

The Anticipatory and Systemic Adjointness of E-Science Computation on the Grid

M. A. Heather* and B. N. Rossiter†

**Sutherland Building, University of Northumbria at Newcastle NE1 8ST, m.heather@unn.ac.uk*

†*Computing Science, Newcastle University NE1 7RU, UK, b.n.rossiter@ncl.ac.uk*

Abstract. Information systems are anticipatory systems providing knowledge of the real world. If e-science is to operate reactively across the Grid it needs to be integrable with other information systems and e-commerce. Theory suggests that four strong-anticipatory levels of computational types are sufficient to provide ultimate systemic closure with a single strong anticipation. Between the four levels are three layers of adjoint 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. Type anticipation can be provided by left (F) or right (G) adjoint functors ($F \dashv G$). These however are weak anticipation. Strong anticipation needs both left and right adjoints at each level or by composition of adjoints for the system as a whole $\bar{F}\bar{F}F \dashv GG\bar{G}$. The ISO standard for the Information Resource Dictionary System (IRDS) is itself an anticipatory system with this four-level architecture of universal types which can be used for design of interoperability across the Grid. The sufficiency of middleware tools for the Grid can be anticipated by reference to this same architecture. Thus for instance RDF, the Resource Description Framework, for the markup language XML seems to lack the top level abstraction of IRDS and to have only left-adjoint functionality and therefore not to qualify as a strong anticipatory system.

Keywords: computational types, strong anticipation, category theory, Grid, XML.

1 INTRODUCTION

Computers are anticipatory systems [19] predicting reactive processes in the real-world. Now high computing power is required for quite average projects in physics, biotechnology, medicine, the environment, economics modelling and business. The success of the Web (as one gigantic anticipatory system) for information has promoted the idea of the Grid for computation. Distributed servers can satisfy a world thirst for knowledge. Why not then a global hunger for more processing? Parallelism popular in the 1980s never came to fruition because it was made redundant by more powerful supercomputers but it threw up problems which were never resolved and which must re-emerge with the construction of the Grid as an implementation of an anticipatory system. It is essential to get the theory right from the outset. Even the name Grid, taken by analogy with national power networks, is a little misleading. The connection is not an orthogonal co-ordinate frame and not just a syntax connection. It is more than a subset of system processors. For there is processing in the connections as well as in the distributed computing elements themselves. It is a mini-universe operating constructively in the same way as the real universe operates. Like the real universe it needs to be internally connectable. E-science may have its own special requirements but should not be separate from other information systems using the same routers at the physical level on the Internet and the same virtual resources as in the World Wide Web. E-science shares many common characteristics of e-business. It would need therefore to be built with the same tools throughout. Early examples of possible components can be found in XML markup for mathematics MathML [5] or for chemistry [14] where the same language can cope with structures from the molecular structure up to the structure of the published document describing it. What is the formal description of the Grid? Is XML adequate to implement it?

The simple partitioning of computation between algorithms and data as traditionally found in high-level programs

or as in databases no longer holds in the new pervasive non-local computational environment of the Grid. The "how" and "is" merge at a higher level into distributed information systems. As these information systems become more sophisticated, an underlying schema is needed to achieve the necessary integrity and security. A schema is no more than a structure of types and their inter-relationships. There is a parallel strand of machine's computation where the emphasis is on data structures and universal typing. The universal feature of information systems including most computational methods is the data typing. Data typing is nevertheless still within the classic Turing Machine as embedded in the finite string of symbols in the cells on the tape which are presented to the machine. The data types may be algorithmic in that the sequence of symbols represents processing rules or may have special significance as in modern implementations in the form of mark-up languages such as SGML. Such strings however operate essentially at the syntactical level while data-typing is at the semantic. To raise SGML to this semantic level, XML has been developed with schema definitions added along with the facility to type [24]. The significance of these abstract data types is their application to anticipate features of the real world (both physical and metaphysical). These then emerge as the pragmatic structure of a database schema.

In the context of anticipatory systems, standards perform the role of the abstract machine. For example a FORTRAN standard seeks to provide a universal view of particular Turing instructions. For high-level programming languages give the algorithms for Turing machines. The ISO family of OSI standards [16] are widely accepted and successfully used as a convention for cooperative work but their value is limited to the syntactical level. For while there is internal consistency in the standard there is no guarantee that the application of the standard will result in a self-consistent system, that is with strong anticipation. This need not cause too much concern for implementers in a local system where everything is under their own control and weak anticipation is adequate. However, as soon as any kind of openness or independent autonomicity is introduced, another level of types appears requiring closure at an even higher level. In terms of logic, higher-order is needed to develop a reference level in its most abstract form which can give a provable ultimate closure, that is with strong anticipation. Mathematics (as the archetypal anticipatory system) gives us this third-level closure through constructive methods for defining some reference model for systems with heterogeneous information and processing as required in e-science and e-commerce. This need has been recognized to a limited extent by standards bodies who have produced reference models which relate local standards across a number of levels. However, true reference models are still few and far between.

It is important to bear in mind how anticipation is given by a reference model [2, 7]. If we consider some examples, the reference model for design and implementation of databases is not itself the set of schema available. It gives a potential means for providing compatible components for different systems. However, while these may be compatible, there is no guarantee that they are consistent. This is because the collection of designs provides a reference model but not a universal reference model. Using higher-order logic in the context of the ISO standard Information Resource Dictionary System (IRDS) [8] [9] we build on previous work [7][20] in this paper to investigate the proposed Grid for criteria of reliability for applications of interoperability and cross-platform software in the belief that with a constructive formal basis the scientific community will be able to rely on its operation and results as a sound anticipatory system.

The Grid from our perspective is more than just a reactive system. In such a system, feedback would be an important feature in the form of perhaps an error signal recording the difference between actual and expected inputs ([19], at p. 41). Although the system would then attempt to react to unexpected circumstances to satisfy users' requests, it would be acting on events that had already occurred. A more powerful Grid would anticipate events.

2 TYPES AND META DATA

System catalogues play an important role in anticipation by relating data types. Nearly all catalogues today are *active* in the sense that they are a dynamic automatic source of naming and typing information for programs accessing the system, rather than a passive static reference. In relational systems meta-data is the relationship between data in the schema and the constructs used (tables, attributes). Providing interoperability between one relational system and another is relatively straight-forward and there are commercial systems that provide this capability. Wrapper constructions are increasingly used to provide a meta-level by encapsulation of programs with pre-determined interfaces. The freeness of the object-oriented paradigm means that the meta-level needs to be constructed with great care though to control the representation of classes, objects, properties, references, inheritance, composition and methods.

3 THE INFORMATION RESOURCE DICTIONARY SYSTEM IRDS

Before embarking on a full formal description of the IRDS as an anticipatory system, some understanding and informal insight into its interpretation might be useful. The IRDS is constructed on four levels [3] [4]. Each type level taken with its adjacent type level acts as a *level pair* so that there are three level pairs across the four levels. This means that each point at each level is directly related to a point at the other level in the *level pair*.

The top level is the Information Resource Dictionary Definition Schema (IRDDS), in which concepts relating to policy and philosophy are defined. For example, object-oriented abstractions are to be declared at this level. In principle, only one instance of an IRDDS need be defined for a platform. In a coherent system there can be only one collection of such types. With the open-ended nature of object-oriented structures, however some extensibility may be required.

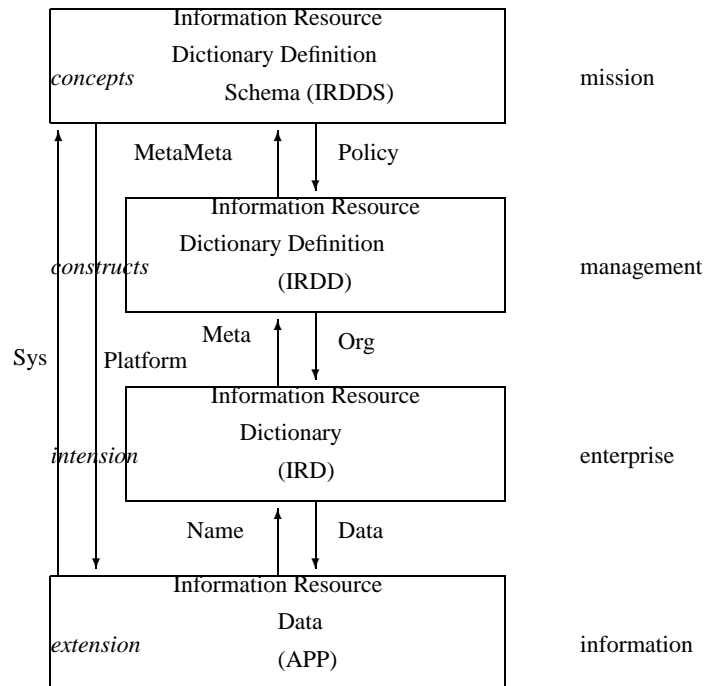


FIGURE 1. Interpretation of IRDS in Schematic Form

The second level is the Information Resource Dictionary Definition (IRDD) in which schema facilities are defined. Each system will have its own IRDD type definition. For example a COBOL IRDD would declare that record-types were an aggregation of single- or multi-valued data field-types while one for SQL would declare that table-types were an aggregation of single-valued data fields.

The third level is the Information Resource Dictionary (IRD) which defines the intension for an application, giving name types and type constraints. There will clearly be many intensions defined in an organization, one for each application. Typing of names and other constraints will be applied to data objects, functions and procedures.

The fourth level is the Information Resource Data (APP) which gives the extension, the data values. There will be one extension for each intension, the values being consistent with the types of names and constraints of the intension. Data values may be simple objects as in SQL or complex objects as in computer-aided design and multimedia systems.

One instance of the Information Resource Dictionary System represents one platform, paradigm or model. Take as an example the relational model. Level 1 would be real-world type abstractions, level 2 the type constructs available, level 3 the data type names and level 4 the data type values.

Between each level the mappings are strictly defined by their starting and terminating points in the respective level types. These may not be immediately obvious in the original standard but they are brought out in the informal diagram of Figure 1 together with more explicit interpretations of the levels. In particular it should be noticed that the

interpretations of the mappings can only be appreciated by considering both directions for each respective mapping. The bottom-up mappings are described in the formal model. The top-down mappings in Figure 1 are as follows:

Between levels 1 and 2 (IRDDS and IRDD), there is the mapping of type *Policy* acting as a level pair. This level pair exists only in IRDS-type systems in which constructive facilities in a system are related to real-world abstractions. For example, *Policy* would indicate the maintenance of constraints such as type inheritance. Between levels 2 and 3 (IRDD and IRD), there is the mapping of type *Org* acting as a level pair. This level pair provides a standard data dictionary function of, for instance, saying which classes are available in an object-based system or which servers are available on a network.

Between levels 3 and 4 (IRD and APP), there is the mapping of type *Data* acting as a level pair. This level pair can be thought of as the state of the art of an information system: to link values to names so that data can be addressed by name rather than by physical location. Between levels 1 and 4 (IRDDS and APP), there is the mapping of type *Platform* acting as a level pair. This level pair short-circuits the navigation through four levels by giving a direct mapping from real-world abstractions to data values. Use of this mapping is described later.

The IRDS standard is the basis for relating heterogeneous types across platform systems, that is systems based on different paradigms. While there is only one instance of the top level type (the IRDDS), this level is extensible and new concepts and abstractions can be added as desired. From the point of view of information systems, the IRDS provides the ability to run an organization with many different paradigms all integrated through the type of structure shown in Figure 1. The critical mapping is type *Platform*, that is the arrow from **IRDDS** to **APP**, relating concepts to values. By determining this mapping for all types of system, the problems arising in re-engineering are avoided to some extent as all types of approach to information systems can be run in an integrated fashion.

The next task is to formalize the diagram in Figure 1 so that a sound scientific basis can be developed for the IRDS model to handle heterogeneous systems.

4 FORMALIZING THE LEVEL TYPES OF THE IRDS

Constructive mathematics attempts to develop logically what can work in practice and can therefore provide the necessary universal typing for interoperability of heterogeneous data systems with consistency and quality assurance in the real-world. Category theory [1] [13] [17] [19] [23] is particularly appropriate for modelling multi-level relationships for it is essentially concerned with links between objects. It has been shown, for instance, to cover adequately dynamic aspects in hypermedia [6]. Rosen ([19] at p.110-124) suggests the use of category theory to represent models. He considers (at p.121) that there is an intimate relation between category theory as a mathematical discipline and the general theory of modelling which includes Model Theory itself. In particular the construction of a functor on a category establishes the sense in which all objects in the category are related to one another so that an implicit model can be produced of each of them. He further notes that a "functorial relationship ... is not in principle a reductionist kind of relation".

Category theory provides a universal construction for formalizing information systems with rigorous typing. It is this uniqueness that provides the universality to form the basis of a general consistent system. An example is now given for a prototype information system focusing on the aspect of a cross-platform system as a heterogeneous distributed database relying on the categorical product construct as a data-type model [15]. In this approach, each class definition can be identified as a collection of arrows (functions) forming a category **IRD** and each family of object values conforming to a particular class definition as a category **APP**. The type mapping from the intension (class definition) to extension (object values) is made by a functor *Data* which enforces the various constraints specified in **IRD**. Category **IRD** is the intension corresponding to the third level in IRDS and **APP** is the extension corresponding to the fourth level type.

The intension category **IRD** is a family of category types, representing definitions of classes, associations (relationships) and coproduct structures indicating inheritance hierarchies. The arrows within it may be methods as in object-based systems, network connections between clients and servers, logical connections as in network databases, or functional dependencies as in relational database schemas. It should be emphasised that categorical approaches naturally include procedures and functions through the underlying arrow concept ensuring that both structure and activity can be modelled in a multi-level manner with rigorous type. The category **APP** is also a family of categories, representing object values and association instances. The functor *Data* mapping from the intension to the extension not only connects a name to its corresponding set of values but also ensures that type constraints specified in the schema, such as functionalities of relationships and functional dependencies, all hold in the extension.

It is relatively straight-forward in category theory to extend the intension and extension two-level structures in a universal manner to handle the four levels of IRDS. In categorial terms each of the four levels of IRDS is defined as a category (i.e. a type). Between each level there is a higher-order function, a functor, which ensures that certain consistency requirements are met in the mapping between the source and target categories. The abstractions level (top) is a category **IRDDS** which defines the various abstractions available for modelling real-world data. The next level is a category **IRDD** defining the various construction facilities available for representing abstractions and data in a particular system. There is therefore, for one instance of **IRDDS**, many instances of **IRDD**, one for each machine type.

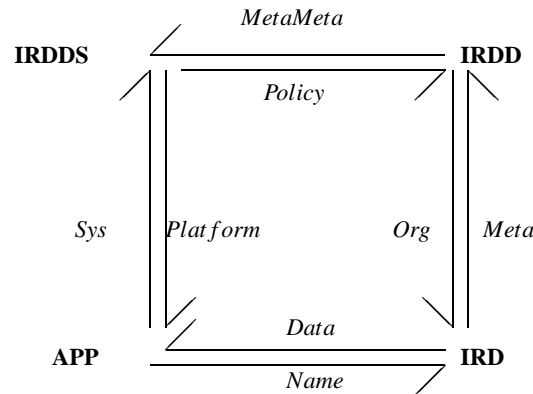


FIGURE 2. IRDS Levels in Functorial Terms

The data functor (level pair) type change *Policy* maps target objects and arrows in the category **IRDDS** to image objects in the category **IRDD** for each type of system. This mapping provides at the meta-meta level the data for each kind of system, that is to say how each abstraction is to be represented. We also label the functor pair *Org* relating for each system the constructions in **IRDD** with the names in a particular application in **IRD**. Combining these new constructions with the product ones above gives the direct and universal representation of IRDS shown in Figure 2.

The remaining functors *MetaMeta*, *Meta* and *Name* are the duals of *Policy*, *Org* and *Data* respectively. *MetaMeta* for a given **IRDD** relates the data modelling facilities provided by a system to the universal collection of abstractions defined in **IRDDS**. *Meta* for a given **IRD** relates the schema definition (intension) to the constructs available in the system defined in **IRDD**. *Meta* therefore relates a name in the intension to a modelling concept in **IRDD** such as a class name to the class construction. *Name* for a given **APP** relates a data value type to its property name as defined in the intension **IRD**.

It will be noted that in Figure 2 all the mappings are two-way and that two compositions emerge. In category theory, Figure 2 is a composition of functors with *Platform* as the overall functor from **IRDDS** \rightarrow **APP**, such that for each type of information system the following compositions hold: $Platform = Data \circ Org \circ Policy$ and $Sys = MetaMeta \circ Meta \circ Name$

An obvious benefit is that we can relate concepts across platforms by comparing the functors $Platform : \mathbf{IRDDS} \rightarrow \mathbf{APP}$ for each of our types of system. However, for full type consistency we should consider the two-way mappings and ensure that composition holds in both directions. Such consistency is achieved in category theory by adjointness. The topic of adjunctions and their composition therefore needs now to be discussed.

5 ADJOINTNESS BETWEEN CATEGORY TYPES

Adjointness characterises the unique relationship between cartesian-closed categories (that is categories of real-world objects). There is a lower-limit functor (F) that preserves co-limits and right-adjoint to F is an upper-limit functor (G) which preserves limits.

The critical comparison is between the arrows (f) in category type **A** and the arrows (g) in type **B**. It is defining the f in terms of the functors F and G and the arrow g . We compare a with the result of $G \circ F(a)$, written simply as

$Gf a$, as assigned to category **A**. In effect an object in **A** is compared with the result obtained by applying functor F and then in turn functor G to the result. This comparison is a natural transformation (η) involving type changing: from $a \longrightarrow Fa \longrightarrow GFa$. This arrow η is the unit of adjunction.

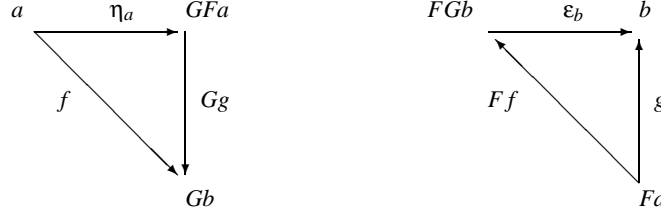


FIGURE 3. Adjoints – unit and counit perspectives

The comparison is made in the context of the corresponding object $G(b)$ which maps b in **B** to **A** so that the left-hand diagram in Figure 3 commutes under the conditions of adjointness, that is $Gg \circ \eta_a = f$. Another view [1], based on equation solving, is that there is a functorial way to relate any arrow $f : a \longrightarrow Gb$ to an arrow $g : Fa \longrightarrow b$ in such a way that g solves the equation $f = G(x) \circ \eta_a$ and that the solution is unique for either some arrow x or object x in category **B**.

An asymmetry, between categories **A** and **B**, apparent between the left-hand and right-hand diagrams of Figure 3, arises in the different viewpoint taken from each side of the adjointness. The perspective of the mapping f can be adjusted to that of the mapping g as in the right-hand diagram of Figure 3. This diagram commutes when $\epsilon_b \circ Ff = g$. The arrow ϵ is the counit of adjunction and a natural transformation comparing $F(G(b))$ to b . The view, based on equation solving, is that there is a functorial way to relate any arrow $g : Fa \longrightarrow b$ to an arrow $f : a \longrightarrow Gb$ in such a way that f solves the equation $g = \epsilon_b \circ F(y)$ and that the solution is unique for either some arrow y or object y in category **A**.

Examples of left adjoints are enrichments such as taking a graph to a category, a set to a group, a set to a preorder and a collection of record keys to hashed addresses. The corresponding right adjoints qualitatively identify the enrichment, ensuring that a number of type restrictions are satisfied.

The notation we use here for an adjunction is as follows. Consider object a in category **A** and object b in category **B** and mappings: $F : \mathbf{A} \longrightarrow \mathbf{B}$, $G : \mathbf{B} \longrightarrow \mathbf{A}$

Then if there is an adjunction between F and G ($F \dashv G$), we write the 4-tuple $\langle F, G, \eta_a, \epsilon_b \rangle : \mathbf{A} \longrightarrow \mathbf{B}$ to indicate the free functor, underlying functor, unit of adjunction and counit of adjunction respectively. From an application viewpoint, a useful view of an adjunction is that of insertion in a constrained environment. The unit η can be thought of as quantitative creation, the counit ϵ as qualitative validation. There is then a relationship between the left and right adjoints such that η represents quantitative identification and ϵ qualitative identification.

An example of adjointness given below to illustrate this property for the pullback category is based on Mac Lane [13] (at p.87 in the second edition). A pullback is shown in Figure 4 with the left-adjoint $\exists : \mathbf{C} \times \mathbf{C} \longrightarrow \mathbf{C}$ taking a pair $\langle a, b \rangle$ to $a + b$ and a right adjoint $\forall : \mathbf{C} \times \mathbf{C} \longrightarrow \mathbf{C}$ taking a pair $\langle a, b \rangle$ to $a \times b$. The pullback shows a relationship between objects a in **A** and b in **B** as ordered pairs $\langle a, b \rangle$ in $\mathbf{C} \times \mathbf{C}$ and as a coproduct in **C**.

Identification is supplied by a pair of insertions $i : a \mapsto a + b$ and $j : b \mapsto a + b$ so that objects in **A** and **B** are inserted into **C** the co-product. The sums are mapped onto the product by the diagonal $\delta : c \mapsto c \times c$. The upper triangle of the diagram in Figure 4 commutes when $\eta_a = \Delta \circ i$, a composition of the insertion and the diagonal. The types are identified as components of the coproduct by the arrows: $c + c \mapsto c$, $i^{-1} : i(c) \mapsto c$, $j^{-1} : j(c) \mapsto c$ and of the components of the product by projections: $a \times b \mapsto a$, $a \times b \mapsto b$. The square of the diagram in Figure 4 commutes when $\epsilon_{c \times c} = j^{-1} \circ i \circ \pi_l$.

The pullback not only applies universally between each level type but also internally at any level. It describes the component subtypes or sorts. Any type can be considered as a limit in $\mathbf{C} \times \mathbf{C}$ of fundamental subtypes of the colimits in **C** (which need not be restricted to sets). The type logic is geometric as between categories. That is the internal language of a topos. The universal pullback, developed through hyperdoctrines, fibre products and sheaf theory, is a general form of an anticipatory system subsuming Russell's type theory, Martin-Löf's theory of types, typed lambda calculus, etc [12] [10]. However, it is the phase space of computation that is to be found in the pullback. Another aspect of the anticipation of the system is computability. That is analagous to the halting problem for the Turing Machine that leads to the confines of NP computability, that is to the limits of predictability in an anticipatory system. We are not

necessarily concerned here with the question of how time-dependent is the computation in the real-world but rather with the matter of consistency and certainty in type computation.

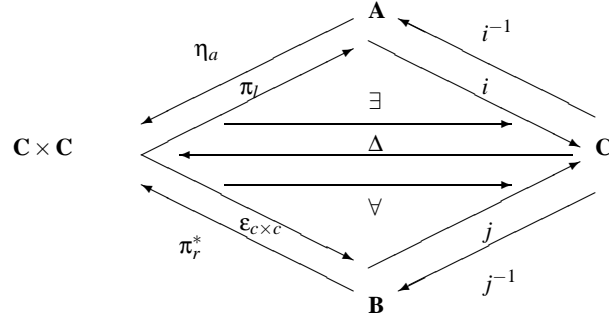


FIGURE 4. Pullback of j along i

5.1 Composition of Adjoints

The IRDS application shown in Figure 2 involves the composition of adjoints, that is an expression is derived in which two or more adjoints are adjacent to each other. It is part of the power of category theory that adjoints can be composed in the same way as other arrows. For example consider the adjoints shown in Figure 5.

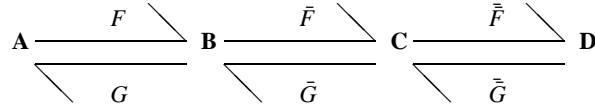


FIGURE 5. Composition of Adjoints

Then we may have six adjoints (if the conditions are satisfied):

$$F \dashv G, \bar{F} \dashv \bar{G}, \bar{\bar{F}} \dashv \bar{\bar{G}}, \bar{F}F \dashv G\bar{G}, \bar{F}\bar{F} \dashv \bar{G}\bar{G}, \bar{\bar{F}}\bar{\bar{F}} \dashv \bar{G}\bar{\bar{G}}$$

These adjunctions give the following isomorphisms:

$$\mathbf{D}(\bar{\bar{F}}\bar{\bar{F}}Fa, d) \cong \mathbf{C}(\bar{F}Fa, \bar{\bar{G}}d) \cong \mathbf{B}(Fa, \bar{G}\bar{\bar{G}}d) \cong \mathbf{A}(a, G\bar{\bar{G}}\bar{\bar{G}}d)$$

where a is an object in \mathbf{A} and d an object in \mathbf{D} . Each equivalent expression represents the collection of arrows from source to target so $\mathbf{D}(\bar{\bar{F}}\bar{\bar{F}}Fa, d)$ represents the collection of arrows from $\bar{\bar{F}}\bar{\bar{F}}Fa$ to d in category \mathbf{D} .

We can define these in more detail with their units and counits of adjunction as follows:

1. $\langle F, G, \eta_a, \varepsilon_b \rangle : \mathbf{A} \longrightarrow \mathbf{B}$
 η_a is the unit of adjunction $1_a \longrightarrow GFa$ and ε_b is the counit of adjunction $FGb \longrightarrow 1_b$
2. $\langle \bar{F}, \bar{G}, \bar{\eta}_b, \bar{\varepsilon}_c \rangle : \mathbf{B} \longrightarrow \mathbf{C}$
 $\bar{\eta}_b$ is the unit of adjunction $1_b \longrightarrow \bar{G}\bar{F}b$ and $\bar{\varepsilon}_c$ is the counit of adjunction $\bar{F}\bar{G}c \longrightarrow 1_c$
3. $\langle \bar{\bar{F}}, \bar{\bar{G}}, \bar{\bar{\eta}}_c, \bar{\bar{\varepsilon}}_d \rangle : \mathbf{C} \longrightarrow \mathbf{D}$
 $\bar{\bar{\eta}}_c$ is the unit of adjunction $1_c \longrightarrow \bar{\bar{G}}\bar{\bar{F}}c$ and $\bar{\bar{\varepsilon}}_d$ is the counit of adjunction $\bar{\bar{F}}\bar{\bar{G}}d \longrightarrow 1_d$
4. $\langle \bar{F}F, G\bar{G}, G\bar{\eta}_aF \bullet \eta_a, \bar{\varepsilon}_c \bullet \bar{F}\varepsilon_c\bar{G} \rangle : \mathbf{A} \longrightarrow \mathbf{C}$
 $G\bar{\eta}_aF \bullet \eta_a$ is the unit of adjunction $1_a \longrightarrow G\bar{G}\bar{F}Fa$ and $\bar{\varepsilon}_c \bullet \bar{F}\varepsilon_c\bar{G}$ is the counit of adjunction $\bar{F}FG\bar{G}c \longrightarrow 1_c$
 The unit of adjunction is a composition of:
 $\eta_a : 1_a \longrightarrow GFa$ with $G\bar{\eta}_aF : GFa \longrightarrow G\bar{G}\bar{F}Fa$

The counit of adjunction is a composition of:

$$\bar{F}\bar{\epsilon}_c\bar{G} : \bar{F}FG\bar{G}c \longrightarrow \bar{F}\bar{G}c \text{ with } \bar{\epsilon}_c : \bar{F}\bar{G}c \longrightarrow 1_c$$

We have retained the symbol \bullet indicating vertical composition [11] as distinct from horizontal composition indicated by the symbol \circ which is normally as here omitted altogether.

$$5. \langle \bar{F}\bar{F}, \bar{G}\bar{G}, \bar{G}\bar{\eta}_b\bar{F} \bullet \bar{\eta}_b, \bar{\epsilon}_d \bullet \bar{F}\bar{\epsilon}_d\bar{G} \rangle : \mathbf{B} \longrightarrow \mathbf{D}$$

$\bar{G}\bar{\eta}_b\bar{F} \bullet \bar{\eta}_b$ is the unit of adjunction $1_b \longrightarrow \bar{G}\bar{G}\bar{F}\bar{F}B$ and $\bar{\epsilon}_d \bullet \bar{F}\bar{\epsilon}_d\bar{G}$ is the counit of adjunction $\bar{F}\bar{F}\bar{G}\bar{G}d \longrightarrow 1_d$

The unit of adjunction is a composition of:

$$\bar{\eta}_b : 1_b \longrightarrow \bar{G}\bar{F}b \text{ with } \bar{G}\bar{\eta}_b\bar{F} : \bar{G}\bar{F}b \longrightarrow \bar{G}\bar{G}\bar{F}\bar{F}b$$

The counit of adjunction is a composition of:

$$\bar{F}\bar{\epsilon}_d\bar{G} : \bar{F}\bar{F}\bar{G}\bar{G}d \longrightarrow \bar{F}\bar{G}d \text{ with } \bar{\epsilon}_d : \bar{F}\bar{G}d \longrightarrow 1_d.$$

$$6. \langle \bar{F}\bar{F}\bar{F}, \bar{G}\bar{G}\bar{G}, \bar{G}\bar{G}\bar{\eta}_a\bar{F}\bar{F} \bullet \bar{G}\bar{\eta}_a\bar{F} \bullet \bar{\eta}_a, \bar{\epsilon}_d \bullet \bar{F}\bar{\epsilon}_d\bar{G} \bullet \bar{F}\bar{F}\bar{\epsilon}_d\bar{G}\bar{G} \rangle : \mathbf{A} \longrightarrow \mathbf{D}$$

The unit of adjunction is a composition of:

$$\bar{\eta}_a : 1_a \longrightarrow \bar{G}\bar{F}a \text{ with } \bar{G}\bar{\eta}_a\bar{F} : \bar{G}\bar{F}a \longrightarrow \bar{G}\bar{G}\bar{F}\bar{F}a \text{ with } \bar{G}\bar{G}\bar{\eta}_a\bar{F}\bar{F} : \bar{G}\bar{G}\bar{F}\bar{F}a \longrightarrow \bar{G}\bar{G}\bar{G}\bar{F}\bar{F}\bar{F}a$$

The counit of adjunction is a composition of:

$$\bar{F}\bar{F}\bar{\epsilon}_d\bar{G}\bar{G} : \bar{F}\bar{F}\bar{F}\bar{G}\bar{G}\bar{G}d \longrightarrow \bar{F}\bar{F}\bar{G}\bar{G}d \text{ with } \bar{F}\bar{\epsilon}_d\bar{G} : \bar{F}\bar{F}\bar{G}\bar{G}d \longrightarrow \bar{F}\bar{G}d \text{ with } \bar{\epsilon}_d : \bar{F}\bar{G}d \longrightarrow 1_d$$

6 RESULTS FROM COMPOSED ADJUNCTIONS

The advantage in deriving compositions is that we have the ability to represent the mappings in either abstract or detailed form [20]. The overall composition gives a simple representation for conceptual purposes; the individual mappings enable the transformations to be followed in detail at each stage and provide a route for implementation. The uniqueness of the components means that a right adjunction can be resolved where there is a component missing. That is the weak anticipation between level pairs provides a determinism but only in one direction.

If a further level \mathbf{E} is added to Figure 5 with the adjoint $\bar{F}\bar{F}\bar{F}\bar{F} \dashv \bar{G}\bar{G}\bar{G}\bar{G}$, categorically the five levels are equivalent to the four levels above because composition is natural. The practical consequence is that a fifth level is equivalent to an alternative fourth level. So there is ultimate closure at a fourth level. As already mentioned, types are in the phase space of the Turing Machine. The halting problem transforms to one of uncertainty and inconsistency. A higher-level is available to resolve inconsistency and uncertainty at any lower level by natural transformation. So while there is weak anticipation across any level-pair with either of the left- or right-adjoint functors, there is locally strong anticipation if both left and right are known. The four-levels consist of a single anticipatory system with the same weak (left or right) or strong (left and right adjointness). This theory suggests that the meta-meta level gives ultimate closure. It is present anywhere; therefore non-local and pervasive computing.

The ability to compose adjoints naturally [13] means that we can combine well together such diverse features as policy, organization and data in a single arrow. Returning to the IRDS representation, we can see the following adjunctions need to be investigated in more detail:

$$Data \dashv Name(\bar{F} \dashv \bar{G})$$

$$Org \dashv Meta(\bar{F} \dashv \bar{G})$$

$$Policy \dashv MetaMeta(F \dashv G)$$

$$Data \circ Org \dashv Meta \circ Name(\bar{F} \circ \bar{F} \dashv \bar{G} \circ \bar{G})$$

$$Org \circ Policy \dashv MetaMeta \circ Meta(\bar{F} \circ F \dashv G \circ \bar{G})$$

$$Data \circ Org \circ Policy \dashv MetaMeta \circ Meta \circ Name(\bar{F} \circ \bar{F} \circ F \dashv G \circ \bar{G} \circ \bar{G})$$

We can construct the 4-tuple to represent the composed adjunctions defined in Figure 2:

$$\langle DOP, AMN, AM\bar{\eta}_{irdds}OP \bullet A\bar{\eta}_{irdds}P \bullet \eta_{irdds}, \bar{\epsilon}_{app} \bullet D\bar{\epsilon}_{app}N \bullet DO\epsilon_{app}MN \rangle$$

where P is the functor *Policy*, O *Org*, D *Data*, A *MetaMeta*, M *Meta* and N *Name*.

If the conditions of this adjunction are met, we can represent the composed adjunction *Platform* \dashv *Sys* by the 4-tuple $\langle Platform, Sys, \eta_{irdds}, \epsilon_{app} \rangle : \mathbf{IRDDS} \longrightarrow \mathbf{APP}$ where *Platform* = *DOP*, *Sys* = *AMN*, η_{irdds} is the unit of adjunction and ϵ_{app} is the counit of adjunction.

This adjunction can be evaluated for each application giving a collection of 4-tuples. Comparison of these 4-tuples then gives the mechanism for computational type closure.

7 THE CONSTRUCTION OF THE GRID

To produce finite results, the Grid must have closure which the universal constructions of category theory show to be natural transformations arising from the adjoint relationship between categories. These categories can be processors or connections on the Grid, that is objects or arrows. The overall adjointness at the level of the system as a whole is constructed as it arises and cannot be determined by any proactive planning at such a high level although the individual components can be efficiently organised from a local perspective. This overall adjointness is a four-level structure that can be resolved into the composition of its constituent lower-level adjoints. Table 1 shows the levels involved in the Grid and their interpretation.

Table 1: Interpretation of Levels in the Grid

Level	Intension	Extension
1	Grid Policy	Program Purpose
2	Grid Organisation	Program Standard
3	Grid Implementation	Program Specification
4	Grid Operation	Program Execution

Like any system the Grid will have contravariant functors between its intension (rather like a global database schema) and the extension of the programs operating across it.

The sufficiency of middleware tools for the Grid can be anticipated by reference to the four-level architecture. Many tools ultimately rely on a markup language such as XML or HTML. In the eXtensible Markup Language (XML), the following basic constructions exist [21] [22]:

1. data enhanced with the XML markup tags.
2. XML document, data marked up with XML tags (a subcategory of HTML) for identifying document semantics.
3. DTD (Document Type Definition) defining the XML document, structures, rules and elements; used to define the tags in an XML document.
4. schema, an alternative to DTD, enabling elements (objects), properties and relationships between elements to be defined.
5. RDF (Resource Description Framework) [18], integrating a variety of web-based metadata activities, providing machine-readable interoperability; resources can be bags, sequences or alternatives.

In terms of our four-level architecture, the data is the category **APP**, the document is the mapping $Data : \mathbf{IRD} \longrightarrow \mathbf{APP}$, the DTD or Schema is the mapping $Org : \mathbf{IRDD} \longrightarrow \mathbf{IRD}$ and RDF is a mapping from $(Data \circ Org : \mathbf{IRDD} \longrightarrow \mathbf{IRD} \longrightarrow \mathbf{APP})$ to another $(Data \circ Org : \mathbf{IRDD} \longrightarrow \mathbf{IRD} \longrightarrow \mathbf{APP})'$. This mapping appears to omit the top level of IRDS, that is **IRDDS**, suggesting that only local interoperability can be achieved.

This assignment is summarised in Table 2.

Table 2: Interpretation of Levels for XML on the Grid

XML feature	function	4-level assignment
Data	data	APP
Document	marked-up data	$Data : \mathbf{IRD} \longrightarrow \mathbf{APP}$
DTD or schema	definition of document (objects, properties, relationships, rules)	$Org : \mathbf{IRDD} \longrightarrow \mathbf{IRD}$
RDF	Resource Description Framework integrating schema	mapping from one $(Data \circ Org)$ to another $(Data \circ Org)'$

8 CONCLUSIONS

Information as a utility available on demand like electricity, gas and water will be a 'pipe-dream' unless it is constructed to account for the complexity shown up by the theory. Those with a vision of the Grid as a world-wide virtual laboratory to process petabyte quantities of heterogeneous data projected to pour from particle-physics experiments like CERN's new large Hadron Collider; genomics, bioinformatic and health applications; vast collaborative engineering projects for next generation aircraft and space stations at orders of magnitude even up to geophysical scales; real-time control of robotic telescopes for real observers distributed across the globe – the structure of the Grid has to be understood for successful interoperability for the processing and management of this knowledge corpus.

The application to the Grid of the categorical representation of the four-level architecture as found in IRDS shows that the top level of the abstraction is apparently missing in the current three-level proposals. Only local interoperability would be achieved with such a limitation. The same is true for the use of middleware tools in XML in its present limited version.

The underlying philosophy of the constructive version of category theory is that the structure of the Grid as outlined here is not just a potential scenario but is the limit of all possible systems up to natural isomorphism realisable in the cartesian closed category of the real-world where we live. Within this limit there is scope for much further development in the form of implementation of the Grid. For instance its internal logic will be intuitionistic with a Heyting lattice-like structure. For as mentioned earlier it is a topos both within the extensional and between the intensional and extensional of Table 1. It is possible to traverse vertical and horizontal potential paths which must be composable in what is sometimes referred to as 2-categories [11] [13]. These are important not only for more technical aspects but the e-business and commercial operation of the Grid and for its management and legal regulation. These are not just add-ons but need to be fully integrated and anticipated from the beginning in letter and in spirit of an immense anticipatory system.

REFERENCES

1. Barr, M, & Wells, C, *Category Theory for Computing Science*, Prentice-Hall (1990).
2. Clements, Alan, *Standardization for Information Technology*, BSI Catalogue no: PP 7315, at page 26 (1987).
3. Gradwell, D, Developments in Data Dictionary Standard, *Computer Bulletin*, September (1987).
4. Gradwell, D J L, The Arrival of IRDS Standards, *8th BNCOD*, York 1990, Pitman 196-209 (1990).
5. Harold, Elliott Rusty, *XML Extensible Markup Language*, DG Books (1998).
6. Heather, M A, & Rossiter, B N, Content Self-awareness in Distributed Multimedia Publishing: the Need for a Unifying Theory, in: Third International Workshop on Principles of Document Processing (PODP'96), edd. C Nicholas & D Wood *Lecture Notes in Computer Science* **1293** Springer-Verlag 59-86 (1997).
7. Heather, M A, & Rossiter, B N, Constructing Standards for Cross-Platform Operation, *Software Qual J*, **7**(2) 131-140 (1998).
8. *Information Resource Dictionary System (IRDS) framework*, Standard ISO/IEC 10027 (1990); 10728 (1993).
9. Information technology - *Reference Model of Data Management*, Standard ISO/IEC 10032 (1993).
10. Jacobs, Bart, *Categorical Logic and Type Theory*, Elsevier, Amsterdam (1999).
11. 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).
12. Mac Lane, S, & Moerdijk, I, *Sheaves in Geometry and Logic*, Springer-Verlag, New York (1991).
13. Mac Lane, S, *Categories for the Working Mathematician*, 2nd ed, Springer-Verlag, New York (1998).
14. Murray-Rust, Peter, *Open Molecule Foundation*, <http://xml-cml.org> (2001).
15. Nelson, D A, & Rossiter, B N, Prototyping a Categorical Database in P/FDM. *Proceedings of the Second International Workshop on Advances in Databases and Information Systems (ADBIS'95)*, Moscow, 27-30 June 1995, Springer-Verlag Workshops in Computing, edd. J Eder and L A Kalinichenko, ISBN 3-540- 76014-8, 432-456 (1996).
16. OSI (Open Systems Interconnection) Standards BS ISO/IEC TR 9571 to 9596, BS ISO/IEC TR 10162 to 10183.
17. Poigné, Axel, Basic Category Theory, in: Abramsky, S, Gabbay, Dov M, & Maibaum, T S E, (edd), *Handbook of Logic and Computer Science*, **I** Background: Mathematical Structures, Clarendon Press, Oxford 416-640 (1992).
18. W3C, Semantic Web Activity: Resource Description Framework (RDF) <http://www.w3.org/RDF/> (2001).
19. Rosen, R, Anticipatory Systems: Philosophical, Mathematical and Methodological Foundations, IFSR International Series on Systems Science & Engineering, Klir, G, (ed.), **1**, Pergamon Press, Oxford (1985).
20. Rossiter, B N, & Heather, M A, Handling Inconsistency with the Universal Reference Model, *MS'2000 International Conference on Modelling and Simulation*, Las Palmas de Gran Canaria, September 2000, 611-618 (2000).
21. Solomon, H, & Simon, Ph D, *XML: eCommerce Solutions for Business and IT Managers*, McGraw-Hill (2001).
22. St Laurent, S, & Cerami, E, *Building XML Applications*, McGraw-Hill (1999).
23. Taylor, P, *Practical Foundations of Mathematics*, Cambridge Studies in Advanced Mathematics, Cambridge (1999).
24. W3C Consortium, *Extensible Markup Language XML*, <http://www.w3.org/XML/> (July 2001).