Process as a World Transaction

B. N. ROSSITER, M. A. HEATHER & D. SISIARIDIS
University of Northumbria at Newcastle NE1 8ST, UK,
nick.rossiter@unn.ac.uk www.computing.unn.ac.uk/staff/CGNR1/

Abstract

Transaction is process closure: for a transaction is the limiting process of process itself. In the process world view the universe is the ultimate (intensional) transaction of all its extensional limiting processes that we call reality. ANPA's PROGRAM UNIVERSE is a computational model which can be explored empirically in commercial database transactions where there has been a wealth of activity over the real world for the last 40 years. Process category theory demonstrates formally the fundamental distinctions between the classical model of a transaction as in PROGRAM UNIVERSE and physical reality. The paper concludes with a short technical summary for those who do not wish to read all the detail.

1 Background

The universe as process has been one of ANPA's themes for the last quarter of a century. However, the concept dates back not just 25 years but at least 25 hundred years to Heraclites. The ANPA interest has particularly been directed to the combinatorial hierarchy where from minimal assumptions about bit strings of one and zero, a number of physical constants of the universe can be derived [23]. The process mechanism to generate these bit strings has been represented in the computational model of PROGRAM UNIVERSE which has been able to predict very precisely many cosmological values [24].

Nevertheless a number of features are awaiting further elucidation especially the underlying philosophy of process that brings it all about. The issues may be listed as:

a The formal representation of process.

b The significance of zero as a notation for nothing from which the whole universe is generated.

c The distinction between a mathematical model of computation and physical computation.

We have already addressed briefly all three issues in [14]. For the formal representation of process we have made use of the arrow of category theory. It then follows naturally that the cartesian closed category represents reality because like the world it is a category with limits and exponentials. This has enabled us to avoid problems with the second feature (b) above because by means of the concept of the arrow we are able to replace 'nothing' with the empty monoid as the generator of everything. However, it is the third feature (c), that of the mathematical/physical divide that we wish to concentrate on in this paper. There are the well-known limitations in current mathematics that rely on a concept of number and axiomatic set theory wedded to Parmedian notion of invariance [17] that requires some platonic belief in idealism [16]. Gödel has shown that the truth of statements about number in axiomatic systems are undecidable. Whereas here we are concerned with PROGRAM UNIVERSE and the generation of universal constants. PROGRAM UNIVERSE is algorithmic and presented in a context to be realised on a machine with a von Neumann architecture, that is finite memory cells and a serial instruction set. The treatment of the algorithm within ANPA has been principally as a mathematical model for physical computation and includes the concept of discriminated subsets that identifies the system of unique bit strings. What then is the physical counterpart to the mathematical process of discrimination? The current practice in conventional science is to cross from mathematical to physical computation by using the bridging Church-Turing thesis. This problem

was recognised early pre-ANPA by Bastin and Kilmister [5] distinguishing simultaneity from similarity of position when checking a putative new string.

This invokes the Church-Turing thesis of computable recursive functions. Gödel's undecidability in this context takes the form of Turing's halting problem, a theorem it should be noted that is not constructive which is symptomatic of those parts of mathematics that can be only applied to physics with very restrictive (local) conditions [13].

However this cannot provide the simultaneity of a real event, that is 'an occasion' in the terminology of Whitehead's *Theory of Process and Reality*. The universe is not therefore a von Neumann processor. There are applications where we would like ordinary computers to go beyond the universal Turing machine. A very important example is a database transaction where extensive study and experience is available, for operations in practice, because of the very many everyday automated transactions like in banking. We will look at these in some detail to provide insight into the nature of physical transactions in the world. The Church-Turing thesis is more like the proverbial plank in a shipwreck than a firm bridge built on foundations between mathematics and physics. The Church-Turing-Deutsch thesis [8] that any recursive function can be computed on a quantum computer may be true if it can be realised in a physical system but that is more than just a mathematical model – the current state of quantum mechanics [13].

2 Introduction to Transactions

Transactions have long been part of human activity. The simplest bargain amounts to a transaction with each party agreeing to terms. After a deal is struck, no withdrawal is possible although

it may be subsequently unpicked if its legality is later challenged. If the results are ineffective, a further deal must be constructed to achieve the intended results.

With the widespread use of computers in business, information systems handle many millions of transactions a day in a routine manner and database technology has been developed to represent logically in software the complex structure of real-world phenomena. The representation of fixed states gives rise to problems of logical data independence and normalisation but for dynamic behaviour there is the additional feature of natural closure, although from the process viewpoint it may be no more than the dynamic limit of normalisation.

This natural closure may be spread out over time. Classical legal analysis reduces the process to two stages: 1) the formation of the contract; 2) the performance of the contract. Anything before stage (1) is usually called an 'invitation to treat' without binding effect except as a matter of interpretation under a contract but even then written terms will prevail over any oral reservations. Any other conditions not contained within stages (1) and (2) need a separate collateral contract or a contract of variation. In general an ill-formed contract is easier to set aside than an ill-performed one because it is easier to go back to square one without disturbing the rights of third parties who will usually be affected by the performance of the contract. For the formation of a contract only affects the rights of the parties while the performance results in a change in the configuration of the world. Thus purchase of goods on a web site results in the formation of a contract when a bargain is struck on-line with usually executed consideration by debit card. The executory consideration to supply the goods is completed but the contract remains yet to be physically performed.

Problems arise from context-sensitive terms which may not have

been expressed at the time. The implied are uncountable but still determinable. A full theory of defeasible reasoning is a current research topic of great interest [15]. All possible paths in the sense of Feynman [9] are solved in the natural closure of the unitary process of the physical computation of the universe. But the universe is more than a universal Turing machine. It achieves a closure that is undecidable on current classical von Neumann machines. To contain this problem in everyday computer transactions, four main principles have evolved in commercial practice. These are recognised in the ISO standards [1] and are spelt out in the acronym ACID ([6] p.572-629):

- Atomicity ensures that the transaction either is completely successful (all rules obeyed) or completely unsuccessful (when at least one rule broken). No partial results are possible.
- Consistency ensures that the business rules governing the integrity of the transaction have been obeyed.
- Isolation ensures that intermediate results, which may be unreliable before every rule has been tested, cannot be released outside of the transaction.
- Durability ensures that once the transaction has been completed, its results will persist. The system attempts to ensure that the results of the transaction are not lost through the use of a transaction log to be described in the example below. The state of the data can still change though – through another transaction.

[1] ISO/IEC CD 9075-2 Information technology – Database languages – SQL – Part 2: Foundation (2003).

3 Example Application - ATM

All information system transactions involve physical processing if we take Landauer's point [19] that information can only exist if it has a physical manifestation. An interesting application as an example of real-world application is that performed by an Automated Teller Machine (ATM) where the physical aspects of the transaction are very evident in the delivery of cash in note form. While the external view of such a task is familiar to all today, what goes on behind the hole in the wall is not so well known even to the average computer professional or banking official. There is no absolute certainty of closure using current digital computers. This raises many problems for the designers of transaction systems. For instance a banking client who withdraws £500 in cash through a transaction has a physical asset matched against a logical debit. The logical system needs to provide closure with a very high degree of certainty if it is to retain confidence. It is a problem of simultaneity for the basic logic AND operator. The cash has to be delivered and the customer's account debited. One cannot be allowed to succeed without completion of the other. The process and its associated problems have been well known for some decades and described in [11].

The detailed exposition of running transactions for a banking system has been written by Jim Gray of Microsoft [12] for a relational database system. The implementation program described in Figure 1 involves one database *theBank* holding four tables (*Branch, Teller, Account, History*). The first three tables have primary keys declared so that each row will be unique, for example the Branch has a primary key of branchID, the Teller has a primary key of tellerID and the Account of accountID. The branchID attribute also appears in the Teller and Account tables. To avoid

inconsistency problems branchID in these tables is declared as a foreign key to branchID in the Branch table meaning that a branch in the Teller and Account tables must already be entered in the Branch table. Data changes are under the control of three stored procedures:

1 *spFillBank* for populating initially the *branch, teller* and *account* tables.

2 *spDebitCredit* for handling a single debit/credit transaction.

3 *spRunDebitCredit* for running many transactions together.

To drive the whole process, a script *ParallelBatch.bat* is provided to run many *spRunDebitCredit* threads in parallel. The purpose of Gray's paper is to demonstrate how an ordinary PC can handle a very large historic volume of traffic of transactions, in this case the entire banking system needs for 1970.

Gray illustrates some of the safeguards, in particular the insert in the *History* table in statement 12 of the transaction details to give a copy of the transaction. This assists durability as if the main tables *Account* or *Branch* are damaged after their updates have been made in statements 11 and 13 respectively, a copy of the transaction's action still exists. The commit in line 14 is important for our subsequent discussion. A commit saves the result of the transaction on disk, updates the transaction log and closes the process. The transaction log is used for holding the commands and results of successful changes and is described in more detail later. The final stage of commit is only performed after the writing of the transaction log has been successful.

```
The classic database part of TPC-A (and DebitCredit)
-- This is a single DebitCredit database transaction.
1     create procedure spDebitCredit @tellerID int,
                  @accountID int, @amount float as
      \* procedure called spDEbidCredit has 3 parameters
      tellerID, accounted and amount *\
2     begin
3     declare @newBalance float,
4           @branchID int,
5           @BranchRadix int
      \* three local variables are declared above *\
6     set @BranchRadix=1000000
7     \* one million is ratio account/teller ID *\
8     set @branchID = @tellerID / @BranchRadix
9     begin transaction
10    update Teller set till = till + @amount
                  where tellerID = @tellerID
      \* till of teller is updated by amount *\
11    update Account
            set @newBalance = balance = balance + @amount
                  where accountID = @accounted
      \* customer account is updated by amount *\
12    insert History (branchID, tellerID, accountID, amount)
            values ( @branchID, @tellerID, @accountID,
                  @amount)
      \* changes are recorded in the History table *\
13    update branch set balance = balance + @amount
                  where branchID = @branched
      \* branch balance is updated by amount *\
14    commit transaction
      \* changes are saved persistently *\
15    end
16    go
```

Figure 1: Stored Procedure *spDebitCredit* for single debit/credit of bank accounts (from [12], Appendix I Debit Credit Sample Code)

The alternative to commit is to rollback to the initial state: no intermediate results are retained unless the task specifically defines a savepoint. However, a return to a savepoint is a temporary reprieve for the transaction to see if it may be performed in a different way. A savepoint is not a viable endpoint. Rollbacks are governed by the atomicity and consistency properties of a transaction: the

transaction completely fails if just one rule is broken, resulting in the return of the data states back to their initial values.

On failure of the database the last saved 'good' copy is restored and the transaction log (described in more detail in the next section) run against this copy so that no transactions will be lost. Transactions in progress at the time of the 'crash' are discarded as partial results are not viable.

While transactions might appear as being logical processes, this is rather naïve when we look at examples. With the ATM, customers withdraw money as cash, the physical form of money. The mix of logical and physical emphasises the need for a very high degree of reliability. Such examples confirm the principle of Landauer [19]: information cannot exist except in the physical form.

3.1 Potential Failure Points

The data structures and processes above are only an outline. In particular business rules and exceptions would be more fully specified:

1 There is a need to check or provide information on whether the account from which payment is to be made actually has the funds available. Such a rule might require the item already credited to be cancelled through a rollback command. This offends Consistency in the ACID criteria.

2 There is no exception handling for when an update or insert fails. Transactions are committed but there is no provision for rollback. For instance if some of the update and insert operations succeed and others fail, there should be a complete return to the starting point of the transaction with all changes aborted. This offends against Atomicity, Consistency and Isolation in the ACID criteria.

In addition, as usual for today's transaction systems, some of the work done by the information system to underpin the transaction concept is not made explicit in the definitions. The saving to the History file is useful but not infallible. For instance the History file may reside on the same disk as the Account file. If there is a disk crash with the whole disk pack failing, both files would be lost. The normal way for recovering from this situation is to use the History table contents to run forward a repeat of the transactions from the last save. This would not be possible if the History file were lost so the requirement of durability is not met.

A common way to achieve durability is to keep a transaction log (or redo log) for recording every successful transaction, that is one whose results are committed. The transaction log is maintained by the database system and is not otherwise accessible to write to or to read. Every committed transaction is written to a number of copies of the log. One may be written to the database disk, others to devices independent of the database and still others over fast networks to other physical sites. In the event of a failure, the redo log is re-run from the last saved version. Such techniques achieve durability to the desired standard because of the many copies of the log in existence.

There is always though a small degree of risk. The problem is that what should be a parallel task is decomposed into a number of sequential steps in which the failure of any part threatens the ACID requirement. A sequence of operations between fixed cells is used because of the von Neumann architecture and locality of view. Obviously businesses can construct systems that are fail-safe from their viewpoint but such systems can adversely affect customer confidence. In addition if the business is exchanging physical assets for logical ones, there is always a risk arising from the differences in rollback operations for physical and logical items: for physical the

business can only request return of goods, for logical the business can readily undo the effect.

In practice some businesses rely on semi-automatic control for difficult cases. Supervisors inspect machine logs and recorded downtimes in order to complete account details and to correct accounts manually if necessary. This is recourse to a higher level of closure, namely by human intervention.

3.2 Security

This has taken us some way beyond PROGRAM UNIVERSE with its limited syntactical view of the world. Nevertheless the semantic and pragmatic aspects are part of the universe and any theory of process will be inadequate if it cannot include them. The anthropic principle [4] is of interest in cosmology not just in its weak form but even in the strong version because of the current debate on intelligent design [7]. In this context of the transaction we have already mentioned in passing one human aspect in connection with the law of contract. But security now plays an important role in any modern commercial transaction. There are legal elements here too but much additional complexity comes from extra technical components needed for security purposes. Therefore although not always made explicit within a transaction, security is a recognised part of database systems being enforced at least in part by the access rights granted to stored procedures. Security is increasingly important in modern networked computer systems as they are exposed to a growing number and a wider variety of threats and vulnerabilities. It is a very complex task ranging from the level of crypto-primitives over crypto-protocols to the level of organizational matters and legislation [1]. A comprehensive analysis of the literature shows that security for distributed information systems is not a local feature but has to be treated globally. Information se-

curity threats are global in nature and usually automated and loose on the Internet. Organizations usually respond to security threats on a piecemeal basis including anti-virus software, anti-spam, and anti-intrusion software, which need to be updated and redeployed if they are to remain effective.

Stephenson [29] argues that it is unlikely that any organization will have exact knowledge of the probability of the occurrence of a particular event. Bottom-up approaches (e.g. risk analysis) are subjective; these are more suited to high-level security risks. On the other hand, top-down approaches (e.g. baseline approaches) [2] leave the choice of control to the user; they are most appropriate for low-level security risks. A complete security strategy needs to be layered to deal with issues such as continuity strategies (threat assessment, risk evaluation and control), security policies, incident response plan, host-based and network-based perimeter and/or perimeterless detection, auditing procedures, fault tolerance and recovery strategies, anti-malware control (intrusion detection, router and firewall security, anti-virus control) as well as legal and regulatory compliance.

A holistic approach embraces all aspects of security, including systems architecture, policies, procedures and user education. It focuses on securing the infrastructure itself by forcing users to adopt best security practices while ensuring that the network is secure by design: that is rather than to apply post-rational customisation. A promising solution is to include security considerations as core processes of the information system itself, where local extensionalities (e.g. local security policies) are interconnected one with another through global intensionality (e.g. global security policy or meta-policy framework). Nevertheless, it is crucial that any so-

[2]such as ISO, Information Security Management ISO/IEC 27001:2005 Specification, *International Organization for Standardization* (2005) and ISO, Information Security Management ISO/IEC 17799:2005 Code of Practice, *International Organization for Standardization* (2005).

lution must remain simple to implement as well as simple to use from an end user perspective. OECD Guidelines [25] have been designed to develop a 'culture of security', suggesting the need for a greater awareness and understanding of security issues. The holistic approach to security follows from the everywhere at every time need to have an alert guard like a guardian angel. There is then a security dimension slice through the ACID principles that might be interpreted as in the table in Figure 2.

Principle	Security Aspect	Holistic Approach
ATOMICITY	*ab initio* conditions restored and changes cancelled when compromised	Secure identity of all or nothing
CONSISTENCY	Only authorised processors executable	Overall integrity maintained
ISOLATION	Internal/external firewalls maintained	Privacy policy and data protection
DURABILITY	Inviolability	Global identity

Figure 2: ACID in Security

A holistic approach with closure seems necessary for any universal description. A holistic approach also offers benefits in providing security for a piecemeal approach would inevitably leave gaps and generate inconsistencies which could be exploited by intruders. Category theory as already mentioned provides a formal approach to process by the use of the arrow. It is inherently holistic and with intrinsic natural closure and will be explored further in the rest of this paper.

4 Theory of Transactions

For a theory of transactions, formal constructions are needed to represent and measure the changes between states as dynamic relationships. Such changes in states should be governed by the ACID criteria.

The four-level architecture, as developed by this research group [26, 27], is an example of category theory adjusted into an architecture to which Information Systems (IS) people, e.g. those in databases, could relate. This architecture involves categories at each level, functors mapping between the levels, natural transformations comparing one functor with another and adjointness representing the relationship between one functor and another mapping in a reverse direction. In our more recent work [27] contravariancy has been introduced in the functors so that the levels could be clearly defined as alternate pairs:

$$\text{value} \longrightarrow \text{name}; \qquad \text{name} \longrightarrow \text{type}$$

The approach can be refined further by introducing subcategories to deal with partial functions and to facilitate multiple inheritance. The architecture has become known as a four-level one but it is perhaps more meaningfully described as three-level in terms of mappings with three functors connecting the four levels. In category theory it is possible to go to higher levels [21] although we have restricted ourselves to the basic levels only. In the broader sense of security it may be that, on the surface anyway, a more refined architecture might appeal. The IS architecture is mapping between data structures at various levels of abstraction. The security one is mapping between pairs of adjoint functors with abstractions still to be decided. However, the building blocks of concepts, constructs, data types and data values, used for categories in the IS approach, is not the only viewpoint. The three levels of mappings between these data structures are Policy, Organisation and Instantiation, which might be equivalent to Policy, Organisation and Mechanism in the security context. There is a difference in the environments for data structures and security. For security the power is in the permutation of the compositions so that with six

composed adjoints it might be possible to insist on all six working at the equivalence level or more plausibly at some lower level of relationship.

The aspect unifying data structures and security is process. Process is relatively neglected in information studies but is arguably the more important aspect of any system in order to define how transitions are made from one state to another. In this paper the realisation in category theory of process, through the transaction concept is further explored. In particular the fine structure of the levels is investigated in more detail.

5 Adjointness

One of the most important features of category theory is adjointness, which gives a degree of measurement of the extent to which the mappings between two categories are equivalent [3, 20].

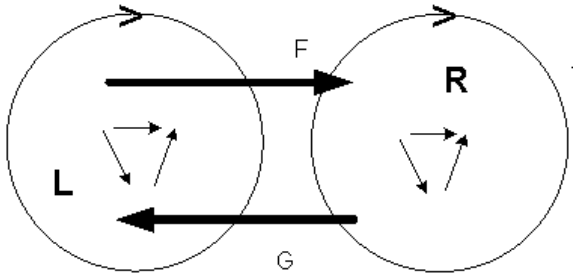

Figure 3: Adjointness between Two Systems

Figure 3 shows two categories denoted **L** for left, **R** for right for historic reasons because they correspond to left- and right-exactness respectively. each containing a canonical triangle to illustrate typical composable arrows. The composition of the arrows (drawn as triangles) represents the natural exactness of real-world interoperability. As we are relying on constructive process not axiomatic sets this interoperability is free from Russell's paradox and free of Gödel's undecidability [28]. The arrows between the

categories are functors F, G, the free and underlying functor respectively. Each of the functors in Figure 3 may be resolved into two component types (technically colimits). These are covariant and contravariant. For a pair of interoperating systems given by these two categories, that is where the triangle in the left-hand category maps into the particular triangle in the right-hand category, then there is a unique contravariant functor G that maps between those triangles in the opposite way. The reverse logic gate $F \dashv G$ is conventionally used to represent adjointness. It is the phenomenon of naturalness. In the vocabulary of axiomatic category theory it is a characteristic of cartesian closed category that applies to all process arrows. It was the publication of [18] that led to the recognition that this effect was ubiquitous. F is left adjoint to G and G is right-adjoint to F. The unit of adjunction η and counit of adjunction ϵ measure respectively the quantitative and qualitative effect of process that is the extent to which the result from composing the functors differs from the initial behaviour:

η_L is the unit of adjunction $1_L \longrightarrow GF(L)$ and ϵ_R is the counit of adjunction $FG(R) \longrightarrow 1_R$ where 1_L and 1_R are the identity functors respectively for the left and right categories.

The categorial basis of process was investigated in our work at ANPA in 2004 [14] with adjoints used as the principal constructions to represent changes between states as dynamic relationships. More specifically adjunctions involve two categories, say, $\mathbf{S}, \mathbf{A}$ and a pair of functors $F : \mathbf{S} \longrightarrow \mathbf{A}$ and $G : \mathbf{A} \longrightarrow \mathbf{S}$. F is termed the free functor which chooses to change the state of the right-hand category $\mathbf{A}$. G is termed the underlying (or forgetful) functor which provides the corresponding change in the state of the left-hand category $\mathbf{S}$. A view of such functors is that F facilitates the desired change in state and G enforces the rules controlling the change in state. An adjunction is completely defined by four para-

metric arrows: $< F, G, \eta, \epsilon >$ where η is the unit of adjunction and ϵ is the counit of adjunction. η measures the process change after applying both F and G in turn to the left-hand category $\mathbf{S}$. ϵ measures the process change after applying both G and F in turn to the right-hand category $\mathbf{A}$. An example of adjointness, mapping the intensional universe, category $\mathbf{S}$, to an extensional information system, category $\mathbf{A}$, is shown in Figure 4.

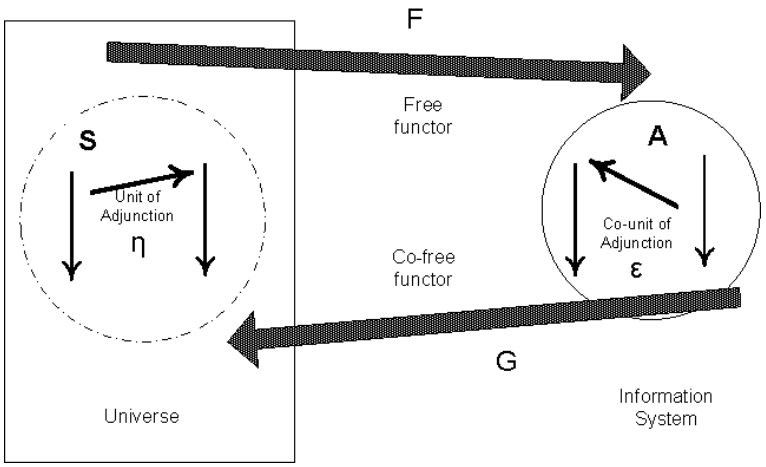

Figure 4: Adjointness from intensional universe $\mathbf{S}$ to extensional information system $\mathbf{A}$

6 Adjointness for Transactions

A transaction may progress in many ways. Here we look at a number of possibilities, using adjointness to measure the relationships between the possible states. Figure 5 shows the initial state with a category $\mathbf{S}$, corresponding to the left-hand category of Figure 4, and a category $\mathbf{A}$, corresponding to the right-hand category of Figure 4. $\mathbf{1_S}$ is the identity functor for the category $\mathbf{S}$ ($\mathbf{1_S} : \mathbf{S} \longrightarrow \mathbf{S}$) and $\mathbf{1_A}$ is the identity functor for the category $\mathbf{A}$. By writing the identity functor for a category, we are treating the category as cartesian closed. To study the transaction as process we will analyse it along an ordered path for which in the banking transaction is real time but is just a disection for a universal physical

transaction which is an occasion of simultaneity (Figure 3). However in the order disection viewpoint no mappings have been made at this point by the functors between the arrow f in $\mathbf{S}$ and $f^\sharp$ in $\mathbf{A}$.

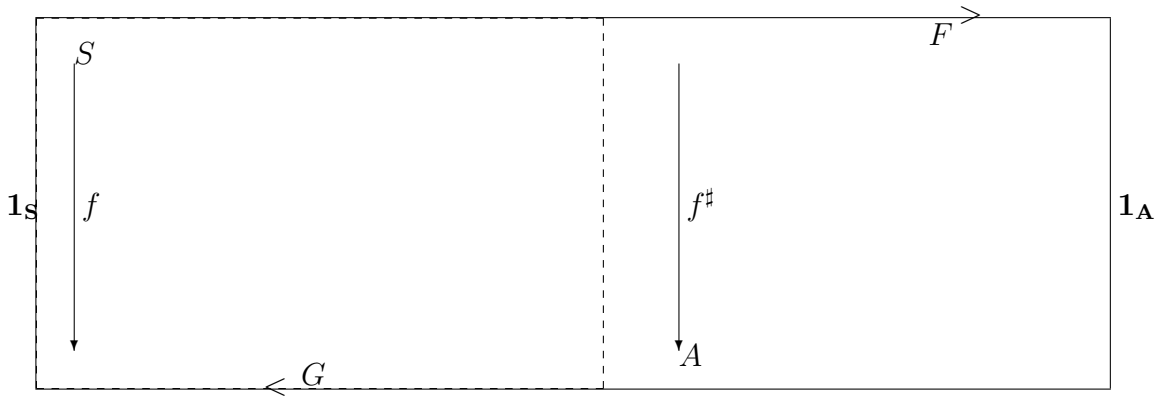

Figure 5: Initial State: Correlation between Arrow f in $\mathbf{S}$ and Arrow $f^\sharp$ in $\mathbf{A}$

In $\mathbf{S}$ and $\mathbf{A}$, the given categories, there will be a given functor which maps f onto $f^\sharp$ so after 'one cycle' [3] applying functor F to $\mathbf{S}$ and G to $\mathbf{A}$, a possible outcome for the categories is shown in Figure 6. This represents no change in f as a result of applying GF. Technically this means that η maps on to $\perp$: the unit of adjunction is mapped on to the initial object. From a transaction viewpoint, no change has occurred in $\mathbf{S}$ indicating a null outcome, probably considered as a failure situation for a customer in the banking transaction, perhaps due to a mistaken pin number but still treated as a successful outcome form the security perspective.

When η maps other than to the initial object $\perp$, the 'single cycle', applying functor F to $\mathbf{S}$ and G to $\mathbf{A}$, gives a change in $\mathbf{S}$

[3] 'single cycle' or 'one cycle' will be used in quotes for the remainder of this paper as a reminder that it is a non-local component of simultaneity in a process transaction in the physical universe. For similar reasons 'two cycles' and 'three cycles' will be used later.

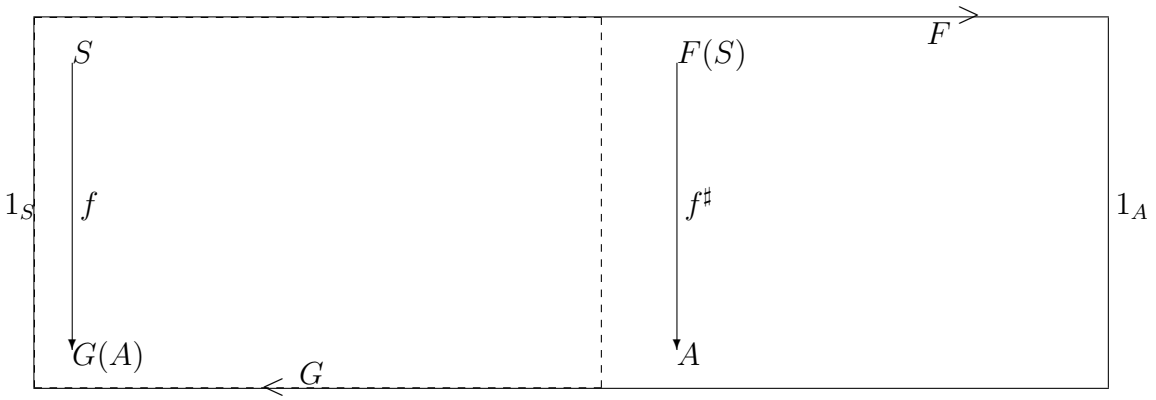

Figure 6: After 'one cycle' GF from left-hand category: $\eta \longrightarrow \perp$, transaction failure

as shown in Figure 7. The triangle, as shown in Figure 8, gives a unique solution such that $f = G(f^\sharp) \circ \eta$. The mapping $\eta : S \longrightarrow GF(S)$ indicates the change in S after applying functors F and G in turn. This state indicates a potentially successful transaction in that change has occurred but we are far from being in a position to satisfy the ACID requirements. This is flagged by the status of TRANSACTION PROGRESSES. For the occasion of a transaction in the physical universe this is 'natural'.

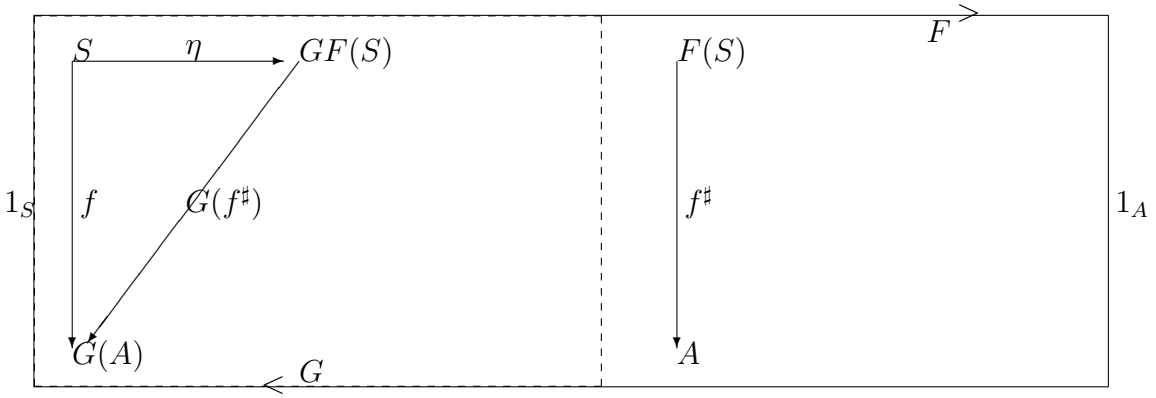

Figure 7: After 'one cycle' GF from left-hand category: η maps to other than $\perp$, 'transaction progresses'

Figure 9 shows the triangle in the broader context of an adjointness diagram, such as that shown in Figure 4.

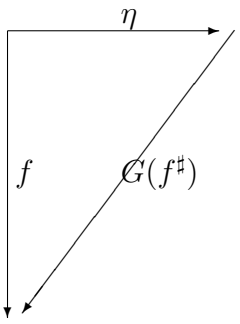

Figure 8: Uniqueness of adjunction: only one possible arrow $G(f^\sharp)$

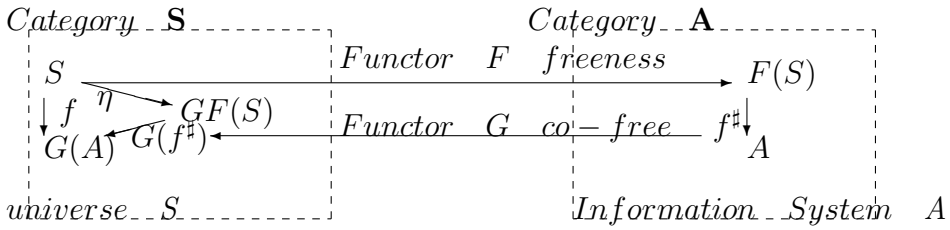

Figure 9: Covariant Mapping between universe and Information System: unit of adjunction η maps on to other than $\perp$

So far we have been looking at the transaction solely from the perspective of the category **S**. To achieve symmetry we need to add the perspective for category **A**. This results in the diagram of Figure 10 which determines the counit of adjunction defined as $\epsilon : FG(A) \longrightarrow A$. The state of the transaction is now that 'one full cycle' has been completed in each direction: GF from left to right to left and FG from right to left to right. We have four possible cases for the dynamical relationships:

1 η maps on to other than $\perp$ and $\top$ is other than ϵ (Figure 10): changes have occurred in both categories and the transaction progresses normally.

2 η maps on to $\perp$ and $\top$ maps on to ϵ (Figure 6): no changes have occurred in either category and the transaction has null success. For more pure mathematicians working in more general category theory this situation might be described as trivial. In the applied categories of a PROGRAM UNIVERSE this

situation is the empty monoid which describes the intensional form of the universe [14].

3 η maps on to $\perp$ and $\top$ is other than ϵ (Figure 10 with η mapping on to $\perp$): changes have occurred in the right-hand category but not in the left-hand category; the transaction is still progressing but there are consistency problems which may arise from the response of some security mechanism.

4 η maps on to other than $\perp$ and $\top$ maps on to ϵ (Figure 10 with $\top$ mapping on to ϵ): changes have occurred in the left-hand category but not in the right-hand category; the transaction is still progressing but there are other consistency problems again possibly from a security procedure.

Cases 1 and 2 both provide consistency, the first for a successful transaction, the second for an unsuccessful one. In the second case, the adjointness relationship is the special one of equivalence. Cases 3 and 4 provide an inconsistent position from a transaction perspective with changes made on one category only.

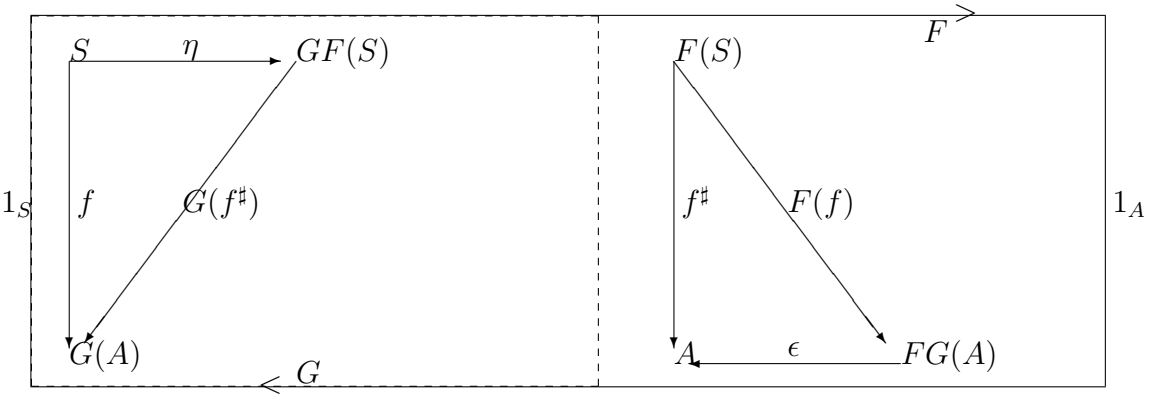

Figure 10: After 'one cycle' GF from left-hand category and one cycle FG from right-hand category: η maps on to other than $\perp$ and $\top$ maps on to other than ϵ, 'transaction progresses'

The single cycles GF and FG provide a provisional view of how the transaction is running. They are insufficient for closure as can

be seen with cases 3 and 4 as inconsistencies may occur. Even with cases 1 and 2, closure is not achieved yet although there are indications as to the likely outcome. It is worth reviewing how the results of this 'single cycle' meet the ACID principles:

- Atomicity: all outcomes are achieved from the single composition of functors (GF and FG).
- Consistency: changes to one category and not to the other suggest changes made may not satisfy all of the rules so a rollback is needed.
- Isolation: results have not been released during the single cycles and it is still not safe to do so because of consistency problems.
- Durability: a commit is needed to maintain consistency.

A further cycle could be employed as in Figure 11. Such a cycle might enable a view to be taken of the results so far and what needs to be done for closure to satisfy the ACID principles.

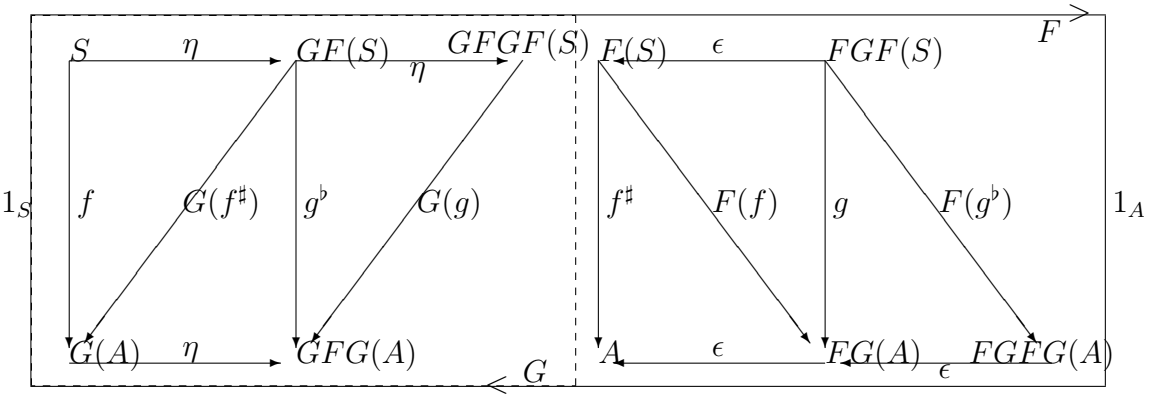

Figure 11: After 'two cycles' $GFGF$ from left-hand category and 'two cycles' $FGFG$ from right-hand category: η maps on to other than $\bot$ and $\top$ maps on to other than ϵ, 'transaction can complete'

One very interesting addition is the arrow g in the category $\mathbf{A}$ [4]. If f is the arrow generated in category $\mathbf{S}$ and sharpened to $f^\sharp$ in $\mathbf{A}$, then g is the arrow generated in category $\mathbf{A}$ and flattened to $g^\flat$ in $\mathbf{S}$. It therefore appears that to gain symmetry with the creation of records in the two categories, 'two cycles' are necessary.

The diagram shows a further application of η and of ϵ. The values for η and ϵ must be the same for each cycle although the values as a whole could be revised during the 'second cycle' to take account of new circumstances. So it would be possible for $\top$ to map on to other than ϵ in a 'single cycle' system but in a 'two cycle' system the solution could be revised so that $\top$ maps on to ϵ in each cycle.

One of the needs for a 'second cycle' may be understood at the experimental level as the requirement to ensure consistency in the categorial structures in the event of a failure during the first cycle. For instance the free functor F may establish a change ϵ in the right-hand category but the underlying functor G may, through the application of a rule, record no change in the left-hand category: η maps on to $\bot$, that is the state of the left-hand category has been returned as the initial object. Generally it is not safe to commit the change ϵ under such circumstances. In a 'second cycle' the change ϵ will be undone in the right-hand category: $\top$ maps on to ϵ, that is the state of the right-hand category has been returned as the terminal object. This 'second cycle' thus achieves a rollback in transactional terms in which the left- and right-hand states are restored to initial and terminal respectively. If the 'first cycle' is apparently successful with changes recorded as η and ϵ, the 'second

[4] We are rather using the traditional language of category theory here. In pure process category theory what we describe as category $\mathbf{A}$ is the partial order valued free functor F and $\mathbf{S}$ is the preorder valued underlying functor G. The arrow g as a potential subarrow of G then becomes the corresponding solution $g = F(f) \circ \epsilon$ for the subarrow f of F given by $f = G(f^\sharp) \circ \eta$.

cycle' is effectively a view to commit or rollback the transaction, potentially confirming the new states as those to be recorded as the output of the transaction.

The 'second cycle' therefore can be used to advance the achievement of ACID as follows:

- Atomicity: all of changes done as single composition of functors ($GFGF$ and $FGFG$) with equal perspective to each category including generation of arrow f in $\mathbf{S}$ and g in $\mathbf{A}$.
- Consistency: partial changes on one category only are subject to rollback by mapping η to $\bot$ or $\top$ to ϵ on the 'second cycle'.
- Isolation: intermediate results are not released during the 'two cycles'.
- Durability: results are still to be committed after 'two cycles' have been completed with consistency. The transaction log has yet to be written so there is still some risk due to a system 'crash'. That is vulnerability persists from the security perspective.

7 'Three cycles'

So if in the 'first cycle' η, ϵ and f are derived and in the 'second cycle' g is derived and the values of η and ϵ are reviewed with a view to commit or rollback, what might a 'third cycle' involve? Such a cycle is concerned with ultimate closure, involving aspects such as issuing error messages from a rollback, writing the transaction log and performing the actual commit. We could persevere with the notation used so far for adjointness but there is a more concise notation available - the monad.

7.1 Monads

Monads represent the multiple cycles identified above. Monads can be viewed as a generalisation of a monoid ([22], p. 137-138) with the set of elements replaced by an endofunctor $T : \mathbf{X} \longrightarrow \mathbf{X}$ with the same category $\mathbf{X}$ as source and target, the cartesian product by composition of functors ($T \circ T.$ that is T^2), the binary multiplication by a natural transformation ($\mu : T^2 \longrightarrow T$) and the unit (identity) object by the unit $\eta : 1_X \longrightarrow T$.

Formally the monad is a triple [2] $< T, \eta, \mu >$ in a category $\mathbf{X}$ which consists of a functor $T : \mathbf{X} \longrightarrow \mathbf{X}$ and two natural transformations:

$$\eta : 1_X \longrightarrow T; \quad \mu : T^2 \longrightarrow T$$

such that the diagrams in Figure 12 and 13 commute.

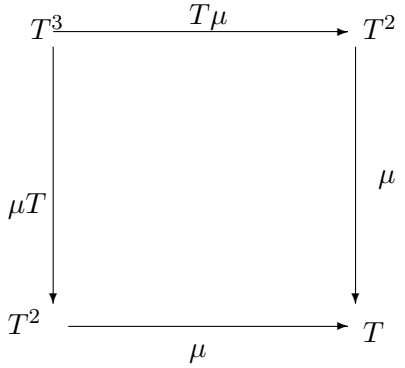

Figure 12: Associative Law for Monad $T =< T, \eta, \mu >$

In Figures 12 and 13 the terms $T\mu$ and μT appear. To illustrate these terms, we first look at an adaptation of a diagram from ([3] p.114) to produce Figure 14, showing categories $\mathbf{S}$, $\mathbf{A}$ and $\mathbf{S}$ again, functors $F : \mathbf{S} \longrightarrow \mathbf{A}$, $G : \mathbf{A} \longrightarrow \mathbf{S}$, $F' : \mathbf{S} \longrightarrow \mathbf{A}$ and $G' :$

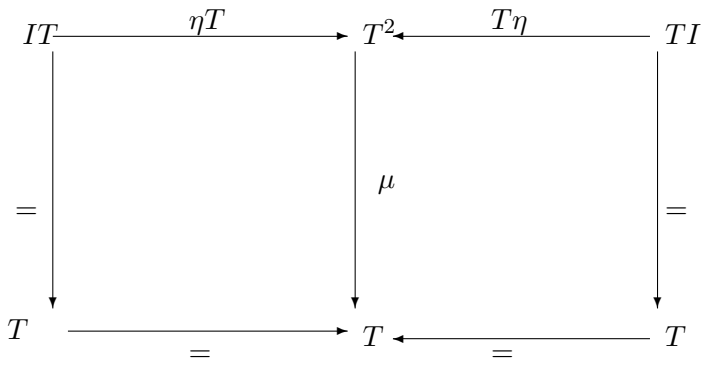

Figure 13: Left and Right Unitary Laws for Monad $T =< T, \eta, \mu >$

$\mathbf{A} \longrightarrow \mathbf{S}$, composition of functors $G \circ F$ and $G' \circ F'$ and natural transformations $\alpha : F \longrightarrow F'$ and $\beta : G \longrightarrow G'$. Category $\mathbf{S}$ is repeated on the right-hand side so that the example covers directly the adjointness case described later.

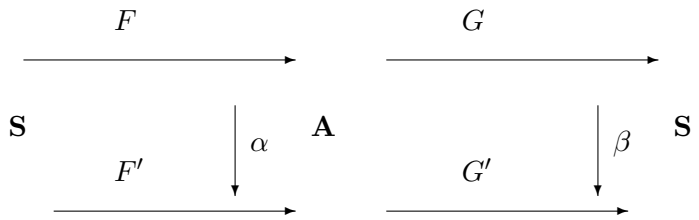

Figure 14: Composition of Functors and Natural Transformations

That the diagrams of Figures 15 and 16 commute, ensures that composition of functors and natural transformations is natural.

Figure 15 defines a natural transformation βF whose value at an object S in $\mathbf{S}$ is the component of β on the object FS. Such a natural transformation is sometimes written as β_F to indicate that we are dealing with a functor-valued natural transformation. The

Figure 15: βF is natural for arrow $f : A \longrightarrow A'$ in $\mathbf{S}$

construction βF is analogous to that of μT in Figure 13. That is μT is the natural transformation μ with a functor value of T.

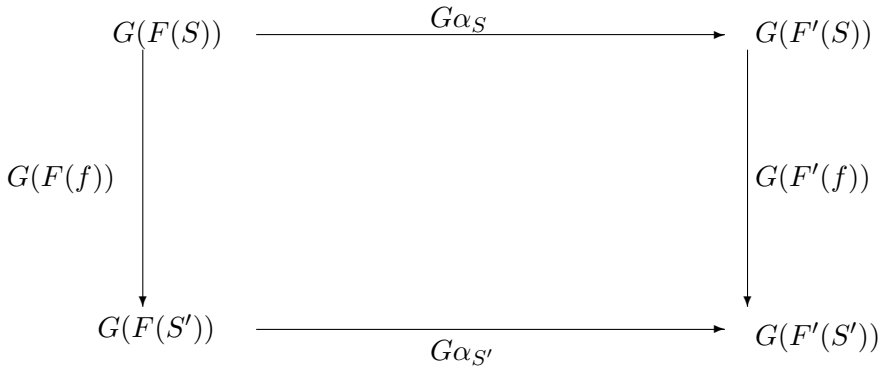

Figure 16: $G\alpha$ is natural for object S of $\mathbf{S}$

Figure 16 defines the composition of a natural transformation α_S with G, written $G\alpha_S$. The construction $G\alpha_S$ is analogous to that of $T\mu$ in Figure 13. That is $T\mu$ is the composition of a natural transformation μ with T.

By the natural logic rules from Godement [10] ([3] section 4.4.7), the functors and natural transformations in Figure 14 compose naturally: βF equals $G\alpha$. So in Figure 12 μT equals $T\mu$ and in Figure 13 ηT equals $T\eta$.

The monad construction can be readily adapted to handle an adjunction. The composition of functors T becomes GF, η remains

the unit of adjunction and μ becomes $G\epsilon F$ where ϵ is the counit of adjunction. Such a construction provides η the unit of adjunction for the first cycle and $G\epsilon F$ the counit of adjunction for the second cycle. That is the monad is $< GF, \eta, G\epsilon F >$ for the adjunction $< F, G, \eta, \epsilon >$. The arrow $G\epsilon F$ is identifiable in the diagram given earlier in Figure 11. For we map with the contravariant functor F taking $S \longrightarrow GF(S)$ in $\mathbf{S}$ to $FGF(S) \longrightarrow F(S)$ in $\mathbf{A}$. Then by applying G to the arrow $\epsilon F : FGF(S) \longrightarrow F(S)$ in $\mathbf{A}$, derive $G\epsilon F : GFGF(S) \longrightarrow GF(S)$, that is μ.

The monad construction with adjointness provides a number of constraints very relevant to the process transaction:

1 There is a unique solution through the adjointness $F \dashv G$.

2 The unit of adjunction η maps on to other than the initial object $\perp$.

3 The natural transformation μ, looking back from the 'second cycle' result to those for the 'first cycle', is defined as $G\epsilon F$, an expression involving the counit of adjunction ϵ. μ would be mapped to $G \top F$ on the 'second cycle' if a rollback was desired, that is partial changes should be rescinded.

4 The arrow $T\mu : T^3 \longrightarrow T^2$ in Figure 12 is a natural transformation comparing the second and third cycles from the viewpoint of the 'third cycle', that is 'looking back' so to speak. $T\mu$ is $GFG\epsilon F$ ([22], p. 138). If $T\mu$ is mapped to $GFG \top F$ then the transaction is rolled back. Otherwise the transaction is committed and redo information written to the transaction log. This 'final cycle' therefore, in giving a final reinforcement of the constraints, facilitates durability.

In the 'third cycle' of the monad, the final value for ϵ is determined (to apply across all cycles). The value for η is not changed

in this cycle. However, the monad only gives half the story from the left perspective. There is also a dual comonad which gives the right-hand perspective. This is needed to represent the full features of a transaction but there is no room here to pursue the comonad.

8 Categorical Approaches to Banking Application

There are a number of ways in which category theory might be used to represent the banking ATM system described earlier. In ascending order of complexity these might include composition, adjoint functors between two categories and the composition of adjoint functors. We first look at the two relatively simple cases and then pursue the third.

8.1 Simple Approaches

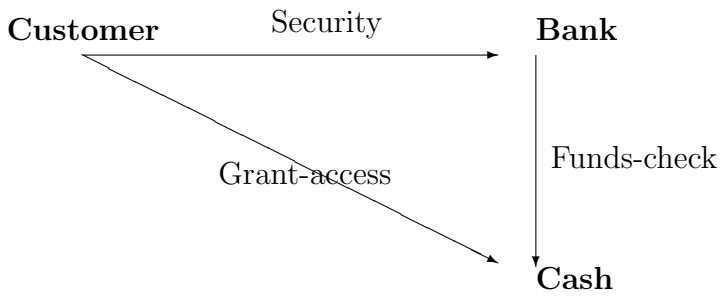

Figure 17: Composition of Functors for Representing ATM System

The diagram in Figure 17 has the simple commutative requirement:

Grant-access = Funds-check ∘ Security.

It is not however a very explicit model of the ATM system as relationships, state changes and rules are not indicated.

A simple formal diagram with logic quantification (with the usual symbols for existential, universal and diagonal quantifiers)

and showing adjoint functors between a customer and a bank is shown in Figure 18. This expands Figure 17 to show a relationship in the adjoint functors between customer and bank of $\exists \dashv f^* \dashv \forall$. There are also the rules that need to be satisfied if the functors are to be adjoint. However, no state changes are indicated with this rather abstract formalism of the logic.

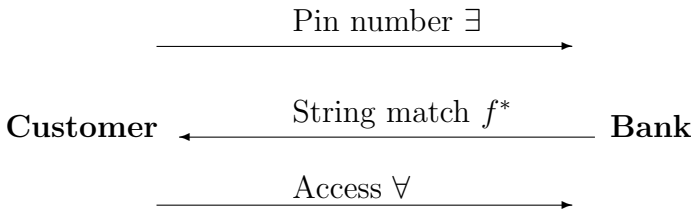

Figure 18: Adjoint Functors between Customer and Bank

8.2 Monad Approach

Applying the monad approach to the banking example results in the following BANKING construction:
$BANKING =< T, \eta, \mu >$ where

- $BANKING$ is the monad
- $T = GF$
- GF is a pair of adjoint functors $F \dashv G$
- F is the free functor **CUSTOMER** $\longrightarrow$ **BANK**, a request for funds
- G is the underlying functor **BANK** $\longrightarrow$ **CUSTOMER**, a status check
- η is the unit of adjunction $\eta : C \longrightarrow GF(C)$ for an object C in **CUSTOMER**

- μ is $GFGF(C) \longrightarrow GF(C)$, that is $T^2 \longrightarrow T$, also expressible as $G\epsilon F$ where ϵ is the counit of adjunction $\epsilon : FG(B) \longrightarrow B$ for an object B in **BANK**
- the diagrams shown in Figures 12 and 13, representing associative and unitary laws respectively, commute; the associativity law introduces $T\mu$, that is $T^3 \longrightarrow T^2$, also expressible as $GFG\epsilon F : GFGFGF(C) \longrightarrow GFGF(C)$.

The monad construction with adjointness provides the constraints described earlier of uniqueness and of non-mapping of η to $\top$, μ to $G\top F$ and $T\mu$ to $GFG\top F$.

The monad approach is more effective than either the composition or simple adjoint approach as it has all the required features: relationship between a customer and a bank through adjoints; specification of rules via constraints (units, counits of adjunction); and state changes between the cycles with μ. Compared to current database representations of transactions, the monad approach highlights the complexity of the ACID requirements, indicating that 'three cycles' (passes) are required ideally.

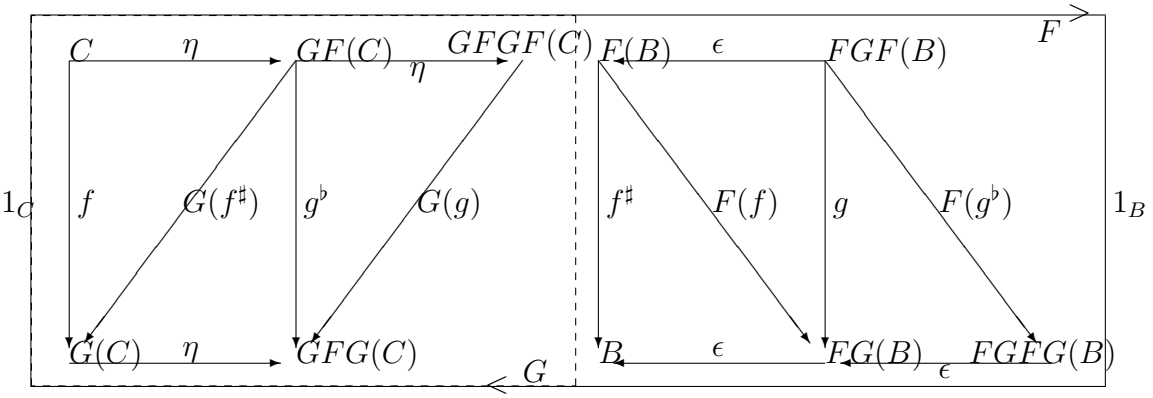

Figure 19: After 'two cycles' of the banking transaction where η maps on to other than $\perp$ and $\top$ maps on to other than ϵ. C is customer, B is bank, F is funds request, G is status check, 'transaction can complete'

Another advantage of the monad approach is that it can be readily combined with a detailed approach such as that shown earlier in Figure 11. For the banking application the diagram in Figures 19 shows details of the changes after 'two cycles' for the adjointness $F \dashv G$, where F is a funds request and G is a status check.

9 Universal Implications

The basic condition for a transaction processing system is that of naturality. Where there is a two-way functor system, the functors are adjoint in real-world systems. However, adjointness is typically achieved in 'one cycle'. Such a single cycle is not sufficient for full satisfaction of ACID principles. It is necessary to have 'two cycles' for achievement of Atomicity, Consistency and Isolation. For ultimate closure with Durability 'three cycles' are necessary as represented in the monad construction that wraps the 'three cycles' into one consistent transaction.

ACID is a deconstruction of a monad. A monad is an atomic entity, which commutes for consistency of identity and association and has independent existence in its own right for isolation. Durability results from its preservation under further processing. The convergence of database theory and category theory in process is striking. The two theories come from different viewpoints: ACID in databases *a posteriori* from commercial practice and monad in category theory from a relatively minor part of the pure theory. The convergence is presumably because both have to be natural.

10 Concluding Summary

Because adjointness is everywhere and the discussion has been wide-ranging we should perhaps conclude with a summary of the main theme of this paper of transaction in information systems as process. The main steps are:

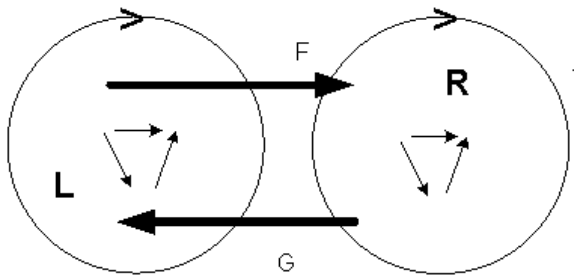

Figure 20: Adjointness between Two Systems

1 An information system has a left adjoint F i.e. right co-exact (physically represented) process which is uniquely coordinated by adjointness with a right adjoint G, left co-exact, underlying logical process $F \dashv G$ as in Figure 3 reproduced here as Figure 20.

2 An ATM banking transaction where the left-hand triangle composes with delivery of cash simultaneously debited from the customer's account lacks a left-adjoint functor $F \dashv G$ in a universal Turing machine and therefore on any electronic digital computer with a von Neumann architecture. However a true quantum computer would not be so restricted.

3 Commercial practice seeks to model the adjointness with a sequential process that adheres to the principle of ACID.

4 The ACID principles which evolve out of good commercial practice can themselves be shown to adhere to a deconstructed monad, a triple over adjointness, as independently developed by category theorists.

References

[1] Anderson, J R, *Security engineering: A guide to building dependable distributed systems* ed., John Wiley (2001).

[2] Barr, Michael & Wells, Charles, *Toposes, Theories, and Triples* Springer-Verlag (1985).

[3] Barr, M, & Wells, C, *Category Theory for Computing Science* third edition, Centre de recherches mathématiques, Montreal 526pp (1999).

[4] Barrow, John, & Tipler, Frank, *The Anthropic Cosmological Principle* Clarendon Press (1986).

[5] Bastin, E, & Kilmister, C, Concept of Order. I. The Space-time Structure, *Proc Camb Phil Soc* **50** 278-286 (1954).

[6] Connolly, T, & Begg, C, *Database Systems* Pearson (2004).

[7] Dembski, W A, & Ruse, M, Debating Design: From Darwin to DNA, Cambridge (2004).

[8] Deutsch, D, Ekert, A, & Lupacchini, R, Machines, Logic and Quantum Physics, *Bull Association Symbolic Logic* **3**(3) 265-283 (2000).

[9] Feynman, R P, & Hibbs, A, *Quantum Mechanics and Path Integrals* McGraw Hill (1965).

[10] Godement, R, *Théorie des faisceaux*, Hermann, Appendix I (1958).

[11] Gray, Jim, The Transaction Concept, Virtues And Limitations, *Proc 7th VLDB* Cannes, France 144-154 (1981).

[12] Gray, Jim, Thousands of DebitCredit Transactions-Per-Second: Easy and Inexpensive, *Microsoft Technical Report Series* MSR-TR-2005-39 (2005).

[13] Heather, M A, & Rossiter, B N, Locality, Weak or Strong Anticipation and Quantum Computing I. Non-locality in Quantum Theory, *International J Comp Anticipatory Sys* **13** 307-326 (2002).

[14] Heather, M A, & Rossiter, B N, The Universe as a freely generated Information System, *Against Bull ANPA* **26** 357-388 (2005).

[15] Heather, Michael, & Rossiter, Nick, The Logic of Foundations and the Foundations of Logic, *UNILOG 2005, 1st World Congress and School on Universal Logic* Montreux, Switzerland, 26 March - 1 April p.69 (2005).

[16] Heather, Michael, & Rossiter, Nick, Kurt Gödel heralds the Age of Mathematical Enlightenment: The Demise of Number and the Axiomatic method, *Horizons of Truth: Logics, Foundations of Mathematics and the Quest for Understanding the Nature of Knowledge, Gödel Centenary 2006* 27-29 April, University of Vienna (2006).

[17] Heather, Michael, & Rossiter, Nick, Process Category Theory *Salzburg Process Philosophy Conference* July (2006).

[18] Kan, D M, Adjoint Functors *Trans Am Math Soc* **87** 294-329 (1958).

[19] Landauer, R, Information is physical, *Physics Today* May 23-29 (1991).

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

[21] Leinster, Tom, *Higher Operads, Higher Categories*, Cambridge (2004).

[22] Mac Lane, S, *Categories for the Working Mathematician*, 2nd ed, Springer-Verlag, New York (1998).

[23] Noyes, H P, A Short Introduction to Bit-String Physics, in *Merologies: Proc. ANPA 18* T Etter, ed, 21-61 (1997).

[24] Noyes, H P, Program Universe and Recent Cosmological Results *Proc 20th Alternative Natural Philosophy Association, Aspects II*, K.G. Bowden, Ed. 192-214; also available as SLAC-PUB-8030 (1999).

[25] OECD Guidelines for the Security of Information Systems: Towards a Culture of Security, *OECD Council* (2002).

[26] Rossiter, B N, & Heather, M A, Quantum Information Processing within the Quantum Processing of Information: The Theory and Practice Compared, *1st IEE Seminar QIP* 20 April, St Anne's College, Oxford University (2005).

[27] Rossiter, Nick, Heather, Michael & Nelson, David, A Natural Basis for Interoperability, *I-ESA'06, Interoperability for Enterprise Software and Applications Conference* University of Bordeaux, 22-24 March 2006. Preproceedings (CD, W11, 22 March 2006); proceedings LNCS Springer (2006).

[28] Rossiter, Nick, & Heather, Michael, Free and Open Systems Theory, *EMCSR- 2006, Cybernetics and Systems, 18th European Meeting on Cybernetics and Systems Research* University of Vienna, April 2006, Trappl, R, (ed) 1 27-32 (2006).

[29] Stephenson, P, Applying forensic techniques to information system risk management - first steps. *Computer Fraud & Security* 2003(12) 17-19 (2003).