ISSN 1369-1961

Department of
Computer Science

Technical Report No. 730

Logical Relations and Data Abstraction

E. P. Robinson

QUEEN MARY
AND WESTFIELD COLLEGE
UNIVERSITY OF LONDON

August 1996

Logical Relations and Data Abstraction

E.P. Robinson
Dept of Computer Science,
Queen Mary and Westfield College,
Mile End Road,
London, E1 4NS, UK

August 19, 1996

Abstract

The aim of this paper is to prove in the context of simple type theory that logical relations are sound and complete for a certain form of data abstraction: that given by equational specifications. Specifically we show that two implementations of an equationally specified abstract type are equivalent if and only if they are linked by a suitable logical relation. The form of equational specification we use allows us to introduce new types and operations of any order on these types, and to impose equations between terms, again of any order. Implementations are required to respect these equations up to a fairly general form of contextual equivalence, and two implementations are regarded as being equivalent if they produce the same contextual equivalence on terms of the enlarged language (in fact we show that there are several different ways of formulating this notion). Logical relations are introduced abstractly, following work of Reynolds, and soundness is almost automatic. Completeness requires more work, and is achieved using a variant of Jung and Tiuryn's Kripke logical relations of varying arity. Our results are expressed and proved categorically.

Introduction

Part of John Reynolds' intuition behind his use of logical relations to formalise parametric polymorphism in [8], was his idea that a polymorphic type variable was in some sense an abstract type. More precisely, a parametrically polymorphic operation must respect data abstraction, in the sense that if we use equivalent representations of some abstract type, parametrically polymorphic operations should not be able to tell the difference. This intuition can easily be turned in on itself to give a definition of parametric polymorphism (a polymorphic operation is parametric if it can't tell the difference ...). Two representations of an abstract type are equivalent if, given that the only types which are observable are concrete, one cannot observe any difference between the two implementations. Reynolds formalised this by requiring that they be linked by a logical relation.

Now, if one can find a family of relations on corresponding types which is a congruence (i.e. are preserved by operations of the theory), and reduce to equality on observable types, then clearly observable operations will be equal in the two representations. Such a family does not necessarily form a logical relation, but logical relations do provide a way of constructing such families by induction on type structure. So the establishment of a logical relation suffices to determine equivalence of representation, at least for simple types. Reynolds' account of parametric polymorphism is aimed at making this hold by definition for parametric polymorphism (and fail for non-parametric!). But a natural question remains: is any pair of equivalent representations necessarily linked by a logical relation? The purpose of this paper is to show that in a certain sense the answer to that is yes, at least for simple types. This was previously known for algebraic theories with first order operations and equations, (and is due to Mitchell, see [7]) but the results given

here apply to fully higher-order theories, largely because of the formalisation in terms of cartesian closed categories rather than Henkin models.

The approach in this paper has two major sources. The first is the work of Achim Jung and Jerzy Tiuryn [4] using a form of logical relations to characterise lambda definability in the simple type hierarchy. Formally speaking the completeness proof sketched out here is a mild variant of theirs. The work presented here, however, is not based on Henkin models, but on cartesian closed categories, and therefore the relevant comparison is the Alimohamed's characterization of lambda definability for ccc's in [2] (which, however, also deals with ML-style polymorphism).

The second major source is the work of Claudio Hermida in his thesis [3], where he explores the connection between logical predicates and hyperdoctrines defined over the semantic category. This provides a formal link between the theory of logical relations and the categorical formalisation of logic, and hence with program logics. That link is still being explored.

Hermida's approach shows an apparent difference between conventional (Kripke) logical relations and those used by Jung and Tiuryn. Kripke logical relations are derived from a standard logic for the types of the semantic category: "standard" meaning reflecting a view of the types as sets in an intuitionistic sense, and the type-formation operators as their standard set-theoretic counterparts. The Jung-Tiuryn relations seem to be derived from a non-standard logic. There is a question of whether this can be avoided, perhaps by considering a family of standard logical relations rather than a single non-standard one.

The notion of abstract type used here is fairly restricted from a practical point of view. It is restricted to equationally specified types, albeit with equations between higher-order operations. This is also the level of generality discussed in the work of Alimohamed, cited above. 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.

Finally, if data abstraction is the study of equivalence of representation, then data refinement is the study of improvements. The two ideas are clearly linked, and this paper has been greatly influenced by ongoing work with Yoshiki Kinoshita, Peter O'Hearn, John Power, Makoto Takeyama, and Bob Tennent on the semantics of data refinement [5]. The formal setting given in this paper is an instance of their general framework.

I have made extensive use of categorical language and machinery. This is because of its power as a tool for abstraction, precision, and organisation. I hope the paper is intelligible despite that. The use of this machinery does, however entail one choice. One has to decide whether morphisms in the category represent open or closed terms. I have chosen to use open terms, mainly because it seems to me that this opens the door to more applications, dependent types in one direction, and first-order in the other.

The plan of the paper is as follows. First we give a categorical account of the theory of contextual equivalence. Then we move on to abstract types, implementations, and equivalence of implementation. In section 3 we introduce the essential aspects of logical relations characterise the diagonal relation, and prove soundness. In section 4 we re-examine Jung and Tiuryn's relations in order to motivate our own, and finally in section 5 we sketch the proof of our completeness theorem.

Quite a number of people have made valuable suggestions to me concerning the material in this paper, and I am very grateful to them. They include Adam Eppendahl, Claudio Hermida, Peter O'Hearn, David Pym, Makoto Takeyama, and Bob Tennent. This research was partially supported by an EPSRC Advanced Research Fellowship.

1 Contextual equivalence

In this section we give a categorical account of a standard Morris-style contextual equivalence. Since 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 assume a given cartesian closed category C. This will represent our base category. We further assume that we are given a set of observable pairs of types: $\mathcal{O} \subseteq \mathrm{ob}(\mathsf{C}) \times \mathrm{ob}(\mathsf{C})$. The intuition behind this is that we are allowed to form judgements of equality

$$\vec{x} : A \vdash f = g : B$$

where $(A, B) \in \mathcal{O}$. An alternative approach is to use a set of observable types, but we believe that ours accords better with the use of an open term interpretation of the lambda calculus, and will allow a simpler extension to dependent type theory. We use this structure of observables to generate a contextual equivalence $\approx$ on C, which allows us to compare $f \approx g$ where f and g are maps with a common domain and codomain (type-theoretically, this amounts to demanding that contextually equivalent terms have the same type and are defined in the same context).

Definition 1.1 *Suppose $f, g : X \to Y$, then $f \sim g$ iff for all $(A, B) \in \mathcal{O}$, and all $\alpha : A \to X$ and $\beta : Y \to B$,*

$$A \xrightarrow{\alpha} X \xrightarrow{f} Y \xrightarrow{\beta} B = A \xrightarrow{\alpha} X \xrightarrow{g} Y \xrightarrow{\beta} B$$

Definition 1.2 *$\approx$ is the largest congruence contained in $\sim$.*

We can be more concrete about the definition of $\approx$.

Lemma 1.3 *For $f, g : X \to Y$ the following are equivalent:*

(i). $f \approx g$

(ii). *for all $U, V \in \mathrm{ob}(\mathsf{C})$, $f^U \times V \sim g^U \times V$*

(iii). *for all $U, V \in \mathrm{ob}(\mathsf{C})$, for all $(A, B) \in \mathcal{O}$, and all $\alpha : A \to X^U \times V$ and $\beta : Y^U \times V \to B$, $\alpha; f^U \times V; \beta = \alpha; g^U \times V; \beta$*

This follows immediately from the following considerations:

Lemma 1.4 *Suppose $\sim$ is an equivalence relation on parallel pairs of arrows in a category C, then*

(i). *Composition factors through quotient by the relation $\sim$, and hence $\mathsf{C}/\!\sim$ is a category, and $\mathsf{C} \to \mathsf{C}/\!\sim$ a functor, if and only if $f \sim g$ implies $\alpha; f; \beta \sim \alpha; g; \beta$.*

(ii). *If C has finite products, then $\mathsf{C}/\!\sim$ has finite products, and passage to the quotient preserves them if and only if (i) and $f \sim g$ implies $f \times V \sim g \times V$, for all $V \in \mathrm{ob}(\mathsf{C})$.*

(iii). *If C is cartesian closed then $\mathsf{C}/\!\sim$ is a cartesian closed category, and passage to the quotient a cartesian closed functor if and only if (i) and (ii) and $f \sim g$ implies $f^U \sim g^U$ for all $u \in \mathrm{ob}(\mathsf{C})$, or equivalently if and only if (i) and $f \sim g$ implies $f^U \times V \sim g^U \times V$, for all $U, V \in \mathrm{ob}(\mathsf{C})$.*

We can also make precise the relationship with Morris-style contextual equivalence. Any cartesian closed category can be regarded as a model of an applied lambda calculus in which we have base types corresponding to the objects of the category, and for each morphism $f : X \to Y$, we are given a constant $\hat{f}$ such that $\vec{x} : X \vdash \hat{f} : Y$. We can now define $e \cong e'$ for terms e and e' if and only if for all relevant contexts $C[\,]$, $[\![C[e]]\!] = [\![C[e']]\!]$, where the semantics is taken in C. A relevant context here is one such that there is a derived rule validating that if $\vec{x} : X \vdash e : Y$ then $\vec{a} : A \vdash C[e] : B$ $((A, B) \in \mathcal{O})$.

Lemma 1.5 *$e \cong e'$ if and only if $e \approx e'$.*

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. □

There is also a corresponding closed term version, in which we take $\hat{f}$ to be closed of type $A \to B$.

Examples 1.6 (i). $\mathcal{O} = \{(1, \mathsf{bool})\}$. 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.

(ii). $\mathcal{O} = \{(1, S)\}$, where S is Sierpinski space. Here 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 S generates a pre-order on terms. This is the pre-order used to quotient the intensionally full abstract model of PCF in Abramsky et al. [1]. So, if we have an observable type which contains values for both termination and nontermination, we can handle the conventional form of contextual equivalence.

(iii). $\mathcal{O} = \mathsf{ob}(\mathsf{C}) \times \mathsf{ob}(\mathsf{C})$. When everything is observable, $\approx$ is just equality.

2 Abstract types

An equationally specified abstract type consists of a new type equipped with operations linking it 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

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

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 inclusion functor $i\colon \mathsf{C} \to \mathsf{D}$. This functor is injective on objects, but in general is neither full nor faithful (we can add extra operations between pre-existing types, and impose new equations between pre-existing operations). Conversely, given any cartesian closed functor $i\colon \mathsf{C} \to \mathsf{D}$, D is equivalent to a category constructed from C in this fashion (we generate it by using all objects in $\mathsf{D} \setminus i(\mathsf{C})$, and greedily take all valid equations). So we can take $i\colon \mathsf{C} \to \mathsf{D}$ as the extension of C by some abstract types.

We can split this construction into two stages: first adjoin types and operations freely to give $i\colon \mathsf{C} \to \mathsf{D}$, and then quotient by the equations to give $q\colon \mathsf{D} \to \mathsf{D}/\approx_E$. In fact we shall never need the free property of D, and can take it to be an arbitrary cartesian closed category.

An implementation of the abstract type is given by a cartesian closed functor $F\colon \mathsf{D} \to \mathsf{C}$ such that $i; F = \mathrm{id}_{\mathsf{C}}$, (i.e. leaving fixed the operations in C), and validating the equations up to observational congruence in the sense we shall now explain.

Any functor $F\colon \mathsf{D} \to \mathsf{C}$ induces a notion of contextual equivalence, $\approx_F$, on D.

Definition 2.1 *$f \approx_F g : X \to Y$ if and only if for all $U, V \in \mathsf{ob}(\mathsf{D})$, for all $(A, B) \in \mathcal{O}$, and for all $\alpha : iA \to X^U \times V$ and $\beta : Y^U \times V \to iB$, $F(\alpha; f^U \times V; \beta) = F(\alpha; g^U \times V; \beta)$.*

The implementation is valid if and only if all equations $\vec{x} : \vec{X} \vdash e = e' : Y$ hold up to contextual equivalence, i.e. $[\![e]\!] \approx_F [\![e']\!]$, or equivalently the quotient $\mathsf{D} \to \mathsf{D}/\approx_F$ factors through $\mathsf{D} \to \mathsf{D}/\approx_E$. The only function of the equational theory is to restrict the notion of implementation. It has no effect on when two implementations are equivalent, and so now plays no further rôle in the story, it just, so to speak, comes along for the ride.

We note some properties of $\approx_F$.

Lemma 2.2 *Let $F: D \to C$ be a cartesian closed functor such that $i;F = id_C$, then the following hold of $\approx_F$:*

(1). For all $f,g: X \to Y$ in D, $Ff \approx Fg \Rightarrow f \approx_F g$.

(2). For all $f,g: X \to Y$ in C, $f \approx g \Rightarrow if \approx_F ig$.

(3). For all $f,g: iX \to iY$ in D, $f \approx_F g$ if and only if $Ff \approx Fg$.

(4). For all $f,g: iA \to iB$ in D, where $(A,B) \in \mathcal{O}$, $f \approx_F g$ if and only if $Ff = Fg$.

Proof. (1). Suppose $Ff \approx Fg$. Then, given $iA \xrightarrow{\alpha} X^U \times V$ and $Y^U \times V \xrightarrow{\beta} iB$, $F(\alpha; f^U \times V; \beta) = F\alpha; Ff^{FU} \times FV; F\beta = F\alpha; Fg^{FU} \times FV; F\beta = F(\alpha; g^U \times V; \beta)$.
(2). Suppose $f \approx g$. Then, given $iA \xrightarrow{\alpha} (iX)^U \times V$ and $(iY)^U \times V \xrightarrow{\beta} iB$, $F(\alpha; (if)^U \times V; \beta) = F\alpha; (Fif)^{FU} \times FV; F\beta = F\alpha; f^{FU} \times FV; F\beta = F\alpha; g^{FU} \times FV; F\beta = F\alpha; (Fig)^{FU} \times FV; F\beta = F(\alpha; (ig)^U \times V; \beta)$.
(3). Suppose $f \approx_F g$. Then, given $A \xrightarrow{\alpha} X^U \times V = (FiX)^U \times V$ and $(FiY)^U \times V = Y^U \times V \xrightarrow{\beta} B$, $\alpha; (Ff)^U \times V; \beta = Fi(\alpha; (Ff)^U \times V; \beta) = F(i\alpha; f^{iU} \times iV; i\beta) = F(i\alpha; g^{iU} \times iV; i\beta) = \alpha; (Fg)^U \times V; \beta$.
(4). $Ff \approx Fg$ if and only if $Ff = Fg$ for observable types. □

It is definitely not always the case that for all $f,g: X \to Y$ in D, $f \approx_F g$ if and only if $Ff \approx Fg$. In particular, we do not necessarily have a functor $D/\approx_F \to C/\approx$. Indeed, consider an array implementation of stacks in which the effect of pop is to move the top pointer, but not delete the old top element. Then the implementation of `push;pop` is not contextually equivalent to the identity. I am indebted to Peter O'Hearn for this example.

Suppose now that F and G are two interpretations:

Lemma 2.3 *The following are equivalent:*

(1). F and G induce the same equivalence: $\approx_F = \approx_G$.

(2). F and G induce the same equivalence on observable operations: $\approx_F |_{i\mathcal{O}} = \approx_G |_{i\mathcal{O}}$.

(3). The interpretations of operations between pre-existing types are contextually equivalent: for all $f: iX \to iY$, $Ff \approx Gf$.

(4). The interpretations of observable operations are contextually equivalent (in fact, equal): for all $f: iA \to iB$, $Ff \approx Gf$, $((A,B) \in \mathcal{O})$.

Proof. (1) ⇒ (2) and (3) ⇒ (4) are immediate.
(2) ⇒ (1): Suppose $f \approx_F g: X \to Y$, and $iA \xrightarrow{\alpha} X^U \times V$, $Y^U \times V \xrightarrow{\beta} iB$. Then $F(\alpha; f^U \times V; \beta) = F(\alpha; g^U \times V; \beta)$. But these are between observable types, hence by lemma 2.2(4), $\alpha; f^U \times V; \beta \approx_F \alpha; g^U \times V; \beta$. So $\alpha; f^U \times V; \beta \approx_G \alpha; g^U \times V; \beta$, and hence by 2.2(4) again $G(\alpha; f^U \times V; \beta) = G(\alpha; g^U \times V; \beta)$.
(1) ⇒ (3): Given $f: iX \to iY$, $Ff = FiFf$, hence by 2.2(1), $f \approx_F iFf$. Therefore $f \approx_G iFf$, and hence by 2.2(3), $Gf \approx GiFf = Ff$.
(4) ⇒ (2): Given $f,g: iA \to iB$, where $(A,B) \in \mathcal{O}$, suppose $f \approx_F g$. Then $Ff = Fg$. But by hypothesis, $Ff = Gf$, and $Fg = Gg$, so $Gf = Gg$. Hence $f \approx_G g$. □

Any of these would be a reasonable definition of equivalence of implementation, which helps justify the following.

Definition 2.4 *F and G are* equivalent implementations *if they satisfy the conditions of lemma 2.3.*

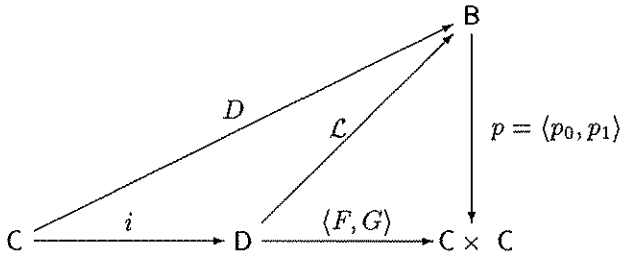
3 Logical relations

In [6], Ma and Reynolds begin by analysing the theory of logical relations for simple types. They focus on the structure needed to prove the fundamental theorem. For binary relations this amounts to a cartesian closed category B equipped with a cartesian closed functor $p = \langle p_0, p_1 \rangle : \mathsf{B} \to \mathsf{C} \times \mathsf{C}$. In their theory this is constructed concretely as a category of relations, and so comes with a standard diagonal $\Delta : \mathsf{C} \to \mathsf{B}$. This is also a cartesian closed functor. The fact that B has, and all the functors preserve cartesian closed structure is required so that logical relations can be constructed inductively on the syntax of type expressions, as usual. Technically, it is not needed for soundness. We take this structure as a characterisation of logical relations, but since our categories will be more abstract than those of Reynolds and Ma we require an abstract characterisation of the diagonal. Moreover we would like the diagonal to characterise contextual equivalence, not equality.

Definition 3.1 *Suppose $p = \langle p_0, p_1 \rangle : B \to C \times C$ is a cartesian closed functor, and $D : C \to B$ is a cartesian closed functor such that $D; p = \Delta : C \to C \times C$. Then we say D is* diagonal *if whenever $f : DX \to DY$, then $p_0 f \approx p_1 f$.*

We think of two maps $g, h : X \to Y$ as being in relation if and only if there is a map $f : DX \to DY$ such that $p_0 f = g$ and $p_1 f = h$. Then this definition says that D is diagonal when two maps are in relation if and only if they are contextually equivalent. Not surprisingly, if we replace contextual equivalence by equality, then this property holds of a true diagonal in a concrete category of logical relations (one where the relations are given as subobjects of a binary product).

Definition 3.2 *Two interpretations F and G are* linked by a logical relation *if and only if we can find a cartesian closed category B, and cartesian closed $p : B \to C \times C$, together with a cartesian closed $\mathcal{L} : D \to B$, lying over $\langle F, G \rangle$, such that $D = i; \mathcal{L}$ is diagonal.*



Soundness is almost immediate from the definitions.

Lemma 3.3 *If two interpretations F and G are linked by a logical relation, then they are equivalent.*

Proof. Suppose $f : iA \to iB$ is a map in D, then $\mathcal{L}f$ is a map from $\mathcal{L}iA = DA$ to $\mathcal{L}iB = DB$ in B. Hence $Ff = p_0(\mathcal{L}f)$, and $Gf = p_1(\mathcal{L}f)$ are contextually equivalent, fulfilling our third criterion for equivalence of implementations, 2.3(3). □

Completeness is more difficult, and involves a variant of Jung and Tiuryn's Kripke logical relations of varying arity.

4 Jung-Tiuryn revisited

In this section we re-examine Jung and Tiuryn's construction of Kripke logical relations "of varying arity" [4].

One way to obtain logical relations is from suitable subobjects. But another is from a logic interpreted as a hyperdoctrine. If the logic admits $\top$, $\wedge$, $\rightarrow$ and $\forall$, then it yields a category of logical relations. 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 exactly furnish a proof. These two approaches meet in the standard logic of predicates interpreted as subobjects.

In standard Kripke logical relations for the simple type hierarchy we are given an indexing category X. The types of the hierarchy are interpreted as constant functors in $\mathrm{Set}^{\mathsf{X}^{\mathrm{op}}}$ $((\Delta d)(x) = d)$, and Kripke logical relations are derived from the lattice of subfunctors of these functors. Note that for connected X, $\Delta : \mathrm{Set} \rightarrow \mathrm{Set}^{\mathsf{X}^{\mathrm{op}}}$ is a cartesian closed functor, and $\mathrm{Set}^{\mathsf{X}^{\mathrm{op}}}$ itself is a model of a form of intuitionistic set theory. Hence we can regard this as the simple type hierarchy on a non-standard set in a non-standard set theory, and have simply used the standard definition of logical relations in this non-standard setting.

In Jung and Tiuryn's work, however, there is an extra twist. X is assumed to be concrete, i.e. consisting of sets and functions. In more categorical terms this amounts to an abstract category equipped with a functor $F : \mathsf{X} \rightarrow \mathrm{Set}$ (technically a faithful functor, but that fact is never used). Jung and Tiuryn's logical relations are derived from subfunctors of $\Delta d ** F$, where $\Delta d ** F x = d^{Fx}$ (the Fx giving the "variable arity"). So one way of looking at things is that we are using $\Delta d ** F$ instead of Δd. Unfortunately, $(\ ) ** F$ does not preserve exponentials in general, so we can no longer interpret the construction as logical relations on a version of the simple type hierarchy in $\mathrm{Set}^{\mathsf{X}^{\mathrm{op}}}$. But at least $(\ ) ** F$ is cartesian, which means that we can instead think of it as a a way of relabelling predicates. A non-standard predicate on d is a Kripke predicate on $\Delta d ** F$, i.e. a subfunctor. Since $\Delta(\ ) ** F$ is cartesian, the resulting logic admits $\top$, $\wedge$, $\rightarrow$, and $\forall$, and hence yields a category of logical relations. This is clearer in terms of the categorical formalism: we are obtaining one hyperdoctrine by change of base along a cartesian functor from another. Since the original admits $\top$, $\wedge$, $\rightarrow$, and $\forall$, so does the result.

So Jung and Tiuryn's relations are in fact unary predicates, despite appearances to the contrary. We shall need a binary version. Moreover, since we are not given a set-based model to begin with, we must first embed our category in a topos. Abstractly, the recipe given in the next section is to embed C in $\mathrm{Set}^{\mathsf{C}^{\mathrm{op}}}$ via the Yoneda embedding. Embed that in $(\mathrm{Set}^{\mathsf{C}^{\mathrm{op}}})^{\mathsf{E}^{\mathrm{op}}} = \mathrm{Set}^{(\mathsf{E} \times \mathsf{C})^{\mathrm{op}}}$ via constant presheaves, and then interpret a non-standard relation on A as a subfunctor of

$$(\Delta \mathcal{Y} A ** \mathcal{F}) \times (\Delta \mathcal{Y} A ** \mathcal{G})$$

where $\mathcal{F}$ and $\mathcal{G}$ are functors derived from the interpretations F and G. This is not a major modification of Jung and Tiuryn's formalism, and the construction we use for completeness also owes much to the one they use for definability.

5 Completeness of logical relations

We construct our cartesian closed category B as follows.

Let E be the category of sequences of objects of D ordered by co-extension, i.e. $U_1 \ldots U_{n+m} \leq U_1 \ldots U_n$.

Given $B \in \mathrm{ob}(\mathsf{C})$ define

$$(\Delta \mathcal{Y} B)(U_1 \ldots U_n, A) = \mathsf{C}(A, B)$$

With the obvious actions on morphisms, $\Delta \mathcal{Y} B$ is a functor $(\mathsf{E} \times \mathsf{C})^{\mathrm{op}} \rightarrow \mathrm{Set}$, and $\Delta \mathcal{Y}$ a functor $\mathsf{C} \rightarrow \mathrm{Set}^{(\mathsf{E} \times \mathsf{C})^{\mathrm{op}}}$.

Given $F : \mathsf{D} \rightarrow \mathsf{C}$, we first define $\mathcal{F} : \mathsf{E} \rightarrow \mathrm{Set}^{\mathsf{C}^{\mathrm{op}}}$ by

$$\mathcal{F}(U_1 \ldots U_n) = \mathcal{Y}(FU_1 \times \ldots FU_n)$$

where $\mathcal{Y}$ is the Yoneda embedding. We then define $(-) ** \mathcal{F} : \mathrm{Set}^{(\mathsf{E}\times\mathsf{C})^{\mathrm{op}}} \to \mathrm{Set}^{(\mathsf{E}\times\mathsf{C})^{\mathrm{op}}}$ by

$$\begin{aligned}
&\theta ** \mathcal{F}(U_1 \ldots U_n, A) \\
&= (\theta(U_1 \ldots U_n, -)^{\mathcal{F}(U_1 \ldots U_n)})(A) \\
&= \mathrm{Set}^{\mathsf{C}^{\mathrm{op}}}(\mathcal{F}(U_1 \ldots U_n) \times \mathsf{C}(-, A), \theta(U_1 \ldots U_n, -)) \\
&= \mathrm{Set}^{\mathsf{C}^{\mathrm{op}}}(\mathcal{Y}(FU_1 \times \ldots \times FU_n \times A), \theta(U_1 \ldots U_n, -))
\end{aligned}$$

If $\theta = \Delta\mathcal{Y}B$ then

$$\begin{aligned}
&\Delta\mathcal{Y}B ** \mathcal{F}(U_1 \ldots U_n, A) \\
&= \mathrm{Set}^{\mathsf{C}^{\mathrm{op}}}(\mathcal{Y}(FU_1 \times \ldots \times FU_n \times A), \mathcal{Y}B) \\
&= \mathsf{C}(FU_1 \times \ldots \times FU_n \times A, B)
\end{aligned}$$

$(-) ** \mathcal{G}$ is defined similarly, and $\Delta\mathcal{Y}B ** \mathcal{G}(U_1 \ldots U_n, A) = \mathsf{C}(GU_1 \times \ldots \times GU_n \times A, B)$.
The objects of our cartesian closed category, B, are subfunctors

$$i : P \rightarrowtail (\Delta\mathcal{Y}B_0 ** \mathcal{F}) \times (\Delta\mathcal{Y}B_1 ** \mathcal{G})$$

Specifically, $P(U_1 \ldots U_n, A)$ is a subset of

$$\{\ (f, g) \mid f : FU_1 \times \ldots \times FU_n \times A \to B_0,\ g : GU_1 \times \ldots \times GU_n \times A \to B_1\}$$

such that if $\pi^{n+m}_{\leq n}$ is the projection

$$X_1 \times \ldots \times X_{n+m} \to X_1 \times \ldots \times X_n$$

and

$$h : A' \to A,$$

then

$$(f', g') \in P(U_1 \ldots U_{n+m}, A')$$

where

$$f' = FU_1 \times \ldots \times FU_{n+m} \times A' \xrightarrow{\pi^{n+m}_{\leq n} \times h} FU_1 \times \ldots \times FU_n \times A \xrightarrow{f} B_0$$

and

$$g' = GU_1 \times \ldots \times GU_{n+m} \times A' \xrightarrow{\pi^{n+m}_{\leq n} \times h} GU_1 \times \ldots \times GU_n \times A \xrightarrow{g} B_1$$

A morphism from $i : P \rightarrowtail (\Delta\mathcal{Y}B_0 ** \mathcal{F}) \times (\Delta\mathcal{Y}B_1 ** \mathcal{G})$ to $i' : P' \rightarrowtail (\Delta\mathcal{Y}B'_0 ** \mathcal{F}) \times (\Delta\mathcal{Y}B'_1 ** \mathcal{G})$ is a pair of maps (β_0, β_1) in C, $\beta_0 : B_0 \to B'_0$ and $\beta_1 : B_1 \to B'_1$, such that $i; (\Delta\mathcal{Y}\beta_0 ** \mathcal{F}) \times (\Delta\mathcal{Y}\beta_1 ** \mathcal{F})$ factors through i' (i.e. $(\Delta\mathcal{Y}\beta_0 ** \mathcal{F}) \times (\Delta\mathcal{Y}\beta_1 ** \mathcal{F})$ maps P into P').

It follows from the abstract category theory that B is a cartesian closed category, and that the obvious forgetful functors into C preserve that cartesian closed structure. We can also obtain an explicit description of the cartesian closed structure on B from the same source. This is used later in the proof of lemma 5.1. The essential part is the intepretation of the exponential. Suppose P is over (X_0, X_1) and Q is over (Y_0, Y_1), then $[P \to Q]$ is over $(Y_0^{X_0}, Y_1^{X_1})$, and

$$(f_0, f_1) \in [P \to Q](U_1 \ldots U_n, A)$$

if and only if for all

$$U_1 \ldots U_{n+m} \leq U_1 \ldots U_n,$$

all maps

$$h : A' \to A,$$

and all pairs

$$(\gamma_0, \gamma_1) \in P(U_1 \ldots U_{n+m}, A'),$$

the pair (β_0, β_1) is in $Q(U_1 \ldots U_{n+m}, A')$, where

$$\beta_i = \langle (\pi^{n+m}_{\leq n} \times h); f_i \ , \ \gamma_i \rangle; \mathrm{ev}$$

It now remains to define our logical relation, and to verify that it has the required properties.

$\mathcal{L}(Y)$ must be a subfunctor of $(\Delta \mathcal{Y} FY \ast\ast \mathcal{F}) \times (\Delta \mathcal{Y} GY \ast\ast \mathcal{F})$. We take $\mathcal{L}(Y)(U_1 \ldots U_n, A)$ to be the set of pairs

$$(h_F, h_G) : \ h_F : FU_1 \times \ldots \times FU_n \times A \to FY, \ h_G : GU_1 \times \ldots \times GU_n \times A \to GY,$$

such that there is a map $h : U_1 \times \ldots \times U_n \times iA \to Y$ in D such that $Fh = h_F$ and $Gh = h_G$. Clearly $\mathcal{L}$ is functorial.

Lemma 5.1 *$\mathcal{L} : D \to B$ is a morphism of cartesian closed categories.*

Proof. This is by an explicit calculation. It is almost identical to the proof of Theorem 5 in [4]. Preservation of products is immediate from the definitions. The significant part is the preservation of exponentials.

Suppose $(\alpha_F, \alpha_G) \in \mathcal{L}(Y^X)(U_1 \ldots U_n, A)$, then there is an α such that $F\alpha = \alpha_F$ and $G\alpha = \alpha_G$.

Now suppose

$$U_1 \ldots U_{n+m} \leq U_1 \ldots U_n,$$
$$h : A' \to A$$

and

$$(\gamma_F, \gamma_G) \in \mathcal{L}(X)(U_1 \ldots U_{n+m}, A')$$

Then there is a

$$\gamma : U_1 \times \ldots \times U_{n+m} \times iA' \to iX$$

such that $F\gamma = \gamma_F$ and $G\gamma = \gamma_G$.

Consider (β_F, β_G), where $\beta_j = \langle (\pi^{n+m}_{\leq n} \times h); \alpha_j \ , \ \gamma_j \rangle; \mathrm{ev}$. Then $(\beta_F, \beta_G) \in \mathcal{L}(X)$ as witnessed by

$$\beta_j = \langle (\pi^{n+m}_{\leq n} \times ih); i\alpha \ , \ i\gamma \rangle; \mathrm{ev}$$

So $\mathcal{L}(Y^X) \leq \mathcal{L}(Y)^{\mathcal{L}(X)}$.

For the converse, suppose that

$$(\alpha_F, \alpha_G) \in \mathcal{L}(Y)^{\mathcal{L}(X)}(U_1 \ldots U_n, A)$$

Consider $\langle \pi^{n+2}_{n+1}, \pi^{n+2}_{n+1} \rangle$, which is in $\mathcal{L}(X)(U_1 \ldots U_n(iX), A)$ as witnessed by

$$\pi^{n+2}_{n+1} : U_1 \times \ldots \times U_n \times iX \times iA \to iX$$

(α_F, α_G) applied to this must be in $\mathcal{L}(Y)$, and hence witnessed by some

$$\bar{\alpha} : U_1 \times \ldots \times U_n \times iX \times iA \to iY$$

Taking the exponential adjoint we obtain

$$\alpha : U_1 \times \ldots \times U_n \times iA \to i(Y^X)$$

witnessing $(\alpha_F, \alpha_G) \in \mathcal{L}(Y^X)(U_1 \ldots U_n, A)$. Hence $\mathcal{L}$ preserves exponentials. □

Lemma 5.2 *If F and G are equivalent interpretations, $(i; \mathcal{L})$ is diagonal.*

Proof. Suppose (β_0, β_1) maps $\mathcal{L}(iB) \to \mathcal{L}(iB')$, then for any $h : U_1 \times \ldots \times U_n \times iA \to iB$ there is an $h' : U_1 \times \ldots \times U_n \times iA \to iB'$, such that $Fh' = Fh; \beta_0$ and $Gh' = Gh; \beta_1$. In particular, taking $h = \mathrm{id}_{iB}$, there is an $h' : iB \to iB'$ such that $Fh' = \beta_0$ and $Gh' = \beta_1$. But F and G are equivalent, so $\beta_0 = Fh' \approx Gh' = \beta_1$. Hence $(i; \mathcal{L})$ is diagonal, as required. □

This then establishes the completeness of logical relations for data abstraction.

Theorem 5.3 *Two interpretations F and $G : D \to C$ are equivalent if and only if they are linked by a logical relation.*

References

[1] S. Abramsky, R. Jagadeesan, and P. Malacaria. Full abstraction for PCF, 1995. To appear.

[2] M. Alimohamed. A Characterization of Lambda Definability in Categorical Models of Implicit Polymorphism. *Theoretical Computer Science*, 146:5–23, 1995.

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

[4] A. Jung and J. Tiuryn. A new characterization of lambda definability. In M. Bezem and J. F. Groote, editors, *Typed Lambda Calculi and Applications*, volume 664 of *Lecture Notes in Computer Science*, pages 245–257. Springer Verlag, 1993.

[5] Y. Kinoshita, P.W. O'Hearn, A.J. Power, E.P. Robinson, M. Takeyama, and R.D. Tennent. A functorial framework for data refinement. unpublished note, 1996.

[6] QingMing Ma and John C. Reynolds. Types, abstraction, and parametric polymorphism, part 2. In Stephen Brookes, Michael Main, Austin Melton, Michael Mislove, and David A. Schmidt, editors, *Mathematical Foundations of Programming Semantics*, volume 598 of *Lecture Notes in Computer Science*, pages 1–40, Berlin, 1992. Springer-Verlag.

[7] J.C. Mitchell. Type systems for programming languages. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume B*, pages 365–458. North-Holland, Amsterdam, 1990.

[8] John C. Reynolds. Types, abstraction and parametric polymorphism. In R. E. A. Mason, editor, *Information Processing 83*, pages 513–523, Amsterdam, 1983. Elsevier Science Publishers B. V. (North-Holland).