Data Structures in Natural Computing: Databases as Weak or Strong Anticipatory Systems

B. N. Rossiter* and M. A. Heather†

*Informatics, Northumbria University, Newcastle upon Tyne NE1 8ST, UK
†Sutherland Building, Northumbria University, Newcastle upon Tyne NE1 8ST

Abstract. Information systems anticipate the real world. Classical databases store, organise and search collections of data of that real world but only as weak anticipatory information systems. This is because of the reductionism and normalisation needed to map the structuralism of natural data on to idealised machines with von Neumann architectures consisting of fixed instructions. Category theory developed as a formalism to explore the theoretical concept of naturality shows that methods like sketches arising from graph theory as only non-natural models of naturality cannot capture real-world structures for strong anticipatory information systems.
Databases need a schema of the natural world. Natural computing databases need the schema itself to be also natural. Natural computing methods including neural computers, evolutionary automata, molecular and nanocomputing and quantum computation have the potential to be strong. At present they are mainly at the stage of weak anticipatory systems.

Keywords: Naturality, Sketches, Natural Computing, Database Schema, Quantum Data Computation

1 INTRODUCTION

1.1 Computation as an Anticipatory System

The Universe as a natural computer calculates [65] its new configuration at every instant of Planck space-time [1]. This computation operates as the ultimate anticipatory system: every other [53] is but its pale reflection. Formally a strong anticipatory system is a proper subcategory; a weak anticipatory system is a local or small subcategory [32]. A classical computer is a weak anticipatory system where its output by means of software interprets the states of its hardware operating as a strong [53, 17, 18] anticipatory system. Weak anticipation can be very accurate even for non-linear phenomena like weather forecasting where the data capture and data processing can be on a large enough scale to contain the complexity. Where this is not possible (for instance in forecasting earthquakes) a weak anticipatory system in the form of a classical computer may not really be adequate.

A strong anticipatory system as part of the Universe operates with the same precision as the universe itself. It is characteristic of this precision that gives natural computing its potential superiority over the von Neumann architecture operating locally and with fixed instructions between bit cells. Today's classical information systems anticipate the real world in the way that they store, organise and search collections of real-world data. By a process of reductionism and normalisation the data is adapted to fit the bit cells and to be manipulated by the fixed instruction set of von Neumann machines. This is the function of databases which because of the reduction and normalisation act as weak anticipatory systems. The same distinction is to be found in experimental work in biology. Processes *in vivo* are naturally strong, those in *vitro* are naturally weak because they are locally constrained as a part of the reactive system of the environment.

[1] See Pagels [50] where however the Universe is treated more as a classical computer and thus a weak anticipatory system. As such by now according to Lloyd [42] the Universe could have factored a million bit number using classical methods or a 10^{60} bit number using Shor's quantum algorithm mentioned below.

1.2 Classical Data Structures

Classical information systems employ some suitable model to mediate between the data and the hardware. A database model is a representation of policies in a structured form according to some perceived view of reality.

The prime objective of a database management system (DBMS) is to make application programs independent of the physical structure of the data. To achieve this objective, the ANSI/SPARC architecture [60] may be used. A conceptual schema or model is defined as a global logical definition of the data structure. This schema relates to the internal (physical) definition by a mapping from the logical level to the physical level. The schema is protected from changes at the physical level by adjusting this mapping. Each user has a particular view (external schema) of the database which may be a restricted view. The architecture including the series of mappings shown in Figure 1 provides aspects such as security and logical data independence. From the early days of computing a number of well-known models have been employed using sets, trees and links to connect data in logical relationships. These have been developed into more elaborate relational, hierarchical, network and functional models. The way in which relationships are expressed is the main classifying feature of databases. In the relational model a key from one table (a foreign key) cites a primary key of another table. In the hierarchical, child-parent links provide the relationships. In the network the address of an object is included in another object to give a pointer-based approach.

Relationships are often performed in a separate process such as earlier Entity-Relationship Modelling or current Unified Modelling [9]. Normalisation is needed to verify schema design, particularly to relate key and non-key attributes. The levels, mappings and relationships all have to be integrated in a consistent database design.

More recent developments include database systems founded on the object-oriented paradigm. Such systems have a more open extensible type system enabling a more flexible approach to data management but their acceptance has been hindered by their lack of a formal basis in set theory.

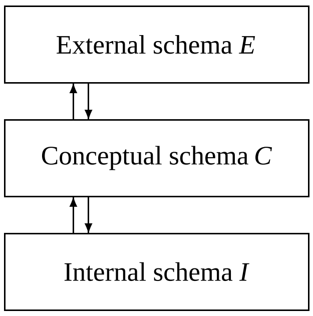

FIGURE 1. Classical ANSI/SPARC Architecture for Databases

1.3 Naturalised ANSI/SPARC

The classical ANSI/SPARC architecture of Figure 1 has the disadvantage that the levels are not independent of each other. This may be compared with the natural architecture of Figure 2 [33]. The four levels (top-down) are categories for concepts (real-world abstractions), constructs (facilities available for schema design), schema (definition of data types available in system) and data (the data itself). The types of the three levels are similar to the external schema but the internal schema is composed of subcategories of the conceptual schema. The top level, the external schema, is not a universal closure of types but a local closure of the conceptual schema. The four-level architecture in Figure 2 has orthogonal types with the relationships between the levels expressed as adjunctions as already applied to structures in GRID data processing [33]. Adjunctions relate one level to another. The relationship between levels is measurable by the unit of adjunction. For instance the adjunction $Policy \dashv MetaMeta$ indicates that the free functor $Policy$ is left adjoint to the underlying functor $MetaMeta$. The unit of adjunction is given by $\eta_{cpt} : 1_{cpt} \longrightarrow MetaMeta \circ Policy(cpt)$.

The ANSI/SPARC architecture was a useful way of capturing abstractions of the relational model in the 1970s and 1980s. It has proved less suitable to facilitate the techniques needed today such as interoperability where systems with different underlying models are required to work together. ANSI/SPARC can be viewed as pseudo-natural. It was developed using mathematical techniques and theories like sets. But there is a gap between classical theory and real-world performance and pragmatics. Triggers are an example of an attempt to patch the weakness of the system by providing some local strong anticipation using Event- Conditions-Actions (ECA) [14].

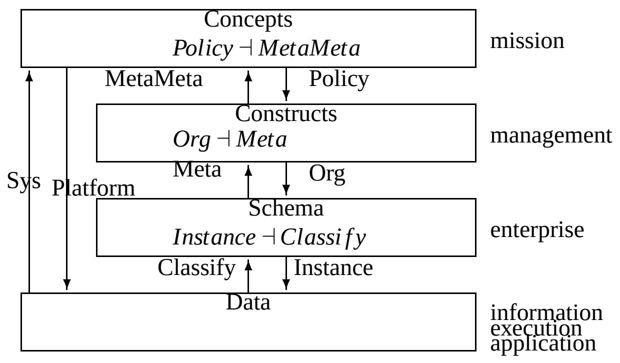

FIGURE 2. Interpretation of Levels: natural schema with strong anticipation

1.4 Natural Computing

Natural computing on the other hand involves data and operations as they are to be found in the natural world. This natural computing would be real-world processing needing not to rely on any model. Data would be input neat without any reductionist pre-processing. The corresponding information system or database using natural computing would therefore be strong anticipatory.

New developing areas promoted as natural computing include: quantum computation which exploits quantum mechanics principles in physics, nanoscale chemistry, bio- and molecular-computing processing as in genetics, Artificial Neural Nets (ANN) and genetic algorithms.

The word *natural* in natural computing does not have a settled meaning linguistically. It is necessary first to look at how the word is used to see if we can give it the rigour of a formal definition. A difficulty arises because the word natural is used analogously in weak anticipatory systems and not reserved for strong anticipation where it might otherwise be simply treated as a synonym. It is the purpose of this paper to review the main methods which might have some claim to be natural computing and then to examine them for data processing in the real world.

2 NATURALITY

Mathematics in western culture deals with collections and operations as its basic components like numbers and arithmetic, sets and functions, matrices and algebra. These may be identified almost at will in pure mathematics but in applied mathematics the collections and related operations need to have some manifestation in the real world to be of use. Data processing is likewise involved with collections and operations existing in the real world. Reality is a cognate word to natural and has similar problems in its definition.

The leading formalism for the last century has been axiomatic set theory promoted by mathematicians like Hilbert and Gödel as having an objective reality *sui generis*. Russell interpreted axioms as comparable to the laws of physics. Their justification lies in the effects they have on our sense perceptions. Although obviously aware of the undecidability of formal arithmetic operations Gödel took the view that constructive weaknesses in this ‘set realism’ could be corrected by additional axioms which have been investigated by Maddy [46] under the label of ‘axiom of constructability’. The effect of this additional axiom is to provide closure by asserting that the Zermelo-Fraenkel axioms taken with the axiom of choice are true. To counter any weakness in this position, Maddy has taken the step further with the conclusion to replace set realism with set naturalism [47], namely that the axioms are justified by the practice of mathematics and its usefulness. Rather strangely the whole of formal axiomatic set theory seems to rest on the ultimate basis that it works in practice. This stance is confirmed by Feferman [24] who highlights the role of the axiom of choice (Moore [48]). It is exercise of the choice that reduces strong anticipation to weak anticipation [31, 32].

The mathematics within the sights of Maddy and Feferman is concerned mostly with collections and operations that exist only in the mind. These can be very fruitful in so far as they have an existence in the world at least locally. The world need not be limited to the physical world to be natural. What goes on the human mind is still natural in some sense. This mathematics is weak anticipatory where it can be quite useful as the last three centuries of classical progress has shown. However the use of such mathematics to model strong anticipatory systems is much less certain. As Francis Bacon pointed out: a stream never rises higher than its first spring. Natural computing on any interpretation will be the operation of a strong anticipatory system. However a concentration on the real world does not mean that we can neglect the theory. Rather we need a more rigorous natural theory. However, to investigate them as practical systems requires close scrutiny of their theory as viable systems logically needed as a precursor to any thought being given to any hardware implementation. As Lenin remarked ‘theory without practice is pointless and practice without theory is mindless’. So we will begin with theories of data processing that we need to apply.

Database management systems as discussed above are classical and weak. Strong versions cannot be ’built on sand’ as Hermann Weyl [2] described axiomatic systems. Category theory, the general formalism sufficiently developed which we have available to us, fortunately is built on a formal concept of naturality in the sense of natural isomorphism.

2.1 Natural Isomorphism

Category theory was originally developed in the 1940s to explore naturality [43]. Drawing together algebra, geometry and topology it provides a mathematical workspace where the notion of natural existence can be rigorously represented. Concepts like ’same’ or ‘equality’ present some difficulty whether expressed in words or symbols, even the very familiar $A = B$ may not be properly defined. There are two components: existence and similarity. Category theory has the advantage that there is only one fundamental symbol that of the arrow and whose use can therefore be much more easily controlled and concepts rigorously defined by usage. The fundamental form of existence corresponding to what is loosely referred to as ‘same’ is isomorphism.

Isomorphism identifies by the existence of an inverse arrow. This is a categorical manifestation of naturalness in the real world – of a natural identity. So when we come to apply this to the world of data processing, we are dealing with naturally identified structures where there is an inverse. So for instance a person is identified as the same person by their <u>intension</u> even though their <u>extension</u> changes for it is always possible to apply a simple inverse to confirm that it is the same person. There is no simple inverse of the extension naturally. This would require rebuilding the person collecting together all the component parts from around the real world. Intension/extension distinction is fundamental to data structure and inherent in natural systems and therefore essential to the processing of natural data. Most DBMS have a basic ‘entity’ corresponding to the intensional form. The object-oriented movement recognises the concept of the object and open internal structure to such an entity which characterises the non-atomic nature of modern data. This is typical of new types of data such as genes. A prime example is the personal DNA in the way that it propagates the particular characteristics of a person.

$$\begin{tikzcd}
& A \arrow[dl, "f"'] \arrow[dr, "gf"] & \\
B \arrow[rr, "g"] \arrow[dr, "fg"'] & & A \arrow[dl, "f"] \\
& B &
\end{tikzcd}$$

FIGURE 3. Composition of Identity Triangles

For objects A, B related by arrows f, g as:

[2] “daßjenes Haus zu einem wesentlichen Teil auf Sand gebaut est”, Das Kontinuum [64], Vorwort p.iv

$$f : A \longrightarrow B \qquad g : B \longrightarrow A$$

then there is an isomorphism between A and B when f, g are mutually inverse. This may be written as $A = B$ and $B = A$. The proof diagram for this is Figure 3 which defines $1_A = gf$ and $1_B = fg$ to describe the isomorphism of A and B by $A = B$ and $B = A$ and $fg = gf$ [3]. This diagram is universal in the sense that it also holds when A, B are categories (conventionally printed **A**, **B** in bold in the category of sets and in gothic for large categories or classes) and f, g are functors (conventionally F, G). Furthermore if in Figure 3 A, B are functors and f, g are natural transformations (conventionally ϕ, γ) $fg = gf$ are the identity natural transformations $(1_\phi, 1_\gamma)$ (Johnstone [38] p.247-248 A1.1.7-1.1.8) [4]. The result is no different if ordinary A, B represent natural transformations. This is natural isomorphism which operates as a natural closure.

2.2 Natural Closure

$$\begin{tikzcd}
 & \mathbf{CST} \arrow[r, "O"', ""{name=O}] & \mathbf{SCH} \arrow[r, "I"', ""{name=I}] & \mathbf{DAT} \\
\mathbf{CPT} \arrow[ur, "P"', ""{name=P}] \arrow[dr, "P'", ""{name=Pp}] & & & \\
 & \mathbf{CST'} \arrow[r, "O'", ""{name=Op}] & \mathbf{SCH'} \arrow[r, "I'", ""{name=Ip}] & \mathbf{DAT'}
\arrow[from=P, to=Pp, "\alpha"]
\arrow[from=O, to=Op, "\beta"]
\arrow[from=I, to=Ip, "\gamma"]
\end{tikzcd}$$

FIGURE 4. Comparison of Mappings in two Systems

To summarise therefore, four levels are required to define an arrow as unique up to natural isomorphism. The four levels are: 1) object or identity arrow (within a category), 2) category (comparing objects), 3) functor (comparing categories) and 4) natural transformation (comparing functors). No more levels are required.

The relationships between one four-level architecture and another can be constructed as in Figure 4, the expanded view of Figure 2. Here for simplicity the mappings are viewed in one direction only. Two systems are compared, one involving categories **CPT**, **CST**, **SCH** and **DAT**, the other **CPT**, **CST**′, **SCH**′ and **DAT**′, representing concepts (**CPT**), constructs (**CST**), schema (**SCH**) and data (**DAT**) from Figure 2. **CPT** is the same in both systems as there is one universal type for concepts. As usual the functors relate the categories. We have now though added natural transformations to relate the mapping between one functor and another. It needs to be emphasised that none of these categories are discrete: all have an internal arrow-based structure so the natural transformations are non-trivial [55]. The functors need to be of the same variance for a meaningful natural transformation to exist between them and this is the case for α, β and γ.

An arrow comparing natural transformations is itself a natural transformation. Some categorists use an older terminology with degrees of ‘cell’ and describe the identity arrow as 0-cell, an arrow in a category as 1-cell and an arrow between arrows as a 2-cell [39]. The figures 5 and 6 show what happens when an arrow maps between two natural transformations. So an arrow from one natural transformation to another gives a composition of the natural transformations, not a new level ([7], at p.85). This means that four levels are needed to give the natural closure [33].

Two squares, derived from Figure 4, are shown. Figure 5(a) commutes for each arrow $f : cpt \longrightarrow cst$ if α is a natural transformation. Similarly Figure 5(b) must commute for each arrow $g : cst \longrightarrow sch$ if β is to be a natural transformation. Note that viewed in this way a natural transformation is not a layer above functors and functions. The levels are interwoven with natural transformations determining how every arrow defined at the lowest level is mapped.

Now if we write the arrow: $\delta : \alpha \longrightarrow \beta$, we can see that δ is a composition $\beta \circ \alpha$ giving the commuting square shown in Figure 6 ([7], at p.85). Theory suggests that four strong-anticipatory levels of computational types are sufficient to provide ultimate systemic closure in unique strong anticipation. Between the four levels are three layers of adjoint

[3] It was for this reason that the early categorists referred to abelian categories [25] where statements were identically true up to natural isomorphism if they held in the dual category obtained by reversing the arrows.

[4] There is a typographical error in the diagram on p.247 of Johnstone [38]. The final B should be A.

functors that relate each type-pair. A free functor allows selection of a target type at a lower level and its right adjoint determines the higher-level type. Because of the uniqueness a higher-level anticipates a lower level and a lower level a higher. Between each level is a functor.

$$\begin{tikzcd}
P(cpt) \arrow[r, "\alpha_{cpt}"] \arrow[d, "P(f)"'] & P'(cpt) \arrow[d, "P'(f)"] & O(cst) \arrow[r, "\beta_{cst}"] \arrow[d, "O(g)"'] & O'(cst') \arrow[d, "O'(g)"] \\
P(cst) \arrow[r, "\alpha_{cst}"'] & P'(cst') & O(sch) \arrow[r, "\beta_{sch}"'] & O'(sch')
\end{tikzcd}$$

FIGURE 5. Commuting Target Square for Natural Transformations (a) $\alpha : P \longrightarrow P'$, comparing policies in two systems; (b) $\beta : O \longrightarrow O'$, comparing use of constructs in two systems

$$\begin{tikzcd}
P(cpt) \arrow[r, "\alpha_{cpt}"] \arrow[d, "P(f)"'] & P'(cpt) \arrow[d, "P'(f)"] \\
O(cst) \arrow[r, "\beta_{cst}"] \arrow[d, "O(g)"'] & O'(cst') \arrow[d, "O'(g)"] \\
O(sch) \arrow[r, "\beta_{sch}"'] & O'(sch')
\end{tikzcd}$$

FIGURE 6. Commuting Target Square for Natural Transformation composition $\beta \circ \alpha$, comparing mapping from concepts to schema in two systems

An alternative view to Figure 4, shown in Figure 7, is closer to the four levels inherent in category theory. The fundamental levels are considered to be data values, named values, classified values and contrasted representation corresponding in category theory to object, category, functor and natural transformation respectively. The natural transformations are now the duals of those shown earlier in Figure 4 as indicated by the *op* superscript. The earlier natural transformations were comparing the downward functorial mapping (towards data) while the current ones compare the upward mapping (away from data) [56].

alternative fundamental levels	category theory levels	four levels of Figure 4
1. data values	objects (identity arrows)	id_{dat}
2. named values	category	**DAT**
3. classified values	functor	C : **DAT** $\longrightarrow$ **SCH**
4. contrasted representation	natural transformation	$\alpha^{op} \circ \beta^{op}$

FIGURE 7. Alternative Interpretation of Levels in the Architecture

It can be seen that the addition of further levels is possible but nothing is gained by it type-wise. Thus addition of an extra level to the top of a four-level architecture simply results in the top level (comparison of mapping from concepts to schema) being a composition of three arrows rather than two. Thus consider the addition of a new top level **PHI** with the mappings F : **PHI** $\longrightarrow$ **CPT**, G : **CPT** $\longrightarrow$ **PHI** and $\alpha^{op\prime} : F \longrightarrow F'$ where $\alpha^{op\prime}$ compares the mappings F and F' in two different approaches. The adjoint is now $I \circ O \circ P \circ F \dashv G \circ A \circ M \circ C$. The level four of Figure 7 is now $\alpha^{op\prime} \circ \alpha^{op} \circ \beta^{op}$ and is still a natural transformation through the rules of composition. The practical consequence is that a fifth level is equivalent to an alternative fourth level. The meta-meta level gives ultimate closure of types.

2.3 Natural Calculus

We therefore have three types of mapping to consider: within a category (for instance from a name to a value), from one category to another (for instance the functor P' from **CPT** to **CST**′) and from one functor to another (for instance the natural transformation α from P to P').

Following the constructive principles of category theory, the composition of these arrows is natural. This consequently gives rise to a natural calculus first expounded by Godement ([27]; [7], pp 94-97) in the form of rules governing composition. The composition of functors and natural transformations is associative so that for instance:

$$(I'O')\alpha = I'(O'\alpha); \quad \gamma(OP) = (\gamma O)P$$

Natural transformations may be composed with each other:

$$\gamma\beta = (\gamma O) \circ (I'\beta); \quad \beta\alpha = (\beta P) \circ (O'\alpha)$$

The consequence of this is that a categorical approach ensures that the various arrows of different types can be composed with each other, irrespective of their level in the system. Equations can be solved, representing an equality of paths, with unknown components that can be determined from an evaluation of the known properties. For instance in comparing methods with the path IOP from **CPT** $\longrightarrow$ **CST** $\longrightarrow$ **SCH** $\longrightarrow$ **DAT** defining one approach, then the path $I'O'\alpha$ from **CPT** $\longrightarrow$ **CST**′ $\longrightarrow$ **SCH**′ $\longrightarrow$ **DAT**′ might define an alternative approach if P' maps onto constructs in the category **CST**′.

2.4 Natural Logic

The identity natural transformation, at the limit for natural isomorphism, is in an applied context a topos. It has been shown that the internal logic of a topos is intuitionistic i.e. Heyting [45, 38].

The world as a topos has intuitionistic logic. The internal logic of the topos is Heyting:

a entails $\neg\neg a$ but $\neg\neg a$ may not entail a

Every Boolean logic is a Heyting logic but not every Heyting logic is Boolean. For an open system category C, object A implies object B:

Heyting implication
classically represented by
$$\boxed{\frac{C \wedge A \leq B}{\overline{C \leq (A \Rightarrow B)}}}$$

Although Heyting logic has been known for some time its possible great significance as the fundamental underlying logic of the world is only now just emerging [5] because it is the natural internal structure of a topos.

[5] [44]; it is to be noted that Mac Lane's *Category Theory for the Working Mathematician* does not deal with Heyting logic even in his second edition although it was referred to in [45] in 1991.

3 NATURAL SCHEMA

3.1 Natural Number Object

The natural numbers are the foundation of most classical systems for computation including numerical indexing and data processing. The collection consists of a sequence of abstract labels and the operations are those of arithmetic. The natural numbers can be built up to give more elaborate structures like the complex, quarternion, octonion, p-adic, Conway and even surreal numbers as well as matrices, spaces from vector up to Hilbert spaces and beyond. Because of their infinite nature the collection needs to rest on some philosophic basis such as Platonism and the fundamental operation of the axiom of choice. It is rather a paradox that natural numbers are only a weak anticipatory system. Because of the choice of definition of the unit the relationship between ordinality and cardinality is not unique. Cardinality is a weak anticipatory system of ordinality.

Because category theory represents naturality in the strong anticipatory sense, it does not therefore have an inherent concept of natural numbers and they are not to be found in the topos. However, because of the importance of natural numbers for the modelling of properties of weak anticipatory systems, categorists introduce the natural number object from a suggestion by Lawvere [41] as categorical formulation of the axiom of infinity ([38] p.108). The natural number object in a category **C** is defined by the following commuting diagram in Figure 8. The terminal object 1 identifies a first natural number object by the arrow $zero\ o : 1 \longrightarrow N$. Successors are then identified by $s : N \longrightarrow N$ so that there is a $1:1$ relationship between each natural object N and some object A in the category.

The successor function picks out the next copy of itself. This gives recursion. However, it is weak and appears inadequate even for applications in pure mathematics where the more elaborate version involving a slice category seems more popular [7]. Johnstone defines the general case for recursion of the natural number object in a cartesian closed category ([38] p.108 *sqq*). Enriching a weak anticipatory system cannot in itself convert weak to a strong anticipatory system. In applied categories in general and to deal with natural computing in particular we cannot rely on this restricted approach [6]. Natural numbers always need to be treated with great caution in real-world data processing.

$$\begin{tikzcd}
 & \mathbf{N} \arrow[r, "s"] \arrow[dd, "g"'] & \mathbf{N} \arrow[dd, "g"'] \\
1 \arrow[ur, "zero"] \arrow[dr, "g_0"'] & & \\
 & A \arrow[r, "f"] & A
\end{tikzcd}$$

FIGURE 8. Natural Number Object

Information systems anticipate the real world. Databases store, organise and search collections of real-world data. The conceptual organisation of data according to its inherent logical structure as a database was early recognised in computing as an efficient method for the storage and retrieval of information. In terms of anticipatory systems, information systems as databases constructed using classical methods are weak anticipatory. This is because of the reductionism and normalisation needed to operate databases on the familiar but idealised machines with von Neumann architectures of fixed instructions between bit cells.

In the first instance category theory was devised (around 1945) to explain why certain manipulations are 'natural' and others are not. This uncovered the notion of a natural transformation as the appropriate way to compare functors.

[6] For instance in quantum processing a copy is not a natural concept.

3.2 Reducing and Rationalising Naturality

A normalised structure in databases is one that satisfies particular requirements as to its form and the relationship between key and non-key attributes. Normalisation rules often seek certain positive features and the absence of other undesirable features.

$$\begin{tikzcd}
A \arrow[r, "y"] \arrow[d, "u"'] \arrow[dr, "z"] & B \arrow[d, "x"] \arrow[dl, phantom, "\oplus", pos=0.3] \\
C \arrow[r, "t"] & D
\end{tikzcd}$$

FIGURE 9. Punctured Commuting Diagram

A commuting diagram is itself a proof of some naturality. Figure 8 is a commuting diagram of a weak naturality but it should still guarantee consistency subject to any assumptions already invoked. An even weaker version of the commuting diagram may be represented by the punctured diagram promoted by Freyd ([26] section 1.251) as in Figure 9 where the puncture mark $\oplus$ removes the commutation of the right-hand triangle although it retains the commutation of the left-hand triangle and the commutation of the rectangle as a whole. That is the puncture sign $\oplus$ removes the equivalent of one equation. In data processing there are examples where this might be used. For instance the problem of representing partial functions. If in Figure 9 A is a library stock, B is the book issue, C is the catalogue, D is accession number, then $tu = z$ and $xy = z$ for issued books but z relates the library to the books. The outer diagram still commutes in the sense that it is still possible to go from A the library through C the catalogue to D the accession numbers. In that sense the outer rectangle commutes but it could be more confusing than illuminating in practical implementations. In this example the punctured diagram has not unequivocally satisfied the requirement of most current on-line library catalogues which need to display whether the book is in the library or out on issue.

The usual approach in relational databases to match the natural data structure to one of a number of standard types is to adopt a process of normalisation expressed in some level of normal form relative to some search/storage key as in Figure 10. These are the normalisation processes for databases relying on the relational model. Other models may use other approximation techniques. For example inheritance is used in hierarchical or network systems and 1NF is bypassed in object based databases where multiple values may occur for an attribute. These may then give rise to mismatch in interoperability between models relying on different rationalisation procedures.

Natural computing needs to operate without any kind of normalisation. Besides these very specific database normalisations, there are other kinds with which there may be interaction. For instance programming languages are often described in BNF [7]. In logic there are normal forms and Skolem functions related to Horn clauses equivalent to a universal Turing machine [1]. In natural language processing formal grammars have been normalised in types 0,1,2,3 by Chomsky [13].

In Rosen's terms data in the world is a reactive system anticipated by normalisation and within Dubois' distribution of weak or strong [18], the normal form is a weak anticipatory system of the data structure to be ascertained by the application of some search method such as SQL. The use of natural numbers for normal forms is a reminder that as anticipatory systems these are weak. The collections are simplified by the process of normalisation but there are still more complex operations on them such as JOIN, PROJECT and SELECT [14]. At first sight hierarchical and network structures might seem more natural than in the relational model. Connections are more explicit and there are natural paths established for navigating data e.g. child to parent links. Hierarchical is false natural and is weak. Network is a

[7] BNF is sometimes known as Backus-Naur Form from the modification by Peter Naur to the original version of Backus Normal Form of John Backus [63].

Normal form	explanation
1	all values are atomic (non-nested)
2	non-key attributes are a function only of the primary key and not any sub-key
3	non-key attributes are mutually independent
BCNF	all sources of dependency are possible keys
4	multi-valued dependencies are reduced to equivalent functional dependencies
5	cyclic dependencies are reduced to acyclic
PJNF	tuples preserved under projection and join

FIGURE 10. Adapting naturality to formal structure

full partial order so may be strong but is heavily constrained in practice and so therefore weak. Serialisation of parallel data transactions is a reductionism and rationalisation of the real-world.

3.3 Sketches

A more sustained use of category theory as a theoretical basis for databases has been pursued using formal sketches as an adaption of graph theory. Data processing in natural computers provides a good theoretical testing ground for the approach. The sketch was originally presented by Charles Ehresmann [21] as a culmination of work on structured categories, in particular the notion of an internal category in any category under the name of *catégorie structurée généralisée* [19]. Wells [62] claimed that the word *sketch* is used with at least three meanings:

1. a structure which is a weakening of the concept of category (the composite may not be defined for all composable pairs) plus specified cones and/or cocones (Ehresmann [22]).
2. a graph with specified cones and/or cocones plus some commutativity conditions on paths (Barr [7], Wells [61]).
3. a category with specified cones and/or cocones.

Apparently at the time Ehresmann had in mind the need for a formal representation of Plato's ideals but thought that this could be achieved by relaxing some categorical requirements. The first two senses above follow Ehresmann's general approach [21]. The third sense [49] develops small, complete *accessible categories* as a mapping from sketches to a 2-cell categorial model. It is the first two that have been employed in work on database theory. In the database area sketches are at first sight appealing as they seem to match categorical concepts to those of design practice in conventional information systems in entity-relationship (ER) modelling. There is then a well-established route, as simple as the use of a function key, from a complete ER model into a definition for a relational database. Sketches mirror the reductionism in formal models for relational databases in that sketches permit non-commuting diagrams. This corresponds to database practice of 1990s.

Not all of the features of the relational model can be handled by sketches [36]. The normalization concept up to third normal form is assumed in the sketch construction rather than being integrated and tested. The restriction of a primary key in a sketch to one attribute is also unrealistic since relations involving associations always have at least two attributes for the primary key, each identifying a participating type. The claim by Johnson *et al* in [36] that the sketch construction is superior to the relational model in aspects such as view updating may be true for the SQL standard implementation but not for the underlying model itself [14].

Sketches can represent schema construction: graphs match the architecture, commuting diagrams match constraints; the cones match relationships; the cocones match attribute collections. There are many variants of sketches. Wells [61] identifies 12 different kinds and Johnstone ([38] D2.1.3 **2** p.863-864) eight. The number of different kinds suggests a lack of consensus as to the fundamental concept. Johnstone considers sketches to be a digression in his development of category theory. He also notes a lack of flexibility:

> "And sketches share with theories the disadvantage of 'signature-dependence': equivalence of sketches is a much stronger condition than the assertion that they have equivalent categories of models in all appropriate categories" ([38] D2.1 **2** p.861- 862).

Johnstone is referring here to the tight correspondence between a signature of a sketch and its subsequent model. The signature comprising functions, relations and constants is defined when the sketch is established. Subsequent changes to the signature, e.g. through adding new functions as in the object-oriented paradigm, require a redefinition of the sketch and a re-working of the model functor. The sketch/model construction is suited more to static type systems such as the relational model than to the extensible object-oriented approach.

The Finite Discrete Disjunctive kind (item (e) in Peter Johnstone's list) has been used by Michael Johnson and his co-workers [37] for relational databases. In their context a sketch D is defined as a 4-tuple $< E, L, R, S >$ where E is a finite graph for the data structure, L is a set of diagrams in E giving the constraints, R is a finite set of discrete cones in E giving the relationships and S a finite set of discrete cocones in E specifying the attributes. Sketches have also been employed by Diskin *et al* [16] for modelling information systems, in particular again the ER model. His use of sketches follows the method identified by Johnstone as type (b) i.e. finitary product sketches in which S is empty and all the cones in R are discrete. Although Diskin compares his use of sketches with those of Makkai [49], his sketches are much less constrained: Makkai is more concerned with provability, Diskin with expressibility. Diskin develops the concept of markers ([16], section 5.2) which are constraints such as identity functions or relationship arcs, treated as diagram predicates. Diagram transformations are used extensively with inputs of an operation converted to outputs embedding the operation in a commuting diagram.

A model functor M can be applied to the graph E in a sketch D by a sketch homomorphism $M : E \longrightarrow C$ to map any finite graph E to a category C. Because of the simple graph-based nature of E, C is a small category so the sketch approach is very much set-based. M preserves graph morphisms in L so that $M(d)$ commutes for every diagram $d : E' \longrightarrow E$ in E. M also preserves limit cones in R in $M(R)$ and colimit cones S in $M(S)$. In the work of Johnson, Rosebrugh and Wood [37] the sketch is the intension and the model functor the database state.

Barr & Wells ([7] p.109 4.7.6, 4.7.7) state that a "model M of a sketch with constants is called a term model if for every node a of the underlying graph, every element of $M(a)$ is reachable by beginning with constants and applying various operations (arrows of the sketch)". The natural numbers are the unique, up to natural isomorphism, initial model. This shows that the foundation for the model functor is natural numbers with a consequent limitation to the representation capability of that concept. Johnstone notes that "the more detailed the sketch, the less value is the model". This presumably refers to the model functor being excessively constrained by an over detailed sketch construction.

It is possible to view sketches in a similar way to set-based approaches such as Hilbert spaces in quantum computation. Both are models of models. It is not possible to recover enrichments that are lost. A categorical model of a set theoretic model cannot recover what is lost in the assumptions of the set theoretic model. The model is weak anticipatory as it is based on the natural number object.

Sketches are weak anticipatory as all structures and constraints have to be pre-specified. In difficult areas such as interoperability sketches are inadequate as they do not offer natural closure. The difference between a natural structure and a sketch is like that between typing and labelling. A graph is richer than an ER model as its arrows are typed with identity functors. Labelling in the ER model is an informal typing whereas the identity arrow is a formal typing.

4 APPLICATIONS

4.1 Parallelism

Early progress with data processing on analogue computers was quickly surpassed in the middle of the last century by the serial digital methods based on the classical von Neumann architecture. Nevertheless interest in alternative approaches continued through the latter part of the century particularly in the use of parallelism. Attempts in the 1980s like the DAP [51] to produce parallel machines based on digital hardware of the time failed to compete with the power of the serial machines because of the complexity issues inherent in parallel methods [8]. The naturality of real-world processing of data is mainly in parallel but serial digital methods can perform surprisingly well with appropriate software to perform tasks in parallel. For databases this includes the collection of parallel data, maintenance of mirror sites with appropriate version management, searching databases in parallel although usually where the parallelism is only homogeneous. These two types are sometimes classified as single instruction multiple data (SIMD) and multiple

[8] particularly in the interface between the syntactic and semantic processing of the data.

instruction multiple data (MIMD) but this may over-simplify the real world whose processes involve heterogeneous parallel operations on parallel heterogeneous data. This is the full naturality to be found in Figures 5 and 6 discussed previously above under *Natural Closure*.

4.2 Natural Architectures

Over the years there have been a number of processing initiatives with alternative architectures with some elements of parallelism and by the 1990s they have formed a general class referred to under the umbrella of ‘natural computing’. In the context of data processing this may divide into four subclasses. 1. neural computers mimic the 10^{12} neurons directly interconnectable on a dendritic tree in the brain and nervous system. 2. evolutionary automata calculate by natural selection possible paths for development as in biology. 3. Molecular and nanotechnology on the other hand rely on comparable processes in chemistry to compute the viable from all possible physical configurations. 4. Quantum computation exploits the parallel property of entanglement.

There is the further constructive (Brown [11] Chapter 8) characteristic of natural world parallelism that it is self-consistent. For instance cellular automata are in Rosen’s terminology [54] reactive systems but with a much simplified version of the laws of physics operating between cells. In von Neumann’s self-reproducing cellular automata the assumption has to be made that the underlying rules are self-consistent [59]. Data applications have already begun using neural and evolutionary methods [23] for data classification with potential for data mining and network analysis or synthesis for use in schedules, timetables and financial and business models [40].

Molecular computing on the other hand seems very much restricted to heavy numerical data processing [52] and therefore a prime candidate for applications in cryptography, where it is claimed [5] that a DNA system is capable of breaking the common DES (Data Encryption Standard). A DNA molecule composed from the four bases ATCG (Adenine, Thymine, Cytosine and Guanine where the first and last pairs respectively complement each other) in some sequence can be treated as a bit-stream memory complex. The processing of these can be halved by using the Sticker model [12] which treats subsequences of the bases as a bit value 1 or its complementary subsequence (a sticker) as a bit value 0. Subsequences are unique and it is therefore possible to construct a programming language of simple logical operations with a XOR gate. All the strands in a test tube will be processed together giving rise to massive data parallel processing. It is estimated that by utilising a library, with 20 oligonucleotide length memory strands, and an overall memory strand of 11,580 nucleotides it should be possible to break a DES with about four months of laboratory work [35]. This is in effect very promising for non-silicon replacement of classical computing. However while the process may be called natural because it involves chemical reactions in test tubes the structure of the data will nevertheless still be in conventional form with the only advantage so far of an increase in speed rather than as a closer approach to naturality.

Following the seminal work of Deutsch [15], Shor [58] was able to produce a quantum version of the fast Fourier transform requiring only n^2 rather than $(2n)^n$ steps. Grover [28] developed this idea of using quantum algorithms for faster searching of databases. Selinger [57] has produced a collection of operations at such a level that they could form the basis of a quantum programming language. Both offer potential for the development of quantum databases. In databases Grover’s transform method of 1996 flips a matrix in such a way that the odd record out (the one sought) is identified through amplitude amplification. The method of Grover has been extended for multiple occurrences but the number of occurrences needs to be known in advance [10].

If databases in general require a conceptual level for the representation, querying and updating of data, the questions needs to be posed: wherein lies the typing and schema in quantum data processing? These of course reside in the inherent natural structure of the data which is then preprocessed by normalisation as already discussed in conventional databases. True natural computing needs no reductionism but to rely on the rationalism that is already in the data. Shor’s algorithm is a look up of a natural number type. For this same reason an ordinary computer can be programmed to model the operation of a quantum computer with Grover’s algorithms [29] but only as a weak classical model of quantum processing and therefore very inefficiently. It will also be true to say that any of the four types of natural computing listed above may be modelled on traditional computers which may be illuminating but can have no predictive power beyond the weak anticipatory system.

As pointed out in the review by Aharonov [4] the Grover iteration can be understood as a product of two reflections. In this way Bhattacharya *et al* [8] have implemented a quantum search algorithm showing that classical waves can search a N-item database just as efficiently. It is claimed that although the lack of quantum entanglement limits the database size, entanglement is neither necessary for the algorithm itself, nor for its efficiency. The present approach relies on low-level operations analogous to classical methods like the CNOT gate (controlled NOT gate) where two input qubits, control and target, are XOR'd and stored in a target superposed qubit. Grover makes use of the 'oracle', treated as a black box and used for collapsing the wave function, that is to determine when a solution has been derived. However, this form of the oracle may lack non-locality [34].

The present position in data processing by natural computing may be summed up in the table below in Figure 11 by classifying each of the four subclasses in respect of their power as weak or strong anticipatory systems. There are the two separate formal components as discussed above – *collections* and *operations*. The former therefore relates to whether the data structure is natural and the latter to whether the processing is natural.

	anticipatory information systems	
subclass	structure	processing
neural computers	strong	weak
evolutionary automata	weak	weak
molecular and nanocomputing	weak	strong
quantum computation	weak	strong

FIGURE 11. Classification of Subclasses as Anticipatory Information Systems

5 CONCLUSIONS

It is hardly news that there is more to the real world than might seem from first impressions. The theory should not surprise us that naturality takes us beyond a simple classical world. Whether in the human physiology of the brain, in biology, in chemistry or in physics there is the same distinctive naturality. Each has its own respective representative version of what must be some ultimate form of natural computation – the strong anticipatory information system that is the Universe. From a database perspective it is clearly still early days where we are operating mainly within weak systems but these should only encourage us to break through the classical ceiling.

REFERENCES

1. Aaby, A, *Horn Clause Logic*, Walla Walla College, WA, http://cs.wwc.edu/ aabyan/Logic/Horn.html (1999).
2. Aarts, E H L, & Lenstra, J K, (edd) *Local Search in Combinatorial Optimisation. Discrete Mathematics and Optimisation* Wiley Chichester (1997).
3. Adleman, L, On constructing a molecular computer, DNA based computers, Lipton, R, and Baum, E, (edd), *DIMACS*, American Mathematical Society 1-21 (1996).
4. Aharonov, D, Quantum Computation – a review, in: Stauffer, D, (ed), *Ann Rev Comp Phys VI World Scientific* (1998).
5. Amos, M, Paun, G, Rozenberg, G, & Salomaa, A, *Theoretical Computer Science* **287**(1) 3-38 (2002).
6. Baeyer, Hans Christian von, *Information*, Weidenfeld & Nicolson, London (2003).
7. Barr, M, & Wells, C, *Category Theory for Computing Science* (1990).
8. Bhattacharya, N, van den Heuvell, H B, and Spreeuw, R J C, Implementation of quantum search algorithm using classical Fourier optics, *Phys Rev Lett* **88** 137901 (2002).
9. Booch, G, Rumbaugh, J, and Jacobson, I, *The Unified Modelling Language - User Guide*, Addison Wesley (1999).
10. Boyer, M, Brassard, G, Hyer, P, & Tapp, A, Tight Bounds on Quantum Searching, *Fortsch Phys Prog Phys* **46** 493-505 (1998).
11. Brown, J R, *Philosophy of Mathematics, An introduction to the world of proofs and pictures*, Routledge (1999).
12. Carroll, Steve, A Programming Environment for DNA Computing, *First Workshop on Non-Silicon Computing*, Cambridge, Massachusetts, 8th International Symposium on High Performance Computer Architecture (HPCA-8) (February 2002).
13. Chomsky, Noam, Three Models for the Description of Language, *IRE Transactions on Information Theory* **2** 113-124 (1956).
14. Date, C J, & Hugh Darwen, *Foundation for Future Database Systems: The Third Manifesto* 2nd Ed, Addison Wesley(2000).
15. Deutsch, D, and Jozsa, R, Rapid solution of problems by quantum computation, *Proc Roy Soc Lond A* **439** 553-558 (1992).
16. Diskin, Z, & Cadish, B, Algebraic Graph-Based Approach to Management of Multidatabase Systems, *NGITS'95* 69-79 (1995).
17. Dubois, D M, Introduction to Computing Anticipatory Systems, *Int J Comp Anticipatory Sys* **2** 3-14 (1998).
18. Dubois, D M, Review of Incursive, Hyperincursive and Anticipatory Systems - Foundation of Anticipation in Electromagnetism. Computing Anticipatory Systems: CASYS'99 - Third International Conference, Dubois, D M, (ed), American Institute of Physics, *AIP Conference Proceedings* **517** 3-30 (2000).
19. Ehresmann, Andrée, On differentiable categories, *Archive*, *Categories List* ftp://tac.mta.ca/pub/categories/, 5/9/02 (2002).

20. Ehresmann, Andrée C, & Vanbremeersch, Jean-Paul, Emergence Processes up to Consciousness Using the Multiplicity Principle and Quantum Physics, CASYS'03, *AIP Conference Proceedings* **627** 221-233 (2002).
21. Ehresmann, Charles, *Introduction to the theory of structured categories* Technical Report 10, Kansas University (1966).
22. Ehresmann, Charles, *Oeuvres complètes et commentées* Parts I to IV, Amiens. (1980-83).
23. Eiben, Agoston E, *Introduction to Evolutionary Computing* Springer, (2003).
24. Feferman, Solomon, *In the Light of Logic* Oxford (1998).
25. Freyd, P, *Abelian Categories, An Introduction to the Theory of Functors* Harper (1964).
26. Freyd, P, & Scedrov, A, *Categories, Allegories*, North- Holland (1990).
27. Godement, R, *Théorie des faisceaux*, Hermann, Appendix I (1958).
28. Grover L K, A Fast Quantum Mechanical Algorithm for Database Search, *Proc 28th Annual ACM Symposium Theory of Computing* 212-219 (1996).
29. Grover, L K, Rapid sampling through quantum computing, *Proc 32nd Ann ACM Symp Theory of Computing* 618-626 (2000).
30. Heather, M A, & Rossiter, B N, *Database Techniques for Text Modelling: the Document Architecture of British Statutes*, Technical Report Series, Computing Laboratory, Newcastle University, no. 227 (1987).
31. Heather, M A, & Rossiter, B N, Locality, Weak or Strong Anticipation and Quantum Computing I. Non-locality in Quantum Theory, *Int J Comp Anticipatory Sys* **13** 307-326 (2002).
32. Heather, M A, & Rossiter, B N, Locality, Weak or Strong Anticipation and Quantum Computing. II. Constructivism with Category Theory, *Int J Comp Anticipatory Sys* **13** 327-339 (2002).
33. Heather, M A, & Rossiter, B N, The Anticipatory and Systemic Adjointness of E-Science Computation on the Grid, *Computing Anticipatory Systems, Proceedings CASYS'01* Liège, Dubois, D M, (ed.), AIP Conference Proceedings **627** 565-574 (2002).
34. Heather, M A, & Rossiter, B N, Database Searching in Quantum and Natural Computing, *InterSymp 2002 - 14th International Conference on Systems Research, Informatics and Cybernetics*, Lasker, G E, (ed), (2002).
35. Jacob, Christian, Beginnings of Molecular Computing, University of Calgary Biological Computation - CPSC 601.73, CPSC 601.73 (2003).
36. Johnson, Michael, Rosebrugh, Robert & Wood, R J, Sketch Data Models, Relational Schema and Data Specification, CATS'02, Computing: the Australasian Theory Symposium *Electronic Notes on Theoretical Computer Science* **61**.
37. Johnson, M, Rosebrugh, R, & Wood, R J, Entity-Relationship-Attribute Designs and Sketches, *TAC* **10** 94-111 (2002).
38. Johnstone, P T, Sketches of an Elephant, A Topos Theory Compendium, *Oxford Logic Guides* 2 vol, **43** Clarendon (2002).
39. Kelly, G M, & Street, R, Review on the Elements of 2-categories, Proceedings Sydney Category Theory Seminar 1972-73, ed. G M Kelly, *Lecture Notes in Mathematics*, Springer-Verlag **420** 75- 103 (1974).
40. Lanzi,P L, Stolzmann, W, & Wilson, S W (edd) Advances in Learning Classifier Systems *LNAI* Springer, Berlin **1996** (2001).
41. Lawvere, F W, An elementary theory of the category of sets. *Proc Nat Acad Sci* **52** 1506-1511 (1964).
42. Lloyd, Seth, Computational Capacity of the Universe, *Phys Rev Lett* **88** no.23 237901-1/4 (2002).
43. Mac Lane, S, Groups, Categories and Duality, *Proc Nat Acad Sci* **34** 263-267 (1948).
44. Mac Lane, S, *Categories for the Working Mathematician* Springer (1972, 2nd ed 1998).
45. Mac Lane, S, & Moerdijk, I, *Sheaves in Geometry and Logic*, Springer-Verlag (1991).
46. Maddy, Penelope, *Realism in Mathematics*, Oxford (1990).
47. Maddy, Penelope, *Naturalism in Mathematics*, Oxford (1997).
48. Moore, Gregory H, *Zermelo's Axiom of Choice: Its Origins, Development and Influence*, Springer-Verlag, Berlin (19820.
49. Makkai, M, & Paré, R, Accessible Categories: the foundations of categorical model theory, *Comtemp Maths* **104** AMS (1989).
50. Pagels, H, *The Cosmic Code: Quantum Physics as the Language of Nature*, Simon & Schuster, New York (1982).
51. Parkinson, Dennis, *Massively Parallel Computing with the DAP* MIT Press (1990).
52. Paun, G, Rozenberg, G, & Salomaa, A, *DNA Computing* Springer (1998).
53. Rosen, Robert, *Anticipatory Systems Philosophical, Mathematical and Methodological Foundations* Pergamon (1985).
54. Rosen, Robert, *Life Itself: A Comprehensive Inquiry into the Nature, Origin, and Fabrication of Life* Columbia (1991).
55. Rossiter, N, From Classical to Quantum Databases with Applied Pullbacks, *78th Meeting Peripatetic Seminar on Sheaves and Logic* Institut de Recherche Mathématique Avancée, Strasbourg University 15-16 February (2003).
56. Rossiter, N, & Heather, M, Four-level Architecture for Closure in Interoperability, *EFIS2003, Fifth International Workshop on Engineering Federated Information Systems*, Coventry, UK, 17-18 July 83-88 (2003).
57. Selinger, P, *Towards a Quantum Programming Language* http://quasar.mathstat.uottawa.ca/ selinger/papers.html#qp1 (2002).
58. Shor, P, Fault-tolerant quantum computation, *Proc 37th Ann Symp Found Com Sci* IEEE (1996).
59. Sipper, Moshe, *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*, Springer (1997).
60. Tsichritzis, D, ANSI/X3/SPARC DBMS Framework 1978, Report of the Study Group on Database Management Systems, *Information Systems* **3**.
61. Wells, Charles, *Sketches: Outline with References* http://www.cwru.edu/artsci/math/wells/pub/pdf/sketch.pdf (1993).
62. Wells, Charles, On sketches, *Archive, Categories List* ftp://tac.mta.ca/pub/categories/, 12/5/01 (2001).
63. Wexelblat, Richard L, *History of Programming Languages* (c1960).
64. Weyl, Hermann, *The Continuum: A Critical Examination of the Foundation of Analysis* (tr.) S Pollard and T Bole, Kirksville, Mo, Thomas Jefferson University Press (1987). (English translation of Das Kontinuum, Leipzig: Veit, 1918.)
65. Zuse, Konrad, The Computing Universe, *International Journal Theoretical Physics* **21**(6/7) 589-600 (1982).