Logical Relations, Data Abstraction, and Structured Fibrations*

John Power
Laboratory for the Foundations of Computer Science
University of Edinburgh
King's Buildings, Edinburgh EH9 3JZ
Scotland
ajp@dcs.ed.ac.uk

Edmund Robinson
Department of Computer Science
Queen Mary and Westfield College
Mile End Road, London, E1 4NS
England
edmundr@dcs.qmw.ac.uk

ABSTRACT

We develop a notion of equivalence between interpretations of the simply typed λ-calculus together with an equationally defined abstract data-type, and we show that two interpretations are equivalent if and only if they are linked by a logical relation. We show that our construction generalises from the simply typed λ-calculus to include the linear λ-calculus and calculi with additional type and term constructors, such as those given by sum types or by a strong monad for modelling phenomena such as partiality or nondeterminism. This is all done in terms of category theoretic structure, using fibrations to model logical relations following Hermida, and adapting Jung and Tiuryn's logical relations of varying arity to provide the completeness results, which form the heart of the work.

Categories and Subject Descriptors

D.3.1 [**Programming Languages**]: Formal Definitions and Theory; F.3.2 [**Logics and Meanings of Programs**]: Semantics of Programming Languages; F.4.1 [**Mathematical Logic and Formal Languages**]: Mathematical Logic

General Terms

Theory

Keywords

logical relations, cartesian closed fibrations, interpretations, λ-calculus, linear λ-calculus.

1. INTRODUCTION

Given a signature in the simply typed λ-calculus, which we assume for simplicity comes equipped with product types, consider the cartesian closed category L whose objects are types generated by the signature and whose arrows are equivalence classes of terms in context. A model of the simply typed λ-calculus together with the signature in a semantic cartesian closed category such as Set amounts to a cartesian closed functor $M : L \longrightarrow Set$. Any such M is uniquely determined by its behaviour on the signature. Now suppose we want to add an abstract data-type such as **stack** to the language and consider its semantics. To do that, we can freely add an object **stack** together with operations and equations to our category L and close the resulting category under cartesian closed structure, yielding a category we could reasonably denote by $L[\mathbf{stack}]$. An interpretation of $L[\mathbf{stack}]$ in Set is then a cartesian closed functor from $L[\mathbf{stack}]$ to Set that extends M. But then the question arises: given two interpretations $F, G : L[\mathbf{stack}] \longrightarrow Set$, when does one say F and G are equivalent interpretations, and is there a body of theory to support that definition? That, generalised from **stack** and Set, and further generalised from the simply typed λ-calculus, is the question we address in this paper.

There are two possible answers, in category theoretic terms, to our question. First, we could give a category theoretic formulation of the idea that two parallel maps g and h in the semantic category Set are contextually equivalent, denoted $g \approx h$, with respect to observable contexts and observable types in L, then we could say F is equivalent to G if for every map in $L[\mathbf{stack}]$ of the form $f : iX \longrightarrow iY$, where i is the inclusion of L in $L[\mathbf{stack}]$, we have $Ff \approx Gf$. We make precise this definition and prove a few relevant results in Sections 2 and 3.

A second possible answer is more subtle. John Reynolds [17] proposed that two interpretations F and G be called equivalent if they are linked by a logical relation. The point of logical relations is that one builds a relation between base types, and that relation is preserved by the type constructors of the language. Moreover, even for base types, the relations are typically given by simple logical constructions. So this notion of equivalence between interpretations must, by definition, respect the structure of the language. A category theoretic formulation of the notion of logical relation was

*This work has been done with the support of EPSRC grants GR/L54639 and GR/M56333, ESPRIT Working Group 216142 (APPSEM) and a British Council grant.

given by Claudio Hermida [3], so we can give an axiomatic version of Reynolds' definition in Section 4. Upon doing so, it is immediate that if two interpretations are linked by a logical relation, then they are equivalent interpretations as given by our first answer. The converse is considerably less obvious, and the main technical content of this paper amounts to an analysis of it.

We give a converse not only for the simply typed λ-calculus but also for the linear λ-calculus and various extensions of either calculus to allow additional type and term constructors. This converse was previously known for algebraic theories with first order operations and equations, albeit not in category theoretic terms (see Mitchell [10, 11]), but the results here apply to fully higher-order theories, largely because of the category theoretic formalisation rather than Henkin models.

The main construction we need for a converse is given by a variant of work of Achim Jung and Jerzy Tiuryn [5], who used a form of logical relations to characterise lambda definability in the simple type hierarchy. But the work here is not based on Henkin models but rather on cartesian closed categories, so a better comparison is Alimohamed's characterisation of lambda definability for cartesian closed categories in [1].

The notion of abstract type we study is quite restricted as we only consider equationally specified types, albeit with equations between higher-order operators. This is also the level of generality of Alimohamed's work. However, it raises a natural question of whether the link with logic exploited in this paper could be further exploited to allow a less restrictive specification language.

Data abstraction is closely related to data refinement, and the fundamental notion of this paper, that of being "linked by a logical relation", is closely related to the notion of L-relation in [6] (see also [18]). The work of that paper, and of this paper, and those on lax and pre-logical relations [4, 13] clearly need to be combined at some point.

An obvious direction for further work is to address models of call-by-value lambda-calculi such as the computational lambda-calculus [12]. Such analysis would probably best be done by moving from cartesian closed categories to some form of closed symmetric premonoidal categories as adumbrated in [15] and [14], most likely in the form of a closed Freyd-category.

This paper may be seen as a development of work in [16]. Formally, the main results here generalise those of [16], but the perspective is different: we can deduce the main result there from the work in this paper by putting $L = C$ and making M the identity in our main theorem; but [16] was a study of implementation rather than interpretation, so its emphasis was different.

The paper is organised as follows. We first give a category theoretic account of the theory of contextual equivalence in Section 2. We then proceed to abstract types, interpretations, and equivalence of interpretations in Section 3. In Section 4, we outline the relationship between logical relations and fibrations, and we state the central result of the paper to be proved in later sections. In Section 5, we give the central construction of the paper. In Section 6, we give a completeness result for simply typed λ-calculus: in Section 7, we give a completeness result for linear λ-calculus, and in Section 8, we give some extensions of our main completeness result to other familiar type constructors such as sum types and strong monads, e.g., for modelling partiality.

2. CONTEXTUAL EQUIVALENCE

In this section, we give a categorical account of a standard Morris-style contextual equivalence. As we are working in the context of simple type theory, and the theory may well be strongly normalising, we use a denotational rather than an operational approach, and observe values produced by computations rather than termination.

The basic structure is as follows. We start with cartesian closed categories L and C together with a cartesian closed functor $M : L \longrightarrow C$. The cartesian closed category L is to be understood to be the cartesian closed category syntactically generated by the simply typed λ-calculus (for which one may harmlessly assume the existence of product types), together with a signature, for instance given by Nat, $Bool$, and some operators such as $0 : 1 \longrightarrow Nat$, $succ : Nat \longrightarrow Nat$, and $true, false : 1 \longrightarrow Bool$, subject to some equations. The category C is intended to be a semantic category. It represents the base category in which we take our denotational semantics. The cartesian closed functor M is a fixed interpretation of L in C, giving a set semantics for our core language. There are many possibilities for the category C, such as the category Set of small sets, or, if we want to be sure to include fixed point operators to model recursion, we could consider C to be the category of ω-cpo's with least element, and with maps being functions that preserve the poset structure and sup's of ω-chains. Other alternatives for C include categories generated by axiomatic domain theory [2], or presheaf categories, or toposes. We simply require that C be cartesian closed.

We further assume that we are given a set of observable pairs of types $Obs \subseteq ob(L) \times ob(L)$. The idea is that we are allowed to form equality judgements

$$\vec{x} : A \vdash e = e' : B$$

where $(A, B) \epsilon Obs$. An alternative would be to use a set of observable types, but this accords better with our interpretation of the simply typed λ-calculus, and will allow a simpler extension to dependent type theory. For example, one could take Obs to be given by allowing A's to be products of copies of Nat and allow Nat to be the only possible B.

We shall use the structure of observables to generate a contextual equivalence $\approx$, allowing us to consider $f \approx g$ where f and g are maps in C with common domain and codomain of the form MX and MY: type-theoretically, this amounts to demanding that contextually equivalent terms have the same type and are defined in the same context. Note here that $\approx$ is not defined on all homsets of C but only with respect to the image of M. Moreover, it is even more delicate than that in that we fix X and Y, then consider whether maps $f, g : MX \longrightarrow MY$ are congruent. So even if $M(X) =$

$M(X')$, we shall distinguish between the equivalences: there will be one for X and Y and another for X' and Y.

Definition 1. Given $f, g : MX \longrightarrow MY$, define $\sim$ by $f \sim g$ if for all $(A, B)\epsilon Obs$, for all $\alpha : A \longrightarrow X$ and for all $\beta : Y \longrightarrow B$, we have

$$MA \xrightarrow{M\alpha} MX \xrightarrow{f} MY \xrightarrow{M\beta} MB$$

equals

$$MA \xrightarrow{M\alpha} MX \xrightarrow{g} MY \xrightarrow{M\beta} MB$$

By a congruence, we mean an equivalence relation that is invariant under products and exponentials with respect to objects of L. The primary definition of this section is as follows.

Definition 2. We define $\approx$ to be the largest congruence contained in $\sim$.

We can characterise $\approx$ by the following proposition. We denote the exponential of X raised to the power U by $[U, X]$, dispensing with tiny subscripts and superscripts. We follow this convention for exponentials in a category, for exponentials of not only objects but maps, and for exponentials of categories, i.e., for functor categories, when we need them.

PROPOSITION 1. *Given $f, g : MX \longrightarrow MY$, the following are equivalent:*

1. $f \approx g$
2. *for all U, V in $ob(L)$, we have*
$$[MU, f] \times MV \sim [MU, g] \times MV$$
3. *for all U, V in $ob(L)$, for all (A, B) in Obs, and for all $\alpha : A \longrightarrow [U, X] \times V$ and $\beta : [U, Y] \times V \longrightarrow B$, we have*

$$\begin{tikzcd}
{[MU, MX] \times MV} \arrow[r, "{[MU, f] \times MV}"] & {[MU, MY] \times MV} \arrow[d, "M\beta"] \\
MA \arrow[u, "M\alpha"] & MB
\end{tikzcd}$$

equals

$$\begin{tikzcd}
{[MU, MX] \times MV} \arrow[r, "{[MU, g] \times MV}"] & {[MU, MY] \times MV} \arrow[d, "M\beta"] \\
MA \arrow[u, "M\alpha"] & MB
\end{tikzcd}$$

The proof is routine.

We can also make precise the relationship with Morris-style contextual equivalence. Any cartesian closed category can be regarded as a model of the simply typed λ-calculus where base types correspond to objects of the category, and for each morphism $f : X \longrightarrow Y$, we have a closed term $\hat{f}$ of type $X \rightarrow Y$. One can define $e \cong e'$ for terms e and e' if and only if for all relevant contexts $C[-]$, $M(C[e]) = M(C[e'])$. A relevant context is one that respects Obs. We then have

PROPOSITION 2. *$e \cong e'$ if and only if $M(e) \approx M(e')$.*

PROOF. In one direction, this is a simple structural induction on the context, and in the other, it depends on the expressibility of substitution via abstraction, application, and beta reduction. □

Example 1.

1. $Obs = \{(1, Bool)\}$ with $C = Set$ and $Bool$ modelled by the two element set. Here, $\approx$ coincides with the version of normal Morris-style contextual equivalence in which contexts produce closed terms of type $Bool$, and where we allow ourselves to observe equality.
2. Obs is given by the set of pairs consisting of finite products of copies of Nat and Nat, with the usual modelling of Nat in Set. Here, $\approx$ means contexts produce terms of type Nat, and we allow ourselves to observe the natural number given by applying a program to a sequence of natural numbers.
3. $Obs = \{(1, S)\}$ where S is $1_\perp$ space. Here, $M(S)$ contains two elements representing termination and non-termination. This variant gives us contextual equivalence where contexts produce closed terms of unit type, and in which we allow ourselves to observe termination. The order on $M(S)$ generates a pre-order on terms. This is the pre-order used to quotient the fully abstract model of PCF by Milner [9]. So, if we have an observable type which contains the values for both termination and non-termination, we can handle the conventional form of contextual equivalence.
4. $Obs = ob(L) \times ob(L)$ and $L = C$. When everything is observable, $\approx$ is just equality.

3. ABSTRACT TYPES

An equationally specified abstract type consists of a new type (or more generally, family of types) equipped with operations, linking it (them) to pre-existing types, and certain equations on terms built from those operations. The canonical example is **stack**. For our purposes, we may as well have a number of new types, and we interpret the operations as (open) terms in context

$$\vec{x} : \vec{X} \vdash e : Y$$

where the $\vec{X}$ and Y may be arbitrary type expressions built up from old and new types. This generates an extended lambda calculus in which the equations are interpreted as equality judgements

$$\vec{x} : \vec{X} \vdash e = e' : Y$$

Such a system freely generates a cartesian closed category D which comes equipped with a cartesian closed functor $i : L \longrightarrow D$. This functor is injective on objects, but in general is neither full nor faithful as we can add extra operations between pre-existing types and impose new equations between pre-existing operations. Conversely, given any injective on objects cartesian closed functor $i : L \longrightarrow D$, the category D is equivalent to a category constructed from L in this fashion, as we may generate it using all objects in $D \backslash i(L)$ and take all valid equations. So we can take $i : L \longrightarrow D$ as the extension of L by some abstract types.

We can split this construction into two stages: first adjoin types and operations freely to give $i : L \longrightarrow D$, and then quotient by the equations to give $q : D \longrightarrow D/\approx_E$. In fact, we shall never use the free property of D and can take it to be an arbitrary cartesian closed category. The free property of such a D is fundamental to the work generalising logical relations for an account of data refinement in [6], in which it was denoted $L[X]$: there was a precise category theoretic definition of an abstract data-type X, and $L[X]$ was the language L extended by X.

Definition 3. An *interpretation* of D in C is a cartesian closed functor $F : D \longrightarrow C$ such that $Fi = M$.

The category theoretic tradition would be to make a slightly more general notion of interpretation than that we have just defined. A more natural category theoretic definition would allow for an isomorphism between Fi and M. The introduction of such an isomorphism should then be propagated through the paper. However, it is a routine generalisation that would detract from our main point, and as the category D is given syntactically anyway in the examples, it is much less an issue than if D was given semantically. So, for simplicity of exposition, we just consider equality here.

Example 2. Let C be *Set* and let D consist of L together with the formal addition of a type **stack** together with its operations and equations. More, C may be any cartesian closed category, such as the category of ω-cpo's with least element and with ω-continuous maps, and with D as before. An interpretation amounts to a realisation of the type **stack** in C together with its operations and subject to its equations.

Now suppose we have two interpretations $F, G : D \longrightarrow C$, and consider conditions under which we regard the two interpretations as equivalent.

Recall that L stands for "language" such as the simply typed λ-calculus with some base types and operations such as *Nat* and *Bool*, C is a semantic category like *Set* or a category of ω-cpo's, and $M : L \longrightarrow C$ is an intended interpretation of L in C. This agrees with the setting of [6]. The observable pairs are pairs of objects of L, and D is understood to be of the form $L[X]$, i.e., given by L together with new types and operations for something like **stack**, closed under cartesian closedness while respecting the cartesian closed structure of L.

Definition 4. We say interpretations $F, G : D \longrightarrow C$ are *equivalent* if for all $f : iX \longrightarrow iY$, we have $Ff \approx Gf$.

This definition extends Lemma 2.3(3) of [16]. There is a crucial point here for our later generalisations beyond the simply typed λ-calculus: our definition of equivalence uses nothing of the specific nature of $\approx$; it only uses the fact that $\approx$ is given by an equivalence relation, for fixed X and Y, on the homset $C(MX, MY)$.

PROPOSITION 3. *Two interpretations F and G are equivalent if and only if for all $f : iA \longrightarrow iB$ where (A, B) is in Obs, one has $Ff = Gf$.*

The proof is mundane, and the condition extends Lemma 2.3(4) of [16]. The condition itself would be a reasonable definition of equivalence of interpretations, so this result, although easy, is encouraging. Note for when we extend beyond the simply typed λ-calculus, this result extends automatically providing we continue to define $\approx$ to be the largest congruence contained in $\sim$: the only difference there is that the notion of congruence varies according to the type constructors of the specific language at hand.

4. LOGICAL RELATIONS AND DIAGONAL FUNCTORS

In [8], Ma and Reynolds analysed logical relations for simple types. For binary relations, they considered a cartesian closed category B equipped with a cartesian closed fibration $p = (p_0, p_1) : B \longrightarrow C \times C$. B is a category of relations. The fact that B is cartesian closed allows the definition of logical relatons by induction from base types. The fact that p preserves cartesian closed structure yields the fundamental lemma of logical relations. The fact that p is a fibration is incidental, but is a property which arises naturally when the relations have a good logical structure. We recall the notion of fibration [3].

Definition 5. A functor $p : E \longrightarrow E'$ is called a *fibration* if for every object X of E and every map of the form $g : A \longrightarrow pX$ in E', there is an arrow $g^* : Y \longrightarrow X$ in E such that $pY = A$ and $p(g^*) = g$, and the following universal property is satisfied:

for every arrow $f : Z \longrightarrow X$ and every arrow $h : pZ \longrightarrow pY$ such that $g \cdot h = p(f)$, there exists a unique arrow $k : A \longrightarrow Y$ such that $p(k) = h$ and $g^* \cdot k = f$.

The notion of fibration may be re-expressed by saying that it consists of a pair of categories and a functor $p : E \longrightarrow E'$ satisfying a condition that is equivalent to the assertion that each arrow $f : A \longrightarrow B$ in E' induces a functor $f^* : p^{-1}(B) \longrightarrow p^{-1}(A)$ and that these functors are collectively functorial in E' up to coherent natural isomorphism. Logically, this amounts to allowing the substitution of a term for a variable in a logical formula.

We make a slightly non-standard definition:

Definition 6. A fibration $p : E \longrightarrow E'$ is called *cartesian closed* if E and F are cartesian closed and p is a cartesian closed functor.

In order to obtain logical relations, following Lawvere [7], one typically uses a logic interpreted as a fibration with structure. If the logic admits $\top$, $\wedge$, $\rightarrow$ and $\forall$, then, provided the base is itself cartesian closed, it yields a cartesian closed fibration. The observation that these connectives are all that is required to define product and exponential of logical relations should make that immediately credible, even if it does not furnish a proof. This is of course more than the structure required in order to define the notion of cartesian closed fibration: we only need the total category to be cartesian closed rather than requiring each fibre to be cartesian closed in order to model logical relations; the role of logic in logical relations is as a predicate over a cartesian closed category. Ma and Reynolds constructed B as a category of relations exactly as in the above example, and we shall take the structure of a cartesian closed fibration together with a cartesian closed functor from L into it as a characterisation of logical relations. See [3] for a careful analysis.

A standard example of a cartesian closed fibration is as follows.

Example 3. Let C be any category with finite limits. Let $Sub_2(C)$ have objects $(X, Y, R \rightarrow X \times Y)$ consisting of a pair (X, Y) of objects of C together with a subobject R of $X \times Y$, and with the evident arrows. Then the forgetful functor $Sub_2(C) \longrightarrow C \times C$ is a fibration. If C is cartesian closed, the fibration is a cartesian closed fibration.

Observe that this example also has a standard diagonal $\Delta : C \longrightarrow B$, which is a cartesian closed functor: that diagonal was fundamental to [8]. We require an abstract characterisation of the diagonal too, and we want the diagonal to characterise contextual equivalence rather than equality.

Recall that we are given cartesian closed categories L and C, a cartesian closed functor $M : L \longrightarrow C$, and a set Obs of pairs of objects of L. Using all of this, we have defined and characterised $\approx$.

Definition 7. Let $p = (p_0, p_1) : B \longrightarrow C \times C$ and $Q : L \longrightarrow B$ be cartesian closed functors such that $p \cdot Q = (M, M) : L \longrightarrow C \times C$. We say Q is *diagonal* if, for all $f : QX \longrightarrow QY$, it follows that $p_0 f \approx p_1 f$.

We regard two maps $g, h : MX \longrightarrow MY$ as being in relation if and only if there is a map $f : QX \longrightarrow QY$ such that $p_0 f = g$ and $p_1 f = h$. So this definition says that Q is diagonal if and only if, when two maps are in relation, they are contextually equivalent.

Now we can make the central definition of the paper.

Definition 8. Two interpretations F and G are *linked by a logical relation* if there is a cartesian closed category B and a cartesian closed fibration $p : B \longrightarrow C \times C$ together with a cartesian closed functor $R : D \longrightarrow B$ lying over (F, G), such that $Q = R \cdot i$ is diagonal.

$$\begin{tikzcd} & & B \arrow[d, "p = (p_0, p_1)"] \\ L \arrow[r, "i"'] & D \arrow[ur, "R"] \arrow[r, "(F,G)"'] & C \times C \end{tikzcd}$$

The soundness result we want, i.e., that two interpretations F and G are equivalent if they are linked by a logical relation, follows immediately from the definition. Completeness is more difficult, and an analysis of it and its extensions provide the content of the rest of the paper.

5. THE CENTRAL CONSTRUCTION

In this section, we give the central construction we need to prove the completeness part of our main result. Given equivalent interpretations $F, G : D \longrightarrow C$, we seek a cartesian closed fibration $p : B \longrightarrow C \times C$ and a cartesian closed functor from D to B. Our construction is motivated by Jung and Tiuryn's construction of Kripke logical relations of varying arity [5]. In this section, we shall largely ignore the cartesian closedness condition and concentrate only on the fibration and the diagonal conditions.

The key construction we need for our completeness result is as follows. Given a functor $F : D \longrightarrow C$, one can turn it around to obtain a functor $\tilde{F} : C \longrightarrow [D^{op}, Set]$ sending X to the functor $C(F-, X) : D^{op} \longrightarrow Set$. Now we can consider the subobject fibration over $[D^{op}, Set]$, and we can take the pullback of that cartesian closed fibration along $\tilde{F}$ to obtain a cartesian closed fibration over C. We shall take a binary version of this.

This involves some delicacy. First, taking a binary version of the above construction is a most unusual construct to make on a fibration in the following sense. It would be normal, given a functor $K : C \longrightarrow [D^{op}, Set]$ to consider the pullback of a fibration over $[D^{op}, Set] \times [D^{op}, Set]$ along $K \times K$. But that is not what we propose. We have functors $\tilde{F}, \tilde{G} : C \longrightarrow [D^{op}, Set]$ and we shall take the pullback along $\tilde{F} \times \tilde{G}$. We do obtain the result we seek, but this is a most unusual construction. For instance, there seems no way to lift $\tilde{F} \times \tilde{G}$ to a functor from the binary subobject fibration $Sub_2(C)$ over $C \times C$ to the binary subobject fibration $Sub_2([D^{op}, Set])$ over $[D^{op}, Set] \times [D^{op}, Set]$, but that is a fundamental construction of fibrations, simply not one we need here.

Second, although the functor F is cartesian closed, it does not follow that $\tilde{F}$ is cartesian closed. Coincidentally, $\tilde{F}$ does preserve finite products, which gives an easy proof for some of the results we seek, but as we shall see later, even that is a curiosity as we shall later consider a tensor product that generalises the role of finite products but which is not preserved by $\tilde{F}$.

Ideally, we would like the construction of our cartesian closed fibration $p : B \longrightarrow C \times C$ to be independent of the choice of

F and G, with F and G only being required in order to construct $R : D \longrightarrow B$. But we cannot see how to do that: hence the relationship with Jung and Tiuryn's work [5], which also has an extra parameter $J : X \longrightarrow Set$.

PROPOSITION 4. *Given a functor $K : F' \longrightarrow F$, and a fibration $p : E \longrightarrow F$, the pullback $K^*(E)$ of p along K gives a fibration $K^*(E)$ over F'.*

This is a standard result which can be verified by direct calculation. A proof appears in Hermida's thesis [3], which also contains much material we shall use later.

COROLLARY 1. *Given $F, G : D \longrightarrow C$, the pullback of $Sub_2([D^{op}, Set])$ along $\tilde{F} \times \tilde{G}$ gives a fibration we denote $(\tilde{F} \times \tilde{G})^*(Sub_2[D^{op}, Set])$ over $C \times C$.*

The fibration $(\tilde{F} \times \tilde{G})^*(Sub_2[D^{op}, Set])$ is the one we want, so we shall duly denote it by B. This is a *sconing* of the ordinary binary subobject fibration on the presheaf category $[D^{op}, Set]$, cf. [3].

Definition 9. Given functors $F, G : D \longrightarrow C$, define $R : D \longrightarrow B$ by making Rd the subfunctor of $C(F-, Fd) \times C(G-, Gd)$ given by the epi-mono factorisation of the natural transformation

$$D(-, d) \longrightarrow C(F-, Fd) \times C(G-, Gd)$$

determined by functoriality of F and G. More explicitly, given an object d of D,

$$Rd(c) = \{(g : Fc \longrightarrow Fd, h : Gc \longrightarrow Gd) \mid$$

$$\exists f : c \longrightarrow d. Ff = g \wedge Gf = h\}$$

THEOREM 1. *For any pair of functors $F, G : D \longrightarrow C$ such that $Fi = Gi = M$ and $Ff \approx Gf$ for all $f : iX \longrightarrow iY$, it follows that $R \cdot i$ is diagonal.*

The proof of this result is very easy, following directly from the definitions of B and R, but we call it a theorem because it underlies the succession of completeness results we seek. Observe that we have not assumed that F and G need preserve any structure, and the result uses nothing of the specific nature of the definition of $\approx$, only using the fact that $\approx$ is defined as an equivalence relation on $C(MX, MY)$ given X and Y in L, hence its generality.

6. COMPLETENESS OF LOGICAL RELATIONS FOR SIMPLY TYPED λ-CALCULUS

In this section, we prove that given a pair $F, G : D \longrightarrow C$ of equivalent interpretations, B and R as defined in the last section yield our completeness proof.

By the Yoneda lemma we have

PROPOSITION 5. *For any $H : D \longrightarrow C$, if C has finite products, then the functor $\tilde{H} : C \longrightarrow [D^{op}, Set]$ preserves finite products.*

PROPOSITION 6. *Given categories E and E' with finite products, a finite product preserving functor $K : E' \longrightarrow E$, and a finite product preserving fibration $p : E'' \longrightarrow E$, then the pullback $K^*(E'')$ of p along K gives a finite product preserving fibration $K^*(E'')$ over E'.*

A proof of this result appears in Hermida's thesis [3], but it can also be verified by direct calculation. It is easy to prove by direct calculation, using pullbacks in the presheaf category, that we have

THEOREM 2. *Given functors $F, G : D \longrightarrow C$, supposing C is cartesian closed, then $B = (\tilde{F} \times \tilde{G})^*(Sub_2[D^{op}, Set])$ is a cartesian closed fibration over $C \times C$.*

PROOF. The only point here that requires checking is that B is cartesian closed and that the functor to $C \times C$ preserves the cartesian closed structure. This follows from a careful analysis of what the objects of B are. Given an object (X, X') of $C \times C$, an object of B over (X, X') is a subfunctor of $C(F-, X) \times C(G-, X')$. Using cartesian closedness of $Sub_2([D^{op}, Set])$ over $[D^{op}, Set] \times [D^{op}, Set]$, given subfunctors of $C(F-, X) \times C(G-, X')$ and $C(F-, Y) \times C(G-, Y')$, one can construct a subfunctor of the exponential

$$[C(F-, X) \times C(G-, X'), C(F-, Y) \times C(G-, Y')]$$

Finally, taking the pullback of that subfunctor along the canonical comparison map from

$$C(F-, [X, Y]) \times C(F-, [X', Y'])$$

to

$$[C(F-, X) \times C(G-, X'), C(F-, Y) \times C(G-, Y')]$$

gives the exponential in B that we seek. $\Box$

This is a specific example of a general construction [3]. Now we have the data we require in order to state and prove our main result.

THEOREM 3. *If F and G are equivalent interpretations, they are linked by a logical relation.*

PROOF. We need to prove $R : D \longrightarrow B$ over (F, G) is cartesian closed. It is routine to verify that R preserves finite products; it takes considerably greater but still routine effort to verify that it preserves cartesian closed structure. $\Box$

Combining this completeness result with the soundness result of Section 4, we have a combined soundness and completeness result as follows.

THEOREM 4. *Two interpretations F and G are equivalent if and only if they are linked by a logical relation.*

7. COMPLETENESS OF LOGICAL RELATIONS FOR THE LINEAR λ-CALCULUS

For the linear λ-calculus, the finite products of L, D and C are replaced by a symmetric monoidal structure, and cartesian closedness is replaced by closedness with respect to that symmetric monoidal structure. In principle, that has the potential to cause considerable difficulty for us as there is no reason to expect $\tilde{F}$ to preserve monoidal structure with respect to any standard monoidal structure on $[D^{op}, Set]$, and a fortiori, the latter category is not closed with respect to such a monoidal structure, yet we used such facts to obtain our result for the simply typed λ-calculus, where we considered cartesian closed structure.

It is routine to modify the definition of contextual equivalence $\approx$ to the linear λ-calculus. Note that our analysis of diagonal functors has nothing to do with the specific nature of $\approx$, merely that it is an equivalence relation on $C(MX, MY)$ given (X, Y). So our constructions of B and R remain the same, yielding the diagonal condition, and our only concern here is whether B has and both $R : D \longrightarrow B$ and the forgetful functor from B to C preserve symmetric monoidal closed structure. This all works similarly for binary relations, i.e., having F and G, as for unary or more generally n-ary relations. So for simplicity of exposition, we shall express our propositions here in terms of the unary case, i.e., where we just have an $F : D \longrightarrow C$, with the observation that the results extend without fuss.

Proposition 7. *Given symmetric monoidal categories C and D and a strong symmetric monoidal functor $F : D \longrightarrow C$, we can define a symmetric monoidal structure on B such that $R : D \longrightarrow B$ and the forgetful functor to C both respect it:*
given subfunctors S and S' of $C(F-, X)$ and $C(F-, X')$, define $S \otimes S'$ to be the subfunctor of $C(F-, X \otimes X')$ with d-component given by the set of all maps $f : Fd \longrightarrow X \otimes X'$ such that there exist objects e and e' of D and maps $g : d \longrightarrow e \otimes e'$ in D and $h : Fe \longrightarrow X$ and $h' : Fe' \longrightarrow X'$ in Se and $S'e'$ respectively, such that the diagram

$$\begin{tikzcd} Fd \arrow[r, "Fg"] \arrow[d, "f"'] & F(e \otimes e') \arrow[d, "\cong"'] \\ X \otimes X' & Fe \otimes Fe' \arrow[l, "h \otimes h'"] \end{tikzcd}$$

commutes. The unit is given by the subfunctor $D(-, I_D)$ of $C(F-, I_C)$.

Note that the binary case we shall actually use has subfunctors S and S' of $C(F-, X) \times C(G-, X)$ and $C(F-, X') \times C(G-, X')$

The proof of (either version of) this proposition is routine. Somewhat more care, but again routine care, is required to prove the following result.

Proposition 8. *Suppose $F : D \longrightarrow C$ is a strong symmetric monoidal closed functor. Then the following construction exhibits B as symmetric monoidal closed with the forgetful functor to C and $R : D \longrightarrow B$ preserving the symmetric monoidal closed structure. Given subfunctors S' and S'' of $C(F-, X')$ and $C(F-, X'')$, define $[S', S'']$ to be the subfunctor of $C(F-, [X', X''])$ with d-component given by the set of maps $h : Fd \longrightarrow [X', X'']$ such that for every object e of D, every map in D of the form $g : e \longrightarrow d \otimes d'$, and every map of the form $h' : Fd' \longrightarrow X'$ in $S'd$, it follows that the composite*

$$\begin{tikzcd} F(d \otimes d') \arrow[r, "\cong"] & Fd \otimes Fd' \arrow[r, "Fd \otimes h'"] & Fd \otimes X' \arrow[d, "\bar{h}"] \\ Fe \arrow[u, "Fg"] & & X'' \end{tikzcd}$$

lies in $S''e$, where $\bar{h}$ is given by unCurrying h.

¿From these propositions, we can immediately prove

Theorem 5. *Two interpretations F and G of the linear λ-calculus with an abstract data-type are equivalent if and only if they are linked by a logical relation.*

Proof. The constructions of B and R are given by Theorem 1. We need only prove that if C and D are symmetric monoidal closed categories and if $F : D \longrightarrow C$ is a strong symmetric monoidal closed functor, then B is symmetric monoidal closed, with both R and the forgetful functor from B to C strong symmetric monoidal. But that follows immediately from the two propositions. $\square$

8. COMPLETENESS OF LOGICAL RELATIONS FOR EXTENSIONS OF THE SIMPLY TYPED λ-CALCULUS AND THE LINEAR λ-CALCULUS

In this section, we consider extensions of the completeness results of the previous two sections to additional type constructors, such as sum types and monads, e.g., for modelling partiality or nondeterminism. Both cases form a pattern, agreeing with that of the previous section; and we believe it is clear how to extend this pattern to other simple type constructors. However, we do not yet have a general, natural category theoretic formulation of the pattern.

As in the previous section, we shall express the propositions of this section in unary terms for convenience of exposition. The results extend routinely to binary and more generally to n-ary logical relations.

For sum types, we need to modify our definition of B. Let an object of B' consist of an object X of C together with a subobject S of $C(F-, X)$ that is closed under finite coproducts in D, i.e., if $f : Fd \longrightarrow X$ and $f' : Fd' \longrightarrow X$ are both in S, then so is the universal arrow $F(d + d') \longrightarrow X$

determined by binary coproduct. And similarly the unique isomorphism from $F0$ to 0 must lie in S.

Proposition 9. *The category B' is a fibration over C, and the functor $R : D \longrightarrow B$ factors through B and is diagonal.*

The proof is mundane.

Proposition 10. *Given categories C and D with finite coproducts and given a finite coproduct preserving functor $F : D \longrightarrow C$, the category B' has finite coproducts and they are preserved by the forgetful functor to C. Moreover, $R : D \longrightarrow B'$ preserves finite coproducts.*

Proof. The following construction yields finite corpoducts on B. Given subfunctors S and S' of $C(F-, X)$ and $C(F-, X')$, let $S + S'$ be the subfunctor of $C(F-, X + X')$ with d-component given by the set of all maps $f : Fd \longrightarrow X + X'$ such that there exist objects e and e' of D and maps $g : d \longrightarrow e + e'$ in D and $h : Fe \longrightarrow X$ and $h' : Fe' \longrightarrow X'$ in Se and $S'e'$ respectively, such that the diagram

$$\begin{tikzcd} Fd \arrow[r, "g"] \arrow[d, "f"'] & F(e+e') \arrow[d, "\cong"'] \\ X + X' & Fe + Fe' \arrow[l, "h+h'"] \end{tikzcd}$$

commutes. Routine verification shows that this does give binary coproducts in B', and by the construction, they are preserved by the forgetful functor to C. An initial object of B' is similar: note that since the unique map from $F0$ to 0 must lie in 0, this is not the empty subfunctor of $C(F-, 0)$.

For the final assertion, this is a routine calculation applied to the definition of R and using naturality. $\square$

Here, we have introduced a new construction B', whereas we proved completeness for simply typed λ-calculus with respect to B. But we have

Proposition 11. *If C has finite products, if D is distributive, and if $F : D \longrightarrow C$ preserves finite products, then B' has and $R : D \longrightarrow B'$ and the forgetful functor from B' to C preserves them.*

Proof. ¿From distributivity of binary product over coproducts in D, one can verify that B' has finite products given by the restriction of finite products in B to B'. $\square$

Proposition 12. *If C is cartesian closed, if D is cartesian closed with pullbacks and with finite coproducts stable under pullbacks, and if $F : D \longrightarrow C$ is cartesian closed and preserves pullbacks, then B' has and $R : D \longrightarrow B'$ and the forgetful functor from B' to C preserve cartesian closed structure.*

Proof. One applies the pullbacks and stability of coproduct under pullback to the construction of exponentials in B to see that if S' and S'' are in B', then so is the exponential of them as taken in B. $\square$

The condition we require on D here is not ideal, as our leading examples have D being syntactically generated, and syntactically generated pullbacks do not seem to have a natural computational meaning.

Theorem 6. *Under the conditions of Proposition 12, two interpretations F and G of the simply typed λ-calculus with sum types and an abstract data-type are equivalent if and only if they are linked by a logical relation.*

Proof. The category B' is a mild variant of the category B. We have been careful in constructing it to preserve the statement and proof of Theorem 1. So the adaption of that, together with the above propositions, immediately yield the result. $\square$

One can, of course, modify the above analysis from the simply typed λ-calculus to the linear λ-calculus.

Finally, we consider the addition of a strong monad T for partiality or nondeterminism for example. This situation is easier than that for sum types as we do not need to modify our construction B.

Proposition 13. *Suppose C and D are equipped with strong monads, with $F : D \longrightarrow C$ strictly respecting them. Then B is also equipped with a strong monad, such that R and the forgetful functor to C respect strong monads.*

Proof. The monad on B is defined by saying $f : Fd \longrightarrow TX$ is in TS if there exist $g : d \longrightarrow Te$ in D and $h : Fe \longrightarrow X$ in C such that the evident diagram commutes. The definitions of the unit and multiplication of the monad are routine, as is the strength, and the laws for a strong monad hold because they hold in C. $\square$

Theorem 7. *Two interpretations of the simply typed λ-calculus together with a strong monad and an abstract data type are equivalent if and only if they are linked by a logical relation.*

Proof. This is an immediate consequence of Theorem 1 and the binary version of the above proposition. $\square$

9. REFERENCES

[1] A. Alimohamed. A characterisation of lambda definability in categorical models of implicit polymorphism. *Theoretical Computer Science*, 146:5–23, 1995.

[2] M. Fiore and G. Plotkin. An axiomatisation of computationally adequate domain-theoretic models of *fpc*. In *Proc LICS 94*, pages 92–102, 1994.

[3] C. Hermida. *Fibrations, Logical Predicates and Indeterminates.* PhD thesis, University of Edinburgh, 1993.

[4] F. Honsell and D. Sannella. Pre-logical relations. In *Proc CSL 99, LNCS 1683*, pages 546–562, 1999.

[5] A. Jung and J. Tiuryn. A new characterisation of lambda definability. In *Proc. TLCA 93, LNCS 664*, pages 245–257, 1993.

[6] Y. Kinoshita, P. O'Hearn, A. Power, M. Takeyama, and R. Tennent. An axiomatic approach to binary logical relations with applications to data refinement. In *Proc TACS 97, LNCS 1281*, pages 191–212, 1997.

[7] F. Lawvere. Equality in hyperdoctrines and the comprehension schema as an adjoint functor. In *Proc New York Symposium on Applications of Categorical Algebra*, pages 1–14, 1970.

[8] Q. Ma and J. Reynolds. Types, abstraction and parametric polymorphism 2. In *Proc MFPS 91, LNCS 598*, pages 1–40, 1991.

[9] R. Milner. Fully abstract models of typed lambda-calculus. *Theoretical Computer Science*, 4:1–20, 1977.

[10] J. Mitchell. Type systems for programming languages. In *Handbook of Theoretical Computer Science B*, pages 365–458. North-Holland, 1990.

[11] J. Mitchell. *Foundations for programming languages. Foundations of Computing Series.* MIT Press, 1996.

[12] E. Moggi. Computational lambda-calculus and monads. In *Proc LICS 89*, pages 14–23, 1989.

[13] G. Plotkin, A. Power, D. Sannella, and R. Tennent. Lax logical relations. In *Proc ICALP 2000*, to appear.

[14] A. Power. Premonoidal categories as categories with algebraic structure. *Theoretical Computer Science*, to appear.

[15] A. Power and E. Robinson. Premonoidal categories and notions of computation. *Math. Structures in Computer Science*, 7:453–468, 1997.

[16] A. Power and E. Robinson. Logical relations and data abstraction. In *Proc CSL 2000*, to appear.

[17] J. Reynolds. Types, abstraction, and parametric polymorphism. *nformation Processing*, 83:513–523, 1983.

[18] R. Tennent. Correctness of data representations in algol-like languages. *A Classical Mind, Essays in Honour of C.A.R. Hoare*, pages 405–417, 1994.