Math. Struct. in Comp. Science (1999), *vol.* 9, *pp.* 1–20. Printed in the United Kingdom

Full abstraction, totality and PCF

GORDON PLOTKIN

Department of Computer Science, University of Edinburgh, King's Buildings, Edinburgh EH9 3JZ, UK.
E-mail: gdp@dcs.ed.ac.uk.

Received 10 July 1997; revised 15 July 1998

Inspired by a question of Riecke, we consider the interaction of totality and full abstraction, asking whether full abstraction holds for Scott's model of cpos and continuous functions if one restricts to total programs and total observations. The answer is negative, as there are distinct operational and denotational notions of totality. However, when two terms are each total in both senses, they are totally equivalent operationally iff they are totally equivalent in the Scott model. Analysing further, we consider sequential and parallel versions of PCF and several models: Scott's model of continuous functions, Milner's fully abstract model of PCF and their effective submodels. We investigate how totality differs between these models. Some apparently rather difficult open problems arise that essentially concern whether the sequential and parallel versions of PCF have the same expressive power, in the sense of total equivalence.

1. Introduction

In the survey article Fiore *et al.* (1996), Jon Riecke raised the interesting question of the relationship between full abstraction and divergence

One final question is in order: how good is the original model [of PCF] used by Scott? ..... the model appears to classify correctly many equations. Indeed the counterexample above due to Plotkin relies on having divergence built into the terms that are operationally equivalent but denotationally distinct. Since programmers (hopefully) do not write divergent subterms, are there counterexamples to full abstraction where divergence is not necessary? At what level of the type hierarchy do such examples occur? Answers to these questions may tell us where reasoning principles for programs can be derived from simpler principles.

Taking a literal reading, it is not hard to construct such a counterexample, adapting the usual one a little. Define (using evident abbreviations) the PCF terms

$$M = \lambda b^o.\lambda f^{o\to o\to o}.\,\mathbf{if}\ (f\,\mathbf{t}\,b)\ \mathbf{and}\ (f\,b\,\mathbf{t})\ \mathbf{and}\ \neg(f\,\mathbf{f}\,\mathbf{f})\ \mathbf{then}\ \mathbf{t}\ \mathbf{else}\ \mathbf{f}$$

and

$$N = \lambda b^o.\lambda f^{o\to o\to o}.\,\mathbf{if}\ (f\,\mathbf{t}\,b)\ \mathbf{and}\ (f\,b\,\mathbf{t})\ \mathbf{and}\ \neg(f\,\mathbf{f}\,\mathbf{f})\ \mathbf{then}\ (b\ \mathbf{or}\ \neg b)\ \mathbf{else}\ \mathbf{f}.$$

Then M and N do not have any divergent subterms. Indeed, they do not have divergence built into them, in the sense that they are (hereditarily) total. That is, given (as is surely fair!) total arguments, they yield total results. On the other hand, M and N are

operationally equivalent but distinct in Scott's model, differing when applied to $\perp$ and 'parallel or'. However, one could still object that the notion of full abstraction should itself be questioned as only total arguments should be considered, and M and N yield identical results when applied to total arguments.

To make all this precise, we must define what we mean by totality, and then give a notion of full abstraction appropriate for totality. It turns out that one can define both operational and semantical notions of totality – as predicates or (equivalently!) relations – and also of equivalence in total contexts. It would be natural to restrict to total terms, and take full abstraction for totality to mean that for total terms the semantical and operational notions of equivalence in total contexts coincide. Unfortunately, the two notions of totality differ, with counterexamples occurring already at type level 2. So this definition of full abstraction for totality does not make sense. However, in the cases where it does make sense, and we restrict to terms total in both senses, it turns out that the two notions of equivalence coincide.

One is therefore in a somewhat unexpected situation, which can perhaps be better understood if we generalise our understanding of full abstraction. Given a notion that has both operational and semantical definitions, we can say that the semantics is fully abstract for that notion if the operational version of the notion holds for a term iff the semantical version holds for its denotation. Now we can say that Scott's model is not fully abstract for totality, but is for total equivalence (if restricted to terms total in both senses). This point of view has been investigated in Longley and Plotkin (1998), defining full abstraction relative to a class of formulae.

One possible objection to our analysis is that programmers might prefer to vary their notions of totality. They may sometimes be satisfied with less stringent requirements, requiring that functions be total only for some inputs, say those satisfying a certain formula in some logical language. However, at a higher type they may then be more stringent, requiring totality for all such functions. Developing this idea would lead to a family of notions of totality at each type, linked to a choice of logical language. Presumably this could still be accommodated within the framework of Longley and Plotkin (1998), and the contents of this paper would then appear as a kind of idealisation.

Such an investigation might lead, rather directly, to a useful *logic of totality*, perhaps as hinted at by Riecke. The logic could perhaps be formulated as a refinement type system of the kind given in Denney (1998), where refinements are given by logical formulae. A closely related subject is that of refinement type systems for totality and other program analyses – see, *e.g.*, Coppo *et al.* (1997). Here one replaces logic and proof by type inference procedures performed during compilation. The semantics of such type systems can be given by assigning to each refinement type a suitable predicate or relation over its underlying type – indeed totality predicates were already used for totality analyses in Abramsky (1990).

Finally, one can ask what light is now thrown on notions of equivalence of (possibly) non-total programs. First, such notions remain interesting even if one is concerned ultimately only with total programs. For at some stage one will have programs that are not yet known to be total (say, if they use the recursion operator), and one would like to prove them total with the aid of equational reasoning, and that may involve working

with non-total subterms. Now we can still argue for interest in full abstraction in the traditional sense, as operational equivalence is easily seen to be the largest contextual equivalence relation on terms that respects (the operational notion of) totality; a related point can be made for operational inequivalence.

The study of continuous total functionals over the natural numbers was originated by Kleene and Kreisel. An excellent account can be found in the paper by Gandy and Hyland (Gandy and Hyland 1977); more recently, Normann has written a brief survey article (Normann 1998). Ershov (Ershov 1975; Ershov 1977) made the connection with the hierarchy of continuous functions. This has been pursued further by Normann and others; they study dependent and inductive types, considering both continuous and stable domain theory – see, *e.g.*, Kristiansen and Normann (1995), Kristiansen and Normann (1997) and Berger (1997). One aim of this work is to find models of Martin-Löf's type theory (Normann 1996; Waagbø 1998); another is to extend the Kleene–Kreisel density theorem to transfinite hierarchies and use that to revisit higher recursion theory – see, *e.g.*, Berger (1997) and Normann (1997b).

Totality is best considered as extra structure on domains, typically a predicate or a partial equivalence relation. These are naturally defined hereditarily on the type structure, as pointed out by Ershov, Scott and Hyland (Ershov 1974; Ershov 1977; Scott 1976; Hyland 1975). Berger has studied another possibility in order to give an abstract treatment of density: he considers two predicates, one on the domain and one on the continuous functions from the domain to the booleans – see Berger (1997) and Stoltenberg-Hansen *et al.* (1994); see also Loader (1997), Kristiansen and Normann (1995) and Berger (1997). Finally, one can argue that the study of totality at higher types is a prerequisite to the study of complexity at higher types; work here has been carried out by Cook and others (Cook 1989; Kapron and Cook 1996).

Below, we proceed systematically, studying two languages and four models. The languages are PCF and PCF^{++}, where the latter is PCF extended with two constants, $\mathbf{por} : o \to o \to o$ for *parallel or*, and $\exists : (\iota \to o) \to o$ for *(continuous) existential quantification.* The language PCF is adapted to sequential functional computation; PCF^{++} is adapted to parallel functional computation. Scott's model $\mathscr{C}$ of continuous functions (Scott 1993) and Milner's model $\mathscr{S}$ of sequential continuous functions (Milner 1977) are the standard fully abstract models of PCF^{++} and PCF, respectively. They have natural submodels $\mathscr{C}^o$ and $\mathscr{S}^o$ consisting of those elements definable in, respectively, PCF^{++} and PCF; equivalently, these are the submodels of the effective elements. Instead of Milner's model, we might have used (the extensional collapse of) the games model (Abramsky *et al.* 1995; Hyland and Ong 1994; Nickau 1996). There is little to choose between the two; Milner's model is, perhaps, a little more convenient as it is known to be cpo-enriched; the more pressing issue here is to settle the well-known open question as to whether the collapse of the games model is cpo-enriched – if so, it is isomorphic to Milner's.

After presenting the required background material in Section 2, we consider operational notions of totality in Section 3, showing that, for total terms, operational equivalence in ground total contexts corresponds to a natural hereditarily defined notion of total equivalence. After discussing semantical notions of totality in Section 4, we consider notions of total full abstraction in Section 5, principally showing relative full abstraction

in Corollary 5.1. We also give a notion that enables the extent to which models of PCF (or PCF^{++}) agree on totality to be compared. We investigate this notion in detail in Section 6; it turns out, for example, that disagreements arise already at level 2, as mentioned above.

While, as we have seen, $\mathscr{S}$ is not fully abstract for totality for PCF, $\mathscr{S}^o$ is; again, although $\mathscr{C}$ is not fully abstract for totality for PCF^{++}, $\mathscr{C}^o$ is. If one takes these points seriously, one is faced with the problem of finding a convenient method of working with the effective models, and, in particular, of finding a pleasant logic. This can be viewed as an additional argument for the approach of synthetic domain theory (Fiore *et al.* 1996), but perhaps a more modest version in which domains can still be considered as partially ordered sets, if convenient, and one wishes only to avoid the tedium of detailed considerations of syntax or the handling of indices of effective elements.

Some interesting open problems arise. It is not known whether $\mathscr{C}$ and $\mathscr{S}$ agree on totality at all levels, nor whether $\mathscr{C}^o$ and $\mathscr{S}^o$ do. However, if certain definability questions have a positive answer, they do. These questions concern whether, in various senses, PCF and PCF^{++} have the same expressive power for total functionals; a question of this kind was first raised by Cook (Cook 1989) and, later, also by Berger (Berger 1993). The two languages do not, as is well known, have the same expressive power for partial functionals, and so the questions have a strong independent interest. In turn, they are related to two other interesting open problems raised by John Longley. One concerns whether the Kleene–Kreisel total functionals coincide with a sequential analogue, and the other concerns the analogous question about Kreisel's hereditarily recursively continuous functionals.

It would be of interest to understand totality and recursive types. For example, given a model of the untyped λ-calculus, one might wish for a notion of totality such that an element is total iff it is when applied to any other total element; however, one then has the undesirable consequence that $(\lambda x.xx)(\lambda x.xx)$ is total. Again, in the context of games models there is a natural question, raised by Abramsky, concerning the relationship between the hereditarily-defined notion of totality and that of being a winning strategy. Finally, there is a need for a theory of totality for other kinds of programming languages – for example, consider higher-order imperative languages or languages for concurrency; operational notions are important and semantical frameworks may be employed other than variations on cpos. Evident questions of full abstraction arise for these languages, as do questions of suitable logics.

2. Preliminaries on PCF

Both PCF and PCF^{++} are applied typed λ-calculi (Barendregt 1984) with two *ground* types, o and ι (the *booleans* and the *natural numbers*). The constants of PCF are the *recursion* combinators $\mathrm{Y}_\sigma : (\sigma \to \sigma) \to \sigma$, one for each type σ, together with boolean and arithmetical constants, as follows

$$\begin{array}{rclrcl} \mathbf{t}, \mathbf{f} & : & o, & \supset_\gamma & : & o \to \gamma \to \gamma, \\ 0 & : & \iota, & +1, -1 & : & \iota \to \iota, \\ \mathrm{Z} & : & \iota \to o, & \Omega_\gamma & : & \gamma. \end{array}$$

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

Here, and below, we reserve γ to range over ground types. The constants of PCF^{++} are those of PCF together with **por** $: o \to o \to o$ and $\exists : (\iota \to o) \to o$. Both of these languages appear, in essence, in Plotkin (1997). The formulations here differ principally in the use of a 'parallel-or' constant rather than a parallel conditional, for which see Stoughton (1991); it is also convenient to include constants Ω_γ for 'undefined' at ground types.

The *operational semantics* of these languages may be given by rules specifying a binary relation $M \Rightarrow c$ of *evaluation* of programs to canonical form – see, for example, Plotkin (1997) and Mitchell (1996). Here a *program* is a closed term of ground type, the *canonical forms* of type o are **t** and **f**, and those of type ι have the form $(+1)^n 0$ (for $n \geqslant 0$).

We employ the usual conventions of the typed λ-calculus, such as omitting parentheses or type superscripts on variables, as convenient. Other conventions are also useful. We use $\alpha, \beta, \sigma, \tau$ to range over types. We use $A, B, \ldots, L, M \ldots$ to range over terms, but reserve C for contexts (see below). Vector notation avoids much indexing. Thus, $\vec{\alpha}$ stands for a list $\alpha_1, \ldots, \alpha_m$ of types (where $m = |\vec{\alpha}|$ is the length of $\vec{\alpha}$), $\vec{A}$ stands for a list $A_1, \ldots, A_m$ of terms, and $\vec{x}$ for a list $x_1^{\alpha_1}, \ldots, x_m^{\alpha_m}$ of variables (which, where necessary, we implicitly assume to be all distinct). Now we have available several very useful abbreviations. First $\vec{\alpha} \to \gamma$ abbreviates $\alpha_1 \to \ldots \to \alpha_m \to \gamma$; every type can be written uniquely in this form. Next, $\lambda \vec{x}.\, M$ and $M\vec{A}$ abbreviate $\lambda x_1^{\alpha_1}. \ldots .\lambda x_m^{\alpha_m}.\, M$ and $MA_1 \ldots A_m$, respectively. We write $\vec{A} : \vec{\alpha}$ to mean that $A_1 : \alpha_1, \ldots, A_m : \alpha_m$ (where it is assumed that $m = |\vec{A}| = |\vec{\alpha}|$), and $M[\vec{x} := \vec{A}]$ stands for the simultaneous substitution of $A_1, \ldots, A_m$ for $x_1, \ldots, x_m$ in M (and it is assumed that $\vec{x} : \vec{\alpha}$ and $\vec{A} : \vec{\alpha}$ for some $\vec{\alpha}$). We employ definitions by recursion, saying that $M : \vec{\alpha} \to \sigma$ is recursively defined by $M\vec{x} = \ldots M \ldots M \ldots$, and meaning that M is defined to be $\mathrm{Y}_{\vec{\alpha} \to \sigma}(\lambda f^{\vec{\alpha} \to \sigma}.\lambda \vec{x}. \ldots f \ldots f \ldots)$.

We give a somewhat more careful treatment of contexts than is usual; our definitions are intended to apply equally to PCF or PCF^{++}. The *σ-contexts* are ranged over by C, and are given by the abstract grammar

$$C ::= \ \cdot_\sigma | \lambda x^\alpha.\, C | CN | MC$$

and we write $C[M]$ for the result of replacing $\cdot_\sigma$ in C by M. We write $C : \tau$ if that follows from the evident adaptation of the usual typing rules to contexts, taking $\cdot_\sigma : \sigma$. Note that if $C : \tau$ is a σ-context and $M : \sigma$, then $C[M] : \tau$. We say that C *binds* x^α if that follows from the following rules

— $\lambda x^\alpha.\, C$ binds x^α,

— $\lambda y^\beta.\, C$, CN and MC all bind x^α if C does (where $y^\beta \neq x^\alpha$).

Two terms $M, N : \sigma$ are *operationally inequivalent*, written $M \leqslant_{\mathrm{op}} N$ iff for every σ-context $C : \gamma$ binding all the free variables of M and N, if $C[M] \Rightarrow c$, then $C[N] \Rightarrow c$ (for any canonical $c : \gamma$). We write $=_{\mathrm{op}}$ for the corresponding equivalence relation; note that it contains $=_{\beta\eta}$.

There are some useful boolean and arithmetical abbreviations. The *pure types* $\underline{n}$ ($n \geqslant 0$) are defined by: $\underline{0} = \iota$ and $\underline{n+1} = \underline{n} \to \iota$. Conditionals $\supset_\gamma BMN$ are written as **if** B **then** M **else** N, using multifix notation; postfix notation is employed for $+1$ and -1; **and** ('left-sequential and') abbreviates $\lambda x^o.\lambda y^o.$ **if** x **then** y **else f**, **or** ('left-sequential or') is

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

defined similarly, and both are written using infix notation; finally, we use $\neg$ ('negation') as an abbreviation for the term $\lambda x^o.$ **if** x **then f else t**.

Set-theoretical models of applied λ-calculi are based on *applicative structures* A; these are type-indexed families of sets A_σ together with *application functions* $App_{\sigma,\tau} : A_{\sigma\to\tau} \times A_\sigma \to A_\tau$. Such a structure is *extensional* if

$$\forall f, g \in A_{\sigma\to\tau}. f = g \;\equiv\; \forall x \in A_\sigma. f \cdot x = g \cdot x$$

where $f \cdot x$ abbreviates $App_{\sigma,\tau}(f, x)$. It is a *type frame* if $A_{\sigma\to\tau} \subset A_\tau^{A_\sigma}$ and application is standard function application. Every extensional applicative structure is evidently isomorphic to a unique type frame with the same sets at ground type. An *environment* is a type-respecting map of variables to the union of the A_σ. A *model* $\mathscr{A}$ *over an applicative structure* A consists of A together with an assignment $\mathscr{A}[\![M]\!](\rho)$ of elements of A_σ to terms $M : \sigma$ (for every σ and any ρ); this assignment is required to satisfy certain standard conditions; such $\mathscr{A}$ model the typed $\lambda\beta$-calculus. In the case of extensional structures the assignment is uniquely determined by its value on constants, and such $\mathscr{A}$ model the typed $\lambda\beta\eta$-calculus. Vector notation will again prove useful; for example we will write $\vec{a} \in A_{\vec{\alpha}}$ and $a \cdot \vec{b}$ with evident meanings. For closed terms M, the element $a = \mathscr{A}[\![M]\!](\rho)$ is independent of ρ and we say that M *defines* a; we may omit ρ writing $\mathscr{A}[\![M]\!]$ (or even just M).

A *logical* (or *hereditarily defined*) binary relation between applicative structures A and B is a family R of relations $R_\sigma \subset A_\sigma \times B_\sigma$ such that for all f in $A_{\sigma\to\tau}$ and g in $B_{\sigma\to\tau}$

$$R_{\sigma\to\tau}(f, g) \equiv \forall x \in A_\sigma, y \in B_\sigma. R_\sigma(x, y) \supset R_\tau(f \cdot x, g \cdot y).$$

Given models $\mathscr{A}$ and $\mathscr{B}$, over A and B, the *logical relations* lemma states that for all terms $M : \sigma$ if $R(\rho, \rho')$ holds (in the sense that $R_\alpha(\rho(x^\alpha), \rho'(x^\alpha))$ holds for all variables x^α), then it follows that $R_\sigma(\mathscr{A}[\![M]\!](\rho), \mathscr{B}[\![M]\!](\rho'))$ holds, provided it does for all constants. All of this generalises to relations of arbitrary degree. In the binary case, if a relation R over A is a partial equivalence relation at base types, it is at all types, that is 'being a partial equivalence relation' is an *inherited* property of binary logical relations. In this case we can define an extensional applicative structure A/R where $(A/R)_\sigma$ consists of the R_σ equivalence classes and with the induced application functions: $[f] \cdot [x] = [fx]$; in the case where R is built up from the equality relations at ground types, A/R is said to be the *extensional collapse* or the *Gandy hull* of A. If $\mathscr{A}$ is a model over A and the denotations of all constants are in the field of the relevant partial equivalence relations, the unique model $\mathscr{A}/R$ over A/R exists, and, for closed terms M, we have that $(\mathscr{A}/R)[\![M]\!] = [\mathscr{A}[\![M]\!]]$. See Mitchell (1996) for a more detailed treatment of these matters.

The Scott model $\mathscr{C}$ of PCF^{++} – see, *e.g.*, Plotkin (1997) – is based on the type frame C_σ of all continuous functions over the flat cpos $\boldsymbol{T}_\perp (= \{tt, ff, \perp\})$ and $\boldsymbol{N}_\perp$ of the booleans and the natural numbers. (See, *e.g.*, Winskel (1993), Abramsky and Jung (1994) and Stoltenberg-Hansen *et al.* (1994) for domain-theoretic background). Milner gave his model $\mathscr{S}$ of PCF by a term model construction (Milner 1977); we prefer to consider S_σ to be the isomorphic type frame with the same ground types as $\mathscr{C}$. In both cases the applicative structures can be order-enriched in that they can naturally be considered as a family of partially-ordered structures with monotonic application functions. They are both *order-extensional* in that,

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

for $\mathscr{C}$, for example,

$$\forall f, g \in A_{\sigma\to\tau}.\, f \leqslant g \;\equiv\; \forall x \in A_\sigma.\, f \cdot x \leqslant g \cdot x.$$

More is true: in both cases the structures are families of ω-algebraic cpos with continuous application functions; further, the finite elements are definable by closed terms containing no occurrence of recursion combinators or $\exists$ (or **por** in the case of $\mathscr{S}$). Both models relate well to the operational semantics, in that they are *adequate* and *inequationally fully abstract*. For PCF^{++} and $\mathscr{C}$ adequacy is that for all programs $M : \gamma$

$$M \Rightarrow c \;\equiv\; \mathscr{C}[\![M]\!] = \mathscr{C}[\![c]\!]$$

(we omit environments for closed terms), and inequational full abstraction is that for all terms $M, N : \sigma$

$$M \leqslant_{\mathrm{op}} N \;\equiv\; \forall \rho.\, \mathscr{C}[\![M]\!](\rho) \leqslant \mathscr{C}[\![N]\!](\rho).$$

The definitions for PCF and $\mathscr{S}$ are similar.

The models $\mathscr{C}^o$ and $\mathscr{S}^o$ are based on the sub-applicative structures C^o_σ and S^o_σ of elements definable by closed terms. The elements of the C^o can equivalently be seen as the effective elements of C – see Plotkin (1997); the same holds for $\mathscr{S}^o$ if we take the effective elements to be those given by recursive strategies in the sense of Abramsky *et al.* (1995). These structures are order-enriched, with the inherited partial orders, and order extensional (since all finite elements of the C_σ and the S_σ are definable). When we refer below to elements of these structures as finite, we mean when considered as elements of the corresponding super-structures. We have for all PCF^{++} terms M that

$$\mathscr{C}^o[\![M]\!](\rho) = \mathscr{C}[\![M]\!](\rho),$$

and similarly for $\mathscr{S}^o$ (the standard conditions are equational). It follows that both models are adequate and inequationally fully abstract.

We use the letters $\mathscr{D}$ and $\mathscr{E}$ to range over $\mathscr{C}$, $\mathscr{S}$, $\mathscr{C}^o$ and $\mathscr{S}^o$ (and D, E range over the corresponding applicative structures).

3. Operational notions of totality

We wish to formalise operational notions of totality, and of the equivalence of total terms in total contexts. We develop these for PCF, but all of the work goes through just as well for PCF^{++}, and we will assume that case too. First we extend the operational termination predicates on programs to *totality* ones on closed terms (at all types) by setting $M \Downarrow_{\sigma\to\tau}$ iff whenever $N \Downarrow_\sigma$ we have $MN \Downarrow_\tau$. These are then extended to open terms by setting $M \Downarrow_\sigma$ iff for every list $\vec{A} \Downarrow_{\vec{\alpha}}$ of closed total terms, $M[\vec{A}/\vec{x}] \Downarrow_\sigma$, where $\vec{x} : \vec{\alpha}$ is a list of the free variables of M. One easily sees that the totality predicates are closed under operational equivalence, and it follows that they are also closed under λ-abstraction, and indeed $M \Downarrow_\sigma$ iff $\lambda x^\alpha.\, M \Downarrow_{\alpha\to\sigma}$.

We are now in a position to define total contexts. Let $C : \tau$ be a closed σ-context. Then C is *total* iff for every total term $M \Downarrow_\sigma$, all of whose free variables it binds, $C[M] \Downarrow_\tau$. (There is a more general definition for open σ-contexts $C : \tau$, *viz* that whenever $M \Downarrow_\sigma$ we have

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

$C[M]\Downarrow_\tau$.) With this we can then easily define when terms are equivalent in total contexts. However, we are only interested in this notion for total terms, and in that case we have a characterisation in terms of a binary analogue of the totality predicate.

Let us define *(operational) total equivalence relations* $\approx_\sigma$ on closed terms of type σ by putting, at ground types, $M \approx_\gamma N$ iff M and N both terminate and evaluate to the same canonical form, and on other types, $M \approx_{\sigma\to\tau} N$ iff whenever $A \approx_\sigma B$ we have $MA \approx_\tau NB$; each $\approx_\sigma$ is a partial equivalence relation. In Section 4 we prove

$$M \approx_\sigma M \;\equiv\; M\Downarrow_\sigma, \tag{*}$$

so there is no ambiguity in the unary notion. It follows that

$$M \approx_{\sigma\to\tau} N \text{ iff for every closed } A\Downarrow_\sigma, MA \approx_\tau NA \tag{**}$$

(also see Section 4). We can again extend to open terms by a substitution process, setting $M \approx_\sigma N$ iff for all lists $\vec{A} : \vec{\alpha}$, $\vec{B} : \vec{\alpha}$ of closed terms, if $\vec{A} \approx_{\vec{\alpha}} \vec{B}$ (using an evident vector notation), then $M[\vec{A}/\vec{x}] \approx_\sigma N[\vec{B}/\vec{x}]$, where $\vec{x} : \vec{\alpha}$ is a list of the free variables of M and N. One easily sees that the relations are closed under $=_{\text{op}}$, and it follows that they are also closed under λ-abstraction: indeed $M \approx_\sigma N$ iff $\lambda x^\alpha.\, M \approx_{\alpha\to\sigma} \lambda x^\alpha.\, N$. One then has that (*) and (**) extend to open terms.

The following proposition shows that operational total equivalence is the same as (ground) operational total equivalence in total contexts. An immediate consequence is that for total terms, (ground) operational equivalence in total contexts is indeed the same as operational total equivalence.

Proposition 3.1. For any terms $M, N : \sigma$, the following are equivalent:

1 $M \approx_\sigma N$,
2 for all closed total σ-contexts, $C : \gamma$, binding all the free variables of M and N, $C[M] \approx_\gamma C[N]$.

Proof. Assume 1, and suppose $C : \gamma$ is a closed total σ-context binding all the free variables of M and N. Let $\vec{x} : \vec{\alpha}$ be a list of all the variables C binds. Then there is a closed term $B : (\vec{\alpha} \to \sigma) \to \gamma$ such that for any term $L : \sigma$, $C[L] =_{\beta\eta} B(\lambda\vec{x}.\, L)$. We claim that $B \Downarrow_{(\vec{\alpha}\to\sigma)\to\gamma}$. For suppose that $A \Downarrow_{\vec{\alpha}\to\sigma}$. Then $A\vec{x}\Downarrow_\sigma$, so, as C is total, $C[A\vec{x}]\Downarrow_\gamma$. But $C[A\vec{x}] =_{\beta\eta} B(\lambda\vec{x}.\, A\vec{x}) =_{\beta\eta} BA$, so we have established the claim. We therefore have by (*) that $B \approx_{(\vec{\alpha}\to\sigma)\to\gamma} B$; we also know that $\lambda\vec{x}.\, M \approx_{\vec{\alpha}\to\sigma} \lambda\vec{x}.\, N$, since $M \approx_\sigma N$. So we have that $C[M] =_{\beta\eta} B(\lambda\vec{x}.\, M) \approx_\gamma B(\lambda\vec{x}.\, N) =_{\beta\eta} C[N]$, concluding the proof that 2 holds.

Conversely, assume 2. Let $\vec{x} : \vec{\alpha}$ be a list of the free variables of M and N. It suffices to show that $\lambda\vec{x}.\, M \approx_{\vec{\alpha}\to\sigma} \lambda\vec{x}.\, N$. To this end we apply (**). Suppose $\sigma = \vec{\beta}\to\gamma$. Then taking closed $\vec{A}\Downarrow_{\vec{\alpha}}$ and $\vec{B}\Downarrow_{\vec{\beta}}$, we have to show that $(\lambda\vec{x}.\, M)\vec{A}\vec{B} \approx_\gamma (\lambda\vec{x}.\, N)\vec{A}\vec{B}$. Set $C : \gamma$ to be the total σ-context $(\lambda\vec{x}.\ \cdot_\sigma)\vec{A}\vec{B}$. Then, by 2, $(\lambda\vec{x}.\, M)\vec{A}\vec{B} \approx_\gamma (\lambda\vec{x}.\, N)\vec{A}\vec{B}$, and the conclusion follows. □

4. Semantical notions of totality

We wish to find semantical notions $\downarrow^{\mathscr{D}}_\sigma$, of totality, and $\sim^{\mathscr{D}}_\sigma$, of total equivalence, for each of our four models $\mathscr{D}$. Logical relations naturally suggest themselves; these are

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

uniquely determined for all our models by the requirement that the relations agree with the operational ones at ground types, so we put

$$x\downarrow_\gamma^{\mathscr{D}} \equiv\ x \neq \bot$$

and

$$x \sim_\gamma^{\mathscr{D}} y\ \equiv\ x = y \neq \bot$$

(note the use of postfix and infix notation). In the cases of $\mathscr{C}$ and $\mathscr{C}^o$, these predicates and relations have already appeared in Ershov (1974) and Ershov (1977).

All our D_σ's are partial orders with binary meets, and we can ask which properties of a totality predicate $\downarrow$ and relation $\sim$ to take as basic on such partial orders P. We consider these to be:

(1) for all x in P, $x\downarrow$ iff $x \sim x$,
(2) the relation $\sim$ is a partial equivalence relation and its equivalence classes are filters (*i.e.*, are upper-closed and closed under binary meets).

The first property ensures that the unary and binary notions of totality agree. The other properties are also reasonable candidates for general properties, except, perhaps, closure under meets; this is a rather useful technical requirement, which happens to hold in the models we consider. An equivalent formulation is that $\downarrow$ is upper-closed, $\sim$ is a partial equivalence relation and $x \sim y$ iff $(x \wedge y)\downarrow$; this formulation already appears in Normann's definition in Normann (1997a) of his category K_2 of domains and totality predicates.

Proposition 4.1. Let P be an applicative type structure, where each P is a partial order equipped with binary meets, and suppose that application is multiplicative in its first argument (*i.e.*, that $(f \wedge g) \cdot x = f \cdot x \wedge g \cdot x$ always holds). Let $\downarrow$ be a logical property on P and $\sim$ be a logical relation on it. Then, if $\downarrow_\gamma$ and $\sim_\gamma$ have properties (1) and (2) above for both ground types γ, so do every $\downarrow_\sigma$ and $\sim_\sigma$.

Proof. The $\sim_\sigma$ are certainly partial equivalence relations and it is easy to show, by induction on types, that their equivalence classes are upper-closed. For the rest it suffices to prove for all σ and x, y in P_σ that $x \sim_\sigma y\ \equiv\ (x \wedge y)\downarrow_\sigma$, which we do by induction on types. The ground case is immediate, by the above remarks.

For $\sigma \to \tau$, suppose first that $f \sim_{\sigma\to\tau} g$. Now assume that $a\downarrow_\sigma$. Then, by the induction hypothesis, $a \sim_\sigma a$, so $f \cdot a \sim_\tau g \cdot a$. But then, as $(f \wedge g) \cdot a = (f \cdot a \wedge g \cdot a)$ and, by the induction hypothesis, $(f \cdot a \wedge g \cdot a)\downarrow_\tau$, it follows that $(f \wedge g)\downarrow_{\sigma\to\tau}$.

Conversely, suppose that $(f \wedge g)\downarrow_{\sigma\to\tau}$, and assume $a \sim_\sigma b$. Then, by the induction hypothesis, $(a \wedge b)\downarrow_\sigma$, so $(f \wedge g) \cdot (a \wedge b)\downarrow_\tau$. Therefore, by another use of the induction hypothesis, $(f \wedge g) \cdot (a \wedge b) \sim_\tau (f \wedge g) \cdot (a \wedge b)$. It then follows from the upwards closure of total equivalence that $f \cdot a \sim_\tau g \cdot b$, showing $f \sim_{\sigma\to\tau} g$, as required. □

The idea of this proof appears already in Longo and Moggi (1984). Our development of totality and its elementary properties differs a little from that found elsewhere, where the unary notion is taken as primary. However, as the unary and binary notions are interdefinable, this is not a point of great mathematical significance. More interestingly, the connection between the two is usually made in terms of the order-theoretic notion of consistency rather than that of meets; for example, see Stoltenberg-Hansen *et al.* (1994,

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

p. 208), or below. This works well for the Scott model, but not the Milner model. For the latter there is another notion of consistency available, which is given below; an abstract treatment would seem to require taking it as extra structure.

The assumptions needed for Proposition 4.1 hold for the applicative structures underlying each of our models. They are evidently partially ordered. They also possess binary meets with the requisite properties. Indeed, for every type σ there is a PCF term M_σ, say, which, in each model, defines $\wedge_\sigma$; this is evident at ground types, and at higher types $\sigma = \vec{\alpha} \to \gamma$, as order-extensionality holds, one can set $M_\sigma = \lambda f^\sigma g^\sigma \vec{x}.\, M_\gamma(f\vec{x})(g\vec{x})$.

The relation with total functionals can be given in terms of the totality equivalence relations. That of the Kleene–Kreisel continuous functionals is isomorphic to $C/\sim^C$; that of Kreisel's hereditarily recursively continuous functionals is isomorphic to $C^o/\sim^{C^o}$ and also to HEO, the hereditarily extensional operations (Berger 1993; Ershov 1975; Ershov 1976; Ershov 1977; Gandy and Hyland 1977; Troelstra 1973). As already mentioned above, there are two interesting open problems here: whether $C/\sim^C$ and $S/\sim^S$ are isomorphic, and whether $C^o/\sim^{C^o}$ and $S^o/\sim^{S^o}$ are isomorphic (and see the discussion in Section 6 for further information).

Note that all constants of PCF^{++} (other than the Ω_γ, the Y_σ and $\exists$) define total elements in each of the models; it follows from the logical relations lemma that closed terms not involving those constants do so too. We can now see that a certain (well-known) *density* property holds in each of our models: that there is a total element above every finite element. This is because, as remarked above, in each of the models, every finite element is definable by some term (of PCF or PCF^{++} as appropriate) containing no occurrences of the Y_σ or $\exists$; replacing every occurrence of an Ω_γ by **t** or 0 (as appropriate) we obtain the required total element. This property of *definable* density already appears in Loader (1997), and, as remarked there, is implicit in previous proofs of density.

Proposition 4.1 implies a characterisation of total equivalence for total elements. Suppose that $x, y \downarrow_\sigma$, where $\sigma = \vec{\alpha} \to \gamma$. Then

$$x \sim_\sigma y \;\equiv\; \forall \vec{a} \downarrow_{\vec{\alpha}} .\, x \cdot \vec{a} = y \cdot \vec{a}.$$

The implication from left to right is obvious; the other direction is essentially just the extensionality of $P/\sim$. It then follows that, in all our models $\mathscr{D}$ we have $x \sim^{\mathscr{D}}_\sigma y$ iff for any γ and $f \downarrow^{\mathscr{D}}_{\sigma\to\gamma}$, $fx \sim^{\mathscr{D}}_\gamma fy$; that is, we can indeed identify $\sim^{\mathscr{D}}_\sigma$ as the semantical notion of equivalence in total ground contexts if we treat 'semantical contexts' simply as functions. The characterisation of total equivalence can be usefully strengthened for our $\mathscr{D}$.

Proposition 4.2. Suppose that $x, y \downarrow^{\mathscr{D}}_\sigma$, where $\sigma = \vec{\alpha} \to \gamma$. Then $x \sim^{\mathscr{D}}_\sigma y$ iff for all $\vec{a}$ definable in PCF without using any of the Ω_γ or the Y_σ, it holds that $x \cdot \vec{a} = y \cdot \vec{a}$.

Proof. Only the implication from right to left is in question. By the above remarks, it is enough to show that for all $\vec{a} \downarrow^{\mathscr{D}}_{\vec{\alpha}}$, $x \cdot \vec{a} = y \cdot \vec{a}$. If this is not the case, there are finite elements $\vec{b}$ such that $x \cdot \vec{b}$ and $y \cdot \vec{b}$ are distinct and total (as some $x \cdot \vec{a}$ and $y \cdot \vec{a}$ are with $\vec{a} \downarrow_{\vec{\alpha}}$). Now, as in the argument for density, we can increase the $\vec{b}$ to total $\vec{c}$, definable in PCF or PCF^{++} (as appropriate) without using Ω_γ, Y_σ or $\exists$. This contradicts the assumption in the cases of $\mathscr{S}$ and $\mathscr{S}^o$. In the cases of $\mathscr{C}$ and $\mathscr{C}^o$, the definitions may contain occurrences of **por**.

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

However, since there we have that **por** $\sim^{\mathscr{D}}_{o\to o\to o}$ **or**, we may replace all such occurrences of **por** by ones of **or**, and again obtain a contradiction. □

The models provide some interesting examples. Left-sequential or is always total. However in $\mathscr{C}$ and $\mathscr{C}^o$ it is not maximal, as **por** is greater. On the other hand, in $\mathscr{S}$ and $\mathscr{S}^o$ it is maximal, but not maximum, in its equivalence class, as right sequential or is incomparable (and also maximal). In those models, another version is total but not maximal, 'strict or,' defined by: $\lambda x^o.\lambda y^o.$ **if** x **then** (**if** y **then t else t**) **else** y. Conversely, $\exists$ defines a maximal element of both $\mathscr{C}$ and $\mathscr{C}^o$, but one that is total in neither. In the cases of $\mathscr{S}$ and $\mathscr{S}^o$, there is a related term $\exists_s : (\iota \to o) \to o$, which can be shown to define a maximal but non-total element. (The term is $\lambda f^{\iota\to o}.\, f(Sf0)$, where the term S is recursively defined by: $Sfx =$ **if** fx **then** x **else** $Sf(x{+}1)$. Note that $\exists_s f$ is ff if $f\bot$ is, and is tt if fn is tt for some $n \geqslant 0$, and fm is ff for $m < n$.) Neither $\exists$ nor $\exists_s$ are finite, and this is necessarily so since, by density, all finite maximal elements are total.

We may also learn a little more about the filters of total elements of $\mathscr{C}$ and$\mathscr{C}^o$. First, in all four models $\mathscr{D}$, if $x \sim^{\mathscr{D}}_{\sigma} y$, then x and y are consistent in the sense that for any $\vec{a} \in D_{\vec{\alpha}}$ (where $\sigma = \vec{\alpha}\to\gamma$) $x \cdot \vec{a}$ and $y \cdot \vec{a}$ have an upper bound. If this were not the case, there would be a counterexample with all the a_i finite, so, by density, there would be a counterexample with all the a_i total, contradicting the assumption that $x \sim^{\mathscr{D}}_{\sigma} y$. In the cases of $\mathscr{C}$ and $\mathscr{C}^o$, it follows that x and y are consistent in the order-theoretic sense (*i.e.*, they have an upper bound), so the filters there are directed; we also then have that for any $x, y \downarrow^{\mathscr{D}}_{\sigma}$ that $x \sim^{\mathscr{D}}_{\sigma} y$ iff x and y are consistent. In the case of $\mathscr{C}$, it follows further that filters have a greatest element, as one can take their supremum. This is not true for $\mathscr{C}^o$: consider the least function f in $C^o_{\iota\to\iota\to o}$ such that fmn is tt, if the mth Turing machine terminates in n steps on the empty input, and is ff otherwise ($m, n \neq \bot$). This is total but can have no effective maximal extension.

It is also interesting to note that in all four models, filters need have no minimal element. An example is provided by the filter containing the 'constantly true' functional $\lambda f^{\iota\to o}.\,\mathbf{t}$. Of course, for types with finitely many elements, that is those σ containing no occurrence of ι, filters have least elements as well as maximal ones, the latter of which, in the cases of $\mathscr{C}$ and $\mathscr{C}^o$, are in fact maximum.

5. Full abstraction for totality

In order to compare operational and denotational notions, some notation is convenient. We write $M \downarrow^{\mathscr{D}}_{\sigma}$ for a term $M : \sigma$ of PCF to mean that $\mathscr{D}[\![M]\!](\rho) \downarrow^{\mathscr{D}}_{\sigma}$ for every total environment ρ (where ρ is total iff for every x^α, $\rho(x^\alpha) \downarrow^{\mathscr{D}}_{\alpha}$). Similarly, we write $M \sim^{\mathscr{D}}_{\sigma} N$ for PCF terms $M, N : \sigma$ to mean that $\mathscr{D}[\![M]\!](\rho) \sim^{\mathscr{D}}_{\sigma} \mathscr{D}[\![N]\!](\rho')$ for every $\rho \sim^{\mathscr{D}} \rho'$. It is straightforward to show that $M \downarrow^{\mathscr{D}}_{\sigma}$ iff $(\lambda x^\alpha.\, M) \downarrow^{\mathscr{D}}_{\alpha\to\sigma}$ and that $M \sim^{\mathscr{D}}_{\sigma} N$ iff $(\lambda x^\alpha.\, M) \sim^{\mathscr{D}}_{\alpha\to\sigma} (\lambda x^\alpha.\, N)$. Now, following the discussion in the introduction, we say that a model $\mathscr{D}$ is *fully abstract for totality for PCF* iff for all σ and $M : \sigma$, $M \Downarrow_\sigma$ holds iff $M \downarrow^{\mathscr{D}}_{\sigma}$ does (for whichever language is under consideration). We define *full abstraction for total equivalence for PCF* similarly. Similar definitions can be made for PCF^{++}.

We have, perhaps unsurprisingly, the following proposition.

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

Proposition 5.1.

1 $\mathscr{S}^o$ is fully abstract for both totality and total equivalence for PCF.
2 $\mathscr{C}^o$ is fully abstract for both totality and total equivalence for PCF^{++}.

Proof. A straightforward induction on types σ proves that $M \Downarrow_\sigma$ iff $M \downarrow_\sigma^{\mathscr{S}^o}$ for closed PCF terms M: the base case is an immediate consequence of adequacy; at higher types one uses the fact that all elements are definable. The result then follows for open terms via the above remarks on λ-abstraction. The proof that $M \approx_\sigma N$ iff $M \sim_\sigma^{\mathscr{S}^o} N$ is similar, as, then, are the proofs for PCF^{++}. □

With this we have operational analogues of the results of the previous section. In particular, statements (*) and (**) of Section 2 follow; it is also worth noting that Proposition 4.2 even yields a strengthening of (**).

We now follow a policy of proving results on totality for pairs of models $\mathscr{D}$, $\mathscr{E}$, and then deducing corresponding results on full abstraction via Proposition 5.1.

Proposition 5.2. For any PCF terms $M, N : \sigma$, if $M \downarrow_\sigma^{\mathscr{D}}$, $M \downarrow_\sigma^{\mathscr{E}}$, $N \downarrow_\sigma^{\mathscr{D}}$ and $N \downarrow_\sigma^{\mathscr{E}}$, then $M \sim_\sigma^{\mathscr{D}} N$ holds iff $M \sim_\sigma^{\mathscr{E}} N$ does. The same holds for PCF^{++} and $\mathscr{C}^o$ and $\mathscr{C}$.

Proof. We can reduce to the case where M and N are closed by the previous remarks on λ-abstraction. The result then follows immediately from Proposition 4.2 and adequacy. □

Corollary 5.1. (Relative Full Abstraction.) Let $M, N : \sigma$ be PCF terms, and suppose that $M \Downarrow_\sigma$, $M \downarrow_\sigma^{\mathscr{C}}$, $N \Downarrow_\sigma$ and $N \downarrow_\sigma^{\mathscr{C}}$. Then $M \approx_\sigma N$ iff $M \sim_\sigma^{\mathscr{C}} N$.

Thus, as long as there is no ambiguity concerning totality, Scott's model is indeed fully abstract for total equivalence. The corresponding results hold for PCF and $\mathscr{C}^o$ or $\mathscr{S}$, and for PCF^{++} and $\mathscr{C}$. In Loader (1997), Loader proves an analogous result for a strongly normalising calculus with a richer type system, including certain inductive types. He considers an operational congruence and two models; one of these models is similar to Girard's qualitative domains and the other uses the category **PER** of partial equivalence relations on the natural numbers. His proof technique for the former model is very much the same as ours, and employs a lemma closely related to Proposition 4.2.

We now give a finer analysis of the full abstraction properties, dividing them into restricted implications, parameterised by types. Full abstraction for totality breaks down into two implications: the first is that for all σ and $M : \sigma$, if $M \Downarrow_\sigma$ holds, then so does $M \downarrow_\sigma^{\mathscr{D}}$; the other is its converse. By the previous remarks on totality and λ-abstraction, each of these implications is equivalent to its restriction to closed terms. A similar analysis can be made of full abstraction for total equivalence. It turns out that the unary and binary implications over closed terms are even equivalent at each type. Indeed we have the following result of this kind for pairs of models.

Proposition 5.3. Let $\mathscr{D}$, $\mathscr{E}$ be any two of $\mathscr{C}$, $\mathscr{C}^o$, $\mathscr{S}$ or $\mathscr{S}^o$. Then the following two statements are equivalent:

1_σ For all closed PCF terms $M : \sigma$, if $M \downarrow_\sigma^{\mathscr{D}}$ then $M \downarrow_\sigma^{\mathscr{E}}$.
2_σ For all closed PCF terms $M, N : \sigma$, if $M \sim_\sigma^{\mathscr{D}} N$ then $M \sim_\sigma^{\mathscr{E}} N$.

The same holds for PCF^{++} and $\mathscr{C}$ and $\mathscr{C}^o$.

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

Proof. Assume 1, and suppose $M \sim^{\mathscr{D}}_{\sigma} N$. Then $M \downarrow^{\mathscr{D}}_{\sigma}$ and $N \downarrow^{\mathscr{D}}_{\sigma}$, so, by 1, $M \downarrow^{\mathscr{E}}_{\sigma}$ and $N \downarrow^{\mathscr{E}}_{\sigma}$. Therefore, by Proposition 5.2, $M \sim^{\mathscr{E}}_{\sigma} N$. That 2 implies 1 is immediate, as, if $M \downarrow^{\mathscr{D}}_{\sigma}$, then $M \sim^{\mathscr{D}}_{\sigma} M$, and so on. The variation for PCF^{++} is proved similarly. □

We now have the basis for a fine-grained comparison of models. Let us say, of two given models $\mathscr{D}$ and $\mathscr{E}$, that $\mathscr{D}$ is *σ-PCF-correct* for $\mathscr{E}$ (for totality) if 1_σ of the proposition holds (or, equivalently, 2_σ); we say it is *PCF-correct* if that holds for all σ, and that it is *l-PCF-correct* if that holds for all σ of order l (and we omit 'PCF' when it can be understood from the context). Then, for PCF, full abstraction for totality of $\mathscr{D}$ is just that $\mathscr{S}^o$ and $\mathscr{D}$ are PCF-correct for each other.

Retracts provide a convenient tool to relate correctness at different types. Fixing two models $\mathscr{D}$ and $\mathscr{E}$, write $\sigma \lhd_t^{\mathscr{D},\mathscr{E}} \tau$ to mean that there are closed PCF terms $F : \sigma \to \tau$ and $G : \tau \to \sigma$, each total in both $\mathscr{D}$ and $\mathscr{E}$ and such that $\lambda x^\sigma.\, G(Fx) \sim^{\mathscr{E}}_{\sigma\to\sigma} \lambda x^\sigma.\, x$. Clearly, if $\sigma \lhd_t \tau \lhd_t \gamma$ (omitting superscripts), then $\sigma \lhd_t \gamma$, and also if $\sigma \lhd_t \tau$ and $\sigma' \lhd_t \tau'$, then $(\sigma \to \sigma') \lhd_t (\tau \to \tau')$; it follows that if $l \leqslant m$, then $\underline{l} \lhd_t \underline{m}$. It is also not hard to show that if σ has level l, then $\sigma \lhd_t \underline{l}$. (Given the previous remarks, this follows once we know that $(\underline{l} \to \underline{l+1}) \lhd_t \underline{l+1}$. This is shown using 'total pairing and projection combinators at level l,' by which we mean terms $\mathrm{Pair} : \underline{l} \to \underline{l} \to \underline{l}$ and $\mathrm{Fst}, \mathrm{Snd} : \underline{l} \to \underline{l}$ total in any of our models $\mathscr{E}$ and such that $\lambda x^{\underline{l}}.y^{\underline{l}}.\, \mathrm{Fst}\,(\mathrm{Pair}\, xy) \sim^{\mathscr{E}}_{\underline{l}\to\underline{l}\to\underline{l}} \lambda x^{\underline{l}}.y^{\underline{l}}.\, x$, and similarly for Snd; such terms are straightforwardly defined in PCF.)

What makes this relation useful to us is that if $\sigma \lhd_t \tau$ and $\mathscr{D}$ is τ-PCF-correct for $\mathscr{E}$, then it is also σ-PCF-correct for $\mathscr{E}$. It follows from the discussion that if $l \leqslant m$ and $\mathscr{D}$ is m-PCF-correct for $\mathscr{E}$, then it is also l-PCF-correct, and that to show correctness for all types of a given level l it is enough to consider $\underline{l}$.

Similar definitions and remarks obtain for PCF^{++}.

6. Comparing models

We now consider to what extent our four models agree on totality for PCF, or for PCF^{++}. Some (anonymous) logical relations will prove helpful: namely those induced between any two of the models $\mathscr{D}, \mathscr{E}$ by the equality relation at ground types. All constants are related; this is easy to see for all of them except, perhaps the fixed-point combinator. For this one notes that in any of the models, $\mathscr{D}$, one has $(\mathrm{Y}_\sigma \cdot F) \cdot \vec{x} \downarrow^{\mathscr{D}}_{\gamma}$ iff for some k, $(\mathrm{Y}^{(k)}_\sigma \cdot F) \cdot \vec{x} \downarrow^{\mathscr{D}}_{\gamma}$, where $\mathrm{Y}^{(k)}_\sigma$ is $\lambda F^{\sigma\to\sigma}.\, F^k(\lambda \vec{x}.\, \Omega_\gamma)$ (where $\sigma = \vec{\alpha} \to \gamma$ and $\vec{x} \in D_{\vec{\alpha}}$). Because of the definability of finite elements, one easily sees that the logical relation between C^o and C is equality at every type, as is that between S^o and S.

At ground types γ, the D_γ are all the same, as are the totality predicates, and each logical relation is equality. At level 1 the types are related by the following inclusions

$$\begin{array}{ccc} C^o_\sigma & \subset & C_\sigma \\ \cup & & \cup \\ S^o_\sigma & \subset & S_\sigma \end{array}$$

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

Furthermore, for $\sigma = \underline{1}$, we even have equalities, $C^o_{\underline{1}} = S^o_{\underline{1}}$ and $C_{\underline{1}} = S_{\underline{1}}$. At this type, the models agree on the totality of their common elements (here, functions), and each logical relation is equality. It follows that all four models are 1-PCF-correct for each other, and that $\mathscr{C}$ and $\mathscr{C}^o$ are 1-PCF^{++}-correct for each other. For example, for PCF we have, for any closed $M : \underline{1}$, that $\mathscr{D}[\![M]\!] = \mathscr{E}[\![M]\!]$ by the logical relations lemma and so, by the remark on totality, $M\downarrow^{\mathscr{D}}_{\underline{1}}$ iff $M\downarrow^{\mathscr{E}}_{\underline{1}}$.

At level 2, we have the following relations for the pure type $\underline{2}$

$$\begin{array}{ccc} C^o_{\underline{2}} & \subset & C_{\underline{2}} \\ \cup & & \cup \\ S^o_{\underline{2}} & \subset & S_{\underline{2}} \end{array}$$

At this type each logical relation is (again) equality, because of the definability of finite functions. Furthermore, $\mathscr{C}$ and $\mathscr{S}$ agree on the totality of common elements of $S_{\underline{2}}$ (here, functionals), so they are 2-PCF-correct for each other; the same holds for $\mathscr{C}^o$ and $\mathscr{S}^o$.

Any functional in $C^o_{\underline{2}}$ that is total in $C_{\underline{2}}$ is total in $C^o_{\underline{2}}$, so $\mathscr{C}$ is 2-PCF^{++}-correct for $\mathscr{C}^o$; similarly, $\mathscr{S}$ is 2-PCF-correct for $\mathscr{S}^o$. However, the converses do not hold. The counterexamples are provided by using Kleene's singular tree $\mathscr{K}$. This is a recursive prefix-closed set of finite binary sequences containing arbitrarily long finite sequences but no *recursive* infinite path. To construct an explicit term, let $\lceil - \rceil$ be an effective coding of finite binary sequences as natural numbers with $\lceil \varepsilon \rceil = 0$, and let $C : \iota \to \iota \to \iota$ and $T : \iota \to o$ be closed PCF terms coding concatenation and $\mathscr{K}$, in that $C \cdot \lceil s \rceil \cdot \lceil s' \rceil = \lceil ss' \rceil$ and $T \cdot \lceil s \rceil = tt$ (if $s \in \mathscr{K}$) and $= ff$ otherwise. Now $A : \iota \to \iota \to (\iota \to o) \to o$ is recursively defined by

$$Awmf = \textbf{if } Tw \textbf{ then } A(Cw(fm))(m{+}1)f \textbf{ else f},$$

and we define Kl to be $A\lceil\varepsilon\rceil 0 : \tau$, where $\tau = (\iota \to o) \to o$ is the 'binary tree' type. Then, regarding total f in $C_{\iota \to o}$ as infinite binary sequences, $\mathscr{C}[\![\mathrm{Kl}]\!] \cdot f$ is $\perp$ if f is a path in $\mathscr{K}$ and is ff otherwise. Thus Kl is total in $\mathscr{C}^o$ but not in $\mathscr{C}$. We therefore have that $\mathscr{C}^o$ is not 2-PCF^{++}-correct for $\mathscr{C}$. Furthermore, since all the terms are in PCF, neither is it 2-PCF-correct for $\mathscr{C}$. Finally, we also have that $\mathscr{S}^o$ is not 2-PCF-correct for $\mathscr{S}$. Consequently, *Scott's model is not fully abstract for totality for either one of PCF or PCF^{++}* and, moreover, *Milner's model is not fully abstract for totality for PCF*. The role of Kleene's singular tree in distinguishing between the continuous and effective models has already been pointed out by Gandy and Hyland in Gandy and Hyland (1977). It would be interesting to find models, other than the effective ones, that are fully abstract for totality for PCF and PCF^{++}.

Let us carry our considerations as far as level 3. As $S_{\underline{2}} \subset C_{\underline{2}}$, with the inclusion respecting totality, we can see that $\mathscr{C}$ is 3-PCF-correct for $\mathscr{S}$. For, suppose that $M : \underline{3}$ is a closed PCF term such that $M\downarrow^{\mathscr{C}}_{\underline{3}}$, in order to show that $M\downarrow^{\mathscr{S}}_{\underline{3}}$. Choose $F\downarrow^{\mathscr{S}}_{\underline{2}}$. Then $F\downarrow^{\mathscr{C}}_{\underline{2}}$, so $\mathscr{C}[\![M]\!](F)\downarrow^{\mathscr{C}}_{\underline{0}}$. But by the logical relations lemma, $\mathscr{S}[\![M]\!](F) = \mathscr{C}[\![M]\!](F)$, so $\mathscr{S}[\![M]\!](F)\downarrow^{\mathscr{S}}_{\underline{0}}$.

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

Similarly, $\mathscr{C}^o$ is 3-PCF-correct for $\mathscr{S}^o$. It turns out that the converses hold too, but first we need a lemma.

Lemma 6.1. For every total element G of $C_{\underline{2}}$ there is an F in $S_{\underline{2}}$ such that $F \leqslant G$ and $F \sim^{\mathscr{C}}_{\underline{2}} G$. The same holds for $\mathscr{C}^o$ and $\mathscr{S}^o$.

Proof. The functional G has the form $\bigvee_{i\geqslant 0} a_i \Rightarrow m_i$, where the a_i are finite functions in $C_{\underline{1}}$ and the m_i are natural numbers, using the notation of Plotkin (1997). We can assume without loss of generality that the a_i are all strict (for, if they are not, we can omit all the non-strict ones from the lub, obtaining a G' such that $G \geqslant G' \sim^{\mathscr{C}}_{\underline{2}} G$).

Now, for every strict finite function a in $C_{\underline{1}}$ there is a closed PCF term $D_a : \underline{1} \to o$ such that $\mathscr{C}[\![D_a]\!]f = tt$ (if $f \geqslant a$) and $= ff$ (if f and a are inconsistent). Define PCF terms $M_i : \underline{2}$ by $M_0 = \lambda f^{\underline{1}}.\Omega_\iota$ and $M_{i+1} = \lambda f^{\underline{1}}.$ **if** $D_{a_i} f$ **then** m_i **else** $M_i f$.

Then we can take $F = \bigvee_{i\geqslant 0} \mathscr{C}[\![M_i]\!]$. This is because, first, $F \in S_{\underline{2}}$ as $\mathscr{C}[\![M_i]\!] = \mathscr{S}[\![M_i]\!]$, by the logical relations lemma, and $\mathscr{S}[\![M_i]\!]$ is increasing and the inclusion $S_{\underline{2}} \hookrightarrow C_{\underline{2}}$ is continuous. Next we have that $F \leqslant G$ as $\mathscr{C}[\![M_i]\!] \leqslant G$. Finally, let f in $C_{\underline{1}}$ be total. Then for any strict a either $f \geqslant a$ or f and a are inconsistent. But, since $f \geqslant$ some a_i (as $Gf \downarrow^{\mathscr{C}}_i$), we therefore have that $Ff \downarrow^{\mathscr{C}}_i$. So F is total, and it follows that $F \sim^{\mathscr{C}}_{\underline{2}} G$. The proof for $\mathscr{C}^o$ and $\mathscr{S}^o$ (the 'effective case') is a straightforward adaptation of that for $\mathscr{C}$ and $\mathscr{S}$ (the 'continuous case'). □

With this one can show that $\mathscr{S}$ is 3-PCF-correct for $\mathscr{C}$. Suppose that $M : \underline{3}$ is a closed PCF term such that $M \downarrow^{\mathscr{S}}_{\underline{3}}$, and choose $G \downarrow^{\mathscr{C}}_{\underline{2}}$. Then, by the lemma, there is an $F \leqslant G$ in $S_{\underline{2}}$ such that $F \sim^{\mathscr{C}}_{\underline{2}} G$. It follows that $F \downarrow^{\mathscr{C}}_{\underline{2}}$, so $F \downarrow^{\mathscr{S}}_{\underline{2}}$, by the remarks made above. Therefore, as $M \downarrow^{\mathscr{S}}_{\underline{3}}$, we have that $\mathscr{S}[\![M]\!]F \downarrow^{\mathscr{S}}_{\underline{0}}$. Using the logical relation between $\mathscr{S}$ and $\mathscr{C}$, we then have that $\mathscr{C}[\![M]\!]F \downarrow^{\mathscr{C}}_{\underline{0}}$, and since $F \leqslant G$, we obtain $\mathscr{C}[\![M]\!]G \downarrow^{\mathscr{C}}_{\underline{0}}$, concluding the proof that $M \downarrow^{\mathscr{C}}_{\underline{3}}$. In the same way, we can show that $\mathscr{S}^o$ is 3-PCF-correct for $\mathscr{C}^o$.

From the previous discussions we know that $\mathscr{C}^o$ is not 3-PCF-correct for $\mathscr{C}$ and neither is $\mathscr{S}^o$ for $\mathscr{S}$, as the failures already occur at level 2. We now show that the level 2 failures also result in the failures of the converses at level 3.

To any F in C_τ, still with $\tau = (\iota \to o) \to o$, we can associate the tree of 'non-past secured binary sequences' $\{w \in \boldsymbol{T}^* | F\overline{w} = \bot\}$. Here, for any binary sequence $w = b_0 \ldots b_{n-1}$, $\overline{w}$ stands for the function $\{(0 \Rightarrow b_0) \vee \ldots \vee (n-1 \Rightarrow b_{n-1})\}$. Clearly, F is total iff its associated tree has no infinite branches, *i.e.*, by König's lemma, iff it is finite. We now define a term $\Delta : \tau \to o$, which, intuitively, performs a depth-first search of such trees, terminating iff they are finite. It is defined using a recursively defined term $B_1 : \tau \to (\iota \to o)$, which, intuitively, yields a branch that keeps going right as long as the corresponding left subtree is finite. We need terms $\mathrm{Left}, \mathrm{Right} : \tau \to \tau$ for finding left and right subtrees. The first is defined to be $\lambda T, f.\, T(\lambda m.$ **if** Zm **then t else** $f(m{-}1))$ and the second similarly (we associate 'left-branching' with 'tt'). A 'definedness' term $\partial : o \to o$ is also useful, and is defined to be $\lambda b^o.$ **if** b **then t else t**. Now B_1 is recursively defined by

$$\begin{array}{lcll} B_1(T) & = & \lambda m^\iota. & \textbf{if} \quad \partial(T(B_1(\mathrm{Left}\ T))) \\ & & & \textbf{then (if } Zm \textbf{ then f else } B_1(\mathrm{Right}\ T)(m{-}1)) \\ & & & \textbf{else} \quad \Omega_o \end{array}$$

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

and then Δ is defined to be $\lambda T^{\tau}.\partial(T(B_1(T)))$. Now one can show by induction on the height of the associated tree that for a total F in C_τ, $\mathscr{C}[\![B_1]\!](F) = \lambda m^{\iota}.\mathbf{f}$ and $\mathscr{C}[\![\Delta]\!](F) = tt$. So we have that $\Delta\downarrow^{\mathscr{C}}_{\tau\to o}$. The *kth iterate* $B_1^{(k)}$ of B_1 is obtained by replacing the occurrence of $\mathrm{Y}_{\tau\to(\iota\to o)}$ in its definition by $\mathrm{Y}^{(k)}_{\tau\to(\iota\to o)}$. One can show by induction on k that for a non-total F in C_τ, $F(\mathscr{C}[\![B_1^{(k)}]\!](F)) = \bot$ and so $\mathscr{C}[\![\Delta]\!](F) = \partial(F(\mathscr{C}[\![B_1]\!](F))) = \bot$. It follows that $\mathscr{C}^o[\![\Delta \mathrm{K} \mathbf{1}]\!] = \bot$, and so $\Delta \not\downarrow^{\mathscr{C}^o}_{\tau\to o}$. We therefore see that $\mathscr{C}$ is not 3-PCF-correct for $\mathscr{C}^o$, and therefore neither is $\mathscr{S}$ for $\mathscr{S}^o$ (recall that $\mathscr{C}$ and $\mathscr{S}$ are 3-PCF-correct for each other as are $\mathscr{C}^o$ and $\mathscr{S}^o$). Thus, in all cases, *full abstraction for totality fails in both directions by level 3.*

It is interesting to note that this search technique can be extended to obtain a PCF term $\Phi : \tau \to \iota$, yielding the *modulus of (uniform) continuity* of a total F in C_τ; the associated Kleene–Kreisel functional is (essentially) the *fan functional* – see Gandy and Hyland (1977). The modulus of continuity is defined to be the least number n such that if two total functions f and g in $C_{\iota\to o}$ have the same values for $0,\ldots,(n-1)$, then $Ff = Fg$. We begin by recursively defining $B_2 : \tau \to (\iota \to o)$ by

$$\begin{array}{lcll} B_2(T) & = & \lambda m^{\iota}. \ \mathbf{if}\ \ Zm & \mathbf{then} \quad \mathrm{TrueBranch}(\mathrm{Left}\ T) \\ & & & \mathbf{else} \quad \mathbf{if}\ \mathrm{TrueBranch}(\mathrm{Left}\ T)\ \mathbf{then}\ B_2(\mathrm{Left}\ T)(m-1) \\ & & & \mathbf{else} \quad B_2(\mathrm{Right}\ T)(m-1) \end{array}$$

where TrueBranch abbreviates $\lambda T^{\tau\to o}.\, T(B_2 T)$. Then Φ is recursively defined by

$$\begin{array}{lcll} \Phi & = & \lambda T^{\tau}. & \mathbf{if} \quad (\mathrm{FalseBranch}\ T)\ \mathbf{and}\ (\mathrm{TrueBranch}\ T) \\ & & & \mathbf{then} \quad \max(\Phi(\mathrm{Left}\ T))(\Phi(\mathrm{Right}\ T))+1 \\ & & & \mathbf{else} \quad 0 \end{array}$$

with FalseBranch T understood in a similar way to TrueBranch T. The idea is that TrueBranch has value tt if there is a branch of T with value tt, and correspondingly for FalseBranch. The PCF definability of the fan functional was already known to Gandy (personal communication: Martin Hyland); Berger gave a definition in his thesis Berger (1990). As the idea is not so well known, it seemed worthwhile repeating it here.

Beyond level 3 the remaining questions are the relations between $\mathscr{C}^o$ and $\mathscr{S}^o$, and between $\mathscr{C}$ and $\mathscr{S}$. While these questions are, perhaps, academic, they are related to interesting expressibility questions including one raised by Cook (Cook 1989) and Berger (Berger 1993), as well as to some significant open problems. Cook and Berger asked whether for every term of PCF^{++} that is total in $\mathscr{C}$, there is a PCF term that is totally equivalent in $\mathscr{C}$ (Berger conjectured this is indeed so). Using Φ, we can obtain a positive answer for all types not containing ι negatively: one shows that $\tau \lhd_t^{\mathscr{C},\mathscr{C}} \iota$, and it easily follows that $\sigma \lhd_t^{\mathscr{C},\mathscr{C}} \iota$ if σ does not contain ι negatively, and $\sigma \lhd_t^{\mathscr{C},\mathscr{C}} (\iota \to o)$, if σ does not contain ι positively. However, this all rests on König's lemma and does not even allow one to settle the question at type 3.

A similar expressiveness question is whether for every PCF^{++} term M, total in $\mathscr{C}^o$, there is a PCF term N, totally equivalent to M in $\mathscr{C}^o$ and, moreover, below M in the operational ordering. It is easy to see that a positive answer implies that $\mathscr{C}^o$ and $\mathscr{S}^o$ agree on the

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

totality of PCF terms. The proof is by induction on n, the level of the pure type $\underline{n}$. So, for one direction, suppose $A : \underline{n+1}$ is a closed PCF term, total in $\mathscr{S}^o$. Let $M : \underline{n}$ be a closed PCF^{++} term total in $\mathscr{C}^o$. By the assumption, there is an $N : \underline{n}$ as above. Since N is total in $\mathscr{C}^o$, by the induction hypothesis, it is also total in $\mathscr{S}^o$, so $AN : \iota$ is total in $\mathscr{S}^o$, and so in $\mathscr{C}^o$. But then AM is total too, as $N \leqslant_{\mathrm{op}} M$, so we have shown that A is total in $\mathscr{C}^o$. The proof in the other direction is straightforward. It is worth noting a further (straightforward) consequence of the expressiveness hypothesis, that the applicative structures $C^o/\sim^{C^o}$ and $S^o/\sim^{S^o}$ are isomorphic.

A corresponding question exists regarding $\mathscr{C}$ and $\mathscr{S}$. First we need the notion of *infinitary PCF*. Let $\preceq_\sigma$ be the least type-indexed family of contextually-closed preorders on terms such that for all $M : \gamma$ it holds that $\Omega_\gamma \preceq_\gamma M$. Then the terms of infinitary PCF of type σ are defined to be the directed $\preceq_\sigma$-ideals of PCF terms of type σ; the denotation function $\mathscr{S}[\![\cdot]\!](\rho)$ is extended to such terms by taking the evident directed lub. Infinitary PCF^{++} is defined in the analogous way. Every element of any C_σ can be defined by a closed infinitary PCF^{++} term. It is an open question if the same holds for PCF. (Indeed that is equivalent to the open question as to whether the extensional collapse of the games model is $\mathscr{S}$, since the elements of this collapse, under their identification with elements of the S_σ, are precisely those definable by closed infinitary PCF terms.)

We may now ask whether for every total x in any C_σ there is a closed infinitary PCF term, $N : \sigma$, such that $\mathscr{C}[\![N]\!]$ is totally equivalent to x and below it in the partial order on C_σ. If this holds, then $\mathscr{C}$ and $\mathscr{S}$ agree on totality for infinitary PCF, and so also for PCF. To show this, we need a lemma. Let R be the logical relation between C and S considered above (the one which is the identity at ground types).

Lemma 6.2.

1 For any closed infinitary PCF term, $M : \sigma$, $R_\sigma(\mathscr{C}[\![M]\!], \mathscr{S}[\![M]\!])$.
2 The relation R_σ is surjective, for all σ.
3 Suppose $R_\sigma(x, y)$ and $R_\sigma(x', y')$. Then $x \sim^{\mathscr{C}}_\sigma x'$ iff $y \sim^{\mathscr{S}}_\sigma y'$.

Proof.

1. We have already noted this for PCF. For infinitary PCF it follows from the easily proved fact that each R_σ is closed under directed lubs.

2. We employ closed PCF 'projection terms' $\Psi^n_\sigma : \sigma \to \sigma$, $(n \geqslant 0)$ following Milner (1977) and Berry *et al.* (1985). They are defined by setting $\Psi^n_o = \lambda b.\, b$, $\Psi^0_\iota = \Omega_{\iota \to \iota}$ and then, inductively, $\Psi^{n+1}_\iota = \lambda m.\, \mathbf{if}\ Z m\ \mathbf{then}\ 0\ \mathbf{else}\ \Psi^{n+1}_\iota(m-1) + 1$. The terms Ψ^n_σ are such that $\mathscr{S}[\![\Psi^n_\sigma]\!]$ is an increasing sequence of projections, each with finite range and with lub the identity, and similarly for $\mathscr{C}$. In what follows it is convenient to confuse Ψ^n_σ with its denotation $\mathscr{S}[\![\Psi^n_\sigma]\!]$ (and similarly for $\mathscr{C}$).

Define $X^n \subset S_\sigma$ by

$$X^n = \{\Psi^n_\sigma \mathscr{C}[\![M]\!] \mid M \text{ is a closed PCF term and } \mathscr{S}[\![M]\!] = \Psi^n_\sigma y\}.$$

Each X^n is finite. It is also non-empty, as $\Psi^n_\sigma y$ is finite and so has a PCF definition. Since we have $\Psi^n_\sigma \circ \Psi^{n+1}_\sigma = \Psi^n_\sigma$, one can show that $\Psi^n_\sigma(X^{n+1}) \subset X^n$. It follows by König's Lemma that there is a sequence x_n in X^n such that $x_n = \Psi^n_\sigma x_{n+1}$. But then x_n is increasing and

https://doi.org/10.1017/S0960129598002692 Published online by Cambridge University Press

$R_\sigma(x_n, \Psi^n_\sigma y)$. Since R_σ is closed under directed lubs, we get that $R_\sigma(\bigvee_{n\geqslant 0} x_n, y)$, showing that R_σ is surjective, as required.

3. The proof is by induction on types. The ground case is evident. For $\sigma \to \tau$ we assume $R_{\sigma\to\pi}(f,g)$ and $R_{\sigma\to\pi}(f',g')$. Suppose, first, that $f \sim^{\mathscr{C}}_{\sigma\to\pi} f'$. Assume $y \sim^{\mathscr{S}}_{\sigma} y'$. By Part 2, there are $R_\sigma(x,y)$ and $R_\sigma(x',y')$, so, by the induction hypothesis, we get $x \sim^{\mathscr{C}}_{\sigma} x'$, so $f\cdot x \sim^{\mathscr{C}}_{\tau} f'\cdot x'$, and so, again by the induction hypothesis, $g\cdot y \sim^{\mathscr{S}}_{\tau} g'\cdot y'$, since $R_\tau(f\cdot x, g\cdot y)$ and $R_\tau(f'\cdot x', g'\cdot y')$. This shows that $g \sim^{\mathscr{S}}_{\sigma\to\pi} g'$.

Conversely, assume that $g \sim^{\mathscr{S}}_{\sigma\to\tau} g'$. Assume $x \sim^{\mathscr{C}}_{\sigma} x'$. Under the assumption that the answer to the question is positive, there are closed infinitary PCF terms M and M' such that, setting $u = \mathscr{C}[\![M]\!]$ and $u' = \mathscr{C}[\![M']\!]$, $u, u' \downarrow^{\mathscr{C}}_{\sigma}$, $u \leqslant x$ and $u' \leqslant x'$. Furthermore, setting $y = \mathscr{S}[\![M]\!]$ and $y' = \mathscr{S}[\![M']\!]$, we get that $R_\sigma(u,y)$ and $R_\sigma(u',y')$. So, by the induction hypothesis, $y \sim^{\mathscr{S}}_{\sigma} y'$ (as $u \sim^{\mathscr{C}}_{\sigma} u'$). Therefore $g\cdot y \sim^{\mathscr{S}}_{\tau} g'\cdot y'$, so we have that, again using the induction hypothesis, $f\cdot x \geqslant f\cdot u \sim^{\mathscr{C}}_{\tau} f'\cdot u' \leqslant f'\cdot x'$. Therefore $f\cdot x \sim^{\mathscr{C}}_{\tau} f\cdot x'$, and we have shown that $f \sim^{\mathscr{C}}_{\sigma\to\tau} f'$.

□

That $\mathscr{C}$ and $\mathscr{S}$ agree on totality for infinitary PCF is an immediate consequence of the third part of this lemma. Another, straightforward, consequence is that the applicative structures $C_\sigma/\sim^C_\sigma$ and $S_\sigma/\sim^S_\sigma$ are isomorphic. We should remark that, still assuming a positive answer to the above question, the analogous results are easily shown for the games model.

Acknowledgments

I would like to thank Samson Abramsky for reawakening my interest in totality and domain theory, and Ulrich Berger, John Longley and Dag Normann for interesting and informative discussions. This research was supported by the UK EPSRC.

References

Abramsky, S. (1990) Abstract interpretation, logical relations and Kan extensions. *Journal of Logic and Computation* **1** (1) 5–39.

Abramsky, S., Jagadeesan, R. and Malacaria, P. (1995) Games and full abstraction for PCF. Manuscript available from the site http://www.dcs.ed.ac.uk. Also, to appear in *J. of Inf. and Comp.*

Abramsky, S. and Jung, A. (1994) Domain theory. In: Abramsky, S., Gabbay, D. and Maibaum, T. S. E. (eds.) *Handbook of Logic in Computer Science* **3**, Clarendon Press 1–168.

Barendregt, H. (1984) *The Lambda Calculus*, North-Holland.

Berger, U. (1990) *Totale Objecte und Mengen in der Bereichstheorie*, Ph. D. Thesis, Department of Mathematics, Ludwig-Maximilians-Universität München, Munich, Germany.

Berger, U. (1993) Total objects and sets in domain theory. *Annals of Pure and Applied Logic* **60** 91–117.

Berger, U. (1997) *Continuous Functionals of Dependent and Transfinite Types*, Habilitationsschrift, Department of Mathematics, Ludwig-Maximilians-Universität München, Munich, Germany.

Berry, G., Curien, P.-L. and Lévy, J.-J. (1985) Full abstraction for sequential languages: the state of the art. In: Nivat, M. and Reynolds, J. C. (eds.) *Algebraic Methods in Semantics*, Cambridge University Press 89–132.

Cook, S. A. (1989) Computability and complexity of higher type functions. In: Moschovakis, Y. (ed.) *Logic from Computer Science*, Springer-Verlag 51–72.

Coppo, M., Damiani, F. and Giannini, P. (1997) On strictness and totality. In: Abadi, M. and Ito, T. (eds.) Proc. Third Int. Sym. on Theoretical Aspects of Computer Software. *Springer-Verlag Lecture Notes in Computer Science* **1281** 138–164.

Denney, E. (1998) Refinement types for specification. In: Gries, D. and de Roever, W.-P. (eds.) *Proc. Programming Concepts and Methods '98*, Chapman and Hall 148–166.

Ershov, Y. L. (1974) Maximal and everywhere-defined functionals. *Algebra i Logika* **13** (4) 347–392.

Ershov, Y. L. (1975) Theorie der numerierungen II. *Zeitschr. f. Math. Logik u. Grundl. d. Math.* **21** 473–584.

Ershov, Y. L. (1976) Hereditarily effective operations. *Algebra i Logika* **15** (6) 642–654.

Ershov, Y. L. (1977) Model C of partial continuous functionals. In: Gandy, R. O. and Hyland, J. M. E. (eds.) *Logic Colloquium '76*, North Holland 455–467.

Fiore, M. P., Jung, A., Moggi, E., O'Hearn, P., Riecke, J., Rosolini, G. and Stark, I. (1996) Domains and denotational semantics: history, accomplishments and open problems. *Bulletin of the European Association for Theoretical Computer Science* (59) 227–256.

Gandy, R. O. and Hyland, J. M. E. (1977) Computable and recursively countable functions of higher type. In: Gandy, R. O. and Hyland, J. M. E. (eds.) *Logic Colloquium '76*, North Holland 407–438.

Hyland, J. M. E. (1975) *Recursion Theory on the Countable Functionals*, D. Phil. thesis, Department of Mathematics, University of Oxford, Oxford, United Kingdom.

Hyland, J. M. E. and Ong, C-H. L. (1994) On full abstraction for PCF: I, II and III. Manuscript available by ftp in directory /pub/Documents/techpapers/Luke.Ong/pcf.ps.gz. at ftp.comlab.ox.ac.uk.

Kapron, B. M. and Cook, S. A. (1996) A new characterization of type-2 feasibility. *SIAM J. on Computing* **25** (1) 117–132.

Kristiansen, L. and Normann, D. (1995) Semantics for some constructors of type theory. In: Behara, M., Fritsch, R. and Lintz, R. (eds.) *Symposia Gaussiana*, de Gruyter 201–224.

Kristiansen, L. and Normann, D. (1997) Total objects in inductively defined types. *Arch. Math. Logic* **36** 405–436.

Loader, R. (1997) Equational theories for inductive types. *Annals of Pure and Applied Logic* **84** 175–217.

Longley, J. and Plotkin, G. (1998) Logical full abstraction and PCF. In: Ginzburg, J., Khasidashvili, Z., Vogel, C., Lévy, J.-J. and Vallduví, E. (eds.) *Tbilisi Symposium on Logic, Language and Information: Selected Papers*, CSLI 333–352.

Longo, G. and Moggi, E. (1984) The hereditarily partial effective functionals and recursion theory in higher types. *Journal of Symbolic Logic* **49** 1319–1332.

Milner, R. (1977) Fully abstract models of typed λ-calculi. *Theoretical Comp. Sci.* **4** 1–22.

Mitchell, J. (1996) *Foundations for Programming Languages*, MIT Press.

Nickau, H. (1996) *Hereditarily Sequential Functionals: A Game-Theoretic Approach to Sequentiality*, Ph. D. Thesis, Department of Mathematics, Universität Siegen, Siegen, Germany. Shaker Verlag.

Normann, D. (1996) A hierarchy of domains with totality, but without density. In Cooper, S. B., Slaman, T. A. and Wainer, S. S. (eds.) *Computability, Enumerability, Unsolvability*, Cambridge University Press 233–257.

Normann, D. (1997) Categories of Domains with Totality. Preprint No. 4, Department of Mathematics, University of Oslo, Oslo, Norway.

Normann, D. (1997) Closing the gap between the continuous functionals and recursion in ^3E. *Arch. Math. Logic* **36** 269–287.

Normann, D. (1998) The continuous functionals. To appear in Griffor, E. R. (ed.) *Handbook of Computability Theory*, Elsevier.

Plotkin, G. (1977) LCF considered as a programming language. *Theoretical Comp. Sci.* **5** 223–255.
Scott, D. (1976) Data types as lattices. *SIAM J. Comput.* **5** 522–587.
Scott, D. (1993) A type-theoretical alternative to CUCH, ISWIM and OWHY. In: Logic, Semantics and Theory of Programming. *Theoretical Comp. Sci.* **121** (1,2) 411–440.
Stoltenberg-Hansen, V., Lindström, I. and Griffor, E. R. (1994) *Mathematical Theory of Domains*, Cambridge University Press.
Stoughton, A. (1991) Interdefinability of parallel operations in PCF. *Theoretical Comp. Sci.* **79** 357–358.
Troelstra, A. S. (1973) Metamathematical Investigations of Intuitionistic Arithmetic and Analysis. *Springer-Verlag Lecture Notes in Mathematics* **344**.
Waagbø, G. (1998) Denotational semantics for intuitionistic type theory using a hierarchy of domains with totality. To appear in *Arch. Math. Logic*.
Winskel, G. (1993) *The Formal Semantics of Programming Languages*, MIT Press.