A Natural Basis for Interoperability

Nick Rossiter[1], Michael Heather[2], and David Nelson[3]

[1] Computing, Engineering and Information Sciences, Northumbria University, NE2 1XE, UK, nick.rossiter@unn.ac.uk
[2] Sutherland Building, Northumbria University, NE1 8ST, UK, michael.heather@btiniternet.com
[3] Computing and Technology, University of Sunderland, SR6 0DD, UK, d.a.nelson@sunderland.ac.uk
WWW home page: http://computing.unn.ac.uk/staff/CGNR1/

Abstract. Successful interoperability of systems requires a sound basis for activity across levels and up to the highest global level. The interoperability is non-local and subject to the conditions of naturality found in reality. The axiomatic models over the last two centuries can guarantee no reliability at higher levels. Category theory is free from these twin problems and can therefore offer a theoretical basis on which to base standards for interoperability. Four levels are used to give closure for policy, organisation, instantiation, naming, classification and meta and metameta relationships. Such constructions provide facilities for relating arrows in general, both descriptive and manipulative, including the specification of constraints and a calculus. The implications for standards are discussed.

1 Introduction

Classical information systems employ some suitable model to mediate between data and hardware. An information system model is a representation of policies in a structured form according to some perceived view of reality. A model is usually regarded as needing to satisfy three requirements: for data structuring, for representing constraints and for data manipulation. Process is perhaps a more profitable way in which to view data manipulation, being more general and a more widely used term in systems theory.

Information system models in current use are almost entirely based on axiomatic set theory. Such models can be employed in a local setting satisfactorily but their use in a global environment, at least without some stronger theoretical underpinning, is much less certain. As an example the ANSI/SPARC architecture was a useful way of capturing abstractions of the relational model in the 1970s and 1980s. It has proved less suitable to facilitate the techniques needed today for interoperability where systems with different underlying models are required to work together. ANSI/SPARC can be viewed as pseudo-natural. It was developed using positivist mathematical techniques and theories like sets. But there is a gap between classical theory and real-world performance and pragmatics.

Triggers are an example of an attempt to patch the weakness of the system by providing some local strong anticipation using Event-Conditions-Actions (ECA) [Date & Darwen 2000].

Interoperability needs natural techniques to deal with levels of types. Current work for dealing with levels of types includes ontologies and, in the context of object models, model driven architectures. Such work will be discussed later as appropriate.

2 Need for Formal Natural Multi-level Type Systems

To handle (non-local) interoperability, formality (for reliability and predictability), naturality (for reality) and multi-level types (for types of types) are required. Categorial methods should replace classical models because models are local and interoperability is non-local. Categorial methods provide formal definitions of levels (as categories), mappings between levels (functors between categories) and comparison of one mapping between levels with another (natural transformation between functors). Categorial techniques are also natural: an arrow within a category is defined as unique up to natural isomorphism.

In areas such as ontologies an informal approach has been taken to naturality which can follow the categorial approach in style, if not in complete formality. Thus in enterprise ontology three levels may be defined: construction model, process/information model and action model [Dietz 1999]. Such ontologies are defining the existence of an object in the context of multiple levels, which is close to the spirit of category theory. In object-based applications MDA (Model-Drive Architecture) has been developed, which separates business and application logic from the underlying platform technology. MDA is based on MOF (Meta-Object Facility) with considerable bias towards UML. An aim of MDA is platform-independence of object-based applications, rather than interoperability between general systems.

Category theory has been used before in information system applications. [Fegaras & Maier 1995] used the monoid calculus in an attempt at standardising the querying of different collection types. [Johnson, Rosebrugh, & Wood 2002] applied sketches to entity-relationship and relational modelling and [Diskin & Cadish 1995] to object databases. None of these approaches though have been multi-level. Sketches are also strictly outside category theory as they permit diagrams that do not commute but they may be mapped onto categories by a model functor.

3 Adjointness

One of the most important features of category theory is adjointness, which gives a degree of measurement of the extent to which the mappings between two categories are equivalent [Barr & Wells 1999].

Figure 1 shows two categories (**L** for left, **R** for right) each containing a canonical triangle to illustrate typical composable arrows. The composition of

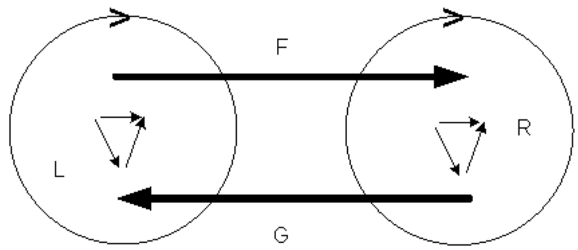

Fig. 1. Adjointness between Two Systems

the arrows (drawn as triangles) represents the natural exactness of real-world interoperability. As we are relying on constructive process not axiomatic sets this interoperability is free from Russell's paradox and free of Gödel's undecidability. The arrows between the categories are functors F, G, the free and underlying functor respectively. Each of the arrows in Figure 1 may be resolved into two component types (technically colimits). These are covariant and contravariant. For a pair of interoperating systems given by these two categories, that is where the triangle in the left-hand category maps into the particular triangle in the right-hand category, then there is a unique contravariant functor G that maps between those triangles in the opposite way. The reverse logic gate $F \dashv G$ is conventionally used to represent adjointness. It is the phenomenon of naturalness. In the vocabulary of axiomatic category theory it is a characteristic of cartesian closed category that applies to all process arrows. It was the publication of [Kan 1958] that led to the recognition that this effect was ubiquitous. F is left adjoint to G and G is right-adjoint to F. The unit of adjunction η and counit of adjunction ϵ measure the extent to which the result from composing the functors differs from the starting point:

η_l is the unit of adjunction $1_l \longrightarrow GF(l)$ and ϵ_r is the counit of adjunction $FG(r) \longrightarrow 1_r$

4 Multi-level Data Structures

The four-level architecture in Figure 2 has orthogonal types with the relationships between the levels expressed as categorical adjunctions as already applied to structures in GRID data processing [Heather & Rossiter 2002]. Categorical adjunctions relate one level to another. The relationship between levels is measurable by the unit of adjunction. For instance the adjunction $Policy \dashv MetaMeta$ indicates that the free functor *Policy* is left adjoint to the underlying functor $MetaMeta$. The unit of adjunction is given by $\eta_{cpt} : 1_{cpt} \longrightarrow MetaMeta \circ Policy(cpt)$.

In Figure 2 the terms used have their normal meaning. Basically in the downward direction, a collection of data structuring concepts (abstractions) are mapped through policies to a collection of constructions (for example classes, ta-

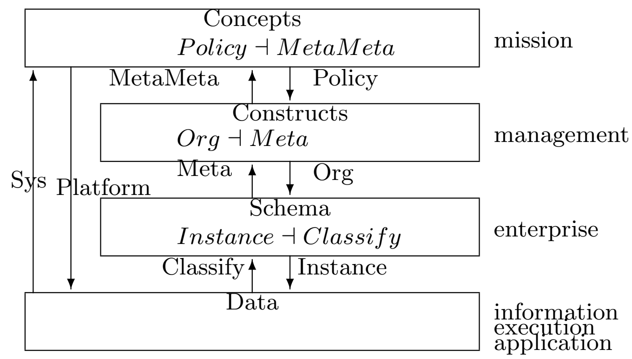

Fig. 2. Interpretation of Levels as Natural Schema in General Terms

bles) which are in turn mapped through organisation to a collection of types (for example, schema definitions) which are finally mapped through instantiation to named data values. In the opposite direction, the named data values are mapped through classification to types, which are in turn mapped through metadata to constructions which are finally mapped to concepts through metameta data.

5 Basis for Interoperability

As mentioned earlier, there are three areas of interoperability that our architecture must satisfy: data structures, constraints and data manipulation. Each is covered in turn.

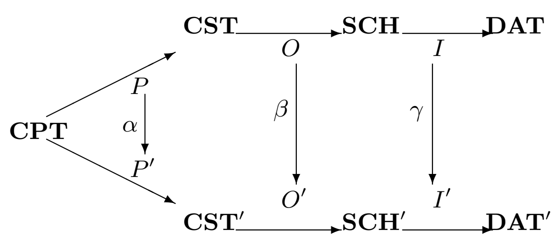

Fig. 3. Comparison of Mappings in two Systems

5.1 Natural Transformation as Data Structures

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

The relationships between one four-level architecture and another can be constructed as in Figure 3, the expanded view of Figure 2. Here for simplicity the mappings are viewed in one direction only. Two systems are compared, one involving categories **CPT**, **CST**, **SCH** and **DAT**, the other **CPT**, **CST′**, **SCH′** and **DAT′**, representing concepts (**CPT**), constructs (**CST**), schema (**SCH**) and data (**DAT**) from Figure 2. **CPT** is the same in both systems as there is one universal type for concepts. As usual the functors relate the categories. We have now though added natural transformations to relate the mapping between one functor and another. It needs to be emphasised that none of these categories are discrete: all have an internal arrow-based structure so the natural transformations are non-trivial [Rossiter 2003]. The functors need to be of the same variance for a meaningful natural transformation to exist between them and this is the case for α, β and γ, all being contravariant as discussed later.

An arrow comparing natural transformations is itself a natural transformation. Some categorists use an older terminology with degrees of 'cell' and describe the identity arrow as 0-cell, an arrow in a category as 1-cell and an arrow between arrows as a 2-cell [Kelly 1972]. An arrow from one natural transformation to another gives a composition of the natural transformations, not a new level ((([Barr & Wells 1999], 1st ed., at p.85); [Rossiter & Heather 2003]). This means that four levels are needed to give the natural closure [Heather & Rossiter 2002].

It may be asked what the levels are going to comprise and what is the nature of the mapping between the levels. Are these constructions essentially arbitrary or do they have definitions, which naturally fall into place? Fortunately the latter seems to apply if we are working in category theory with its property of adjointness. [Lawvere 1969] in his study of adjointness showed that the relationship between intension and extension is contravariant, indicating that the mapping from the category representing the extension, say **DAT**, to the category representing the intension, say **SCH**, is from codomain in **DAT** to domain in **SCH** and domain in **DAT** to codomain in **SCH**.

For matching across the levels in a contravariant manner, the intension **SCH** should be defined with arrows of the form:

$$\text{name} \longrightarrow \text{type}$$

and the extension **DAT** with arrows of the form:

$$\text{value} \longrightarrow \text{name}$$

Both these arrows are functions since associated with each value is one name and associated with each name is one type. Mapping from extension to intension then maps the codomain *name* in **DAT** onto the domain *name* in **SCH** and the domain *value* in **DAT** onto the codomain *type* in **SCH**. This mapping effectively embeds values in types in the context of a name.

The alternative covariant mapping would be from domain to domain and codomain to codomain. The arrow in **SCH** then needs to be reversed to name $\longrightarrow$

value for the two levels to be related. However, name ⟶ value is not a function so a covariant functor from **DAT** to **SCH** lacks naturality.

The four levels of Figure 3 can now be viewed as the two intension-extension pairs in Figure 4. The pairs are for **CPT/CST** (concepts/constructs) and **SCH/DAT** (schema/data). For interoperability purposes, it has been shown by the fundamental nature of category theory that four levels are sufficient for all purposes [Rossiter & Heather 2005]. Further levels are possible but unnecessary. To maintain the coherence of the present approach it would be necessary to go up to six levels as the next step to maintain the intension- extension pairings.

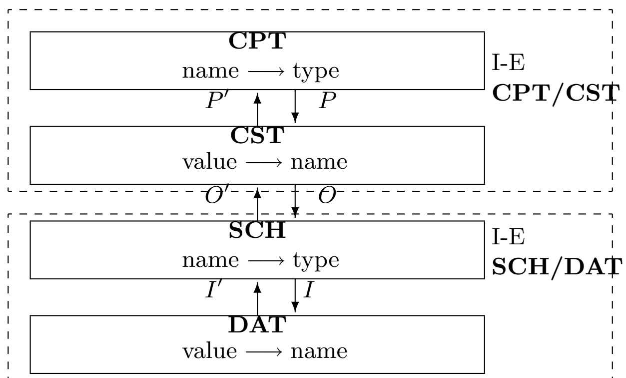

Fig. 4. Defining the Four Levels with Contravariant Functors and Intension-Extension (I-E) Pairs

The table in Figure 5 shows the four levels of concepts, constructs, schema and data with the functors between them of P' (metameta), O' (meta) and I' (classify). The arrows shown for the functors indicate the contravariant nature of the mapping with domain to codomain and codomain to domain. The three examples, from left to right, are for a property, aggregation with relational tables and encapsulation with an abstract data type (ADT). The latter shows the mapping from a binary-tree object named aTree through the class BST and the ADT construction to the encapsulation concept.

5.2 Adjoints for Constraints

In category theory there is a unique solution if adjointness holds between two functors as in Figure 1. This figure can be readily extended to handle four levels, shown in Figure 6, as the composition of adjoints is natural. In Figure 6, with categories and functors as in Figure 4, there are six adjoints

Level	Template	Property	Relational Database (aggregation)	Abstract Data Type (encapsulation)
CPT	name $\longrightarrow$ type	attribute $\longrightarrow$ property	table $\longrightarrow$ aggregation	ADT $\longrightarrow$ encapsulation
P'	$\nearrow$ $\nwarrow$	$\nearrow$ $\nwarrow$	$\nearrow$ $\nwarrow$	$\nearrow$ $\nwarrow$
CST	value $\longrightarrow$ name	registration_no $\longrightarrow$ attribute	birth_type $\longrightarrow$ table	BST $\longrightarrow$ ADT
O'	$\nearrow$ $\nwarrow$	$\nearrow$ $\nwarrow$	$\nearrow$ $\nwarrow$	$\nearrow$ $\nwarrow$
SCH	name $\longrightarrow$ type	car_reg $\longrightarrow$ registration_no	birth_record $\longrightarrow$ birth_type	aTree $\longrightarrow$ BST
I'	$\nearrow$ $\nwarrow$	$\nearrow$ $\nwarrow$	$\nearrow$ $\nwarrow$	$\nearrow$ $\nwarrow$
DAT	value $\longrightarrow$ name	'x123yng' $\longrightarrow$ car_reg	<'Smith', 25 mar 1980, 'Torquay' > $\longrightarrow$ birth_record	instance of tree (nodes/links) $\longrightarrow$ aTree

Fig. 5. Examples of Levels in the Four-Level Architecture

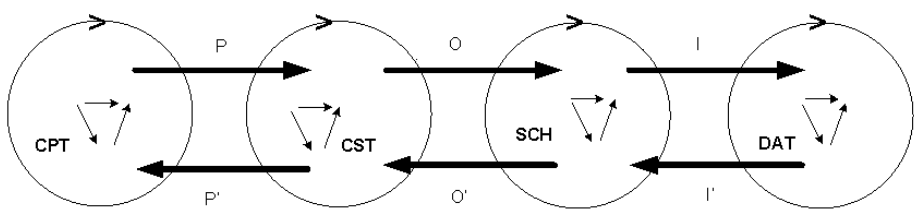

Fig. 6. Composition of Adjoints is Natural

[Heather & Rossiter 2002], one for each functor and its mapping in the opposite direction (first three below, 1-3), one for each pair of adjacent functors and its mapping in the opposite direction(4-5 below) and the one for all three functors composed together and its mapping in the opposite direction (6 below). These adjoints are defined in detail in the following six expressions:

$$< P, P', \eta_{cpt}, \epsilon_{cst} >: \mathbf{CPT} \longrightarrow \mathbf{CST} \quad (1)$$

η_{cpt} is the unit of adjunction $1_{cpt} \longrightarrow P'P(cpt)$ and ϵ_{cst} is the counit of adjunction $PP'(cst) \longrightarrow 1_{cst}$

$$< O, O', \bar{\eta}_{cst}, \bar{\epsilon}_{sch} >: \mathbf{CST} \longrightarrow \mathbf{SCH} \quad (2)$$

$\bar{\eta}_{cst}$ is the unit of adjunction $1_{cst} \longrightarrow O'O(cst)$ and $\bar{\epsilon}_{sch}$ is the counit of adjunction $OO'(sch) \longrightarrow 1_{sch}$

$$< I, I', \bar{\bar{\eta}}_{sch}, \bar{\bar{\epsilon}}_{dat} >: \mathbf{SCH} \longrightarrow \mathbf{DAT} \quad (3)$$

$\bar{\bar{\eta}}_{sch}$ is the unit of adjunction $1_{sch} \longrightarrow I'I(sch)$ and $\bar{\bar{\epsilon}}_{dat}$ is the counit of adjunction $II'(dat) \longrightarrow 1_{dat}$

$$< OP, P'O', P'\bar{\eta}_{cst}P \bullet \eta_{cpt}, \bar{\epsilon}_{sch} \bullet O\epsilon_{cst}O' >: \mathbf{CPT} \longrightarrow \mathbf{SCH} \tag{4}$$

$P'\bar{\eta}_{cst}P \bullet \eta_{cpt}$ is the unit of adjunction $1_{cpt} \longrightarrow P'O'OP(cpt)$ and $\bar{\epsilon}_{sch} \bullet O\epsilon_{cst}O'$ is the counit of adjunction $OPP'O'(sch) \longrightarrow 1_{sch}$

The unit of adjunction is a composition of:
$\eta_{cpt} : 1_{cpt} \longrightarrow P'P(cpt)$ with $P'\bar{\eta}_{cst}P : P'P(cpt) \longrightarrow P'O'OP(cpt)$

The counit of adjunction is a composition of:
$O\epsilon_{cst}O' : OPP'O'(sch) \longrightarrow OO'(sch)$ with $\bar{\epsilon}_{sch} : OO'(sch) \longrightarrow 1_{sch}$

We have retained the symbol $\bullet$ indicating vertical composition [Kelly 1972] as distinct from horizontal composition indicated by the symbol $\circ$ which is normally as here omitted altogether.

$$< IO, O'I', O'\bar{\bar{\eta}}_{sch}O \bullet \bar{\eta}_{cst}, \bar{\bar{\epsilon}}_{dat} \bullet I\bar{\epsilon}_{sch}I' >: \mathbf{CST} \longrightarrow \mathbf{DAT} \tag{5}$$

$O'\bar{\bar{\eta}}_{sch}O \bullet \bar{\eta}_{cst}$ is the unit of adjunction $1_{cst} \longrightarrow O'I'IO(cst)$ and $\bar{\bar{\epsilon}}_{dat} \bullet I\bar{\epsilon}_{sch}I'$ is the counit of adjunction $IOO'I'(dat) \longrightarrow 1_{dat}$

The unit of adjunction is a composition of:
$\bar{\eta}_{cst} : 1_{cst} \longrightarrow O'O(cst)$ with $O'\bar{\bar{\eta}}_{sch}O : O'O(cst) \longrightarrow O'I'IO(cst)$

The counit of adjunction is a composition of:
$I\bar{\epsilon}_{sch}I' : IOO'I'(dat) \longrightarrow II'(dat)$ with $\bar{\bar{\epsilon}}_{dat} : II'(dat) \longrightarrow 1_{dat}$.

$$< IOP, P'O'I', P'O'\bar{\bar{\eta}}_{sch}OP \bullet P'\bar{\eta}_{cst}P \bullet \eta_{cpt},$$

$$\bar{\bar{\epsilon}}_{dat} \bullet I\bar{\epsilon}_{sch}I' \bullet IO\epsilon_{cst}O'I' >: \mathbf{CPT} \longrightarrow \mathbf{DAT} \tag{6}$$

$P'O'\bar{\bar{\eta}}_{sch}OP \bullet P'\bar{\eta}_{cst}P \bullet \eta_{cpt}$ is the unit of adjunction $1_{cpt} \longrightarrow O'I'IO(cpt)$ and $\bar{\bar{\epsilon}}_{dat} \bullet I\bar{\epsilon}_{sch}I' \bullet IO\epsilon_{cst}O'I'$ is the counit of adjunction $IOO'I'(dat) \longrightarrow 1_{dat}$

The unit of adjunction is a composition of:
$\eta_{cpt} : 1_{cpt} \longrightarrow P'P(cpt)$ with $P'\bar{\eta}_{cst}P : P'P(cpt) \longrightarrow P'O'OP(cpt)$ with $P'O'\bar{\bar{\eta}}_{sch}OP : P'O'OP(cpt) \longrightarrow P'O'I'IOP(cpt)$

The counit of adjunction is a composition of:
$IO\epsilon_{cst}O'I' : IOPP'O'I'(dat) \longrightarrow IOO'I'(dat)$ with $I\bar{\epsilon}_{sch}I' : IOO'I'(dat) \longrightarrow II'(dat)$ with $\bar{\bar{\epsilon}}_{dat} : II'(dat) \longrightarrow 1_{dat}$

The expressions above specify the conditions to be satisfied if adjointness occurs in all possible cases in Figure 6. From these constraints we derive values for the various units of adjunction η and counits of adjunction ϵ. If a unit of adjunction is 1, that is for example $1_{cpt} = P'P(cpt)$, then the application of functors P and P' in turn returns the starting point (1_{cpt}). If a counit of adjunction is 0, that is for example $PP'(cst) = 1_{cst}$ then the application of functors P' and P in turn returns the starting point (1_{cst}). These are special cases. In other cases of adjointness η measures the difference between the starting and finishing points after applying in turn the free and underlying functors. ϵ measures the difference between the starting and finishing points after applying in turn the underlying and free functors.

5.3 Natural Calculus for Data Manipulation

Looking at Figure 3, we have three types of mapping to consider: within a category (for instance from a name to a value), from one category to another (for instance the functor P' from **CPT** to **CST**$'$) and from one functor to another (for instance the natural transformation α from P to P').

Following the constructive principles of category theory, the composition of these arrows is natural. This consequently gives rise to a natural calculus first expounded by [Godement 1958] and ([Barr & Wells 1999], 1st ed., pp 94-97) in the form of rules governing composition. The composition of functors and natural transformations is associative.

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

The diagram in Figure 7 shows the application of the Godement calculus to handle semantic interoperability, defined as the interoperation of one system with another at the level of meaning of the data, that is at the metadata level.

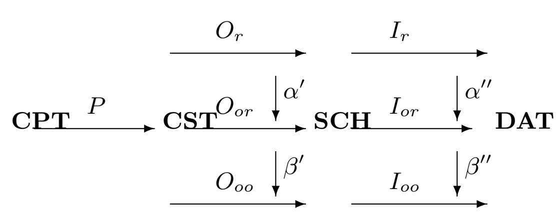

Fig. 7. Semantic Interoperability in terms of Godement

The composition of the top line of functors $I_r \circ O_r \circ P$ gives the mapping from concepts to data for say a relational system r. The composition of the middle line of functors $I_{or} \circ O_{or} \circ P$ gives the mapping from concepts to data for say an object-relational system or. The composition of the bottom line of functors $I_{oo} \circ O_{oo} \circ P$ gives the mapping from concepts to data for say an object-oriented system oo. Comparing these compositions gives a framework for interoperability. For instance the natural transformation α' compares how the mapping is performed from constructions to types in a relational system r with that from constructions to types in an object-relational system *or*. The natural transformation β'' compares how the mapping is performed from types to data in an object-relational system *or* with that from types to data in an object-oriented system *oo*. The advantage of the Godement approach is that arrows of any type can be composed with each other so that any route can be taken.

To extend the categorical framework to handle organisational interoperability, defined as the interoperation of systems at the business process level, we need to vary the functor P for each environment so that the metameta level is variable. The required diagram is shown in Figure 8.

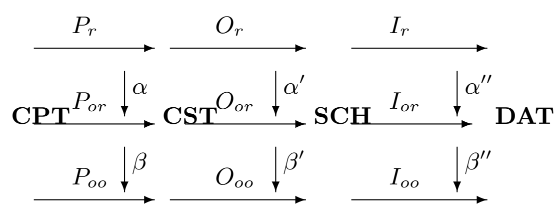

Fig. 8. Organisational Interoperability in terms of Godement

The following canonical rules hold according to the Godement calculus:

$$(\beta' \circ \alpha')(\beta \circ \alpha) = (\beta'\beta) \circ (\alpha'\alpha) \tag{7}$$

$$(I_{or} \circ O_r)\alpha = I_{or}(O_r\alpha) \tag{8}$$

$$\alpha'(O_r \circ P_{or}) = (\alpha' O_r)P_{or} \tag{9}$$

$$I_r(\beta' \circ \alpha')P_{or} = (I_r\beta' P_{or}) \circ (I_r\alpha' P_{or}) \tag{10}$$

$$\alpha''\alpha' = (\alpha'' O_{or}) \circ (I_r\alpha') = (I_{or}\alpha') \circ (\alpha'' O_r) \tag{11}$$

A number of general principles in composition are shown by the equations. Equation 7 indicates that of commutativity (the interchange law); equations 8... 9 indicate that of associativity; equation 10 indicates that of permutation of paths. The last equation, 11, shows the production of simultaneous equations representing different paths through the diagram. This is an important feature as it facilitates the solution for an unknown mapping. For example, in equation 11 above, if the values α', α'' and I_{or} are known, then O_r is the only unknown and a solution can be found for it. That is if it is known how the mapping from constructions to types and from types to data varies between a relational system

r and an object-relational system or and what the mapping is between types and data in an object relational system or, then the mapping between constructions and types in the relational system r can be derived.

6 Discussion

One of the purposes of developing a formalism for a problem area is to provide a rationale in which standards can be planned and discussed. It is perhaps only in the ideal world that standards are based entirely on a theoretical basis. Nevertheless some of the idiosyncrasies and inconsistencies of SQL have been attributed to not rigorously applying axiomatic set theory to the standard [Date & Darwen 2000].

Category theory is a promising candidate as a formalism to assist in the preparation of an interoperability standard because of its pedigree as a workspace for relating different mathematics. The work here has shown that it can indeed perform this role with information systems and cover three critical areas of data structuring, constraints and manipulation (process) in an integrated manner. Recent advances in category theory are likely to improve its match with reality: 2-categories enable some of the strict criteria for composition and associativity to be relaxed to some extent [Baez & Dolan 1998].

The approach developed here is close in a number of respects to the IRDS standard for a reference model [1]. IRDS was based too on a multi-level approach with intension-extension pairs. However, IRDS has had limited success and we would attribute this to its reliance on set theory. This has made it difficult to handle multiple levels and has given an emphasis on data structure over important aspects such as process. Implementation has therefore been difficult.

Another standards approach which appears to have been used more is OSI [2]. OSI has a reference model containing seven layers: Application, Presentation, Session, Transport, Network, Data and Physical. OSI clearly covers more aspects of information systems than IRDS and is expressed at a lower level conceptually so it is easier to implement a complete system. It would be interesting to attempt to represent the OSI standard in terms of our four-level architecture. Seven layers could be equivalent to four categories interconnected by three functors as in our approach. OSI has omissions in security and business processes which are very important in current distributed web-based applications. Of direct interest is a final draft proposal for enterprise modelling [3] which attempts to standardise constructs for enterprise modelling including business process modelling. Without a formal basis, such a standard will be difficult to apply non-locally.

---

[1] ISO/IEC 10027:1990, Information technology – Information Resource Dictionary System (IRDS) framework through to ISO/IEC 13238-3:1998 Information technology – Data Management – Part 3: IRDS export/import facility.

[2] ISO/IEC 7498-1:1994, Information technology - Open Systems Interconnection through to ISO/IEC 10026-2:1998, Information technology - Open Systems Interconnection - Distributed Transaction Processing .

[3] ISO/FDIS 19439. Final Draft International Standard Enterprise integration – Framework for enterprise modelling.

There are two basic tensions that arise with the use of standards: variety and naturality, manageable when local, but irreducible in the non-locality of globalisation. In simple examples uniformity arising from a fixed and narrow standard can result in a loss of variety on account of stringent reductionism. By Ashby's law of requisite variety [Ashby 1947] a system is driven down if it lacks the necessary variety to provide a source for development, originality and creativity. Two or more interoperable systems require a sufficient interacting variety to operate, otherwise they will be driven down, that is seize up. The use of naturality in a formal context, as in the work presented here, is seen as a step forward in raising the quality of interoperability in the real world.

References

[Ashby 1947] William Ross Ashby, Principles of the Self-Organizing Dynamic System, *Journal of General Psychology* **37** 125-128 (1947).

[Baez & Dolan 1998] Baez, J, & Dolan, J, Higher-dimensional algebra III: n-Categories and the algebra of opetopes *Adv Math* **135** 145-206 (1998).

[Barr & Wells 1999] Barr, M, & Wells, C, *Category Theory for Computing Science*, Prentice-Hall (1990, 1995), Centre de Recherches Mathématiques, Montréal (1999).

[Date & Darwen 2000] Date, C J, & Darwen, Hugh, *Foundation for Future Database Systems: The Third Manifesto* 2nd Ed, Addison Wesley (2000).

[Dietz 1999] Dietz, Jan L G, *DEMO Modelling Handbook* version 2.0 (1999).

[Diskin & Cadish 1995] Diskin, Z, & Cadish, B, Algebraic Graph-Based Approach to Management of Multidatabase Systems, *NGITS'95* 69-79 (1995).

[Fegaras & Maier 1995] Fegaras, L, & Maier, D, Towards an Effective Calculus for Object Query Languages *Proc 1995 ACM SIGMOD* 47-58 (1995).

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

[Heather & Rossiter 2002] Heather, M A, & Rossiter, B N, The Anticipatory and Systemic Adjointness of E-Science Computation on the Grid, *Computing Anticipatory Systems* Liège, Dubois, D M, (ed.), AIP Conf Proc **627** 565-574 (2002).

[Johnson, Rosebrugh, & Wood 2002] Johnson, M, Rosebrugh, R, & Wood, R J, Entity-Relationship-Attribute Designs and Sketches, *TAC* **10** 94-111 (2002).

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

[Kelly 1972] 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).

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

[OMG 2005] OMG, *Model Driven Architecture* http://www.omg.org/mda/ (consulted December 2005).

[Rossiter 2003] Rossiter, N, From Classical to Quantum Databases with Applied Pullbacks, *78th Meeting Peripatetic Seminar on Sheaves and Logic* Institut de Recherche Mathématique Avancée, Strasbourg University 15-16 February (2003).

[Rossiter & Heather 2003] Rossiter, N, & Heather, M, Four-level Architecture for Closure in Interoperability, *EFIS2003, Fifth International Workshop on Engineering Federated Information Systems*, Coventry, UK, 17-18 July 83-88 (2003).

[Rossiter & Heather 2005] Rossiter, Nick, & Heather, Michael, Conditions for Interoperability, *7th International Conference on Enterprise Information Systems (ICEIS)* Florida, USA, 25-28 May 2005, 92-99 (2005).