Some syntactic and categorical constructions of lambda-calculus models

Gérard Berry

INRIA

CENTRE
SOPHIA ANTIPOLIS

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P. 105
78153 Le Chesnay Cedex
France
Tél.: 954 90 20

Rapports de Recherche

N° 80

SOME SYNTACTIC AND CATEGORICAL CONSTRUCTIONS OF LAMBDA-CALCULUS MODELS

Gérard BERRY

Juin 1981

SOME SYNTACTIC AND CATEGORICAL CONSTRUCTIONS OF LAMBDA-CALCULUS MODELS

Gérard BERRY

Centre de Mathematiques Appliquees
Ecole des Mines de Paris
Sophia-Antipolis
06560 Valbonne - FRANCE

ABSTRACT

We present a new definition of pure or typed lambda-calculus models together with new syntactic and semantic model constructions. We motivate our definition by the construction of structures which are not models in the usual definitions although they should definitilely be (in our opinion). The syntactic constructions concern initial and fully abstract models in the sense of Milner. The semantic constructions show how to obtain models from cartesian closed categories. In all cases we put a special emphasis on the additional properties of continuity and approximation continuity.

RESUME

Nous donnons une nouvelle definition des modeles des lambda-calculs purs ou types et de nouvelles constructions syntaxiques et semantiques de modeles. Nous motivons notre definition en construisant des structures qui ne satisfont pas les definitions classiques alors qu'elles devraient clairement etre des modeles. Nos constructions syntaxiques produisent des modeles initiaux et des modeles completement adequats au sens de Milner. Sur le plan semantique nous montrons comment obtenir des modeles a partir de categories cartesien fermees. Dans tout les cas nous etudions plus specialement les proprietes de continuite et de continuite par rapport aux approximations.

Introduction

There are two main methods for constructing models of λ-calculi or programming languages. We shall call the first one the syntactic method. It is characterized by the fact that the elements of the model are derived from the terms themselves. They may be for example equivalence classes of terms, infinite terms etc. [1,4,14,32]. On the other hand, in the semantic method the objects have some intuitive meaning on their own, independently of the modelled language. They may be integers, functions, relations etc. [6,26,28,29]. Our purpose here is to present some syntactic and semantic model constructions for the free lambda calculus and for Milner LCF languages [19,20,24], which we take as examples of "programming languages" (these languages are typed λ-calculi with first-order primitives and recursion operators). The syntactic constructions will concern initial models and fully abstract models [20]. The semantic construction will show how to obtain models from cartesian closed categories. In all cases we shall be particularly interested in some additional properties of the models, namely continuity and approximation continuity (a continuous model is said to be approximation continuous when the value of any term is the lub of the values of its syntactic approximations [14,31]. The models D^{∞}, P^{ω}, T^{ω}, [28,29 ,25 ,31] are typical examples of approximation continuous models).

Of course the first thing to do is to define the notions of model, continuous model and approximation continuous model. In most classical definitions one introduces a value domain, an environment domain which is the set of functions from variables to values, and one interprets any term as a function from environments to values in the traditional logical style. We shall not follow this technique, which has in our opinion two major drawbacks :

- When considering models which are not extensional, i.e. do not consist of functions, the binding of a variable by a λ in a context may be of a different nature from its binding by an environment, which is always extensional by definition.

This is certainly odd, since environments are used for passing bindings through lambdas. Hindley-Longo [9] introduces an extra condition in order to have also extensional binding by lambdas. It is met by models such as P^{ω} or T^{ω} [29, 25], but not by some other models such as the sequential algorithms model of [6].

- For continuous and approximation continuous models one is lead to put some extra structure on the interpretation domains (typically a cpo structure). It is often impossible to give the required extra structure to the set of functions or even continuous functions from environments to values, while it may be easy to do it for other sets. Typically in the sequential algorithms model of [6] it is obviously natural to interpret a term as an algorithm from environments to values and not as a function.

The solution we propose is a simple one. For example in the pure λ-calculus we add to the usual combinatory algebra $< D, . >$ a set E and a map $eval = Ex\ Env \longrightarrow D$. The terms are interpreted in E and their value in environments is computed through eval, more or less like in LISP [16]. The extra structure is added by a suitable choice of E. The classical models are just recovered by taking E as the set of functions from environment to values and eval as functional application, although we shall see that this choice is rarely interesting. The notion of morphism of models becomes quite straightforward. Notice that the definition is actually quite different in spirit from those of [1, 9, 18]. In particular our models may very well satisfy the η-rule without being extensional, which is impossible in [1 , 9]. Again the sequential algorithms of [6] are an example of such a non-extensional η-model. See [7] for a comparizon between the model definitio

Our first constructions are straightforward and concern initial models. The initial (η-) model is simply obtained by taking E and D as the (η-) interconvertibility classes of open and closed terms, eval being just ordinary substitution. The initial approximation continuous model is obtained in the same way from Böhm-trees [1 , 14]. A more interesting model is the initial approximation continuous η-model for PL languages. Here we use the fact that the η-expansion always terminates in typed

λ-calculi [10], so that any η-equivalence class of Böhm trees has a canonical element in maximal η-expansion. The set of η-expanded Böhm trees has many interesting properties : it is closed by abstraction, application, substitution and β-conversion. From this one checks that η-expanded Böhm trees form the required model. Moreover we show in theorem 3.3.9. that this model is order extensional when the language has enough basic function symbols, in other words that $A < A'$ holds iff $AB < A'B$ for all B when A and A' are Böhm trees of the same type. This new result has some similarities with Böhm's theorem [8] and will be reused later on.

The second constructions concern fully abstract models of PL, as introduced in [20,24]. We show how to extend Milner's construction [20] for combinatory logic to λ-calculus, which is not entirely straightforward (notice in particular that contexts may bind free variables, which is of course impossible in combinatory logic. The proof of Milner's first context lemma is therefore much more delicate). Moreover we give new fully abstract model constructions for the initial (or free, or symbolic) interpretation to which Milner's construction does not apply. We notice that the initial approximation continuous η-model constructed above is fully abstract w.r.t. the initial interpretation because of the theorem 3.3.9. As a consequence we see that <u>any</u> model of the symbolic interpretation is fully abstract, contrarily to what happened in [20] where the fully abstract model was unique.

The last construction concerns categorical models. We obtain a model from any retraction $\theta = C \to (C \to C)$ in a cartesian closed category. We show that it always satisfies η when θ is an isomorphism, without needing to be extensional.

The main application of these results are to the categories of stable functions [3,4], for which a classical proof by extensionnality is possible but tedious, and to the category of sequential algorithms on concrete data structures [6], where the categorical proof is really necessary since the objects are not functions anymore.

Clearly the idea of relating the Lambda-calculi and cartesian closed categories is not a new one. The first explicit relation to our knowledge appears in Lambeck [13], who shows the connexion between proofs in intuitionistic logic, combinatory algebras and cartesian closed category. (Notice that the "polynomial equality" between expressions considered in [13] is close to our model equality and is not defined extensionnally on environments). Categorical constructions are implicit in Scott's construction of D^{∞}, and are made explicit by Wand [33] and Smyth-Plotkin [30] (who also propose categorical treatment of universal domains such as P^{ω} or T^{ω}). But the property of being a model is generally proved by some extensionnality arguments [25, 28, 29]. We show here that these arguments are indeed not necessary, and claim that the cartesian closure is really the key property for semantic model constructions. Notice that the sequential algorithms of [6] are in no way extensional, so that the classical proofs by extensionality do not work for them.

The first section recalls necessary preliminaries. In the second section we present our model definitions. The third section is devoted to the study of initial models, while we give fully abstract model constructions in the fourth section. The categorical constructions are presented in the last section.

1. PRELIMINARIES

We recall briefly some basic definitions about partial orders. We introduce the syntax and basic properties of our two languages Λ and PL, together with their notions of syntactic approximation. We follow the definitions of [1,4,14,31]

1.1. Complete partial orders

Let $\langle \mathcal{D}, \leq \perp \rangle$ be a partial order with least element $\perp$. A nonempty subset $\Delta \subset \mathcal{D}$ is *directed* if for all $x, y \in \Delta$ there exists $z \in \Delta$ such that $x \leq z$ and $y \leq z$, and $\mathcal{D}$ is a *complete partial order* or *cpo* if any directed $\Delta \subset \mathcal{D}$ has a least upper bound (or *lub*) $\cup\Delta$. Given two cpo's $\mathcal{D}$ and $\mathcal{D}'$, a function $h: \mathcal{D} \to \mathcal{D}'$ is *continuous* if it is monotonic (increasing) and if it preserves lub's of directed sets. A *strict function* or *morphism of cpo's* is a continuous function $h: \mathcal{D} \to \mathcal{D}'$ such that $h(\perp) = \perp$. Let $h: \mathcal{D} \to \mathcal{D}'$ be continuous. A *fixpoint* of h is an $x \in \mathcal{D}$ such that $h(x) = x$. The element $Yh = \bigcup_n h^n(\perp)$ is the least fixpoint of h.

Let $\langle \mathcal{D}, \leq, \perp \rangle$ be a partial order. A *directed ideal* $I \subset \mathcal{D}$ is a directed set such that $x \in \mathcal{D}$, $y \in I$, $x \leq y$ imply $x \in I$. The completion by ideals $\langle \mathcal{D}^\infty, \leq, \perp \rangle$ is the set of directed ideals of $\mathcal{D}$ ordered by inclusion. For $x \in \mathcal{D}$, let $i(x) = \{y \leq x\} \in \mathcal{D}^\infty$. Then $\mathcal{D}^\infty$ is characterized (up to isomorphism) by the fact that for any cpo $\mathcal{D}'$ any monotonic $h: \mathcal{D} \to \mathcal{D}'$ extends in a unique way into a continuous $h^\infty: \mathcal{D}^\infty \to \mathcal{D}'$ such that $h = h^\infty \circ i$.

○

1.2. The pure λ-calculus

The λ-calculus Λ is generated from a set $V = \{x, y, \ldots\}$ of variables by $x \in \Lambda$, $(\lambda x M) \in \Lambda$ and $(MN) \in \Lambda$ if $x \in V$, $M, N \in \Lambda$. We use classically right association for λ's and left association for application, so that $\lambda xy.xyz$ abbreviates $(\lambda x(\lambda y((xy)z)))$. We consider the usual α- and β-conversion rules defined by $\lambda xM \underset{\alpha}{\to} \lambda yM[y/x]$, y not free in M, and $(\lambda xM)N \underset{\beta}{\to} M[N/x]$. (We shall not view the η-rule as a conversion rule but only as an extensionnality principle, see 2.1.2.).

We call *context* $C[\ ]$ an expression with a *hole* $[\ ]$ to be filled by another expression M, giving result $C[M]$. Notice that in this process a free variable M may be bound by a λ in $C[\ ]$, as in $\lambda x.[x]$. Such a capture is not possible by substitution.

We call *head normal form* any expression of the form $\lambda x_1 x_2 \ldots x_m . x M_1 M_2 \ldots M_n$, also written $\lambda \vec{x}.x\vec{M}$. Let Ω be a new symbol called the *syntactic undefined*. We define the set N of Ω-β-*normal forms* [31] by

$$\begin{cases} \Omega \in N \\ \lambda x_1 x_2 \ldots x_m . x a_1 a_2 \ldots a_n \in N \text{ if } a_1, a_2, \ldots, a_n \in N \end{cases}$$

For $a,b \in N$ we write $a < b$ if either $a = \Omega$ or $a = \lambda \vec{x}.x a_1 a_2 \ldots a_n$, $b = \lambda \vec{x}.x b_1 b_2 \ldots b_n$, $a_i < b_i$ for $1 \leq i \leq n$.

We denote by $\langle N^\infty, <, \Omega \rangle$ the completion by ideals of $\langle N, <, \Omega \rangle$, and denote the elements of N^∞ by A, A', B... [Notice that N^∞ is an ω-algebraic coherent cpo [4,26] with N as basis]. Given $M \in \Lambda$ we set $\omega(M) = \Omega$ if M is not in hnf and $\omega(\lambda \vec{x}.xM) = \lambda \vec{x}.x\overrightarrow{\omega(M}$ Clearly $\omega(M) \in N$. We set $BT(M) = \cup\{\omega(M') \mid M \xrightarrow{*} M'\} \in N^\infty$ and call $BT(M)$ the *syntactic value* of M in N^∞, see [1,14,15]. If $A \in N^\infty$ satisfies $A < BT(M)$ then A is called an *approximation* of M. If $A \in N$ then A is called a *finite approximation* of M. We write $A < M$ instead of $A < BT(M)$ when no confusion is possible.

It is also convenient to add Ω to the language Λ itself (as a new variable which cannot be bound), thus obtaining the language Λ^Ω of *partial* λ-*expressions*. The Ω-match ordering $\sqsubseteq$ on Λ^Ω is of course defined by $M \sqsubseteq N$ if $M = \Omega$, or $M = \lambda x.M_1$, $N = \lambda x.N_1$, $M_1 \sqsubseteq N_1$, or $M = M_1 M_2$, $N = N_1 N_2$, $M_1 \sqsubseteq N_1$, $M_2 \sqsubseteq N_2$.

Notice that N is included in Λ^Ω and that $a < a'$ is equivalent to $a \sqsubseteq a'$ for $a, a' \in N$. If $a \in N$ and $M \in \Lambda$, then $a < M$ holds if $a \sqsubseteq N$ for some N such that $M \xrightarrow{*} N$.

A *closed term* in Λ, Λ^{Ω} or N is a term without free variables. A closed term of N^{∞} is an $A \in N^{\infty}$ which has only closed finite approximations. The sets of closed terms will be written Λ_c, Λ_c^{Ω}, N_c and N_c^{∞}.

1.3. The languages PL(F)

These languages are typed λ-calculus with Y combinators and a set F of first-order basic function symbols, see [19,20,24] When F is assumed from the context, we simply write PL instead of PL(F).

We consider a set $K = \{\kappa_1, \kappa_2, .., \kappa_n\}$ of *ground types*, and generate a set T of *types* by $\kappa \in T$ if $\kappa \in K$ and $(\sigma \to \tau) \in T$ if $\sigma, \tau \in T$, see [20]. We write $\sigma_1 \times \sigma_2 \times \ldots \times \sigma_n \to \tau$ instead of $(\sigma_1 \to (\sigma_2 \to \ldots \to (\sigma_n \to \tau)..))$ and notice that any type σ has a unique decomposition of the form $\sigma_1 \times \sigma_2 \times \ldots \times \sigma_n \to \kappa$, $\kappa \in K$. A type of the form $\kappa_1 \times \kappa_2 \times \ldots \times \kappa_n \to \kappa$, $\kappa_i, \kappa \in K$, $i \leqslant n$, is called a *first-order types*. We also consider a set $F = \{f^{\sigma}, g^{\tau}, \ldots\}$ of *basic function symbols* where each $f^{\sigma} \in F$ has a first-order type.

Given a set $V = \bigcup_{\sigma} V^{\sigma}$ of *variables* (where each V^{σ} is denumerable) and a set $\{Y^{((\sigma\to\sigma)\to\sigma)}, \sigma \in T\}$ of *fixpoint combinators*, we generate the language $PL = PL(F) = \bigcup_{\sigma} PL(F)^{\sigma}$ by :

(i) $x^{\sigma} \in PL(F)^{\sigma}$ for $x^{\sigma} \in V$

(ii) $f^{\sigma} \in PL(F)^{\sigma}$ for $f^{\sigma} \in F$

(iii) $(\lambda x^{\sigma}.M^{\tau}) \in PL(F)^{(\sigma\to\tau)}$ for $x^{\sigma} \in V$, $M^{\tau} \in PL(F)^{\tau}$

(iv) $(M^{(\sigma\to\tau)} N^{\sigma}) \in PL(F)^{\tau}$ for $M^{(\sigma\to\tau)} \in PL(F)^{\sigma\to\tau}$, $N^{\sigma} \in PL(F)^{\sigma}$

(v) $(Y^{((\sigma\to\sigma)\to\sigma)} M^{(\sigma\to\sigma)}) \in PL(F)^{\sigma}$ for $M^{(\sigma\to\sigma)} \in PL(F)^{(\sigma\to\sigma)}$

Of course we shall omit types unless strict necessity. Notice that we allow Y to appear only in combinations. This corresponds to common programming practice, simplifies many results, but is not essential to these results (see [4]). The conversion rules here are α- and β-conversions as before,

plus Y-*conversion* defined by $(YM) \underset{Y}{\longrightarrow} M(YM)$.

The contexts are defined in the same way as in Λ, except that the holes must be typed and filled with expression of their types. We shall also need contexts with several holes $[\ ]_1^{\sigma 1}$, $[\ ]_2^{\sigma 2}$, ...

With respect to head normal forms or approximations, the symbols of F act just as variables which cannot be bound. Hence hnf's have either the form $\lambda\vec{x}.x\vec{M}$ or the form $\lambda\vec{x}.f\vec{M}$. The approximations are constructed as in Λ with typed symbols Ω^σ, $\sigma\in T$, and form sets $N(F)=\underset{\sigma}{\cup}N(F)^\sigma$ and $N^\infty(F)=\underset{\sigma}{\cup}N(F)^{\sigma\infty}$. The language PL^Ω is constructed in the same way as Λ^Ω.

The *closed terms* are the terms without free variables in V and form sets Λ_c, Λ_c^Ω, N_c and N_c^∞.

The restriction that any $f\in F$ has a first-order type has an important consequence : for any ground type $\kappa\in K$ the elements of N_c^κ and $N_c^{\kappa\infty}$ only contain symbols in F and ground Ω's (no λ's, no variables, no Y's).

In other words N_c^κ and $N_c^{\infty\kappa}$ are the free ordered and continuous algebras of type κ generated by F, so that N_c^κ is generated by

(i) $\Omega^\kappa\in N_c^\kappa$

(ii) $f^\kappa\in N_c^\kappa$ for $f^\kappa\in F$

(iii) $fa_1a_2\ldots a_n\in N_c^\kappa$ for $f^{\kappa 1\times\kappa 2\times\ldots\times\kappa n\to\kappa}\in F$ and $a_i\in N_c^{\kappa i}$, $1\le i\le n$

As an example we shall consider Plotkin's language PCF[24], suited to arithmetic computations. It has two basic types N (integers) and B (booleans), and the following list of basic function symbols :

$\underline{0},\underline{1},\underline{2},\ldots\underline{n},\ldots$: N

$\underline{tt},\underline{ff}$: B

$\underline{+1},\underline{-1}$: N→N

$\underline{\text{zero?}}$: N→B

$\underline{\text{Cond}}_N$: B×N×N→N

$\underline{\text{Cond}}_B$: B×B×B→B

where $\underline{\text{cond}}_N$ and $\underline{\text{cond}}_B$ define conditional expressions, so that $\underline{\text{cond}}_N P_1 P_2 P_3$ should be read as $\underline{\text{if}}$ P_1 $\underline{\text{then}}$ P_2 $\underline{\text{else}}$ P_3 $\underline{\text{endif}}$.

2. THE MODEL DEFINITIONS

We start by the definition of a model of Λ, which is the simplest one. Then we turn to approximation continuous models of Λ and least fixpoint models of PL(F).

2.1. Models of Λ

2.1.1. Definition : A *semantics* S of Λ is defined by a set E_ϕ and a mapping $S[\![\ ]\!] : \Lambda \to E_\phi$ which satisfies the two following conditions for all $M,N \in \Lambda$.

(Cong) $S[\![M]\!]=S[\![N]\!] \Rightarrow \forall C[\ ].S[\![C[M]]\!]=S[\![C[N]]\!]$

(Conv) $M \xrightarrow{*} N \Rightarrow S[\![M]\!]=S[\![N]\!]$

A semantic is *syntactically extensional* or is a η-*semantics* if it satisfies one of the three following equivalent conditions (equivalence proof left to the reader) :

(se_1) $\forall M,N,\ (\forall P\ S[\![MP]\!]=S[\![NP]\!]) \Rightarrow S[\![M]\!]=S[\![N]\!]$

(se_2) $\forall M,N$, x not free in MN,

$$S[\![Mx]\!]=S[\![Nx]\!] \Rightarrow S[\![M]\!]=S[\![N]\!]$$

(η) $\forall M$, x not free in M, $S[\![\lambda x.Mx]\!]=S[\![M]\!]$

A *premodel* $M=\langle E_M, D_M, \mathrm{eval}, \bullet \rangle$ of Λ is defined by :

- a set E_M called the *semantic domain* ;
- a set $\mathcal{D}_M$ called the *value domain* ;
- a set Env_M called the *environment domain*, which is the cartesian product of denumerably many copies of $\mathcal{D}$ indexed by the variables of V. We write $\rho,\rho',\ldots \in \mathrm{Env}$. For $\rho \in \mathrm{Env}$ we call $\rho(x) \in \mathcal{D}$ the x-th component of ρ and for $d \in \mathcal{D}$ we call $\rho[x \leftarrow d]$ the environment such that $\rho[x \leftarrow d](x)=d$ and $\rho[x \leftarrow d](y)=\rho(y)$ for $y \neq x$;
- An *evaluation mapping* $\mathrm{eval} : E_M \times \mathrm{Env}_M \to \mathcal{D}_M$, which will allow to compute the value of an expression in an environment. For $\phi \in E$, $\rho \in \mathrm{Env}$ we write $\phi\rho = \mathrm{eval}(\phi,\rho)$;
- An *application mapping* $. : \mathcal{D} \times \mathcal{D} \to \mathcal{D}$

A premodel is *environment extensional* if $\phi\rho=\phi'\rho$ for all ρ implies $\phi=\phi'$ in E, *value extensional* if d.e=d'.e for all e implies d=d' in $\mathcal{D}$, *extensional* if it is both environment and value extensional.

A *model* M of Λ is a premodel M together with a semantic mapping $M[\![\ ]\!]:\Lambda\to E$ which satisfies (Cong) and (Conv) above -i.e. is a semantics- and satisfies the following conditions for any M,N,x,ρ,ρ',d :

(Var) $\quad M[\![x]\!]\rho=\rho(x)$

(app) $\quad M[\![MN]\!]\rho=(M[\![M]\!]\rho).(M[\![N]\!]\rho)$

(lambda) $\quad (M[\![\lambda xM]\!]\rho).d=M[\![M]\!]\rho[x\leftarrow d]$

(Free) $\quad M[\![M]\!]\rho=M[\![M]\!]\rho'$ if $\rho(x)=\rho'(x)$ for all x free in M.

We write $M=_M N$ for $M[\![M]\!]=M[\![N]\!]$ and call $=_M$ *the M-equality*.

It is straightforward to see that the semantics is syntactically extensional when the model M is extensional. We shall see that the converse is not necessarily true.

○

Notice that our model definition differs from those of [1 , 9]. The first difference lies in the explicit introduction of the semantic domain E, which is implicitely always assumed to be the set of functions from Env to $\mathcal{D}$ in [1, 9]. We shall see that in many case this implicit choice of E is not satisfactory. This will be quite clear on approximation continuous models where some extra structure is needed on E, which call not always be defined on the set of (continuous) functions from Env to E. The second difference lies in the requirement (ξ) $M=_M N\Rightarrow\lambda xM=_M\lambda xN$ which follows from (Cong). Hindley-Longo [9] require a stronger condition which will not necessarily be satisfied by the categorical models of section 6, and is indeed not satisfied by the algorithm model of [6]. The introduction of E gives us useful flexibility without loss of generality. The precise relations between the various model definitions are studied in [7].

Let us turn to continuous semantics and models :

2.1.2. Definition : A *continuous semantics* S is defined by a cpo $\langle E_S, \sqsubseteq, \bot\rangle$ and a mapping $S[\![\ ]\!]: \Lambda \to E_S$ which satisfies the following conditions for all $M,N,\mathcal{C}[\ ]$:

(conv) $M \xrightarrow{*} N \Rightarrow S[\![M]\!]=S[\![N]\!]$

(mon) $S[\![M]\!] \sqsubseteq S[\![N]\!] \Rightarrow S[\![\mathcal{C}[M]]\!] \sqsubseteq S[\![\mathcal{C}[N]]\!]$

(lim) For all $X \subset \Lambda$, if the set $S[\![X]\!]=\{S[\![M]\!], M\in X\}$ is directed and has lub $S[\![N]\!]$ then the directed set $S[\![\mathcal{C}[X]]\!]=\{S[\![\mathcal{C}[M]]\!], M\in X\}$ is directed and has lub $S[\![\mathcal{C}[M]]\!]$.

We write $M \sqsubseteq_S N$ for $S[\![M]\!] \sqsubseteq S[\![N]\!]$. Notice that (mon) implies (cong), so that S is a semantics in the previous sense. The equations (mon) and (conv) express the monotonicity and continuity of the formation laws of Λ. For example (mon) can obviously be replaced by two conditions.

$\forall M,M',N,N'$, $S[\![M]\!] \sqsubseteq S[\![M']\!]$ and $S[\![N]\!] \sqsubseteq S[\![N']\!] \Rightarrow S[\![MN]\!] \sqsubseteq_S S[\![M'N']\!]$

$\forall M,N,x$, $S[\![M]\!] \sqsubseteq S[\![M']\!] \Rightarrow S[\![\lambda x M]\!] \sqsubseteq S[\![\lambda x M']\!]$.

Notice that a continuous semantics S is syntactically extensional iff it is *syntactically order-extensional*, i.e satisfies one of the three equivalent conditions

(soe1) $\forall M,N$, $(\forall P\ S[\![MP]\!] \sqsubseteq S[\![NP]\!]) \Rightarrow S[\![M]\!] \sqsubseteq S[\![N]\!]$

(soe2) $\forall M,N$, x not free in MN, $S[\![Mx]\!] \sqsubseteq S[\![Nx]\!] \Rightarrow S[\![M]\!] \sqsubseteq S[\![N]\!]$

(η) $\forall M$, x not free in M, $S[\![\lambda x Mx]\!]=S[\![M]\!]$

Hence the introduction of the ordering makes no difference for syntactic extensionnality.

A continuous semantics S is *approximation continuous* if it satisfies in addition the two conditions

(omega) $S[\![M]\!]=\bot$ if $BT(M)=\Omega$

(approx) $S[\![M]\!]=\cup\{S[\![a]\!], a<M\}$ for all $M\in\Lambda$.

Hence the value of M must be the lub of the values of its finite approximations. A continuous model satisfying only (omega) will be called *sensible*, following [1,12]. In any sensible model the semantics $S[\![\ ]\!]:\Lambda\to E$ extends to $S[\![\ ]\!]:\Lambda^{\Omega}\to E$ by setting $S[\![\Omega]\!]=\bot$. Also $S[\![M]\!]=\bot$ implies $S[\![\lambda xM]\!]=\bot$ and $S[\![MN]\!]=\bot$ for all x and N (use (cong) and $BT(\lambda xM)=BT(MN)=\Omega$ when $BT(M)=\Omega$). In any approximation continuous model $S[\![\ ]\!]$ extends furthermore to N^{∞} by setting $S[\![A]\!]=\cup\{S[\![a]\!]\mid a<A\}$, and is continuous. Then (approx) rewrites into $S[\![M]\!]=S[\![BT(M)]\!]$.

○

Having finished with semantics we turn to models.

2.1.3. Definition : A *continuous premodel* is defined as a premodel $M=(\langle E_M,\sqsubseteq,\bot\rangle,\langle \mathcal{D}_M,\leq,\bot\rangle,\langle Env_M,\leq,\bot\rangle,\ eval,\bullet)$ where E_M and $\mathcal{D}_M$ are cpo's, where Env_M is ordered component-wise and where eval and $\bullet$ are continuous and left strict (i.e. such that $eval(\bot,\rho)=\bot$ and $\bot\bullet d=\bot$ for all ρ and d).

A premodel M is *environment-order extensional* if $\phi\rho\leq\phi'\rho$ for all ρ implies $\phi\sqsubseteq\phi'$ in E_M, *value order extensional* if $d\bullet e\leq d'\bullet e$ implies $d\leq d'$ in $\mathcal{D}_M$, *order extensional* if both conditions are satisfied.

A *continuous model* M is a continuous premodel together with a continuous semantics $M[\![\ ]\!]:\Lambda\to E_M$ which satisfies the model equations (var), (app), (lambda) and (free). It is *sensible* or *approximation continuous* if the semantics $M[\![\ ]\!]$ is.

○

We grouped the different conditions to satisfy in appendix A. Classical examples of approximation continuous models are the models $\mathcal{D}^{\infty}$ of [28], which is order-extensional, and the models P^{ω}[29] and T^{ω}[25] which are not extensional. It is shown in [2] that $BT(M)<BT(N)$ is equivalent to $M\sqsubseteq_{T^{\omega}}N$.

2.2. Models of PL(F)

The basic definitions are similar, with types added of course. Here we also need to define the semantics of the combinations (YM) and of the basic function symbols of F. For (YM) we shall use least fixpoints. For symbols of F we shall follow Milner [20] and use first-order interpretations. It would be also interesting to introduce first-order rewriting rules as in [11]. However this would raise important syntactic problems which are certainly outside the scope of this paper. In the particular case of PCF, an equivalence proof of the two techniques is given in [24]. We start with interpretations.

2.2.1. Definition : Let K a set of ground types and F a set of basic function symbols typed on K. An *interpretation* A of (K,F) is defined by :

- For each $\kappa \in K$, a cpo $\mathcal{D}_A^\kappa$
- For each $f^\sigma \in F$ with $\sigma = \kappa_1 \times \kappa_2 \times \ldots \times \kappa_n \to \kappa$ a continuous function $A(f) : \mathcal{D}_A^{\kappa_1} \times \mathcal{D}_A^{\kappa_2} \times \ldots \times \mathcal{D}_A^{\kappa_n} \to \mathcal{D}_A^{\kappa}$

As a typical example, the standard interpretation of PCF is defined as follows :

$A(\underline{n})=n$, $A(\underline{tt})=tt$, $A(\underline{ff})=ff$, $A(\underline{+1})(\perp)=A(\underline{-1})(\perp)=\perp$

$A(\underline{+1})(n)=n+1$, $A(\underline{-1})(0)=\perp$, $A(\underline{-1})(n+1)=n$, $A(\underline{zero?})(\perp)=\perp$,

$A(\underline{cond})(tt,d,d')=d$, $A(\underline{cond})(ff,d,d')=d'$ where $\underline{cond}$ stands for $\underline{cond}_N$ or $\underline{cond}_B$.

An important interpretation is the *symbolic interpretation* E defined by $\mathcal{D}_E^\kappa = N_c^{\kappa\infty}$ (see 1.3.) and $E(f)(A_1 A_2 \ldots A_n) = f A_1 A_2 \ldots A_n$. Then E is just made of expressions, and is of course initial : let us call *morphisms of interpretations* $h : A \to A'$ any collection $\{h^\kappa : \mathcal{D}_A^\kappa \to \mathcal{D}_{A'}^\kappa\}$ of morphisms of cpo's such that

$$h(A(f)(d_1,d_2,..,d_n)) = A'(f)(h(d_1),h(d_2),..,h(d_n))$$

holds for all $f,d_1,d_2,..,d_n$. Then there exists one and only one morphism $h:E\to A$, which we shall denote by $A[\![\ ]\!]$. Hence $A[\![A]\!]$ is the value of A in A.

○

2.2.2. Definition : A *continuous semantics* S of PL is defined by a collection of cpo's $<E_S^\sigma,\sqsubseteq^\sigma,\perp^\sigma>,\sigma\in T$ and a mapping $S[\![\ ]\!]:PL\to E_S=\bigcup_\sigma E_S^\sigma$ which respects types and satisfies the typed versions of (conv), (mon) and (cont). It is *sensible* if it satisfies (omega) at all types, and *approximation continuous* if it satisfies (approx). It is *syntactically (order-) extensional* or is a *η-semantics* iff it satisfies one of the five equivalent conditions (se_1), (se_2), (soe_1), (soe_2), (η). The extension of $S[\![\ ]\!]$ to PL^Ω and N^∞ is done as in Λ.

A *continuous premodel* M is defined by a collection of semantic domains $<E_M^\sigma,\sqsubseteq_M^\sigma,\perp^\sigma>,\sigma\in T$, a collection of value domains $<\mathcal{D}_M^\sigma,\leq_M^\sigma,\perp^\sigma>,\sigma\in T$, a collection of continuous mappings $eval^\sigma$: $E_M^\sigma\times Env_M\to\mathcal{D}_M^\sigma$ and a collection of continuous left-strict application mappings $\bullet^{\sigma,\tau}:\mathcal{D}_M^{\sigma\to\tau}\times\mathcal{D}_M^\sigma\to\mathcal{D}_M^\tau$, $\sigma,\tau\in T$. Extensionnality is defined as in Λ.

Let A be an interpretation of F. A *least fixpoint model* M is a continuous premodel M together with a sensible continuous semantics $M[\![\ ]\!]:PL\to E$ which satisfies (var), (app), (lambda), (free) and the following conditions for all $f,d_1,d_2,..,d_n,\rho,M$:

(int) $\quad M[\![f]\!]\rho\bullet d_1\bullet d_2\bullet\ldots\bullet d_n = A(f)(d_1,d_2,..,d_n)$

(fix) $\quad M[\![YM]\!]\rho = Y\bullet M[\![M]\!]\rho$

where in (fix) we understand $Y\bullet d$ as $\bigcup_n d\bullet(d\ldots\bullet(d\bullet\perp)\ldots)$, $\underbrace{\qquad\qquad}_{n}$

i.e as the least fixpoint of the function $e\to d\bullet e$ from $\mathcal{D}_M^\sigma$ to $\mathcal{D}_M^\sigma$ if M has type $\sigma\to\sigma$.

The first condition expresses that M respects A, the second condition says that Y is interpreted as a least fixpoint operator. ○

Notice that we did not require (approx) in the least fixpoint model definition. This is because (approx) is indeed a consequence of (fix).

2.2.3. Theorem : If M is a least fixpoint model, then $M[\![\ ]\!]$ is approximation continuous.

Sketch of proof (complete proof in [4] : We have to show $M[\![M]\!]=\cup\{M[\![a]\!],a<M\}$. Assume $a<M$. Then there exists P such that $a\subset P$ and $M\overset{*}{\to}P$. By (cong) and (conv) we get $M[\![a]\!]\sqsubseteq M[\![P]\!]=M[\![N]\!]$. Therefore $\cup\{M[\![a]\!],a<M\}\sqsubseteq M[\![M]\!]$. The other direction is harder, and inspired from Wadsworth [31,32] and Levy [14,15]. Define the approximate semantics M_n by setting $M_n[\![M]\!]=M[\![C[N^n\Omega]]\!]$ if $C[\ ]$ is the context of the subexpressions YN in M. Using (lim), show $M[\![M]\!]=\cup M_n[\![M]\!]$. Then show that for any n there exists $a<M$ such that $M_n[\![M]\!]\sqsubseteq M[\![a]\!]$. For this purpose, introduce a labeled calculus in which any Y receives an integer as label and in which the Y-rule is changed into $Y^pM\to M(Y^{p-1}M)$. Show that this calculus has the Church-Rosser and strong normalization properties. Then given $n\geq 0$, let M^n be M where all Y's are labeled with n. Let P be the normal form of M^n in the labeled calculus, let $a=\omega(P)$. Check $M_n[\![M]\!]\sqsubseteq M[\![a]\!]$.

□

3. CONSTRUCTION OF INITIAL MODELS

3.1. Morphisms of models

The notions of morphisms are quite straightforward from the model definition.

3.1.1. Definition : Let S, S' be two semantics of Λ. Then a *morphism* $\theta:S\to S'$ is a mapping $\theta:E_S\to E_S$, such that $\theta(S[\![M]\!])=\theta'[\![M]\!]$ for all $M\in\Lambda$. Let M, M' be two models. Then a morphism $\theta:M\to M'$ is defined by two mappings $\theta_E:E_M\to E_{M'}$ and $\theta_D:D_M\to D_{M'}$ such that θ_E is a morphism of semantics and such that

$$\forall x,y\in D_M,\quad \theta_D(x.y)=\theta_D(x).\theta_D(y)$$

$$\forall \phi\in E_M,\rho\in Env_M,\quad \theta_D(\phi\rho)=\theta_E(\phi)\theta_D(\rho)$$

where $\theta_D(\rho)$ is of course defined by $\theta_D(\rho)(x)=\theta_D(\rho(x))$ for all $x\in V$.

Let S,S' be two (approximation) continuous semantics. Then $\theta:S\to S'$ is a *morphism of (approximation) continuous semantics* if it is a morphism of cpos together with a morphism of semantics. Similarly $\theta:M\to M'$ is a morphism of (approximation) continuous models if θ_D and θ_E are morphisms of cpo's and if θ is a morphism of models.

Of course a semantics (or model) S of a given class is *initial* if for any semantics S' of the same class there exists a unique morphism $\theta:S\to S'$.

The definitions for PL(F) are similar, considering here collections of mappings θ_E^σ and θ_D^σ, $\sigma\in T$. Notice that if M is a model of A and if M' is a model of A', then any morphism $\theta:M\to M'$ defines a morphism of interpretations $\theta:A\to A'$ given by the θ^κ, $\kappa\in K$.

○

3.2. Initial models of Λ

The construction of the initial (η)-model is quite straightforward.

3.2.1. Definition : Call *substitution* in Λ any mapping $\sigma:V\to\Lambda$ and for $M\in\Lambda$ having free variables $x_1,x_2,..x_n$, write $M\sigma=M[\sigma(x_1)/x_1,\ldots,\sigma(x_n)/x_n]$.

Define the model $\mathcal{I}$ in the following way :

- $E_{\mathcal{I}}$ is the set of β-interconvertibility classes in Λ.
  The class of M is written $[M]$.
- $\mathcal{D}_{\mathcal{I}}$ is the set of β-interconvertibility classes of closed term in Λ_c.
- An element of $Env_{\mathcal{I}}$ is then a substitution $\rho:V\to\Lambda_c$
- For $[M]\in E_{\mathcal{I}}$ and $\rho\in Env_{\mathcal{I}}$, $[M]\rho=[M\rho]$
- For $[M_1],[M_2]\in\mathcal{D}_{\mathcal{I}}$, $[M_1]\cdot[M_2]=[M_1M_2]$

The model $\mathcal{I}_\eta$ is defined in exactly the same way, replacing β-interconvertibility by $\beta\eta$-interconvertibility.

○

3.2.2. Proposition : The model (semantics) $\mathcal{I}$ is the initial model (semantics). The model (semantics) $\mathcal{I}_\eta$ is the initial η-model (semantics).

Proof : That $\mathcal{I}$ and $\mathcal{I}_\eta$ are models is immediate. Given another model M, define $\theta_E([M])=M[\![M]\!]$ for $M\in\Lambda$ and $\theta_D([M])=M[\![M]\!]\rho$ for $M\in\Lambda_c$ where ρ is any environment. Then θ is the unique morphism from $\mathcal{I}$ to M. If M is a η-model, define $\theta_\eta:\mathcal{I}_\eta\to M$ in the same way but considering η-interconvertibility classes. Then θ_η is the unique morphism from $\mathcal{I}_\eta$ to M.

□

3.2.3. Remark : Although I_η is the initial η-model, it is not extensional - more precisely it is neither environment - extensional nor value extensional. This is shown by Plotkin counterexample to the ω-rule, see [1,22].

○

For constructing the initial approximation continuous model we need the basic *syntactic continuity theorem* of Wadsworth [32] and Levy [14].

3.2.4. Theorem : Assume $BT(M)<BT(N)$. Then for any context $C[\ ]$ one has $BT(C[M])<BT(C[N])$. □

3.2.5. Definition : A *substitution* σ in N^∞ is a mapping $\sigma:V\to N^\infty$. Given $\sigma:V\to N$ and $a\in N$ with free variables $x_1, x_2,..x_n$, we define $a\sigma=BT(a[\sigma(x_1)/x_1,..,\sigma(x_n)/x_n])$. Given $\sigma:V\to N$ and $A\in N^\infty$, we define $A\sigma=\cup\{a\sigma \mid a<A\}$. Eventually given $\sigma:V\to N^\infty$ and $A\in N^\infty$, we set

$$A\sigma=\cup\{A\tau \mid \tau:V\to N,\ \tau<\sigma\}$$

(all these definitions make sense because of 3.2.4.).

The model $\mathcal{B}$ is defined by $E_{\mathcal{B}}=N^\infty$, $\mathcal{D}_{\mathcal{B}}=N^\infty_c$, $A\rho$ as before for $\rho\in Env_{\mathcal{B}}$ (which is just a substitution in N^∞), and $A.B=\cup\{BT(ab) \mid a<A, b<B\}$ for $A,B,\in N^\infty_c$ (again this definition makes sense by 3.2.4. : consider the context $[\ ]_1\ [\ ]_2$).

○

3.2.6. Proposition : The model (semantics) $\mathcal{B}$ is the initial approximation continuous model (semantics).

Proof : That $\mathcal{B}$ is a model follows from 3.2.4.. Its initiality is obvious.

□

We shall not construct initial approximation continuous η-models, although this would be certainly possible using the relation $\sim_\eta$ of [12] between Böhm trees. We think this construction is too complicated and not enough informative in Λ. However it will be quite simple and interesting in PL, as we shall see the next section.

3.3. Initial models of PL

The definitions of morphisms of models and the construction of the initial models I, I_η and B are of course the same as in Λ, and we shall concentrate on the construction and properties of the initial approximation continuous η-model. Here we take advantage of a convenient property of the typed λ-calculus : the η-*expansion* $M \to \lambda x Mx$ always terminates, so that we can construct the model by terms in maximal η-expansion.

3.3.1. Definition : The set N_η^σ of η-*expanded approximations* is the least subset of N^σ such that

(i) $\Omega^\sigma \in N_\eta^\sigma$

(ii) $\lambda x_1^{\sigma_1} x_2^{\sigma_2} \ldots x_m^{\sigma_m} . s^\tau a_1^{\theta_1} a_2^{\theta_2} \ldots a_n^{\theta_n} \in N_\eta^\sigma$

if $\sigma = \sigma_1 \times \sigma_2 \times \ldots \times \sigma_m \to \kappa$, $\kappa \in K$, $s \in V \cup F$

$\tau = \theta_1 \times \theta_2 \times \ldots \times \theta_n \to \kappa$, $a_i^{\theta_i} \in N_\eta^{\theta_i}$ for $1 \le i \le n$.

Define $N_\eta^{\sigma\infty}$ as the ideal completion of N_η^σ. Notice that $A \in N_\eta^\infty$ and $B < A$ in N^∞ imply $B \in N_\eta^\infty$.

For $A, B \in N^\infty$, write $\lambda x.A = \cup\{\lambda x.a \mid a < A\}$, $A.B = \cup\{BT(ab) \mid a < A, b < B\}$ and finally $A[B/x] = \cup\{BT(a[b/x]) \mid a < A, b < B\}$. Again these notations make sense by the continuity theorem 3.2.4. which also holds in PL, see [4].

For $x^\sigma \in V$, define the η-*expansion* $\eta(x^\sigma)$ by induction on the size of σ :

(i) $\eta(x^{\kappa})=x^{\kappa}$

(ii) $\eta(x^{\sigma})=\lambda x^{\sigma_1}x^{\sigma_2}\ldots x^{\sigma_n}.x^{\sigma}\eta(x^{\sigma_1})\eta(x^{\sigma_2})\ldots\eta(x^{\sigma_n})$

if $\sigma=\sigma_1\times\sigma_2\times\ldots\times\sigma_n\to\kappa$.

Then $\eta(x^{\sigma})\in N_{\eta}^{\sigma}$. For $f\in F$ of type $\kappa_1\times\kappa_2\times\ldots\times\kappa_n\to\kappa$, define similarly $\eta(f)=\lambda x_1^{\kappa_1}x_2^{\kappa_2}\ldots x_n^{\kappa_n}.fx_1x_2\ldots x_n$, so that $\eta(f)\in N_{\eta}$. Let $X_{\eta}^{n\sigma}$ be inductively defined by

(i) $X_{\eta}^{0\sigma}=\Omega^{\sigma}$

(ii) $X_{\eta}^{n+1\sigma}=\lambda x_1^{\sigma_1}x_2^{\sigma_2}\ldots x_m^{\sigma_m}.x^{\sigma\to\sigma}X_{\eta}^{n\sigma}\eta(x_1)\eta(x_2)\ldots\eta(x_m)$

if $\sigma=\sigma_1\times\sigma_2\times\ldots\times\sigma_m\to\kappa$.

Intuitively $X_{\eta}^{n\sigma}$ is the η-expansion of $x^n\Omega$ and will be used for interpreting combinations YM.

○

3.3.2. Lemma : If $A,B\in N_{\eta}^{\infty}$, then $\lambda xA\in N_{\eta}^{\infty}$, $A.B\in N_{\eta}^{\infty}$ and $A[B/x]\in N_{\eta}^{\infty}$.

Proof : The result is obvious for λxA. Assume $A^{\sigma\to\tau}\in N_{\eta}^{\infty}$ and $B^{\sigma}\in N_{\eta}^{\infty}$. We show $A.B\in N_{\eta}^{\infty}$ by induction on the size of σ. Let $\sigma=\sigma_1\times\sigma_2\times\ldots\times\sigma_m\to\kappa_1$ and $\tau=\tau_1\times\tau_2\times\ldots\times\tau_n\to\kappa_2$. We have to show $BT(ab)\in N_{\eta}^{\infty}$ for all $a<A,b<B$. We show this by induction on the size of a

case 1 : $a=\Omega$. Obvious since $BT(ab)=\Omega$

case 2 : $a=\lambda x^{\sigma}y_1^{\tau_1}y_2^{\tau_2}\ldots y_n^{\tau_n}.sa_1a_2\ldots a_p$ with $s\neq x^{\sigma}$.
Then $BT(ab)=\lambda\vec{y}.s.\vec{A}$ where $A_i=BT(a_i[b/x])$. Since $A_i=BT((\lambda x^{\sigma}a_i)b)$ where the size of λxa_i is smaller than the size of a, one has $A_i\in N_{\eta}^{\infty}$ by induction on a and the result follows.

case 3 : $a=\lambda x^{\sigma}y_1^{\tau_1}y_2^{\tau_2}\ldots y_n^{\tau_n}.x^{\sigma}a_1^{\sigma_1}a_2^{\sigma_2}\ldots a_m^{\sigma_m}$, with $\kappa_1=\kappa_2$.

Then $ab \to \lambda\vec{y}.ba_1[b/x]a_2[b/x]\ldots a_m[b/x]$. Let $A_i = BT(a_i[b/x])$. Then $A_i \in N_\eta^\infty$ as in case 2. Now one has $BT(ab) = \lambda\vec{y}.(\ldots((b \bullet A_1) \bullet A_2)\ldots) \bullet A_m$. Since the size of every σ_i is less than the size of σ, one has $b \bullet A_1 \in N_\eta^\infty$, $(b \bullet A_1) \bullet A_2 \in N_\eta^\infty, \ldots, BT(ab) \in N_\eta^\infty$.

Of course $A[B/x] \in N_\eta^\infty$ when $A, B \in N_\eta^\infty$ since $A[B/x] = (\lambda x A) \bullet B$.

□

3.3.3. Definition : The model $\mathcal{B}_\eta$ is defined in the following way

- $E = N_\eta^\infty$
- $D = N_{\eta c}^\infty$
- Env is again the set of substitutions in $N_{\eta c}^\infty$, and $A\rho$ is defined as the result of the substitution ρ in A.
- $\bullet : D \times D \to D$ is defined in 3.3.1.
- The semantic function $\mathcal{B}_\eta[\ ]$ is defined by

(i) $\mathcal{B}\eta[\![x^\sigma]\!] = \eta(x^\sigma)$

(ii) $\mathcal{B}\eta[\![f^\sigma]\!] = \eta(f^\sigma)$

(iii) $\mathcal{B}\eta[\![MN]\!] = \mathcal{B}_\eta[\![M]\!] \bullet \mathcal{B}_\eta[\![N]\!]$

(iv) $\mathcal{B}_\eta[\![\lambda x M]\!] = \lambda x \bullet \mathcal{B}_\eta[\![M]\!]$

(v) $\mathcal{B}_\eta[\![YM]\!] = \cup \mathcal{B}_\eta[\![X_\eta^n[M/x]]\!]$

All these definitions make sense by 3.3.2.

○

3.3.4. Proposition : The model $\mathcal{B}_\eta$ is the initial approximation continuous η-model.

Proof : no difficulty. □

In general the model $\mathcal{B}_\eta$ is not extensional. For example assume $F = \emptyset$. Then the only closed term of type κ is Ω^κ, and the two terms $\lambda x^\kappa.\Omega^\kappa$ and $\lambda x^\kappa.x^\kappa$ in $N_\eta^{\kappa \to \kappa}$ are such that $(\lambda x^\kappa.\Omega^\kappa).\Omega^\kappa = (\lambda x^\kappa.x^\kappa).\Omega^\kappa$ but are different. We now see that $\mathcal{B}_\eta$

is indeed order-extensional when F is rich enough.

3.3.5. Definition : We say that $t \in N_\eta^\kappa$ is written only with $Fu\{x_1,..,x_n\}$ if the symbols in t are all included in this set (hence no λ's allowed). We say that F *separates* N_η if the two following conditions are satisfied :

(i) For any ground type κ there exist two terms t_1^κ, t_2^κ written only with $Fu\{x^\kappa\}$ and of the form $t_1 = f_1 \vec{t}$, $t_2 = f_2 \vec{t}$, $f_1 \neq f_2$.

(ii) For any list $L=\{x_1^{\kappa_1}, x_2^{\kappa_2}, .. x_n^{\kappa_n}\}$ of ground variables and any $\kappa \in K$ there exists a term $u_L \in N_\eta$ written only with $Fu L$ and containing at least an occurence of each $x_i^{\kappa_i}$ ○

For example it is immediately seen that the basic arithmetic functions of PCF separate N_η. For showing that B_η is order extensional we shall introduce two separators s_1^σ and s_2^σ at each type σ with the property that $a \neq a'$ holds if $a.s_1 \neq a'.s_1$ or $a.s_2 \neq a'.s_2$ holds at type τ for any closed a,a' of type $(\sigma \to \tau)$. This obviously shows the result by induction on types. For instance take $\sigma = \kappa\times\kappa\to\kappa$. Then $s_1^\sigma = \lambda xy.h_1xy$ and $s_2^\sigma = \lambda xy.h_2xy$, $h_1 \neq h_2$, are adequate separators. For example if $a=\lambda x.x(xfg)f$ and $a'=\lambda x.x(xff)f$ then $a.s_1 = h_1(h_1fg)f \neq h_2(h_2fg)f = a'.s_1$ and similarly $a.s_2 \neq a'.s_2$. Two separators are needed because s_1 alone would be unable to separate the two terms $a=\lambda x.x(xfg)f$ and $a'=\lambda x.h_1(h_1fg)f$. Notice that the separation problem here is solved in the free λ-calculus by Böhm's combinator [8] :

$$s=(\lambda x_1x_2x_3.x_3x_1x_2)(\lambda xy.x)(\lambda xy.y).$$

But this combinator cannot be adequately typed, so we must act with basic function symbols.

3.3.6. Notation : Assume that F separates N_η. Call E_n the set of sequences $\varepsilon=(i_1,i_2,\ldots i_n)$, $i_j=1,2$ for $1\leq j\leq n$. Write $\vec{u}_\varepsilon=u_{i_1}u_{i_2}\ldots u_{i_n}$. For $\sigma=\sigma_1\times\sigma_2\times\ldots\times\sigma_n\to\kappa$ write $\sigma_i=\sigma_i^1\times\sigma_i^2\times\ldots\times\sigma_i^{p_i}\to\kappa_i$. Call u_σ^κ a term of type κ written only with $F\cup X$, where X is the set of variables $x_{i,\varepsilon}^{\kappa_i}$ with $\varepsilon\in E_{p_i}$, and containing all these variables. Write $\sigma v_1=t_1^\kappa[u_\sigma^\kappa/x^\kappa]$ and $\sigma v_2=t_2^\kappa[u_\sigma^\kappa/x^\kappa]$, where t_1^κ and t_2^κ are as in 3.3.5.. Define finally the separators s_1^σ and s_2^σ in $N_{\eta c}^\sigma$ by induction on σ in the following way :

$$s_1^\sigma=\lambda x_1^{\sigma_1}x_2^{\sigma_2}\ldots x_n^{\sigma_n}.\sigma v_1[x_i^{\sigma_i}\vec{s}_2/x_{i,\varepsilon}^{\kappa_i},\varepsilon\in E_{p_i}]$$

$$s_2^\sigma=\lambda x_1^{\sigma_1}x_2^{\sigma_2}\ldots x_n^{\sigma_n}.\sigma v_2[x_i^{\sigma_i}\vec{s}_2/x_{i,\varepsilon}^{\kappa_i},\varepsilon\in E_{p_i}]$$

(The induction starts at n=0 by $s_1=\sigma v_1=t_1^\kappa[u_\kappa^\kappa/x^\kappa]$ and $s_2=\sigma v_2=t_2^\kappa[u_\kappa^\kappa/x^\kappa]$).

○

3.3.7. Lemma : Assume that F separates N_η. Then for all $\sigma=\sigma_1\times\sigma_2\times\ldots\times\sigma_n\to\kappa$ and all $a,b\in N_{\eta c}^\sigma$ the following conditions are equivalent :

1. $a<b$
2. $a.s_i^{\sigma_1}<b.s_i^{\sigma_2}$ for $i=1,2$
3. $a.\vec{s}_\varepsilon<b.\vec{s}_\varepsilon$ for all $\varepsilon\in E_n$.

Proof : The only non-trivial implication is 3=>1. Assume 3 satisfied and assume 1<=>2<=>3 for all term of size less than the size of a.

case 1 : $a=\Omega$. Trivial

case 2 : $a=\lambda x_1^{\sigma_1}x_2^{\sigma_2}\ldots x_n^{\sigma_n}.x_k^{\sigma_k}a_1a_2\ldots a_{p_k}$

Clearly $x_k^{\sigma_k}$ is bound since a is closed.

Let $\sigma_k=\sigma_k^1\times\sigma_k^2\times\ldots\times\sigma_k^{p_k}\to\kappa_k$, let $\sigma_k^j=\tau_1^j\times\tau_2^j\times\ldots\times\tau_{q_j}^j\to\kappa'_j$.

Let $a'_j=\lambda x_1x_2\ldots x_n.a_i$ for $1\leq j\leq p_k$. Then a'_j has type

$$\sigma_1\times\sigma_2\times\ldots\times\sigma_n\times\tau_i^j\times\tau_2^j\times\ldots\times\tau_{q_j}^j\to\kappa'_j.$$

Let $\theta=\sigma_k$. We start with the case where b has the same form as a, then show that the other cases are impossible.

<u>case 2.1</u>. : $b=\lambda x_1x_2\ldots x_n.x_k^{\sigma_k}b_1b_2\ldots b_{p_k}$.

We need to show $a'_j<b'_j=\lambda\vec{x}.b_j$ for $1\leq j\leq p_k$, or equivalently $a'_j\bullet\vec{s}_\varepsilon<b'_j\bullet\vec{s}_\varepsilon$ for all $\varepsilon\in E_{n+q_j}$ by induction. Any $\varepsilon\in E_{n+q_j}$ decomposes into $\varepsilon=\varepsilon_1\varepsilon_2$ where $\varepsilon_1\in E_n$ and $\varepsilon_2\in E_{q_j}$. But we know that $a\bullet\vec{s}_{\varepsilon_1}<b\bullet\vec{s}_{\varepsilon_1}$ holds by hypothesis. By construction, $a\bullet\vec{s}_{\varepsilon_1}$ has the following form :

$$a\bullet\vec{s}_{\varepsilon_1}=\theta v_{\varepsilon_1(k)}[a'_j\bullet\vec{s}_{\varepsilon_1}\bullet\vec{s}_{\varepsilon_2}/x_{j,\varepsilon_2}^{\kappa'_j},\ 1\leq j<p_k,\ \varepsilon_2\in E_{q_j}]$$

By construction of the term $\theta v_{\varepsilon_1(k)}$, the term $a'_j\bullet\vec{s}_{\varepsilon_1}\bullet\vec{s}_{\varepsilon_2}$ has at least one occurrence in $a\bullet\vec{s}_{\varepsilon_1}$. Similarly $b'_j\bullet\vec{s}_{\varepsilon_1}\bullet\vec{s}_{\varepsilon_2}$ has the same occurrence in $b\bullet\vec{s}_{\varepsilon_1}$, so that $a'_j\bullet\vec{s}_{\varepsilon_1}\bullet\vec{s}_{\varepsilon_2}<b'_j\bullet\vec{s}_{\varepsilon_1}\bullet\vec{s}_{\varepsilon_2}$, which is the required result.

<u>case 2.2</u>. : $b=\Omega$. Clearly impossible since $a\bullet\vec{s}_\varepsilon\neq\Omega$.

<u>case 2.3</u>. : $b=\lambda\vec{x}.f\vec{b}$. Impossible since the term $b\bullet\vec{s}_\varepsilon$ always begins by f while the term $a\bullet\vec{s}_\varepsilon$ may begin by two different functions symbols according to the choice of ε.

<u>case 2.4</u>. : $b=\lambda\vec{x}.x_\ell^{\sigma_\ell}\vec{b}$, $\ell\neq k$

If we take ε such that $\varepsilon(k)=1$ and $\varepsilon(\ell)=2$, then $a.\vec{s}_\varepsilon$ and $b.\vec{s}_\varepsilon$ begin with two different function symbols, which is impossible.

case 3 : $a=\lambda\vec{x}.f\vec{a}$. As in case 2 one first shows the result when b has the same form and one then shows that the other cases are impossible. □

3.3.8. Lemma : For all $a\in N_{\eta c}$ and all i, $1\leq i\leq 2$, one has $a.s_i\in N_{\eta c}$, so that $a.s_i$ is a finite term.

Proof : Easy induction on the size of a. □

3.3.9. Theorem : For any F, the model $\mathcal{B}_\eta$ is an initial least fixpoint η-model. If F separates N_η then $\mathcal{B}_\eta$ is order-extensional and is initial among extensional and order-extensional models.

Proof : The first part is easy. Assume that F separates N_η. Let $A,A'\in N_{\eta c}^{\sigma\infty}$, assume $A.B<A'.B$ for all B. For showing $A<A'$ we must show $a<A'$ for all $a<A$. By hypothesis, $a.s_1<A'.s_1$ and $a.s_2<A'.s_2$. Since $a.s_1$ and $a.s_2$ are finite by 3.3.8, there exists $a'<A'$ such that $a.s_1<a'.s_1$ and $a.s_2<a'.s_2$, by application of 3.2.4.. Hence $a<a'<A'$ by 3.3.7.. Therefore N_η is value-extensional. Let $A,A'\in N_\eta^{\sigma\infty}$, let $\rho\in$Env. We must show that $A\rho<A'\rho$ for all ρ implies $A<A'$. Assume $a<A$. Then if a has its free variables in $\{x_1,\ldots x_k\}$, we have $a\rho=(\lambda\vec{x}.a).\overrightarrow{\rho(x_i)}$, and the proof goes as before, taking environments with values in $\{s_1,s_2\}$. Hence $\mathcal{B}_\eta$ is order-extensional. Its initiality is obvious. □

4. FULLY ABSTRACT MODELS OF PL

4.1. The operational ordering

We recall the definitions of [20,24].

4.1.1. Definition : Let F a set of function symbols and A an interpretation of F. A *program* is a closed term of ground type, and programs are called $P,P',P_1,\ldots$. We write $P \sqsubset_{op} P'$ if $A[\![BT(P)]\!] \leq A[\![BT(P')]\!]$, see 2.1.1.. For M,M' arbitrary terms, we write $M \subset_{op} M'$ if $P[M] \subset_{op} P[M']$ for any context $P[\ ]$ such that $P[M]$ and $P[M']$ are programs. We write $M \equiv_{op} M'$ if $M \subset_{op} M'$ and $M' \subset_{op} M$. ○

Notice that $P \sqsubset_{op} P'$ is not trivially equivalent to $P \subset_{op} P'$. This will indeed follow from 4.1.3.

4.1.2. Lemma : If $M < M'$ then $M \subset_{op} M'$. If $M \overset{*}{\to} M'$, then $M \equiv_{op} M'$. If $M \subset_{op} M'$, then $C[M] \subset_{op} C[M']$ for all $C[\ ]$.

Proof : Immediate using 3.2.4. □

A very useful characterization of $\subset_{op}$ is given by "context lemma" [20]. We give a new proof of this result, obtained with J.J. Levy. Notice that Milner's original proof does not extend simply to the λ-calculus

4.1.3. Proposition : Let $\sigma = \sigma_1 \times \sigma_2 \times \ldots \times \sigma_m \to \kappa$, let M_1^σ, M_2^σ be two terms having their free variables among $\{x_1^{\tau_1}, x_2^{\tau_2}, \ldots, x_n^{\tau_n}\}$. Let $\overline{M}_i = \lambda x_1^{\tau_1} x_2^{\tau_2} \ldots x_n^{\tau_n}. M_i$ for i=1,2. Then $M_1 \subset_{op} M_2$ holds iff $\overline{M}_1 \vec{N} \sqsubset_{op} \overline{M}_2 \vec{N}$ holds for any vector $\vec{N}$ such that $\overline{M}_1 \vec{N}$ and $\overline{M}_2 \vec{N}$ are programs.

Proof : The "only if" direction is obvious, considering the context $P[\ ] = \lambda \vec{x}.[\ ]\vec{N}$. Conversely, assume the condition satisfied

and let $P[\ ]$ be a program context such that $P[M_1]$ and $P[M_2]$ are programs. We show $A[\![P[M_1]]\!] \leq A[\![P[M_2]]\!]$ by induction on n, and for each n we show $A[\![a]\!] \leq A[\![P[M_2]]\!]$ for each $a < BT(P[M_1])$. This is of course trivial if $P[M_1]$ has no hnf. Otherwise let $\|a\|$ be the size of a, let $d(P[M_1])$ be the length of the head reduction of $P[M_1]$ (i.e. the left most outermost reduction to head normal form, see [14, 31]), and let $n(P[\ ])$ be the number of occurrences of the hole $[\ ]$ in $P[\ ]$. We operate by induction on $<\|a\|, d(P[M_1]), n(P[\ ])>$. The context $P[\ ]$ has only four possible forms.

Form 1 : $P[\ ] = [\ ]C[\ ]$.

Then M_1 and M_2 must be closed. Let $P'[\ ] = M_1 C[\ ]$. Clearly $P[M_1] = P'[M_1]$ and the induction works since $n(P'[\ ]) = n(P[\ ]) - 1$. Therefore $A[\![a]\!] \leq A[\![P'[M_2]]\!]$. But the terms $C[M_2]$ are closed, so that the global hypothesis gives $A[\![P[M_2]]\!] = A[\![M_1 C[M_2]]\!] \leq A[\![M_2 C[M_2]]\!]$ and finally $A[\![a]\!] \leq A[\![P[M_2]]\!]$ which is the required result.

Form 2 : $P[\ ] = fP[\ ]$. Then either $a = \Omega$ and the result is trivial or $a = fa$ and the induction on $\|a\|$ works.

Form 3 : $P[\ ] = (\lambda x.C_0[\ ])C_0'[\ ]C[\ ]$.

Let $D_0[\ ]$ be the context with two holes $[\ ]$ and $[\ ]_1$ where the occurrences of $[\ ]$ in $C_0[\ ]$ are renamed into $[\ ]_1$ provided that they are not in a subcontext $\lambda x.D[\ ]$ of $C_0[\ ]$ (formal definition left to the reader).

Case 3.1. : The hole $[\ ]_1$ has at least one occurrence in $D_0[\ ]$ and x is free in $M_1 M_2$. Let then

$$P'[\ ] = (\lambda x.D_0[\ ]\,[M_1]_1)C_0'[\ ]C[\ ].$$

Clearly $P'[M_1]=P[M_1]$ and $n(P'[\ ])<n(P[\ ])$. Hence by induction $A[\![a]\!]\leq A[\![P'[M_2]]\!]$. Let $P_1[\ ]=D_0[M_2]\ [\ ]_1\overrightarrow{C[M_2]}$. Then $P'[M_2]\to P_1[M_1[C_0[M_2]/x]]$ and $P[M_2]\to P_1[M_2[C_0[M_2]/x]]$. But $C_0[M_2]$ is closed, and $M'_1=M_1[C_0[M_2]/x]$, $M'_2=M_2[C_0[M_2]/x]$ have their free variables among $\{x_1,x_2,\ldots,x_n\}-\{x\}$. Then by induction $A[\![P'[M_2]]\!]=A[\![P_1[M'_1]]\!]\leq A[\![P[M_2]]\!]$, and the result follows.

Case 3.2. : Not in case 3.1.. Let then $P'[\ ]=C_0[\ ]\ [C'_0[\ ]/x]\overrightarrow{C[\ ]}$. Then $P[M_1]\to P'[M_1]$, $P[M_2]\to P'[M_2]$, and $P[M_1]\to P'[M_1]$ is the first step of the head reduction of $P[M_1]$. Hence $A[\![a]\!]\leq A[\![P'[M_2]]\!]=A[\![P[M_2]]\!]$ by induction on $\ell(P[\ ])$.

Form 4 : $P[\ ]=YC_0[\ ]\overrightarrow{C[\ ]}$. As in case 3.2., setting $P'[\ ]=C_0[\ ](YC_0[\ ])\overrightarrow{C[\ ]}$. □

4.1.4. Corollary : If $M_1^{\sigma\to\tau}$ and $M_2^{\sigma\to\tau}$ are closed, then $M_1 c_{op} M_2$ holds iff $M_1 N c_{op} M_2 N$ holds for all closed N. For all M_1^σ, M_2^σ, $M_1 c_{op} M_2$ holds iff $M_1\rho c_{op} M_2\rho$ holds for all substitution ρ by closed terms. If P_1 and P_2 are programs, then $P_1 c_{op} P_2$ holds iff $A[\![P_1]\!]\leq A[\![P_2]\!]$.

Proof : Immediate. □

4.2. Fully abstract models

The next result shows that any model is adequate w.r.t. the operational ordering c_{op}.

4.2.1. Proposition : Let M be a least fixpoint model of A. Then $A[\![P]\!]=M[\![P]\!]\perp$ for any program P, and $M_1\sqsubseteq_M M_2$ implies $M_1 c_{op} M_2$ for any M_1, M_2 of type σ.

Proof : The first part is clear from the structure of $N^{K\infty}_{\eta c}$ and

from the approximation continuity of M. The second part immediately follows. □

4.2.2. Definition : A model M of A is *fully abstract* w.r.t. A if $M_1 \sqsubseteq_M M_2$ is equivalent to $M_1 \subset_{op} M_2$ for all M_1, M_2. ○

Not all models are fully abstract, as shown for example in [24]. We know of two constructions and characterizations of fully abstract models. The first one is due to Milner [20] and applies to *articulate interpretations*. Milner's construction for combinatory logic easily extends to PL (and indeed one can simplify the construction in replacing the inverse limit construction by a directed ideal completion, see [4]). The obtained model is unique (up to isomorphism) and characterized by the fact that any isolated point is definable by a term. The other construction was already presented in section 3, and concerns the symbolic interpretation E of 2.2.1., which is not articulate. (the projections are not definable).

4.2.3. Proposition : The model $\mathcal{B}_\eta$ is fully abstract w.r.t. the symbolic interpretation E when F separates N_η.

Proof : Immediate from 3.3.9. and 4.1.3. □

Here the fully abstract model is not unique, and moreover *any* model is fully abstract.

4.2.4. Lemma : Assume M, M' are models of A such that M is fully abstract w.r.t. A and such that there exists a morphism $\theta : M \to M'$. Then M' is also fully abstract.

Proof : Immediate. □

4.2.5. Corollary : Any model of the symbolic interpretation E is fully abstract when F separates N_η.

Proof : Follows from 4.2.3., 4.2.4. and the initiality of $\mathcal{B}_\eta$. □

Therefore the situation of the symbolic interpretation is quite different from that of articulate interpretations. If M is a model of E it is impossible to find two terms M_1, M_2 such that $M_1 \sqsubseteq M_2$ and $M_1 \sqsupseteq_{op} M_2$, mainly because of the lack of definable functions. On the contrary, this is generally easy in models of articulate interpretations, as shown by Milner's uniqueness theorem [20]. Also in articulate interpretations the only morphisms from the fully abstract models are isomorphisms. All these remarks suggest that it might be difficult to construct fully abstract models of arbitrary interpretations by quotients of the fully abstract model $\mathcal{B}_\eta$ of E.

5. CATEGORICAL MODELS

We briefly recall some preliminaries on categories, see [4, 17] for details. Then we construct models of Λ, and study when they are approximation continuous. We finish with models of PL.

5.1. Preliminaries

5.1.1. Definition :

We denote the categories by $C, C', \ldots$, their objects by $\mathcal{C}, \mathcal{C}', \mathcal{C}_1, \ldots$, the arrows by $f, g, \ldots$, the set of arrows from $\mathcal{C}$ to $\mathcal{C}'$ by $C[\mathcal{C}, \mathcal{C}']$, the identity of $\mathcal{C}$ by $1_{\mathcal{C}}$. A *retraction* (resp *coretraction*) is an arrow $\theta : \mathcal{C}_1 \to \mathcal{C}_2$ such that there exists $\theta' : \mathcal{C}_2 \to \mathcal{C}_1$ satisfying $\theta \circ \theta' = 1$ (resp $\theta' \circ \theta = 1$). A *Cartesian closed category* is a quadruple $(C, T, \times, \to)$ where T is a specified terminal object, where $\times$ is a specified product functor (right adjoint to the diagonal) and where $\to$ is a specified exponentiation functor (such that $\mathcal{C} \to .$ is right adjoint to $\mathcal{C} \times .$ for any object $\mathcal{C}$). In more elementary terms the following conditions must be met.

- One must specify a *terminal object* T, i.e. an object such that for any $\mathcal{C}$ there exists a unique arrow $\mathcal{C}^T$ from $\mathcal{C}$ to T. In categories of sets and functions, T is generally a one-point set and the set $C[T, \mathcal{C}]$ is in immediate bijection with the set $\mathcal{C}$. This is how elements are turned into arrows in a standard way, which we shall use constantly.

- For any objects $\mathcal{C}_1$ and $\mathcal{C}_2$, one must specify an object $\mathcal{C}_1 \times \mathcal{C}_2$, called the *product* of $\mathcal{C}_1$ and $\mathcal{C}_2$, together with a pair of arrows $\pi_1 : \mathcal{C}_1 \times \mathcal{C}_2 \to \mathcal{C}_1$ and $\pi_2 : \mathcal{C}_1 \times \mathcal{C}_2 \to \mathcal{C}_2$, called the *projections*, which are such that for any $f_1 : \mathcal{C} \to \mathcal{C}_1$ and $f_2 : \mathcal{C} \to \mathcal{C}_2$ there exists a unique arrow $\langle f_1, f_2 \rangle : \mathcal{C} \to \mathcal{C}_1 \times \mathcal{C}_2$ satisfying $f_1 = \pi_1 \circ \langle f_1, f_2 \rangle$ and

$f_2=\pi_2\circ\langle f_1,f_2\rangle$, see fig.1. Given two arrows $f_1:C_1\to C_1'$ and $C_2\to C_2'$ one defines the *product arrow* $f_1\times f_2:C_1\times C_2\to C_1'\times C_2'$ by $f_1\times f_2=\langle f_1\circ\pi_1', f_2\circ\pi_2'\rangle$, see fig. 2.

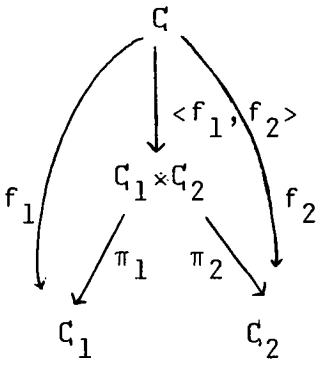


Fig. 1

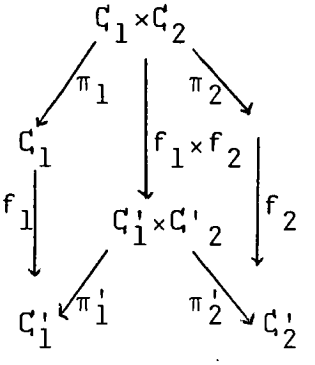


Fig. 2

In categories of sets and functions, $C_1\times C_2$ is generally the usual cartesian product, with $\langle f_1,f_2\rangle(c)=(f_1(c),f_2(c))$ and $(f_1\times f_2)(c_1,c_2)=(f_1(c_1),f_2(c_2))$. In all cases notice that the set $C[C,C_1\times C_2]$ is in immediate bijection with the cartesian product $C[C,C_1]\times C[C,C_2]$.

- For any objects C_1 and C_2, one must specify an object $(C_1\to C_2)$, called the *exponentiation* of C_1 and C_2, together with an arrow $app:(C_1\to C_2)\times C_1\to C_2$ having the following universal property : for any object C, for any arrow $f:C\times C_1\to C_2$, there exists a unique arrow $curry(f):C\to(C_1\to C_2)$ such that $f=app\circ(curry(f)\times 1)$, see fig. 3. Given $f_1:C_1'\to C_1$ and $f_2:C_2\to C_2'$, one defines $(f_1\to f_2):(C_1\to C_2)\to(C_1'\to C_2')$ by $(f_1\to f_2)=curry\,(g\circ app\circ(1\times f))$, see fig. 4.

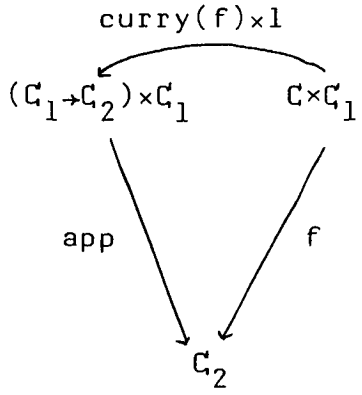


Fig. 3

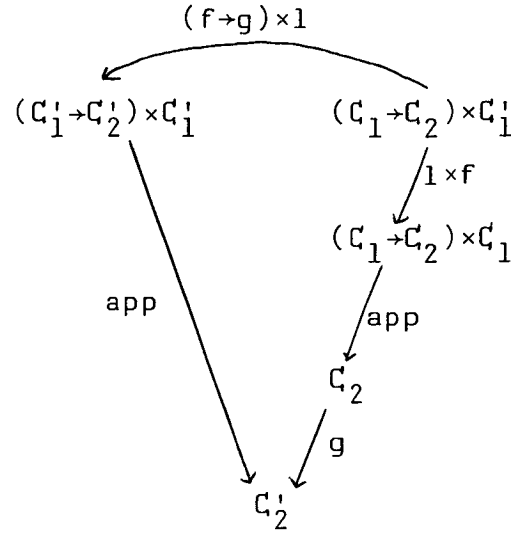


Fig. 4

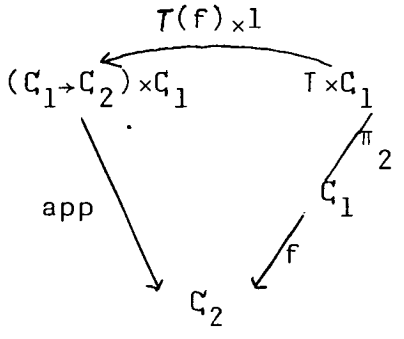
Notice that $(f_1 \to f_2) \circ (g_1 \to g_2) = (g_1 \circ f_1 \to f_2 \circ g_2)$: the functor $\to$ is contravariant in its first argument.

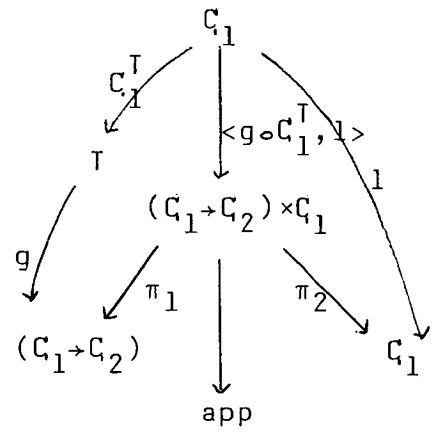
In categories of sets and functions, $(C_1 \to C_2)$ is generally the set of arrows from C_1 to C_2, app is just functional application, and curry(f) is just classically determined by $\text{curry}(f)(\alpha)(\beta) = f(\alpha,\beta)$. See [6] for examples not in this case.

In all cases there is an important bijection between the arrow set $C[C_1,C_2]$ and the arrow set $C[T,(C_1 \to C_2)]$, i.e. the set of "points" of $(C_1 \to C_2)$ as said before. Define the two mappings $\mathcal{T}: C[C_1,C_2] \to C[T,(C_1 \to C_2)]$ and $\mathcal{F}: C[T,(C_1 \to C_2)] \to C[C_1,C_2]$ in the following way :

$$\mathcal{T}(f) = \text{curry}(f \circ \pi_2)$$

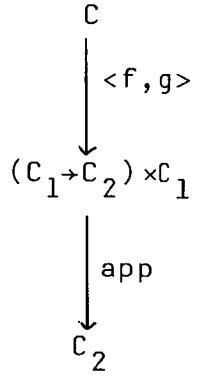
$$\mathcal{F}(g) = \text{app} \circ \langle g \circ C_1^T, 1 \rangle$$





We leave to the reader to check $\mathcal{F}(\mathcal{T}(f))=f$ and $\mathcal{T}(\mathcal{F}(g))=g$ for all f and g, and also that $\mathcal{T}$ and $\mathcal{F}$ have the simple meaning described above for functions.

To finish we define an essential operation, which will be the application in all our models. Given $f : \mathcal{C} \to (\mathcal{C}_1 \to \mathcal{C}_2)$ and $g : \mathcal{C} \to \mathcal{C}_1$, we define $f*g : \mathcal{C} \to \mathcal{C}_2$ by $f*g = app \circ \langle f,g \rangle$



○

As examples of cartesian closed categories we have :

- The category of sets and functions. Any one point set is terminal.
- The category of cpo's and continuous functions. The terminal object is the cpo $\{\perp\}$. The exponentiation is the set of functions ordered by the classical extensional ordering $f \subset g$ if $f(x) \subset g(x)$ for all x. Two particularly interesting subcategories which are also cartesian closed are Plotkin's category SFP [23] and the category of ω-algebraic consistently complete cpo's [26].
- The categories of stable functions of [3,4]. Here the exponentiation ordering of functions is not any more the extensional ordering.
- The category of sequential algorithms on concrete data structures [6]. Its arrows are not any more functions.

To end the section we give some useful basic lemmas :

5.1.2. Lemma : Let $f:\mathcal{C}_1 \times \mathcal{C}_2 \to \mathcal{C}_3$ and $g:\mathcal{C}_4 \to \mathcal{C}_1$.

Then $curry(f)\circ g = curry(f\circ(g\times 1_{C_2}))$.

Proof : One has $app\circ((curry(f)\circ g)\times 1) = app\circ(curry(f)\times 1)\circ(g\times 1)$ and the result follows. □

5.1.3. Lemma : Let $f:C\times C_1\to C_2$ and $g:C\to C_1$. Then $curry(f)*g = app\circ\langle curry(f),g\rangle = f\circ\langle 1_C,g\rangle$.

Proof : One has $\langle curry(f),g\rangle = (curry(f)\times 1)\circ\langle 1,g\rangle$ and the result follows. □

5.1.4. Lemma : Let $f:C\to C'$ and $g:T\to C$. Then $T(f)*g = f\circ g$.

Proof : Immediate. □

5.2. Categorical models of Λ.

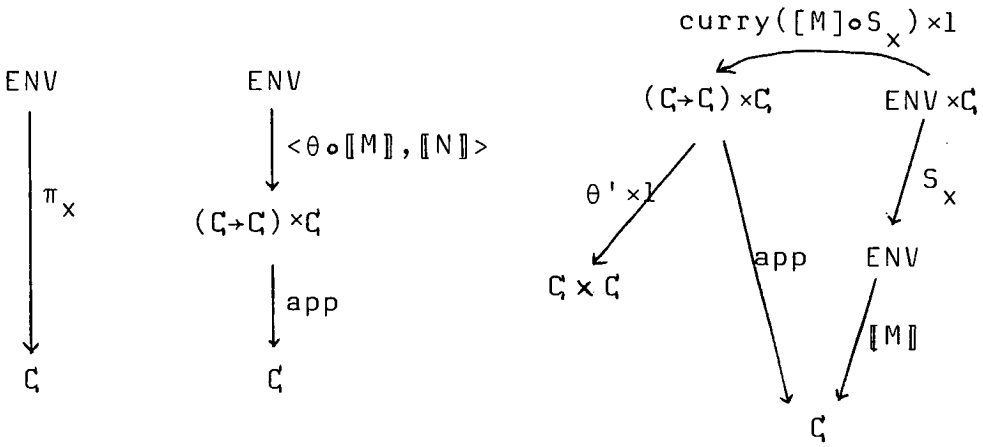
5.2.1. Definition : A *categorical model* M is given by a cartesian closed category C, an object C of C and a pair of arrows $\theta:C\to(C\to C)$ and $\theta':(C\to C)\to C$ with $\theta\circ\theta'=1$ (hence θ is a retraction). For simplicity we shall assume that C is closed by ω-products, although this is not necessary : it simply avoids counting variables, and allows to introduce an object ENV in C which is the categorical product of denumerably many copies of C indexed by the variables in V. Let $\pi_x:ENV\to C$ be the x-projection, let $S_x:ENV\times C\to ENV$ be defined by $\pi_x\circ S_x=\pi_2$ and $\pi_y\circ S_x=\pi_y\circ\pi_1$ for $y\neq x$ (intuitively $S_x(\rho,\alpha)=\rho[x\leftarrow\alpha]$). We start by defining a semantics from M, setting $E_M=C[ENV,C]$ and simply calling $[\![\ ]\!]$ the semantic function. We shall later on turn this semantics into a model. The semantic mapping $[\![\ ]\!]$ is defined in the following way :

(i) $[\![x]\!]=\pi_x$

(ii) $[\![MN]\!]=(\theta\circ[\![M]\!])*[\![N]\!]$

(iii) $[\![\lambda xM]\!]=\theta'\circ curry([\![M]\!]\circ S_x)$.

Of course these equations are just obtained from (var), (app) and (lambda) by removing elements ρ and d, in a classical category-theoretic style.



Equation (i) Equation (ii) Equation (iii)

Let us give the main result, the proof of which will require some lemmas.

5.2.2. Theorem : Let $\mathcal{M}$ be a categorical model. Then $[\![\ ]\!]$ is a semantics. It is a η-semantics if θ is an isomorphism.

The first lemma is the abstract presentation of the elementary properties $(\rho[x\leftarrow d])[x\leftarrow d']=\rho[x\leftarrow d']$ and $(\rho[x\leftarrow d])[y\leftarrow d']=(\rho[y\leftarrow d'])[x\leftarrow d]$ for $x\neq y$.

5.2.3. Lemma : For any $x,y\in V$, $x\neq y$ one has $S_x\circ(S_x\times 1_C) = S_x\circ(\pi_1\times 1_C)$ and $S_y\circ(S_x\times 1_C) = S_x\circ(S_y\times 1_C)\circ\langle\pi_1\times 1_C,\pi_2\circ\pi_1\rangle$

Proof : Elementary verifications (compose with π_x,π_y,π_z for $z\neq x,y$. □

5.2.4. Lemma : Assume x not free in M. Then $[\![M]\!]\circ S_x = [\![M]\!]\circ\pi_1$.

Proof : By induction on the size of M. Let us treat for example the case $M=\lambda y M'$, $y\neq x$

$$\begin{aligned}
[\![M]\!]\circ S_x &= \theta'\circ curry([\![M']\!]\circ S_y)\circ S_x \\
&= \theta'\circ curry([\![M']\!]\circ S_y\circ(S_x\times 1)) \quad \text{by 5.1.2.} \\
&= \theta'\circ curry([\![M']\!]\circ S_x\circ(S_y\times 1)\circ\langle\pi_1\times 1,\pi_2\circ\pi_1\rangle) \quad \text{by 5.2.3.} \\
&= \theta'\circ curry([\![M']\!]\circ\pi_1\circ(S_y\times 1)\circ\langle\pi_1\times 1,\pi_2\circ\pi_1\rangle) \quad \text{by induction} \\
&= \theta'\circ curry([\![M']\!]\circ S_y\circ\pi_1\circ\langle\pi_1\times 1,\pi_2\circ\pi_1\rangle) \quad \text{for } \pi_1\circ(S_y\times 1)=S_y\circ\pi_1 \\
&= \theta'\circ curry([\![M']\!]\circ S_y\circ(\pi_1\times 1)) \\
&= \theta'\circ curry([\![M']\!]\circ S_y)\circ\pi_1 \quad \text{by 5.1.2.} \\
&= [\![M]\!]\circ\pi_1 \quad \square
\end{aligned}$$

5.2.5. Lemma : If y is not free in M then $[\![\lambda y M[y/x]]\!]=[\![\lambda x M]\!]$.

Proof : It is sufficient to show $[\![M[y/x]]\!]\ S_y = [\![M]\!]\ S_x$.

The cases $M=x$, $M=z\neq y$ are immediate. The case $M=M_1M_2$ easily works by induction. The case $M=\lambda x M'$ is immediate from 5.2.4. The case $M=\lambda y M'$ reduces to the case $M=\lambda z M'$, $z\neq x$, $z\neq y$, by definition of the substitution. Then we have

$$\begin{aligned}
[\![M[y/x]]\!]\circ S_y &= [\![\lambda z.M'[y/x]]\!]\circ S_y \\
&= \theta'\circ curry([M'[y/x]]\!]\circ S_z)\circ S_y \\
&= \theta'\circ curry([\![M'[y/x]]\!]\circ S_z\circ(S_y\times 1)) \quad \text{by 5.1.2.} \\
&= \theta'\circ curry([\![M'[y/x]]\!]\circ S_y\circ(S_z\times 1)\circ\langle\pi_1\times 1,\pi_2\circ\pi_1\rangle) \quad \text{by 5.2.3.} \\
&= \theta'\circ curry([\![M']\!]\circ S_x\circ(S_z\times 1)\circ\langle\pi_1\times 1,\pi_2\circ\pi_1\rangle) \quad \text{by induction} \\
&= \theta'\circ curry([\![M']\!]\circ S_z\circ(S_x\times 1)) \\
&= \theta'\circ curry([\![M']\!]\circ S_z)\circ S_x \quad \text{by 5.1.2.} \\
&= [\![M]\!]\circ S_x \quad \square
\end{aligned}$$

5.2.6. Lemma : For all x,M,N, one has $[\![(\lambda x M)N]\!] = [\![M[N/x]]\!]$.

Proof : Let $A=[\![(\lambda x M)N]\!]$ and $B=[\![M[N/x]]\!]$. Then

$A=app\circ\langle\theta\circ\theta'\circ curry([\![M]\!]\circ S_x),[\![N]\!]\rangle$

$A=[\![M]\!]\circ S_x\circ\langle 1_{ENV},[\![N]\!]\rangle$ by 5.1.3. and $\theta\circ\theta'=1$

We operate by induction on the size of M.

case 1 : M=x. Then $A=\pi_x\circ S_x\circ\langle 1,[\![N]\!]\rangle = \pi_2\circ\langle 1,[\![N]\!]\rangle = [\![N]\!] = B$

case 2 : M=y. Then $A=\pi_y\circ S_x\circ\langle 1,[\![N]\!]\rangle = \pi_y\circ\pi_1\circ\langle 1,[\![N]\!]\rangle = \pi_y = B$

case 3 : $M=M_1M_2$. Then :

$A=app\circ\langle\theta\circ[\![M_1]\!],[\![M_2]\!]\rangle\circ S_x\circ\langle 1,[\![N]\!]\rangle$

$A=app\circ\langle\theta\circ[\![M_1]\!]\circ S_x\circ\langle 1,[\![N]\!]\rangle,[\![M_2]\!]\circ S_x\circ\langle 1,[\![N]\!]\rangle\rangle$

$A=app\circ\langle\theta\circ[\![M_1[N/x]]\!],[\![M_2[N/x]]\!]\rangle$ by induction

$A=[\![M_1[N/x]M_2[N/x]]\!]$

$A=B$

case 4 : $M=\lambda x M'$. Then :

$A=\theta'\circ curry([\![M']\!]\circ S_x)\circ S_x\circ\langle 1,[\![N]\!]\rangle$

$A=\theta'\circ curry([\![M']\!]\circ S_x)\circ\pi_1\circ\langle 1,[\![N]\!]\rangle$ by 5.2.4.

$A=\theta'\circ curry([\![M']\!]\circ S_x)$

$A=B$

case 5 : $M=\lambda y M'$, $y\neq x$, y not free in N (using possibly α-conversion as justified by 5.2.5.)

$A=\theta'\circ curry(\llbracket M'\rrbracket\circ S_y)\circ S_x\circ\langle 1,\llbracket N\rrbracket\rangle$

$A=\theta'\circ curry(\llbracket M'\rrbracket\circ S_y\circ(S_x\times 1)\circ(\langle 1,\llbracket N\rrbracket\rangle\times 1))$ by 5.1.2.

$A=\theta'\circ curry(\llbracket M'\rrbracket\circ S_x\circ(S_y\times 1)\circ\langle\pi_1\times 1,\pi_2\circ\pi_1\rangle\circ(\langle 1,\llbracket N\rrbracket\rangle\times 1))$ by 5.2.3.

$A=\theta'\circ curry(\llbracket M'\rrbracket\circ S_x\circ\langle S_y,\llbracket N\rrbracket\circ\pi_1\rangle)$ by elementary computations

$A=\theta'\circ curry(\llbracket M'\rrbracket\circ S_x\circ\langle S_y,\llbracket N\rrbracket\circ S_y\rangle)$ by 5.2.4.

$A=\theta'\circ curry(\llbracket M'\rrbracket\circ S_x\circ\langle 1,\llbracket N\rrbracket\rangle\circ S_y)$

$A=\theta'\circ curry(\llbracket M'[N/x]\rrbracket\circ S_y)$ by induction

$A=\llbracket\lambda y.M'[N/x]\rrbracket$

$A=B$

□

5.2.7. Lemma : Assume that θ is an isomorphism, let x not free in M. Then $\llbracket\lambda x.Mx\rrbracket=\llbracket M\rrbracket$.

Proof : $\llbracket\lambda x.Mx\rrbracket=\theta'\circ curry(app\circ\langle\theta\circ\llbracket M\rrbracket,\pi_x\rangle\circ S_x)$

$=\theta'\circ curry(app\circ\langle\theta\circ\llbracket M\rrbracket\circ S_x,\pi_x\circ S_x\rangle)$

$=\theta'\circ curry(app\circ\langle\theta\circ\llbracket M\rrbracket\circ\pi_1,\pi_2\rangle)$ by 5.2.4.

$=\theta'\circ curry(app\circ((\theta\circ\llbracket M\rrbracket)\times 1))$

$=\theta'\circ curry(app)\circ\theta\circ\llbracket M\rrbracket$

$=\llbracket M\rrbracket$ for $curry(app)=1$ and $\theta'\circ\theta=1$.

Proof of theorem 5.2.2. : The satisfaction of (cong) is immediate from the definition of $\llbracket\ \rrbracket$. The satisfaction of (conv) then follows from 5.2.5. and 5.2.6. If θ is an isomorphism, then $\llbracket\ \rrbracket$ is a η-semantics by 5.2.7. □

To turn M into a model, it is of course natural to set $D_M=C[T,C]$, with $\bullet:D_M\times D_M\to D_M$ defined by $d\bullet d'=(\theta\circ d)*d'$. Then Env_M is just $C[T,ENV]$. One could of course keep $E_M=C[ENV,C]$ and $\llbracket\ \rrbracket$ as before, defining $eval:E_M\times Env_M\to D_M$ by $eval(\phi,\rho)=\phi\circ\rho$. But we prefer to compose the semantics $\llbracket\ \rrbracket$ with the isomorphism T, in order to define eval from * and therefore to gain uniformity.

5.2.8. Definition : Let M be a categorical model, and call also M the premodel defined in the following way :

- $D_M = C[T, C]$ with $d \cdot d' = (\theta \circ d) * d'$
- $E_M = C[T, (ENV \to C)]$ with $eval\ (\phi, \rho) = \phi * \rho$

Let the function $M[\![\ ]\!] : \Lambda \to E_M$ be defined by $M[\![M]\!] = T([\![M]\!])$. Notice that $Env_M = C[T, ENV]$ with $\rho(x) = \pi_x \circ \rho$ and $\rho[x \leftarrow d] = S_x \circ <\rho, d>$.

5.2.9. Theorem : Any categorical model defines a model of Λ. Moreover this model is a η-model if θ is an isomorphism.

Proof : The only equations remaining to be checked are (var), (app), (lambda) and (free). Notice that $M[\![M]\!]\rho = [\![M]\!] \circ \rho$ for all M by 5.1.4.

(var) : $M[\![x]\!]\rho = \pi_x \circ \rho = \rho(x)$ by definition

(app) : $M[\![MN]\!]\rho = app \circ <\theta \circ [\![M]\!], [\![N]\!]> \circ \rho$

$= app \circ <\theta \circ [\![M]\!] \circ \rho, [\![N]\!] \circ \rho>$

$= M[\![M]\!]\rho * M[\![N]\!]\rho$

(lambda) : $M[\lambda x M]\rho \cdot d = (\theta \circ \theta' \circ curry([\![M]\!] \circ S_x) \circ \rho) \cdot d$

$= [\![M]\!] \circ S_x \circ <\rho, d>$ by elementary computation

$= M[\![M]\!]\rho[x \leftarrow d]$

(free) : Immediate induction on the size of M.

Notice that M needs not be extensional to be a η-model.

There are several ways of obtaining retractions of the form $\Theta : (C \to C) \to C$:

- One may construct category-theoretic solutions to $C = (C \to C)$ as in the D^∞ construction [28]. A general solution to this problem is presented in [30].

- One may construct a universal domain such as P^ω, T^ω. See [25,29,30].

- One may apply a set-theoretic construction as in [6]. The domain there is again a solution to $D = (D \to D)$, but no category-theoretic arguments are needed to build it.

Most constructions involve ordering and continuity. However retractions $\Theta : (\mathcal{C} \to \mathcal{C}) \to \mathcal{C}$ do not necessarily give approximation continuous models, see Park [21]. Wadsworth [31,32] shows that Scott's original D^∞ is indeed approximation continuous, and its proof applies equally well to P^ω, T^ω, the stable models of [3,] and the sequential algorithms model of [6]. We believe that Wadsworth's proof extends to categories satisfying appropriate conditions, but we did not work out that point.

5.3. Categorical models of PL

The construction of categorical models of PL is very similar, but of course there is no need for solving domain equations. We need cpo's there, and the natural way to introduce them is to consider ordered categories as in [0,33], i.e. categories where every set of arrows $C[\mathcal{C},\mathcal{C}']$ is turned into a cpo by an ordering $\leqslant_{\mathcal{C},\mathcal{C}'}$ in such a way that the composition is order continuous. We shall give no details here, referring to [4]. We shall just state the result.

5.3.1. Definition

A PL-category is given by an ordered cartesian closed category C together with the assignment of an object $\mathcal{C}^\tau$ for each type $\tau \in T$ with the following conditions :

(i) The composition is left-strict, i.e. $\perp \circ f = f$ for all f

(ii) The $\langle\ ,\ \rangle$ operation is order-continuous and strict.

(iii) The operation * is left-strict, i.e. $\perp * f = \perp$ for all f

(iv) The curry function is order-continuous

(v) For each $\sigma,\tau \in T$ the object $\mathcal{C}^{(\sigma\to\tau)}$ is the exponentiation of $\mathcal{C}^{\sigma}$ and $\mathcal{C}^{\tau}$.

5.3.2. Theorem: Let A be an interpretation of (K,F), let C be a PL-category such that $C[\perp,\mathcal{C}^{K}]$ and D_A^K are equal as cpo's for each $\kappa \in K$. For any $f^{\sigma\to\kappa} \in F$ let $f_C \in C[\perp,(\mathcal{C}^{\sigma}\to\mathcal{C}^{K})]$ be an arrow of C such that:

$$f_C * \langle \alpha_1,\alpha_2,..,\alpha_n\rangle = A(f)(\alpha_1,\alpha_2,..,\alpha_n).$$

For each $\sigma \in T$ let $D_M^{\sigma} = C[\perp,\mathcal{C}^{\sigma}]$ and $E_M^{\sigma} = C[\perp,(ENV\to C^{\sigma})]$ as cpo's, define eval and • as the * functions. Define the semantic function $M[\![\ ]\!] : PL^{\sigma}\to E_M^{\sigma}$ as $M[\![\ ]\!] = \mathcal{T}\circ[\![\ ]\!]$ where $[\![\ ]\!]$ is defined as follows

$$[\![x]\!] = \Pi_x$$

$$[\![f]\!] = \mathcal{F}\circ f_C$$

$$[\![MN]\!] = [\![M]\!] * [\![N]\!]$$

$$[\![\lambda x M]\!] = curry([\![M]\!]\circ S_x)$$

$$[\![YM]\!] = curry(FIX\circ\Pi_2)$$

where $FIX = \bigsqcup_n \underbrace{(1*(1*(...*(1*\perp)...)))}_{n \text{ times}}$

(with $1 = 1_{\mathcal{C}^{\sigma}}$ and $\perp \in C[T,\mathcal{C}^{(\sigma\to\sigma)\to\sigma}]$ if M is of type σ).

Then M is a least fixpoint model of A and is approximation continuous.

□

CONCLUSION

We hope to have convinced the reader that our model definition is fruitful and nevertheless necessary to discuss correctly continuity and approximation continuity. The various syntactic constructions receive their main application in the study of full abstraction, while the categorical construction is applied in [6]. It would be interesting to study further approximation continuity and term equalities in categorical models, in the line of [12].

APPENDIX A

The model equations :

Semantics

(cong) $S[\![M]\!] = S[\![N]\!] \Rightarrow \forall\, C[\ ].S[\![C[M]]\!] = S[\![C[N]]\!]$

(conv) $M \overset{*}{\to} M' \Rightarrow S[\![M]\!] = S[\![M']\!]$

(sel) $\forall\, M,N,(\forall P.S[\![MP]\!] = S[\![NP]\!]) \Rightarrow S[\![M]\!] = S[\![N]\!]$

(se2) $\forall\, M,N,\ x$ not free in MN
$S[\![Mx]\!] = S[\![Nx]\!] \Rightarrow S[\![M]\!] = S[\![N]\!]$

(η) $\forall\, M,\ x$ not free in M, $S[\![\lambda xM]\!] = S[\![M]\!]$

Models

(var) $M[\![x]\!]\rho = \rho(x)$

(app) $M[\![MN]\!]\rho = (M[\![M]\!]\rho).(M[\![N]\!]\rho)$

(lambda) $M[\![\lambda xM]\!]\rho.d = M[\![M]\!]\rho[x \leftarrow d]$

(free) $M[\![M]\!]\rho = M[\![M]\!]\rho'$ if $\rho(x) = \rho'(x)$ for all x

Continuous models

(mon) $S[\![M]\!] \sqsubseteq S[\![N]\!] \Rightarrow S[\![C[M]]\!] \sqsubseteq S[\![C[N]]\!]$ for all $C[\ ]$

(lim) For all $X \subset \Lambda$, if the set $S[\![X]\!] = \{S[\![M]\!] \mid M \in X\}$ is directed and has lub $S[\![N]\!]$, then for all context $C[\ ]$ the set $S[\![C[X]]\!]$ has lub $S[\![C[N]]\!]$.

BIBLIOGRAPHY

[1] H.P. Barendregt, "The Type-free Lambda-calculus", Hanbook of Mathematical Logic, North-Holland, 1977, p.1091-1132.

[2] H.P. Barendregt, G. Longo, "Equality of Lambda-terms and Recursion-Theoretic reducibility in the model T^{ω}", University of Utrecht, Dept. of Mathematics, Preprint n.107.

[3] G. Berry, "Stable Models of Typed Lambda-Calculi", Proc. 5th Coll. "Automata, Languages and Programming" Udine, Italy, july 1978, Springer-Verlag Lecture Notes in Computer Science n. 62, pp. 62-90.

[4] G. Berry, "Modeles Completement Adequats et Stables des Lambda Calculs Types", These de Doctorat d'Etat, Universite Paris VII, march 1979.

[5] G. Berry, J.J. Levy, "Minimal and Optimal Computations of Recursive Programs", Journal of ACM, vol. 26, No 1, jan. 1979, pp. 148-175.

[6] G. Berry, P.L. Curien, "Sequential Algorithms on Concrete Data Structures", to appear in Theoretical Computer Science.

[7] G. Berry, "On the Definition of Lambda-Calculus Models", Proc International Coll. on Formalization of Programming Concepts, Peniscola, Spain, April 1981, to appear. INRIA Report to appear.

[8] C. Boehm, "Alcune Proprieta della Forme $\beta\eta$-normali del $\lambda\beta\eta$-Calcolo", Publicazioni dell' Istituto per le Applicazioni del Calcolo, n. 696, Roma, Italy, 1968.

[9] R. Hindley, G. Longo, "Lambda-Calculus Models and Extensionality", Universita di Pisa, Istituto di Scienze dell' Informazione,Note Scientifiche S-78-4, july 1978.

[10] G. Huet, Resolution d'Equations dans les Langages d'ordre $1,2,\ldots\omega$", These de Doctorat d'Etat, Universite Paris VII, 1976.

[11] G. Huet, J.J. Levy, "Computations in Nonambiguous Linear Term Rewriting Systems, IRIA-LABORIA Report n. 359, aug. 1979.

[12] M. Hyland, "A Syntactic Characterization of Equality in Some Models of the Lambda-Calculus", J. London Math. Soc., vol. 12, n.2, 1976.

[13] J. Lambeck, "Deductive Systems and Categories: 3. Cartesian Closed Categories, Intuitionistic Propositional Calculus and Combinatory Logic", Proc. Dalhousie Conference on

Toposes, Algebraic Geometry and Logic, Springer-Verlag Lecture Notes in Mathematics, n. 274, 1971, pp. 57-82.

[14] J.J. Levy, "An Algebraic Interpretation of the $\lambda\beta\kappa$- Calculus and an Application of a Labelled Lambda-Calculus", Theoretical Computer Science, vol. 2, n. 1, 1976.

[15] J.J. Levy, "Reductions Correctes et Optimales dans le Lambda-Calcul", These de Doctorat d'Etat, Universite Paris VII, 1978.

[16] J. Mc Carthy, "A new Eval Function", M.I.T. Artificial Intelligence memo 34, 1962.

[17] S. Mac Lane, "Categories for the Working Mathematician", Spinger-Verlag Graduate Texts in Mathematics, 1971.

[18] A.R. Meyer, "What is a Lambda-Calculus Model?", Laboratory for Computer Science report MIT/LCS/TM-171, M.I.T., August 1980.

[19] R. Milner, "Models of LCF", Comp. Science Dept. Memo AIM-186/CS-232, Stanford University, 1973..

[20] R. Milner, "Fully Abstract Models of Typed Lambda-Calculi", Theoretical Computer Science, vol. 4, No 1, feb. 1977, pp. 1-23.

[21] D.M.R. Park, "The Y-Combinator in Scott's Lambda-Calculus Models", Theory of Computation Report n. 13, University of Warwick, Great Britain, 1976.

[22] G.D.Plotkin, "The Lambda-Calculus is ω-incomplete", J. Symbolic Logic, vol. 39, pp. 313-317.

[23] G.D. Plotkin, "A Powerdomain Construction", SIAM Journal on Computing, vol. 5, n. 3, sept. 1976, pp. 452-488.

[24] G.D. Plotkin, "LCF Considered as a Programming Language", Theoretical Computer Science, vol.5, No 3, dec. 1977, pp. 223-256.

[25] G.D. Plotkin, "T^{ω} as a Universal Domain",Tech. Rep. 28, Dept. of Artificial Intelligence, Univ. of Edinburgh, 1977.

[26] G.D. Plotkin, "The Category of Complete Partial Orders: A Tool for Making Meanings", Proc. Summer School on Foundations of Artificial Intelligence and Computer Science, Istituto di Scienze dell' Informazione, Universita di Pisa, Italy, june 1978.

[27] D. Scott, C. Strachey, "Towards a Mathemetical Semantics of Programming Languages", Proc. Symposium on Computers and Automata, Pol. Inst. of Brooklynn, vol. 21, 1971, pp. 19-46.

[28] D. Scott, "Continuous Lattices", Proc. 1971 Dalhousie Conf. on Toposes, Algebraic Geometry and Logic, Springer-Verlag Lecture Notes in Mathematics No 274, 1971, pp. 97-136.

[29] D. Scott, "Data Types as Lattices", SIAM Journal on Computing, vol. 5, n. 3, sept. 1976, pp. 522-587.

[30] M. Smyth, G. Plotkin, "The Category-Theoretic Solution of Recursive Domain Equations", Proc. 18th Symp. on Foundations of Computer Science, Providence, R.I., 1977.

[31] C.P. Wadsworth, "The Relation between Computational and Denotational Properties for Scott's D^{∞} Model of the Lambda-Calculus", SIAM Journal on Computing, vol. 5, n. 3, sept. 1976, pp.488-522.

[32] C.P. Wadsworth, "Approximate Reductions and Lambda-Calculus Models", SIAM Journal on Computing, vol. 7, n. 3, aug. 1978, pp. 337-357.

[33] M. Wand, "Fixed-points Constructions in Order-Enriched Categories", Research Report, Indiana University.