MONADIC DESIGN FOR UNIVERSAL SYSTEMS

NICK ROSSITER, MICHAEL HEATHER
MICHAEL BROCKWAY

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

nick.rossiter1@btinternet.com; michael.heather@trinity.cantab.net;
michael.brockway@unn.ac.uk
http://www.nickrossiter.org/process/

Abstract: The work described here builds on recent work presented at ANPA on structure and process in the universe. The internal structure of the topos is explored further with particular emphasis on the nature of the pasted pullback, including the conditions for a pasting to be valid and the inherent recursive nature of pullback structures. Dolittle diagrams are employed for representing the intension/extension relationships. A banking example is explored, leading to the nature of the external processes acting upon the topos such as transactions. These processes are represented by monads, giving a three-level closure on the activity. The nature of monads is explored. The T-algebra enables changes to be made in the monad structure, giving the potential for adaptability. Monads, that have been strengthened by the Kleisli lift to the Cartesian form, can be composed naturally, facilitating the construction of large-scale information systems with reliability, as required for transactions in the banking world.

1 INTRODUCTION

The fundamental categorical facilities identified for a Universe, whether from any Universe of Discourse up to the Universe, is primarily the Topos as a structural data-type including the Monad for process. The application of the monad to a topos gives the operation of a process on data at the highest level, defined as a unique solution up to natural isomorphism. We will demonstrate such an application, explore how its performance relates to alternative techniques and discuss further work required.

Succinctly the topos is a fundamental Cartesian Closed Category (CCC), a category with limits and exponentials subobjects. A CCC has an internal logic of the typed λ-calculus, an identity functor and the interchangeability of levels, with nodes being either objects or categories. A topos has additional properties beyond a CCC ([19], at p.106) including a subobject classifier, the internal logic of Heyting, that is intuitionistic logic, and a reflective subtopos as a category for recursive query closure. A topos is closed at the top but open at the bottom. An important issue is the flavour of category theory that is being employed. The starting point is what might be termed Eilenberg/-Mac Lane category theory (EML) as set out in pure mathematical terms by Saunders Mac Lane in his book *Categories for the Working Mathematician* [19]. The employment of category theory in metaphysics and the implementation of the theory in programming languages has shown some limitations to EML, in particular with the distributive law, resulting from a restricted view of naturality. Many of these points are discussed in detail later.

The application of the topos to data was established in papers at ANPA 35 [29], introducing the topos/monad approach, and ANPA 36 [30], bringing out the interoperable use of allegories for legacy relational systems. Structures developed as a topos include pasted pullbacks, to represent relatedness, and recursion in which any juncture in the structure is a pullback in its own right. The exact nature of the match, in the pasting operation, is discussed later in Section 4.2. The relationship of the topos structures to Fifth Normal Form (5NF) [12], also known as PJNF (Project-Join Normal Form), is of relevance. A working definition of PJNF is that a table T satisfies the normal form if it cannot have a lossless decomposition into any number of smaller tables. The decomposition is a projection into tables with smaller numbers of columns; lossless implies that the join of the resulting tables returns the original table T unchanged. The pullback structure has an inherent feature for handling PJNF: the adjointness between the projections π from the product onto the coproduct and the diagonal join Δ from the coproduct onto the product. This indicates the strength of the topos data structuring method for handling a challenging ultimate stage in relational database design. Other less powerful normalisation techniques are considered to be so set-based that any categorial approach would be categorification. The Cocartesian dual to the topos may offer further insights into the data structuring process: the mapping from PRE to POS is effectively the data normalisation process in relational databases, where PRE is the preorder with no initial object and POS is the partial order with no cycles and both an initial and a terminal object, satisfying the closed world assumption (CWA). It should be emphasised though that the topos is a

more general data structure than a relational database, which while closed at the top is open at the bottom, thus not restricted by the CWA.

The use of the allegories of Freyd [7] as a basis for data structures was attempted [30] but rejected because of their lack of naturality as set-based relations; the allegories will have use though in interoperability as a wrapper for legacy relational databases. Internal queries on a topos are handled by the subobject classifier, which may be Boolean (0 or 1) or the more general double powerobject. Both forms were illustrated in the two ANPA papers cited above. The provision of examples of Heyting intuitionistic logic for an application remains an objective. Internal queries are more akin to data searches, such as through Google, but do not provide a well-defined process capability.

The application used was of grading for students by modules in a university context, which was adequate from the data structure viewpoint but limited from a data process angle. A more interesting application from the process perspective is banking, including the handling of transactions. This was first studied by us in ANPA 27 [28].

Monadic design is a novel technique for handling the dynamic aspects of an application. Aspects to be investigated are the adjointness, inherent in the approach, the flavours of monad which are most suited to process applications and the T-algebra for modifying the adjunction.

The intention in this paper is therefore to introduce a new application, banking, which provides a more suitable test for an external process of a monad on a topos data structure. The mechanism of pasting is to be investigated in detail and the relationship of the topos to database normalisation is to be clarified. Monadic design will be developed for the topos.

2 PULLBACK: SINGLE RELATIONSHIP FOR STUDENT GRADING

The topos has limits and the pullback is a limit. Figure 11.1 shows for the student application, studied in the ANPA 35/36 papers, a simple pullback of assessment: with limit $\mathbf{S} \times_{\mathbf{G}} \mathbf{M}$, the product of Student and Module in the context of Grading. The relationship between the product $\mathbf{S} \times_{\mathbf{G}} \mathbf{M}$ and $\mathbf{G}$ is adjoint, with the following logic condition holding: $\exists \dashv \Delta \dashv \forall$. The functor Δ selects pairs of $\mathbf{S}$ and $\mathbf{M}$ in a relationship in the context of $\mathbf{G}$, such that $\exists$ is left adjoint to Δ and $\forall$ is right adjoint to Δ, with a consequential facility for consistent logical operation. The arrows can be interpreted as follows: π_1

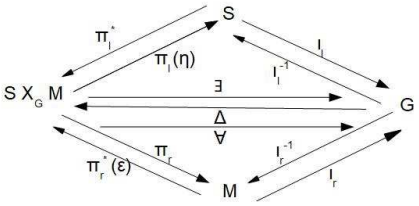

Figure 11.1: Pullback for a Single Relationship $\mathbf{S} \times_{\mathbf{G}} \mathbf{M}$: Grading by Student and Module; categories are **S** Student, **M** Module, **G** Grading

identifies the participating students, π_r the modules taken, ι_l is candidature, and ι_r is marking.

A diagram with such adjointness was termed by Lawvere a hyperdoctrine in the early days of EML category theory [14]. The existential functor $\exists$ records the decision for each student (under the free functor F) of a grade for a specific module. The diagonal Δ sorts student and modules as a component of the underlying functor G. Thr universal functor $\forall$ produces the final grading list generated by F. This shows the fine structure of the adjointness $F \dashv G$.

Other arrows are interpreted as follows. Projections π are from the product onto its constituents, left π_l and right π_r, with dual arrows left π_l^* for student capability, achievement and creativity and right π_r^* for quality of work, respectively. Inclusions ι are into the sum $\mathbf{S} + \mathbf{G} + \mathbf{M}$ from its constituents, left ι_l and right ι_r, providing type checks on the values for students and modules, respectively, participating in the relationship. The dual arrows, ι_l^{-1} and ι_r^{-1}, indicate the participation of students and modules, respectively, in the relationship.

2.1 CLOSURE TO RECURSION: DOLITTLE DIAGRAMS FOR INTENSION/EXTENSION

S, **M**, **G** are each categories, with an optional internal pullback structure, giving a recursive pullback structure with potential unlimited depth. The closure to recursion is that the diagrams at the bottom level are of the Dolittle[1] type

1 named after *The Story of Doctor Dolittle* by Hugh Lofting (1920), involving the mythical Pushmi-pullyu creature.

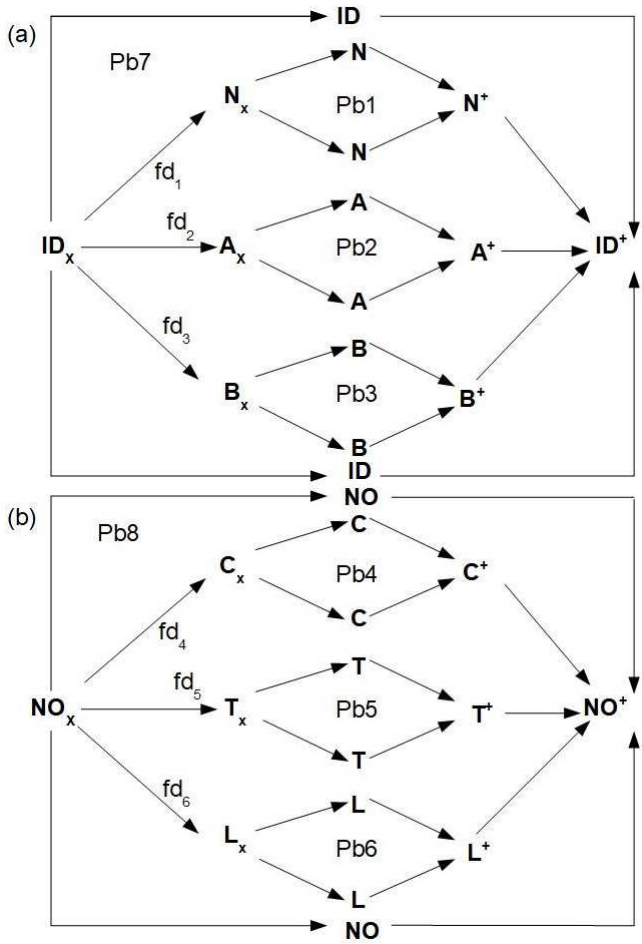

Figure 11.2: a) Dolittle diagram for category **S** for Student with outer diagram Pb7 of limit $\mathbf{ID}_\times$ and colimit $\mathbf{ID}^+$ (Identifier); inner diagrams of Pb1 with limit $\mathbf{N}_\times$ and colimit $\mathbf{N}^+$ (Name), Pb2 with limit $\mathbf{A}_\times$ and colimit $\mathbf{A}^+$ (Address), Pb3 with limit $\mathbf{B}_\times$ and colimit $\mathbf{B}^+$ (Birthdate).

b) Dolittle diagram for category **M** for Module with outer diagram Pb8 of limit $\mathbf{NO}_\times$ and colimit $\mathbf{NO}^+$ (Number); inner diagrams of Pb4 with limit $\mathbf{C}_\times$ and colimit $\mathbf{C}^+$ (Course), Pb5 with limit $\mathbf{T}_\times$ and colimit $\mathbf{T}^+$ (Title), Pb6 with limit $\mathbf{L}_\times$ and colimit $\mathbf{L}^+$ (Level).

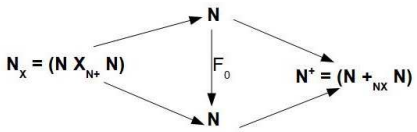

Figure 11.3: Dolittle Diagram for a Single Relationship: Student Names with limit $N_X = \mathbf{N} \times_{\mathbf{N}+_{N_X}\mathbf{N}} \mathbf{N}$, colimit $N^+ = \mathbf{N} \times_{\mathbf{N}\times_{\mathbf{N}^+}\mathbf{N}} \mathbf{N}$, top object **N** of type 1-category, bottom object **N** of type 0-category, F_0 maps 1-category to 0-category.

with a mapping from the data in the left-hand pullback object, a qualified product or $\times$, to equivalent data in the right-hand pushout object, a qualified coproduct or $+$ [9]. Such diagrams are also called Bicartesian squares [2] or pulation squares ([1] pp.205-206). Bicartesian squares as defined by Banach are regular relations, a particular type of pullback in which there is a N:1 relationship between the objects at the top and the bottom, rather than the N:M in the more general case. The special case of a 1:1 relationship would also satisfy the Bicartesian requirement. We agree with Lambek & Scott ([13] pp.65-68) that the two objects are different in purpose, with for instance the top object, the independent variable, being the intension (definition) and the bottom object, the extension (values). There is a functional relationship between the top and bottom objects in a Dolittle diagram, resulting in an ordered product as a string. A helpful view from n-categories is that the top object is a 1-category, defining the data structure in terms of objects and arrows, and the bottom object is a 0-category, holding data values as discrete objects. The arrow from the 1-category to the 0-category is then a 0-functor [25], written as F_0, mapping intension to extension. Everything is related implicitly in a $+$ context; the relationship in a $\times$ context is stronger with explicit connections. Relatedness in Heyting logic is expressed by the condition: $C \times A \leqslant B$ is isomorphic with $C \leqslant A \Rightarrow B$. Such a view of relatedness is not available in set theory.

The Dolittle categories **S** and **M** are shown in Figures 11.2(a) and (b) respectively. Each of these categories comprises a number of further Dolittle diagrams, one for each type of data. Each category has a key or identifier, which provides unique identification of other data properties. The key for the category for student **S** is ID (identifier) and the data properties are N (name), A (address), B (birthdate). Each data property is a Dolittle diagram representing

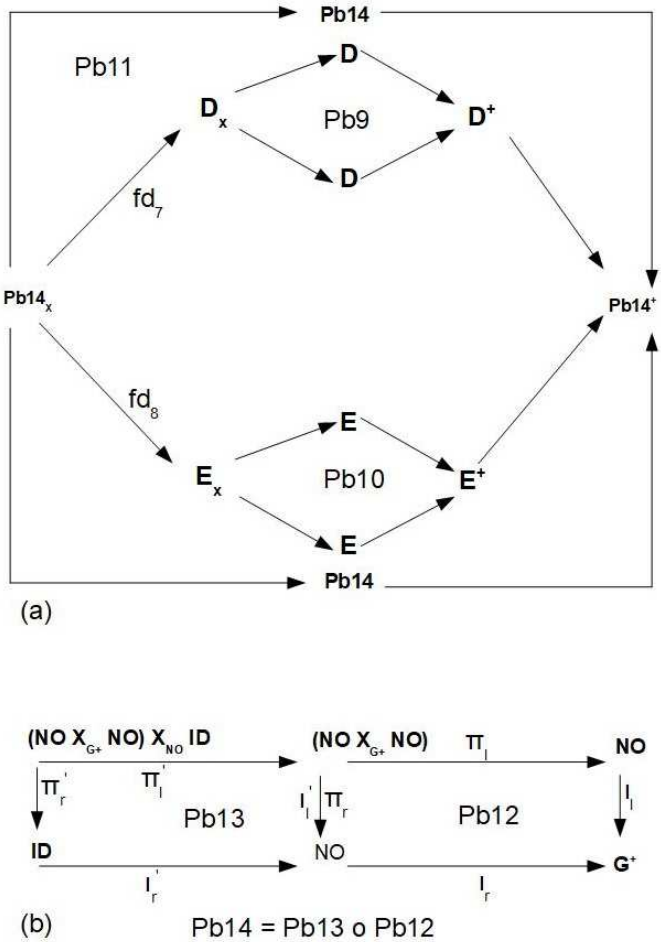

Figure 11.4: a) Dolittle diagram for category **G** with outer diagram Pb11 of limit $\mathbf{Pb14}_{\times}$ and colimit $\mathbf{Pb14}^{+}$; inner diagrams of Pb9 with limit $\mathbf{D}_{\times}$ and colimit $\mathbf{D}^{+}$ (Date of decision), Pb10 with limit $\mathbf{E}_{\times}$ and colimit $\mathbf{E}^{+}$ (Exam board);
b) Pasted pullback diagram for internal key structure for **G**; Pb12 is outer Dolittle Pb8 in Figure 11.2(b); Pb13 is pasting of outer Dolittle in Figure 11.2(a) onto Pb12; Pb14 is composition of Pb13 with Pb12.

the intension-extension relationship as in the diagram shown in Figure 11.3 for the property Name of Students labelled Pb1 in Figures 11.2(a). The limit of this diagram $\mathbf{N} \times_{\mathbf{N}^+} \mathbf{N}$, written shorthand as $\mathbf{N}_\times$, is a product of the intension $\mathbf{N}$ of 1-category type and extension $\mathbf{N}$ of 0-category type in the context of the colimit $\mathbf{N} +_{\mathbf{N}_\times} \mathbf{N}$, written shorthand as $\mathbf{N}^+$. These shorthand abbreviations are used throughout the paper to improve readability. The 0-functor F_0 maps the 1-category intension to the 0-category extension. The colimit identifies, for a pair of intension/extension values in the limit, the values for the names as a sum in the colimit.

Simple diagrams such as Figure 11.3 are the building blocks at the bottom level for the two categories shown in Figures 11.2. The data properties are determined by the key through the functional dependency arrows. For instance the functional dependency $\mathtt{fd}_1 : \mathrm{ID} \longrightarrow \mathrm{N}$ indicates that for each ID, there is a unique N. The dependency is actually defined in the diagram with more detail as the arrow $\mathtt{fd}_1$ from an intension/extension pair for ID, $\mathbf{ID}_\times$, to an intension/extension pair for N, $\mathbf{N}_\times$. There are two further functional dependencies $\mathtt{fd}_2 : \mathbf{ID}_\times \longrightarrow \mathrm{A}_\times$ and $\mathtt{fd}_3 : \mathbf{ID}_\times \longrightarrow \mathrm{B}_\times$ in our diagram. The inner Dolittle diagrams Pb1, Pb2 and Pb3 represent the data properties N, A and B respectively. Completing Figure 11.2(a), there is an outer Dolittle diagram Pb7 with limit of $\mathbf{ID}_\times$ and colimit of $\mathbf{ID}^+$, representing the intension/extension for the key ID. The internal structure of the data properties Name, Address and Birthdate, may be complex, with for instance a name being subdivided into title, first name, middle name, last name. Further there is no obligation for a value in the extension to be an atomic value as in the relational model [12]; the value could be a structured set or a complex object, as in the nested relational and object-oriented approaches respectively.

Figure 11.2(b) is another example of a Dolittle diagram with a 0-functor. The key is NO (module number), forming the outer pullback Pb8 with limit $\mathbf{NO}_\times$ and colimit $\mathbf{NO}^+$. The data properties, represented by the inner pullbacks, labelled Pb4, Pb5 and Pb6, are C (Course), T (Title of module) and L (Level), respectively. The functional dependencies are $\mathtt{fd}_4 : \mathbf{NO}_\times \longrightarrow \mathrm{C}_\times$; $\mathtt{fd}_5 : \mathbf{NO}_\times \longrightarrow \mathrm{T}_\times$; $\mathtt{fd}_6 : \mathbf{NO}_\times \longrightarrow \mathrm{L}_\times$.

The category for student grading by module $\mathbf{G}$ is more complex in that its identifier is a combination of the student identifier ID and the module number NO, necessary to define uniquely a grading. That is the key cannot be described simply as an unqualified product: it needs to be factored through the grading to represent the pairs that actually occur. This requires a pasted[2]

2 pasting is described further in Section 3.2 and discussed in Section 4.

pullback as shown in Figure 11.4(b). The inner pullback Pb12 is the same as the outer Dolittle Pb8, representing the key structure in category **M** in Figure 11.2(b). The outer pullback Pb13 is a pasting of the key structure in category **S**, in Figure 11.2(a), on to Pb12. The pasted key structure is not a Dolittle diagram in itself but is employed in the top and bottom nodes of the outer Dolittle diagram Pb11 in Figure 11.4(a). Pb14 is the overall result from composing Pb12 with Pb13, that is the whole diagram in Figure 11.4(b). The Dolittle diagram for the category **G** is shown in Figure 11.4(a). For the outer Dolittle diagram Pb11 the top object is the intension for the key Pb14 and the bottom object the extension for the key Pb14. The inner Dolittle diagrams, labelled Pb9 and Pb10, represent D (Date of decision) and E (Exam board) respectively with functional dependencies $\mathtt{fd}_7 : \mathtt{Pb14}_\times \longrightarrow \mathtt{D}_\times$ and $\mathtt{fd}_8 : \mathtt{Pb14}_\times \longrightarrow \mathtt{E}_\times$ respectively.

3 BANKING EXAMPLES

We now introduce the Banking example, which is a more suitable subject for illustrating the action of process on a topos as a core feature of banking is the use of transactions for handling changes in a secure manner. We commence with a simple data example, gradually making it more realistic.

3.1 PULLBACK: SINGLE RELATIONSHIP

The simple pullback is shown in Figure 11.5, defined as $\mathbf{P} \times_{\mathbf{T}} \mathbf{A}$, that is the product of Procedure and Account in the context of Transaction, with **P** the category Procedure, **A** the category Account, and **T** the category Transactions. An Account can belong to many users; the Procedure is the type of the transaction, for example: standing order, direct debit, ATM cash withdrawal; the transaction is a transfer of funds according to data processing requirements. **P**, **A**, **T** are categories, with internal pullback structure, giving in general a recursive structure with closure at the Dolittle level, as in Figures 11.2, 11.4 for the student example.

3.2 PULLBACK: TWO PASTED RELATIONSHIPS

In pasted pullbacks two relations are joined together to form a square. An additional category is introduced for User (customer) of **U**. Each user may

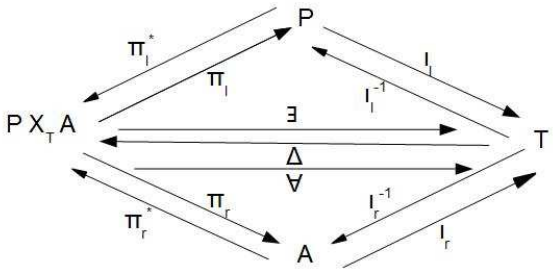

Figure 11.5: Pullback - Single Relationship $\mathbf{P} \times_{\mathbf{T}} \mathbf{A}$: Bank Transactions by Procedure and Account; categories are **P** Procedure, **A** Account, **T** Transaction

have multiple accounts across the banking network: there is a many-to-many (N:M) relationship between **U** and **A**. The second pullback is the product of the subproduct of the first pullback $\mathbf{P} \times_{\mathbf{T}} \mathbf{A}$ with **U** in the context of **A**, as shown in Figure 11.6. The resulting relationship is of account transactions by users. For the purpose of discussion, the pullbacks can be labelled Pb1 for the first square $\mathbf{P} \times_{\mathbf{T}} \mathbf{A}$ and Pb2 for the second square $(\mathbf{P} \times_{\mathbf{T}} \mathbf{A}) \times_{\mathbf{A}} \mathbf{U}$). By EML category theory ([19] pp.71-72) if the squares Pb1 and Pb2 are valid pullbacks, then the whole outer square is also a pullback Pb2 × Pb1. We therefore have three pullback diagrams in a valid pasted relationship.

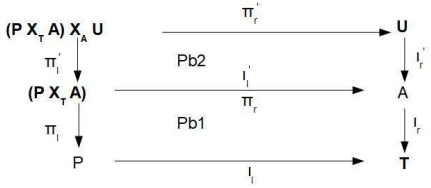

Figure 11.6: Pullback: Two Pasted Relationships: Bank Transactions by Account by User, in Portrait Layout

The vertical stacking of the pasted pullbacks, one above the other, in portrait form is suited to practical applications which could involve 5-10 relation-

ships in a deep nested structure. In EML category theory text books, pasted structures are usually written in horizontal (landscape) form as in Figure 11.7, which is logically identical to that in Figure 11.6.

The aim of pasting in topology is to 'glue together' two continuous functions to create another continuous function. The specific pasting condition for the pullback Pb2 × Pb1 is that $\iota'_l = \pi_r$ after Freyd's Pasting Lemma [7].

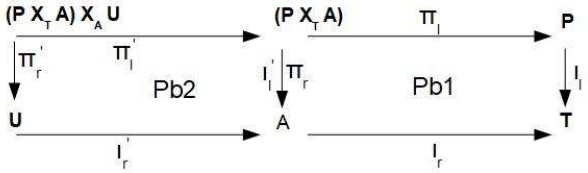

Figure 11.7: Pullback: Two Pasted Relationships: Bank Transactions by Account by User, in Conventional Landscape Layout

To make the application more realistic we add two further categories, those of **B** for Branch and **C** for (banking) Company. Branch:User is also a N:M relationship as each Branch has many Users and each User has many Branches but Company:Branch is a 1:N relationship: each Company has many Branches, each Branch is within one Company. The overall relationship is $(((\mathbf{P} \times_{\mathbf{T}} \mathbf{A}) \times_{\mathbf{A}} \mathbf{U}) \times_{\mathbf{U}} \mathbf{B})$ with **C** in the context of **B** giving the pullback diagram shown in Figure 11.8. The representation of N:M and 1:N relationships is the same in terms of pullback structures, giving a useful symmetry in data design.

Figure 11.8 involves six categories: **C** company, **B** branch, **U** user, **A** account, **P** procedure, **T** transaction, and ten pullbacks: Pb4, Pb3, Pb2, Pb1; Pb4 × Pb3, Pb3 × Pb2, Pb2 × Pb1; Pb4 × Pb3 × Pb2, Pb3 × Pb2 × Pb1, Pb4 × Pb3 × Pb2 × Pb1. The relations within a banking system are shown in more conventional form in Figure 11.9(a) where each single-headed arrow represents a 1:N (one-to-many) relationship and each double-headed arrow represents a N:M (many-to-many) relationship.

For our purposes, a pasted pullback is only a valid pullback if all inner and outer diagrams are pullbacks. There are some theorems in EML category theory ([19] pp.71-72) which enable some deductions to be made based on partial knowledge: for example, with the diagram in Figure 11.7, if the inner

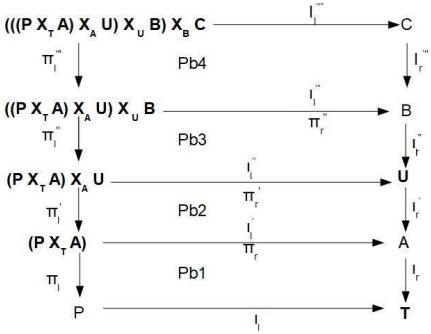

Figure 11.8: Pullback: Four Pasted Relationships: Bank Transactions by Account by User by Branch by Company

diagrams are pullbacks then the outer diagram is a pullback, as stated earlier, and if the outer diagram and the right-hand diagram are pullbacks then the left-hand diagram is a pullback. Such deductions could be facilitated in any practical system but are a distraction from developing a simple robust solution.

As an example of an invalid pullback, consider the diagram in Figure 11.10 where the relationship diagram has been modified to that in Figure 11.9(b). There are seven valid pullbacks in the diagram: Pb4, Pb3, Pb2, Pb1; Pb3 × Pb2, Pb2 × Pb1; Pb3 × Pb2 × Pb1, but not all squares are pullbacks, for example Pb4 × Pb2. Therefore the whole diagram is not a valid pullback.

For any valid pullback, the logic of adjointness holds for the outer square and all inner squares. Therefore for Figure 11.8 with its six valid pullback diagrams, the logic $\exists \dashv \Delta \dashv \forall$ holds across every diagram. An example of this logic is shown in Figure 11.11 for the outer square.

3.3 SUBOBJECT CLASSIFIER

As a pullback the pasted structure is a Cartesian Closed Category (CCC) with products, terminal object and exponentials. Further it is a topos as a CCC with subobject classifier and internal Heyting Logic. The subobject classifier provides an internal query language for which a Boolean example is shown in Figure 11.12.

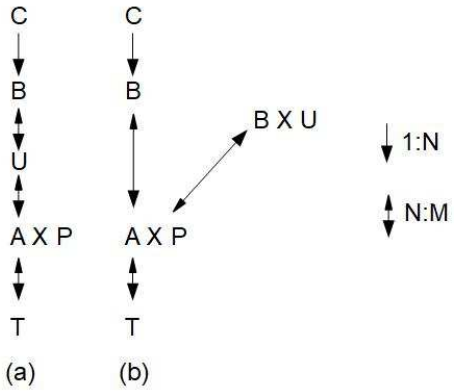

Figure 11.9: Relations within a Banking System corresponding to (a) Figure 11.8 and (b) to Figure 11.10. C is Company, B branch, U user, A account, P procedure, T transaction.

The subobject classifier facilitates simple database or information retrieval queries:

- $\Omega\{0,1\}$ is the subobject classifier with subobjects classified as either 0 or 1
- χ_j is the characteristic function, a query mapping from the object S to $\{0,1\}$, false or true
- 1_{topos} is the terminal object of the topos, giving a handle on the topos
- $\jmath$ is the mapping from the subtopos U, the result of the query, to the object S
- U is the identity of the subtopos, giving query closure

The diagram may be viewed as a pullback of *true* along χ_j, with U as $1_{topos} \times_{\Omega\{0,1\}} S$.

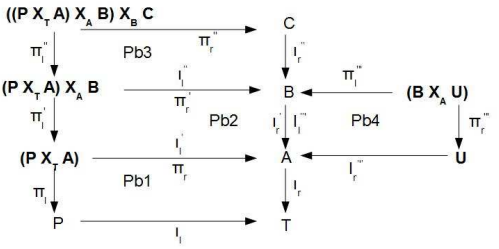

Figure 11.10: Invalid Pullback Diagram, corresponding to Relations in Figure 11.9(b)

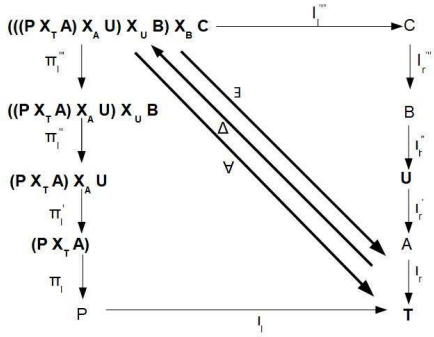

Figure 11.11: Adjointness Holds for all Pullbacks: $\exists \dashv \Delta \dashv \forall$

4 PASTING PULLBACKS: DISCUSSION

To summarise, in a pasted diagram, all pullbacks as inner or outer squares must commute for the diagram to be a valid pullback as a whole. The structure is recursive in that a pullback node may itself be a pullback diagram. Two aspects are worthy of further discussion: how does the pullback diagram relate to data normalisation in conventional data structuring and can the pasting condition be expressed in other forms, drawing out the nature of the '=' condition?

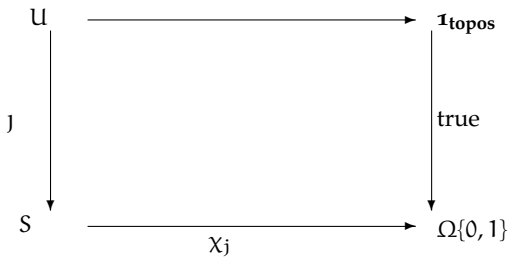

Figure 11.12: Pullback Square for Boolean Subobject Classifier: Definition of Characteristic Function $\chi_j : S \longrightarrow \Omega\{0, 1\}$

4.1 NORMALISATION

Normalisation is the standard technique for evaluating a data design, in particular to determine how closely the logical design matches the physical world. A number of stages have been developed for the set-theoretic relational model: 1NF (First Normal Form), 2NF, 3NF, BCNF, 4NF, 5NF. The last and most demanding stage 5NF concerns us here, not just for its rigour but for its definition in category theory terms, indicated by its alternative name of Project-Join Normal Form (PJNF).

In set theoretic terms, the definition of 5NF is that the structures resulting from the projections can be joined together to return the original structure without loss or gain of information [12]. Looking at the simple pullback diagram, as in Figure 11.5, the projections are the π arrows, π_l and π_r, and the join arrow is the diagonal Δ. PJNF holds through the adjointness in every pullback: $\exists \dashv \Delta \dashv \forall$. The arrows $\exists$ and $\forall$ involve the projections through the compositions: $\exists = \iota \circ \pi$ and $\forall = \iota \circ \pi$. In more complex data structures, the same logic applies. For instance in Figure 11.11 with ten pullback squares (including undrawn inner ones), PJNF will hold if the whole structure and all inner squares are pullbacks with the logic: $\exists \dashv \Delta \dashv \forall$. Surprisingly pullbacks have rarely been used in normalisation studies, an exception being the work of Levene & Vincent [17] who briefly mention the pullback inference rule, following from the interaction between functional dependencies $\exists$ and inclusion dependencies ι.

It should be emphasised that the pullback is not categorification of the set-theoretic approach to normalisation of 5NF, as in earlier work with category theory and databases [11]. The form 5NF was a belated move by set-theoretic

adherents to find a viable approach to normalisation after many earlier attempts had been only partially successful. The pullback follows basic category theory principles and is a *natural* choice for an effective data structure.

4.2 THE PASTING CONDITION

The Pasting Condition is $\iota_l' = \pi_r$, that is the left-inclusion of the outer square equals the right-projection of the inner square. On the surface this looks rather set theoretic, where the '=' would be without context, but in EML category theory the '=' is defined naturally as unique up to natural isomorphism, through the adjointness inherent in the pullback category.

Moreover any pullback can be represented as an equalizer [26], as in Figure 11.13, which is equivalent to Figure 11.5. In the equalizer diagram the product of **P** and **A** in the context of **T**, $\mathbf{P} \times_{\mathbf{T}} \mathbf{A}$, maps onto the product $\mathbf{P} \times \mathbf{A}$ which in turn maps onto **T** where the two paths, $\iota_l \circ \pi_l$ and $\iota_r \circ \pi_r$, converge.

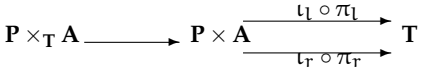

Figure 11.13: Pullback in Figure 11.5 Represented as an Equalizer

Equalizer diagrams can also be constructed for pasted pullbacks, as in Figure 11.14, which is equivalent to Figure 11.6. In the equalizer diagram the product of $\mathbf{P} \times_{\mathbf{T}} \mathbf{A}$ and **U** in the context of **A**, $(\mathbf{P} \times_{\mathbf{T}} \mathbf{A}) \times_{\mathbf{A}} \mathbf{U}$, maps onto the product $(\mathbf{P} \times_{\mathbf{T}} \mathbf{A}) \times \mathbf{U}$ which in turn maps onto **T** where the two paths, $\iota_l \circ \pi_l \circ \pi_l'$ and $\iota_r \circ \iota_r' \circ \pi_r'$, converge.

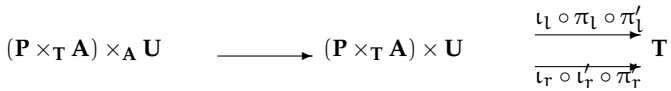

Figure 11.14: Pasted Pullback in Figure 11.6 Represented as an Equalizer

5 EXTERNAL PROCESS

The concept of process is underpinned by metaphysics, as defined in the writing of authors such as Alfred North Whitehead, in his book *Process and Reality* [31]. For any entity in the universe, the actions possible upon it and the rules for such actions are a critical part of the whole system. First we look at the technical features within category theory for representing process. We next look at the requirements for the real world and review the facilities of the theory that appear to be most relevant.

5.1 PROCESS IN CATEGORY THEORY

An internal process is a morphism (arrow) within a topos, such as $p : A \longrightarrow B$, where the process p takes object A to object B in the same topos. Such arrows play a natural role in the category construction. An external process is activity on a topos $\mathbf{E}$, taking it to another topos $\mathbf{E}'$, such as provided by a functor F with $F : \mathbf{E} \longrightarrow \mathbf{E}'$. Both $\mathbf{E}, \mathbf{E}'$ must conform to the natural rules for topos construction. Constraints on the transition between $\mathbf{E}$ and $\mathbf{E}'$ are enforced through adjointness between F ($\mathbf{E} \longrightarrow \mathbf{E}'$) and its dual G ($\mathbf{E}' \longrightarrow \mathbf{E}$), such that $F \dashv G$ and the 4-tuple $< F, G, \eta, \epsilon >$ exists where η is the unit of adjunction $\eta : 1_E \longrightarrow GFE$, ϵ is the counit of adjunction $\epsilon : FGE' \longrightarrow 1_{E'}$ and E, E' are objects in categories $\mathbf{E}$ and $\mathbf{E}'$ respectively. The pair of adjoint functors $F \dashv G$ may be written as T and the dual $G \dashv F$ as S.

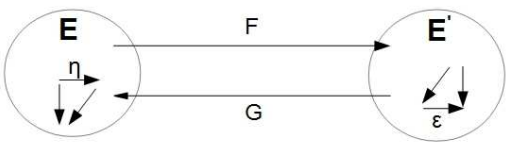

Figure 11.15: Multiple 'Cycles' $GFGFGF(T^3)$ for adjointness $< F, G, \eta, \epsilon >$

The cycle T can be enhanced by performing it three times, T^3, to achieve closure. Such a construction is termed a monad, with its dual S^3 a comonad. The functors and their constraints are illustrated in Figure 11.15. The monad

is a generalisation of the single-level monoid, which has a single operation, binary multiplication $M \times M \longrightarrow M$, and the identity $1 \longrightarrow M$, for an object M.

5.2 REAL-WORLD REQUIREMENTS

The process is represented in information systems by the transaction, which has been the subject of intense study because of its criticality to applications such as banking and internet-based commerce. However, the concept is a very general one, applying for instance to drafting where a transaction may last several days as a technical drawing is modified from one consistent state to another, or maybe months, as a legal document is modified similarly. The notion of transaction in a categorial context was developed in earlier ANPA papers, more generally at ANPA 31 [10], and in considerable detail at ANPA 27 [28]. The principles of the transaction are summarised as ACID: Atomicity, Consistency, Isolation, Durability. Atomicity ensures that the process, however complicated, is viewed as a single arrow. Consistency ensures that all rules have to be satisfied before the transition is made. Isolation ensures that any intermediate results in the process are not revealed. Durability ensures that once a transaction is performed, the results persist until changed by another transaction. The transaction is a logical technique for controlling the real world.

5.3 APPLICABILITY OF THE THREE CYCLES

A transaction is viewed naturally as three 'cycles' of adjointness [28]. The first cycle performs the actual work required; the second checks for any errors or inconsistencies resulting from the first cycle; the third cycle consolidates the changes made provisionally in the first cycle and checked in the second cycle. The 'cycles' are not separate stages; all three cycles are performed as a single snap: the prehension, or grasping, of Whitehead [31]. This single snap satisfies the atomicity and isolation requirements. The second cycle satisfies the consistency requirement, through review against the rules. The third cycle satisfies the durability requirement, through consolidating the results. If adjointness does not hold in any cycle, the transaction is abandoned. We now look at the application of the monad in more detail.

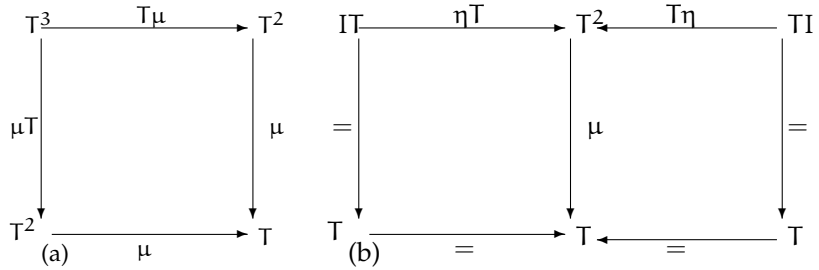

Figure 11.16: (a) Associative Law for Monad $< T, \eta, \mu >$; (b) Left and Right Unitary Laws for Monad $< T, \eta, \mu >$

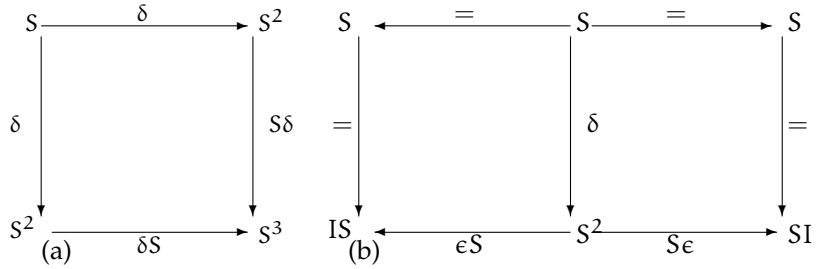

Figure 11.17: (a) Associative Law for Comonad $< S, \epsilon, \delta >$ (b) Left and Right Unitary Laws for Comonad $< S, \epsilon, \delta >$

5.4 TECHNICAL DETAILS OF THE MONAD APPROACH

For a monad, the diagrams for the associative laws and unitary laws are shown in Figure 11.16. These diagrams provide the formal basis for the approach. Figure 11.16(a) shows the relationship between T^3, T^2 and T where T is the endofunctor $GF : \mathbf{X} \longrightarrow \mathbf{X}$, $\mathbf{X}$ being any category. An endofunctor is a functor with the same source and target. A pair of adjoint functors F and G is an endofunctor as the source of $F : \mathbf{X} \longrightarrow \mathbf{Y}$ is $\mathbf{X}$ and the target of $G : \mathbf{Y} \longrightarrow \mathbf{X}$ is also $\mathbf{X}$. The unit or identity of the monad is $\eta : 1 \longrightarrow T$ from Figure 11.16(b) and the multiplication of the monad is $\mu : T^2 \longrightarrow T$ from Figure 11.16(a). We therefore write the monad T as the object $< T, \eta, \mu >$, with the category $\mathbf{X}$, on which the monad is based, omitted as it is inferred from the functors involved. However, it is not wrong to write the monad as the object $< \mathbf{X}, T, \eta, \mu >$ where the nature of $\mathbf{X}$ has a bearing on the arguments being made. Further it is often useful to say on which category the monad is based.

For a comonad, the dual of the monad, the diagrams for the associative laws and unitary laws are shown in Figure 11.17. Figure 11.17(a) shows the relationship between S, S^2 and S^3 where S is the endofunctor $FG : \mathbf{Y} \longrightarrow \mathbf{Y}$, **Y** being any category. The counit or identity of the comonad is $\epsilon : S \longrightarrow 1$ from Figure 11.17(a) and the comultiplication of the comonad is $\delta : S \longrightarrow S^2$ from Figure 11.17(b). We therefore write the comonad S as the object $< S, \epsilon, \delta >$ or $< \mathbf{Y}, S, \epsilon, \delta >$.

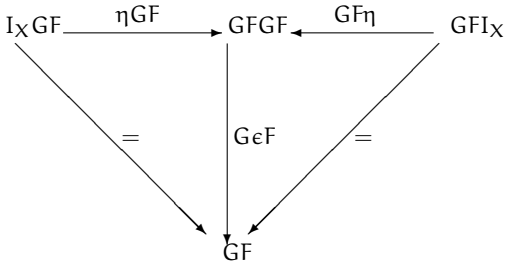

Figure 11.18: The Monad in the category **X**: Triangular Identities defining ϵ

Figure 11.18 shows the two triangular identities for the monad in the category **X**, derived by applying the interchange law to Figure 11.16(b). Through commutativity Figure 11.18 defines the arrow $G\epsilon F : GFGF \longrightarrow GF$. This arrow is the multiplication of Figure 11.16, that is $\mu : T^2 \longrightarrow T$. Therefore we can rewrite the monad $< T, \eta, \mu >$ as $< T, \eta, G\epsilon F >$ for an alternative view, based on the units and counits of adjunction, η and ϵ respectively.

5.5 HISTORICAL AND PRESENT USAGE OF THE MONAD TERM

According to Hippolytus ($170 - 235$ AD), the worldview was inspired by the Pythagoreans, who called the first thing that came into existence the *monad*, from which came the dyad, triad, tetrad, etc. [4]. Gnosticism is a modern term for a multitude of Jewish religious ideas and systems from the first and second century AD, with the highest God, Supreme Being or the One, termed the Monad. The Syrian-Egyptian school depicts creation as coming from a primal monadic source, finally resulting in the creation of the material universe.

The monad entered metaphysics as the *Monadology* of Leibniz, written from 1712-1714 as *Principes de la nature et de la grâce fondé en raison*, which has since been published in various forms and languages [16]. Leibniz allows

just one type of element in the building of the universe, which is given the name monad or entelechy, and described as a simple substance, which has no parts, hence indivisible. Monads are elementary particles with blurred perceptions of one another and have been described as eternal, indecomposable, individual, subject to their own laws, un-interacting, each reflecting the entire universe in a pre-established harmony; monads are centres of force; substance is force, while space, matter, and motion are merely phenomenal. Like atoms, monads are irreducible but differ in their complete mutual independence, and in their following of a preprogrammed set of instructions peculiar to itself, so that a monad 'knows' what to do at each moment. Each monad is like a little mirror of the universe.

The monad term is also used in music, where it is a single note, with a dyad being 2 notes, a triad 3 notes, etc., and in biology where it is a unicellular organism.

In functional programming, the monad is an increasingly popular construction as an abstract data type, with promising developments in the language Haskell [8, 18], named after Haskell B Curry, who developed the transformation of functions through currying in the $\lambda-$calculus. The monad in Haskell is formally classified as an extension of the monad developed in category theory, involving the notion of a strong monad [22, 24]. Such a monad is defined in higher-order category theory as a bicategory construction. In more concrete terms a strong monad is defined as a (categorial) monad with strengthening with respect to products and idempotency. The strengthening with products leads to the concept of a Cartesian monad where, if the underlying categories are pullbacks, the monad T preserves pullbacks and μ and η are Cartesian, then the monad is Cartesian. Such a construction facilitates the use of T in transformations where a Cartesian type is expected. The strengthening with idempotency provides resilience as further operations are performed. So with the underlying category for the monad $\mathbf{X}$ being Cartesian with the object $A \times B$, there is a natural transformation $\tau_{A,B}$ from the Cartesian operation $(A \times TB)$ to $T(A \times B)$ such that strengthening with the identity I is immaterial, consecutive applications of strength commute, and strength commutes with monad unit and multiplication [23]. Further details of the Cartesian monad are found later in this paper in Section 7, in the work by Mulry [24] and in Appendix C of Leinster's book *Higher Operads, Higher Categories* [15].

Category theory is regarded as a unifying force so might be able to provide an insight into all of the above notions of the monad. The notion of unit applies to all the various usages and this is continued into the categorial version with the unit in the monad definition $< T, \eta, \mu >$ of $\eta : 1 \longrightarrow T$ and

the counit in the comonad definition $< S, \epsilon, \delta >$ of $\epsilon : S \longrightarrow 1$. The monad of Leibniz is similar to the categorial version in respect of their following a preprogrammed set of instructions with each monad being a little mirror of the universe. However, there is a major difference – Leibniz's monad is a particle and the categorial monad is a process – emphasising the set-based nature of Leibniz's work. The use of the term monad in music appears to reflect the physical reality of a single note. From a more constructive point of view, musical units, and hence monads, might also include chords and other logical combinations of notes. An application of the categorial monad to music is under active consideration. The use of the term monad for a unicellular organism has lapsed, maybe because the general term was confusable with its use for specific unicellular organisms, the *Monas*. The comparison between the monad of functional programming and that in category theory is the most useful: this shows that the Cartesian monad selected for functional programming is indeed the type of monad needed for information systems as the underlying Haskell category has products, in particular pullbacks, which form the basis of our structural approach.

6 PROCESS ON A TOPOS

The monad and comonad processes are applied to a topos, defining the structure of the data, to perform the transactions. The design of the processes ia therefore termed Monadic Design. We write the process on a topos as:

$$T : \mathbf{E} \longrightarrow \mathbf{E}'$$

where T is the Cartesian monad $< T, \eta, \mu >$ for a category $\mathbf{E}$ with endofunctor T, that is $GF : \mathbf{E} \longrightarrow \mathbf{E}$, unit of adjunction $\eta : 1 \longrightarrow T$ and unit of multiplication $\mu : T^2 \longrightarrow T$.

The source topos is $\mathbf{E}$ and the target topos is $\mathbf{E}'$, with the topos based on pullbacks, including the pasted types, as described in Section 3. The type (intension) of the source and target is the same but the data values (extension) will vary. Closure is achieved as the type is preserved.

For the running bank example, the Cartesian monad T is the banking system transaction, the source information system is $\mathbf{E}$ and the target information system is $\mathbf{E}'$. There may be more than one adjunction for a monad T, based on a category $\mathbf{E}$. For instance $< F, G, \eta, \epsilon >$ may be one adjunction for $\mathbf{E} \longrightarrow \mathbf{E}'$ with another of $< F_A, G_A, \eta_A, \epsilon_A >$ for $\mathbf{A} \longrightarrow \mathbf{E}$, where $\mathbf{A}$ is a subcategory

of **E**. So a variety of adjunctions may be handled by a single monad, over various subcategories of a particular category. This gives flexibility in handling different data-sets with the same underlying structure.

For the process there will also be a comonad:

$$S : \mathbf{E}' \longrightarrow \mathbf{E}$$

where S is the Cartesian comonad $< S, \epsilon, \delta >$ for a category $\mathbf{E}'$ with endofunctor S, that is $FG : \mathbf{E}' \longrightarrow \mathbf{E}'$, counit of adjunction $\epsilon : S \longrightarrow 1$, counit of multiplication $\delta : S \longrightarrow S^2$.

Categories of algebras can be defined over the monad and comonad. From the algebraic perspective, there are two approaches employing the monad/-comonad as the underlying categories. The category of algebras over a monad is traditionally called its Eilenberg-Moore category [6] ([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). The Kleisli category of a monad transforms a monad into a form more suitable for implementation in a functional language such as Haskell. Compared to the EML form of Mac Lane, Kleisli strength generalises the notion of commutativity and guarantees that products lift to the corresponding Kleisli categories [24]. From the point of view of products, the monads developed to Kleisli strength are applicable in a much wider range of computing applications. Kleisli categories are discussed in more detail in Section 7.

6.1 THE T-ALGEBRA

The T-algebras are one of the algebraic forms resulting from the work of Eilenberg and Moore [6]. Such algebras facilitate changing the definition of a monad and therefore permitting fundamental changes to the operand of our process. For any category **X**, which in our case is a topos **E**, the T-algebra produces a new consistent state of adjunction for a modified intension.

In more detail, applying the T-algebra to a topos **E**, in the monad with adjunction $< GF, \eta, \mu >$, yields a new monad adjunction $< G^T F^T, \eta^T, G^T \epsilon^T F^T >$: $\mathbf{E} \longrightarrow \mathbf{E^T}$; that is a new monad adjunction $F^T \dashv G^T$ is defined to accommodate the changed category $\mathbf{E^T}$. A T-algebra is $< e, h >$ where e is an object

in **E**. The structure map of the algebra is $h : Te \longrightarrow e$ such that the diagrams in Figure 11.19 commute. Beck's Theorem provides rules on which categorial transformations in the T-algebra $X \longrightarrow X^T$ are valid [3]. This is sometimes called PTT (Precise Tripleability Theorem).

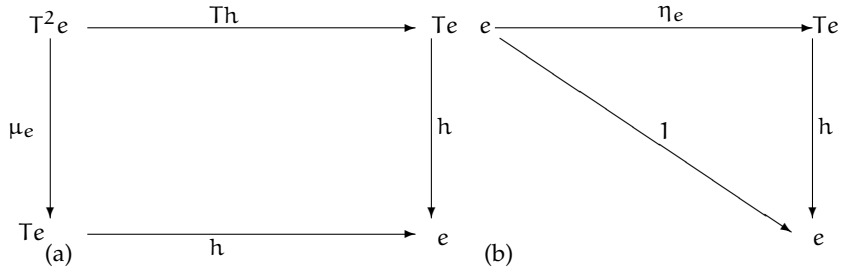

Figure 11.19: T-algebra: (a) Associative Laws, (b) Unitary Laws

7 APPLICATION

The categorial monadic approach is being used for the Blockchain [20], 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 [27]. Meredith indicates that compositionality is the key to reliability but offers few details on how this is achieved in the monad. Compositionality is a cornerstone of category theory, defined as a minimum up to some level of isomorphism. In monad/comonad definitions there is the choice of the Mac Lane (EML) or Kleisli algebras as introduced above in Section 6. It is the approach owing to Heinrich Kleisli that has elevated compositionality to a higher level, through the Kleisli lift, described for instance by Mulry [24]. In the diagram in Figure 11.20, H is a monad $< H, \eta, \mu >$ in **X** and K is a monad $< K, \gamma, \rho >$ in **Y**. The Kleisli categories, representing the free algebras, are $\mathbf{X_H}$ and $\mathbf{Y_K}$. The Kleisli lift of functor F is the functor $\bar{F} : \mathbf{X_H} \longrightarrow \mathbf{Y_K}$ such that the diagram in Figure 11.20 commutes. Associated with this diagram is the definition of the lifting natural transformation $\lambda : FH \longrightarrow KF$ in Figure 11.21, derived through applying the interchange law

to the component functors and natural transformations in the two monads defined above.

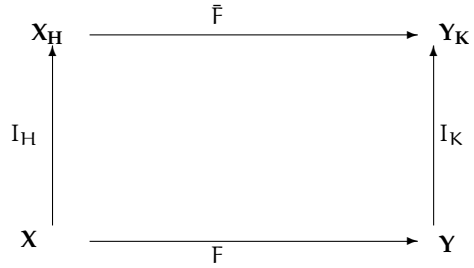

Figure 11.20: Kleisli Lifting of Functor $F : \mathbf{X} \longrightarrow \mathbf{Y}$ to $\bar{F} : \mathbf{X_H} \longrightarrow \mathbf{Y_K}$

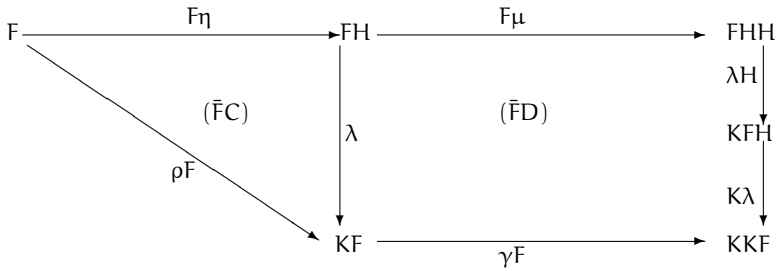

Figure 11.21: Kleisli Lifting of Functor F to $\bar{F}$: the lifting natural transformation $\lambda : FH \longrightarrow KF$

So far the Kleisli lift applies to any category, giving what is termed Kleisli prestrength. We now need to consider the Kleisli lifting of a bicategory, one involving a product of two categories. This is essential if the products are to be well defined for compositional purposes as indicated in Section 6. The lifting gives rise to what is termed Kleisli strength, forming the basis of the Cartesian monad, a term introduced earlier in our overview of the Haskell programming language in Section 5.5. The terms Cartesian monad and strong monad encountered in the literature are for our purposes interchangeable. The enhanced compositionality is achieved firstly by defining a natural transformation $\tau_{A,B} : A \times TB \longrightarrow T(A \times B)$ for objects A, B, C in the category $\mathbf{X}$ with monad $< T, \eta, \mu >$ such that the diagram in Figure 11.22 commutes. A further natural transformation $\lambda_{TA} : I \times TA \longrightarrow TA$ is also defined, as shown in the commuting diagram in Figure 11.23, to reinforce the interchange laws employed in Figure 11.21. Both the diagrams defining the Cartesian monad

involve the Cartesian product, the most relevant for information systems, but the theory is actually more general covering the tensorial (outer) product $A \otimes B$, which may have more relevance for studies involving vectors. Further diagrams are required when the product is tensorial, rather than Cartesian, involving multiplication through the arrow $\mu_{A\times B}$ and associativity though the arrow $\tau_{A,B\times C}$. A major advantage of Kleisli strength monads is that they can, in general, be composed naturally, unlike monads of weaker strength. Such composability increases reliability and scalability, both of which are vital for large scale information systems. Kleisli strength facilitates the discovery of distributive laws.

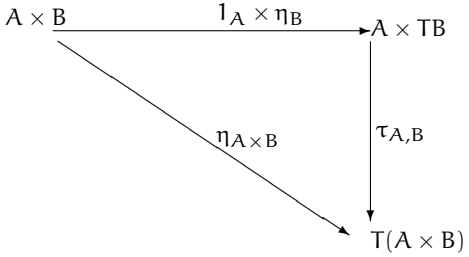

Figure 11.22: Cartesian Monad: Diagram defining the natural transformation $\tau_{A,B}$

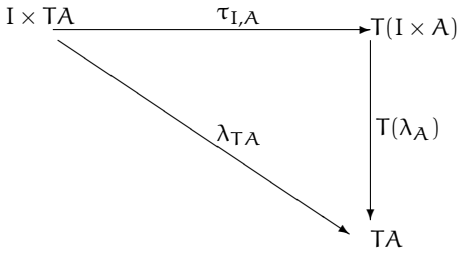

Figure 11.23: Cartesian Monad: Diagram defining the natural transformation λ_{TA}

Meredith [20] envisages that the monadic design patterns, providing a broad range of transactional semantics, would have a front-end data sublanguage of the applied $\pi-$calculus, a compositional process calculus developed for concurrent programming by Milner [21]. However, other presentational

techniques from category theory are available, such as bigraphs, and should also be evaluated before a choice is made.

In the functional programming language Haskell, monadic design patterns are employed. The design pattern for a category **C** is $\mathbf{H} =< H, \eta, \mu >$ where **H** is the monad with type constructor H, η is a return function, $\mu : HHA \longrightarrow A$ is a join function. In more conventional monad terminology H is the endofunctor, η the unit of adjunction and μ the multiplication [24]. If the monad is of the *Maybe* type, there are facilities for exception handling. To facilitate monad composition, the monad is lifted into a Kleisli category, with the power of a strong monad or a Cartesian monad. A monad composition operator, also known as the Kleisli composition operator, is available for composing one monad with another naturally [5].

Returning to our banking example we can see that composition of processes is readily available if our monads are Cartesian, with the Kleisli lift. So for two monads $\mathbf{T} =< T, \eta, \mu >$ and $\mathbf{U} =< U, \gamma, \rho >$, we can write UT for the composite process, where say T is the banking transaction with checks for its feasibility and U is a task establishing remote mirror facilities, as in distributed data recovery systems, for recording the results persistently. Such compositionality could be enforced over large distributed systems by involving many individual monads. So monads can be used either in the small individually in a local environment or, through composition, in the whole universe of the information system. The efficacy of the monad approach can be proven through category theory, thereby increasing the reliability and robustness of a system, where every transaction is critical. Further the monad can be directly implemented in the programming language Haskell, enabling experimental results to be derived.

8 SUMMARY

The combination of the topos, as the underlying data-type, and the monad, as the process or transformer, appears to satisfy the requirements of information systems. The topos is based on pullbacks, which can be nested recursively or pasted together for complex relationships. The bottom level of Dolittle diagrams holds both the intension and extension for the data held. Data normalisation arises naturally through the rules of pullback construction. The subobject classifier of a topos facilitates internal queries on the information system. The monad is defined as three components for operations on a category: an endofunctor that is often an adjunction, the unit of adjunction and the unit

of multiplication. There are two main approaches for applying the monad as an algebra: Eilenberg-Moore (EML) and Kleisli. The Kleisli approach finds favour, with its lift to Cartesian monads handling products, providing compositionality across a succession of monads and a route for experimental implementation in Haskell.

9 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] Banach, R, Regular Relations and Bicartesian Squares, Theoretical Computer Science **129**(1) 187-192 (1994). https://doi.org/10.1016/0304-3975(94)90086-8

[3] Beck, Jonathan Mock, Triples, Algebras and Cohomology, Reprints in Theory and Applications of Categories, Columbia University PhD thesis, **2** 1-59, MR 1987896, originally published 1967 (2003). http://www.tac.mta.ca/tac/reprints/articles/2/tr2abs.html

[4] Bunsen, Christian Karl Josias, Freiherr von; Hare, Julius Charles & Bernays, Jacob, Hippolytus and his Age, published Longman, Brown, Green, & Longmans, London 577 pp (1854).

[5] Diehl, Stephen, Monads made difficult. http://www.stephendiehl.com/posts/monads.html

[6] Eilenberg, Samuel, & Moore, John C, Adjoint Functors and Triples, Illinois J Math **9**(3) 381-398 (1965). http://projecteuclid.org/euclid.ijm/1256068141.

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

[8] λ—Haskell: an advanced, purely functional programming language (2017) https://www.haskell.org/

[9] Heather, Michael, & Rossiter, Nick, Logical Monism: The Global Identity of Applicable Logic, Advanced Studies in Mathematics and Logic **2** 39-52 (2005). http://nickrossiter.org.uk/process/advstudiesmathsmonism.pdf

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

[11] 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

[12] Kent, William, A Simple Guide to Five Normal Forms in Relational Database Theory, Communications of the ACM 26(2) 120-125 (1983). http://www.bkent.net/Doc/simple5.htm

[13] Lambek, J, & Scott, P J, Introduction to Higher Order Categorical Logic, Cambridge (1986). https://github.com/Mzk-Levi/texts/blob/master/Lambek%20J.,%20Scott%20P.J.%20Introduction%20to%20Higher%20Order%20Categorical%20Logic.pdf

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

[15] Leinster, Tom, Higher Operads, Higher Categories, London Mathematical Society Lecture Note Series 298, Cambridge (2004).

[16] Leibniz G W, Monadologie 1714; translated by Nicholas Rescher, 1991. The Monadology: An Edition for Students. University of Pittsburgh Press. Ariew and Garber 213, Loemker S67, Wiener III.13, Woolhouse and Francks 19. Online translations: Jonathan Bennett's translation; Latta's translation; French, Latin and Spanish edition, with facsimile of Leibniz's manuscript at the Wayback Machine (archived July 4, 2012); further editions établie par E Boutroux, Paris LGF 1991; Lamarra, A, Contexte GènGètique et Première Réception de la Monadologie, Revue de Synthese 128 311-323 (2007).

[17] Levene, Mark, & Vincent, Millist W, Justification for Inclusion Dependency Normal Form, IEEE Transactions on Knowledge and Data Engineering **12**(2), pp. 281-291 (2000). http://eprints.bbk.ac.uk/196/1/Binder1.pdf

[18] Lipovača, Miran, Learn You a Haskell for Great Good!, A Beginner's Guide, William Pollock, San Francisco (2011).

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

[20] 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

[21] Milner, Robin, Communicating and Mobile Systems: The π−calculus, Cambridge (1999).

[22] Moggi, Eugenio, Computational Lambda-Calculus and Monads, Proceedings of the Fourth Annual Symposium on Logic in Computer Science 14-23 (1989).

[23] Moggi, Eugenio, Notions Of Computation And Monads, Information And Computation **93** 55-92 (1991).

[24] Mulry, Philip, Notions of Monad Strength, Banerjee, A, Danvy, O, Doh, K-G, Hatcliff, J, (edd.) David A. Schmidt's 60th Birthday Festschrift, EPTCS **129** 67-83, doi:10.4204/EPTCS.129.6 (2013). https://arxiv.org/pdf/1309.5132.pdf

[25] ncatlab, 0-category https://ncatlab.org/nlab/show/0-category

[26] ncatlab, Pullback as an Equalizer https://ncatlab.org/nlab/show/pullback

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

[28] Rossiter, B N, Heather, M A, & Sisiaridis, D, Process as a World Transaction, Proceedings ANPA 27 Conceptions, 122-157 (2006). http://nickrossiter.org.uk/process/anpa064.pdf

[29] 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

[30] Rossiter, Nick, & Heather, Michael, Abstract Relations and Allegorical Categories, ANPA 36, Explorations II, Anton L. Vrba (ed.) 103-134 (2016). http://nickrossiter.org.uk/process/Rossiter-ANPA-PROC-36updated.pdf

[31] Whitehead, Alfred North, Process and Reality: An Essay in Cosmology, Macmillan, New York (1929); corr.ed., eds. David Ray Griffin and Donald W. Sherburne, New York: Free Press (1978). https://monoskop.org/images/4/40/Whitehead_Alfred_North_Process_and_Reality_corr_ed_1978.pdf