PERs Generalise Projections for Strictness Analysis (Extended Abstract)*

Sebastian Hunt
Department Of Computing
Imperial College
London SW7 2BZ

Abstract

We show how Wadler and Hughes's use of *Scott projections* to describe properties of functions ("Projections for Strictness Analysis", FPCA 1987) can be generalised by the use of *partial equivalence relations*. We describe an analysis (in the form of an *abstract interpretation*) for identifying such properties for functions defined in the simply typed λ-calculus. Our analysis has a very simple proof of correctness, based on the use of *logical relations*. We go on to consider how to derive 'best' correct interpretations for constants.

1 Introduction

In [WH87], Phil Wadler and John Hughes suggested a method of describing properties of functions using *projections*, and developed an analysis for identifying such properties for functions defined in a first-order functional language. (A projection is a continuous map on a cpo $\alpha : D \to D$, such that $\alpha \sqsubseteq \mathrm{id}_D$ and $\alpha \circ \alpha = \alpha$.)

For an example of [WH87]'s use of projections, let $\mathcal{L}$ be the domain of finite, partial and infinite lists of elements of some domain (the particular choice of element domain is not important for the purposes of the example) ordered in the usual way. The projection $H : \mathcal{L} \to \mathcal{L}$ is defined such that

$$\begin{array}{lcl} H(\bot) & = & \bot \\ H([\,]) & = & [\,] \\ H(\bot : y) & = & \bot \\ H(x : y) & = & x : (H\ y) \quad \text{if } x \neq \bot, \end{array}$$

where $[\,]$ is the empty list and $(:)$ is the cons operation. Suppose we have a lazy functional language in which we form an expression, $e_1 \circ e_2$ with denotation $[\![e_1 \circ e_2]\!] = [\![e_1]\!] \circ [\![e_2]\!]$. With the analysis of [WH87], we may be able to establish that the function $[\![e_1]\!]$ satisfies the property

$$[\![e_1]\!] \circ H = [\![e_1]\!],$$

*In: *Proceedings of the Third Annual Glasgow workshop on Functional Programming, Ullapool, 13–15 August, 1990* (Springer Verlag workshop series). The full version of this paper is available as Departmental Report DOC 90/14, Dept. of Computing, Imperial College.

so that $[\![e_1 \circ e_2]\!] = [\![e_1]\!] \circ (H \circ [\![e_2]\!])$. As is suggested in [WH87], we could then modify the code generation for the sub-expression e_2 such that every call to (:) which "contributes to the output" of e_2 is implemented by a 'head-strict' (i.e., left-strict) cons operation.

The analysis described in [WH87] is a backwards analysis, propagating information about the context in which a function is called to yield information about the way in which its arguments are used. Other forms of program analysis, in particular that using abstract interpretation as described in [Myc81, BHA86, Bur87], work in a forwards manner. The work presented here is motivated by the fact that each of the two methods has advantages over the other:

1. The [WH87] analysis can detect cases where a function is 'head-strict' as in the above example, and cases where a function ignores its argument. The analyses of [Myc81, BHA86, Bur87] are unable to incorporate this kind of test.
2. The [BHA86] and [Bur87] analyses are defined for the typed lambda calculus. The [WH87] analysis is restricted to a first-order language and has proved difficult to extend to higher-order languages (but see [Hug87] for one approach).

In this paper we describe an analysis in the form of an abstract interpretation which aims to combine the advantages of both methods. Our approach depends on using *partial equivalence relations* to generalise [WH87]'s use of projections to describe properties of functions.

The rest of this paper is organised as follows. Section 2 recalls the form of the condition tested for by the [WH87] analysis and gives the generalised version of it. Sections 3 and 4 review the formalism and results we will be using from [Abr90]. Section 5 describes the basic form of the analysis and proves it correct. Section 6 is concerned with the existence of best possible interpretations for constants. In section 7 we extend the analysis to a richer language. In section 8 we consider the relationship between our analysis and the analyses of [WH87] and [BHA86, Bur87] in a little more detail. Section 9 concludes.

2 Projections and Partial Equivalence Relations

All forms of strictness analysis aim to infer *safe* information about the denotation of a term. For simple strictness analysis we require a sound and effective test for the condition that $[\![e]\!]\ \perp = \perp$. It is well known how to achieve this using the technique of abstract interpretation ([Myc81], [BHA86], [Abr90]).

In an analysis based on the use of projections, the condition tested for is slightly more involved. Returning to the example in the introduction, having established that $[\![e_1 \circ e_2]\!]$ $= [\![e_1]\!] \circ (H \circ [\![e_2]\!])$ by analysing e_1, we may wish to analyse e_2. This time, however, we need not restrict ourselves to identifying a projection α such that $[\![e_2]\!] = [\![e_2]\!] \circ \alpha$, but can use our information about the context in which e_2 occurs and try to find a projection α such that $H \circ [\![e_2]\!] = H \circ [\![e_2]\!] \circ \alpha$. In general, for an expression (in a first-order functional language) denoting a function $f : D \to D'$, given projections $\alpha : D \to D$ and $\beta : D' \to D'$, the analysis technique of [WH87] can provide a sound and effective test for the condition that:

$$\beta \circ f = \beta \circ f \circ \alpha. \tag{1}$$

As is shown in [WH87], with a little inventiveness it is possible to describe ordinary strictness using projections. For a function $f : D \to D'$, let $f_\perp : D_\perp \to D'_\perp$ be the function defined by

$$f_\perp\ x = \begin{cases} \perp_{D'_\perp} & \text{if } x = \perp_{D_\perp} \\ \mathit{lift}(f\ y) & \text{if } x = \mathit{lift}(y) \end{cases}$$

where *lift* is the evident injection into a lifted domain. It is then easy to verify that

$$f\ \perp_D = \perp_{D'} \iff \mathit{Str}_{D'} \circ f_\perp = \mathit{Str}_{D'} \circ f_\perp \circ \mathit{Str}_D,$$

where $\mathit{Str}_D : D_\perp \to D_\perp$ is the projection defined by

$$\mathit{Str}_D\ x = \begin{cases} \perp_{D_\perp} & \text{if } x = \perp_{D_\perp} \\ \perp_{D_\perp} & \text{if } x = \mathit{lift}(\perp_D) \\ x & \text{otherwise} \end{cases}$$

The use of projections is not restricted to descriptions of strictness. Another property which can conveniently be expressed using projections is that of a function being constant, i.e., of a function's result being independent of its argument. To do this we can simply use the projection $\lambda x.\perp$, since f is constant iff $f \circ \lambda x.\perp = f$.

2.1 Partial Equivalence Relations

A *partial equivalence relation* (per) on a set D, is a transitive and symmetric relation $S \subseteq D \times D$. If S is such a per, then $|S|$ is the set $\{x \in D \mid (x,x) \in S\}$. We will write $x : S$ if $x \in |S|$. For each $x \in |S|$, $[x]_S$ is the equivalence class of x by S. The set of pers on D is closed under arbitrary intersection.

Partial equivalence relations over various applicative structures have been used to construct models of the polymorphic lambda calculus (see, for example, [Asp90]). In [AP90], pers over a *domain* are used for the same purpose. We will be using pers to describe properties of functions over Scott domains, in such a way that abstract interpretation may be used to test whether the properties hold of functions defined in a simply typed lambda calculus. Although the relationship is not explored in this paper, it seems clear that there are strong connections between our use of pers and that described in [AP90].

Definition 2.1 For each projection, $\beta : D \to D$, the equivalence relation $E_\beta \subseteq D \times D$ is defined by: $x\ E_\beta\ y \iff \beta(x) = \beta(y)$. □

Definition 2.2 For Scott domains D, D' relations $S \subseteq D \times D$ and $T \subseteq D' \times D'$, the relation $S \Rightarrow T \subseteq [D \to D']^2$ is defined by:

$$f\ (S \Rightarrow T)\ g \iff \forall x, y \in D.\, x\ S\ y \Rightarrow (f\ x)\ T\ (g\ y)$$

where $[D \to D']$ is the Scott domain of continuous maps from D to D'. □

It is straightforward to show that if S and T are pers, then so is $S \Rightarrow T$. (In fact, there is a CCC with pers as objects and the $S \Rightarrow T$ as exponent objects. See, e.g. [Asp90].) Borrowing the types convention, we will write a per of the form $S_1 \Rightarrow (S_2 \Rightarrow (\ldots(S_n \Rightarrow T)\ldots))$ as $S_1 \Rightarrow S_2 \Rightarrow \ldots S_n \Rightarrow T$. Recalling that for a per P we write $x : P$ to mean $x \in |P|$, we note the following equivalence:

$$\begin{gathered} f : S_1 \Rightarrow \ldots \Rightarrow S_k \Rightarrow T \\ \iff \\ (x_1\ S_1\ x'_1) \wedge \ldots \wedge (x_k\ S_k\ x'_k) \Rightarrow (f\ x_1\ \ldots\ x_k)\ T\ (f\ x'_1\ \ldots\ x'_k). \end{gathered}$$

Proposition 2.3 For projections $\alpha : D \to D$, $\beta : D' \to D'$, and function $f \in [D \to D']$

$$\beta \circ f = \beta \circ f \circ \alpha \iff f : E_\alpha \Rightarrow E_\beta$$

□

Thus any test for a condition of form (1) can be re-expressed as a test for a condition of the form

$$f : S \Rightarrow T \tag{2}$$

where S and T are pers.

Clearly, since E_α and E_β in the above proposition are equivalence relations, we do not need pers if we simply wish to rephrase statements of form (1). However, there are two good reasons for allowing the generality of pers:

1. With pers, we can use form (2) both to rephrase statements of form (1) and, without lifting domains, to describe ordinary strictness. For each domain D, let Bot_D be the per $\{(\perp_D, \perp_D)\}$. Then for any $f : D \to D'$

$$f \perp_D = \perp_{D'} \iff f : Bot_D \Rightarrow Bot_{D'}.$$

2. In general, even if P and Q are both equivalence relations on domains, $P \Rightarrow Q$ will be a *partial* equivalence relation (consider $E_{\lambda x.\perp} \Rightarrow E_{id}$, for example). As we shall see in what follows, this makes it natural to consider pers rather than just equivalence relations when analysing definitions of higher-order functions.

The analysis presented in [WH87] is a first-order backwards analysis implementing tests of form (1). Here we present a higher-order forwards analysis which implements tests of the more general form (2).

With the exception of sub-section 4.2, the next two sections are essentially a review of the first three sections of [Abr90].

3 The Typed Lambda Calculus

Given a set of base types $\{A, B, \ldots\}$ we build type expressions, $\sigma, \tau, \ldots$ as follows:

$$\tau ::= A \mid \tau_1 \to \tau_2$$

A *Language* L is specified by giving a set of base types and a set of *typed constants* $\{c_\sigma\}$. For each type σ, we assume an infinite set of typed variables $Var_\sigma = \{x^\sigma, y^\sigma, \ldots\}$. Then $\Lambda_T(L)$ consists of typed terms $e : \sigma$ built according to the usual rules for the simply typed λ-calculus.

An *interpretation* I of L is specified by $I = (\left\{D^I_A\right\}, \left\{c^I_\sigma\right\})$, such that for each base type A, D^I_A is a Scott domain, and for each c_σ, $c^I_\sigma \in D^I_\sigma$, where the D^I_A are extended to higher types such that $D^I_{\sigma\to\tau} = [D^I_\sigma \to D^I_\tau]$ We will write the least element of D^I_σ as $\perp^I_\sigma$.

An interpretation I determines the semantic valuation function

$$[\![\cdot]\!]^I : \Lambda_T(L) \to Env^I \to \cup D^I_\sigma$$

where $Env^I = \left\{Env^I_\sigma\right\}$, $Env^I_\sigma = Var_\sigma \to D^I_\sigma$.

Assume that for each type σ there is a constant $Y_{(\sigma\to\sigma)\to\sigma}$. Following [Abr90], we say that I is a *normal* interpretation if

$$Y^I_{(\sigma\to\sigma)\to\sigma}(f) = \bigsqcup_{n=0}^{\infty} f^n(\perp^I_\sigma).$$

4 Logical Relations

[Abr90] shows how logical relations can be applied to the proof of correctness of program analyses based on abstract interpretation.

A *binary logical relation* $R : I, K$ is a family $\{R_\sigma\}$, with $R_\sigma \subseteq D^I_\sigma \times D^K_\sigma$ such that for all σ, τ, for all $f \in D^I_{\sigma\to\tau}, h \in D^K_{\sigma\to\tau}$:

$$\begin{array}{l} \quad\quad\quad\quad f \; R_{\sigma\to\tau} \; h \\ \quad\quad\quad\quad\quad \Longleftrightarrow \\ \forall x \in D^I_\sigma, a \in D^K_\sigma .\, x \; R_\sigma \; a \Rightarrow (f \; x) \; R_\tau \; (h \; a). \end{array}$$

While binary logical relations are sufficient for establishing correctness of analyses such as those of [BHA86, Bur87], they are not quite enough for our purposes. We wish to use a family of *ternary* relations $R = \{R_\sigma\}$ with $R_\sigma \subseteq D^I_\sigma \times D^I_\sigma \times D^K_\sigma$, such that each domain point $a \in D^K_\sigma$ is associated with a per given by $\left\{(x, y) \in D^I_\sigma \times D^I_\sigma \mid R_\sigma(x, y, a)\right\}$. For this reason, in reviewing the results of [Abr90] we make the (routine) generalisation to the n-ary case.

A *logical relation* $R : I_1, \ldots, I_n$ is a family $\{R_\sigma\}$, with $R_\sigma \subseteq D^{I_1}_\sigma \times \ldots \times D^{I_n}_\sigma$ such that for all σ, τ, for all $f_1 \in D^{I_1}_{\sigma\to\tau}, \ldots, f_n \in D^{I_n}_{\sigma\to\tau}$:

$$\begin{array}{c} (f_1, \ldots, f_n) \in R_{\sigma\to\tau} \\ \Longleftrightarrow \\ \forall a_1 \in D^{I_1}_\sigma, \ldots, a_n \in D^{I_n}_\sigma .\, (a_1, \ldots, a_n) \in R_\sigma \Rightarrow (f_1 \; a_1, \ldots, f_n \; a_n) \in R_\tau. \end{array} \tag{3}$$

Any family of relations on the base types extends uniquely to a logical relation by using (3) as a definition of R_σ at the higher types.

A property P of relations is *inherited* if for all logical relations $R : I_1, \ldots, I_k$, whenever $P(R_A)$ holds for all base-types, A, then $P(R_\sigma)$ holds for all types σ.

4.1 Relating Interpretations of Terms

If $R : I_1, \ldots, I_k$ is a logical relation, then for $\rho_i \in \mathit{Env}^{I_i}$ we will write $(\rho_1, \ldots, \rho_k) \in R$, meaning that $\forall x^\tau .\, (\rho_1 \; x^\tau, \ldots, \rho_k \; x^\tau) \in R_\tau$.

Proposition 4.1 ([Abr90]) Let $R : I_1, \ldots, I_k$ be a logical relation. Suppose that for each constant c_τ, $(c^{I_1}_\tau, \ldots, c^{I_k}_\tau) \in R_\tau$. Then for all $e : \sigma$, for all $\rho_i \in \mathit{Env}^{I_i}$

$$(\rho_1, \ldots, \rho_k) \in R \Rightarrow ([\![e]\!]^{I_1} \; \rho_1, \ldots, [\![e]\!]^{I_k} \; \rho_k) \in R_\sigma$$

$\square$

4.2 Concretisation maps

In what follows it will often be convenient to present a logical relation in an alternative form. Given a logical relation $R : I, K$, we may form the family of *concretisation maps* $\gamma^R = \left\{\gamma^R_\sigma\right\}$ as follows:

$$\begin{aligned} \gamma^R_\sigma &: D^K_\sigma \to \wp(D^I_\sigma) \\ \gamma^R_\sigma \; a &= \left\{x \in D^I_\sigma \mid (x, a) \in R_\sigma\right\} \end{aligned}$$

where $\wp(D)$ is the power set of D. Clearly R and γ^R may be used interchangeably, since $(x, a) \in R_\sigma$ iff $x \in \gamma^R_\sigma \; a$. Now for $f : D \to D'$, $X \subseteq D$, and $Y \subseteq D'$, let $(X \Rightarrow Y) \subseteq [D \to D']$ be defined by

$$f \in (X \Rightarrow Y) \iff \forall x \in X.\, f\ x \in Y.$$

Then we may express the fact that R is logical purely in terms of γ^R:

Lemma 4.2 $R : I, K$ is logical iff for all σ, τ for all $f \in [D^K_\sigma \to D^K_\tau]$

$$\gamma^R_{\sigma\to\tau}\ f = \bigcap_{a \in D^K_\sigma} \gamma^R_\sigma\ a \Rightarrow \gamma^R_\tau(f a),$$

□

We can make analogous definitions given a family of relations $R : I, I, K$. We form the concretisation maps as follows:

$$\begin{aligned} \gamma^R_\sigma &: D^K_\sigma \to \wp(D^I_\sigma \times D^I_\sigma) \\ \gamma^R_\sigma\ a &= \left\{(x,y) \in D^I_\sigma \times D^I_\sigma \mid (x,y,a) \in R_\sigma\right\} \end{aligned}$$

Again, R and γ^R may be used interchangeably. Furthermore, we may express the fact that R is logical purely in terms of γ^R:

Lemma 4.3 $R : I, I, K$ is logical iff for all σ, τ for all $f \in [D^K_\sigma \to D^K_\tau]$

$$\gamma^R_{\sigma\to\tau}\ f = \bigcap_{a \in D^K_\sigma} \gamma^R_\sigma\ a \Rightarrow \gamma^R_\tau(f a)$$

□

The analyses of [BHA86, Bur87] test for conditions of the form $f \in (X \Rightarrow Y)$ (in more familiar notation, $f(X) \subseteq Y$), where for $f : D \to D'$, $X \subseteq D$ and $Y \subseteq D'$ are Scott-closed sets. Both analyses use an abstract interpretation, say K, and can be proved correct using a binary logical relation $R : I, K$ where each $\gamma^R_\sigma\ a$ is a Scott-closed set[1].

We wish to test for conditions of the form $f : S \Rightarrow T$ where S and T are pers. We will use an abstract interpretation J and prove correctness in terms of a ternary logical relation $P : I, I, J$ where each $\gamma^P_\sigma\ a$ is a per.

5 The Analysis

We assume a standard interpretation I which is normal, in which $D^I_{bool} = \{tt, ff\}_\perp$, the $\mathrm{cond}^I_{bool\to A\to A\to A}$ are conditionals, and all other first-order constants $c_{A_1\to\cdots\to A_n\to B}$ are strict. Our analysis consists of two parts: an abstract interpretation (together with a logical relation) and a proof of correctness.

5.1 The Abstract Interpretation J

Our abstract interpretation has base domains

$$D^J_A = \mathbf{3} = \{\text{BOT}, \text{ID}, \text{ALL}\},\ \text{with BOT} \sqsubset \text{ID} \sqsubset \text{ALL}.$$

The values in the abstract domains are related to those in the standard domains by the logical relation $P : I, I, J$, where:

1. $\gamma^P_A\ \text{ALL} = \left\{(x,y) \mid x, y \in D^I_A\right\}$

[1] The development of [BHA86, Bur87] preceded that of [Abr90] and correctness was actually proved without using logical relations.

2. $\gamma^P_A\ \text{ID} = \left\{(x, x) \mid x \in D^I_A\right\}$

3. $\gamma^P_A\ \text{BOT} = \left\{(\perp^I_A, \perp^I_A)\right\}$

It is straightforward to verify that for each $a \in D^J_A$, $\gamma^P_A\ a$ is a per, and using lemma 4.3, that this property is inherited at the higher types. Hence:

Proposition 5.1 For all σ, for each $a \in D^J_\sigma$, $\gamma^P_\sigma\ a$ is a per. □

(For the points ID and ALL, the pers correspond to projections, with $\gamma^P_A\ \text{ID} = E_{id}$ and $\gamma^P_A\ \text{ALL} = E_{\lambda x.\perp}$.)

J is a normal interpretation with:

- $$\text{cond}^J_{bool\to A\to A\to A}\ a\ b\ c \;=\; \begin{cases} \text{BOT} & \text{if } a = \text{BOT} \\ b \sqcup c & \text{if } a = \text{ID} \\ \text{BOT} & \text{if } a = \text{ALL and } b \sqcup c = \text{BOT} \\ \text{ALL} & \text{if } a = \text{ALL and } b \sqcup c \neq \text{BOT} \end{cases}$$

- $$c^J_{A_1\to\cdots\to A_n\to B}\ a_1 \ldots a_n \;=\; \begin{cases} \text{ID} & \text{if } n = 0 \\ \text{BOT} & \text{if } \exists 1 \leq i \leq n.\, a_i = \text{BOT} \\ (\bigsqcup_{i=1}^{n} a_i) & \text{otherwise} \end{cases}$$

Intuitively, we may think of the abstract domain points as describing different degrees of "fixedness", where decreasing in the abstract domain ordering corresponds to an increasing degree of "fixedness". For example, we may think of BOT as meaning "fixed at bottom", ID as meaning "fixed at an unknown value", and ALL as meaning "varying". The definition of cond^J can then be understood in the following terms. If we fix the first argument to cond^I at bottom, then the result is fixed at bottom. If we fix the first argument to cond^I at an unknown value, then the result is at least as fixed as the most variable of the other two arguments. If we allow the first argument to cond^I to vary, then the result may vary unless both of the other arguments are fixed at bottom, in which case the result will be fixed at bottom.

Proposition 5.2 For each c_σ, $(c^I_\sigma, c^I_\sigma) \in \gamma^P_\sigma\ c^J_\sigma$. □

5.2 Correctness of J

We will say that J is *correct* iff for all terms $e : \sigma_1 \to \ldots \to \sigma_n \to B$, for all $a_i \in D^J_{\sigma_i}$, $1 \leq i \leq n$, and environments $\rho \in Env^I$, $\rho' \in Env^J$ such that $(\rho, \rho)\ P\ \rho'$:

$$([\![e]\!]^J \rho' a_1 \ldots a_n = b) \Rightarrow ([\![e]\!]^I \rho) : \gamma^P_{\sigma_1} a_1 \Rightarrow \ldots \Rightarrow \gamma^P_{\sigma_n} a_n \Rightarrow \gamma^P_B b \tag{4}$$

Theorem 5.3 J is correct. □

5.3 Example

We conclude this section with an example application of the analysis. In the following we suppress the types for the sake of clarity. Consider the following function definition:

$$\begin{aligned} \text{pfac} = Y(\lambda f.\lambda x.\lambda y.\,\text{cond}\ & (x = 0) \\ & 1 \\ & x * (f\ (x-1)\ (y+1))) \end{aligned}$$

The standard interpretation of pfac is a 'perverse' factorial function which takes two arguments, the second of which is never needed. The interpretation under J is the function pfac^J, where for all $a, b \in \mathbf{3}$, $\text{pfac}^J\ a\ b = a$. Our correctness result allows us to infer the following about the standard interpretation pfac^I:

1. pfac^I is strict in its first argument, since pfac^J BOT ID = BOT, hence:

$$\begin{aligned}&\forall (x_1, x_2) \in \gamma^P \text{ BOT}.\, \forall (y_1, y_2) \in \gamma^P \text{ ID}.\, (\text{pfac}^I\ x_1\ y_1, \text{pfac}^I\ x_2\ y_2) \in \gamma^P \text{ BOT}\\ &\Rightarrow \forall y.\, \text{pfac}^I \perp y = \perp\end{aligned}$$

2. pfac^I ignores its second argument, since pfac^J ID ALL = ID, hence:

$$\begin{aligned}&\forall (x_1, x_2) \in \gamma^P \text{ ID}.\, \forall (y_1, y_2) \in \gamma^P \text{ ALL}.\, (\text{pfac}^I\ x_1\ y_1, \text{pfac}^I\ x_2\ y_2) \in \gamma^P \text{ ID}\\ &\Rightarrow \forall x.\, \forall y_1, y_2.\, \text{pfac}^I\ x\ y_1 = \text{pfac}^I\ x\ y_2\end{aligned}$$

In the full paper we show how J can be used to test for strictness and constancy at all types.

6 Abstraction Maps and Best Interpretations

While we have presented a correct analysis, we have not yet dealt with a language incorporating list types. It is possible to extend the analysis to a richer language, but to do so we have to construct correct interpretations of higher type constants. This is less straightforward than for the first-order constants we have so far considered and it would be helpful to have some systematic way of deriving such interpretations.

6.1 Left Adjoints for Concretisation Maps

We will say that a logical relation $R : I, I, K$ satisfies property (M) if each γ^R_σ preserves meets, i.e., for each subset $X \subseteq D^K_\sigma$

$$\gamma^R_\sigma(\bigsqcap X) = \bigcap_{a \in X} \gamma^R_\sigma a$$

Note that this implies monotonicity of the γ^R_σ.

Proposition 6.1 When each D^K_A (hence each D^K_σ) is finite, property (M) is inherited. □

Freyd's Adjoint Functor Theorem ([Lan71]), immediately gives us the following:

Corollary 6.1.1 If each D^K_A is a finite lattice, then each γ^R_σ has a left adjoint iff each γ^R_A preserves meets. □

In the remainder of this section we will assume that each γ^R_σ has left adjoint, α^R_σ. The left adjoint of γ^R_σ can be constructed as:

$$\alpha^R_\sigma\ X = \bigsqcap \left\{ a \in D^K_\sigma \mid X \subseteq \gamma^R_\sigma\ a \right\}$$

However, it is more useful to have an 'iterated' construction for the family $\alpha^R = \left\{ \alpha^R_\sigma \right\}$:

Proposition 6.2 For each σ, τ, subset $F \subseteq [D^I_\sigma \to D^I_\tau]^2$, for each $a \in D^K_\sigma$:

$$\alpha^R_{\sigma \to \tau}\ F\ a = \alpha^R_\tau \left\{ (f\ x, f'\ x') \mid (f, f') \in F, (x, x') \in \gamma^R_\sigma\ a \right\}$$

□

We will say that c^K_σ is a *correct interpretation* for c_σ if $(c^I_\sigma, c^I_\sigma) \in \gamma^R_\sigma\ c^K_\sigma$. We will say further that c^K_σ is the *best* interpretation for c_σ if it is the minimum correct interpretation, since this guarantees that the set $\gamma^R_\sigma\ c^K_\sigma$ is as small as possible.

We can use the α^R_σ to derive best interpretations for the c_σ by defining:

$$\begin{aligned} abs^R_\sigma &: D^I_\sigma \times D^I_\sigma \to D^J_\sigma \\ abs^R_\sigma(x,y) &= \alpha^R_\sigma\ \{(x,y)\} \end{aligned}$$

Proposition 6.3 For all σ, for all $x, y \in D^I_\sigma$, for all $a \in D^K_\sigma$

$$(x,y) \in \gamma^R_\sigma\ a \iff abs^R_\sigma(x,y) \sqsubseteq a$$

Proof: We observe that $(x,y) \in \gamma^R_\sigma\ a$ iff $\{(x,y)\} \subseteq \gamma^R_\sigma\ a$ and hence that $(x,y) \in \gamma^R_\sigma\ a \iff \alpha^R_\sigma\ \{(x,y)\} \sqsubseteq a$, since α^R_σ left adjoint to γ^R_σ. □

Thus $abs^R_\sigma(c^I_\sigma, c^I_\sigma)$ is the best interpretation for c_σ.

Observing that the α^R_σ preserve joins, it is straightforward to derive the following iterated construction for abs^R from that for α^R (proposition 6.2):

$$abs^R_{\sigma\to\tau}(f,g)\ a = \bigsqcup \left\{ abs^R_\tau(f\ x, g\ y) \mid abs^R_\sigma(x,y) \sqsubseteq a \right\}$$

We note that there is a striking similarity between our abs^R maps and the abs^S of [Abr90], but also that the abs^R are less well behaved than the abs^S in that in general they are not monotone.

7 Extending the Analysis

In this section we sketch how the analysis described in section 5 can be extended to a language including list types.

We extend the language L introduced in section 4 by adding the type *Alist* for each existing base type A (in what follows A, B will continue to range over the original base types) and by adding the constant $\mathrm{case}_{B\to(A\to Alist\to B)\to Alist\to B}$ for each A, B.

We assume that the standard interpretation is extended so that D^I_{Alist} is the domain of finite, partial and infinite lists of elements of D^I_A, ordered in the usual way. The standard interpretation of a case constant is:

$$\mathrm{case}^I_{B\to(A\to Alist\to B)\to Alist\to B}\ b\ f\ l = \begin{cases} \bot^I_B & \text{if } l = \bot^I_{Alist} \\ b & \text{if } l = [\,] \\ f\ x\ y & \text{if } l = x : y \end{cases}$$

There are numerous ways in which J might be extended, we consider a particularly simple example by way of illustration. The extension to list types is:

$$D^J_{Alist} = \{\mathrm{BOT}, \mathrm{ID}, \mathrm{H}, \mathrm{ALL}\} \text{ with } \mathrm{BOT} \sqsubset \mathrm{ID} \sqsubset \mathrm{H} \sqsubset \mathrm{ALL}.$$

The relation P is extended such that $\gamma^P_{Alist}\ \mathrm{H} = E_H$, where H is the projection defined in the introduction.

The case constants are interpreted as follows:

$$\begin{aligned} \mathrm{case}^J\ b\ h\ \mathrm{BOT} &= \mathrm{BOT} \\ \mathrm{case}^J\ b\ h\ \mathrm{ID} &= b \sqcup (h\ \mathrm{ID}\ \mathrm{ID}) \\ \mathrm{case}^J\ b\ h\ \mathrm{H} &= \begin{cases} b \sqcup (h\ \mathrm{ID}\ \mathrm{H}) & \text{if } h\ \mathrm{BOT}\ \mathrm{ID} = \mathrm{BOT} \\ \mathrm{ALL} & \text{otherwise} \end{cases} \\ \mathrm{case}^J\ b\ h\ \mathrm{ALL} &= \begin{cases} \mathrm{BOT} & \text{if } b = \mathrm{BOT} \text{ and } h = \bot^J_{A\to Alist\to B} \\ \mathrm{ALL} & \text{otherwise} \end{cases} \end{aligned}$$

Lemma 7.1 P satisfies property (M). □

Proposition 7.2 Let σ range over types of the form $B \to (A \to \mathit{Alist} \to B) \to \mathit{Alist} \to B$. Then for each σ, case^J_σ is correct; i.e. $\mathrm{case}^J_\sigma \sqsupseteq \mathit{abs}^P_\sigma(\mathrm{case}^I_\sigma, \mathrm{case}^I_\sigma)$. □

Note: it is disappointing that we have been unable to show that the case^J are best rather than just correct. The proof of proposition 7.2 would go through with the inequality strengthened to equality in the case that for all σ, in addition to property (M) we had $\alpha^P_\sigma \circ \gamma^P_\sigma = \mathit{id}$ (equivalently, γ^P_σ injective). Although the property that $\alpha^R_\sigma \circ \gamma^R_\sigma = \mathit{id}$ is *not* inherited for arbitrary meet-preserving γ^R, the question of whether it holds for γ^P is still open.

7.1 Example

We will say that a function $f \in [D^I_{\mathit{Alist}} \to D^I_B]$ is *head-strict* if $f = f \circ H$, or equivalently, if $f : \gamma^P_{\mathit{Alist}}\ \mathrm{H} \Rightarrow \gamma^P_B\ \mathrm{ID}$.

The following example illustrates that our analysis can detect head-strictness. The standard interpretation of the following definition is czero^I, a function which tests whether or not a list of integers contains a 0:

$$
\begin{array}{llll}
\mathrm{czero} = Y(\lambda f.\lambda l.\,\mathrm{case} & \mathrm{false} & & \\
 & \lambda x.\lambda y.\,\mathrm{cond} & (x = 0) & \\
 & & \mathrm{true} & \\
 & & (f\ y) & \\
 & l\,) & &
\end{array}
$$

The interpretation under J is the function czero^J, where:

$$
\mathrm{czero}^J\ a \quad = \quad \begin{cases} \mathrm{ID} & \text{if } a = \mathrm{H} \\ a & \text{otherwise} \end{cases}
$$

Thus czero^I is head-strict.

8 Relationship to Other Analyses

In this section we consider the relationship between our analysis and the analyses of [WH87] and [BHA86, Bur87] in a little more detail.

8.1 Comparing PER and Projection Analyses

It is fairly easy to compare the kinds of questions which may be asked using pers with those which may be asked using projections. Proposition 2.3 shows that any question about a function f which can be asked using projections in form $\beta \circ f = \beta \circ f \circ \alpha$, can also be asked using pers in form $f : P \Rightarrow Q$. Since there are pers which do not correspond to projections, there are questions which can be asked using pers (and answered using abstract interpretation) which cannot be asked using projections.

On the other hand, it is difficult to compare a per-based *analysis* with a projection-based analysis. Our analysis is not restricted to a first-order language and so may be said to be more general than that of [WH87]. But the analysis of [WH87] is a backwards analysis, whereas ours is a forward analysis. This makes it hard to reason about the relative quality of the answers which may be obtained in the cases where both analyses are applicable. It is beyond the scope of this paper to attempt such a comparison (but see the paper "Towards Relating Forwards and Backwards Analyses" by John Hughes and John Launchbury, in this proceedings).

8.2 Comparing PER and Set Analyses

The analyses of [BHA86, Bur87] use abstract interpretation and can be proved correct in terms of logical relations of the form $R : I, K$, so that each $\gamma_\sigma^R\ a$ is a set. (Recall from section 4.2 that for $R : I, K$ the concretisation maps are defined by $\gamma_\sigma^R\ a = \left\{x \in D_\sigma^I \mid R_\sigma(x, a)\right\}$.) Such analyses may be called set-based, in that they test for conditions of the form $f(X) \subseteq Y$, where X and Y are subsets of the domain and co-domain of f respectively. It is clear that questions such as whether f is constant, cannot be asked in this way. (We can ask whether f is the *particular* constant function which always returns e, for example, by asking whether $f(D) \subseteq \{e\}$, but we cannot just ask if f is constant.) We have seen that a per-based analysis *can* be used to ask such questions, posed in the form $f : P \Rightarrow Q$.

We now wish to establish that any set-based analysis using abstract interpretation can be expressed as one based on pers, thus demonstrating what is intuitively clear, that the use of pers gives us at strictly more power than the use of sets. To show this in a simple way, we concentrate on set-based analyses in which correctness is given by what [Abr90] calls a $\top$-*universal* logical relation, where $R : I, K$ is $\top$-universal iff for each type σ, D_σ^K has a greatest element $\top_\sigma^K$, and $\gamma_\sigma^R\ \top_\sigma^K = D_\sigma^I$. ($\top$-universality is an inherited property.)

If K and I are interpretations, related by a logical relation $R : I, K$, we define the logical relation $P(R) : I, I, K$ at the base types such that for all $x, y \in D_A^I$, and $a \in D_A^K$:

$$P(R)_A(x, y, a) \iff (x = y) \wedge R_A(x, a).$$

Equivalently, we may define $P(R)$ at the base-types via the concretisation maps thus:

$$\gamma_A^{P(R)}\ a = \Delta(\gamma_A^R\ a),$$

where for any set X, $\Delta(X) = \{(x, x) \mid x \in X\}$.

Proposition 8.1 If $R : I, K$ is a $\top$-universal logical relation, then for all types σ, for all $a \in D_\sigma^K$

$$\gamma_\sigma^{P(R)}\ a = \Delta(\gamma_\sigma^R\ a).$$

□

Corollary 8.1.1 If $R : I, K$ is a $\top$-universal logical relation, then for all types σ, for all constants c_σ, c_σ^K is correct with respect to R iff c_σ^K is correct with respect to $P(R)$. □

9 Conclusions

We have shown how partial equivalence relations can be used to describe properties of functions in a way that subsumes both the use of projections in [WH87] and the use of sets in [BHA86, Bur87].

In its relation to the use of projections, our work has three main advantages over [WH87]. Firstly, the use of pers fits in naturally with the use of logical relations, thus allowing us to develop an analysis in the form of an abstract interpretation with a very simple proof of correctness. Secondly, the use of pers extends smoothly to the higher order case (this is really another aspect of the first advantage). Lastly, we are able to formulate a test for strictness which does not involve lifting the domains (if nothing else, this has the attraction of relative simplicity).

The development of the material in this paper is far from complete. There are two areas in particular which we hope to be able to improve upon:

- Our failure to derive best interpretations for the case_σ constants in section 7 highlights the fact that we have been unable to show that the γ_σ^P are injective. This implies that there may be 'redundant' points in the D_σ^J at higher types.

- The development of section 6.1 is lacking, in that it does not directly address the per structure inherent in the abstract interpretation. The use of the power set for the γ_σ^P also seems unsatisfactory since it ignores the domain structure of the D_σ^I. This raises the question of whether there is a class of pers which takes account of the underlying domain structure and which is appropriate to static program analysis (the class of 'good' pers of [AP90] is one possibility).

It seems likely that an investigation of the latter area will shed light on the former.

Acknowledgements

Thanks to Dave Sands and Thomas Jensen for the many hours of their time spent discussing the ideas presented in this paper. Thanks also to all those who reviewed this paper (particularly John Hughes), for their numerous constructive criticisms.

References

[Abr90] Samson Abramsky. Abstract interpretation, logical relations and Kan extensions. *Journal of Logic and Computation*, 1(1):5–39, 1990.

[AP90] M. Abadi and G. Plotkin. A per model of polymorphism and recursive types. In *Logic in Computer Science*. IEEE, 1990.

[Asp90] A. G. Asperti. *Categorical Topics in Computer Science*. PhD thesis, Università di Pisa, 1990.

[BHA86] Geoffery L. Burn, Chris Hankin, and Samson Abramsky. Strictness analysis of higher-order functions. *Science of Computer Programming*, 7:249–278, November 1986.

[Bur87] Geoffery L. Burn. *Abstract Interpretation and the Parallel Evaluation of Functional Languages*. PhD thesis, Department of Computing, Imperial College of Science and Technology, University of London, 1987.

[Hug87] R.J.M. Hughes. Backwards analysis of functional programs. DoC Research Report CSC/87/R3, University of Glasgow, March 1987.

[Lan71] Saunders Mac Lane. *Categories for the Working Mathematician*. Springer-Verlag, Berlin, 1971.

[Myc81] Alan Mycroft. *Abstract Interpretation and Optimising Transformations for Applicative Programs*. PhD thesis, University of Edinburgh., 1981.

[WH87] Philip Wadler and R.J.M. Hughes. Projections for strictness analysis. In *LNCS 274. Functional Programming Languages and Computer Architectures, Oregon, USA*, 1987.