An Axiomatic Approach to Binary Logical Relations with Applications to Data Refinement

Yoshiki Kinoshita*
Electrotechnical Laboratory, Japan

Peter W. O'Hearn†
Queen Mary and Westfield College, London, England

A. John Power‡
University of Edinburgh, Scotland

Makoto Takeyama
Chalmers University of Technology, Göteburg, Sweden

Robert D. Tennent§
Queen's University, Kingston, Canada

Abstract

We introduce an axiomatic approach to logical relations and data refinement. We consider a programming language and the monad on the category of small categories generated by it. We identify abstract data types for the language with sketches for the associated monad, and define an axiomatic notion of "relation" between models of such a sketch in a semantic category. We then prove three results: (i) such models lift to the whole language together with the sketch; (ii) any such relation satisfies a soundness condition, and (iii) such relations compose. We do this for both equality of data representations and for an ordered version. Finally, we compare our formulation of data refinement with that of Hoare.

*This work has been done with the support of the MITI Cooperative Architecture Project. This author also acknowledges the support of Kaken-hi.

†This author achnowledges the support of the MITI Cooperative Architecture Project.

‡This author acknowledges the support of EPSRC grant GR/J84205, Frameworks for programming languages and logics, and the MITI Cooperative Architecture Project.

§This author acknowledges the support of an operating grant from the Natural Sciences and Engineering Research Council of Canada.

1 Introduction

Logical relations provide an effective means to reason about abstract data types [20, 11]. Consider two set-based interpretations A and C of an ADT X added to a language L. One finds an invariant relation between the underlying sets of the two interpretations that is preserved by all the operations. Logical relations extend this invariant from X to the whole of the language $L(X)$ containing X, and the main lemma of logical relations ensures that the induced relations are themselves invariants, respected by all programs in the language. What one infers from a logical relation depends on the relations assigned to the built-in types of the language or other types lying outside an ADT. For instance, built-in types are typically assigned equality relations, whereas the possible interpretations of an abstract type may vary; then existence of an invariant relation implies that any program of a built-in type has the same meaning under both interpretations. Thus, logical relations provide a means of reasoning about equivalence of data representations.

A mildly different formulation has types interpreted as *ordered* sets, with an order relation $\leq$ assigned to each built-in type; then if P is any program of such a type, the existence of an invariant relation allows one to conclude that $A(P) \leq C(P)$. If we think of $\leq$ as an ordering of "improvement"—where $x \leq y$ means that y is better defined or more determinate than x—then this is reminiscent of Hoare's criterion for the correctness of data refinement (by up-simulations): it says that the "concrete" interpretation C is an improvement on the "abstract" interpretation A.

In this paper we give an axiomatic treatment of these uses of binary logical relations. We prove three main results: (i) such relations automatically lift from an abstract data type to a surrounding language; (ii) any such relation satisfies a soundness condition, and (iii) such relations compose. We do this for both equality of data representations and for an ordered version. Finally, we compare our formulation of data refinement with that of Hoare.

Our development draws on previous work on categorical aspects of logical relations, principally that of Ma and Reynolds [10] and O'Hearn and Tennent [13], and is strongly influenced by the categorical approach to data refinement initiated by Hoare [4] and formulated in terms of categories with algebraic structure by Kinoshita and Power [7, 8]. We give an account of this background in some detail in Section 2.

We then proceed with the paper. In Section 3, we explain the notion of sketch, which gives us a precise category theoretic formulation of the notion of abstract data type. In Section 4, we move on to the central construct of the paper, the notion of L-relation for a monad L on **Cat**, as explained in the background. The notion of L-relation is a mild weakening of an axiomatic notion of logical relation as a structure-preserving functor. We prove a soundness result, which links existence of a logical relation with equality or order. This is a standard result to look for, from the point of view of logical relations. It is often left implicit, or unstated, in work on data refinement. We then consider lifting, where a "relation" between two models of an abstract data type X is lifted to the whole surrounding language $L(X)$. This is analogous to both the main lemma of logical relations and the lifting results in data refinement.

We then study composition in Section 5. This is prompted by Hoare's treatment of data refinement, where composition is regarded as fundamental. Here we face a problem: logical relations do not compose in general, a stumbling block in previous attempts to apply logical relations to refinement. Our solution is to arrange the definition of L-relation so that the resulting notion is slightly weaker than the usual notion of logical relation, while still strong enough for us to prove soundness. The main result is that L-relations compose. Our account has the pleasant consequence of making the "identity" functor, which in the leading example of relations, literally delivers identity relations, acquire a precise status in the axiomatic theory as the identity construction in a category object.

In Section 6, we consider the case of partial order in more detail. For the bulk of the paper, we consider categories, but here we pass to locally ordered categories in order to account for improvement as outlined above. Essentially everything works as for categories except for the account of soundness. That requires a slightly different account of identities.

We end the paper in Section 7 by describing the connection between our work and Hoare's approach to data refinement.

One issue we do not address is polymorphism. In particular, Reynolds has emphasised the importance of the identity extension lemma in his approach to parametricity [20, 10]; one might hope to analyse this axiomatically along the lines of our work here, and perhaps relate it to the logic from [16] (ideas of [22] should be relevant for this). We leave this as a problem for future work.

We are grateful to Edmund Robinson for discussions and feedback throughout this work. In particular, his treatment of identities and soundness in [21], though slightly different, had an influence on us here. It would be interesting to see how far his completeness result, formulated in terms of logical relations for Cartesian closed categories, could be generalised, for other algebraic structures on **Cat**.

2 Background

Our starting point is Ma and Reynolds's categorical treatment of logical relations [10]; see also [12]. Starting with a Cartesian closed category (with finite limits) M, they construct a category E of "logical relations". The paradigmatic example has $M = \mathbf{Set}$, with E the category whose objects are binary relations $R \subseteq S \times T$ and in which a morphism from $R \subseteq S \times T$ to $R' \subseteq S' \times T'$ is a pair of functions $(f: S \to S', g: T \to T')$ such that $fxR'gy$ whenever xRy. The main lemma of logical relations is expressed in terms of a Cartesian closed forgetful functor $U: E \to M \times M$. Further, the "identity extension lemma", which says that the interpretations of types preserve equality relations, is explained in terms of a Cartesian closed functor $J: M \to E$ which, in the paradigmatic example, takes each set to the equality relation on it.

If we consider an interpretation of typed λ-calculus as a Cartesian closed functor from a syntactically-defined category L to another category, then in Ma and Reynolds's theory, a logical relation between two interpretations A and C is a Cartesian closed functor q making the following diagram commute

$$\begin{tikzcd} & E \arrow[d, "U"] \\ L \arrow[ur, "q"] \arrow[r, "\langle C, A\rangle"'] & M \times M \end{tikzcd}$$

Thus, we may understand logical relations in terms of structure-preserving functors, given an appropriate category E of relations. This view is prominent in the work of Hermida [3], who develops a theory of logical relations in which the functor U is required to be a fibration. That allows a connection between the categorical formulation of logical relations and predicates in a formal logical language. The fibrational theory offers excellent prospects for incorporating logical relations into a logic of specifications, a potential that has not been properly exploited yet.

Ma and Reynolds emphasized the construction of E and the accompanying functors J and U. Our concern here is not so much with the construction, but with the abstract properties it satisfies. Following O'Hearn and Tennent [13], we consider two categories with algebraic structure, and structure-preserving functors between them as follows:

$$\begin{tikzcd} E \arrow[r, bend left, "\mathsf{dom}"] \arrow[r, bend right, "\mathsf{cod}"'] & M \arrow[l, "\mathsf{id}"'] \end{tikzcd}$$

We demand only that this forms a reflexive graph of categories (i.e., id composed with dom or cod is the usual identity functor on M), subject to an additional condition. The weakness of our axiomatic conditions allows our definitions and results to apply in cases where E is quite unlike classical logical relations. In particular, we give a precise sense in which our definitions strictly generalise those in Hoare's approach to data refinement [4, 5, 7, 8]. This allows us to extend Hoare's approach to higher types in a less restrictive way than the extension using embedding-projection pairs proposed in [4]. Reflexive graphs have also been used by Robinson and Rosolini [22], to give a construction of parametric models of polymorphism, and by Pitts [15] in his study of logical relations and recursive domains.

So Ma and Reynolds used Cartesian closed categories, while O'Hearn and Tennent used reflexive graphs of categories with no additional structure. For other kinds of languages, one would require other structures on categories. So we seek an axiomatic formulation that allows us to consider a range of possible structures. From the perspective of data refinement, this has already been considered [7], using categories with algebraic structure. We can adopt that approach here as well.

The study of categories with algebraic structure is equivalent to the study of finitary monads on **Cat**. The term finitary can be ignored here. A category with algebraic structure is precisely an algebra for some given monad. There is a monad not only for Cartesian closed categories, but more generally for a wide class of structures one might like to consider, such as symmetric monoidal closed categories, categories with finite coproducts, categories with a natural numbers object, and assorted combinations of these structures. Our results will apply in all of these cases.

Given a syntax for a language and some equations, we automatically have a signature and equations on **Cat** and hence a monad L on **Cat**; the algebras for L are small categories with the structure determined by the signature, subject to the equations. Following Hoare, we may identify the language (subject to the equations) with the free algebra on the empty category; as such, types in the language are objects of the free algebra and terms are arrows (or equivalence classes of arrows). Now, we shall identity a data type on the language with what we shall call an L-sketch for the monad L [9]. Then we shall prove that each sketch X has a lifting $L(X)$; that is, a free algebra on X, and so satisfies the property that a model of the sketch (= data type) in a semantic category (= L-algebra) M lifts uniquely to a map of L-algebras from $L(X)$ to M. The L-algebra $L(X)$ may be seen as the language extended by the data type X, and the lifting result shows how a model of the data type lifts uniquely to give a model of the language extended by the data type.

3 Abstract Data Types and Sketches

To help anchor our development, we use the functional language PCF [23, 17] as a running example; but we give our abstract definitions and their analysis more generally.

We begin by describing how PCF gives rise to a monad on **Cat**.

Example 3.1 PCF is a simply-typed λ-calculus, with types

$$t \ ::= \ \mathbf{nat} \mid \mathbf{bool} \mid \mathbf{1} \mid t \to t \mid t \times t \ .$$

It also has several constants, including the arithmetic operations, a conditional, and fixed-point operators $\mathbf{Y}_t\colon (t \to t) \to t$. ∎

With PCF, we associate a signature and some equations. The signature contains all the data for CCC structure, specified objects (types) **nat** and **bool**, and information to the effect that each constant has the indicated type, cf [9]. There is considerable leeway in the choice of equations. In fact, it is often wise to choose fewer rather than more equations, especially in cases where one does not know what the characterising equations should be, as in variable declarations in imperative languages. Here we shall take them to be the equations for Cartesian closedness; although we could have, we do not include equations for, say, the arithmetic constants of PCF. In any case, the signature information in any specific model will be sufficient to determine a unique interpretation for each constant.

This signature and these equations generate a monad L on **Cat**. The functor part of L takes a category, adjoins the **nat** and **bool** types, thus obtaining a graph, and then forms the Cartesian closed category freely generated by this graph together with the operators. An L-algebra (B, b) consists of a small category B and a functor $b\colon L(B) \to B$ that satisfies two equations [1]. For the monad for PCF, an L-algebra is a small Cartesian closed category with additional data to interpret the constants. It follows that b is a Cartesian closed functor, thus providing unique interpretations for all the operations (which appear in $L(B)$) as well as the CCC structure. Following Hoare [4, 7], we identify the language with the initial L-algebra; i.e., with the free L-algebra $L(\emptyset)$ on the empty category.

A leading example of a semantic category M for PCF is the category ω-**Cppo** of ω-complete pointed partial orders and continuous functions. As a category with structure, M comes equipped with specified objects **nat** and **bool**, and specified maps for successor, **Y**, and so on; these are given exactly as in the standard model of PCF [17].

Categories of the form L-**Alg** for a monad L on **Cat** include the category of small Cartesian closed categories, small categories with finite coproducts, and small monoidal categories. All of our development, except for a soundness result whose statement requires additional data, goes through with **Cat** replaced by any locally presentable category. This is important, for instance, to account for call by value, possibly with side effects or continuations, where a natural structure [18, 24] seems to be centrally closed premonoidal categories, which are monadic over a variant of **Cat**. But for now we work simply in terms of **Cat**.

The following is an example of an ADT that one might wish to add to PCF.

Example 3.2 Consider adjoining an ADT for finite sets of natural numbers to PCF. PCF(*FinSet*) is the extension of PCF obtained by adding a new primitive type *FinSet* and operations such as

$$\begin{array}{rcl} \mathit{empty} & : & \mathit{FinSet} \\ \mathit{exists} & : & \mathbf{nat} \times \mathbf{nat} \times \mathit{FinSet} \to \mathbf{bool} \\ \mathit{include} & : & \mathit{FinSet} \times \mathbf{nat} \to \mathit{FinSet} \\ \mathit{exclude} & : & \mathit{FinSet} \times \mathbf{nat} \to \mathit{FinSet} \end{array}$$

Besides these types, other types may be obtained using $\to$ and $\times$, such as

$$(\mathit{FinSet} \to \mathit{FinSet}) \to \mathit{FinSet}$$

A variant with a higher-order operation is obtained by replacing *empty* by

$$\mathit{new} \quad : \quad (\mathit{FinSet} \to \mathbf{nat}) \to \mathbf{nat}$$

The idea is that $new(\lambda x.\, C)$ works by allocating a new finite set and binding it to x for use in C. Note how the scope of the finite set is explicitly limited here. This kind of higher-order operation is typical of variable declarations in imperative languages, and object creation in object-oriented languages.

For ease of exposition, we do not in this example include any equations between derived terms. ■

Note that specifying primitive operations of an ADT may involve constructions that use operations of the L-structure. For example, *include* is a primitive operation, but has the domain type $FinSet \times \mathbf{nat}$ given by the specific PCF type construction $(-) \times \mathbf{nat}$. Similarly, *new* involves the construction $(-) \rightarrow \mathbf{nat}$.

Let L be a monad on **Cat**. We will define L-sketches, identify these with abstract data types, and consider an interpretation as a structure-preserving functor $L(X) \rightarrow M$, where M is any semantic category (= L-algebra) and $L(X)$ is obtained by adjoining an abstract data type (= L-sketch) X to the language. In order to make our constructions and prove our abstract result, we need to put a size condition on the monad L: we must assert that L has what is called a rank. All monads that arise from languages such as PCF have a rank. An analysis may be found in several papers, for instance [6]. For simplicity, we shall assume the rank is ω; i.e., the monad is finitary. Again, this is a size condition that needs no further analysis to understand the ideas we present here. Also, we need to refer to finitely presentable categories. All finite categories are finitely presentable, and that is all one needs to know.

So, suppose L is a finitary monad L on **Cat**. We first define what we call a family of diagram types.

Definition 3.3 A family $\mathcal{D}$ of diagram types is a small family of 4-tuples $(c_i, d_i, j_i{:}\, c_i \rightarrow d_i, k_i{:}\, d_i \rightarrow Lc_i)$, where c_i and d_i are finitely presentable categories, and j_i and k_i are functors, subject to the condition that the following diagram, dropping the subscripts, commutes (where η is the unit of the monad L):

$$\begin{tikzcd}
d \arrow[rr, "k"] & & Lc \\
& c \arrow[ul, "j"] \arrow[ur, "\eta_c"'] &
\end{tikzcd}$$

■

We generally suppress j and k, leaving them implicit in c and d.

Example 3.4 For our running example, Lc is the category of PCF types and terms freely generated from indeterminate types and terms that correspond to the objects and morphisms of c; η_c is the inclusion. (An indeterminate term $\alpha{:}\, x \rightarrow y$ here differs from a primitive operation in an ADT in that x and y must be indeterminate types and cannot have specified constructions.) In other words, each object and morphism of Lc is a type or term construction of PCF that takes arguments as specified by c.

We formulate the information that the ADT *FinSet* (the variant version) involves the specific constructions $(-)\times\mathbf{nat}$ and $(-)\rightarrow\mathbf{nat}$ using the diagram-type 4-tuple

$$(c_0=\{x\},\ d_0=\{x, x\times\mathbf{nat}, x\rightarrow\mathbf{nat}\},\ j_0\colon c_0\hookrightarrow d_0,\ k_0\colon d_0\hookrightarrow Lc_0)$$

where j_0 and k_0 are the inclusions. The first component c_0 specifies that this diagram-type is concerned with constructions having one type argument. The second component d_0 specifies the two type constructions to be used in the ADT; furthermore, d_0 includes the trivial construction x, as this is required by the commutativity construction.

The ADT also involves the nullary construction **bool**, for which we have another diagram type $(\{\},\ \{\mathbf{bool}\}, (\text{incl.}), (\text{incl.}))$. The family $\mathcal{D}$ of diagram types consists of such 4-tuples. ■

Now assume we are given a finitary monad L together with a family $\mathcal{D}$ of diagram types.

Definition 3.5 An $\langle L,\mathcal{D}\rangle$-*sketch* X consists of a small category X together with a $\mathcal{D}$-indexed family of functors $\varphi_i\colon d_i\rightarrow X$. A model of (X,φ) in an L-algebra (B,b) is a functor $f\colon X\longrightarrow B$ such that the following diagram commutes:

$$\begin{tikzcd} d \arrow[r, "k"] \arrow[d, "\varphi"'] & Lc \arrow[d, "b\cdot L(f\varphi j)"] \\ X \arrow[r, "f"'] & B \end{tikzcd}$$

■

Intuitively, the category X is that of primitive types and operations. Unlike freely adjoined indeterminates, however, there are some conditions on how these can be interpreted in an L-algebra (B,b); the family φ of functors determines these conditions.

Example 3.6 Before we explain the $\langle L,\mathcal{D}\rangle$-sketch for *FinSet*, we consider a simpler case. For the moment, assume that $\mathcal{D}$ has only the one component (c_0,d_0,J_0,k_0) in Example 3.4, and that X is the discrete category $\{u,v,w\}$ of primitive types. Define $\varphi\colon d_0\rightarrow X$ by

$$x\mapsto u,\quad x\times\mathbf{nat}\ \mapsto v,\quad x\rightarrow\mathbf{nat}\ \mapsto w.$$

An L-algebra (B,b) is a CCC $(B,\times_B,\rightarrow_B,\ldots)$ with additional data $\mathbf{nat}_B,\ldots$ to interpret the constants. A model f of (X,φ) in (B,b) gives the interpretations $f(u)$, $f(v)$, and $f(w)$ in B. The above commutativity, together with the definition of φ, implies that these interpretations satisfy the equations

$$f(u)\times_B\mathbf{nat}_B=f(v),\quad f(u)\rightarrow_B\mathbf{nat}_B=f(w).$$

So, φ determines the stipulations that $u \times \mathbf{nat}$ equal v and $u \rightarrow \mathbf{nat}$ equal w that any model f must validate. Note that these conditions cannot be formulated directly because X does not have the relevant operations.

The sketch (X, φ) for *FinSet* is given as follows. First we replace each constructed type used in the ADT (including each PCF type constant such as **bool**) by a new primitive type. The category X has as objects the primitive type *FinSet* together with these new ones; the morphisms of X are the primitive operations (closed under composition and the identities). Finally, we stipulate, by a suitable family φ of functors, that any model should interpret each new primitive type by the corresponding construction on the interpretation of *FinSet* (or by the data for constants, in the case of **bool**, etc.). ■

Note that we can also have equations between two constructed types or terms (under interpretations). For example, in the simpler case above, if we had φ sending both $x \times \mathbf{nat}$ and $x \rightarrow \mathbf{nat}$ to v, then in effect we have stipulated that $u \times \mathbf{nat}$ equals $u \rightarrow \mathbf{nat}$ (equals v). Similarly, an equation between two terms constructed from primitive operations can be expressed by φ sending the two corresponding term constructions to a single morphism of X.

Also note that the family $\mathcal{D}$ may have repeated components; this would allow use of the same construction on different arguments in formulating desired conditions.

To model an ADT X we must specify how to interpret it in a semantic category M; i.e., we must give a model of the sketch X in the L-algebra M. The following universal property shows how such an interpretation lifts uniquely to an interpretation of the language $L(X)$ obtained by adjoining an ADT.

Theorem 3.7 [9] For any $\langle L, \mathcal{D}\rangle$-sketch X, there is an L-algebra $L(X)$ and a model $\beta\colon X \longrightarrow L(X)$ of X in $L(X)$ such that for any L-algebra (B, b), composition with $\beta\colon X \longrightarrow L(X)$ induces a bijection between the set of models of X in (B, b) and the set of L-algebra maps from $L(X)$ to (B, b). ■

Example 3.8 We specify two models of *FinSet*. The abstract interpretation, A, uses the set of finite subsets of N, lifted:

$$A(\mathit{FinSet}) \;=\; (\mathcal{P}_{\mathit{fin}} N)_\perp \;.$$

The concrete interpretation, C, uses lists:

$$C(\mathit{FinSet}) \;=\; (N^*)_\perp \;.$$

The abstract interpretation interprets the various operations as the evident operations on finite sets. The interpretation of *exists*, for example, is strict in all three arguments and is such that

$$A(\mathit{exists})(n, m, x) \quad = \quad \begin{cases} \mathit{true} & \text{if } \exists k.\, m \le k \le n \wedge k \in x \\ \mathit{false} & \text{if } \neg\exists k.\, m \le k \le n \wedge k \in x \end{cases}$$

For the concrete interpretation, *exists* works by looking through a list for an element, *include* by consing an element onto the front of the list, and so on.

These interpretations work for both the first-order PCF($FinSet$) ADT and the variation with the second-order operation new. For example, the abstract interpretation of $empty$ is simply the empty set, while the abstract interpretation of new is the function $\lambda f.\, f(\emptyset)$. Theorem 3.7 shows in both cases how the models lift to the whole of PCF($FinSet$). ∎

4 L-relations and Soundness

We now consider how two interpretations can be linked by a logical relation.

Example 4.1 In our example of PCF with ω-**Cppo**, a logical relation between two interpretations $A, C\colon L(X) \to M$ is a family of subsets $R_t \subseteq C_t \times A_t$, subject to a completeness condition, where the $\times$ and $\to$ cases are determined inductively.

$$\begin{aligned} R_{t\times t'} &= \{((x,y),(x',y')) \mid xR_tx' \wedge yR_{t'}y'\} \\ R_{t\to t'} &= \{(f,g) \mid xR_ty \Rightarrow fxR_{t'}gy\} \end{aligned}$$

∎

Categorically, logical relations have been analysed in terms of structure-preserving functors (L-algebra maps) between the semantic category M and a category of relations [10]. The axiomatic conditions we require of the functors are that they form a reflexive graph. A reflexive graph in L-**Alg** consists of two L-algebras E and M and structure-preserving functors between them like so:

$$\begin{tikzcd} E \arrow[r, bend left, "\mathsf{dom}"] \arrow[r, bend right, "\mathsf{cod}"'] & M \arrow[l, "\mathsf{id}"'] \end{tikzcd} \qquad (1)$$

These functors must be such that both of the composites $\mathsf{dom}\cdot\mathsf{id}$ and $\mathsf{cod}\cdot\mathsf{id}$ are the usual identity functor on M.

Example 4.2 (relations) Take M to be **Set** and let E be **Rel**, the category whose objects are triples (R, S, T) where $R \subseteq S \times T$. A morphism from (R, S, T) to (R', S', S') in **Rel** is a pair of functions $(f\colon S \to S', g\colon T \to T')$ such that $fxR'gy$ whenever xRy. The preservation condition is the familiar "logical relation" property. The exponentiation $(R', S', T')^{(R,S,T)}$ is (Z, S'^S, T'^T) where Z consists of those pairs of functions satisfying the logical relation property. The reflexive graph structure is obtained by taking dom and cod to be the evident functors, and defining id by $\mathsf{id}(S) = (\{(x,x) \mid x \in S\}, S, S)$ and $\mathsf{id}(f) = (f, f)$. Note that **Rel** is a full sub-CCC of the functor category $\mathbf{Set}^V$, where V is the category with objects $0, 1, 2$ and non-identity maps $0 \to 1$ and $0 \to 2$.

A slight variation on this example is to take M to be ω-**Cppo** and E to be the category whose objects are complete, pointed relations, in order to handle the fixed-point operator. In this variation E is again Cartesian closed, and is a full subcategory of M^V. ∎

Let $\emptyset$ denote the empty $\langle L, \mathcal{D}\rangle$-sketch. By the universal property for sketches, there is a unique structure-preserving functor $!_B\colon L(\emptyset) \to B$ for any L-algebra (B, b).

Definition 4.3 (L-relations) Let X be an $\langle L, \mathcal{D}\rangle$-sketch, and A and C be structure-preserving functors from $L(X)$ to M. Then an L-*relation* from C to A is a functor $q\colon L(X) \to E$ such that the right-hand triangle in the following diagram commutes and the restriction of q to $L(\emptyset)$ along $!_{L(X)}$ is structure-preserving:

$$\begin{tikzcd}
 & & E \arrow[d, "\langle \mathsf{dom}, \mathsf{cod}\rangle"] \\
L(\emptyset) \arrow[r, "!_{L(X)}"] & L(X) \arrow[ur, "q"] \arrow[r, "\langle C, A\rangle"'] & M \times M
\end{tikzcd}$$

■

Note that q as a whole is *not* required to be structure-preserving.

The idea behind the definition of L-relation is as follows. First, we require an account of lifting. We know that any model A_0 of a sketch X in M lifts to a structure-preserving functor from $L(X)$ to M. For the same reason, as we shall see below, any model of X in E lifts to a structure-preserving functor from $L(X)$ to E. So a first approximation for the definition of L-relation would be to take a structure-preserving functor $q\colon L(X) \longrightarrow E$ subject to equations linking dom with C and cod with A. However, we also want an account of composition, but, as is well known, logical relations, equivalently structure-preserving functors as above, do not compose in the right way to give a logical relation: we shall see more of that in Section 5. So we need something more subtle. To simply drop the requirement that q preserves any structure is too crude: we desire a soundness requirement to the effect that, given any map f in $L(X)$ between two objects that existed in L, we have $C(f) = A(f)$ if C is related to A. Putting this all together gives the above definition; we shall see through the course of the paper that it allows us to prove the results we have mentioned here: lifting, soundness, and composition.

Back to our formal development, we observe that any two interpretations A and C must agree on $L(\emptyset)$. That is, the composites

$$L(\emptyset) \xrightarrow{!} L(X) \xrightarrow{A} M$$

and

$$L(\emptyset) \xrightarrow{!} L(X) \xrightarrow{C} M$$

must be equal, since A and C preserve structure, and so both composites equal $!_M\colon L(\emptyset) \longrightarrow M$. Further, since $\mathsf{id}\colon M \to E$, the "identity" map in the reflexive graph, is structure-preserving, the composite

$$L(\emptyset) \xrightarrow{!_M} M \xrightarrow{\mathsf{id}} E$$

equals $!_E$, which in turn equals

$$L(\emptyset) \xrightarrow{!} L(X) \xrightarrow{q} E$$

Hence, q assigns the "identity" relation to any object in $L(\emptyset)$.

In terms of PCF, these observations imply that **nat** and **bool** receive equal interpretations under A and C, and that q assigns identity relations to them. In our framework, this follows from the structure-preservation requirement in the definition of L-relation, and is not an additional explicit requirement.

Now we state an additional condition which we need to treat soundness.

Definition 4.4 A reflexive graph is said to satisfy the *identity condition* if $\mathsf{dom}(f) = \mathsf{cod}(f)$ whenever $S, T \in \mathrm{ob}(M)$ and $f : \mathsf{id}(S) \to \mathsf{id}(T)$ in E. ■

An example of a reflexive graph that doesn't satisfy the identity condition is with **Set** and **Rel**, taking dom and cod as above, and with $\mathsf{id}(X) = (X \times X, X, X)$, the everywhere-true relation.

The identity condition expresses a property we expect of identity relations. We read it informally as saying that "if two maps $\mathsf{dom}(f)$ and $\mathsf{cod}(f)$ are in relation then they are equal." In the leading example of **Rel** the condition characterises the functor id uniquely. Consider, for example, any map $f : \mathsf{id}(\mathbf{1}) \to \mathsf{id}(X)$; since id must preserve CCC structure we know that $\mathsf{id}(\mathbf{1})$ is a terminal object in **Rel**, and hence f consists of two elements $a, b \in R \subseteq X \times X$, where $\mathsf{id}(X) = (R, X, X)$. The identity condition says that these two elements must be equal, so R is contained in the diagonal relation on X; the reverse containment follows from the conditions for reflexive graphs.

Although the motivation for the identity condition seems reasonable, we feel that the condition itself is still rather *ad hoc*. One problem is that its formulation depends on the fact that we are working with **Cat**; we will see that it needs to be altered slightly when we consider order. Also, one would prefer to have a more abstract condition, stated in more categorical terms. For the **Cat**-based formulation, we can give a simple sufficient condition.

Lemma 4.5 A reflexive graph (1) satisfies the identity condition if the functor id is full. ■

(Note that faithfulness of id follows automatically from conditions on reflexive graphs.) All naturally occurring examples of which we are aware satisfy the fullness requirement, and it is a simpler condition. But we will see that it needs to be modified substantially in the case of order. For all of these reasons (and the slightly different treatment in [21]) we do not feel that we have a full understanding of the significance of identities, or that our treatment is definitive.

Nevertheless, the identity condition is sufficient to secure the following soundness property, which we regard as a minimal sanity check.

Proposition 4.6 (Soundness) Suppose the reflexive graph (1) satisfies the identity condition, and there is an L-relation from C to A. If $f\colon !x \rightarrow !y$ is an arrow in $L(X)$ between $L(\emptyset)$-objects then $A(f) = C(f)$.

Proof We know from above that the composite $L(\emptyset) \xrightarrow{!_M} M \xrightarrow{\mathsf{id}} E$ is equal to $L(\emptyset) \xrightarrow{!_{LX}} L(X) \xrightarrow{q} E$. So, f is sent by q to $qf\colon \mathsf{id}(\,!_M x) \longrightarrow \mathsf{id}(\,!_M y)$ in E. Thus, by Definition 4.4, $\mathsf{dom}(qf) = \mathsf{cod}(qf)$. So $Af = Cf$ follows from the triangle in Definition 4.3. ∎

The idea behind this result is that types in the language $L(\emptyset)$ are our primary concern, and the ADT is introduced simply as a convenient means for constructing and structuring programs. A program between $L(\emptyset)$ types might make use of the abstract data type, but we do not have access to the internals of the type to enable us to differentiate between different but "abstractly the same" representations; thus, what ultimately matters is equality between programs in $L(X)$ (which may use the ADT), but between $L(\emptyset)$ types.

Finally, we present a lifting result, which shows how a relation for an ADT X extends to the whole language $L(X)$. This is analogous to both lifting results in data refinement and to the main lemma of logical relations.

Lemma 4.7 (Lifting) Let X be an $\langle L, \mathcal{D}\rangle$-*sketch*. Any models A_0, C_0 and q_0 of X such that

$$\begin{tikzcd}
 & E \arrow[d, "{\langle \mathsf{dom}, \mathsf{cod} \rangle}"] \\
X \arrow[ur, "q_0"] \arrow[r, "{\langle C_0, A_0 \rangle}"'] & M \times M
\end{tikzcd}$$

commutes uniquely determine an L-relation q between structure-preserving functors A and C. Furthermore, q is structure-preserving.

Proof We get structure-preserving functors such that

$$\begin{tikzcd}
 & E \arrow[d, "{\langle \mathsf{dom}, \mathsf{cod} \rangle}"] \\
L(X) \arrow[ur, "q"] \arrow[r, "{\langle C, A \rangle}"'] & M \times M
\end{tikzcd}$$

commutes from Theorem 3.7, and $L(\emptyset) \xrightarrow{!} L(X) \xrightarrow{q} E$ is structure-preserving because q is. ∎

Example 4.8 Continuing Examples 3.2, 3.4, 3.6 and 3.8, we describe a complete relation $R_{FinSet} \subseteq C(FinSet) \times A(FinSet)$. This relation contains $(\bot, \bot)$ and all pairs $(\langle n_1, \ldots, n_k\rangle, \{n_1, \ldots, n_k\})$ where $\langle n_1, \ldots, n_k\rangle$ is a sequence of natural numbers. This provides a model of the ADT *FinSet* in **Rel** (the ω-**Cppo** version), because each operation preserves the relation. The lifting lemma lifts this interpretation to an L-relation for the whole of PCF(*FinSet*), and soundness tells us a sense in which the two interpretations are equivalent.

This relation can be used to treat both the first-order *FinSet* ADT and the variation with the second-order operation *new*. ■

5 Composition

Consider any example of refinement in which we have $M = \mathbf{Set}$ and $E = \mathbf{Rel}$. Suppose C and A are models, with q an L-relation from C to A. Then $C(X)$ and $A(X)$ are sets, and we should like to know precisely which elements a of $A(X)$ are related by $q(X)$ to which elements c of $C(X)$. Next, given another L-relation q' from C' to C, we have the same situation for elements c of $C(X)$ and c' of $C'(X)$. We can now compose the L-relations, and that tells us which elements of $A(X)$ are modelled correctly by which elements of $C'(X)$. This holds equally in Hoare's situation of sets and functions, as we see in Section 7. So we axiomatize the notion of composition of L-relations.

Historically, there has been a fundamental problem here, because logical relations do not compose. However, we have been careful in our definitions to ensure that we do not have the full strength of the definition of logical relation, which would state that q must preserve L-structure. Our definition of L-relation has been more delicate: strong enough to allow our soundness result, but weak enough to include our leading example of logical relations and their composition, as we shall see by our main result of this section.

The key point is that while our L-relations need not be (structure preserving) logical relations, they do need to map $L(\emptyset)$ objects to "identity relations." In terms of the specific examples L-relations are preserved by composition because although logical relations generally do not compose, identity relations do.

The fact that we cannot assume that composition preserves the structure leads us to pass from L-**Alg** to **Cat**, as we do not want to assert that our composition functor comp preserves L-structure. So, we require a functor $\mathsf{comp}\colon E \times_M E \to E$, where

$$\begin{tikzcd} E \times_M E \arrow[r, "\pi_1"] \arrow[d, "\pi_0"'] \arrow[dr, phantom, very near start, "\lrcorner"] & E \arrow[d, "\mathsf{cod}"] \\ E \arrow[r, "\mathsf{dom}"'] & M \end{tikzcd}$$

is a pullback in **Cat**, satisfying the associativity and unity properties necessary to make

$$\begin{tikzcd} E \times_M E \arrow[r, "\mathsf{comp}"] & E \arrow[r, bend left, "\mathsf{dom}"] \arrow[r, bend right, "\mathsf{cod}"'] & M \arrow[l, "\mathsf{id}"'] \end{tikzcd}$$

a category in **Cat**.

Example 5.1 (Example 4.1 continued) An object of $E \times_M E$ is a pair (R, S) of relations, where $R \subseteq Y \times X$ and $S \subseteq Z \times Y$. The functor comp takes (R, S) to its composite. For a counterexample to structure preservation, if $R = (\emptyset, X, X)$ and $S' = (\emptyset, Y, Y)$ then $\mathsf{comp}(R', S')^{\mathsf{comp}(R,S)}$ need not equal $\mathsf{comp}(R'^R, S'^S)$. But note that the graph of $\mathsf{comp}(R'^R, S'^S)$ is always a subset of $\mathsf{comp}(R', S')^{\mathsf{comp}(R,S)}$ in **Rel**.

When we move to second-order functions this inclusion is flipped, so that if (f, g) and (g, h) are second-order functions related by logical relations $R_1^{(R_2^{R_3})}$ and $S_1^{(S_2^{S_3})}$, it does not follow that that (f, h) is in the logical relation obtained by applying the relational exponent to the composites of R_i's and S_i's. So it is not evident how the idea of using the logical relation generated by composites at base types could work for stepwise refinement. ■

We remark that our use of id as the name of the functor from M to E is now justified: it is the identity construction for an internal category, the functor from the category of objects to the category of morphisms that delivers identity morphisms.

Now fix a category in **Cat** as above, with underlying reflexive graph from L-**Alg**. Recall that, since there is a unique map $!_B$ in L-**Alg** from $L(\emptyset)$ to any L-algebra (B, b), then an ordinary functor from $L(\emptyset)$ into B is an L-algebra map if and only if it is equal to $!_B$. We let $\mathsf{id}_{ME}\colon E \to E \times_M E$ denote $(\mathsf{id} \cdot \mathsf{cod}, \mathrm{id}_E)$, the unique functor for which

$$\begin{tikzcd} E \arrow[r, dashed, "\mathsf{id}_{ME}"] \arrow[rr, bend left, "\mathrm{id}_E"] \arrow[d, "\mathsf{cod}"'] & E \times_M E \arrow[r, "\pi_1"] \arrow[d, "\pi_0"] \arrow[dr, phantom, "\lrcorner", very near start] & E \arrow[d, "\mathsf{cod}"] \\ M \arrow[r, "\mathsf{id}"] \arrow[rr, bend right, "\mathrm{id}_M"'] & E \arrow[r, "\mathsf{dom}"] & M \end{tikzcd}$$

commutes.

Lemma 5.2 Let $q, q'\colon L(X) \to E$ be functors such that $q \cdot !_{L(X)} = !_E = q' \cdot !_{L(X)}$

and $\mathsf{dom} \cdot q = \mathsf{cod} \cdot q'$. Then $\mathsf{comp} \cdot (q, q') \cdot !_{L(X)} = !_E$.

$$\begin{tikzcd}
& E \times_M E \arrow[d, shift left, "\mathsf{comp}"] \\
L(X) \arrow[ur, "(q, q')"] & E \arrow[u, shift left, "\mathsf{id}_{ME}"] \\
L(\emptyset) \arrow[u, "!_{L(X)}"] \arrow[ur, "!_E"'] &
\end{tikzcd}$$

Proof First, we show that $(q, q') \cdot !_{L(X)} = !_{E \times_M E}$ as in the above diagram. Since $q \cdot !_{L(X)}$ and $q' \cdot !_{L(X)}$ are L-algebra maps, so is $(q \cdot !_{L(X)},\, q' \cdot !_{L(X)})$, since pullbacks lift from L-**Alg** to **Cat**. But $\pi_0 \cdot (q, q') \cdot !_{L(X)} = q \cdot !_{L(X)}$ and similarly for q', so $(q, q') \cdot !_{L(X)} = (q \cdot !_{L(X)},\, q' \cdot !_{L(X)})$ by the uniqueness of the mediating arrow of the pullback in **Cat**. Thus $(q, q') \cdot !_{L(X)}$ is an L-algebra map, and so $(q, q') \cdot !_{L(X)} = !_{E \times_M E}$.

Now, since id_{ME} and $!_E$ are L-algebra maps, it follows that $\mathsf{id}_{ME} \cdot !_E = !_{E \times_M E}$; and since comp is a retract of id_{ME}, we have

$$\mathsf{comp} \cdot (q, q') \cdot !_{L(X)} = \mathsf{comp} \cdot !_{E \times_M E} = \mathsf{comp} \cdot \mathsf{id}_{ME} \cdot !_E = !_E.$$

■

As an immediate consequence of this we have

Proposition 5.3 If q and q' are L-relations from C to A and C' to C, respectively, then $\mathsf{comp} \cdot (q, q')$ is an L-relation from C' to A. ■

The ability to compose L-relations can now be phrased in terms of a category of such. Specifically, given a sketch X, we define the category L-$\mathbf{Rel}(X)$: the objects are structure-preserving functors from $L(X)$ to M, the arrows are L-relations between such functors, composition is given by $\mathsf{comp} \cdot (q, q')$, and the identity on $A\colon L(X) \to M$ is $\mathsf{id} \cdot A$, which can easily be shown to be an L-relation from A to A.

Example 5.4 The concrete interpretation of *FinSet* in Example 4.8 uses lists, and we do not have lists available in our (impractical!) version of PCF. So we refine further. We introduce a third interpretation C' where $C'(\mathit{FinSet})$ is $(N_\perp \to N_\perp) \times N_\perp$, corresponding to the PCF type $(\mathbf{nat} \to \mathbf{nat}) \times \mathbf{nat}$. We write programs that implement list operations in terms of this representation. The relation we use to show correctness relates a list $\langle n_1, \ldots, n_k \rangle$ to (f, j) if $j = k$

and $\langle f(1), \ldots, f(k) \rangle = \langle n_1, \ldots, n_k \rangle$. This finally gives us an implementation of finite sets in PCF, and composing the relations here and in Example 4.8 gives us a relation saying which concrete values model which abstract values. By Proposition 5.3, we know that this composite is an L-relation. ∎

6 The Locally Ordered Case

In defining the notion of soundness, we have required that for any two objects x and y in $L(\emptyset)$, and for any map $f: x \to y$ in $L(X)$, we have $A(f) = C(f)$. In his account of refinement, Hoare proposed to replace equality with order, to allow for the possibility of the concrete interpretation being an "improvement" on the abstract one. Thus, we wish to extend our previous work to allow for order.

All the work of the previous sections, except for that directly about soundness, extends immediately from categories, functors, and structure preserving functors, to locally ordered categories, locally ordered functors, and structure-preserving locally ordered functors. By a locally ordered category, we mean a category together with the structure of a poset on each homset, respecting composition on both sides. The reason all the work extends is ultimately because everything we have said about sketches applies in any locally presentable category, for which **Cat** is one example and **LocOrd** is another; and other than sketches, we have only used elementary properties of a category with pullbacks, and any locally presentable category has them. So, extending our definitions as above, an L-relation now becomes a locally ordered functor from $L(X)$ to E subject to some conditions, a model of a sketch X lifts uniquely to $L(X)$, and composition is treated the same way.

Example 6.1 To account for order we make several changes in our examples. First, we consider ω-**Cppo** as enriched over itself, and add the requirement that **Y** is the least fixed-point operator. The category of such ω-**Cppo**-enriched categories is monadic over **LocOrd**. Next, Example 4.1 with $M = \omega$-**Cppo** uses a different category of relations. Category $E = \mathbf{Rel}^{\leq}$ has as objects complete relations such that if $x' \leq x$ and xRy and $y \leq y'$, then $x'Ry'$, and the identity functor is modified by putting $\mathsf{id}(D) = (\leq_D, D, D)$. This restriction on the objects is precisely what is needed to get a category object with this id as the unit. ∎

The revised identity condition is as follows.

Definition 6.2 The reflexive graph (1) is said to satisfy the *identity condition* if $\mathsf{dom}(f) \geq \mathsf{cod}(f)$ whenever $S, T \in \mathrm{ob}(M)$ and $f: \mathsf{id}(S) \to \mathsf{id}(T)$ in E. ∎

The motivation for this case is essentially the same as in the case for equality. In fact, it includes the previous condition as a special case, when the order on homsets in M and E is discrete. And we have, by essentially the same proof as before,

Proposition 6.3 (Soundness) Suppose the reflexive graph (1) satisfies the identity condition, and there is an L-relation from C to A. If $!x \xrightarrow{f} !y$ is an arrow in $L(X)$ between $L(\emptyset)$-objects then $A(f) \leq C(f)$. ■

Example 6.4 In the interpretation of the *exists* operation, we might require $exists(m, n, x)$ to have the expected behaviour only when $m \leq n$, as in [19]; that is, the specification of the operation has $m \leq n$ as a precondition. To handle this, we change the abstract interpretation of exists so that $exists(m, n, x) = \perp$ if $m > n$, and otherwise leave it as before. We interpret $\perp$ here as a "don't care" condition, a form of underspecification. Now, when $m > n$ the concrete interpretation may return a non-$\perp$ element, say $true$. To reason about these we use a relation between abstract and concrete interpretations containing $(\perp, \perp)$ and all pairs $(\langle n_1, \ldots, n_k\rangle, \{n_1, \ldots, n_k\})$ as before, but also containing $(\langle n_1, \ldots, n_k\rangle, \perp)$. This is a relation in $\mathbf{Rel}^{\leq}$, so we use lifting and soundness to conclude that $AP \leq CP$ if P is a closed term of type **nat**.

More realistic examples of this kind can be found in the refinement literature, using nondeterminism to allow for underspecification. ■

This all makes the case of order look very similar to that for equality. But there are some differences. One, as we have already seen, is the need to alter the category of relations in the example of ω-**Cppo**. Another is that the fully-faithful characterisation does not generalise directly.

We need some work to generalise the fully faithful characterisation. First, note that $\mathsf{id}: M \to E$ being fully faithful may be expressed equivalently as the assertion that, if the (bijective on objects, fully faithful)-factorizaton of id is given by

$$\begin{tikzcd} M \arrow[r, "\kappa"] \arrow[dr, "\mathsf{id}"'] & I \arrow[d, "\rho"] \\ & E \end{tikzcd}$$

then κ is an isomorphism. This is equivalent to the assertion that the composite functor from I to E to $M \times M$ is isomorphic to the diagonal functor $\Delta: M \to M \times M$.

Now, given any locally ordered category M, we may define the locally ordered category $\widetilde{M}$ to have the same objects as M, with an arrow in $\widetilde{M}$ from S to T being given by a pair of maps (g, h) in M from S to T, with $g \geq h$. The rest of the data to make $\widetilde{M}$ into a locally ordered category is evident. We let $\Delta: \widetilde{M} \longrightarrow M \times M$ denote the locally ordered functor that is the diagonal on objects and takes (g, h) to (g, h). Then we can finally state

Lemma 6.5 A reflexive graph (1) in **LocOrd** satisfies the identity condition if there is an isomorphism $j\colon \widetilde{M} \longrightarrow I$, such that the following diagram commutes.

$$\begin{tikzcd}
I \arrow[r, "\rho"] & E \arrow[d, "\langle \mathsf{dom}, \mathsf{cod} \rangle"] \\
\widetilde{M} \arrow[u, "j"] \arrow[r, "\Delta"'] & M \times M
\end{tikzcd}$$

■

7 Data Refinement

We conclude with a detailed analysis of the relationship between the definitions here and Hoare's approach to data refinement. Hoare defines a refinement from a functor $C\colon L(X) \to M$ to $A\colon L(X) \to M$ to be a natural transformation from C to A. We may regard the naturality diagram

$$\begin{tikzcd}
Ax \arrow[r, "Af"] & Ay \\
Cx \arrow[u, "\alpha_x"] \arrow[r, "Cf"'] & Cy \arrow[u, "\alpha_y"']
\end{tikzcd}$$

as a correctness requirement, saying that the concrete program Cf followed by the abstraction function α_y gives the same result as the abstraction function α_x followed by the abstract program.

To see this in our terms, take E to be the arrow category $M^{\to}$. A morphism from $g\colon S \to T$ to $g'\colon S' \to T'$ in $M^{\to}$ is a commutative square

$$\begin{tikzcd}
T \arrow[r, "h"] & T' \\
S \arrow[u, "g"] \arrow[r, "k"'] & S' \arrow[u, "g'"']
\end{tikzcd}$$

There is an evident reflexive graph structure, with $E = M^{\to}$, id sending S to $\mathrm{id}_S\colon S \to S$, and dom and cod evident. Now, with this reflexive graph, we have

Proposition 7.1 Consider structure-preserving functors $A, C\colon L(X) \to M$; then there is a bijection between L-relations from C to A and natural transformations from C to A that are the identity on $L(\emptyset)$.

Proof The natural transformation $\gamma\colon \mathsf{dom} \Rightarrow \mathsf{cod}$ sends $f\colon x \to y$ to itself. The bijection then takes an L-relation q from C to A to the natural transformation $\gamma q\colon C \Rightarrow A$ obtained via composition. ■

For the statement of this proposition we have to assume that the monad L is obtained from covariant structure; i.e., when L is a 2-monad, as is the case for categories with small coproducts, products or monoidal structure. In those cases, $M^{\rightarrow}$ also has such structure, and the embedding into **Rel** is essentially the embedding of $M^{\rightarrow}$ into M^V sending $f: S \to T$ to (id_S, f). The case of order can also be treated by replacing natural transformations by lax transformations and $M^{\rightarrow}$ by $Oplax(\mathbf{2}, M)$ [7, 8]. Note that the use of 2-categorical, quasi-2-categorical, and enriched structure in the treatments of data refinement based on natural (or lax) transformations [5, 7] is not required in the logical-relation formulation.

Several remarks are in order. First, for the reflexive graph with $E = M^{\rightarrow}$, the notion of L-relation is *asymmetric*: if there is a refinement (= L-relation or natural transformation) from C to A, then it does not imply that there is a refinement from A to C. The point is that our axiomatic setup does not build in symmetry, so we can account for cases where asymmetry is natural.

Second, in contrast to **Rel**, the functor dom does not in this case preserve exponents, so this graph does not immediately give an analysis of higher-order structure. However, an analysis of a first-order ADT, done exclusively via Hoare's method of abstraction functions, can be consistent with higher-order structure: embed $M^{\rightarrow}$ in another E that supports higher order. If we take E to be **Rel**, we can use the embedding that sends a function to the relation with the same graph.

Third, the way Hoare proposes to address higher-order structure is via total simulations; i.e., embedding-projection pairs. To illustrate the difference between the two approaches, consider an interpretation of *FinSet* (Example 3.2) in **Set**, where

$$\begin{aligned} A(FinSet) &= \mathcal{P}_{\mathit{fin}} N \\ C(FinSet) &= N^* \end{aligned}$$

We can reason about these representations using the relation that relates a finite set to a list with the same elements in some order, possibly with repetitions, and this immediately extends to higher order. But it is less clear how to treat this example using embedding-projection pairs. The natural relation here is not an embedding-projection pair, but it is still a relation that we can and should like to consider. So we regard the use of logical relations here as better in this regard than that of embedding-projection pairs.

We should like to emphasize the debt we owe to Hoare's work through the whole of this paper. In particular, his insistence that data refinement could be understood abstractly in terms of category-theoretic structure, and the composability of his data refinements, influenced our definition of L-relation, which we regard as the fundamental definition of the paper.

References

[1] Michael Barr and Charles Wells. *Toposes, Triples and Theories*, volume 278 of *Grundlagen der mathematischen Wissenschaften*. Springer-Verlag, 1985.

[2] D. Gries, editor. *Programming Methodology, A Collection of Articles by IFIP WG 2.3*. Springer-Verlag, New York, 1978.

[3] Claudio A. Hermida. *Fibrations, Logical Predicates, and Indeterminates*. PhD thesis, The University of Edinburgh, 1993. Published as CST–103–93, also as ECS–LFCS–93–277.

[4] C.A.R. Hoare. Data refinement in a categorical setting. Unpublished manuscript, 1987.

[5] C.A.R. Hoare and He Jifeng. Data refinement in a categorical setting. Oxford Computing Laboratory Technical Monograph PRG-90, 1990.

[6] G.M. Kelly and A.J. Power. Adjunctions whose counits are coequalizers, and presentations of finitary enriched monads. *Journal of Pure and Applied Algebra*, 89:163–179, 1993.

[7] Y. Kinoshita and A.J. Power. Data refinement and algebraic structure. ETL Technical Report TR96–2, Electrotechnical Laboratory, January 1996.

[8] Y. Kinoshita and A.J. Power. Lax naturality through enrichment. *Journal of Pure and Applied Algebra*, 112(1):53–72, 1996.

[9] Yoshiki Kinoshita, A. John Power, and Makoto Takeyama. Sketches. Submitted, 1996.

[10] QingMing Ma and J. C. Reynolds. Types, abstraction, and parametric polymorphism, part 2. In S. Brookes et al., editors, *Mathematical Foundations of Programming Semantics, Proceedings of the 7th International Conference*, volume 598 of *Lecture Notes in Computer Science*, pages 1–40. Springer-Verlag, Berlin, 1992, Pittsburgh, PA, March 1991.

[11] J. C. Mitchell. Representation independence and data abstraction. pages 263–276.

[12] J.C. Mitchell and A. Scedrov. Notes on sconing and relators. In E. Boerger et al., editors, *Computer Science Logic: 6th Workshop, CSL '92: Selected Pape rs*, volume 702 of *Lecture Notes in Computer Science*, pages 352–378, San Miniato, Italy, 1992. Springer-Verlag, Berlin.

[13] P. W. O'Hearn and R. D. Tennent. Parametricity and local variables. *J. ACM*, 42(3):658–709, May 1995. Also in [14].

[14] P.W. O'Hearn and R.D. Tennent, editors. *Algol-like Languages*, volume 2. Birkhauser, Boston, 1997.

[15] A. M. Pitts. Relational properties of domains. *Information and Computation*, 127:66–90, 1996.

[16] G. Plotkin and M. Abadi. A logic for parametric polymorphism. In M. Bezen and J. F. Groote, editors, *Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*, pages 361–375, Utrecht, The Netherlands, March 1993. Springer-Verlag, Berlin.

[17] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–255, 1977.

[18] A. J. Power. Premonoidal categories as categories with algebraic structure. Submitted.

[19] J. C. Reynolds. User-defined types and procedural data structures as complementary approaches to data abstraction. In S. A. Schuman, editor, *New Advances in Algorithmic Languages 1975*, pages 157–168. Inst. de Reserche d'Informatique et d'Automatique, Rocquencourt, France, 1975. Reprinted in [2], pages 309-317.

[20] J. C. Reynolds. Types, abstraction and parametric polymorphism. In R. E. A. Mason, editor, *Information Processing 83*, pages 513–523. North Holland, Amsterdam, 1983.

[21] E. P. Robinson. Logical relations and data abstraction. Extended abstract.

[22] E. P. Robinson and G. Rosolini. Reflexive graphs and parametric polymorphism. In *Proceedings, 9th Annual IEEE Symposium on Logic in Computer Science*, Paris, 1994. IEEE Computer Society Press, Los Alamitos, California.

[23] D. S. Scott. A type-theoretical alternative to CUCH, ISWIM, OWHY. Privately circulated memo, Oxford University, October 1969. Published in *Theoretical Computer Science*, 121(1/2):411–440, 1993.

[24] H. Thielecke. Continuation passing style and self adjointness. *Continuations Workshop*, Paris, January, 1997.