An Axiomatisation of Computationally Adequate Domain Theoretic Models of FPC

Marcelo P. Fiore[*] and Gordon D. Plotkin[†]
Department of Computer Science
Laboratory for Foundations of Computer Science
University of Edinburgh, The King's Buildings
Edinburgh EH9 3JZ, Scotland
<mf@dcs.ed.ac.uk> <gdp@dcs.ed.ac.uk>

April 1994

Synopsis

Categorical models of the metalanguage FPC (a type theory with sums, products, exponentials and recursive types) are defined. Then, domain-theoretic models of FPC are axiomatised and a wide subclass of them —the non-trivial and absolute ones— are proved to be both computationally sound and adequate.

Examples include: the category of cpos and partial continuous functions and functor categories over it.

1 Introduction

This paper is an investigation into *axiomatic categorical domain theory* as needed for the denotational semantics of deterministic programming languages. We particularly consider a metalanguage FPC, a typed functional language with sums, products, exponentials and recursive types equipped with a *call-by-value* operational semantics (see [Plo85, Gun92]). We wish to axiomatise domain-theoretic models of FPC and prove that they provide a *computationally sound* and *adequate* denotational semantics.

Such a theorem holds for **pCpo** [Plo85], the category of small cpos (posets, possibly without bottom, closed under lubs of ω-chains) and partial continuous functions. The aim of this paper is to generalise to a wide class of *order-enriched* categories (Section 2); one can compare this endeavour to [SP82, Fre90, Fre92] where a similar programme was carried out for the solution of recursive domain equations.

[*]Research partially supported by Fundación Antorchas and The British Council grant ARG 2281/14/6 and SERC grant RR30735.

[†]Research supported by an SERC Senior Fellowship.

In order to provide a *direct* semantic treatment of non-termination, we consider order-enriched categories of *partial maps* (Section 3). Computational soundness for FPC is then that if the evaluation of a program terminates its denotation is total; adequacy is the converse implication. Here programs are taken to be closed terms of *any* closed type.

The categorical structure needed for interpreting the type constructors of FPC is discussed (Section 4) in the order-enriched setting. A uniform treatment of type constructors with mixed variance is provided by transforming them into covariant type constructors on *universal involutory categories* (those that are self-dual via an involution). One can then interpret recursive types (Section 5) following [Fre91] and using a notion of *parameterised algebraic compactness*.

Categorical models of FPC are defined to be parameterised algebraically compact partial cartesian closed categories with finite coproducts; they enable a *denotational semantics* to be given (Section 6). Abstract examples of categorical models are provided by *domain-theoretic* models; the leading example of such a model is —naturally— **pCpo**. These models are specified by order-theoretic conditions intended to be easy to verify.

The main technical contribution of the paper (Section 7) is that non-trivial domain-theoretic models of FPC satisfying an *absoluteness axiom* are computationally sound and adequate.

It follows that the domain-theoretic model specified by **pCpo** —as well as many full subcategories of domains— is computationally adequate. Other examples are functor categories over **pCpo**. Categories of stable functions may provide further models as may *synthetic domain theory* [Tay91] —though internally in a topos.

Our proof of adequacy depends on certain *formal-approximation* relations between semantic and syntactic values (see Subsection 7.2). Lemma 7.11 identifies the essential properties needed to prove their existence. Recently, a new method of proof has been proposed by Pitts [Pit93] in the context of $\mathbf{Cppo}_\perp$ (the category of small pointed cpos and strict continuous functions). We hope his approach will extend to our axiomatic setting. We also wonder whether there is an abstract version of Abramsky's approach to adequacy via Stone duality [Abr90].

We have taken both partiality and order as primitive notions. An attractive alternate possibility would be to take partiality as the *sole* primitive notion. In Section 3 a definition of the order in terms of totality is proposed. It would be interesting to find natural axioms on partiality which would yield an (absolute) domain-theoretic model. Such axioms would provide a computational justification of Scott's original consideration of ordered structures. (For an initial investigation in this direction see [Fio94].) A related possibility is to generalise from order-enrichment to general enrichment. The first problem there is to provide a good notion of $\mathcal{V}$-domain-structure; any $\mathcal{V}$-category with such a structure should yield a $\mathcal{V}$-category of partial maps. As to adequacy, we believe that if Lemma 7.11 generalises to arbitrary cartesian $\mathcal{V}$, the formal-approximation relations can be extracted.

It seems that so straightforward an adequacy theorem would not be available if one instead axiomatised **Cppo** (the category of small pointed cpos and continuous functions) —see [MC88]. The essential difficulty is that the least element does not correspond to non-termination at higher types. Again, although **pCpo** is equivalent to $\mathbf{Cppo}_\perp$ they are conceptually different. The first seems computationally more natural, fitting with standard formulations of recursion theory that emphasise partial functions. The latter has a natural generalisation to models of intuitionistic linear type theory with recursion [Plo93]; the connection with operational notions is unclear.

It would be interesting to make abstract categorical studies of other notions of computation, for example nondeterminism or probabilistic computation. There behaviour concerns more than termination and semantics more than existence and one should extend the metalanguage, e.g. with a construct for (probabilistic) choice. One might even investigate notions of computation in general. Semantical suggestions have been made: Moggi has proposed the use of Kleisli categories [Mog89]; models of linear type theory with recursion form another possibility. However, a correspondingly general view of behaviour is completely lacking.

2 Order-enriched category theory

For a thorough treatment of enriched category theory consult [Kel82]. In the rest of the paper we let $\mathcal{V}$ stand for either **Poset** (the category of small posets and monotone functions) or **Cpo** (the category of small cpos and continuous functions). But we stress that all our $\mathcal{V}$-notions with their associated results generalise to arbitrary cartesian $\mathcal{V}$ (see [Fio94]).

A **Poset**-*category* (**Cpo**-*category*) is a locally small category whose hom-sets come equipped with a partial order (complete partial order) with respect to which composition of morphisms is a monotone (continuous) operation.

Both **Poset** and **Cpo** with each hom-set ordered pointwise are examples of **Poset**-categories; **Cpo** is even a **Cpo**-category.

The *terminal* $\mathcal{V}$*-category* $\mathbf{1}$ has a singleton set of objects while the only hom is the terminal object in $\mathcal{V}$. The *product,* $\mathcal{A}\times\mathcal{B}$*, of the* $\mathcal{V}$*-categories* $\mathcal{A}$ *and* $\mathcal{B}$, is the $\mathcal{V}$-category with $|\,\mathcal{A}\times\mathcal{B}\,| = |\,\mathcal{A}\,|\times|\,\mathcal{B}\,|$, and homs defined as

$$(\mathcal{A}\times\mathcal{B})\big((A,B),(A',B')\big) = \mathcal{A}(A,A')\times\mathcal{B}(B,B').$$

For a $\mathcal{V}$-category $\mathcal{A}$ its *dual*, $\mathcal{A}^{op}$, is the $\mathcal{V}$-category with the same objects as $\mathcal{A}$, but with $\mathcal{A}^{op}(A,A') = \mathcal{A}(A',A)$. Notice that the order in each hom remains unchanged.

A $\mathcal{V}$*-functor* $F:\mathcal{A}\to\mathcal{B}$ between $\mathcal{V}$-categories $\mathcal{A}$ and $\mathcal{B}$, consists of a mapping associating every $A\in|\,\mathcal{A}\,|$ with some $FA\in|\,\mathcal{B}\,|$ and a functorial mapping associating every $A,A'\in|\,\mathcal{A}\,|$ with some $F_{A,A'}:\mathcal{A}(A,A')\to\mathcal{B}(FA,FA')$ in $\mathcal{V}$. An ordinary functor $F:\mathcal{A}\to\mathcal{B}$ is said to $\mathcal{V}$*-enrich* if for every $A,A'\in|\,\mathcal{A}\,|$, the function $F_{A,A'}$ is in $\mathcal{V}$.

A $\mathcal{V}$*-adjunction* $\chi : F\dashv U:\mathcal{A}\to\mathcal{B}$ is given by $\mathcal{V}$-categories $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{V}$-functors $F:\mathcal{B}\to\mathcal{A}$ and $U:\mathcal{A}\to\mathcal{B}$, and a natural isomorphism

$$\chi:\mathcal{A}(F_{-},{=})\cong\mathcal{B}({-},U_{=}):\mathcal{B}^{op}\times\mathcal{A}\to\mathcal{V}.$$

A **Poset**-adjunction involving **Cpo**-categories establishes a **Cpo**-adjunction because the inclusion functor **Cpo** → **Poset** creates isomorphisms.

3 Partial maps

For a thorough survey of categories of partial maps consult [RR88].

Definition 3.1 [Mog86, Ros86] A *domain structure* is a pair $(\mathcal{K}, \mathcal{D})$ consisting of a category $\mathcal{K}$ and a well-powered category $\mathcal{D}$, such that $\mathcal{D}$ is a full-on-objects subcategory of $\mathcal{K}$ all of whose morphisms are monos in $\mathcal{K}$, with the following closure property: every diagram $X \xrightarrow{f} A \overset{n}{\leftarrowtail} D$ with $f \in \mathcal{K}$ and $n \in \mathcal{D}$ has a pullback in $\mathcal{K}$ and if $X \overset{f^{-1}(n)}{\leftarrowtail} f^{-1}(D) \xrightarrow{n^*f} D$ is any such pullback then $f^{-1}(n) \in \mathcal{D}$. $\square$

Convention. $\mathcal{K}$ is called the category of *total* maps, and $\mathcal{D}$ is called the category of *admissible* monos. For every $A \in |\,\mathcal{D}\,|$, we write $\mathcal{D}(A)$ for the representative small set of subobjects of A in $\mathcal{D}$ guaranteed by well-powerness.

Definition 3.2 The *category of partial maps* $p(\mathcal{K}, \mathcal{D})$ induced by a domain structure $(\mathcal{K}, \mathcal{D})$ has the same objects as $\mathcal{K}$; a partial map $[m, f] : A \rightharpoonup B$ is an equivalence class of spans $A \overset{m}{\leftarrowtail} D \xrightarrow{f} B$ in $\mathcal{D} \times \mathcal{K}$, where two spans $A \overset{m}{\leftarrowtail} D \xrightarrow{f} B$ and $A \overset{n}{\leftarrowtail} E \xrightarrow{g} B$ are equivalent iff $m = n \circ i$ and $f = g \circ i$ for some isomorphism $i : D \cong E$. Composition of partial maps is given as for relations, by pullback; identities have the form $[\mathrm{id}_A, \mathrm{id}_A]$. $\square$

Convention. When $\mathcal{D}$ is clear from the context we simply write $p\mathcal{K}$.

$\mathcal{K}$ appears as a full-on-objects subcategory of $p\mathcal{K}$ via the faithful *inclusion* functor $J : \mathcal{K} \to p\mathcal{K}$ sending a total map f to the partial map $[\mathrm{id}, f]$. To indicate that a partial map u is total (i.e. it is in the image of J) we write $u\downarrow$.

The motivating example of a domain structure is $(\mathbf{Cpo}, \mathbf{\Sigma})$ where $\mathbf{\Sigma}$ is the subcategory of $\mathbf{Cpo}$ consisting of all those order-reflecting monos whose subobjects are Scott-open. We have: $p(\mathbf{Cpo}, \mathbf{\Sigma}) \cong \mathbf{pCpo}$.

We now consider the interaction of partiality and order-enrichment.

Definition 3.3 Given a domain structure $(\mathcal{K}, \mathcal{D})$ and a **Poset**-enrichment $\sqsubseteq$ for $\mathcal{K}$, we define a partial order $\underset{\sim}{\sqsubset}_{A,B}$ on every $p\mathcal{K}(A, B)$ by setting $u \underset{\sim}{\sqsubset}_{A,B} v$ iff for every $x : X \to A$,

$$u \circ x\downarrow \Rightarrow\ (v \circ x\downarrow\ \wedge u \circ x \sqsubseteq_{X,B} v \circ x). \qquad \square$$

Proposition 3.4 For every $[m, f], [n, g] : A \rightharpoonup B$,

$[m, f] \underset{\sim}{\sqsubset} [n, g] \Leftrightarrow m = n \circ i\ \wedge\ f \sqsubseteq g \circ i$ for unique i.$\square$

See [Fio93] for characterisations of when categories of partial maps induced by arbitrary domain structures or *uniform* domain structures (in the sense of the definition below) respectively **Poset**-enrich or **Cpo**-enrich with respect to $\underset{\sim}{\sqsubset}$.

Definition 3.5 A domain structure $(\mathcal{K}, \mathcal{D})$ is said to be *uniform* if whenever $\langle [m_k] \rangle$ is an ω-chain in $\mathcal{D}(A)$ with lub [m], where $m_k : D_k \rightarrowtail A$ and $m : D \rightarrowtail A$, it follows that $\mu : \langle D_k, {m_{k+1}}^{-1}(m_k) \rangle \stackrel{\cdot}{\to} D$ is colimiting in $\mathcal{K}$ where $\mu_k = m^{-1}(m_k)$. $\square$

The explicit distinction between partial and total maps can be used to define a *contextual approximation* preorder: for $u, v : A \rightharpoonup B$,

$$u \underset{\sim}{\sqsubset}^c v \Leftrightarrow \forall \text{ context } C[\_] \text{ in } p\mathcal{K}.\, C[u]\downarrow \Rightarrow\ C[v]\downarrow$$

where a context is an *incomplete* composite $w \circ \_ \circ x$.

The description of partial maps based on domain structures provides another computationally natural notion of approximation. For this purpose, admissible monos are regarded as predicates describing *observable properties*. The *specialisation* preorder for partial maps is defined as follows:

$$\begin{array}{ll} & u \underset{\sim}{\sqsubset}^s v : A \rightharpoonup B \text{ in } p\mathcal{K} \\ \Leftrightarrow & \\ & u^{-1}(\_) \subseteq v^{-1}(\_) : \mathcal{D}(B) \to \mathcal{D}(A) \text{ in } \mathbf{Poset} \end{array}$$

where the *inverse image* of $[n]$ under $[m, f]$ is $[m \circ f^{-1}(n)]$.

The notions of approximation obtained by testing and by observing partial maps coincide. Writing $\underset{\sim}{\sqsubset}$ for either of them and $\sqsubseteq$ for the restriction to total maps, it can be shown that if $\sqsubseteq$ is a partial order (as in **Cpo**) then $\underset{\sim}{\sqsubset}$ arises from $\sqsubseteq$ as in the more general situation considered in Definition 3.3. (For details see [Fio94].)

4 Type constructors

4.1 Binary partial products

Let $\mathcal{K}$ have binary products. The *partial pairing* of $u = [m, f] : B \rightharpoonup A_1$ and $v = [n, g] : B \rightharpoonup A_2$ is the partial map $B \rightharpoonup A_1 \times A_2$ defined by

$$\langle\!\langle u, v \rangle\!\rangle \quad = \quad [m \cap n, \langle f \circ m^{-1}(n), g \circ n^{-1}(m) \rangle]$$

where $\langle \_, = \rangle$ is the pairing of total maps. The product functor $\_ \times = : \mathcal{K} \times \mathcal{K} \to \mathcal{K}$ extends to a *partial product* functor $\_\otimes = : p\mathcal{K} \times p\mathcal{K} \to p\mathcal{K}$ sending a pair of objects (A, B) to $A \times B$ and a pair of partial maps (u, v) to $\langle\!\langle u \circ \pi_1, v \circ \pi_2 \rangle\!\rangle$.

Proposition 4.1 Let $(\mathcal{K}, \sqsubseteq)$ and $(p\mathcal{K}, \underset{\sim}{\sqsubset})$ be **Poset**-categories. If $\mathcal{K}$ has binary **Poset**-products then $\_\otimes =$ **Poset**-enriches. $\square$

4.2 Partial exponentials

As usual for higher types, exponentiation arises as right adjoint to multiplication. In our case, we let $\mathcal{K}$ have binary products and for every $A \in |\, \mathcal{K} \,|$, we ask that $p\lambda : \_\otimes A \dashv A \underline{\rightharpoonup} \_ : p\mathcal{K} \to \mathcal{K}$ is an adjunction. We write ε for the counit of this adjunction and call each $A \underline{\rightharpoonup} B$ a *partial-exponential*. Partial-exponentials extend to a functor $\_ \underline{\rightharpoonup} = : p\mathcal{K}^{op} \times p\mathcal{K} \to p\mathcal{K}$ sending pairs of objects (A, B) to $A \underline{\rightharpoonup} B$ and pairs of partial maps (u, v) to $p\lambda\big(v \circ \varepsilon \circ (\mathrm{id} \otimes u)\big)$.

Definition 4.2 [LM84] A category of partial maps $p\mathcal{K}$ is *partial cartesian closed* if $\mathcal{K}$ is cartesian and $p\mathcal{K}$ has partial exponentials. $\square$

Proposition 4.3 Let $(\mathcal{K}, \mathcal{D})$ be a uniform domain structure and let $\sqsubseteq$ be a **Cpo**-enrichment for $\mathcal{K}$ such that $(p\mathcal{K}, \underset{\sim}{\sqsubset})$ is a **Cpo**-category. Assume $p\mathcal{K}$ is partial cartesian closed. If $\_ \times = : \mathcal{K} \times \mathcal{K} \to \mathcal{K}$ **Cpo**-enriches then so does $\_ \otimes = : p\mathcal{K} \times p\mathcal{K} \to p\mathcal{K}$. $\square$

Proposition 4.4 Let $(\mathcal{K}, \sqsubseteq)$ and $(p\mathcal{K}, \underset{\sim}{\sqsubset})$ be **Poset**-categories. Assume $\mathcal{K}$ has binary **Poset**-products. If $p\lambda$ determines a **Poset**-adjunction then $\_ \underline{\rightharpoonup} =$ **Poset**-enriches.

Moreover, whenever $(\mathcal{K}, \sqsubseteq)$ and $(p\mathcal{K}, \underset{\sim}{\sqsubset})$ are **Cpo**-categories, if $\_ \otimes =$ **Cpo**-enriches then so does $\_ \underline{\rightharpoonup} =$. $\square$

4.3 Colimits

Proposition 4.5 Let $J \dashv L : (p\mathcal{K}, \underset{\sim}{\sqsubset}) \to (\mathcal{K}, \sqsubseteq)$ be a **Poset**-adjunction.

1. If $\mathcal{K}$ has binary **Poset**-coproducts then so does $p\mathcal{K}$.
2. Assuming $\mathcal{K}$ has a terminal object, if $\mathcal{K}$ has colimits of ω-chains of embeddings then so does $p\mathcal{K}$. $\square$

4.4 Involutory categories

Following a suggestion of John Power, we focus on *involutory* categories (those that are self-dual via an involution). This enables us to transform mixed-variance functors into covariant ones in a *universal* way.

Definition 4.6 The large category **InvCAT** of *involutory* locally small categories has objects $(\mathcal{C}, (\_)^c)$ where $\mathcal{C} \in |\, \mathbf{CAT} \,|$ and $(\_)^c : \mathcal{C} \to \mathcal{C}^{op}$ is an involution; a morphism $F : (\mathcal{A}, (\_)^a) \to (\mathcal{B}, (\_)^b)$ is a functor $F : \mathcal{A} \to \mathcal{B}$ such that $F^{op} \circ (\_)^a = (\_)^b \circ F$; and composition and identities as in **CAT**. $\square$

Examples of involutory categories abound: for every $\mathcal{C} \in |\, \mathbf{CAT} \,|$, $(\breve{\mathcal{C}}, (\_)^\S) = (\mathcal{C}^{op} \times \mathcal{C}, \langle \Pi_2, \Pi_1 \rangle)$ is involutory. As a morphism $F : (\breve{\mathcal{A}}, (\_)^\S) \to (\breve{\mathcal{B}}, (\_)^\S)$ is a functor such that $F(f', f)_1 = F(f, f')_2$, functors in **InvCAT** are called *symmetric*. We also call objects X such that $X = X^c$ symmetric. Thus, FX is symmetric if F and X are.

The involutory categories $(\breve{\mathcal{B}}, (\_)^\S)$ are universal in that they are characterised by a natural bijective correspondence

$$\frac{\mathcal{A} \xrightarrow{F} \mathcal{B}}{(\mathcal{A}, (\_)^a) \xrightarrow[\text{symmetric}]{\breve{F}} (\breve{\mathcal{B}}, (\_)^\S)}$$

given by the mapping $F \mapsto \breve{F} = \langle F^{op} \circ (\_)^a, F \rangle$. This property allows us to turn mixed-variance functors on a category $\mathcal{C}$ into covariant symmetric functors on $\breve{\mathcal{C}}$.

Note that **InvCAT** is cartesian with terminal object $(\mathbf{1}, \mathrm{Id})$ and products $(\mathcal{A}, (\_)^a) \times (\mathcal{B}, (\_)^b) = (\mathcal{A} \times \mathcal{B}, (\_)^a \times (\_)^b)$.

5 Recursive types

In [Fre91, Fre92], Peter Freyd defined an *algebraically complete category* as one such that each of its endofunctors has an initial algebra and remarked that this should be understood in a 2-categorical setting; that is, a setting in which the phrase "every endofunctor" refers to an understood class. For us, this will be determined by enrichment. One possibility is:

Definition 5.1
(c.f. [Fre91]) A $\mathcal{V}$-category is *$\mathcal{V}$-algebraically complete* if every $\mathcal{V}$-endofunctor on it has an initial algebra. $\square$

Algebraic completeness guarantees the existence of *parameterised* initial algebras. Let $\mathcal{A}$ be $\mathcal{V}$-algebraically complete and $F : \mathcal{X} \times \mathcal{A} \to \mathcal{A}$ be a $\mathcal{V}$-functor. As for every $X \in |\, \mathcal{X} \,|$ we have that $F(X, \_) : \mathcal{A} \to \mathcal{A}$ is a $\mathcal{V}$-functor, we can set $(F^\dagger X, \iota^F_X)$ to be an initial $F(X, \_)$-algebra. To extend the action of $F^\dagger$ to morphisms, for every $f : X \to X'$ in $\mathcal{X}$, let $F^\dagger f : F^\dagger X \to F^\dagger X'$ be the unique $F(X,\_)$-algebra morphism from $(F^\dagger X, \iota^F_X)$ to $\big(\, F^\dagger X', \iota^F_{X'} \circ F(f, F^\dagger X')\,\big)$. By the universal property of initial algebras, $F^\dagger$ is a functor $\mathcal{X} \to \mathcal{A}$ and, by construction, ι^F is a natural transformation $F\,\langle \mathrm{Id}, F^\dagger\rangle \stackrel{.}{\to} F^\dagger$. The pair $(F^\dagger, \iota^F)$ is called an *initial parameterised F-algebra.*

Unfortunately $F^\dagger$ need not $\mathcal{V}$-enrich. To see this let $\mathcal{Z}$ be the **Poset**-category induced by the ordered monoid $Z = (\mathbb{Z}, \leq, 0, +)$ and consider the **Poset**-functor $F : \mathcal{Z} \times \mathcal{Z} \to \mathcal{Z} : (m, n) \mapsto m + 2n$. Every $i \in \mathbb{Z}$ is an initial algebra for $F(Z, \_) : \mathcal{Z} \to \mathcal{Z} : n \mapsto 2n$ and $F^\dagger m$ is a morphism from i to $i + F(m, F^\dagger Z) = i + m$ iff $F^\dagger m = -m$, hence $F^\dagger : \mathcal{Z} \to \mathcal{Z}$ does not **Poset**-enrich.

Definition 5.2 A $\mathcal{V}$-category $\mathcal{A}$ is *parameterised $\mathcal{V}$-algebraically complete* if it is $\mathcal{V}$-algebraically complete and for every $\mathcal{V}$-functor $F : \mathcal{X} \times \mathcal{A} \to \mathcal{A}$ and every family $\{\iota^F_X : F(X, F^\dagger X) \to F^\dagger X\}_{X \,\in\, |\,\mathcal{X}\,|}$ of initial $F(X,\_)$-algebras, the induced functor $F^\dagger : \mathcal{X} \to \mathcal{A}$ $\mathcal{V}$-enriches.

A *parameterisation* $\big((\_)^\dagger, \iota^{(-)}\big)$ on $\mathcal{A}$ is a pair of mappings that associates every $\mathcal{V}$-functor $F : \mathcal{X} \times \mathcal{A} \to \mathcal{A}$ with a $\mathcal{V}$-functor $F^\dagger : \mathcal{X} \to \mathcal{A}$ and a natural transformation $\iota^F : F\,\langle \mathrm{Id}, F^\dagger\rangle \stackrel{.}{\to} F^\dagger$ such that $(F^\dagger, \iota^F)$ is an initial parameterised F-algebra. ☐

Convention. From now on, in the situation of the above definition, the family of initial algebras determining $F^\dagger$ will be left implicit.

Definition 5.3 1. (c.f. [Fre91]) A $\mathcal{V}$-category is *$\mathcal{V}$-algebraically compact* if it is $\mathcal{V}$-algebraically complete and the initial algebra of every $\mathcal{V}$-endofunctor on it is *free*, in the sense that its inverse is a final coalgebra.

2. A $\mathcal{V}$-category is *parameterised $\mathcal{V}$-algebraically compact* if it is $\mathcal{V}$-algebraically compact and parameterised $\mathcal{V}$-algebraically complete. ☐

Theorem 5.4 1. [Fio94] (c.f. [Bar92, Adá93]) A **Cpo**-category with an e-initial object (an initial object such that every morphism with it as source is an embedding) and colimits of ω-chains of embeddings is **Cpo**-algebraically complete.

2. [Fre92] A **Cpo**-algebraically complete category with pointed homs and strict composition is **Cpo**-algebraically compact.

3. **Cpo**-algebraic completeness (resp. compactness) implies parameterised **Cpo**-algebraic completeness (resp. compactness). ☐

Theorem 5.5 [Product Theorem] (c.f. [Fre91]) If $\mathcal{A}$ and $\mathcal{B}$ are parameterised $\mathcal{V}$-algebraically compact then so is $\mathcal{A} \times \mathcal{B}$. ☐

Algebraic compactness is a *self-dual* property. Hence:

Corollary 5.6 1. If $\mathcal{A}$ is (parameterised) $\mathcal{V}$-algebraically compact then so is $\mathcal{A}^{op}$.

2. If $\mathcal{A}$ is parameterised $\mathcal{V}$-algebraically compact then so is $\breve{\mathcal{A}}$. ☐

The last property of compactness that will be needed is that symmetric functors are closed under initial parameterised algebras:

Theorem 5.7 Let $\mathcal{X}$ and $\mathcal{A}$ be $\mathcal{V}$-categories. Assume that $\mathcal{A}$ is parameterised $\mathcal{V}$-algebraically compact. For a symmetric $\mathcal{V}$-functor $F : \breve{\mathcal{X}} \times \breve{\mathcal{A}} \to \breve{\mathcal{A}}$, every initial parameterised F-algebra $(F^\dagger, \iota^F)$ canonically induces an initial parameterised F-algebra $(F^\ddagger, \varphi^F)$ such that $F^\ddagger$ is a symmetric $\mathcal{V}$-functor and $\varphi^\S_X = \varphi^{-1}_X$ for every symmetric X. ☐

Remark. When $\mathcal{X} = \mathbf{1}$ and $F = \breve{G}$ for $G : \breve{\mathcal{A}} \to \mathcal{A}$, the above theorem provides a canonical and minimal fixed-point for G.

6 FPC

6.1 The Language

The grammar of FPC requires two syntax classes of variables: *type* variables (ranged over by `T`) and *expression* variables (ranged over by `x`). The syntax of the language is as follows:

Types $\tau ::= \mathtt{T} \mid \tau_1 + \tau_2 \mid \tau_1 \times \tau_2 \mid \tau_1 \rightharpoonup \tau_2 \mid \mu\mathtt{T}.\tau$,

Expressions

$$\begin{array}{rl} e \;\; ::= & \mathtt{x} \mid \mathtt{inl}_{\tau_1,\tau_2}(e) \mid \mathtt{inr}_{\tau_1,\tau_2}(e) \mid \\ & \mathtt{case}\ e\ \mathtt{of}\ \mathtt{inl}(\mathtt{x}_1).e_1\ \mathtt{or}\ \mathtt{inr}(\mathtt{x}_2).e_2 \mid \\ & \langle e_1, e_2\rangle \mid \mathtt{fst}(e) \mid \mathtt{snd}(e) \mid \lambda\mathtt{x} : \tau.e \mid e_1(e_2) \mid \\ & \mathtt{intro}_{\mu\mathtt{T}.\tau}(e) \mid \mathtt{elim}(e). \end{array}$$

A well-formed type $\Theta \vdash \tau$ consists of a list of distinct type variables Θ and a type τ whose free type variables appear in Θ. Well-formed expression contexts $\Theta \vdash \Gamma$ are finite mappings assigning well-formed types to expression variables. For the definition of well-formed expressions $\Theta, \Gamma \vdash e : \tau$ see [Gun92].

6.2 Operational semantics

The operational semantics is defined by an *evaluation* relation ($\rightsquigarrow$) on expressions. The call-by-value evaluation rules are:

$$\boxed{\text{var}} \qquad \frac{}{\mathtt{x} \rightsquigarrow \mathtt{x}}$$

$$\boxed{+\text{Il}} \quad \frac{e \rightsquigarrow v}{\mathtt{inl}(e) \rightsquigarrow \mathtt{inl}(v)} \qquad \boxed{+\text{Ir}} \quad \frac{e \rightsquigarrow v}{\mathtt{inr}(e) \rightsquigarrow \mathtt{inr}(v)}$$

$$\boxed{+\text{El}} \quad \frac{e \rightsquigarrow \mathtt{inl}(v) \qquad e_1[\mathtt{x}_1 \mapsto v] \rightsquigarrow v'}{\mathtt{case}\ e\ \mathtt{of}\ \mathtt{inl}(\mathtt{x}_1).e_1\ \mathtt{or}\ \mathtt{inr}(\mathtt{x}_2).e_2 \rightsquigarrow v'}$$

$$\boxed{+\text{Er}} \quad \frac{e \rightsquigarrow \mathtt{inr}(v) \qquad e_2[\mathtt{x}_2 \mapsto v] \rightsquigarrow v'}{\mathtt{case}\ e\ \mathtt{of}\ \mathtt{inl}(\mathtt{x}_1).e_1\ \mathtt{or}\ \mathtt{inr}(\mathtt{x}_2).e_2 \rightsquigarrow v'}$$

$$\boxed{\times\text{I}} \quad \frac{e_1 \rightsquigarrow v_1 \qquad e_2 \rightsquigarrow v_2}{\langle e_1, e_2\rangle \rightsquigarrow \langle v_1, v_2\rangle}$$

$$\boxed{\times\text{E}_1} \quad \frac{e \rightsquigarrow \langle v_1, v_2\rangle}{\mathtt{fst}(e) \rightsquigarrow v_1} \qquad \boxed{\times\text{E}_2} \quad \frac{e \rightsquigarrow \langle v_1, v_2\rangle}{\mathtt{snd}(e) \rightsquigarrow v_2}$$

$$\boxed{\underline{\rightharpoonup}\text{I}} \qquad \frac{}{\lambda\mathtt{x} : \tau.e \rightsquigarrow \lambda\mathtt{x} : \tau.e}$$

$$\boxed{\underline{\rightharpoonup}\text{E}} \quad \frac{e_1 \rightsquigarrow \lambda\mathtt{x} : \tau.e \qquad e_2 \rightsquigarrow v \qquad e[\mathtt{x} \mapsto v] \rightsquigarrow v'}{e_1(e_2) \rightsquigarrow v'}$$

$$\boxed{\mu\text{I}} \quad \frac{e \rightsquigarrow v}{\mathtt{intro}(e) \rightsquigarrow \mathtt{intro}(v)}$$

$$\boxed{\mu\text{E}} \quad \frac{e \rightsquigarrow \mathtt{intro}(v)}{\mathtt{elim}(e) \rightsquigarrow v}$$

The expressions v such that $v \rightsquigarrow v$, called *values*, are characterised by the grammar,

$$\begin{array}{lrl} \text{Values} \quad v & ::= & \mathtt{x} \mid \mathtt{inl}_{\tau_1,\tau_2}(v) \mid \mathtt{inr}_{\tau_1,\tau_2}(v) \mid \\ & & \langle v_1, v_2\rangle \mid \lambda\mathtt{x} : \tau.\, e \mid \mathtt{intro}_{\mu\mathtt{T}.\tau}(v). \end{array}$$

For closed τ, we define

$$\begin{array}{rcl} \text{Values}(\tau) & = & \{v \in \text{Values} \mid \ \vdash v : \tau\}, \\ \text{Programs}(\tau) & = & \{p \in \text{Expressions} \mid \ \vdash p : \tau\}. \end{array}$$

Convention. We write $\Theta, \Gamma \vdash e \rightsquigarrow v : \tau$ to indicate that $\Theta, \Gamma \vdash e : \tau$ is derivable and that $e \rightsquigarrow v$. We also write $\Theta, \Gamma \vdash e\checkmark : \tau$ when $\Theta, \Gamma \vdash e \rightsquigarrow v : \tau$ for some value v.

6.3 Categorical models

Essentially, the categorical structure needed to interpret FPC is a parameterised algebraically compact partial cartesian closed category with finite coproducts.

Definition 6.1 A $\mathcal{V}$*-model of FPC* is specified by

- a domain structure $(\mathcal{K}, \mathcal{D})$,
- a $\mathcal{V}$-enrichment for $p\mathcal{K}$,
- $\mathcal{V}$-functors $+, \otimes : p\mathcal{K} \times p\mathcal{K} \to p\mathcal{K}$ and $\underline{\rightharpoonup} : p\mathcal{K}^{op} \times p\mathcal{K} \to p\mathcal{K}$, and
- mappings $(\_)^\dagger$ and $\iota^{(-)}$ associating every $\mathcal{V}$-functor $F : \mathcal{X} \times p\mathcal{K} \to p\mathcal{K}$ with a $\mathcal{V}$-functor $F^\dagger : \mathcal{X} \to p\mathcal{K}$ and a natural transformation $\iota^F : F\langle \text{Id}, F^\dagger\rangle \xrightarrow{\cdot} F^\dagger$

such that

- the underlying functors of $+$, $\otimes$ and $\underline{\rightharpoonup}$ are, respectively, coproduct, partial product and partial exponential functors, and
- $p\mathcal{K}$ is parameterised $\mathcal{V}$-algebraically compact with parameterisation $\big((\_)^\dagger, \iota^{(-)}\big)$. $\square$

Remark. It follows that $p\mathcal{K}$ has zero object 0 (viz. a free $\text{Id}_{p\mathcal{K}}$-algebra), $\mathcal{K}$ has terminal object $1 = 0 \underline{\rightharpoonup} 0$ and $J \cong \_ \otimes 1 : \mathcal{K} \to p\mathcal{K}$ has right adjoint.

6.4 Denotational semantics

Until the end of this subsection fix a $\mathcal{V}$-model of FPC. Well-formed types, $\Theta \vdash \tau$, are interpreted as symmetric functors $[\![\Theta \vdash \tau]\!] : p\breve{\mathcal{K}}^{\,|\,\Theta\,|} \to p\breve{\mathcal{K}}$ and well-formed expressions, $\Theta, \Gamma \vdash e : \tau$, are interpreted as families of partial maps

$$[\![\Theta, \Gamma \vdash e : \tau]\!]_A : [\![\Theta \vdash \Gamma]\!](A)_2 \rightharpoonup [\![\Theta \vdash \tau]\!](A)_2$$

for $A \in |\, p\breve{\mathcal{K}}\,|^{\,|\,\Theta\,|}$ symmetric.

6.4.1 Interpretation of Types

Definition 6.2 $[\![\Theta \vdash \tau]\!] : p\breve{\mathcal{K}}^{\,|\,\Theta\,|} \to p\breve{\mathcal{K}}$.

- $[\![\Theta \vdash \Theta_i]\!] = \Pi_i \qquad (1 \le i \le |\,\Theta\,|)$.
- $[\![\Theta \vdash \tau_1 + \tau_2]\!] = (+ \circ \langle \Pi_2[\![\Theta \vdash \tau_1]\!], \Pi_2[\![\Theta \vdash \tau_2]\!]\rangle)\breve{}$.
- $[\![\Theta \vdash \tau_1 \times \tau_2]\!] = (\otimes \circ \langle \Pi_2[\![\Theta \vdash \tau_1]\!], \Pi_2[\![\Theta \vdash \tau_2]\!]\rangle)\breve{}$.
- $[\![\Theta \vdash \tau_1 \underline{\rightharpoonup} \tau_2]\!] = (\underline{\rightharpoonup} \circ \langle \Pi_1[\![\Theta \vdash \tau_1]\!], \Pi_2[\![\Theta \vdash \tau_2]\!]\rangle)\breve{}$.
- $[\![\Theta \vdash \mu\mathtt{T}.\tau]\!] = [\![\Theta, \mathtt{T} \vdash \tau]\!]^\ddagger$. $\square$

Remark. The parameterisation $(\_)^{\ddagger}$ on $p\breve{\mathcal{K}}$ appearing in the last item of the above definition is the one obtained, by Corollary 5.6 (2) and Theorem 5.7, from the given parameterisation on $p\mathcal{K}$.

By construction and because of Theorem 5.7, $[\![\Theta \vdash \tau]\!]$ is a symmetric $\mathcal{V}$-functor. The interpretation of types respects a substitution lemma (Lemma 6.3), establishing that evaluation is the semantic counterpart of syntactic substitution. This is needed to interpret expressions.

Lemma 6.3 (Substitution Lemma) There exists a canonical natural isomorphism

$$\beta : [\![\Theta \vdash \tau_1[\mathtt{T} \mapsto \tau_2]]\!] \cong [\![\Theta, \mathtt{T} \vdash \tau_1]\!]\langle \mathrm{Id}, [\![\Theta \vdash \tau_2]\!]\rangle$$

such that $\beta_A^{\S} = \beta_A^{-1}$ for every symmetric A. □

6.4.2 Interpretation of Expression Contexts

Definition 6.4 $[\![\Theta \vdash \Gamma]\!] : p\breve{\mathcal{K}}^{\mid \Theta \mid} \to p\breve{\mathcal{K}}$.

- $[\![\Theta \vdash \langle\rangle]\!] = K_{(1,1)}$ (the constantly $(1,1)$ functor).
- $[\![\Theta \vdash \Gamma, \mathtt{x} : \tau]\!] = (\otimes \circ \langle \Pi_2 [\![\Theta \vdash \Gamma]\!], \Pi_2 [\![\Theta \vdash \tau]\!]\rangle)\breve{}$. □

Remark. $[\![\Theta \vdash \Gamma]\!]$ is a symmetric $\mathcal{V}$-functor.

6.4.3 Interpretation of Expressions

The interpretation of expressions is defined in the standard way. Variables correspond to projections, `inl`/`inr` to coproduct injections, `case` to coproduct selection, $\langle \_, \_ \rangle$ to partial pairing, `fst`/`snd` to projections, $\lambda \mathtt{x}.\_$ to currying, $\_(\_)$ to evaluation and `intro`/`elim` to folding/unfolding a recursive type.

Definition 6.5 For symmetric $A \in \mid p\breve{\mathcal{K}} \mid^{\mid \Theta \mid}$ we define $[\![\Theta, \Gamma \vdash e : \tau]\!]_A$ as follows:

- $[\![\Theta, \Gamma \vdash \Gamma_i]\!]_A = \pi_i \qquad (1 \leq i \leq \mid \Gamma \mid)$.
- $[\![\Theta, \Gamma \vdash \mathtt{inl}(e) : \tau_1 + \tau_2]\!]_A = \amalg_1 \circ [\![\Theta, \Gamma \vdash e : \tau_1]\!]_A$.
- $[\![\Theta, \Gamma \vdash \mathtt{inr}(e) : \tau_1 + \tau_2]\!]_A = \amalg_2 \circ [\![\Theta, \Gamma \vdash e : \tau_2]\!]_A$.
- $[\![\Theta, \Gamma \vdash \mathtt{case}\ e\ \mathtt{of}\ \mathtt{inl}(\mathtt{x}_1).e_1\ \mathtt{or}\ \mathtt{inr}(\mathtt{x}_2).e_2 : \tau]\!]_A$
  $= \big[[\![\Theta, \langle \Gamma, \mathtt{x}_1 : \tau_1\rangle \vdash e_1 : \tau]\!]_A, [\![\Theta, \langle \Gamma, \mathtt{x}_2 : \tau_2\rangle \vdash e_2 : \tau]\!]_A \big]$
  $\circ\, \delta_A \circ \langle\!\langle \mathrm{id}, [\![\Theta, \Gamma \vdash e : \tau_1 + \tau_2]\!]_A \rangle\!\rangle$

  where δ_A is the canonical natural isomorphism

$$\begin{array}{l} [\![\Theta \vdash \Gamma]\!](A)_2 \otimes \big([\![\Theta \vdash \tau_1]\!](A)_2 + [\![\Theta \vdash \tau_2]\!](A)_2\big) \\ \cong \\ \big([\![\Theta \vdash \Gamma]\!](A)_2 \otimes [\![\Theta \vdash \tau_1]\!](A)_2\big) \\ + \\ \big([\![\Theta \vdash \Gamma]\!](A)_2 \otimes [\![\Theta \vdash \tau_1]\!](A)_2\big). \end{array}$$

- $[\![\Theta, \Gamma \vdash \langle e_1, e_2 \rangle : \tau_1 \times \tau_2]\!]_A$
  $= \langle\!\langle [\![\Theta, \Gamma \vdash e_1 : \tau_1]\!]_A, [\![\Theta, \Gamma \vdash e_2 : \tau_2]\!]_A \rangle\!\rangle$.
- $[\![\Theta, \Gamma \vdash \mathtt{fst}(e) : \tau_1]\!]_A = \pi_1 \circ [\![\Theta, \Gamma \vdash e : \tau_1 \times \tau_2]\!]_A$.
- $[\![\Theta, \Gamma \vdash \mathtt{snd}(e) : \tau_2]\!]_A = \pi_2 \circ [\![\Theta, \Gamma \vdash e : \tau_1 \times \tau_2]\!]_A$.
- $[\![\Theta, \Gamma \vdash \lambda : \tau_1 \mathtt{x}.e : \tau_1 \rightharpoonup \tau_2]\!]_A$
  $= p\lambda([\![\Theta, \langle \Gamma, \mathtt{x} : \tau_1 \rangle \vdash e : \tau_2]\!]_A)$.
- $[\![\Theta, \Gamma \vdash e(e_1) : \tau_2]\!]_A$
  $= \varepsilon \circ \langle\!\langle [\![\Theta, \Gamma \vdash e : \tau_1 \rightharpoonup \tau_2]\!]_A, [\![\Theta, \Gamma \vdash e_1 : \tau_1]\!]_A \rangle\!\rangle$.
- $[\![\Theta, \Gamma \vdash \mathtt{intro}(e) : \mu\mathtt{T}.\tau]\!]_A$
  $= I_A \circ [\![\Theta, \Gamma \vdash e : \tau[\mathtt{T} \mapsto \mu\mathtt{T}.\tau]]\!]_A$
  where $I_A = (\varphi_A^{[\![\Theta,\ \mathtt{T} \vdash \tau]\!]} \circ \beta_A)_2$.
- $[\![\Theta, \Gamma \vdash \mathtt{elim}(e) : \tau[\mathtt{T} \mapsto \mu\mathtt{T}.\tau]]\!]_A$
  $= E_A \circ [\![\Theta, \Gamma \vdash e : \mu\mathtt{T}.\tau]\!]_A$
  where $E_A = (\varphi_A^{[\![\Theta,\ \mathtt{T} \vdash \tau]\!]} \circ \beta_A)_1$. □

Remark. By Theorem 5.7 and Lemma 6.3, I_A and E_A are mutual inverses.

The interpretation of expressions is well-defined:

Proposition 6.6 For $A \in \mid p\breve{\mathcal{K}} \mid^{\mid \Theta \mid}$ symmetric,

$$[\![\Theta, \Gamma \vdash e : \tau]\!]_A : [\![\Theta \vdash \Gamma]\!](A)_2 \rightharpoonup [\![\Theta \vdash \tau]\!](A)_2. \quad □$$

Remark. It can be shown [Fio94] that the interpretation of expressions in **Poset**-models induced by domain structures with a **Poset**-category of total maps is parametric with respect to representations [Rey74].

6.5 Domain-theoretic models

In this subsection, the results of Sections 4 and 5 are combined to produce **Cpo**-models of FPC. Domain-theoretic models correspond to uniform domain structures with a **Cpo**-category of total maps whose category of partial maps is a **Cpo**-model with respect to the induced order (Definition 3.3) and where algebraic compactness arises from the limit/colimit coincidence.

Definition 6.7 A *domain-theoretic* model of FPC is specified by a uniform domain structure $(\mathcal{K}, \mathcal{D})$ and a **Cpo**-enrichment $\sqsubseteq$ for $\mathcal{K}$ satisfying:

DTM1. $(\mathcal{K}, \sqsubseteq)$ has chosen binary **Poset**-coproducts.

DTM2. $(\mathcal{K}, \sqsubseteq)$ has chosen binary **Poset**-products.

DTM3. $(p\mathcal{K}, \underset{\sim}{\sqsubset})$ is a **Poset**-category with chosen **Poset**-partial-exponentials.

DTM4. $p\mathcal{K}$ has a chosen zero object.

DTM5. $\mathcal{K}$ has chosen colimits of ω-chains of embeddings. □

Theorem 6.8 [Fio94] If $(\mathcal{K}, \mathcal{D}, \sqsubseteq)$ is a domain-theoretic model of FPC then $(p(\mathcal{K}, \mathcal{D}), \underset{\sim}{\sqsubset})$ is a **Cpo**-model of FPC. □

Example 6.9 For $\mathcal{W}$ a small **Cpo**-category, let $([\mathcal{W}, \mathbf{Cpo}], \sqsubseteq)$ be the **Cpo**-functor-category of **Cpo**-functors $\mathcal{W} \to \mathbf{Cpo}$ and natural transformations ordered pointwise, and let $\mathbf{\Sigma C}_{[\mathcal{W},\mathbf{Cpo}]}$ be the subcategory of $[\mathcal{W}, \mathbf{Cpo}]$ of all pointwise-admissible and cartesian natural transformations. Then, $([\mathcal{W}, \mathbf{Cpo}], \mathbf{\Sigma C}_{[\mathcal{W},\mathbf{Cpo}]}, \sqsubseteq)$ specifies a domain-theoretic model. □

7 Computational adequacy

The relationship between the operational notion of termination and the denotational notion of totality is examined. It is shown that in any model of FPC the interpretation of expressions is *computationally sound*, meaning that if the evaluation of a program terminates then it has a total interpretation. Further, certain *absolute* domain-theoretic models are shown to be *computationally adequate*, meaning that if a program has a total interpretation then its evaluation terminates.

7.1 Computational Soundness

The strategy for proving computational soundness is as usual: observe that values have total interpretations and that the interpretation of programs is preserved under evaluation:

Lemma 7.1 In any $\mathcal{V}$-model of FPC,

1. $[\![\Theta, \Gamma \vdash v : \tau]\!]_A \downarrow$ for every value v.

2. if $\Theta, \Gamma \vdash e \rightsquigarrow v : \tau$ then $[\![\Theta, \Gamma \vdash e : \tau]\!]_A = [\![\Theta, \Gamma \vdash v : \tau]\!]_A$. □

Theorem 7.2 (Computational Soundness) In any $\mathcal{V}$-model of FPC, if $\Theta, \Gamma \vdash e\checkmark : \tau$ then $[\![\Theta, \Gamma \vdash e : \tau]\!]_A \downarrow$. □

7.2 Adequacy

Convention. We write $[\![\vdash \tau]\!]$ for $[\![\vdash \tau]\!]()$ and $[\![\vdash p : \tau]\!]$ for $[\![\vdash p : \tau]\!]_{()}$.

Definition 7.3 A $\mathcal{V}$-model of FPC is *computationally adequate* if, for every $p \in \text{Programs}(\tau)$,

$$[\![\vdash p : \tau]\!] \downarrow \text{ implies } \vdash p\checkmark : \tau. \qquad \square$$

Fix a domain-theoretic model of FPC. Computational adequacy cannot be proved directly by induction. To establish it (see Lemma 7.13), we inductively prove the following stronger version

$$[\![\vdash p : \tau]\!] \downarrow \Rightarrow \ \exists\, v.\, p \rightsquigarrow v \ \wedge \ [\![\vdash p : \tau]\!] \preceq_\tau v$$

where $\left\{ \preceq_\tau \ \subseteq \ \mathcal{K}(1, [\![\vdash \tau]\!]_2) \times \text{Values}(\tau) \right\}$ is a family of *formal-approximation* relations with the following closure properties:

R1. $x \preceq_{\sigma+\tau} \texttt{inl}(v) \Leftrightarrow \exists\, x' \preceq_\sigma v.\ x = \amalg_1 \circ x'$

$x \preceq_{\sigma+\tau} \texttt{inr}(v) \Leftrightarrow \exists\, x' \preceq_\tau v.\ x = \amalg_2 \circ x'$

R2. $x \preceq_{\sigma\times\tau} \langle v_1, v_2 \rangle$
$\Leftrightarrow$
$(\pi_1 \circ x \preceq_\sigma v_1) \ \wedge \ (\pi_2 \circ x \preceq_\tau v_2)$

R3. $f \preceq_{\sigma \rightharpoonup \tau} \lambda \texttt{x} : \sigma.\, e$
$\Leftrightarrow$
$(\forall\, x \in \mathcal{K}(1, [\![\vdash \sigma]\!]_2).\ \forall\, v \in \text{Values}(\sigma).$
$x \preceq_\sigma v \ \Rightarrow \ \varepsilon \circ \langle\!\langle f, x \rangle\!\rangle \precsim_\tau e[\texttt{x} \mapsto v])$

where $u \precsim p \Leftrightarrow (u \downarrow \Rightarrow \ \exists\, v.\, p \rightsquigarrow v \ \wedge \ u \preceq v)$.

R4. $x \preceq_{\mu\texttt{T}.\,\tau} \texttt{intro}(v) \Leftrightarrow E \circ x \preceq_{\tau[\texttt{T} \mapsto \mu\texttt{T}.\,\tau]} v$.

We now embark on the major task of establishing the existence of the formal-approximation relations. The difficulty is that they are defined recursively and one should guarantee that this definition makes sense. This is better discussed with the aid of some definitions.

For closed τ, generalising from the formal-approximation relations, we set

$$|\,\mathcal{R}(\tau)\,| = \{(A, \preceq) \mid A \in |\,\mathcal{K}\,| \wedge \preceq \subseteq \mathcal{K}(1, A) \times \text{Values}(\tau)\}$$

and given $\preceq \ \subseteq \ \mathcal{K}(1, A) \times \text{Values}(\tau)$, we define its extension to partial maps and programs $\precsim \subseteq p\mathcal{K}(1, A) \times \text{Programs}(\tau)$, by

$$u \precsim p \Leftrightarrow (u \downarrow \Rightarrow \ \exists\, v.\, p \rightsquigarrow v \ \wedge \ u \preceq v).$$

Now, to provide an equational description of the closure properties of $\preceq_{\sigma\star\tau}$ ($\star \in \{+, \times, \rightharpoonup\}$) and to clarify the closure property of $\preceq_{\mu\texttt{T}.\tau}$, the type constructors are *logically* extended to relations following (R1)–(R3):

Definition 7.4 For closed σ and τ,

1. Let $+_{\sigma,\tau} : |\,\mathcal{R}(\sigma)\,| \times |\,\mathcal{R}(\tau)\,| \to |\,\mathcal{R}(\sigma+\tau)\,|$ be the function mapping $\big((A,\preceq_A),(B,\preceq_B)\big)$ to $(A+B, \preceq_A + \preceq_B)$ where
$$x\ (\preceq_A + \preceq_B)\ \mathtt{inl}(v) \iff (\exists\, x' \preceq_A v.\ x = \amalg_1 \circ x')$$
and
$$x\ (\preceq_A + \preceq_B)\ \mathtt{inr}(v) \iff (\exists\, x' \preceq_B v.\ x = \amalg_2 \circ x').$$
2. Let $\otimes_{\sigma,\tau} : |\,\mathcal{R}(\sigma)\,| \times |\,\mathcal{R}(\tau)\,| \to |\,\mathcal{R}(\sigma\times\tau)\,|$ be the function mapping $\big((A,\preceq_A),(B,\preceq_B)\big)$ to $(A\otimes B, \preceq_A \otimes \preceq_B)$ where
$$\begin{array}{l} x\ (\preceq_A \otimes \preceq_B)\ \langle v_1, v_2\rangle \\ \iff \\ (\pi_1 \circ x \preceq_A v_1)\ \wedge\ (\pi_2 \circ x \preceq_B v_2). \end{array}$$
3. Let $\rightharpoonup_{\sigma,\tau} : |\,\mathcal{R}(\sigma)\,| \times |\,\mathcal{R}(\tau)\,| \to |\,\mathcal{R}(\sigma \rightharpoonup \tau)\,|$ be the function mapping $\big((A,\preceq_A),(B,\preceq_B)\big)$ to $(A \rightharpoonup B, \preceq_A \rightharpoonup \preceq_B)$ where
$$\begin{array}{l} f\ (\preceq_A \rightharpoonup \preceq_B)\ \lambda \mathtt{x}:\sigma.\, e \\ \iff \\ (\forall\, x \in \mathcal{K}(1,A).\ \forall\, v \in \mathrm{Values}(\sigma). \\ \qquad x \preceq_A v\ \Rightarrow\ \varepsilon \circ \langle\!\langle f, x\rangle\!\rangle \precsim_B e[\mathtt{x} \mapsto v]). \end{array}$$ □

Then, (R1)–(R3) can be written as
$$\begin{aligned} \preceq_{\sigma+\tau} &= \preceq_\sigma + \preceq_\tau, \\ \preceq_{\sigma\times\tau} &= \preceq_\sigma \otimes \preceq_\tau, \\ \preceq_{\sigma\rightharpoonup\tau} &= \preceq_\sigma \rightharpoonup \preceq_\tau\ . \end{aligned}$$
In the case of recursive types (R4) we intuitively have that
$$E:\ \preceq_{\mu\mathtt{T}.\tau} \cong \mathbb{I}\,(\preceq_{\tau[\mathtt{T}\mapsto\mu\mathtt{T}.\tau]}) \qquad (1)$$
where $x\ (\mathbb{I} \preceq)\ \mathtt{intro}(v) \iff x \preceq v$. And this is exactly where the difficulty in constructing $\preceq_{\mu\mathtt{T}.\,\tau}$ resides as we have to solve a recursive equation between relations!

We make $|\,\mathcal{R}(\tau)\,|$ into the objects of a category; the notion of morphism is suggested by (1):

Definition 7.5 The category $\mathcal{R}(\tau)$ has objects $|\,\mathcal{R}(\tau)\,|$ and morphisms $u : (A,\preceq_A) \rightharpoonup (B,\preceq_B)$ where $u : A \rightharpoonup B$ is such that for every $x \in \mathcal{K}(1,A)$ and every $v \in \mathrm{Values}(\tau)$, if $x \preceq_A v$ then $u\circ x \precsim_B v$. Composition and identities are as in $p\mathcal{K}$. □

Proposition 7.6 The functions $+_{\sigma,\tau}$, $\otimes_{\sigma,\tau}$ and $\rightharpoonup_{\sigma,\tau}$ extend to functors satisfying
$$\begin{aligned} U_{\sigma+\tau} \circ +_{\sigma,\tau} &= + \circ (U_\sigma \times U_\tau), \\ U_{\sigma\times\tau} \circ \otimes_{\sigma,\tau} &= \otimes \circ (U_\sigma \times U_\tau), \\ U_{\sigma\rightharpoonup\tau} \circ \rightharpoonup_{\sigma,\tau} &= \rightharpoonup \circ (U_\sigma^{op} \times U_\tau) \end{aligned}$$
where $U_\tau : \mathcal{R}(\tau) \to p\mathcal{K}$ is the evident forgetful functor. □

The type constructors on $p\mathcal{K}$ have been extended to the $\mathcal{R}(\tau)$'s but there are too many relations in them to be able to find recursive types. The way out is to find subcategories $p\mathcal{K}(\tau)$ of $\mathcal{R}(\tau)$ in which this is possible.

Definition 7.7 Let $p\mathcal{K}(\tau)$ be the full subcategory of $\mathcal{R}(\tau)$ consisting of all those objects $(A,\preceq)$ such that for every $v \in \mathrm{Values}(\tau)$, $\preceq^{-1}(v)$ is closed under lubs of ω-chains in $p\mathcal{K}(1,A)$. □

In order that the notion of approximation on the $p\mathcal{K}(\tau)$'s be inherited from $p\mathcal{K}$ (i.e. that the forgetful functor $p\mathcal{K}(\tau) \to p\mathcal{K}$ be a **Cpo**-functor) we restrict attention to absolute models:

Definition 7.8 A **Cpo**-model of FPC is *absolute* if for every $A \in |\,\mathcal{K}\,|$, $\mathcal{K}(1,A)$ is inaccessible by lubs of ω-chains in $p\mathcal{K}(1,A)$. □

In domain-theoretic models absoluteness has a simple characterisation:

Proposition 7.9 A domain-theoretic model of FPC is absolute iff $[\mathrm{id}_1]$ is inaccessible by lubs of ω-chains in $\mathcal{D}(1)$. □

Remark. The above could have been stated as: the $\mathcal{D}$-classifier $1 \rightarrowtail \Sigma$ (i.e. the counit of the representation $\mathcal{K}(\_,\Sigma) \cong p\mathcal{K}(\_,1) \cong \mathcal{D}(\_) : \mathcal{K}^{op} \to \mathbf{Set}$) is inaccessible by lubs of ω-chains in $\mathcal{K}(1,\Sigma)$.

Example 7.10 For a small **Cpo**-category $\mathcal{W}$, the domain-theoretic model $([\mathcal{W},\mathbf{Cpo}], \mathbf{\Sigma C}_{[\mathcal{W},\mathbf{Cpo}]})$ is absolute iff $\mathcal{W}$ has a finite number of connected components. □

From now on let our fixed domain-theoretic model of FPC be non-trivial and absolute. Then, we have the following lemma which displays the essential categorical structure needed to prove the existence of the formal-approximation relations.

Lemma 7.11

A1. $(p\mathcal{K}(\tau), \underset{\sim}{\sqsubseteq})$ is a **Cpo**-category and the forgetful functor $p\mathcal{K}(\tau) \to p\mathcal{K}$ **Cpo**-enriches.

A2. $(p\mathcal{K}(\tau), \underset{\sim}{\sqsubseteq})$ is parameterised **Cpo**-algebraically compact.

A3. For $\star = +, \otimes$ (resp. $\rightharpoonup$), the functor $\star_{\sigma,\tau}$ restricts to a **Cpo**-functor $p\mathcal{K}(\sigma) \times p\mathcal{K}(\tau) \to p\mathcal{K}(\sigma \star \tau)$ $\big($resp. $p\mathcal{K}(\sigma)^{op} \times p\mathcal{K}(\tau) \to p\mathcal{K}(\sigma\rightharpoonup\tau)\big)$.

A4. $\mathbb{I}_{\mu\mathtt{T}.\,\tau} \ : \ p\mathcal{K}(\tau[\mathtt{T} \mapsto \mu\mathtt{T}.\,\tau]) \ \cong \ p\mathcal{K}(\mu\mathtt{T}.\,\tau) \ : \ \mathbb{E}_{\mu\mathtt{T}.\,\tau}$ where

$$\begin{array}{lll} \mathbb{I}_{\mu\mathtt{T}.\,\tau} : (A, \preceq) & \mapsto & (A, \mathbb{I} \preceq) \\ \quad \text{with } x \ (\mathbb{I} \preceq) \ \mathtt{intro}(v) \Leftrightarrow x \preceq v & & \end{array}$$

and

$$\begin{array}{lll} \mathbb{E}_{\mu\mathtt{T}.\,\tau} : (A, \preceq) & \mapsto & (A, \mathbb{E} \preceq) \\ \quad \text{with } x \ (\mathbb{E} \preceq) \ v \Leftrightarrow x \preceq \mathtt{intro}(v). & & \end{array}$$

A5. [Strict Uniformity] For every pair of symmetric **Cpo**-functors F and G, if

$$\begin{tikzcd} p\mathcal{K}(\sigma_1)\breve{} \times \ldots \times p\mathcal{K}(\sigma_m)\breve{} \times p\mathcal{K}(\sigma)\breve{} \arrow[r, "F"] \arrow[d] & p\mathcal{K}(\sigma)\breve{} \arrow[d] \\ \breve{p\mathcal{K}} \times \ldots \times \breve{p\mathcal{K}} \times \breve{p\mathcal{K}} \arrow[r, "G"'] & \breve{p\mathcal{K}} \end{tikzcd}$$

commutes then for every initial parameterised G-algebra $(G^\dagger, \iota^G)$ there exists an initial parameterised F-algebra $(F^\dagger, \iota^F)$ such that

$$\begin{tikzcd} p\mathcal{K}(\sigma_1)\breve{} \times \ldots \times p\mathcal{K}(\sigma_m)\breve{} \arrow[r, bend left, "F\langle \mathrm{Id}, F^\dagger \rangle"] \arrow[r, bend right, "F^\dagger"'] \arrow[r, phantom, "\overset{\iota^F}{\cong}"] \arrow[d] & p\mathcal{K}(\sigma)\breve{} \arrow[d] \\ \breve{p\mathcal{K}} \times \ldots \times \breve{p\mathcal{K}} \arrow[r, bend left, "G\langle \mathrm{Id}, G^\dagger \rangle"] \arrow[r, bend right, "G^\dagger"'] \arrow[r, phantom, "\overset{\iota^G}{\cong}"] & \breve{p\mathcal{K}} \end{tikzcd}$$

commutes. $\square$

Once the Properties (A1)–(A5) are known to hold, the formal-approximation relations are constructed by providing a *non-standard interpretation of types* in the $p\mathcal{K}(\tau)$'s (Proposition 7.12) which extends the interpretation of types (Definition 6.2) respecting the folding and unfolding of recursive types and the substitution of types.

Convention. Given $\Theta \equiv \mathtt{T}_1, \ldots, \mathtt{T}_m$ and closed σ_i $(1 \leq i \leq m)$ we write $\vec{\sigma}$ for $\sigma_1, \ldots, \sigma_m$ and $[\Theta \mapsto \vec{\sigma}]$ for $[\mathtt{T}_1 \mapsto \sigma_1, \ldots, \mathtt{T}_m \mapsto \sigma_m]$.

Proposition 7.12 There exists an interpretation

$$\langle\!|\Theta \vdash \tau|\!\rangle(\vec{\sigma}) : p\mathcal{K}(\sigma_1)\breve{} \times \ldots \times p\mathcal{K}(\sigma_{|\Theta|})\breve{} \to p\mathcal{K}(\tau[\Theta \mapsto \vec{\sigma}])\breve{}$$

such that $\langle\!|\Theta \vdash \tau|\!\rangle(\vec{\sigma})$ is a symmetric **Cpo**-functor and

$$\begin{tikzcd} p\mathcal{K}(\sigma_1)\breve{} \times \ldots \times p\mathcal{K}(\sigma_{|\Theta|})\breve{} \arrow[r, "\langle\!|\Theta \vdash \tau|\!\rangle(\vec{\sigma})"] \arrow[d] & p\mathcal{K}(\tau[\Theta \mapsto \vec{\sigma}])\breve{} \arrow[d] \\ \breve{p\mathcal{K}} \times \ldots \times \breve{p\mathcal{K}} \arrow[r, "\llbracket \Theta \vdash \tau \rrbracket"'] & \breve{p\mathcal{K}} \end{tikzcd}$$

commutes. $\square$

From the non-standard interpretation of types, the formal-approximation relations are easily extracted as the second component of $\langle\!|\vdash \tau|\!\rangle()_2$.

The following main lemma is now proved by induction on the structure of e:

Lemma 7.13 For $\Gamma \equiv \mathtt{x}_1 : \tau_1, \ldots, \mathtt{x}_n : \tau_n$, if $x_i \preceq_{\tau_i} v_i$ $(1 \leq i \leq n)$ then

$$\llbracket \Gamma \vdash e : \tau \rrbracket \circ \langle\!\langle x_1, \ldots, x_n \rangle\!\rangle \precsim_\tau e[\mathtt{x}_1 \mapsto v_1, \ldots, \mathtt{x}_n \mapsto v_n]. \square$$

Finally, since $\llbracket \vdash p : \tau \rrbracket \precsim_\tau p$, we conclude:

Theorem 7.14 (Computational Adequacy) Every absolute non-trivial domain-theoretic model of FPC is computationally adequate. $\square$

Remark. Using (A3), domain structures with a **Cpo**-category of total maps satisfying the axioms (DTM1)–(DTM4) and absoluteness, and equipped with structure for interpreting natural numbers and booleans can be shown to be computationally adequate for call-by-value PCF with sums and products (c.f. [BCL85]).

Corollary 7.15 For every non-empty small **Cpo**-category $\mathcal{W}$, the domain-theoretic model specified by $([\mathcal{W}, \mathbf{Cpo}], \mathbf{\Sigma C}_{[\mathcal{W}, \mathbf{Cpo}]})$ is computationally adequate. $\square$

Remark. Thus there are computationally-adequate *non*-absolute domain-theoretic models.

Acknowledgements. We are grateful to Barry Jay and John Power for discussions on categorical matters.

References

[Abr90] S. Abramsky. The lazy λ-calculus. In *Logical Foundations of Functional Programming*. Addison-Wesley, 1990.

[Adá93] J. Adámek. Data types in algebraically ω-complete categories. To appear in Information and Computation, 1993.

[Bar92] M. Barr. Algebraically compact functors. *Journal of Pure and Applied Algebra*, 82:211–231, 1992.

[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 University Press, 1985.

[Fio93] M.P. Fiore. **Cpo**-categories of partial maps. In *CTCS-5 (Category Theory and Computer Science Fifth Biennial Meeting)*, pages 45–49. CWI, September 1993.

[Fio94] M.P. Fiore. Axiomatic domain theory in categories of partial maps. Forthcoming Ph.D. thesis, 1994.

[Fre90] P.J. Freyd. Recursive types reduced to inductive types. In *5th LICS Conf.*, pages 498–507. IEEE, Computer Society Press, 1990.

[Fre91] P.J. Freyd. Algebraically complete categories. In A. Carboni, M.C. Pedicchio, and G. Rosolini, editors, *Category Theory*, volume 1488 of *Lecture Notes in Mathematics*, pages 131–156. Springer-Verlag, 1991.

[Fre92] P.J. Freyd. Remarks on algebraically compact categories. In M.P. Fourman, P.T. Johnstone, and A.M. Pitts, editors, *Applications of Categories in Computer Science*, volume 177 of *London Mathematical Society Lecture Note Series*, pages 95–106. Cambridge University Press, 1992.

[Gun92] C.A. Gunter. *Semantics of Programming Languages: Structures and Techniques.* The MIT Press, 1992.

[Kel82] G.M. Kelly. *Basic Concepts of Enriched Category Theory.* Cambridge University Press, 1982.

[LM84] G. Longo and E. Moggi. Cartesian closed categories of enumerations for effective type-structures. In G. Kahn, D. MacQueen, and G. Plotkin, editors, *Symposium on Semantics of Data Types*, volume 173 of *Lecture Notes in Computer Science.* Springer-Verlag, 1984.

[MC88] A.R. Meyer and S.S. Cosmadakis. Semantical paradigms: Notes for an invited lecture. In *3rd LICS Conf.*, pages 236–253. IEEE, Computer Society Press, 1988.

[Mog86] E. Moggi. Categories of partial morphisms and the partial lambda-calculus. In *Proceedings Workshop on Category Theory and Computer Programming, Guildford 1985*, volume 240 of *Lecture Notes in Computer Science*, pages 242–251. Springer-Verlag, 1986.

[Mog89] E. Moggi. Computational lambda-calculus and monads. In *4th LICS Conf.*, pages 14–23, 1989.

[Pit93] A.M. Pitts. Relational properties of domains. Technical Report 321, Cambridge Univ. Computer Laboratory, December 1993.

[Plo85] G.D. Plotkin. Denotational semantics with partial functions. Lecture at C.S.L.I. Summer School, 1985.

[Plo93] G.D. Plotkin. Type theory and recursion (extended abstract). In *8th LICS Conf.*, page 374. IEEE, Computer Society Press, 1993.

[Rey74] J. Reynolds. Towards a theory of type structure. In B. Robinet, editor, *Programming Symposium '74*, volume 19 of *Lecture Notes in Computer Science.* Springer-Verlag, 1974.

[Ros86] G. Rosolini. *Continuity and Effectiveness in Topoi.* PhD thesis, University of Oxford, 1986.

[RR88] E. Robinson and G. Rosolini. Categories of partial maps. *Information and Computation*, 79:95–130, 1988.

[SP82] M.B. Smyth and G.D. Plotkin. The category-theoretic solution of recursive domain equations. *SIAM Journal of Computing*, 11(4):761–783, November 1982.

[Tay91] P. Taylor. The fixed point property in synthetic domain theory. In *6th LICS Conf.*, pages 152–160. IEEE, Computer Society Press, 1991.