# Handling Inconsistency with the Universal Reference Model

B.N. Rossiter[1] M.A. Heather[2]

**Abstract**

The problems of inconsistent data in information systems are discussed. A four-level architecture, based on the Information Resource Dictionary System (IRDS), is introduced as a potential solution to such problems. A formalization of the IRDS is performed with category theory by composing adjoint functors. An example is given using various date systems to illustrate the need for four-level addressing of data to overcome inconsistency problems.

## Introduction

Inconsistency is not a global property. It arises from particularity where there is only half the story. Real-world integral systems fully operate free from inconsistency for natural processes remove or, like the laws of physics, forbid inconsistencies. Inconsistencies exist between parts of a system. The whole system has to be coherent. A model is normally a partial view of such a system and it is the nature of the type of uncertainties that arise in the modelling process that need to be understood by the professional software engineer for the construction of modern information systems. Inconsistencies come about from implicit type changes, and anomalies when parts only of a system are considered in isolation.

Software does not have the same natural self-organizing properties of hardware. Development of information systems requires a full understanding of the universal underlying concepts if information systems are to be constructed according to high professional engineering standards. Inconsistencies can be understood from the viewpoint of standards in the ISO universal reference model as implemented in the Information Resource Dictionary System. Only recently has fully-formal abstract reasoning been possible with the advent of category theory in mainstream mathematics to understand the implementation of a model (from the real-world structure through the abstract data-type to the data values on disk together with the access and query methods) as a composition of functors. Universal representation of all information systems needs only three levels of transformation across level-pairs with a fourth 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.

## Example of Inconsistency

An 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 medicine, law, nuclear safety, stock control, etc. 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 category theory which are categories in their own right, namely 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*. The implicit ordering of the integers are ordinary categorial arrows.

## Need for Multi-level Structures

We consider that a way to begin to cope with managing inconsistency is to first understand inconsistency in its archetypal form. Here we use the IRDS standard as the basis for relating heterogeneous systems across platforms, that is systems based on different paradigms. 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.

## Information Resource Dictionary System

The IRDS [3, 4] 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. In principle, only one instance of an IRDDS need be defined for a problem area. In a coherent system there can be only one collection of such concepts. The second level is the Information Resource Dictionary Definition (IRDD) in which schema construction facilities are defined. The third level is the Information Resource Dictionary (IRD) which defines the intension for an application, giving names and constraints. The fourth level is the Information Resource Data (APP) which gives the extension, the data values.

There are mappings in each direction between the levels, termed level-pairs. We interpret these mappings in the top-down direction as follows. There is a composite mapping *Model* broken down into constituent functors *Policy*, *Organize* and *Data* representing respectively 1) the policy by which abstractions are represented as constructions in a model, 2) the organization of these constructions as schematic objects and 3) the population of the schema by data values. Each of these functors is termed a level-pair, relating one level in the standard to another. More details on our interpretation of the standard can be found in [2].

## Formalizing the IRDS

$$\begin{tikzcd}
\mathbf{IRDDS} \arrow[r, shift right, "Policy"'] \arrow[d, shift left, "Model"] & \mathbf{IRDD} \arrow[l, shift right, "MetaMeta"'] \arrow[d, shift right, "Organize"'] \\
\mathbf{APP} \arrow[u, shift left, "System"] \arrow[r, shift right, "Name"'] & \mathbf{IRD} \arrow[l, shift right, "Data"'] \arrow[u, shift right, "Meta"']
\end{tikzcd}$$

Figure 1: IRDS Levels in Functorial Terms

The next task is to formalize the ideas advanced for the IRDS so that a sound scientific basis can be developed for the approach. This is first done at a general level as in Figure 1. 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 MetaMeta 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 *Organize* 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 1.

The remaining functors $MetaMeta$, $Meta$ and $Name$ are the duals of $Policy$, $Organize$ 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 1 all the mappings are two-way and that two compositions emerge. In category theory, Figure 1 is a composition of functors with $Model$ as the overall functor from **IRDDS** $\longrightarrow$ **APP**, such that for each type of information system the following compositions hold:
$Model = Data \circ Organize \circ Policy$
$System = MetaMeta \circ Meta \circ Name$

An obvious benefit is that we can relate concepts across models by comparing the functors $Model$ : **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 adjunctions. The topic of adjunctions and their composition is therefore now discussed.

## Adjoints

Adjointness is a development in category theory for expressing the relationship between two categories as a two-way mapping. Adjointness is often expressed in terms of a free functor ($F$) in one direction (from left to right, from source category **A** to target **B**) and an underlying functor ($G$) in the other (target to source, right to left). If certain conditions hold, $F$ is said to be left-adjoint to $G$ and $G$ right-adjoint to $F$.

The critical comparison is between object $a$ in category **A** and the result of $G \circ F(a)$, usually written simply as $GFa$, as assigned to category **A**. In effect an object in **A** is compared with the result obtained by applying $F$ and $G$ to it in turn. This comparison is a natural transformation as it involves a type change: from $A \longrightarrow Fa \longrightarrow GFa$. It is usually written $\eta_a$ and called the unit of adjunction.

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

Figure 2: Adjointness – unit and counit perspectives

The comparison is made in the context of a third object $G(b)$, usually written simply as $Gb$, so that the left-hand diagram in Figure 2 commutes if adjointness exists, that is if $Gg \circ \eta_a = f$. The relationship between categories **A** and **B** is not asymmetric as suggested by the left-hand diagram of Figure 2. The perspective can be adjusted to that of the mapping from $g$ as in the right-hand diagram of Figure 2. This diagram commutes if $\epsilon_b \circ Ff = g$. The arrow $\epsilon_b$ is known as the counit of adjunction and is a natural transformation comparing $F(G(b))$ to $b$. 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 for an adjunction is as follows. Consider object $a$ in category $\mathbf{A}$ and object $b$ in category $\mathbf{B}$ and mappings:
$F : \mathbf{A} \longrightarrow \mathbf{B}, \;\; G : \mathbf{B} \longrightarrow \mathbf{A}$

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 $\eta$ can be thought of as creativity, the counit $\epsilon$ as a quality validation. There is then a relationship between the left and right adjoints such that $\eta$ represents quantitative identification and $\epsilon$ qualitative identification.

## Composition of Adjoints

The IRDS application shown in Figure 1 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 3.

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

Figure 3: Composition of Adjoints

Then we may have six adjoints (if the conditions are satisfied):
$F \dashv G, \bar{F} \dashv \bar{G}, \bar{\bar{F}} \dashv \bar{\bar{G}}, \bar{F}F \dashv G\bar{G}, \bar{\bar{F}}\bar{F} \dashv \bar{G}\bar{\bar{G}}, \bar{\bar{F}}\bar{F}F \dashv G\bar{G}\bar{\bar{G}}$

With hom sets 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 hom set represents the collection of arrows from the first object to the second 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:

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

   $\eta_a$ is the unit of adjunction $1_a \longrightarrow GFa$ and $\epsilon_b$ 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. $< \bar{F}F, G\bar{G}, G\bar{\eta}_a F \bullet \eta_a, \bar{\epsilon}_c \bullet \bar{F}\epsilon_c\bar{G} >: \mathbf{A} \longrightarrow \mathbf{C}$

   The symbol $\bullet$ indicates vertical composition, rather than the normal horizontal composition indicated by $\circ$. Vertical composition is of arrows while horizontal composition is of objects. The two types of composition are equivalent but vertical composition is more in the spirit of category theory, being arrow-based, and is used extensively in structures in categories, particularly 2-Categories, by Mac Lane [5] at p.40-44, 272-275.

   $G\bar{\eta}_a F \bullet \eta_a$ 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 $G\bar{\eta}_a F : GFa \longrightarrow G\bar{G}\bar{F}Fa$

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

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

## 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 compositions need to be investigated

in more detail:
$Data \circ Organize \circ Policy$ (model perspective)
$MetaMeta \circ Meta \circ Name$ (system perspective)

We can construct the 4-tuple to represent the composed adjunctions defined in Figure 1:
$< DOP, AMN, AM\bar{\bar{\eta}}_{irdds}OP \bullet A\bar{\eta}_{irdds}P \bullet \eta_{irdds},$
$\bar{\bar{\epsilon}}_{app} \bullet D\bar{\epsilon}_{app}N \bullet DO\epsilon_{app}MN >$
where P is the functor *Policy*, O *Organize*, D *Data*, A $MetaMeta$, M $Meta$ and N $Name$.

If the conditions of this adjunction are met, we can represent the composed adjunction:
$Model \dashv System$
by the 4-tuple: $< Model, System, \eta_{irdds}, \epsilon_{app} >$: $\mathbf{IRDDS} \longrightarrow \mathbf{APP}$
where $Model = DOP$, $System = AMN$, $\eta_{irdds}$ is the unit of adjunction and $\epsilon_{app}$ 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.

A simple example is shown in Figure 4 of the composed adjoints 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*, *Organize*, *Data*, the composition of these functors $Model$, the natural transformation comparing the composed functor $Model$ for two different systems and the composed adjunction $Model \dashv System$.

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 *Organize* 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. *Organize* provides data dictionary facilities and *Data* database facilities.

The adjoint given by the 4-tuple $< Model, System, \eta_{irdds}, \epsilon_{app} >$ defines the two-way mapping between $Model$ and $System$ at an abstract level. The detailed form, given as a composition of the three functors involved in each direction, provides a basis for machine representation. Implementing these adjunctions will give a rigorous method for relating heterogeneous systems.

## Example of Dates Revisited

Figure 4 shows a four-level representation of dates. 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 $\bar{\eta}_{dmyAD}$:

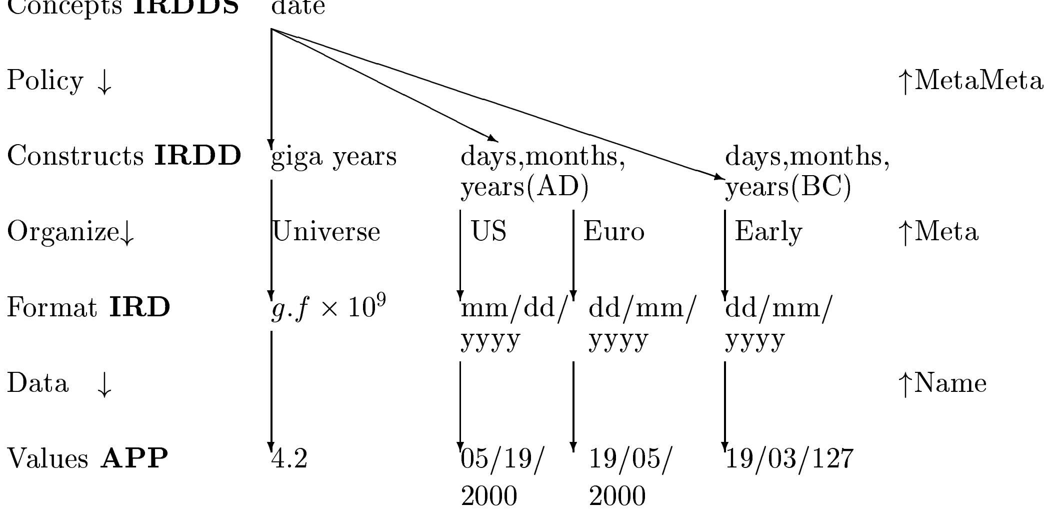

Figure 4: Consistent Handling of Dates in Four-level Architecture

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

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 4.

## References

[1] Barr, M, & Wells, C, *Category Theory for Computing Science*, Prentice-Hall, 2nd ed. (1995).

[2] Heather, M A, & Rossiter, B N, Constructing Standards for Cross-Platform Operation, *Software Quality Journal*, **7**(2) 10pp (1998).

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

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

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

1 Computing Science, Newcastle University NE1 7RU, UK;
email: B.N.Rossiter@newcastle.ac.uk; Tele: ++44 191 222 7946.
2 Sutherland Building, University of Northumbria at Newcastle NE1 8ST.