IRIA

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P.105 78150 Le Chesnay
France
Tél. 954 90 20

Rapports de Recherche

N° 46

ON THE DEFINITION OF LAMBDA-CALCULUS MODELS

Gérard BERRY

Décembre 1980

ON THE DEFINITION OF LAMBDA-CALCULUS MODELS

Gérard BERRY

Centre de Mathématiques Appliquées
Ecole Nationale Supérieure des Mines de Paris
Sophia-Antipolis, 06560 VALBONNE, FRANCE

Résumé: La plupart des auteurs définissent les modèles du lambda-calcul comme des modèles logiques au sens de Tarski. Cependant nous avons construit des "modèles" sémantiquement naturels qui n'obéissent pas à ces définitions. Nous proposons une définition de modèle qui est strictement plus générale que les définitions classiques, englobe tous les cas connus, mais est un peu plus compliquée. Nous justifions cette définition par plusieurs exemples et la comparons aux définitions classiques.

Abstract: Most authors use a classical Tarski-like definition of lambda-calculus models. However in earlier papers we constructed semantic "models" which do not obey this definition. We propose a new definition of models which is strictly more general than the usual one, takes care of all known cases, but is slightly more complicated. We illustrate the need for this definition by various examples, and compare it to the usual definition.

ON THE DEFINITION OF LAMBDA-CALCULUS MODELS

G. BERRY

Centre de Mathematiques Appliquees
Ecole Nationale Superieure des Mines de Paris
Sophia-Antipolis, 06560 VALBONNE, FRANCE

1. INTRODUCTION

When the first lambda-calculus models where constructed by D. Scott [15] there was not much concern in giving a precise definition of what a model should be, since there was no doubt that the constructed objects ought to be models in any reasonable sense. Later on many more models were constructed and such a definition became necessary. Quite surprisingly the problem happened out to be not entirely trivial, and we know of four more or less independent definition attempts: by Barendregt [1], by Hindley-Longo[6], by Meyer[11] and by the author[3,5]. Although quite different in their presentation, the three first mentioned definitions turned out to be equivalent (it has to be noted that Meyer's definition is "algebraic", i.e. presented independently of the lambda-calculus itself). We shall call the corresponding models BHLM-models. Our own definition [3,5] is however different, both strictly more general and slightly more complicated. The purpose of this paper is to motivate the need for it on the basis of the following arguments:

- There are objects which we definitely want to be models and which are not BHLM-models. A typical example is the sequential algorithms model of [4]. Another example is the term model, which should clearly be initial, but is not a BHLM-model.

- The BHLM-definition is not adequate when one needs to add some extra structure to models: for example orderings, continuity etc...

- More generally there are many interesting properties of models in our sense which are not even expressible in the BHLM framework. We can express them in a natural way, and we can still express exactly as before all the properties of BHLM-models.

We shall develop our arguments using three main examples: the sequential algorithms model of [4], the term models and the interior of models [6]. We shall give no

details, the precise constructions being all completely presented in [3,4,5].

Let us explain briefly the differences between the approaches. The BHLM definitions follow a classical point of view in logic, generally known as the Tarski point of view: one starts from an algebra $<D,*>$ where $*$ is a binary operation on D called the application, and one interprets every lambda-term as a function from environments to values (an environment is an association of values to the variables). There is absolutely no problem when considering extensional models, i.e. models where $*$ is such that $d*d''=d'*d''$ for all d'' implies $d=d'$. However there is a difficulty with non-extensional models: we have two ways of binding variables, by environments and by λ's; the normal use of environments is just to pass values bound by λ's into subexpressions, so that the two bindings should behave exactly in the same way. However the binding by an environment is always extensional by definition, while the binding by a λ need not be. In other words the value of an expression as a function from environments to D is determined by its value in all possible environments, while the value of an abstraction in a given environment is not determined by its value on all possible arguments. Hence in all BHLM-definitions one adds an extra-condition which ensures that the binding by a λ has a "quasi-extensional" character. The condition has several different formulations, and we will be especially interested in Hindley-Longo's one, which is related to the interpretation of the (ξ) rule $M=N \Rightarrow \lambda x.M=\lambda x.N$. Denoting by $[[\ ]]$ the semantic function from terms to functions from environments to D, Hindley-Longo [6] require the following condition:

$(\xi 1)$ for all M,N, for all environment σ,
$(\forall d \in D.\ [[\lambda x.M]]\sigma * d = [[\lambda x.N]]\sigma * d) \Rightarrow [[\lambda x.M]]\sigma = [[\lambda x.N]]\sigma$

The condition clearly ensures "quasi-extensionality" by saying that in any extensionality class in D at most one point can be selected as a value for abstractions. The non-extensionals models T^{ω} and P^{ω} [14,15] typically satisfy the condition. The (ξ)-rule could also be interpreted in the following way:

(ξ) $[[M]]=[[N]] \Rightarrow [[\lambda x.M]]=[[\lambda x.N]]$

Then (ξ) and (β) would imply the equivalence of $[[M]]=[[N]]$ and $[[\lambda x.M]]=[[\lambda x.N]]$ for any x, which is more or less what we want. However Hindley and Longo reject this interpretation of (ξ) as being nonstandard in logic and furthermore having no semantic meaning.

Our definition was instead motivated by the construction of new denotational semantics of programming languages [2,3,4]. In particular we constructed in [4] a "model" in

which the objects are not functions but "sequential algorithms" of a perfectly clear semantic nature, and we discovered that we had to forget about functions and to interpret any term as a sequential algorithm from environments to values, for the following reasons:

- functions were not anymore natural objects in our universe, and many properties of the models could not even be stated if we stuck to the classical interpretations of terms as functions.

- The sequential algorithm model did not satisfy the Hindley-Longo condition ($\xi 1$) for natural semantic reasons. It would even not have satisfied the weaker (ξ) if terms had to be interpreted as functions. However (ξ) and even (η) were satisfied in our interpretation.

The sequential algorithms form a cartesian closed category with solutions to domain equations, and the same problem would actually arise in any such categories where arrows are not functions: we claim that it is then both semantically more natural and technically necessary to forget about functions and to interpret terms as arrows of the category itself, see [5] (this idea already appeared in Lambeck [7]). Then (ξ) becomes semantically natural, while ($\xi 1$) is generally not satisfied. The same will happen for term models and for interiors.

Therefore we are lead to define a model as a structure $<E,D,eval,*,[[\ ]]>>$ where D and * are as above, where E is a set called the semantic domain, where the interpretation function $[[\ ]]$ goes from terms to E and where eval is a function from $E \times ENV$ to D, ENV being as usual the set of environments over D. The eval operator is used exactly as in LISP [10] to evaluate expressions in environments: given a term M and an environment σ the value of M in σ is just $eval([[M]],\sigma)$; similarly the * operator corresponds to LISP's "apply" (again we intentionally write "operator" instead of "function": in the sequential algorithm model eval and * themselve become sequential algorithms). We require the condition (ξ) above to be satisfied, but not the condition ($\xi 1$). The equality of terms induced by the model is simply $[[M]]=[[N]]$, which may be different from $eval([[M]],\sigma)=eval([[N]],\sigma)$ for all σ, hence is different in spirit from the BHLM notion of equality.

Of course E may just be some set of functions from ENV to D as in BHLM-models - and every BHLM-model is indeed a model in our sense. In this case eval is just function application and we say that the model is environment-extensional. But there are many cases where other choices for E are preferable or even necessary: besides the algorithm model we shall see how to obtain various initial models by choosing appropriate sets of term equivalence classes for both D and E (then eval turns out to be defined from usual substitution, while * is of course the syntactic

application). Again the initial models are not BHLM-models because of the Plotkin counter-example [13]. We shall also see that interiors of BHLM-models [6], which are not themselves BHLM-models, remain perfectably reasonable models in our sense: they just loose the environment-extensionality property. We shall finally notice that the explicit introduction of E may be useful even for straight BHLM-models: in denotational semantics one often considers continuous models where ordering structures are introduced. Then E is naturally chosen as the set of continuous functions from environments to values, since the ordering relation is only interesting for continuous functions.

To summarize, the main novelties of our definition are the introduction of E and the abandon of (ξ1). The difference with the BHLM-definition is indeed quite big, since we allow many more structures to be models, claiming that they are indeed natural models, and since we abandon the Tarski notion of interpretation and satisfaction. As a matter of fact we consider that the λ operator not only defines functions but defines semantic objects which in turn define functions; for examples procedures, sequential algorithms, infinite terms etc... The price to pay is a slightly more complicated definition, which has to our knowledge no purely algebraic equivalent. Of course the BHLM-models remain very interesting ones: the condition (ξ1) implies the validity of (β), which we have to assume, and the equivalence of (η) and extensionality, which leads to simple proofs for many properies (this equivalence does not hold for our models: the sequential algorithm model is not extensional but does satisfy (η)).

The model definition is given in section 2. The BHLM-models are defined in section 3, and we show that they are the naturally environment-extensional models. The problem of adding extra-structure to models is studied in section 4. Section 5 is devoted to morphisms of models and initial models, and we conclude in section 6 by showing that BHLM-"pseudo-models" such as the interior of P^{ω} are nothing but normal models which cannot be made environment-extensional.

2. The model definition.

We feel convenient to define models in two steps: we first define an intermediate object called a semantics without introducing environments, then we define a full model. Many properties which do not involve environments are more naturally expressed as properties of semantics (typically the Wadsworth approximation continuity property, see [3,8,18]).

We assume basic knowledge of the lambda-calculus. We use letters M,N,... to denote terms, Λ to denote the set of all terms. The variables form a set V={x,y,...}. The result of the substitution of x by N in M is written M[N/x]. Contexts (expressions with holes) are written C[ ].

2.1. Semantics.

A semantics S=<E,[[ ]]> is defined by a set E called the semantic domain and a semantic mapping [[ ]]:Λ->E satisfying the following conditions for all M,N,x:

(μ) [[M]]=[[N]] => $\forall$ P. [[PM]]=[[PN]]
(ν) [[M]]=[[N]] => $\forall$ P. [[MP]]=[[NP]]
(ξ) [[M]]=[[N]] => $\forall$ x. [[λx.M]]=[[λx.N]]
(α) [[λx.M]]=[[λy.M[y/x]]], y not free in M.
(β) [[(λx.M)N]]=[[M[N/x]]]

A semantics is a η-semantics if it satisfies one of the following equivalent conditions for all M,N:

(η1) [[M]]=[[λx.Mx]] x not free in M
(η2) [[Mx]]=[[Nx]] => [[M]]=[[N]]
x not free in MN
(η3) ($\forall$ P. [[MP]]=[[NP]]) => [[M]]=[[N]]

Remark: The conjunction of (μ),(ν),(ξ) is clearly equivalent to the following condition:

(cong) [[M]]=[[N]] => $\forall$ C[ ]. [[C[M]]]=[[C[N]]]

Of course this says that [[ ]] is a congruence w.r.t. the formation laws of Λ.

2.2. Models.

A model M=<E,D,eval,*,[[ ]]> is given by a semantic set E, a value set D, an application mapping *:D×D->D, an evaluation mapping eval:E×ENV->D where ENV is the set of mappings from V to D, and a semantic mapping [[ ]]:Λ->D such that the following conditions are satisfied for all x,M,N,σ,d$\in$D:

(μ),(ν),(ξ),(α),(β) as before
(i.e. <E,[[ ]]> is a semantics)
(var) eval([[x]],σ) = σ(x)
(app) eval([[MN]],σ) = eval([[M]],σ) * eval([[N]],σ)
(lam) eval([[λx.M]],σ) * d = eval([[M]],σ[x<-d])
(free) eval([[M]],σ) = eval([[M]],σ')
if σ(x)=σ'(x) for all x free in M

As usual $\sigma[x<-d]$ denotes the environment σ' such that $\sigma'(x)=d$ and $\sigma'(y)=\sigma(y)$ for $y=x$. The notation eval([[M]],σ) being very heavy we use the natural abbreviation

$e\sigma=eval(e,\sigma)$ for $e\in E$ and $\sigma\in ENV$

and the equation above take the more classical form

(var) $[[M]]\sigma = \sigma(x)$
(app) $[[MN]]\sigma = ([[M]]\sigma)*([[N]]\sigma)$
(lam) $[[\lambda x.M]]\sigma*d = [[M]]\sigma[x<-d]$
(free) $[[M]]\sigma = [[M]]\sigma'$ if $\sigma(x)=\sigma'(x)$ for all x free in M

A model is environment extensional if $e\sigma=e'\sigma$ for all σ implies $e=e'$, is value extensional if $d*d"=d'*d"$ for all d" implies $d=d'$, and is extensional if it is both environment- and value-extensional. A model is a η-model if its semantics is a η-semantics. Clearly any extensional model is a η-model. The converse is not true: for example the sequential algorithms model of [4] is a η-model but is neither environment- nor value-extensional.

3. BHLM-models.

A BHLM-model [1,6,11] is an environment extensional model such that one of the following equivalent conditions is satisfied for all x,M,N,σ:

(ξ1) $(\forall d.\ [[M]]\sigma[x<-d]=[[N]]\sigma[x<-d]) \Rightarrow [[\lambda x.M]]\sigma=[[\lambda x.N]]\sigma$

(ξ2) $(\forall d.\ [[\lambda x.M]]\sigma*d=[[\lambda x.N]]\sigma*d) \Rightarrow [[\lambda x.M]]\sigma=[[\lambda x.N]]\sigma$

The conditions are certainly not very natural in our approach, and are better looked at from a classical logical point of view, see [6]. As we already said the BHLM-models can be defined purely algebraically, see [11], while we know of no algebraic definition of our models.

To understand the role of the conditions, let us ask the following question: given a model in our sense, can we transform it into a more classical environment-extensional model by forgetting about the extra set E? Let us call $||\ ||$ the function from ENV to D defined by $||M||(\sigma)=[[M]]\sigma$ for all σ. It is easy to see that $||\ ||$ satisfies all required equations except may be (ξ): indeed we have no way to infer $||\lambda x.N||=||\lambda x.N||$ from $||M||=||N||$ since $[[M]]\sigma=[[N]]\sigma$ for all σ does not imply $[[M]]=[[N]]$ in general. But for models which satisfy the additional requirement (ξ1), everything goes well: $||M||=||N||$ implies $[[M]]\sigma[x<-d]=[[N]]\sigma[x<-d]$ for all σ and d, hence $[[\lambda x.M]]\sigma=[[\lambda x.N]]\sigma$ for all σ, hence $||\lambda x.M||=||\lambda x.N||$. In conclusion the models (in our sense) which satisfy the BHLM condition (ξ1) can be turned in a natural way into "classical" environment extensional BHLM models.

Let us make the following remarks:

- When (ξ1) is imposed the definition of the semantic function [[ ]] is equivalent to the choice of an element in each extensionality class of D, or equivalently to the choice of a retraction from (D->D) to D: this is Meyer's algebraic definition [11]. If only (ξ) is imposed then several distinct elements may be chosen as values for abstractions (indeed quite naturally in the algorithm model of [4]).

- The validity of beta-conversion has not to be assumed for BHLM-models: it follows from the conditions (ξ1) or (ξ2), as shown by an easy proof. In our models it has to be assumed, and we indeed know of cases where the proof is not as easy: as an example, cartesian closure implies validity of beta-conversion, see [3,5,7]; the proof is quite lengthy, although it is basically an "abstract version" of the proof for BHLM-models where equality of objects is shown from cartesian closure instead of extensionality.

- A BHLM-model is a η-model if and only if it is extensional. This is not the case for our models. Again cartesian closed categories give non-extensional η-models, see [3,5]. We believe that (η) is the really interesting property in practice: extensionality clearly leads to very simple proofs for most properties, but many of these properties simply relie on the validity of (η). Typically the sequential algorithms of [4] may be used in most cases just as functions since they satisfy (η).

4. Adding extra-structure to models.

In denotational semantics it is very common to add extra structure to models, in particular orderings (or topology) and continuity. Indeed the first constructed models were "continuous models". Since we are interested in precise definitions for models, we should also be able to define precisely continuous models.

Of course one first needs to turn D into a cpo $\langle D, \leq, \perp \rangle$ requiring * to be continuous. But this is not sufficient, for one is also interested in defining an ordering on terms. The classical "functional" approach would immediately lead to order terms by $M \leq N$ iff $[[M]]\sigma \leq [[N]]\sigma$ for all σ, see [12]. We claim however that this ordering is not the only one of interest: in the models of stable functions [2,3] and sequential algorithms [4] other semantic orderings must be considered (and indeed the pointwise ordering has little properties there). Again the explicit introduction of E and eval will make things easy by allowing a maximal freedom on the choice of the ordering on E.

In our definition [3,5] a continuous model is given by a cpo $\langle E, \sqsubseteq, \bot \rangle$, a cpo $\langle D, \leq, \bot \rangle$, two continous mappings $eval: ENV \times E \to D$ and $*: D \times D \to D$ (1), and a semantic mapping $[[\ ]]: \Lambda \to E$ which satisfies the following conditions:

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

(cont) If $[[M1]], [[M2]], \ldots, [[Mn]], \ldots$ is an increasing chain of limit $[[M]]$ in E then $[[C[M1]]], [[C[M2]]], \ldots, [[C[Mn]]], \ldots$ has limit $[[C[M]]]$ in E.

The importance of the second condition is shown in [3,5].

We also define approximation continuous models in the sense of Wadsworth [18] as continuous models where the following conditions are satisfied:

(sens) $[[M]] = \bot$ if M is unsolvable

(appcont) $[[M]] = \bigsqcup \{ [[a]] \mid a$ finite approximation of $M \}$

In [3,5] we show how to obtain approximation continuous models from cartesian closed order-enriched categories. The orderings on D and E are defined from the category and have in general no relation with the pointwise ordering.

Remark: The conditions (mon), (cont), (sens), (appcont) are all set at the level of the semantics in E and do not mention environments.

5. Initial models.

As we say in the introduction, we should be able to build initial models from the terms themselves. Of course we first need to define morphisms.

Let $S = \langle E, [[\ ]] \rangle$ and $S' = \langle E', [[\ ]]' \rangle$ be two semantics. Then a morphism $h: S \to S'$ is given by a mapping $h_E: E \to E'$ such that $h_E([[M]]) = [[M]]'$ for all M. A morphism of (approximation) continuous semantics is a morphism of semantics which is also order-continuous. It is obvious to construct an initial (β)-semantics: just take for E the set of (β)-interconvertibility classes of terms and interpret any term by its class. Similarly to obtain an initial approximation continuous semantics take E as the set of

---

(1) The set ENV is best thought of as a cartesian product of infinitely many copies of D indexed by the variables. Hence the ordering on ENV should always be the component-wise ordering.

infinite Boehm-trees [1,9,18] and interpret any term by its Boehm tree.

Given two models $\mathcal{M}$ and $\mathcal{M}'$ a morphism of models is given by two mappings $h : E \to E'$ and $h : D \to D'$ such that the three following conditions are satisfied for any M,e,σ,d1,d2:

$h_E([[M]]) = [[M]]'$

$h_D(eval(e,\sigma)) = eval'(h_E(e), h_D(\sigma))$
(where $h_D(\sigma)$ is of course defined pointwise)

$h_D(d1 * d2) = h_D(d1) *' h_D(d2)$

To construct an initial model take E as before and take D as the set of interconvertibility classes of closed terms. Then an environment is just a subsitution of the variables by closed terms. Let |M| be the interconvertibility class of M. Define eval(|M|,σ) as $|M[\sigma(x1)/x1, \sigma(x2)/x2, \ldots, \sigma(xn)/xn]|$ if x1,x2,...,xn are the free variables of M. For |M|,|N| in D, define $|M| * |N| = |MN|$. Then the constructed model is initial. Similarly for constructing an approximation continuous initial model take E as the set of Boehm trees, D as the set of closed Boehm trees, define eval and * by continuity from the substitution and application on finite approximations. See [3,5] for details.

6. Satisfaction in models. About "pseudo" models.

The BHLM definition consider the environment extensional sense of satisfaction as the only one possible, and get some trouble with term models and interiors of models. We have already seen that term models become very simple non-environment extensional models, and the same happens with interiors. Intuitively the interior of a model is the part of it which is reached by interpretation of lambda-expressions.

Hindley-Longo [6] define the interior of a BHLM-model $<D,*,[[\ ]]>$ as the structure $<D^\circ,*^\circ,[[\ ]]^\circ>$ where

$$D^\circ = \{[[M]]\sigma \mid M \text{ closed}, \sigma \text{ arbitrary}\} \subset D$$

and where $*^\circ$ and $[[\ ]]^\circ$ are the restrictions of * and [[ ]] to D°. The interior of a BHLM-model is not itself a BHLM-model in general: it may fail ($\exists 1$) or even ($\exists$), the typical example being P^ω. It is called a "pseudo-model" in [6].

Given a model $<E,D,eval,*,[[\ ]]>$ we define its interior $<E^\circ,D^\circ,eval^\circ,*^\circ,[[\ ]]^\circ>$ in the following way:

- $E^\circ = \{ [[M]] \mid M \in \Lambda \} \subset E$

- $D^\circ = \{ [[M]]\sigma \mid M$ closed, σ arbitrary $\} \subset D$

- $eval^\circ$, $*^\circ$, $[[\ ]]^\circ$ are the restrictions of eval, * and [[ ]] to E and D .

It is immediate from the definition that the interior of a model is a model, and that the equalities of terms defined by a model and its interior are the same. Coming back to the interior of P^ω, the only thing which happens for us is that P^ω is not anymore environment-extensional, and cannot be turned into an environment-extensional model. The equality $[[M]]\sigma = [[M]]\sigma$ for all σ may indeed have bizarre properties without any harm: for us it is not the natural equality induced by $P^{\omega\circ}$!

All these arguments tend to show that the equality $[[M]]\sigma = [[N]]\sigma$ for all σ may be a quite unnatural object. However in all cases we were able to exhibit a semantically natural equality presenting no problem by choosing an appropriate E. It is also true that we can always study the equality $[[M]]\sigma = [[N]]\sigma$ for all σ if we want to, since it is perfectly expressible in our setting. On the converse the equalities induced by non environment-extensional models in our sense cannot be defined or studied in the Tarski framework.

To finish notice that it is very easy to weaken the model equality in our definition: given a model $<E,D,eval,*,[[\ ]]>$ one can always construct a new model $<E',D,eval',*,||\ ||>$ with the same algebra $<D,*>$ but where E' and $||\ ||$ are just taken as in the initial semantics. Equality in this new model is nothing but interconvertibility, and D becomes indeed perfectly useless! Of course the interesting models over a given algebra $<D,*>$ are those which identify as many terms as possible. In general there is no doubt in the choice of E: in syntactic models E is always related to closed terms while D is related in the same way to closed terms; in categorical models D is a solution to $D=(D\to D)$, or is a universal domain, and E is the categorical exponentiation of ENV and D, see [5,14,15] etc...

CONCLUSION

All the example given here suggest that the Tarski point of view may not be very well adapted to the lambda-calculus, since it weakens the strength of the binding operation λ . We suggested to add to the model definition a new set E and a new mapping eval. These objects are very often as natural as the usual carrier $<D,*>$. We also showed that the (ξ1) condition is often two strong. The main weakness of our definition is certainly

the absence of an algebraic formulation. But after all the lambda-calculus is not a very classical algebraic object (as opposed to combinatory logic).

BIBLIOGRAPHY

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

[2] G. Berry,"Stable Models of Typed Lambda-Calculi", Proc. 5th Int. Coll. "Automata, Languages and Programming", Udine, Italy, 1978. Springer-Verlag LNCS n. 62, pp. 72-90.

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

[4] G. Berry, P.L. Curien, "Sequential Algorithms on Concrete Data Structures", submitted to publication, nov. 1979.

[5] G.Berry, "Some Syntactic and Categorical Constructions of Lambda-Calculus Models", to appear.

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

[7] 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.

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

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

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

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

[12] R. Milner, "Models of LCF", Comp. Sci. Dept. Memo AIM-186/ CS-332, Stanford University, 1973.

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

[14] G.D. Plotkin, "T as a Universal Domain",Tech. Rep. 38, Dept. of Artificial Intelligence, Univ. of Edinburgh, 1977.

[15] 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.

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

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

[18] 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.