Reasoning about Sequential Functions via Logical Relations

Kurt Sieber *

1. Introduction

In his seminal paper [Plo77], Plotkin introduced the functional language PCF ('programming language for computable functions') together with the standard denotational model of cpo's and continuous functions. He proved that this model is computationally adequate for PCF, but *not* fully abstract.

In order to obtain full abstraction he extended PCF by a parallel conditional operator. The problem with this operator is, that it changes the nature of the language. All computations in the original language PCF can be executed sequentially, but the new operator requires expressions to be evaluated in parallel.

Here we address the problem of full abstraction for the original *sequential* language PCF, i. e. instead of extending the language we try to improve the model. Our approach is to cut down the standard model with the aid of certain logical relations, which we call *sequentiality relations*. We give a semantic characterization of these relations and illustrate how they can be used to reason about sequential (i. e. PCF-definable) functions. Finally we prove that this style of reasoning is 'complete' for proving observational congruences between closed PCF-expressions of order ≤ 3. Technically, this completeness can be expressed as a full abstraction result for the sublanguage which consists of these expressions.

2. Sequential PCF

In [Plo77], PCF is defined as a simply typed λ-calculus over the ground types ι (of integers) and o (of Booleans). For the sake of simplicity we throw out the

*Current affiliation: Institut für Angewandte Informatik und formale Beschreibungsverfahren, Universität Karlsruhe. Permanent address: FB 14 Informatik, Universität des Saarlandes, W-6600 Saarbrücken, Germany.

Booleans and simulate them by integers; 0 stands for 'true', all other integers for 'false'. Hence the set *Type* of all types τ is defined by

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

and the constants c with their types are

$$\begin{array}{rll} \underline{n}: & \iota & \text{for each } n \in \mathbb{N}, \\ succ, pred: & \iota \to \iota, & \\ cond: & \iota \to \iota \to \iota \to \iota, & \\ \mathrm{Y}_\tau: & (\tau \to \tau) \to \tau & \text{for each type } \tau. \end{array}$$

PCF-expressions $M, N, \ldots$ are defined to be the well-typed λ-expressions over these constants and over a set of typed variables $x^\tau, y^\tau, \ldots$. A PCF-*program* is a closed PCF-expression of ground type ι. Instead of $cond\, M\, N\, P$ we also use the more familiar notation **if** M **then** N **else** P **fi**. Ω_τ is used as an abbreviation for the (diverging) expression $\mathrm{Y}_\tau(\lambda x^\tau.\, x)$.[1]

The standard denotational semantics for PCF associates a cpo D^τ with each type τ, namely

$$\begin{array}{rcll} D^\iota & = & \mathbb{N}_\perp & \text{(the flat cpo of natural numbers),} \\ D^{\tau_1 \to \tau_2} & = & (D^{\tau_1} \to D^{\tau_2}) & \text{(the cpo of continuous functions).} \end{array}$$

The constants are given their usual interpretation. In particular $[\![Y_\tau]\!]$ is the fixed point operator on D^τ and $[\![cond]\!]$ is the sequential conditional, defined by

$$[\![cond]\!]\, b\, d\, e = \begin{cases} d & \text{if } b = 0, \\ e & \text{if } b \in \mathbb{N} \setminus \{0\}, \\ \perp & \text{if } b = \perp. \end{cases}$$

The meaning $[\![M]\!]\,\rho$ of an expression M in an environment ρ is defined as usual. If M is closed, then we abbreviate $[\![M]\!]\,\rho$ to $[\![M]\!]$, because it doesn't depend on ρ. An element $d \in D^\tau$ is called (PCF-)*definable* if there is a closed PCF-expression M such that $[\![M]\!] = d$.

We will now define the notions of computational adequacy and full abstraction. Let '$\to$' be the one step transition relation between (closed) PCF-expressions as defined in [Plo77] (the exact definition of '$\to$' is not important for our purpose). Then the *observable behavior* of a PCF-program P can be defined by

$$beh(P) = \{n \mid P \xrightarrow{*} \underline{n}\}.$$

This set is either empty, if P diverges, or a singleton $\{n\}$, if P terminates with normal form $\underline{n}$. A denotational semantics is called *computationally adequate*

[1] From now on we will use the type superscript τ only for the binding occurence of a variable.

if it correctly describes the observable behavior of programs. For PCF this is made precise in the following theorem.

Theorem 2.1 *Let* $[\![\ ]\!]$ *be the standard semantics of cpo's and continuous functions. Then*

$$[\![P]\!] = n \quad \textit{iff} \quad beh(P) = \{n\} \quad (\textit{iff} \quad P \xrightarrow{*} \underline{n}).$$

for every PCF-program P *and every* $n \in \mathbb{N}$.

An immediate consequence is, that for any two programs P and Q:

$$[\![P]\!] = [\![Q]\!] \quad \text{iff} \quad beh(P) = beh(Q).$$

We take Theorem 2.1 for granted; its proof can be found in [Plo77].

Definition 2.2

(a) *Two PCF-expressions* M *and* N *of the same type are called* observationally congruent *(notation:* $M \approx N$*) if* $beh(C[M]) = beh(C[N])$ *for every program context* $C[\ ]$.

(b) *A denotational semantics is called* fully abstract, *if observational congruence coincides with semantic equality, i. e. if for any two expressions* M *and* N*:*

$$M \approx N \quad \textit{iff} \quad [\![M]\!] = [\![N]\!].$$

Note that—in the light of computational adequacy—observational congruence and full abstraction can be defined purely in terms of the denotational semantics: $M \approx N$ holds iff $[\![C[M]]\!] = [\![C[N]]\!]$ for every program context $C[\ ]$. Hence full abstraction means that $[\![M]\!] = [\![N]\!]$ iff $[\![C[M]]\!] = [\![C[N]]\!]$ for all program contexts $C[\ ]$. One direction of this equivalence is usually trivial; if the denotational semantics is defined by induction on the structure of expressions, then $[\![M]\!] = [\![N]\!]$ implies $[\![C[M]]\!] = [\![C[N]]\!]$. Hence a proof of full abstraction only consists of the other direction: If $[\![M]\!] \neq [\![N]\!]$, then one must find some program context $C[\ ]$ such that $[\![C[M]]\!] \neq [\![C[N]]\!]$.

For proving observational congruence of two closed expressions, it is sufficient to consider *applicative* contexts. This is made precise by the following theorem, known as the *Context Lemma*.

Theorem 2.3 *Two closed PCF-expressions* $M, N : \sigma_1 \to \ldots \to \sigma_n \to \iota$ *are observationally congruent iff* $[\![MP_1 \ldots P_n]\!] = [\![NP_1 \ldots P_n]\!]$ *for all closed expressions* $P_1 : \sigma_1, \ldots, P_n : \sigma_n$.

3. Logical Relations

It is well-known why full abstraction fails for PCF, but we repeat the argument here, because we want to illustrate the use of logical relations. Consider the following three equations for a function f of type $\iota \to \iota \to \iota$:

$$\begin{array}{llll} f & 0 & \bot & = 0 \\ f & \bot & 0 & = 0 \\ f & 1 & 1 & = 1 \end{array} \tag{1}$$

Such a function f exists in the model, namely the *parallel or* operator *por*, defined by

$$por\, d\, e = \begin{cases} 0 & \text{if } d = 0 \text{ or } e = 0, \\ 1 & \text{if } d, e \in \mathbb{N} \setminus \{0\}, \\ \bot & \text{otherwise.} \end{cases}$$

On the other hand, there is no PCF-definable function f which satisfies (1). In [Plo77] this was proved with the aid of a so-called *Activity lemma*; here we will use logical relations for the proof. But let us first see how full abstraction fails. The point is that it can be tested with closed PCF-expressions whether a function f satisfies (1). For $i = 1, 2$, let $PORTEST_i$ of type $(\iota \to \iota \to \iota) \to \iota$ be defined by[2]

$$PORTEST_i \equiv \lambda y^{\iota \to \iota \to \iota}.\, \mathbf{if}\ y\, \underline{0}\, \Omega_\iota = \underline{0} \wedge y\, \Omega_\iota\, \underline{0} = \underline{0} \wedge y\, \underline{1}\, \underline{1} = \underline{1}\ \mathbf{then}\ \underline{i}\ \mathbf{else}\ \Omega_\iota\ \mathbf{fi}.$$

Then

$$[\![PORTEST_i]\!]\, f = \begin{cases} i & \text{if } f \text{ satisfies (1)}, \\ \bot & \text{otherwise.} \end{cases}$$

This means that $[\![PORTEST_1]\!] \neq [\![PORTEST_2]\!]$, because they differ on the function *por*, but $PORTEST_1 \approx PORTEST_2$ by Theorem 2.3, because $[\![PORTEST_1 M]\!] = [\![PORTEST_2 M]\!]$ for every closed PCF-expression M.

We now introduce logical relations.

Definition 3.1

(a) *An* n*-ary* logical relation R *on a* λ*-model* $(D^\tau)_{\tau \in Type}$ *is a family of relations* $R^\tau \subseteq (D^\tau)^n$ *such that for all types* σ, τ *and* $f_1, \ldots, f_n \in D^{\sigma \to \tau}$

$$R^{\sigma \to \tau}(f_1, \ldots, f_n) \Leftrightarrow \forall d_1, \ldots, d_n . R^\sigma(d_1, \ldots, d_n) \Rightarrow R^\tau(f_1 d_1, \ldots, f_n d_n).$$

(b) *An element* $d \in D^\tau$ *is called* invariant *under* R *if* $R^\tau(d, \ldots, d)$ *holds.*

Note that a logical relation R is uniquely determined by its definition on ground type(s). The importance of logical relations is expressed by the following theorem, known as the *Main Lemma for Logical Relations* ([Plo80]).

[2] We freely use functions like $=, \wedge, \leq : \iota \to \iota \to \iota$ which are obviously definable.

Theorem 3.2 *Let R be a logical relation on a λ-model such that $[\![c]\!]$ is invariant under R for every constant c. Then $[\![M]\!]$ is invariant under R for every closed λ-term M over these constants.*

We are now ready to prove with mathematical rigor that no PCF-definable function f can satisfy (1). Let R be the 3-ary logical relation defined by

$$R^\iota(d_1, d_2, d_3) \Leftrightarrow (d_1 = \bot) \vee (d_2 = \bot) \vee (d_1 = d_2 = d_3).$$

It is easy to see (and follows from Theorem 3.4) that $[\![c]\!]$ is invariant under R for each PCF-constant c. Hence $[\![M]\!]$ is invariant under R for every closed PCF-expression M. But then (1) cannot hold for $f = [\![M]\!]$, because R^ι holds for the argument columns $(0, \bot, 1)$ and $(\bot, 0, 1)$, but *not* for the result column $(0, 0, 1)$.

The example shows us how we can rule out elements like *por* (which destroy full abstraction) with the aid of logical relations. Two questions naturally arise:

- Is there a semantic characterization of all logical relations under which the PCF–constants are invariant?
- Are logical relations sufficient to rule out *all* elements which destroy full abstraction?

For the first question we will immediately give a positive answer; the second question will be investigated in section 4.

Definition 3.3

(a) *For each $n \geq 0$ and each pair of sets $A \subseteq B \subseteq \{1, \ldots, n\}$ let $S^n_{A,B} \subseteq (D^\iota)^n$ be defined by*

$$S^n_{A,B}(d_1, \ldots, d_n) \Leftrightarrow (\exists i \in A.\, d_i = \bot) \vee (\forall i, j \in B.\, d_i = d_j).$$

An n-ary logical relation R is called a sequentiality relation *if R^ι is an intersection of relations of the form $S^n_{A,B}$.*

(b) *An element $d \in D^\tau$ is called* logically sequential *if it is invariant under all sequentiality relations.*

Note that the logical relation R which we have used to rule out *por*, was a sequentiality relation, namely $R^\iota = S^3_{\{1,2\},\{1,2,3\}}$.

Theorem 3.4 *The sequentiality relations are the* only *logical relations under which the meanings of all PCF-constants are invariant.*

Proof :

(a) Let R be a sequentiality relation.

The meaning of a ground type constant is invariant under R, because $R^\iota(d,\ldots,d)$ holds for all $d \in D^\iota$.

For a first order constant c, it is sufficient to prove that $[\![c]\!]$ is invariant under the relations $S^n_{A,B}$. This is obvious for the functions $[\![pred]\!]$ and $[\![succ]\!]$, because they are strict. For $[\![cond]\!]$ assume that $(b_1,\ldots,b_n),(d_1,\ldots,d_n)$ and $(e_1,\ldots,e_n)$ are in $S^n_{A,B}$ and let $v_i = [\![cond]\!]\, b_i\, d_i\, e_i$ for $i = 1,\ldots,n$. If $b_i = \bot$ for some $i \in A$, then $v_i = \bot$. Otherwise $b_i = b_j$ for all $i,j \in B$, hence either $v_i = d_i$ for all $i \in B$ or $v_i = e_i$ for all $i \in B$. In all three cases it follows that $(v_1,\ldots,v_n)$ is in $S^n_{A,B}$.

In order to see that the fixed point operators are invariant under R, first note that $R^\iota(\bot_\iota,\ldots,\bot_\iota)$ and hence $R^\tau(\bot_\tau,\ldots,\bot_\tau)$ by induction on τ. Moreover, each R^τ is closed under lubs of directed sets (also by induction on τ). Hence, if $R^{\tau\to\tau}(f,\ldots,f)$ for some $f \in D^{\tau\to\tau}$, then we first obtain $R^\tau(f^n\bot_\tau,\ldots,f^n\bot_\tau)$ for all $n \in \mathbb{N}$ and this implies $R^\tau([\![\mathrm{Y}_\tau]\!]\, f,\ldots,[\![\mathrm{Y}_\tau]\!]\, f)$.

(b) Let R be a logical relation, under which the meanings of all PCF-constants—and hence, by Theorem 3.2, the meanings of all closed PCF-expressions—are invariant. Let S be the intersection of all $S^n_{A,B}$, which contain R^ι. If we can prove that $S \subseteq R^\iota$, then $S = R^\iota$ and this means that R is a sequentiality relation.

Let $d = (d_1,\ldots,d_n) \in S$. By induction on the size of $C \subseteq \{1,\ldots,n\}$ we show that there is some vector $e = (e_1,\ldots,e_n) \in R^\iota$ with

$$e_i = d_i \text{ for all } i \in C. \tag{2}$$

Setting $C = \{1,\ldots,n\}$ we obtain the desired result that $d \in R^\iota$.

For $|C| \leq 1$ we can choose some constant vector e; hence let $|C| \geq 2$.

Case 1: $d_i \neq \bot$ for all $i \in C$.

Let $T = \{v \in (D^\iota)^n \mid v_i \neq \bot \text{ for all } i \in C\}$. Each $v \in T$ defines an equivalence relation $\sim_v$ on C by: $i \sim_v j \Leftrightarrow v_i = v_j$. Let $u \in R^\iota \cap T$ be such that $\sim_u$ is minimal in $\{\sim_v \mid v \in R^\iota \cap T\}$, i. e. no other vector $v \in R^\iota \cap T$ is 'finer' than u (note that $R^\iota \cap T \neq \emptyset$, because $R^\iota(d,\ldots,d)$ for all $d \in D^\iota$). We will prove $R^\iota \subseteq S^n_{B,B}$ for each equivalence class B of $\sim_u$.

Let $i \in C$, let $B = \{j \in C \mid u_j = u_i\}$ be the equivalence class of i, and assume that $v \in R^\iota \setminus S^n_{B,B}$. Then $v_j \neq \bot$ for all $j \in B$ and $v_j \neq v_k$ for some pair of

indices $j, k \in B$. This allows us to find a vector $w \in R^\iota \cap T$ which is finer than u, namely: Let $u_0 \neq u_l$ for all $l \in C$ and define

$$M \equiv \lambda x^\iota.\, \lambda y^\iota.\, \textbf{if } x = \underline{u_i} \textbf{ then if } y = \underline{v_j} \textbf{ then } \underline{u_0} \textbf{ else } x \textbf{ fi else } x \textbf{ fi}.$$

Then $w = (\llbracket M \rrbracket\, u_1\, v_1, \ldots, \llbracket M \rrbracket\, u_n\, v_n) \in R^\iota \cap T$ and it is easy to see that $\sim_w \subset \sim_u$ (namely the equivalence class B of $\sim_u$ is split into two equivalence classes of $\sim_w$). This contradicts the minimality of $\sim_u$, hence the assumption was wrong and $R^\iota \subseteq S^n_{B,B}$ is proved.

It follows that $S \subseteq S^n_{B,B}$ and hence $d \in S^n_{B,B}$ for all equivalence classes B of $\sim_u$. This implies $\sim_u \subseteq \sim_d$, because $d_i \neq \bot$ for all $i \in C$. But then it is easy to find a closed PCF-expression N such that $d_i = \llbracket N \rrbracket\, u_i$ for all $i \in C$, i.e. $e = (\llbracket N \rrbracket\, u_1, \ldots, \llbracket N \rrbracket\, u_n) \in R^\iota$ satisfies (2).

Case 2: $d_i = \bot$ for some $i \in C$.

If $R^\iota \not\subseteq S^n_{C\setminus\{i\},C}$, then there is some $u \in R^\iota$ such that $u_j \neq \bot$ for *all* $j \in C \setminus \{i\}$ and $u_k \neq u_i$ for *some* $k \in C \setminus \{i\}$. By induction hypothesis there are vectors $v, w \in R^\iota$ with

$$v_j = d_j \text{ for all } j \in C \setminus \{i\} \quad \text{and} \quad w_j = d_j \text{ for all } j \in C \setminus \{k\}.$$

These two vectors can now be 'merged' by a closed PCF-expression. Let

$$M \equiv \lambda x^\iota.\, \lambda y^\iota.\, \lambda z^\iota.\, \textbf{if } x = \underline{u_k} \textbf{ then } y \textbf{ else } z \textbf{ fi}$$

and let $e = (\llbracket M \rrbracket\, u_1\, v_1\, w_1, \ldots, \llbracket M \rrbracket\, u_n\, v_n\, w_n)$. Then e is in R^ι and a straightforward calculation shows that it satisfies (2).

If $R^\iota \subseteq S^n_{C\setminus\{i\},C}$, then $S \subseteq S^n_{C\setminus\{i\},C}$ and hence $d \in S^n_{C\setminus\{i\},C}$. As $d_i = \bot$, this implies $d_k = \bot$ for some $k \in C \setminus \{i\}$. Let $v, w \in R^\iota$ be defined as above (wrt the new index k). If $v_i = \bot$, then we can take $e = v$, otherwise let

$$M \equiv \lambda x^\iota.\, \lambda y^\iota.\, \textbf{if } x = \underline{v_i} \textbf{ then } y \textbf{ else } x \textbf{ fi}$$

and take $e = (\llbracket M \rrbracket\, v_1\, w_1, \ldots, \llbracket M \rrbracket\, v_n\, w_n)$. In both cases e is in R^ι and satisfies (2). □

We conclude this section with a sophisticated example of 'reasoning about sequential functions'. It is essentially the last example on p. 269 of [Cur86]. We use the familiar notation for threshold functions ([Plo77]), defined by

$$(u_1 \Rightarrow \ldots \Rightarrow u_n \Rightarrow v)\, d_1 \ldots d_n = \begin{cases} v & \text{if } d_i \sqsupseteq u_i \text{ for } i = 1, \ldots, n, \\ \bot & \text{otherwise.} \end{cases}$$

Example 3.5 *Let* $g_1 = (\bot \Rightarrow 1 \Rightarrow 0) \sqcup (1 \Rightarrow 0 \Rightarrow 0)$, $g_2 = (0 \Rightarrow \bot \Rightarrow 0)$, $g_3 = (\bot \Rightarrow 0 \Rightarrow 0)$ *and* $g_4 = \bot$. *Then there is no PCF-definable function* f *of type* $(\iota \to \iota \to \iota) \to \iota$, *which satisfies the equations*

$$\begin{aligned} f\ g_1 &= 0 \\ f\ g_2 &= 0 \\ f\ g_3 &= 0 \\ f\ g_4 &= \bot \end{aligned}$$

Proof : Let R be the sequentiality relation defined by

$$R^\iota = S^4_{\{1,2\},\{1,2\}} \cap S^4_{\{1,3\},\{1,3\}} \cap S^4_{\{1,2,3\},\{1,2,3,4\}}.$$

The result column $(0,0,0,\bot)$ is *not* in $S^4_{\{1,2,3\},\{1,2,3,4\}} \supseteq R^\iota$, hence we must only convince ourselves that $(g_1,\ldots,g_4)$ is in $R^{\iota\to\iota\to\iota}$. Assume that $g_i\, d_{i1}\, d_{i2} = e_i$ for $i = 1,\ldots,4$, where $(e_1,\ldots,e_4) \notin R^\iota$. This is only possible for $(e_1,\ldots,e_4) = (0,0,0,\bot)$, and then one of the following two cases must occur:

$$\begin{array}{lll} (d_{11},\ldots,d_{41}) \sqsupseteq (\bot,0,\bot,\bot) & \text{and} & (d_{12},\ldots,d_{42}) \sqsupseteq (1,\bot,0,\bot) \\ (d_{11},\ldots,d_{41}) \sqsupseteq (1,0,\bot,\bot) & \text{and} & (d_{12},\ldots,d_{42}) \sqsupseteq (0,\bot,0,\bot). \end{array}$$

In the first case $(d_{12},\ldots,d_{42}) \notin S^4_{\{1,3\},\{1,3\}} \supseteq R^\iota$ and in the second case $(d_{11},\ldots,d_{41}) \notin S^4_{\{1,2\},\{1,2\}} \supseteq R^\iota$. Thus we have proved that $(g_1,\ldots,g_4)$ is in $R^{\iota\to\iota\to\iota}$. □

Of course Example 3.5 can be continued in the same sense as the *parallel or*-example: There is a function f in the model which satisfies the 4 equations and it is easy to construct PCF-expressions of type $((\iota \to \iota \to \iota) \to \iota) \to \iota$, which differ in f iff f satisfies these equations.

4. A Full Abstraction Result

Theorem 4.1 *Let* $\tau = \sigma_1 \to \ldots \to \sigma_n \to \iota$ *be a type of order 1 or 2 and let* $f \in D^\tau$ *be a logically sequential function. Moreover, let* $(d_{i1},\ldots d_{in}) \in D^{\sigma_1} \times \ldots \times D^{\sigma_n}$ *for* $i = 1,\ldots,m$. *Then there is a closed PCF-expression* M *of type* τ *such that*

$$f d_{i1} \ldots d_{in} = [\![M]\!]\, d_{i1} \ldots d_{in} \quad \text{for } i = 1,\ldots,m.$$

Proof : Let R be the logical relation defined by

$$R^\iota(e_1,\ldots,e_m) \Leftrightarrow \exists \text{ closed } M : \tau.\, \forall i \in \{1,\ldots,m\}.\, e_i = [\![M]\!]\, d_{i1},\ldots,d_{in}.$$

We must prove that $R^\iota(f\, d_{11} \ldots d_{1n},\ldots,f\, d_{m1} \ldots d_{mn})$ holds.

(a) First we prove that R is a sequentiality relation, i.e. that all PCF-constants are invariant under R. This is obvious for a ground type constant c: $R^\iota([\![c]\!], \ldots, [\![c]\!])$ holds, because $[\![\lambda x_1 \ldots \lambda x_n . c]\!]\, d_{i1} \ldots d_{in} = [\![c]\!]$ for $i = 1, \ldots, m$. Now let c be a first order constant, say $c : \iota \to \iota \to \iota$. If $R^\iota(u_1, \ldots, u_m)$ and $R^\iota(v_1, \ldots, v_m)$, then there are closed PCF-expressions U and V such that for $i = 1, \ldots, m$

$$u_i = [\![U]\!]\, d_{i1} \ldots d_{in} \quad \text{and} \quad v_i = [\![V]\!]\, d_{i1} \ldots d_{in}$$

and hence

$$[\![c]\!]\, u_i\, v_i = [\![\lambda x_1 \ldots \lambda x_n . c\,(U\, x_1 \ldots x_n)\,(V\, x_1 \ldots x_n)]\!]\, d_{i1} \ldots d_{in}.$$

This means $R^\iota([\![c]\!]\, u_1\, v_1, \ldots, [\![c]\!]\, u_m\, v_m)$, hence $R^{\iota\to\iota\to\iota}([\![c]\!], \ldots, [\![c]\!])$ is proved.

(b) Now we prove that $R^{\sigma_j}(d_{1j}, \ldots, d_{mj})$ holds for $j = 1, \ldots, n$. The proof is similar as in (a). If σ_j is the ground type ι, then $R^\iota(d_{1j}, \ldots, d_{mj})$ holds because $d_{ij} = [\![\lambda x_1 \ldots \lambda x_n . x_j]\!]\, d_{i1} \ldots d_{in}$ for $i = 1, \ldots, m$. If σ_j is a first order type, say $\sigma_j = \iota \to \iota \to \iota$, and $(u_1, \ldots, u_m), (v_1, \ldots, v_m)$ are as above, then

$$d_{ij}\, u_i\, v_i = [\![\lambda x_1 \ldots \lambda x_n . x_j\,(U\, x_1 \ldots\, x_n)\,(V\, x_1 \ldots\, x_n)]\!]\, d_{i1} \ldots d_{in}$$

for $i = 1, \ldots m$, and this proves $R^{\iota\to\iota\to\iota}(d_{1j}, \ldots, d_{mj})$.

From (a) it follows that $R^\tau(f, \ldots, f)$ holds, and together with (b) this yields the desired result $R^\iota(f\, d_{11} \ldots d_{1n}, \ldots, f\, d_{m1} \ldots d_{mn})$. □

Theorem 4.1 tells us that for each logically sequential function f there is always a definable function which coincides with f on finitely many given arguments. But we need a stronger result: f must be the *least upper bound* of definable elements. The clue for proving this result is the definability of finite projections.

Definition 4.2 *Let D be a cpo. A function $p : D \to D$ is called a* projection, *if $p \circ p = p$ and $p \sqsubseteq id_D$ (the identity on D). A projection is called* finite *if $p(D)$ is a finite set. D is called an* SFP-domain, *if there is a sequence of finite projections $p_1 \sqsubseteq p_2 \sqsubseteq \ldots$ such that $id_D = \bigsqcup_{i\in\mathbb{N}} p_i$.*

It is well-known that the cpo's D^τ are SFP-domains with the projections p_i^τ defined as follows:

$$\begin{aligned} p_i^\iota\, d &= \begin{cases} d & \text{if } d \in \{0, \ldots, i\}, \\ \bot & \text{otherwise,} \end{cases} \\ p^{\sigma\to\tau} f &= p^\tau \circ f \circ p^\sigma. \end{aligned}$$

These projections are definable by closed PCF-expression P_i^τ, defined by:

$$\begin{aligned} P_i^\iota &\equiv \lambda x^\iota . \mathbf{if}\ x \leq \underline{i}\ \mathbf{then}\ x\ \mathbf{else}\ \Omega_\iota\ \mathbf{fi} \\ P_i^{\sigma\to\tau} &\equiv \lambda y^{\sigma\to\tau} . \lambda x^\sigma . P_i^\tau (y\,(P_i^\sigma x)) \end{aligned}$$

Theorem 4.3 *Let $\tau = \sigma_1 \to \ldots \sigma_n \to \iota$ be a type of order 1 or 2 and let $f \in D^\tau$ be a logically sequential function. Then f is the lub of a directed set of definable elements.*

Proof: We know that $f = \bigsqcup_{i \in \mathbb{N}} p_i^\tau f$, where the p_i^τ are the finite projections. Hence it is sufficient to prove that $p_i^\tau f$ is definable for each $i \in \mathbb{N}$. Now note that $p_i^\tau g\, d_1 \ldots d_n = p_i^\iota (g\,(p_i^{\sigma_1} d_1) \ldots (p_i^{\sigma_n} d_n))$ for all $g \in D^\tau$, $d_1 \in D^{\sigma_1}, \ldots, d_n \in D^{\sigma_n}$. The set $p_i^{\sigma_1}(D^{\sigma_1}) \times \ldots \times p_i^{\sigma_n}(D^{\sigma_n})$ is finite, hence—by Theorem 4.1—there is a closed PCF-expression M such that

$$[\![M]\!]\,(p_i^{\sigma_1} d_1) \ldots (p_i^{\sigma_n} d_n) = f\,(p_i^{\sigma_1} d_1) \ldots (p_i^{\sigma_n} d_n)$$

for all $(d_1, \ldots, d_n) \in D^{\sigma_1} \times \ldots \times D^{\sigma_n}$ and this finally implies

$$p_i^\tau f = p_i^\tau [\![M]\!] = [\![P_i^\tau M]\!]. \qquad \square$$

Theorem 4.3 implies that sequentiality relations are sufficient to prove all observational congruences between closed expressions of order ≤ 3:

Theorem 4.4 *For each type τ let L^τ be the set of logically sequential elements of D^τ. If $\tau = \sigma_1 \to \ldots \to \sigma_n \to \iota$ is a type of order ≤ 3 and M, N are closed PCF-terms of type τ, then*

$$M \approx N \Leftrightarrow [\![M]\!]\, d_1 \ldots d_n = [\![N]\!]\, d_1 \ldots d_n \text{ for all } (d_1, \ldots, d_n) \in L^{\sigma_1} \times \ldots \times L^{\sigma_n}.$$

Proof:

'$\Rightarrow$': Assume that $[\![M]\!]\, d_1 \ldots d_n \neq [\![N]\!]\, d_1 \ldots d_n$ for some logically sequential elements $d_1, \ldots, d_n$. By Theorem 4.3, $d_1, \ldots, d_n$ can be approximated by definable elements. Hence the continuity of M and N implies that there are closed PCF-expressions $P_1, \ldots, P_n$ such that $[\![M P_1 \ldots P_n]\!] \neq [\![N P_1 \ldots P_n]\!]$, i.e. $M \not\approx N$.

'$\Leftarrow$': Follows immediately from the Context Lemma. $\square$

Theorem 4.4 nearly looks like a full abstraction result. A technical difficulty is that $L = (L^\tau)_{\tau \in Type}$ is not a λ-model because it is not extensional. But if $\hat{L}$ is the extensional collapse of L, then we obtain:

Theorem 4.5 *For all closed PCF-terms M, N of order ≤ 3*

$$M \approx N \quad \textit{iff} \quad [\![M]\!] = [\![N]\!] \text{ in the model } \hat{L}.$$

Hence the model $\hat{L}$ is fully abstract for the sublanguage of closed expressions of order ≤ 3, (and this immediately generalizes to arbitrary expressions of order ≤ 3 whose only free variables are of order ≤ 2).

Unfortunately we do not know what happens at order > 3. The proof of Theorem 4.1 heavily relies on the fact that the functions are of order 1 or 2, hence it doesn't give us any hint how to generalize the result to order > 2. A similar difficulty appeared in [Plo80], where logical relations are used to characterize functions of order ≤ 2, which are definable in the *pure* λ-calculus. Plotkin succeeded to generalize the result to order > 2 with the aid of *Kripke* logical relations. But his proof seems to be closely tailored to the pure λ-calculus, so we couldn't see how a similar idea might work in our case.

5. Conclusion

Of course this is not the first approach to characterize sequential functions.

(a) A model which is fully abstract for sequential PCF has been defined in [Mul86]. This model is generally criticized because of its 'syntactic flavor', which does not provide any new insight into the *logic* of the language.

(b) Models of *stable* and—most recently—*strongly stable* functions have been developed (cf. [BCL85, BE91, Cur86]). Stability rules out the *parallel or* operator, but still allows some first order functions which are not sequential (cf. [Cur86]). Strong stability fills this gap, but hasn't made much progress on second order functions (cf. the conclusion of [BE91]).

(c) An alternative approach are *sequential algorithms* ([BCL85, Cur86]), which lead to a model which is *not extensional* (and hence can't be fully abstract for PCF). Most recently, a variant of sequential algorithms has been used in [CF91] to obtain a fully abstract model for a new 'sequential' *extension* of PCF.

How does our result relate to these approaches?

Technically, there is some similarity with (a), but our semantic characterization of the relations (Theorem 3.4) *does* give us an insight into the logic of PCF: It provides a method for proving observational congruences and we know that this method is in some sense complete. On the other hand we have achieved more than (b), because we could completely characterize second order functions. Finally, (c) is of course a result about a different language.

References

[BCL85] G. Berry, P.-L. Curien, and J.-J. Lévy. Full abstraction for sequential languages: The state of the art. In M. Nivat and J. C. Reynolds, editors, *Algebraic Methods in Semantics*, pages 89–132. Cambridge Univ. Press, 1985.

[BE91] A. Bucciarelli and T. Ehrhard. Sequentiality and strong stability. In 5^{th} *Symposium on Logic in Computer Science*, pages 138–145. IEEE, 1991.

[CF91] R. Cartwright and M. Felleisen. Observable sequentiality and full abstraction. Draft, Dept. of Comp. Sc., Rice University, Houston TX, 1991.

[Cur86] P.-L. Curien. *Categorical Combinators, Sequential Algorithms and Functional Programs.* Research Notes in Theoretical Computer Science. Pitman, Wiley, 1986.

[Mul86] K. Mulmuley. *Full Abstraction and Semantic Equivalences.* ACM Doctoral Dissertation Award. MIT Press, 1986.

[Plo77] G. D. Plotkin. LCF considered as a programming language. *Theoretical Computer Science*, 5:223–256, 1977.

[Plo80] G. D. Plotkin. Lambda-definability in the full type hierarchy. In J. Seldin and J. Hindley, editors, *To H. B. Curry: Essays on Combinatory Logic, Lambda Calculus and Formalism*, pages 363–374. Academic Press, 1980.