A Categorical Formalism for Interoperability based on the Information Resource Dictionary Standard (IRDS)

B.N. Rossiter
Computing Science
Newcastle University NE1 7RU, UK
email: B.N.Rossiter@newcastle.ac.uk
Tele: ++44 191 222 7946
Fax: ++44 191 222 8232

M.A. Heather
Sutherland Building
University of Northumbria at Newcastle NE1 8ST

David A.Nelson
School of Computing, Engineering and Technology
University of Sunderland SR6 0DD

December 2000

Abstract

Interoperability is considered in the context of the ISO standards for the Information Resource Dictionary System (IRDS) which provide a complete definition of an information system from real-world abstractions through constructs employed for data and function descriptions to the physical data values held on disk. The IRDS gives a four-level architecture[1] which is considered 1) informally in terms of an interpretation of the levels and the level-pairs between them, 2) in terms of mappings between the levels and 3) formally in terms of a composition of functors and adjoints across the various levels. Two examples are given of the application of IRDS in a categorical context, one comparing the mappings from abstractions to values in relational and object-based systems, the other comparing the mappings from the concept of time to date representations in a number of different approaches. Such comparisons provide a route for interoperability between heterogeneous systems.

[1]A modelling perspective [25] of this work was presented at the MS'2000 International Conference on Modelling and Simulation held at Universidad de Las Palmas de Gran Canaria in September 2000 and organised by AMSE (International Association for Advancement of Modelling and Simulation).

Keywords: interoperability, heterogeneous systems, standards, formalization.

1 Introduction

The ISO family of OSI standards [22] 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. This need not cause too much concern for implementers in a local system where everything is under their own control. A programming language like Java and Visual Basic is a close enough approximation to a closed system where little difficulty may be experienced in practice and an intranet may achieve the same purpose. However, as soon as any kind of openness or independent autonomicity is introduced, another level 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. Mathematics gives us this third-level closure through constructive methods for defining the reference model for applications like system servers. 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 what is meant by a reference model [3, 12]. If we consider the client/server example, the reference model for OSI is not itself the set of protocols for a communications system. It is a framework for the identification and design of protocols for existing or for future communications systems. It gives a potential means for commercial suppliers to provide compatible components for different systems. However, while these may be compatible, there is no guarantee that they are consistent. This is because the OSI set of standards together provide a reference model but not a universal reference model. In this paper we build on previous work [12], using higher-order logic, to seek to show that the IRDS has a constructive formal basis and can be relied upon for this reason.

2 Levels and the Meta Concept

System catalogues play an important role in relating subsystems. 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. The prefix *meta* is often used within the discipline of interoperability to mean 'information about'. 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, for example Platinum Products [4]. In COBOL systems, *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 to control the representation of classes, objects, properties, references, inheritance, composition and methods.

Considerable work in improving interoperability between object-based systems has been done by Crawley [5]. His group developed the Meta-Object Facility (MOF) which introduces an information layer above the layers of schema information and the information itself. In their terminology, meta-information is information about objects in the form of a type schema and meta-meta-information is information about meta-information in the form of the meta-schema containing types of types and types of relationships. MOF provides a framework for managing meta-information and has been accepted by the Object Management Group (OMG) as a way forward in achieving interoperability between object systems [2]. Work at Unisys [17] also highlights progress, within the object context, in achieving interoperability using UML and MOF. In addition IBM have improved interoperability in object systems using XML [14] which is employed in the W3C Resource Dictionary Framework project. Markup languages have the advantage of being robust data models but do not readily provide complex semantic constraints because they are essentially at the syntactic level.

However, neither OSI, MOF or XML provide a universal basis for interoperability. They facilitate the connection of, say, one object system to another but lack a top neutral level relating constructs to universal abstractions, which would enable handling of the full range of heterogeneity, such as objects, relations, trees and data processing record-types. There is also some confusion in terminology with relational systems using the term meta as a relationship between levels and the object systems using it as a level descriptor. Web search engines perhaps typify the present usage of meta-data to achieve interoperability in a local context: IQSeek for example employs meta-data to enable users to search across a number of sites with a single request. Such a technique, applied in effect to a union of document databases, is often termed meta-searching [9]. There is considerable difficulty in standardising output formats and retrieval rankings as each system will have its own specialist methods which may not be fully defined at the user interface. This introduces an uncertainty so that a user cannot be sure what method has led to the results obtained. Critical features may be unused and serious consequences follow if the conditions cannot be relied on with confidence.

Mediation [28] is another technique employed for enabling heterogeneous systems to communicate one with another. Mediation is typically used for linking systems which employ different models and attempts to relate schema constructs in one system to those in another. It suffers from a lack of generality with a considerable amount of manual work required to connect systems which are not closely related.

Another approach has been schema integration often involving the construction of federated database systems in which a number of independently designed databases are permitted to interact with each other. Schema integration in such cases may be an intensive human activity and Duwairi, Fiddian & Gray [6] discuss how the meta-knowledge gener-

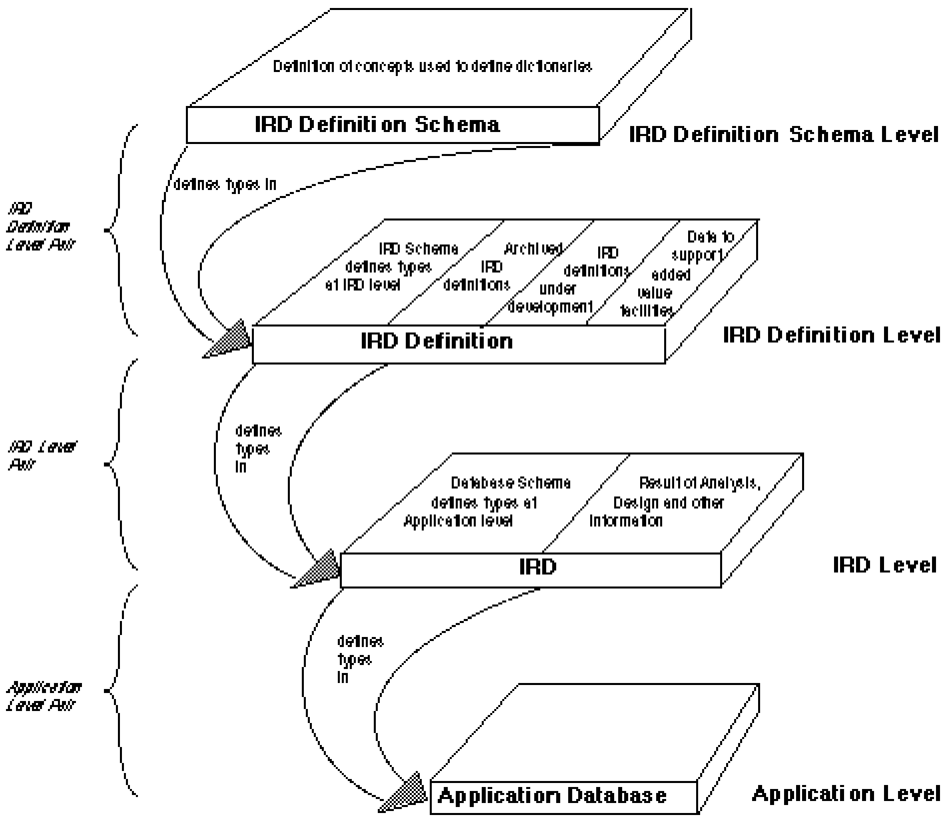

Figure 1: The IRDS in terms of four levels

ated in the activity may be classified and re-used. Higgs & Cottman [13] introduce the idea of deep semantic interoperability, achievable through a Universal Data Access Broker but the scope appears to be towards transaction management rather than complete data integration.

Because of all these anticipated problems, the ANSI Standard Reference Model (X3.138) was developed in the 1980s [7], emerging in 1993 [16] as an associated part of the standard for a framework for Information Resource Dictionary System (IRDS) developed in 1990 and 1993 [15]. Among early language bindings in IRDS were specifications for Pascal, COBOL, Ada, C and SQL. Language bindings can in principle be defined for any language. Use of the IRDS is yet to be exploited in full. Users are still reluctant to extend the levels beyond standard data dictionaries. The tendency is to design flat systems where the need to cross levels is avoided. Further information on the IRDS objectives has been compiled by Gradwell [8].

A basic scheme for the IRDS is shown in Figure 1. The schematic representation of the IRDS Standard [8, 15] shows the level-pairs viewed in the direction of reducing abstraction. In the rest of this paper, we examine more closely the IRDS including these four levels and their inter-relations. We then discuss how the four levels of the IRDS can be shown to capture the universal nature of the reference model because it can be demonstrated that these can be constructed formally. When expressed formally the significance of the adjointness with the corresponding increasing abstraction comes to the fore. This will be shown here in the categorial representation but the main attention will be focussed on the direction of reducing abstraction as in the informal IRDS diagram. Technically this is the lower limit of the adjoint pair [27]. The full significance of the upper limit, that is the increasing abstraction, will be explored in a subsequent technical report.

3 The Information Resource Dictionary System IRDS

Before embarking on a full formal description of the IRDS, some understanding and informal insight into its interpretation might be useful. The IRDS is constructed on four levels. Each level taken with its adjacent 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 concepts. With the open-ended nature of object-oriented structures, however some extensibility may be required.

The second level is the Information Resource Dictionary Definition (IRDD) in which schema facilities are defined. Each system will have its own IRDD 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 names and constraints. There will clearly be many intensions defined in an organization, one for each application. Names, types and other constraints will be given to data objects, network connections, protocol names and signatures and server and client functions.

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 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 abstractions, level 2 the constructs available, level 3 the data names and level 4 the data values. The four levels with their terms, instances, interpretation and the corresponding

level	instances	interpretation	example (relational model)
1. Meta-Meta /policy level	usual abstractions (aggregation, composition, inheritance, association, etc)	mission (natural concepts)	aggregation
2. Meta/ operational level	abstractions at policy level	organizational (high-level operational/analytical tools)	table
3. intension level	network features and functions	formal declaration, schema and labels	Student (name, id, course)
4. extension level	data values satisfying the intension	the information itself	Student ('John Brown', '1275', 'G500')

Figure 2: Interpretation of Levels in the IRDS

components of the relational model can be summarized in the table of Figure 2.

Between each level the mappings are strictly defined by their starting and terminating points in the respective levels. These may not be immediately obvious in the original standard but they are brought out in the informal diagram of Figure 3 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 3 are as follows:

- Between levels 1 and 2 (IRDDS and IRDD), there is the mapping *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 how a network-centric capability is made available in a particular approach.
- Between levels 2 and 3 (IRDD and IRD), there is the mapping *Org* acting as a level

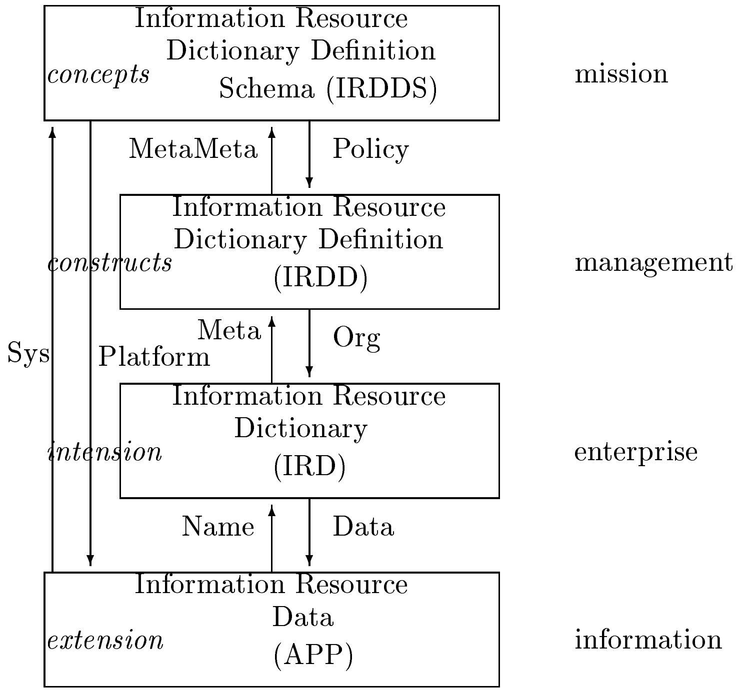

Figure 3: Interpretation of IRDS in Schematic Form

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 *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 *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. The use of this mapping is described later.

The IRDS standard is the basis for relating heterogeneous systems across platforms, that is systems based on different paradigms. While there is only one instance of the top level (the IRDDS), this level is extensible and new concepts and abstractions can be added as desired. From the point of view of client/servers, the IRDS provides the ability to run an organization with many different paradigms all integrated through the type of structure shown in Figure 3. The critical mapping is *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 3 so that a sound scientific basis can be developed for the IRDS model to handle heterogeneous systems.

4 Formalizing the IRDS

Constructive mathematics attempts to develop logically what can work in practice and can therefore provide the necessary universality for interoperability of heterogeneous data systems with consistency and quality assurance in the real-world. Category theory [1, 20, 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 [11].

Category theory provides a universal construction for formalizing information systems. 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 model [21]. 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 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.

The intension category **IRD** is a family of categories, 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. 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 constraints specified in the schema, such as functionalities of relationships, membership classes 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. 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 type of model such as object-oriented, relational, network and simple record-types as in data processing.

$$
\begin{tikzcd}[column sep=huge, row sep=huge]
\mathbf{IRDDS} \arrow[r, shift right, "Policy"'] \arrow[d, shift left, "Platform"] & \mathbf{IRDD} \arrow[l, shift right, "MetaMeta"'] \arrow[d, shift right, "Org"'] \\
\mathbf{APP} \arrow[u, shift left, "Sys"] \arrow[r, shift right, "Name"'] & \mathbf{IRD} \arrow[l, shift right, "Data"'] \arrow[u, shift right, "Meta"']
\end{tikzcd}
$$

Figure 4: IRDS Levels in Functorial Terms

The data functor (level pair) *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 4.

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 to its property name as defined in the intension **IRD**.

It will be noted that in Figure 4 all the mappings are two-way and that two compositions emerge. In category theory, Figure 4 is a composition of functors with $Platform$ as the overall functor from **IRDDS** $\longrightarrow$ **APP**, such that for each type of information system the following compositions hold:
$Platform = Data \circ Org \circ Policy$
$Sys = MetaMeta \circ Meta \circ Name$

An obvious benefit is that we can relate concepts across platforms by comparing the functors $Platform$: **IRDDS** $\longrightarrow$ **APP** for each of our types of system. However, for a full 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 is therefore now discussed.

5 Adjointness

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 **A** and the arrows (g) in **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 GFa, 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 called the unit of adjunction.

$$\begin{tikzcd} a \arrow[r, "\eta_a"] \arrow[dr, "f"'] & GFa \arrow[d, "Gg"] \\ & Gb \end{tikzcd} \qquad \begin{tikzcd} FGb \arrow[r, "\epsilon_b"] & b \\ & Fa \arrow[ul, "Ff"] \arrow[u, "g"'] \end{tikzcd}$$

Figure 5: Adjointness – 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 5 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 5, 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 5. 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 $\mathbf{A}$ and object b in category $\mathbf{B}$ and mappings:

$$F : \mathbf{A} \longrightarrow \mathbf{B}, \quad G : \mathbf{B} \longrightarrow \mathbf{A}$$

Then if there is an adjunction between F and G ($F \dashv G$), we write the 4-tuple: $< F, G, \eta_a, \epsilon_b >: \mathbf{A} \longrightarrow \mathbf{B}$ 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 the development in the second edition (at p.87) of Mac Lane [20]. A pullback is shown in Figure 6 with the left-adjoint $\exists : \mathbf{C} \times \mathbf{C} \longrightarrow \mathbf{C}$ taking a pair $< a, b >$ to $a + b$ and a right adjoint $\forall : \mathbf{C} \times \mathbf{C} \longrightarrow \mathbf{C}$ taking a pair $< a, b >$ to $a \times b$. The pullback shows a relationship between objects a in $\mathbf{A}$ and b in $\mathbf{B}$ as ordered pairs $< a, b >$ in $\mathbf{C} \times \mathbf{C}$ and as a coproduct in $\mathbf{C}$.

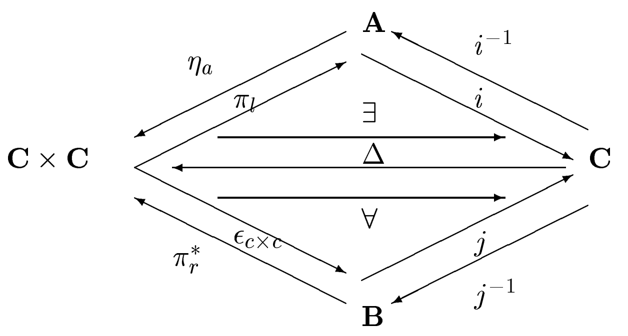

Figure 6: Pullback of j along i

Creativity is supplied by a pair of insertions $i : a \mapsto a + b$ and $j : b \mapsto a + b$ so that objects in $\mathbf{A}$ and $\mathbf{B}$ are inserted into $\mathbf{C}$ 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 6 commutes when $\eta_a = \Delta \circ i$, a composition of the insertion and the diagonal.

Quality validations applied are the identifications of the 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 6 commutes when $\epsilon_{c \times c} = j^{-1} \circ i \circ \pi_l$.

5.1 Composition of Adjoints

The IRDS application shown in Figure 4 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 7.

$$\begin{tikzcd} \mathbf{A} \arrow[r, harpoon, shift left, "F"] & \mathbf{B} \arrow[l, harpoon, shift left, "G"] \arrow[r, harpoon, shift left, "\bar{F}"] & \mathbf{C} \arrow[l, harpoon, shift left, "\bar{G}"] \arrow[r, harpoon, shift left, "\bar{\bar{F}}"] & \mathbf{D} \arrow[l, harpoon, shift left, "\bar{\bar{G}}"] \end{tikzcd}$$

Figure 7: Composition of Adjoints

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

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

These adjunctions give the following isomorphisms:

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

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 $\mathbf{D}(\bar{\bar{F}}\bar{F}Fa, d)$ represents the collection of arrows from $\bar{\bar{F}}\bar{F}Fa$ to d in category $\mathbf{D}$.

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

1. $< F, G, \eta_a, \epsilon_b >: \mathbf{A} \longrightarrow \mathbf{B}$

   η_a is the unit of adjunction $1_a \longrightarrow GFa$ and ϵ_b is the counit of adjunction $FGb \longrightarrow 1_b$

2. $< \bar{F}, \bar{G}, \bar{\eta}_b, \bar{\epsilon}_c >: \mathbf{B} \longrightarrow \mathbf{C}$

   $\bar{\eta}_b$ is the unit of adjunction $1_b \longrightarrow \bar{G}\bar{F}b$ and $\bar{\epsilon}_c$ is the counit of adjunction $\bar{F}\bar{G}c \longrightarrow 1_c$

3. $< \bar{\bar{F}}, \bar{\bar{G}}, \bar{\bar{\eta}}_c, \bar{\bar{\epsilon}}_d >: \mathbf{C} \longrightarrow \mathbf{D}$

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

4. $< \bar{F}F, G\bar{G}, G\bar{\eta}_a F \bullet \eta_a, \bar{\epsilon}_c \bullet \bar{F}\epsilon_c\bar{G} >: \mathbf{A} \longrightarrow \mathbf{C}$

   We have retained the symbol $\bullet$ indicating vertical composition as distinct from normal horizontal composition indicated by the symbol $\circ$. The two different types of composition are to the fore in work on 2-Categories. However, they are normally shown to be equivalent and the distinction may therefore be otiose in applying category theory to real-world applications. Nevertheless there may be an important shift of emphasis because vertical composition draws out relationships between arrows and not just between objects as seen in horizontal composition [18].

$G\bar{\eta}_aF \bullet \eta_a$ is the unit of adjunction $1_a \longrightarrow G\bar{G}\bar{F}Fa$ and $\bar{\epsilon}_c \bullet \bar{F}\epsilon_c\bar{G}$ 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:
$\bar{F}\epsilon_c\bar{G} : \bar{F}FG\bar{G}c \longrightarrow \bar{F}\bar{G}c$ with $\bar{\epsilon}_c : \bar{F}\bar{G}c \longrightarrow 1_c$

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

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

   The unit of adjunction is a composition of:
   $\bar{\eta}_b : 1_b \longrightarrow \bar{G}\bar{F}b$ with $\bar{G}\bar{\bar{\eta}}_b\bar{F} : \bar{G}\bar{F}b \longrightarrow \bar{G}\bar{\bar{G}}\bar{\bar{F}}\bar{F}b$

   The counit of adjunction is a composition of:
   $\bar{\bar{F}}\bar{\epsilon}_d\bar{\bar{G}} : \bar{\bar{F}}\bar{F}\bar{G}\bar{\bar{G}}d \longrightarrow \bar{\bar{F}}\bar{\bar{G}}d$ with $\bar{\bar{\epsilon}}_d : \bar{\bar{F}}\bar{\bar{G}}d \longrightarrow 1_d$.

6. $< \bar{\bar{F}}\bar{F}F, G\bar{G}\bar{\bar{G}}, G\bar{G}\bar{\bar{\eta}}_a\bar{F}F \bullet G\bar{\eta}_aF \bullet \eta_a, \bar{\bar{\epsilon}}_d \bullet \bar{\bar{F}}\bar{\epsilon}_d\bar{\bar{G}} \bullet \bar{\bar{F}}\bar{F}\epsilon_d\bar{G}\bar{\bar{G}} >: \mathbf{A} \longrightarrow \mathbf{D}$

   The unit of adjunction is a composition of:
   $\eta_a : 1_a \longrightarrow GFa$ with $G\bar{\eta}_aF : GFa \longrightarrow G\bar{G}\bar{F}Fa$ with $G\bar{G}\bar{\bar{\eta}}_a\bar{F}F : G\bar{G}\bar{F}Fa \longrightarrow G\bar{G}\bar{\bar{G}}\bar{\bar{F}}\bar{F}Fa$

   The counit of adjunction is a composition of:
   $\bar{\bar{F}}\bar{F}\epsilon_d\bar{G}\bar{\bar{G}} : \bar{\bar{F}}\bar{F}FG\bar{G}\bar{\bar{G}}d \longrightarrow \bar{\bar{F}}\bar{F}\bar{G}\bar{\bar{G}}d$ with $\bar{\bar{F}}\bar{\epsilon}_d\bar{\bar{G}} : \bar{\bar{F}}\bar{F}\bar{G}\bar{\bar{G}}d \longrightarrow \bar{\bar{F}}\bar{\bar{G}}d$ with $\bar{\bar{\epsilon}}_d : \bar{\bar{F}}\bar{\bar{G}}d \longrightarrow 1_d$

The advantage in deriving these compositions is that we have the ability to represent the mappings in either abstract or detailed form. 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 an adjunction can be resolved where there is a component missing.

6 Composed Adjunctions in IRDS

The ability to compose adjoints naturally 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:

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

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

We can construct the 4-tuple to represent the composed adjunctions defined in Figure 4:
$< DOP, AMN, AM\bar{\bar{\eta}}_{irdds}OP \bullet A\bar{\eta}_{irdds}P \bullet \eta_{irdds},$
$\bar{\bar{\epsilon}}_{app} \bullet D\bar{\epsilon}_{app}N \bullet DO\epsilon_{app}MN >$
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:
$< Platform, Sys, \eta_{irdds}, \epsilon_{app} >$: $\mathbf{IRDDS} \longrightarrow \mathbf{APP}$
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 interoperability between applications both heterogeneous and homogeneous.

7 Two Examples of Adjoints in IRDS

7.1 Example 1: Information Systems based on Different Models

A simple example is shown in Figures 8 and 9 of the adjointness found when a comparison is made of the mapping from the top level **IRDDS** to data **APP** for relational and object systems holding similar data definitions for students. The example shows the categories involved (**IRDDS**, **IRDD**, **IRD**, **APP**), the mappings between these categories as the functors *Policy*, *Org* and *Data*, the composition of these functors $Platform$, the natural transformation comparing the composed functor $Platform$ for two different systems and the composed adjunction $Platform \dashv Sys$.

There is one top-level **IRDDS** as there is one collection of universal abstractions; many functors *Policy* each one taking the abstractions to a collection of constructs available in a particular approach; many functors *Org* each one taking the constructs available to the data definitions (schema) in a particular database and many *Data* each one taking the schema to the data values in a particular database. *Org* provides data dictionary facilities and *Data* database facilities. So *OrgUniDDR* and *OrgUniDDO* are data dictionaries for the student database in relational and object systems respectively; *DataUniDBR* and *DataUniDBO* are the student databases (intension-extension mappings) in relational and object systems respectively.

The comparison in the example is between an SQL-92 relational system and an object-based system. Inheritance is not captured naturally by the SQL-92 system but is rep-

Categories: example objects
IRDDS: Aggregation, Inheritance, Supertype, Subtype, <Subtype, Supertype>,Classification, Association,...;
IRDD: Base_table, Tuple, Inher_table, Class, Tree, Struct, Bag, Set, List, Array, Base_Class, Supertype_Class, Subtype_Class, ,...;
IRD: Student <id, name, address>, Module <title,semester>, Postgrad_student<id, thesis_title>, Student <id, name, address, thesis_title>,...;
APP: StudentO {<1234, James, '15 Montfort Ave'>, <1345, Roberts, '27 Park Road'>, ...}, Postgrad {<1234, 'The Language X'>,...}...; Module {<databases,2>, ...}, StudentR {<1234, James, '15 Montfort Ave','The Language X'>, <1345, Roberts, '27 Park Road',''>, ...}

Functors
Policy: **IRDDS** $\longrightarrow$ **IRDD**
PolicyR takes:
Aggregation $\longrightarrow$ Tuple $\longrightarrow$ Base_Table;
spt: Supertype $\longrightarrow$ Tuple; sbt: Subtype $\longrightarrow$ Tuple;
Inheritance $\longrightarrow$ spt(Supertype) $\bigcup$ sbt(subtype) $\longrightarrow$ Inher_Table
PolicyO takes:
Aggregation $\longrightarrow$ Struct $\longrightarrow$ Base_Class;
Inheritance $\longrightarrow$ <Subtype,Supertype>;
π_r <Subtype,Supertype> $\longrightarrow$ Struct $\longrightarrow$ Supertype_Class;
π_l <Subtype,Supertype> $\longrightarrow$ Struct $\longrightarrow$ Subtype_Class

Org: **IRDD** $\longrightarrow$ **IRD**
OrgUniDDR takes:
Base_Table $\longrightarrow$ Module<title, semester>;
st: Tuple $\longrightarrow$ Student<id, name, address>;
po: Tuple $\longrightarrow$ Postgrad<id, thesis_title>;
Inher_Table $\longrightarrow$ st(Tuple) $\bigcup$ po(Tuple) $\longrightarrow$ StudentR<id, name, address, thesis_title>
OrgUniDDO takes:
Base_Class $\longrightarrow$ Module<title,semester>; <subtype,supertype> $\longrightarrow$ <Postgrad, Student>;
st: Struct $\longrightarrow$ Student<id, name, address>; po: Struct $\longrightarrow$ Postgrad<id, thesis_title>;
Supertype_Class $\longrightarrow$ st(Struct); Subtype_Class $\longrightarrow$ po(Struct)

Data: **IRD** $\longrightarrow$ **APP**
DataUniDBR takes:
Module <title,semester> $\longrightarrow$ Module {<databases,2>, ...};
StudentR<id, name, address, thesis_title> $\longrightarrow$ StudentR{<1234, James, '15 Montfort Ave','The Language X'>, <1345, Roberts, '27 Park Road',''>, ...}

Figure 8: Example of Adjoint $Platform \dashv Sys$ for object and relational systems for student data: a) Categories and Functors

Functors (continued)
DataUniDBO takes:
Module<title,semester> $\longrightarrow$ Module{<databases,2>,...};
Student<id, name, address> $\longrightarrow$ StudentO{<1234, James, '15 Montfort Ave'>, <1345, Roberts, '27 Park Road'>, ...};
Postgrad<id, thesis_title> $\longrightarrow$ Postgrad{<1234, 'The Language X'>,...}

Composed functors
Platform: **IRDDS** $\longrightarrow$ **APP**

PlatformUniR = DataUniDBR $\circ$ OrgUniDDR $\circ$ PolicyR

PlatformUniO = DataUniDBO $\circ$ OrgUniDDO $\circ$ PolicyO

PlatformUniR takes:
Aggregation $\longrightarrow$ Module{<databases,2>,...};
Inheritance $\longrightarrow$ Student{<1234, James, '15 Montfort Ave','The Language X'>, <1345, Roberts, '27 Park Road',''>,...}

PlatformUniO takes:
Aggregation $\longrightarrow$ Module{<databases,2>,...};
Inheritance $\longrightarrow$ π_r <Subtype,Supertype> $\longrightarrow$ StudentO{<1234, James, '15 Montfort Ave'>, <1345, Roberts, '27 Park Road'>, ...};
Inheritance $\longrightarrow$ π_l <Subtype,Supertype> $\longrightarrow$ Postgrad{<1234, 'The Language X'>,...}

Natural Transformation
α_{irdds} : PlatformUniR $\longrightarrow$ PlatformUniO
where irdds is an abstraction in IRDDS.

Adjoint
$<$ Platform, Sys, η_{irdds}, ϵ_{app} $>$: **IRDDS** $\longrightarrow$ **APP**
where irdds is an abstraction in **IRDDS** and app is values for a table or class in **APP**.
Platform = DOP, Sys = AMN, η_{irdds} is the unit of adjunction and ϵ_{app} is the counit of adjunction.
P is the functor Policy, O Org, D Data, A MetaMeta, M Meta and N Name

Figure 9: Example of Adjoint $Platform \dashv Sys$ for object and relational systems for student data: b) Functors (continued), Composed Functors, Natural Transformation and Adjoint

resented by the summing of the attributes in the supertype and subtype to give one table-type at the schema and application levels. Inheritance is represented more naturally in the object system with the <subtype,supertype> pair captured in the schema and instances generated in the application for both subtype and supertype when insertion is made at the subtype level.

$PlatformUniR$ shows the mapping from abstraction to data values in a relational system; $PlatformUniO$ shows this mapping in an object system. A comparison of these two mappings is given by the natural transformation α_{irdds}. This enables the relational and object systems shown to be integrated readily as there is a common entry point to the information systems in the **IRDDS** category. The adjoint given by the 4-tuple: $< Platform, Sys, \eta_{irdds}, \epsilon_{app} >$ defines the two-way mapping between $Platform$ and Sys at an abstract level. The detailed form for a composed adjunction, involving three functors in each direction, provides a basis for machine representation. Implementing these adjunctions will give a rigorous method for relating heterogeneous systems.

7.2 Example 2: Open System with Dates

A further example is considered in open systems, involving the problems caused by the inconsistent treatment of times and dates. It is shown that in open systems the existence of local standards may be controlled by defining an appropriate *Policy* mapping indicating how the standard relates to some universal standard. The relationship between one *Policy* mapping and another is captured by a natural transformation relating the two *Model* mappings involved. Our work shows that it is essential to move to this higher level to resolve inconsistencies. The suggestion is that universal representation of all information systems needs only three levels of transformation across level-pairs with a fourth, a natural transformation comparing the overall models, to give ultimate absolute closure. Inconsistency is a failure in composition between the pairs which can be corrected by an appropriate natural transformation interpreted as policy. A simple example of a common occurrence of inconsistency can perhaps give more insight into the salient points in preparation for the theory. There is a well-known inconsistency in an international context of the way that dates are represented. The string *2/3/98* would refer in England to the second day of March but in the United States to the third of February. It is obvious that confusion arising from this example could have various serious consequences in areas such as medicine, law, nuclear safety and stock control. The inconsistency itself arises from the type change between the two formats latent in the different ordering of the numeric fields.

The concept of *date* is as an ordinal applied to the configuration of the solar system and in particular to the motion of the earth around the sun and to its rotation on its axes, as viewed and interpreted from a particular geographical location on earth. Storing dates on a computer illustrates the classic components of any information system. The calendar is a conceptualisation of solar observations which are converted to some numeric format for storage in electronic form. In terms of the ANSI SPARC Standard for Database Architecture, the solar system is the real world phenomenon to be modelled in terms of abstract data types, the calendar is the conceptual schema and the observational procedures provide the external schema. The storage definitions for the fields of day, month and year form part of the internal schema to provide values for disk access and query methods like comparing two dates. From the perspective of the universal formalism of category theory, *date* is a category (or type) consisting of objects. The current object-oriented paradigm has a less developed understanding of objects as objects. The object-oriented term *object* usually refers to a *category* in the sense of category theory, for example the numeric data fields. The order in which the numeric fields occur and the interpretation (convention or policy) determining which is the *month* and which is the *day* are functors. The data (numbers) are in category theory *objects*. Lawvere [19] has introduced the concept of a natural number object where the implicit ordering of the integers are ordinary categorial arrows.

From the IRDS perspective, a date is some measure of time at the conceptual level of **IRDDS**. The constructs available at the **IRDD** level are giga years, years since various events, months, days, carbon dates and any others used for a particular purpose. The **IRD** or schema level specifies the formats avilable such as mm/dd/yyyy, dd/mm/yyyy and g.f (d, m and y indicate digits specified for day, month and year respectively; g.f is a real number). The **APP** level holds the values such as 1950 (for yyyy) or 05/30/1967 (for mm/dd/yyyy).

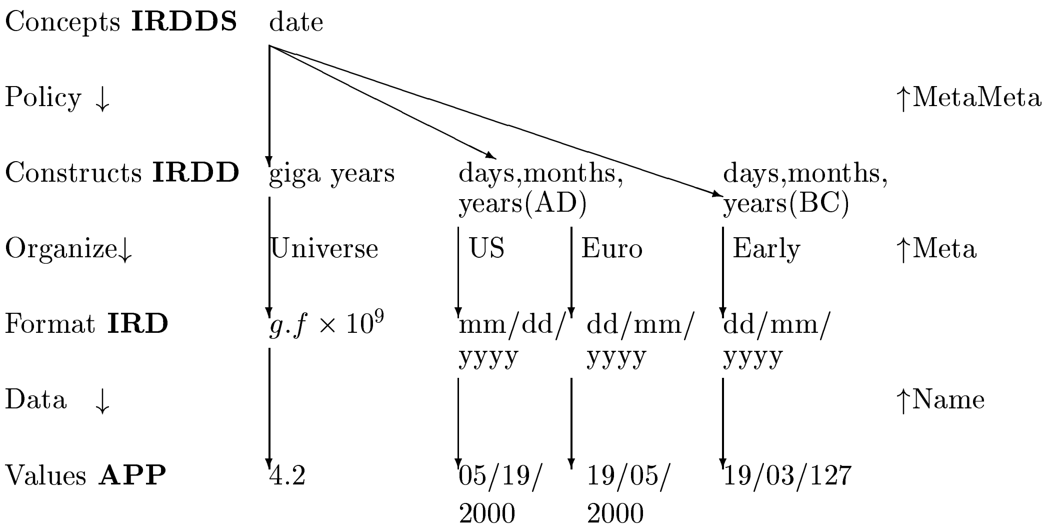

Figure 10: Relationships for Consistent Handling of Dates in Four-level Architecture

Figure 10 shows a four-level representation of dates as outlined above. The mapping *Policy* takes the concept of date into a number of constructions available such as giga years, days, months and years of variable baselines. *Organize* takes the constructs into a number of formats, with the US mapping having a different target to the European. *Data* takes the format to the associated values. Many relationships can be derived from the diagram including that of American and European dates by $\bar{\eta}_{dmyAD}$:

$$\bar{\eta}_{dmyAD} : US \longrightarrow Euro$$

where *dmyAD* is the object in **IRDD** for days. months and years (AD). More generally any date values *date1* and *date2* compared by the natural transformation $\bar{\bar{\epsilon}}_{app}$:

$$\bar{\bar{\epsilon}}_{app} : System(date1) \longrightarrow System(date2)$$

can be related in a consistent manner through the composed adjunctions evaluated earlier and applied to the four-level architecture of Figure 10.

8 Implementation

For experimental testing purposes, an example prototype information system was developed called the Categorical Product Data Model (CPDM) which can formalize, for instance, the object-relational model [24]. The purpose of this prototype was to test the theoretical representation and to examine the ease with which such a representation could be implemented on the computer.

A prototype [21] of CPDM was realized using the platform of P/FDM [10], an implementation of the functional database model developed at Aberdeen University. Categories and functors were coded so that the two categories **IRD** and **APP** could be defined together with the mapping *Data* between them. In the actual work described in [21] the categories **IRD** and **APP** are called **INT** (for intension) and **EXT** (for extension) respectively. The universal nature of category theory means that it is a simple matter to extend the product data model CPDM, defined as a two-level structure, to handle the four levels of IRDS. Further work is planned to explore further the details of the management of the four-level architecture for assisting in the control of cross-platform operations.

9 Discussion

A formal universal description, based on the ISO standards for Information Resource Dictionary System (IRDS), has been developed to provide a complete definition of an open information system from the physical data values held to the concepts employed for data and function representation and real-world abstractions. Such a multi-level formalism can be used to control the evolution of information systems by creating an environment where heterogeneous systems can be compared for cross-platform performance. Current trends towards more structured programming techniques and more disciplined software engineering environments can very usefully exploit the IRDS approach. The prototype system developed CPDM also shows that an implementation based on the categorical model is feasible.

The potential which the IRDS possesses, however, will only be realized for consistent cross-level operation if a formal underpinning of the standard is achieved as described in this paper. A categorial IRDS achieves greater power than that envisaged in the original standards. For instance Spurr [26] comments on the difficulties of using the IRDS standard dictionary with CASE tools because of the lack in IRDS of a natural way of modelling object structures. Our work shows the natural correspondence between categorical databases and object structures making possible a complete and faithful representation of the object-oriented paradigm across the four levels [21]. Our work also shows that in open representation systems such as dates, the four-level approach always gives ultimate closure.

Our conclusion is that the theory shows that for situations where there are open heterogeneous systems, which have to be coherently designed and implemented, the optimal way forward is the four-level closure path described in this paper.

References

[1] Barr, M, & Wells, C, *Category Theory for Computing Science*, Prentice-Hall (1990).

[2] Cattell, R G G, *The Object Data Standard ODMG 3.0*, Morgan Kaufmann (2000).

[3] Clements, Alan, *Standardization for Information Technology,* BSI Catalogue no: PP 7315, at page 26 (1987).

[4] Computer Associates, *Platimum Products and Solutions*, at http://www.cai.com/products/platinum/products.htm (September 2000).

[5] Crawley, S, Davis, S, Indulska, J, McBride, S, Raymond, K, Meta-meta is better-better!, IFIP WG 6.1 International Working Conference on Distributed Applications and Interoperable Systems, *DAIS'97*, Cottbus, Germany 12pp (1997).

[6] Duwairi, R M, Fiddian, N J, & Gray, W A, Schema Integration Meta-knowledge: Classification and Re-use, *14th BNCOD*, Edinburgh 1996, Springer 1-17 (1996).

[7] Gradwell, D, Developments in Data Dictionary Standard, *Computer Bulletin*, September (1987).

[8] Gradwell, D J L, The Arrival of IRDS Standards, *8th BNCOD*, York 1990, Pitman 196-209 (1990).

[9] Gravano, L, & Papakonstantinou, Y, Mediating and Metasearching on the Internet, *Data Engineering* **21**(2) 28-36 (1998).

[10] Gray, P M D, Kulkarni, K G, & Paton, N W, *Object-Oriented Databases: A Semantic Data Model Approach*, Prentice Hall (1992).

[11] 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).

[12] Heather, M A, & Rossiter, B N, Constructing Standards for Cross-Platform Operation, *Software Quality Journal*, **7**(2) 131-140 (1998).

[13] Higgs, M, & Cottman, B, Solving the Data Inter-operability Problem using a Universal Data Access Broker, *Data Engineering* **21**(3) 34-42 (1998).

[14] IBM, *IBM's updated XML Toolkit*, http://www.software.ibm.com/ad/features/xmi. html (November 1999).

[15] Information technology - *Information Resource Dictionary System (IRDS) framework*, Standard ISO/IEC 10027 (1990); 10728 (1993).

[16] Information technology - *Reference Model of Data Management*, Standard ISO/IEC 10032 (1993).

[17] Ivengar, S, A Universal Repository Architecture using UML and MOF spanning Platforms by addressing Semantic and Object Interoperabilities, *Component Strategies*, May 1999, 38-52 (1999).

[18] Kelly, G M, & Street, R, Review on the Elements of 2-categories, Proc. Sydney Category Theory Seminar 1972-73, ed. G M Kelly, *Lecture Notes in Mathematics*, Springer-Verlag **420** 75-103 (1974).

[19] Lawvere, B, Functorial Semantics of Algebraic Theories, *Proc. National Academy of Sciences USA*, 50(1) 869-872 (1963).

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

[21] 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).

[22] OSI (Open Systems Interconnection) Standards, include BS ISO/IEC TR 9571 to 9596 and BS ISO/IEC TR 10162 to 10183.

[23] 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).

[24] Rossiter, B N, Nelson, D A, & Heather, M A, *The Categorical Product Data Model as a Formalism for Object-Relational Databases*, Computing Science Technical Report no.505, University of Newcastle upon Tyne (1994) (42pp).

[25] Rossiter, B N, & Heather, M A, Handling Incosistency with the Universal Reference Model, *MS'2000 International Conference on Modelling and Simulation*, Las Palmas de Gran Canaria, September 2000, edd. R Berriel, V Hernández, R Montenegro, J Rocha, 611-618 (2000).

[26] Spurr, K, CASE Tools, Does the ISO Standard IRDS provide Sufficient Support? in: *Fourth Generation Systems*, ed. S Holloway, Chapman and Hall, Unicom 36-47 (1990).

[27] Taylor, P, *Practical Foundations of Mathematics*, Cambridge Studies in Advanced Mathematics, Cambridge University Press (1999).

[28] Wiederhold, G, Mediators in the Architecture of Future Information Systems, *IEEE Computer* **25** 38-49 (1992).