory and Reality, Proc XVII International Conference on the Science and Quality of Life, June 29 - July 2 2013, Vilnius, Lithuania, edited Romuald Brazis, Studium Vilnense A 11 204-213 (2014). http://nickrossiter.org.uk/process/204-213_Heather_17icsq_StudiumVilnense_vol11.pdf

[24] Herrlich, H, & Strecker, G E, Category Theory, Allyn Bacon, Boston, 1973, 2nd ed. Heldermann, Berlin (1979).

[25] Johnson, M, & Rosebrugh, R, Sketch Data Models, Relational Schema and Data Specifications, ENTCS, **61**(6) 1-13 (2002).

[26] Johnson, M, Rosebrugh, R, & Wood, R J, Entity-relationship-attribute designs and sketches, Theory and Applications of Categories **10** 94-112 (2002).

[27] Johnson, Michael, & Rosebrugh, Robert, Implementing a Categorical Information System, J. Meseguer and G. Rožsu edd. AMAST 2008, LNCS **5140** 232-237 (2008). http://comp.mq.edu.au/~mike/papers/66.pdf

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

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

[30] Lo, C-Y, On Criticisms of Einstein's Equivalence Principle, Physics Essays **16**(1) (2003).

[31] Mac Lane, Saunders, & Moerdijk, Ieke, Sheaves in Geometry and Logic: a first introduction to topos theory. Corrected reprint of the 1992 edition, Universitext, Springer-Verlag, New York (1994). `http://ncatlab.org/nlab/show/Sheaves+in+Geometry+and+Logic`

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

[33] Pettigrew, Richard, An introduction to toposes, Department of Philosophy, University of Bristol `http://www.mcmp.philosophie.uni-muenchen.de/students/math/toposes.pdf`.

[34] Rosebrugh, R, & Wood, R J, Relational Databases and Indexed Categories, Proceedings of the 1991 Category Theory Meeting, published by the AMS. `http://www.mta.ca/~rrosebru/articles/rdic.pdf`

[35] Rosebrugh, Robert, Entity Attribute Sketch Implementation Kit, Mount Allison University `http://mathcs.mta.ca/research/rosebrugh/Easik/`

[36] Rossiter, N, From Classical to Quantum Databases with Applied Pullbacks, 78th Meeting Peripatetic Seminar on Sheaves and Logic, Institut de Recherche Mathématique Avancée, Strasbourg University 15-16 February (2003). `http://nickrossiter.org.uk/process/seminarstrasbourg15feb2003-1.pdf`

[37] Rossiter, Nick, & Heather, Michael, The Natural Metaphysics of Computing Anticipatory Systems, IJCAS **23** 3-20 (2011). `http://nickrossiter.org.uk/process/liege09metaphys.pdf`

[38] Sisiaridis, Dimitris, Heather, Michael, & Rossiter, Nick, The Contravariancy of Anticipatory Systems, 8th BCSCMsG International Symposium on Computational Self-Organised Emergence, organised Peter J. Marcer, The British Computer Society Cybernetic Machine specialist Group, IJCAS, Edited D M Dubois **27** 181-202 (2014). http://nickrossiter.org.uk/process/27-2-2-101-sisiaridis-heather-rossiter-CASYS11-IJCAS-21pp.pdf

[39] Spivak, David I, Databases are categories, Presented on 2010/06/03, Galois Connections, Mathematics Department, University of Oregon (2010) http://math.mit.edu/~dspivak/informatics/talks/galois.pdf

[40] Spivak, David I, Categorical databases, Mathematics Department, Massachusetts Institute of Technology, Presented on 2012/01/13 (2013). http://math.mit.edu/~dspivak/informatics/talks/CTDBIntroductoryTalk

[41] Spivak, David I, Category Theory for the Sciences, MIT Press (2014); also see draft Category Theory for Scientists, September 19, 2013, Section 3.5 Databases: schemas and instances. http://arxiv.org/pdf/1302.6946.pdf

[42] Vickers, Paul, Faith, Joe, & Rossiter, Nick, Understanding Visualization: A Formal Approach using Category Theory and Semiotics, IEEE Transactions On Visualization And Computer Graphics 158. IEEE Transactions On Visualization And Computer Graphics, 2012 Jun;19(6):1048-61. doi: 10.1109/TVCG.2012.294 (2013).

[43] Weaver, Warren, & Shannon, Claude Elwood, The Mathematical Theory of Communication, University Illinois Press (1963).

[44] Wells, Charles, Sketches: Outline with References, 1993 with addendum 2009, http://www.cwru.edu/artsci/math/wells/pub/pdf/Sketch.pdf (2009).

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