The Lambda-Calculus: connections to higher type Recursion Theory, Proof-Theory, Category Theory

Giuseppe Longo

On church's formal theory of functions and functionals: the Lambda-Calculus: connections to higher type Recursion Theory, Proof-Theory, Category Theory

Giuseppe Longo

On church's formal theory of functions and functionals:

The Lambda-Calculus: connections to higher type Recursion Theory, Proof-Theory, Category Theory[1]

Giuseppe Longo
Dipartimento di Informatica, Università di Pisa

Introduction

Church proposed his calculus of λ-conversion as "A set of postulates for the Foundation of Logic" (Church[1932-3]). Church's ideas and program were part of the leading Hilbert's school, at the time, whose aim was still a unified formalist approach to the foundation of Mathematics. In the following years, though, the growth of Recursion Theory, which soon became an independent mathematical discipline, led many authors to consider mostly the computational power of λ-calculus, i.e. its expresiveness in terms of the definable class of number-theoretic functions. Church himself, in view of the results of Kleene and Turing, proposed his welknown "Thesis", which is intended to characterize the computational power of finitistic systems (see Odifreddi[1986] for an updated discussion). This lecture is not concerned with the issue of "computability" as focused by Church's Thesis; however, the relevance of this claim on the expressiveness of formal systems must be acknowledged. On one side it sets a limit to feasible computations by finitistic methods, on the other it suggests that there is no other reasonable understanding of computability, besides the one established within the Hilbert - Brouwer lively debate in the twenties and early thirties. Almost everybody agrees nowadays that, as long as we do not have a counterexample, we may rely on Church's Thesis, provided that its use is not mathematically misleading. Namely, the philosophical point raised by the Thesis is surely crucial, but do we really need it when working out results ? In case a new system for general computations is proposed, it is then better to check carefully whether it still computes exactly the classically computable functions (what a discovery if it were not so !). If, instead, one is using a well established formal system, such as λ-calculus or Turing Machines, "hand waving" or "short cuts" should not be confused with reference to Church's Thesis. Hartley Rogers' book on Recursion Theory, for example, makes very little use of Church's Thesis, even if it is mentioned very frequently. Most of the time, an argument is only based on an incomplete sketch of

---

[1] A short course in λ-calculus, University of Rome I, Spring 1998 ; these notes are based on an invited lecture delivered at the Conference, **"Church's Thesis after 50 years"** Zeiss (NL), June 1986 (*On Church's Formal Theory of Functions and Functionals,* published in **Annals Pure Appl. Logic**, 40: 93-133, 1988).

an algorithm, within the intended formalism, whose complete implementation is left to the reader. This a very common and well established use of informal reasoning in Mathematics: by a frequent, but sound, reference to it, that book can summarize hundreds of results in a difficult area. Probably, from a "non human" point of view, from the point of view of a computer, say, ninety per cent of Mathematics is just "hand waving". This has nothing to do, though, with Church's methodological stand on the maximality of the expressiveness of hilbertian formalisms for computations.

This lecture will begin with classical computability and soon go further towards more general structures. Indeed, the point I want to raise here, following the original program of Church, is that lambda calculus is not just one of the many formalisms for computations, but should be looked at as the core Formal Theory of (computable) Functions and Functionals similarly as Peano Arithmetic is *the* core Formal Theory of Numbers. And *numbers* are as relevant in Mathematics and its foundation as much as functions are relevant in constructive proofs, categories, computations.

The foundational role of λ-calculus will be stressed by giving a brief survey of the main connections between λ-calculus and three major areas in Logic: higher type Recursion Theory, Category Theory, Proof Theory. These relations will be understood in a unified framework designed by the underlying mathematical structures, which give mathematical meaning to the terms of typed and type-free λ-calculi.

Church originally proposed a calculus of type-free terms: the "fregean paradise" of a type-free universe always fascinated logicians. But, when flying too high in order to comprehend as much as possible, one may have the wings burned: the first system invented by Church led to contradictions. Inconsistencies, though, frequently occur in early versions of interesting formal systems: Frege's set theory, Church's "set of postulates", Martin-Löf's type theory were all inconsistent. This was due on the breadth of the mathematical intuition required to handle the structures they had in mind, on their importance and on the interconnections whith the rest of Mathematics: the more these are, the more it may happen that the first formalization is excessively powerful.

The inconsistent version of λ-calculus was later "repaired" in two different ways, which started separeted branches of the topic. Firstly, by reducing the logical expressiveness of the untyped calculus (see Curry&al[1958,1972] or §.3). Secondly, by introducing types, according to Russell's answer to inconsistencies (Church[1940], Curry&Feys[1958]).

For a while, the challenges and the still strong expressiveness of the type-free calculus attracted more researchers than its typed counterpart. Scott's model of λ-calculus which started "denotational semantics" in Computer Science was a model of the type-free calculus; the entire book of Barendregt (Barendregt[1984]) is devoted to results in the type-free theory. Only in the last few years, mostly because of the practical success of typed functional languages and of the computer-science interest in

the ideas in Logic of Girard and Martin-Löf, have types become an even more successful area. In this paper we will restrict our attention to the "theory of types" of λ-calculus, as the relevant kernel of Type Theory.

Type Theory on one hand provided the mathematical connections to Proof Theory, on the other hand it suggested type disciplines in programming (Reynolds[1974], Milner[1978]). More precisely, types help avoiding paradoxes in Logic as well as protecting from errors in programming. Type-checking is one of the very few actually implemented partial correctness algorithms since it gives effective and significant partial correctness proofs of programs (Gordon&al[1979], Nordstrom [1981], Burstall&Lampson[1984]). It may be sound to compare type-checking to "dimension analysis" in Physics, both because types are very much like "dimensions", and because the analysis of dimensions is a commonly used tool for partial correcteness of mathematical computations in Phisics, similarly as in Programming. Besides this specific but relevant point, λ-calculus provided the core of functional languages and their type disciplines, whose practical success is due to their suitability for solving or focusing many of the concerns of actual computing. As a matter of fact, the practical relevance of λ-calculus and Type Theory for computing goes together with the variety and depth of the Mathematics involved. In particular, it largely depends on those results which relate this topic to other areas, since the richness of the theory directly embeds into the expressiveness and the facilities of actual programming, by suggesting extensions or modifications or even the design of new programming languages (further references will be given in the Conclusion).

As Mathematics is relevant when it is both beautiful and applicable, I think that the founders of λ-calculus and related systems should be happy with all of this.

As it should be clear by now, the focus of this lecture will be more on the interface of λ-calculus with other theories than on its "pure theory". By this one usually means the technical results inside the field, such as the Church-Rosser's theorem or Böhm's theorem or others based on the specific syntax of the system. This is clearly an extremely relevant area, as λ-calculus, among the various formalisms for computability suggested in the thirties, is the only one with plenty of interesting "machine dependent" results: one should consult Barendregt[1984] in order to appreciate the varity and depth of the work carried within this specific formal system. Also in this case, the point is that, more than a formalization of a "computing device" or a toy programming language, λ-calculus is and has to be viewed as the formalization of the abstract notion of function, including higher type and higher order functions; thus, the results of the formal theory often turn out to be relevant in applications or in the general understanding of functional behaviour. By this and by the connections discussed below, there should be no doubt that looking at λ-calculus just as one of the many formalisms for computing the partial recursive functions is like being interested in Peano Arithmetic only because one can represent that class of functions in it and forget its foundational relevance as

Formal Theory of Numbers.

This presentation will begin with very elementary tools and little mathematical structure: the natural numbers and the partial recursive functions on them, as the least class containing the usual base functions and closed under composition, primitive recursion and minimalization. A recursion theoretic and constructive perspective (in the sense of Intuitionistic Logic) will be stressed as structures invented for the semantics of typed and type-free λ-calculus will be looked at within a constructive frame. In particular, an inductive construction of higher type computations will be given and then studied within the very simple category of countable (and numbered) sets, in the sense of Malcev, §1; then "subcountable" and still effective sets will be investigated, the quotient subsets of numbers, §2. The formalization of these structures as categories, §3, will lead us to λ–calculus and higher order type theories, §4, and, finally, to their (constructive) models, §5. In particular, §5 presents and discusses models of second order λ-calculus over "retractions" and quotient sets in the same framework; by this some connections are pointed out.

An "organized" bibliography concludes the paper: the references are classified into four parts, according to the three interconnecting areas they roughly belong to (plus one for general references). Thus, Scott[II-1980], say, may be found in Part II, while Church[1932] is in the final group. A preliminary version of parts of this lecture was presented at the "Logic and Computer Science" Conference (Torino, October, 1986; Rendic. Sem. Matem. Torino, in print).

1. From gödel-numberings to higher types.

As promised, we begin with very simple notions and structures: natural numbers, pairings, gödel-numberings. With these elementary tools we will define higher type computations.

Let ω be the natural numbers and $<\,,\,>\,:\ \omega^2 \longleftrightarrow \omega$ any bijective (primitive) recursive coding of pairs; denote by $\lambda xy\cdot g(x,y)$ the map $<x,y> \mid\!\rightarrow g(x,y)$. As usual, (P)R are the (partial) recursive functions. A simple observation may help to understand the intuition on which definition 1.1 below is based.

***Note**: for any (acceptable) goedel-numbering $\phi : \omega \rightarrow PR$,*

(P.1.5) $\qquad \lambda xy.\phi(x)(y) \in PR$.

(P.1.5) is satisfied by any gödel-numbering, but does not characterize them: it simply tells us that the universal function for PR is still in PR . Set now $C^{(0)} = \omega$, $C^{(1)} = PR$. The idea is to use property (P.1.5) as a definition for a class $C^{(1.5)}$ of total functions in $C^{(0)} \rightarrow C^{(1)}$ and inherit this

at higher types (see 1.12 on partiality vs. totality). More precisely, we will define, for each n, a set of functions $C^{(n+1)} \subseteq C^{(n)} \to C^{(n)}$, the Hereditary Partial Effective Functionals, by inductively using a set $C^{(n.5)} \subseteq C^{(n-1)} \to C^{(n)}$.

1.1 Definition *(HPEF; Longo[I-1982]). (i) Let* $\phi : C^{(n-1)} \to C^{(n)}$. *Then*

$$\phi \in C^{(n.5)} \iff \lambda xy \cdot \phi(x)(y) \in C^{(n)}.$$

(ii) Let $\tau : C^{(n)} \to C^{(n)}$. *Then*

$$\tau \in C^{(n+1)} \iff \forall \phi \in C^{(n.5)} \ \ \tau \circ \phi \in C^{(n.5)}.$$

In order to understand definition 1.1, the reader should first check the types of the functions in $C^{(n.5)}$. However, there is a crucial hidden point in the two lines definition of the HPEF: they are well defined provided that at each higher type one can give a "coding of pairs". More exactly, for each n, in analogy to $\omega x \omega \cong \omega$ via $< , >$, an "acceptable" isomorphism

$$(2) \qquad C^{(n)} x C^{(n)} \cong C^{(n)}$$

must be found in order to set (i) in the definition (actually, a retraction, in the sense of 3.5, may suffice). Our elementary tools are sufficient to understand, quite in general, what "acceptable" means.

1.2 Definition. *Let* U *be a set and* $F \subseteq U \to U$. *Then*

$< , > : UxU \to U$ *is an* ***acceptable pairing w.r.t.*** F *if:*

1) $\exists p_1 , p_2 \in F \quad \forall x_1, x_2 \in U \quad p_i(<x_1, x_2>) = x_i \quad (p_i \text{ total}).$

2) $\forall f , g \in F \quad \lambda x.<f(x), g(x)> \in F.$

Thus we need to define at each type n an acceptable pairing w.r.t. type $n+1$; this is what we assumed at type 0, the numbers, w.r.t. type 1, the partial recursive functions. Before getting into this, there is another well known property at type 0 and 1, which one may hope to inherit at higher types as well.

Remark *(s-m-n theorem): for any gödel-numbering* $\phi_1 : \omega \to C^{(1)}$, *one has*

$$(3) \qquad \forall \phi \in C^{(1.5)} \ \ \exists f \in C^{(1)} \ \ \phi = \phi_1 \circ f.$$

This generalizes at higher types as follows.

$$(4) \qquad \forall n > 0 \quad \exists \phi_n \in C^{(n.5)} \quad \forall \phi \in C^{(n.5)} \quad \exists f_n \in C^{(n)} \quad \phi = \phi_n \circ f_n.$$

Observe that, if (4) holds, then each $C^{(n)}$ is countable, as ϕ_n must be surjective. Thus, in order

to prove (2) and (4) one may try to work within a category of countable objects and "effective" morphisms. A good candidate for this could be Malcev's category **EN** of numbered sets (enumerations), the simplest generalization of Recursion Theory to an abstract setting.

1.3 Definition. *The Category **EN** has as **objects** pairs* $\underline{A}=(A,e_A)$, *where* A *is a set and* $e_A:\omega\to A$ *is a surjective map.* ***Morphisms*** *are defined by*

$$f\in EN[\underline{A},\underline{B}] \quad iff \quad \exists f'\in R \quad f\circ e_A = e_B\circ f'.$$

Clearly, **EN** is a category and it has several interesting (closure) properties. For example, one can look at the product of two numbered sets as a numbered set itself: just enumerate the product by using the given bijective pairing of numbers. However, since we are interested in higher type computations, as given by the HPEF, we need also other kinds of higher type objects, such as exponentiations, in the category. Unfortunately, there is no general way to enumerate the set of morphisms of two arbitrary objects in **EN**, if one wants that products and the representation of mophisms spaces commute in the sense of Cartesian Closed Categories, i.e. $EN[\underline{A}\times\underline{B}, \underline{C}] \cong EN[\underline{A}, \underline{C}^{\underline{B}}]$. Consider, say, $\underline{\omega} = (\omega,id)$ as a (trivially) numbered set. Then, $EN[\underline{\omega}, \underline{\omega}] = R$. This is surely a countable set, but for no enumeration e_R and $\underline{R} =(R,e_R)$, one has $EN[\underline{\omega}\times\underline{\omega}, \underline{\omega}] \cong EN[\underline{\omega}, \underline{R}]$, as e_R would be an effective enumeration of R, which is impossible; or, if preferred, the "uncurrying" u of e_R, $u(n,m) = e_R(n)(m)$, would be a computable universal function for R.

One may think of two main ways to preserve the effective flavour of the category **EN** and obtain the required closure properties which guaranty the existence of higher type objects in the category: the first is to look inside **EN**, the other is to extend **EN** in order to to get Cartesian Closure without loosing the simplicity of this category.

Scott and Ershov suggested a way to stay *inside* **EN**. As we want the HPEF to satisfy property (4) , this is also what we are looking for. Observe that Scott's motivation was the construction of mathematical structures where one could interpret languages for computer programming; as programs compute (possibly higher type) functions, the idea was related to generalized computability as well. Ershov, partly following earlier work of Scott, wanted to relate in a unified mathematical framework the ideas of Kleene and Kreisel for Higher Type Recursion Theory, a topic in turn motivated by the semantics of Intuitionistic Logic (see Scott[I-1970,1976,1982], Ershov[I-1976]; surveys may be found in Smyth[I-1977], Giannini&Longo[I-1984] and many others).

The interesting point is that both authors used, in some essential way, topological properties in their work. That is, some numbered sets are picked up according to some extra structure they can be given, based on a topological notion of convergence (approximation). The idea is, given a poset $(X,\leq)$, to generalize first some key properties of finite sets to a subset X_o of X and use X_o in

order to approximate arbitrary elements of X. More precisely,

(Compact) $\mathbf{x \in X_0}$ iff, for all directed subset D of X, $x \leq \sup D$ implies $\exists d \in D\ x \leq d$.

Clearly, (Compact) is a "compactness" property for the elements in X_0 and it generalizes a simple fact which characterizes the finite sets in any powerset, partially ordered by set inclusion. By this, the elements of X_0 are sometimes called "finite". We prefer to refer to them as "compact" or "noetherian", as pointed out in Remark 1.8 below.

Then $(X,X_0,\leq)$ is **algebraic** iff, for all $x \in X$, $\wedge\mathbf{x} = \{x_0 \in X_0 \,/\, x_0 \leq x\}$ is directed and $x = \sup \wedge x$. A directed complete poset, cpo, $(X,\leq)$ is **bounded complete** iff each bounded subset of X has a least upper bound. A poset is a (Scott) **domain** iff it is a bounded complete algebraic cpo. Define finally $\uparrow x = \{y \in X \,/\, x \leq y\}$.

1.4 Definition. *Let $\underline{X} = (X,X_0,\leq)$ be a domain. The **Scott topology** τ_S on $\underline{X}$ is given by the base $\{\uparrow x_0 \,/\, x_0 \in X_0\} \cup \{\emptyset\}$.*

1.4 is a good definition, as pairs of elements of X_0, bounded by $x \in X$, have an upper bound in X_0, smaller than x.

The next step, in order to get into effectiveness, is to assume countability of X_0 (ω-algebraicity) plus the decidability properties you would expect from "finite" sets (of numbers). That is, given an injective enumeration $e_0 : \omega \to X_0$ of X_0, a domain $(X,X_0,e_0,\leq)$ is **effectively given** if it is decidable whether two elements of X_0 have an upper bound and their least upper bound within X_0 can be uniformely effectively found. By this, $e_0(n) \leq e_0(m)$ is decidable in n, m.

The point now is to obtain sufficiently rich, but countable and, possibly, enumerated posets.

1.5 Definition. *$\underline{X_C} = (X_C,X_0,e_0,\leq)$ is a **Constructive Domain** iff there exists an effectively given domain $\underline{X} = (X,X_0,e_0,\leq)$ such that $X_C \subseteq X$ and, for any directed subset D of X_0, one has:*

$$x \in X_C \text{ iff } \{n \,/\, e_0(n) \leq x\} \text{ is a recursively enumerable set}$$

Clearly, the effectively given domain in 1.5, $\underline{X} = (X,X_0,e_0,\leq)$, is uniquely determined (up to isomorphisms) by $\underline{X_C}$, and conversely, since a domain is the completion over all directed sets of its base set X_0, while a constructive domain $\underline{X_C} = (X_C,X_0,e_0,\leq)$ is the completion only over all r.e. directed subsets of X_0 (use for this the decidability of $e_0(n) \leq e_0(m)$). By the latter property, one can easily and effectively enumerate the entire poset X_C, i.e. one may define canonically a surjective map $e: \omega \to X_C$ by using the properties of e_0 and an enumeration of the r.e. sets (see Weyrauch[I-1981] and Giannini&Longo[I-1984], for details). Clearly, e doesn't need to be injective. Take for

example the constructive domain $(PR,PR_O,\varphi_O,\leq)$ of the partial recursive functions: in this case the compact elements, PR_O, are given by the functions with a finite graph, enumerated in some canonical way, φ_O, say. Then $\varphi:\omega\to PR$ is just an (acceptable) gödel-numbering. The same applies to the domain of r.e. sets. Observe then that any (constructive) domain $\underline{X}$ has a least element $\perp_X$, say, as $\varnothing$ is directed. By this, $\underline{w} = (w, id)$, with the identical enumeration, is not a cconstructive domain (see remark 2.3).

From now on, we consider each constructive domain $\underline{X}_C$ also a numbered set (X_C, e), where e is derived from e_O as mentioned. However, any such numbered set $\underline{X}_C$ happens to have some "structure", the partial order and the topology, which nicely relate by

(Poset) $\quad \forall x,y\in X_C \ (x\leq y \ \Leftrightarrow \ (\forall A\in \tau_S \ x\in A \ \Rightarrow \ y\in A))$.

Moreover, these topological tools define the usual set of continuous functions; they turn out to be exactly the monotone functions which commute w.r.t. to sups of directed sets, when they exist. As usual, the continuous functions can be partially ordered pontwise. Some continuous functions are more basic than others: consider, say, for $x_O\in X_O$ and $y_O\in Y_O$,

$$\underline{\text{step}}x_Oy_O(x) = \underline{\text{if}} \ x_O\leq x \ \underline{\text{then}} \ y_O \ \underline{\text{else}} \ \perp_Y .$$

By taking the sups of finite collections of compatible "step" functions, one obtains a countable collection of continuous functions, $Cont(X,Y)_O$ say. An enumeration e' of $Cont(X,Y)_O$ can be easily (and canonically) given by using $e_O:\omega\to X_O$ and $e_1:\omega\to Y_O$. Define then

$$\mathbf{Cont(X,Y)_C} = \{ f\in Cont(X,Y) \ / \ \{n \ / \ e'(n) \leq f \} \ \text{is r.e.}\}.$$

It is not difficult then to check the following.

1.6 Lemma. *If $\underline{X}_C = (X_C,X_O ,e_O,\leq)$ and $\underline{Y}_C = (Y_C,Y_O ,e_1,\leq)$ are constructive domains, then $(Cont(X,Y)_C , Cont(X,Y)_O, e', \leq)$ is a Constructive Domain. Moreover, if $f\in Cont(X,Y)_C$, then $\forall x\in X_C \ \ f(x)\in Y_C$.*

By 1.6, $Cont(X_C,Y_C)_C$ are exacly the restrictions to X_C of the functions in $Cont(X,Y)_C$, when X_C and Y_C are given the induced topology.

1.7 Theorem. *The category **CD** of Constructive Domains, where morphisms are defined by **CD[A,B]** = $Cont[A,B]_C$, is Cartesian Closed.*

(*Notation*: $\underline{\mathbf{B}^{\mathbf{A}}}$ is the exponent object, which internally represents CD[A,B], in accordance with the categorical use.)

The lemma and the theorem essentially prove that the "compactness" and "effectiveness" properties of X_O, in a fixed constructive domain $\underline{X}_C$, are inherited at higher types: this is obvious

for cartesian products (and implicit in 1.7), hints were given for the construction of the compact elements in function spaces. As already mentioned, "compactness", as given in (4), characterizes the finite elements in any powerset; however, another characterizing property of finite sets (or of functions with finite domain) is lost at higher types:

1.8 Remark. *In any powerset (or subposet of it), exactly the finite sets have finitely many subsets. This is not thrue if one considers the compact elements of an arbitrary effective poset. Indeed, take a (constructive) domain* $\underline{X}^1$ *with an infinite collection of pairwise incompatible elements or with an infinite ascending chain, and set* $\underline{X}^{n+1} = CD[\underline{X}^n,\underline{X}^n]$*; then, for any* $n>1$ *and* $x_o \in (X^n)_o$, $\hat{}x_0$ *is infinite. This can be easely understood, by observing that any step function* $\underline{step}ab$ *is antimonotone in* a .

Recently, Girard[II-1985], following Berry[II-1979], suggested to consider a subcategory of Scott's domains, the qualitative domains, made out of subsets of powersets and where only some continuous functions are morphisms: the "stable" functions, which preserve also finite intersections of compatible elements As an elegant consequence, one then has that in any type each compact element x *has a finite* $\hat{}x$. *Stable functions originated in Berry[II-1979] and have some deep connections with Girard's ideas in Proof Theory, as his dilators similarly preserve pullbacks and direct limits (see Girard[1986]); moreover, an insight is also given into sequentiality, as stable functions are tidily related to sequentiality (see Berry&Curien[1982]). This seems to suggest an alternative approach to higher type recursion, still to be explored, since the crucial 1.10 below doesn't hold any more (see Asperti[I-1987] for some preliminary work).*

As pointed out, every object in **CD** is a numbered set; thus **CD** is a subcategory of Malcev's **EN**. The point is to understand how the definition of morphism in **EN**, which is so tidely effective and is only based on the recursive functions, and the morphisms in **CD**, which are particular continuous functions, relate.

In should be clear by now that the role of continuity comes in quite smoothly from considering a functional f to be computable when it is continuous, i.e. it computes with compact approximations (which are finitely coded) of its (possibly infinite) input (i.e. $f(x) = \sup\{f(\hat{}x)\}$) and f itself is the r.e. limit of its compact approximants in its own type.

The next lemma clarifies how Geometry and Recursion Theory relate over constructive domains.

Given a numbered set (X,e) , observe first that the set $\{ A \subseteq X / e^{-1}(A)$ is r.e.$\}$ satisfies the requiremennts for a topological base. Call the induced topology the **Malcev-Ershov** topology.

1.9 Lemma *(Generalized Rice-Shapiro Theorem). Let* $\underline{X} = (X,X_o,e_o,\leq)$ *be in* ***CD*** . *Then the Scott topology on* $(X,\leq)$ *coincides with the Malcev-Ershov topology on* (X,e).

Note that the Malcev-Ershov topology comes from Recursion Theory and has little to do with approximation and orders. *(Poset)* above, say, defines a non trivial partial order iff the topology is T_0 , which does not need to hold outside **CD**. Moreover, each morphism in **EN** turns out to be continuous, w.r.t. this topology, just by an obvious recursion theoretic argument (the inverse image of an r.e. set by a recursive function is r.e.). A proof of 1.9 may be found in Giannini& Longo[I-1984] or in Rosolini[II-1986]. In the latter, this discussion is carried on in a sound category-theoretic setting, by considering categorical models of Intuitionistic Logic. This evidenciates also the connections to the constructive aspects of the metamathematical frame: for example, the proof of 1.9 is intuitionistically acceptable provided that Markov principle is considered (see Beeson[1980], Hyland[II-1982], McCarty[I-1984]). The significance of 1.9 is that the Scott topology, which is apparently added as extra structure, is indeed inherited by suitable enumerations of the objects of **CD**, as numbered sets.

From the lemma one may easily derive a precise connection between continuity and effectiveness for functions. Namely, that the morphisms between (X,e) and (Y,e') as constructive domains, which are continuous maps, coincide with the morphisms between (X,e) and (Y,e') as numbered sets, which are recursive functions over indices:

1.10 Theorem *(Generalized Myhill-Shepherdson)**.** The CCC **CD** is a full subCategory of **EN** .*
(Proof of the key point: Let $\underline{X}$, $\underline{Y} \in Ob_{CD} \subseteq Ob_{EN}$ and $f \in EN[\underline{X},\underline{Y}]$; then f is continuous by the lemma and the subsequent observation).

Again, the relevance of these facts is based on the naturalness of the partial order on (constructive) domains and, thus, of the induced topology (see *(Poset)* above): these are not defined "ad hoc" for the purposes of 1.9 or 1.0, but come out of obvious generalizations of set inclusion. !.9 and 1.10 are the mathematical reason for the common motto in denotational semantics of programming languages: "the computable functions and functionals are continuous".

CD has further interesting properties, which do not hold in **EN**: for example, **CD** is closed under inverse limits of projections and limits are also preserved by the product and exponentiation functors. By this, say, one may construct countable models of the type-free λ-calculus, as there are objects in **CD** which satisfy equations such as $\underline{X} \cong \underline{X^X}$ (see Smyth[I-1977], Kanda[I-1979], Smyth& Plotkin[II-1982], Giannini&Longo[I-1984]).

We finally are in the position to understand the properties of the HPEF and, in particular, to check the crucial facts conjectured in (2) and (4) above (listed under (2) and (4) also in 1.11 below). This is done by using the full and faithful embedding of **CD** in **EN**.

Let $P^{(1)}$ be the (effectively given) domain of the partial number theoretic functions and set $P^{(n+1)}=Cont(P^{(n)},P^{(n)})$. $\{P^{(n)}_C / n\in\omega\}$ are exactly the Ershov-Scott higher type partial computable functionals, i.e. the (integer) type structure in **CD** generated by PR, the partial recursive functions, as P_C=PR.

1.11 Theorem. *For all n>1,*

1) $C^{(n)}=P^{(n)}_C$

2) $C^{(n)}xC^{(n)} \cong C^{(n)}$ *in* ***CD*** .

3) $C^{(n.5)} = Cont(P^{(n)}, P^{(n+1)})_C$

4) $\exists\varphi_n\in C^{(n.5)}\ \forall\varphi\in C^{(n.5)}\ \exists f_n\in C^{(n)}\ \ \varphi = \varphi_n \circ f_n$.

(The proof goes by combined induction and by an "essential" use of 1.10 and may be found in Longo&Moggi[I-1984] or, in a more categorical style, in Rosolini[II-1986])

From 1.11(2) it immediately follows that the isomorphism $C^{(n)}xC^{(n)} \cong C^{(n)}$ is acceptable w.r.t $C^{(n+1)}$, in the sense of 1.2.

Theorem 1.11 should convince the reader of two facts. First, by the HPEF one can get at higher types by very elementary tools, with no apparent use, in the definition, of categories, continuity or whatsoever. However, topological and related notions seem to be essential for proving even the countability of the type structure so easily defined (point 4 above). In a sense, one may say that the HPEF are defined in a purely combinatorial or algebraic way, while analytic tools come in the proofs (see Longo[1984]). Second, Scott and Ershov definition of higher type computable functions is indeed a very natural one as it may be recovered by 1.1, i.e. just starting with PR and an acceptable gödel-numbering of it. The way these *partial* functionals relate to the *total* "countable and continuous" functionals of Kleene and Kreisel is established in Ershov[I-1976] and Longo& Moggi[I-1984].

1.12 Remark *(Why are the HPEF partial ?). Indeed, definition 1.1 begins with PR which is a set of partial maps. However the functions in* $C^{(1.5)}$ *are total, as defined, and thus the same are the maps in* $C^{(2)}$, $C^{(2.5)}$ *etc. Why do we soundly call them partial ? Of course, the same can be said of the morphisms in the frame category* ***CD****: they are total morphisms as well, though we all agree that they define partial higher type computations. At an informal level the answer is very simple: beginning with the domain of type 1, i.e. PR, each higher type has a least element, the empty set (or the function with the empty graph). This is usually considered as the interpretation of "divergence" in the corresponding type. More rigourously, this may be understood in category theoretic terms. It is not difficult to define categories of partial morphisms, in the proper sense of "possibly not defined*

maps" (see Rosolini[II-1986], Robinson&Rosolini[II-1987] and Moggi[1987] for recent approaches, surveys and references, but the reader should just trust his intuition for a definition). In one such a category $\boldsymbol{C}$*, one may easily define a "lifting functor"* $\_^{\perp}$ *by the natural isomorphism* $C_T[A,B^{\perp}] \cong C[A,B]$ *, where* C_T *are the total morphisms. This simply says that* $\perp$ *"interpret" the divergence, as the partial maps with target* B *are the same as the total ones with target* $B^{\perp}$ *. In Longo& Moggi[II-1984a] a new categorical notion is derived from this; namely, in a partial category, an object* B *is* ***complete*** *iff* $B<B^{\perp}$ *, i.e.* B *is a retract of its lifting (see also Asperti&Longo[II-1986] for an updated presentation and some applications). The point is that the complete objects are exactly those objects* B *such that each partial morphism with target* B *can be uniquely extended to a total morphism. It is now easy to check that (constructive) domains are complete in the intended partial categories, with the obvious definition of partial continuous morphism (the domain of convergence must be Scott open). By this, the partial morphisms in those categories may be soundly identified with the total ones.*

2. From countable to subcountable sets

The basic idea in the definition of the subCategory **CD** of **EN** was the choice of some structured objects in **EN** which could form a category with enough closure properties as for the purposes of higher type computations. The suggested structures were topological ones.

As pointed out in §1, there are many countable sets which cannot be soundly enumerated in the category **EN**; as an example we mentioned the set R of the total recursive functions. In a sense though, one may say that R, even if not (effectively) countable, is "subcountable", i.e. it can be (effectively) enumerated by a subset of ω. The second idea one may think of, then, is to *enlarge* the category **EN** as to include this sorts of exponents (function spaces), i.e. sets (effectively) enumerated by subsets of ω.

Observe first that any numbered set defines an equivalence relation on ω (and, thus, a quotient of ω) and, conversely, any equivalence relation on ω uniquely determines a numbered set: just set to be equivalent any two numbers which code the same element and viceversa. Indeed, from now on, any numbered set $\underline{A}=(A,e_A)$ will be equivalently referred to as a quotient $\underline{A}$ of ω, where

$$n\underline{A}m \quad \text{iff} \quad e_A(n) = e_A(m).$$

Clearly, given numbered sets $\underline{A}$ and $\underline{B}$, not any $f' \in R$ induces an $f \in \mathbf{EN}[\underline{A},\underline{B}]$, as f' must preserve $\underline{A}$-equivalences: that is one must have $n\underline{A}m \Rightarrow f'(n)\underline{B}f'(m)$. This suggests a way to introduce higher type objects and thus to define a cartesian closed extension of **EN**.

Let $\{\varphi_i\}_{i\in\omega}$ be an acceptable goedel-numbering of PR. Define then

(Quot.) $p\underline{B}^{\underline{A}}q$ iff $n\underline{A}m \Rightarrow \varphi_p(n)\underline{B}\varphi_q(m)$.

$\underline{A}^{\underline{B}}$ is a **partial equivalence relation** on ω, as it is defined on a *subset* of ω. Indeed, for $\underline{A}$, $\underline{B}$ non trivial, $dom(\underline{B}^{\underline{A}}) = \{p / p\underline{B}^{\underline{A}}p\} \neq \omega$, and a partial numbering (i.e. a partial surjective map) $\pi_{AB}: dom(\underline{B}^{\underline{A}}) \to \underline{B}^{\underline{A}}$ is given by $\pi_{AB}(n) = \{m/ n\underline{B}^{\underline{A}}m\}$. Of course, $dom(\pi_A) = dom(\underline{A})$.

In general, given a set C , each partial surjective $\pi: \omega \to C$ (or **partial numbering**) uniquely defines a partial equivalence relation (and conversely). It may be fair to call these new objects "modest", as suggested by Scott, as they are just and simply (quotient) subsets of ω .

2.1 Definition. *The category **M** of partial equivalence relations on ω (the **modest sets**) has as objects the subsets of ω modulo an equivalence relation. Given objects $\underline{A} = (A,\pi_A)$ and $\underline{B} = (B,\pi_B)$, where π_A , π_B are partial numberings, the morphisms are defined by*

$$f \in \mathbf{M}[\underline{A},\underline{B}] \quad iff \quad \exists f' \in PR \quad f \circ \pi_A = \pi_B \circ f' .$$

As $f \in \mathbf{M}[\underline{A},\underline{B}]$ is total, one has that $dom(f') \supseteq dom(\underline{A})$.

Note that the representative $\underline{B}^{\underline{A}}$ of $\mathbf{M}[\underline{A},\underline{B}]$ is partially enumerated by the quotient subset of ω determined by the partial relation $\underline{B}^{\underline{A}}$ (see (Quot.) above). That is,

$$\pi_{AB}(i) = f \quad iff \quad f \circ \pi_A = \pi_B \circ \varphi_i .$$

By this, one obtains, for example, a partial, but effective, enumeration of $R = \underline{\omega^{\omega}}$ by a surjective map defined on a subset of ω .

2.2 Theorem. ***M*** *is a CCC and includes **EN** as a full subCartesian Category.*

Indeed, one may prove, by using also 1.10 , that the full and faithful embedding from **CD** into **EN** and, then, into **M** is such as to preserve products and exponentiations from **CD** into **M**.

M is a natural generalization of the Hereditary Extensional Operations (HEO) in Troelstra[1973], where they are introduced for the purposes of Intuitionistic Logic and its Proof Theory (see also Girard[III-1972] and §.5 below).

In Computer Science, **M** is also known as the quotient set semantics of types over ω, following the ideas in Scott[I-1976] on λ-models (see also Hindley[1983], Coppo[1984], Longo& Moggi[II-1986] for details and further work on arbitrary (partial) combinatory algebras).

2.3 Remark. *Observe that PR and R are enumerated in **M** in entirely different ways. As mentioned in several places, R does not live in **CD** , while PR cannot be enumerated as an object of **M** by similar tricks as hinted above for R : the maps in PR are partial, while we are looking at categories with total maps as morphisms, as usual (see 2.1). The idea is to extend ω to $\omega^{\perp}$ in **CD***

by adding a least, undefined, element $\perp$ and enumerate $\omega^{\perp}$ following the procedure suggested for constructive domains, based on the enumeration of the compact elements (see after 1.5); $\perp$, say, turns out to be coded by the complement of an r.e. non recursive set (see Spreen[I-1984] and Asperti&Longo[II-1986] for details). By this the enumeration of PR satisfies the s-m-n theorem, a weakly universal properties, whose generalization was relevant for the definition of the HPEF (see the discussion between 1.2 and 1.3). Note that this way of enumerating objects which gives the classical gödel-numberings in case of PR, is also required, quite generally, for the sake of 1.9 and also gives the functorial embedding of ***CD*** *into* ***M*** *which preserves products and, hence, exponentials.*

3. The formal theory of functions

In §.1 and 2 we have been looking at mathematical generalizations to higher types of the notion of function on a ground type of data. This was done on countable sets, because of the foundational motivations for constructive aspects of Logic and for Computer Science we assumed. Moreover, that work has some mathematical relevance in view of the new structures and the general frame proposed. It may be then the case to formalize in a theory of functions the key properties we dealt with.

Functions may be based on three main notions: application, abstraction and tupling (in order to handle several arguments functions). That is,

(App) - apply a function f to an argument a : f(a)

(Abs) - abstract a function from an expression f(x) , possibly depending on a variable x : $\lambda x.f(x)$

(Pair) - construct a pair from elements a, b : (a,b) .

These notions need now to be formalized and typed. Let then **At** be a set of atomic type symbols and let **Type** be the least set containing **At** and such that:

$$\sigma, \tau \in \mathbf{Type} \;\Rightarrow\; \sigma\to\tau,\ \sigma\times\tau \in \mathbf{Type}\ .$$

3.1 Definition *(Typed λ-terms).* x^{σ} *(variable of type σ)*

$(M^{\sigma\to\tau}N^{\sigma})^{\tau}$

$(\lambda x^{\sigma}.M^{\tau})^{\sigma\to\tau}$

$(M^{\sigma}, N^{\tau})^{\sigma\times\tau}$

$fst(M^{\sigma\times\tau})^{\sigma}$

$snd(M^{\sigma\times\tau})^{\tau}$.

3.2 Definition *(Typed λ-calculus with surjective pairing). The axioms of* $\boldsymbol{\lambda\beta\eta SP_t}$**:**

(β) $(\lambda x^{\sigma}.M^{\tau})N^{\sigma} = [N^{\sigma}/x^{\sigma}]M^{\tau}$

(η) $\lambda x^{\sigma}.M^{\tau}x^{\sigma} = M^{\tau}$

(fst) $fst(M^{\sigma},N^{\tau}) = M^{\sigma}$

(snd) $snd(M^{\sigma}, N^{\tau}) = N^{\tau}$

(SP) $(fst(M^{\sigma\times\tau}), snd(M^{\sigma\times\tau})) = M^{\sigma\times\tau}$.

The inference rules for $\lambda\beta\eta SP_t$ are exactly what is needed to turn "=" into a congruence relation.

The next theorem sets some mathematical base to the claim concerning the relevance of $\lambda\beta\eta SP_t$ as a theory of functions.

Category Theory is often considered the alternative *functional* foundation for Mathematics, w.r.t. Set Theory, as functions are first described and sets, if needed, are a derived concept. In particular, the theory of Cartesian Closed Categories, which contain function spaces, seems a sound setting for functionality. Theorem 3.3 proves that we may view types as objects, in the sense of Categories.

3.3 Theorem *(Types-as-objects). The models of* $\lambda\beta\eta SP_t$ *are exactly the (concrete) CCC's.*

This result may be found in Lambek[II-1980], Scott[II-1980] (see also Lambek&Scott[II-1986], Curien[II-1986], Breazu-Tannen&Meyer[II-1985]).

Thus, we started with particular structures for higher type functions, then we formalized functionality and got to a formal Theory of (typed) Functions, $\lambda\beta\eta SP_t$. Similarly, mathematicians had first in mind particular structures (rotations of a cube, relative numbers....) and then invented Group Theory. Of course, Group Theory has many more models than those; in the same way, there are many more CCC's than **CD** or **M**. However, these specific models have some further relations to the theory, as they are defined by using the class of (partial) recursive number-theoretic functions, which are exactly the formally definable functions in the *type-free* λ-calculus, $\boldsymbol{\lambda\beta\eta}$.

Indeed, when first formalizing the intuitive notion of computation and suggesting a language for the foundation of Mathematics, Church did not consider types. That is, $\lambda\beta\eta SP$ is defined just by erasing type constraints in term formation rules ($\lambda\beta\eta$ in Church[1941] does not have (SP) either). The ambition was to live in a type-free Fregean paradise and preserve as much expressiveness of Mathematics as possible.

Shoenfinkel and Curry had an other idea on how to describe functions (and Mathematics), in a typeless way:

3.4 Definition *(Combinatory Logic, CL). Terms of* ***CL*** *are variables x, y...*

S, K

(MN) .

The axioms are:

(KM)N = M

((SP)Q)R = (PR)(QR) .

When adding

(ext) $Mx = Nx \Rightarrow M = N$

to the obvious inference rules for "=", CL(ext) turns out to be equivalent to $\lambda\beta\eta$ (see Hindley&Seldin[1986]). For the key step write

$[x]x = (SK)K$

$[x]y = Ky$ for $y \neq x$

$[x](MN) = S([x]M)([x]N)$.

Then $[x]M$ translates $\lambda x.M$ and conversely (note that $[x]M$ does not contain x, or, equivalently, x is not free in $\lambda x.M$). But now comes the rub. In Logic (and in Computer Science) types help to avoid paradoxes or inconsistencies and Church original system was proved inconsistent by Rosser. Rosser's remark was concerned with the handling of implication in λ-calculus; we may understand it in terms of Curry's paradoxical combinator Y, the fixed point operator, and formal negation. As xx is well formed in $\lambda\beta\eta$ and CL, so is Y, where

$$Y \equiv \lambda y.(\lambda x.y(xx))(\lambda x.y(xx))$$

is such that $YM = M(YM)$. Thus the original system of Church, which included a term representing negation, led to a paradox.

Once this excess in expressiveness was eliminated, the consistency of $\lambda\beta\eta$ could be proved by purely syntactic tools (Church&Rosser[1936]). However, even though these calculi were designed in order to formalize meaningful notions from Mathematics, formalization and syntax went beyond Mathematics; that is, no mathematical model was known till Scott's construction (Scott[II-1972]).

Let's understand it in the following way.

Clearly, any model of CL, the weakest theory (see below), is an applicative structure $(A,\cdot)$, as it must intepret formal application of type-free terms. Indeed, one may use any model $(A,\cdot)$ of CL, instead of Kleene's $(\omega,\cdot)$, and perform the same construction of the CCC **M** in 2.1 (see Longo& Moggi[II-1986]). Write $\mathbf{M}_A$ for this relativized construction. Observe finally that in a CCC **D** any $f \in D[A,A^A]$ turns A into $(A,\cdot)$ by setting, informally, $a\cdot b = f(a)(b)$.

3.5 Definition. *Given objects A and B in a category* ***C****, a* ***retraction pair*** *from B into A is a pair (i,j) such that* $i \in C[B,A]$, $j \in C[A,B]$ *and* $j \circ i = id_B$ *(we write* $B < A$ *via (i,j)). A morphism*

*$p \in C[A,B]$ is **principal** if $\forall f \in C[A,B]\ \exists g \in C[A,A]\ f = p \circ g$. Isomorphisms "$\cong$" are well known.*

***3.6 Theorem**. Let **C** be a CCC and A an object of **C**. Then*

(1) $A^A \cong A \Rightarrow$ A is a model of $\lambda\beta\eta$

(2) $A^A < A \Rightarrow$ A is a model of $\lambda\beta$

(3) $\exists p \in C[A,A^A]$ principal and $A \times A < A \Rightarrow$ A is a model of CL .

Conversely,

(1) A is a model of $\lambda\beta\eta \Rightarrow A^A \cong A$ in $\mathbf{M}_A$

(2) A is a model of $\lambda\beta \Rightarrow A^A < A$ in $\mathbf{M}_A$

(3) A is a model of CL $\Rightarrow \exists p \in M_A[A,A^A]$ principal and $A \times A < A$ in $\mathbf{M}_A$.

It is easy to observe directly that $A^A < A$ implies $\exists p \in C[A,A^A]$ principal and $A \times A < A$; the converse does not hold, as CL is a weaker system than $\lambda\beta$.

The core of (1) and (2) in 3.6 is in Berry[II-1979], Koymans[II-1982] and Obtulowicz& Wiweger[II-1982] (as usual, though, the main reference for the type-free calculi is Barendregt[1984]). In Scott[II-1972] a CCC was given, essentially a subcategory of Scott's domains, and an object A such that $A \cong A^A$.

(3) may be found in Longo&Moggi[II-1986], where principal morphisms were defined.

***3.7 Remark** (The HPEF and Category Theory). (i) Principal morphisms are not exactly "universal arrows", in the sense of Category Theory, since there is no request that g in definition 3.5 is unique. The reader may easily observe that principal morphisms are the category theoretic generalization of property (4) in 1.11, the key property on the "hereditary gödel-numbering" of type $C^{(n+1)}$ by type $C^{(n)}$ in the HPEF. This is where the notion originated.*

(ii) There is some more Category Theory hidden in the HPEF: the intermediate type $C^{(1.5)}$ contains an implicit "currying-uncurrying" operation, the same which relates the universal function to the gödel-numbering of PR . The extension to $C^{(n.5)}$ in 1.2 gives a meaningful type-structure, once that the isomorphism $C^{(n)} \times C^{(n)} \cong C^{(n)}$ in (3) of 1.11 proves that the "currying-uncurrying" trick can be inherited at higher types. And the currying operation is the core of Cartesian Closure for categories. On the grounds of this observation Rosolini[II-1986] suggested an alternative very elegant proof of 1.11, where the hard, but elementary, work on enumerations and induction is distilled in a nice unified frame of topoi and Intuitionistic Logic (see the "Effective Topos" in §.5).

Surprisingly enough there is no known example of mathematical model of CL which is not a

model of $\lambda\beta$ (i.e. except for the term model)! In other words, any known structure satisfying the conditions in 3.6(3) does happen to yield a retraction, in the sense of 3.6(2). Of course, any gödel-numbering of PR is just a principal morphism which is not a retraction within **CD** (nor **EN**), as there is no computable inverse to a gödel-numbering. However, if one wants a model of CL , a total applicative structure is needed and, when extending ω to its lifting $\omega^{\perp}$(see 1.12), $\omega^{\perp}\times\omega^{\perp}< \omega^{\perp}$ is clearly lost, in **CD**. The construction of one such a model would shed some structural light on the minimal conditions for functional completeness, besides the category theoretic notions of principal morphisms and "A×A<A" .

Note that 3.6 characterizes all the three basic type-free theories of functionality: CL, $\lambda\beta$, $\lambda\beta\eta$. 3.3, instead, only characterizes typed $\lambda\beta\eta$ (plus SP). Indeed, typed $\lambda\beta$ has some categorical meaning: **weak** CCC's, were the usual natural isomorphism $C[AxB,C] \cong C[A,C^B]$ is only a natural retraction $C[AxB,C] < C[A,C^B]$, characterize typed $\lambda\beta$ (see Hayashi[1986], Martini[1986]).

In conclusion, Categories fit nicely with effective type-structures and λ-calculus, both in the typed and untyped case. Observe also that one may look at type-free models as at special case of typed ones: namely, those CCC's which have a "reflexive" object, i.e. an object A such that $A \cong A^A$ or the weaker properties in 3.6(2-3) hold. Conversely, from any type-free structure (A,.) one may recover the CCC $\mathbf{M}_A$..

This correspondence has a nice syntactic counterpart: type-free terms may be given a type, if any. More precisely, there is an algorithm which decides whether a type-free term possesses a type and, if so, assigns it to the term (Hindley[1969]). The inference system for types to terms, due to Curry, is both sound and complete w.r.t. the semantics of types over type-free structures given by the $\mathbf{M}_A$ construction. For this one may consult Hindley[1983], Coppo[1984] or Longo&Martini[1986]. In the latter completeness is shown by interpreting types and terms over a recursion theoretic type-structure (a special case of the $\mathbf{M}_A$ model); this establishes further relations between λ-calculus and higher type Recursion Theory.

3.8 Remark *(Some philosophy). As the reader may have noticed, we gave priority here to a model-theoretic view point, as we went from structures to theories. A beautiful unified framework, from an alternative, formalist, perspective, may be found in Huet[I-1986]. The line we followed may be considered as the usual and historical path in Mathematics, for functions in extenso, such as in Geometry or Phisics, were known before Church and Curry's formalizations of the Theory of Functionality. Even if the latter authors had a computational, algorithmic approach in mind, the formalization in Geometry of mathematical structures was the paradigm Church explicily referred to*

in his foundational activity (Church[1932]).

However, purely formal descriptions and results added plenty of information to Mathematics. For example, the original ideas contained in Church's formalization of Function Theory, the λ–calculus, required the construction of new structures: the $A \cong A^A$ models, say, which are non trivial Mathematics. In turn, models suggested "extensions" of the extant theories. $\lambda\beta\eta SP$ is the simplest example and it has, so far, only model-theoretic consistency proofs (see Barendregt[1984]); a richer extension of $\lambda\beta\eta$, also inspired by semantics, is given in Amadio&Longo[1986], for example. Besides extensions, structures sometimes suggest modifications of formal systems: a most relevant example is given by Girard's Linear Logic, where the meaning of "$\rightarrow$" in qualitative domains (see 1.9) guided a rewriting of inference (Girard[II-1985,1987]).

The formal behaviour of computers raised syntactic descriptions into a prominent place. However, the blending and interaction of denotation and meaning is a matter of riches of human thought : this is why both perspectives and, in particular, their interplay are relevant.

4. From higher types to higher order

The working mathematician often makes assertions concerning arbitrary functions in a given collection (when describing integration, say) or arbitrary subsets of a given set (when dealing with *all* the directed subsets of a c.p.o., say) or even with arbitrary sets within a given category or class of sets (*all* c.p.o.'s have a least element...). In view of the analogy "types-as-objects" given by theorem 3.3, the latter quantification would formally correspond to the possibility of quantifying over arbitrary types.

In the previous section we have been dealing with a language for higher type functions. Functional abstraction (i.e. $\lambda x....$) was defined w.r.t. to variables ranging over ground elements, functions, functionals and so on, in any finite higher type. Note that functional abstraction may be understood as a form of quantification; thus, as each boolean valued function determines a set, abstracting w.r.t. a variable which ranges over boolean valued functions is like quantifying over sets of a *given* type.

However, we were not allowed to quantify explicitly over *types*. Indeed, there is some implicit quantification over types in the systems mentioned at the end of §3. Church-Curry types are defined as type schemata: e.g. the identity $\lambda x.x$ has type schema $\alpha\rightarrow\alpha$, i.e. $\lambda x.x$ has type $\sigma\rightarrow\sigma$ for *any* type σ, or the collection of its possible types is obtained by consistently instantiating α in $\alpha\rightarrow\alpha$ by every type.

Mathematical practice and this implicit use of quantification suggest a language where one could explicitly consider all types: thus, a higher order language.

The language (λ_2), whose core is described below, is a variant of the "**system F**" due to Girard[III-1972]. The system F was invented for the purposes of second order Arithmetic, as its (inhabited) types correspond to (provable) formulas of a second order logic language (see the rules below).

The point with λ_2 is that one can quantify over type variables and term variables, as well. We first define the terms of the language, in a rather broad fashion: the formation rules for types and terms will tell us which are the legal types and terms and, at once, what are the types of the terms. We start with **Tp**, the symbol for the (collection of) types, and a set of **atomic** types or predicate letters. These may contain variables.

Terms: $a ::= Tp \mid Atomic \mid var \mid (aa) \mid (\lambda var{:}a.a) \mid (\forall var{:}a.a)$

As usual abc stands for ((ab)c) . We write capital letters for terms which are types or Tp itself, i.e. for terms A such that, for some assignment Γ , $\Gamma \vdash A : Tp$ or $A \equiv Tp$.

Well formed assignments: $\Gamma(x{:}A)$ stands for $\Gamma \cup \{(x{:}A)\}$; Γ is an ordered list.

ass.1 $\quad \varnothing$ ok (the empty assignment is well formed)

ass.2 $$\frac{\Gamma \text{ ok} , \ \Gamma \vdash A : Tp \ , \ x \notin dom(\Gamma)}{\Gamma(x{:}A) \ \text{ok}}$$

ass.3 $$\frac{\Gamma \text{ ok} , \ x \notin dom(\Gamma)}{\Gamma(x{:}Tp) \ \text{ok}}$$

We stress again the crucial point above: assignements are formed by allowing A to be a type (in ass.2, A:Tp) or to be Tp itself (in ass.3, $A \equiv Tp$). According to which possibility applies, in C.2 and C.3 bellow quantification is over term or type variables.

From now on, we agree that $\Gamma \vdash$ .... implies that Γ is ok.

Typing rules:

$$\text{C.0}\quad \frac{\Gamma \vdash A : Tp \qquad \Gamma (x{:}B)\ \ ok}{\Gamma (x{:}B) \vdash A : Tp} \qquad \text{(weakening)}$$

$$\text{C.1}\quad \frac{(x{:}A) \in \Gamma}{\Gamma \vdash x : A} \qquad \text{(assumption)}$$

$$\text{C.2}\quad \frac{\Gamma(x : A) \vdash B : Tp}{\Gamma \vdash (\forall x : A\ .\ B) : Tp} \qquad \text{(types' quantification)}$$

$$\text{C.3}\quad \frac{\Gamma(x : A) \vdash a : B}{\Gamma \vdash (\lambda x : A\ .\ a) : \forall x : A\ .\ B} \qquad \text{(abstraction or } \forall\text{–intro)}$$

$$\text{C.4}\quad \frac{\Gamma \vdash a : \forall x : A\ .\ B\ ,\ \ \Gamma \vdash b : A}{\Gamma \vdash (ab) : [b/x]B} \qquad \text{(application or } \forall\text{-elim)}$$

$$\text{C.5}\quad \frac{\Gamma \vdash a : A \qquad \vdash A = B}{\Gamma \vdash a : B} \qquad \text{(conversion for types)}$$

4.1 Notation. If $x \notin FV(B)$, set $A \to B \equiv \forall x : A.B$.

Remark. λ_2 has types depending on terms, as we allow atomic types to contain term variables; otherwise we would have exactly system F (by 4.1). Substitution and the typing of variables may be explicitly given for each atomic type P containing n variables, by

$$\text{(At)}\quad \frac{\Gamma \vdash a_i : A_i \qquad i \le n}{\Gamma \vdash [a_i/x_i]P : Tp}$$

Observe that (At) is the basis for forming ok assignements; in particular, by (At) and by ass.2, if $\{(x_1:A_1),..., (x_n:A_n)\}$ is ok and $x \in FV(A_i)$, then $x = x_j$, for some $j < i$.

The congruence relation "=" above is derived by the following **conversion rules**.

$$(\beta) \qquad \frac{\Gamma \vdash ((\lambda x : A \,.\, a)\, b) : B}{\Gamma \vdash (\lambda x : A \,.\, a)\, b = [b/x]a}$$

$$(\eta) \qquad \frac{\Gamma \vdash (\lambda x : A \,.\, ax) : B\,, \quad x \notin FV(a)}{\Gamma \vdash (\lambda x : A \,.\, ax) = a\,.}$$

Assignements rules and C.1 are self explanatory: they formalize assumptions made on the types of variables.

C.2 is the key rule. If A is a type (i.e. A:Tp), then C.2 and C.4 are first order rules, as quantification is over term variables ranging within a given type. Otherwise (i.e. if $A \equiv Tp$), $\forall x:Tp$ is clearly a second order quatification. Now, there are (at least) two possibilities: Girard suggested the approach we basically follow. As we want $\forall x:Tp.B$ to be a type (i.e. $(\forall x:Tp.B):Tp$), types are defined in an *impredicative* way: their collection (Tp), which is being defined, includes elements, such as $(\forall x:Tp.B)$, which are defined by refering (quantifying) over the collection itself.

Martin-Löf[III-1982] instead gives a second order predicative approach by stratifying the universe of types into several layers, Tp_1 , Tp_2 .... In short, for $Tp_1 = Tp$, if $A:Tp_1$ and $B:Tp_1$, then $(\forall x:A.B):Tp_1$, while $(\forall x:Tp_1.B):Tp_2$ and so on.

C.3 and C.4 tell us which terms live in universally quantified types and how they behave. In short, terms in $(\forall x:A.B)$ are functions (C.3) such that, when fed with a term b in A, they give as output a term of type $[b/x]B$ (C.4). Thus both the output and the type of the output both depend on the input. This is the core of dependent types and the main problem for the mathematical semantics of second order, jointly to impredicativity. It will be discussed in the next section.

As already mentioned, Girard invented second order λ-calculus as a tool for the proof theoretic investigation of second order Arithmetic: type formation rules, such as C.2, give second order formulas. The approach is soundly viewed as an intuitionistic perspective in Proof Theory:

$\lambda x{:}A.a$, say, is an effective proof of $\forall x{:}A.B$, as, given any term (proof) b of A , it computes a term (proof) of $[b/x]B$.

This language, though, which we presented in an extended version, somewhat in the style of the "Constructions" of Coquand and Huet (see Coquand&Huet[III-1985], Amadio&Longo[1986]), turned out to be relevant in itself, besides its proof-theoretic interest, mostly since the work started in Computer Science by Reynolds[1974].

We conclude this section by recalling that the terms of this calculus strongly normalize, i.e. any reduction strategy takes to a normal form (Girard[III-1972], Coquand[III-1985]). Girard applied this property to the proof theory of second order Arithmetic, as normalization of terms corresponds to normalization of proofs. By this, Takeuti conjecture on the normalizability of second order proof was settled, as well as its consequences: consistency, interpolation.

This crucial correspondence, in Proof Theory, may be summarized as follows. Consider this "simplification" or "reduction" of a natural deduction:

Inversion

$$\frac{\Gamma, x{:}A \vdash b{:}B}{\Gamma \vdash (\lambda x{:}A.\, b){:}\, \forall x A.B}\ (\forall\text{–intro}) \qquad \frac{\Gamma \vdash (\lambda x{:}A.\, b){:}\, \forall x A.B \qquad a{:}\, A}{(\lambda x{:}A.\, b)a : [a/x]B}\ (\forall\text{-elim})$$

reduces to

$$a{:}\, A \vdash [a/x]b : [a/x]B$$

By looking at terms, the above reduction rule for types (propositions) corresponds to β-reduction, i.e. $(\lambda x{:}A.\, b)a \;>\; [a/x]b$.

Observe that, if $x \notin FV(B)$ (and $A \neq Tp$), one obtains exactly Gentzen-Prawitz rule:

$$\frac{\Gamma, x{:}A \vdash b{:}B}{\Gamma \vdash (\lambda x{:}A.\, b){:}\, A \to B}\ (\to\text{–intro}) \qquad \frac{\Gamma \vdash (\lambda x{:}A.\, b){:}\, A \to B \qquad a{:}\, A}{(\lambda x{:}A.\, b)a : B}\ (\to\text{-elim})$$

reduces to

$$a: A \vdash [a/x]b : B$$

Again, this corresponds to $(\lambda x{:}A.\ b)a > [a/x]b$, which is exactly classical β-reduction.

4.2 Theorem *(Girard[III-1972]). (Second order) typed terms have a normal form (actually, they Strongly Normalize).*

4.3 Corollary*. Any deduction can be reduced to a normal deduction*
[i.e. where ($\rightarrow$–intro) and ($\rightarrow$-elim) are not used sequentially]

There are two more very relevant results which relate λ-calculus and Proof Theory, in this framework. One is concerned with the computational expressiveness of λ_2 , the other is a "concrete" independence result.

4.4 Theorem*. (Girard[III-1972]) Let $f : \omega \rightarrow \omega$. Then*

$$PA_2 \vdash \forall x \exists y f(x)=y \quad \Leftrightarrow \quad f \text{ is } \Lambda_2\text{-definable.}$$

Thus exactly the recursive functions which are provably total in PA_2 are definable in λ_2 .

As for independence, observe that the normalization property in 4.2 (Norm(λ_2), say) can be formalized in PA_2 and, by that very theorem, it is true in the standard model. However:

4.5 Theorem*. $PA_2 \not\vdash Norm(\lambda_2)$.*

(The proof in (Girard[III-1972]) is given by showing that

$$PA_2 \vdash Norm(\lambda_2) \rightarrow Cons(PA_2)) .$$

That proof of the independence result is very informative, as it also guaranties the (truth of the) consistency of PA_2 . However, a simpler one may be given, based on 4.4.

Given a term b , let b' be its normal form. Define then the function $\Phi(b) = b'$. Of course, modulo gödelization of terms, this is a number theoretic map and a universal function or "interpeter" in the sense of programming. Assume now that $PA_2 \vdash$ Norm(λ_2) or, equivalently, that $PA_2 \vdash \forall x \exists y \Phi(x) = y$. Then, by 4.4, Φ would be Λ_2-

definable, which is impossible, by the usual diagonal argument.

Remark. *One of the earliest and most relevant contributions to the triangular connection λ-calculus, higher type Recursion Theory, Proof Theory has been entirely omitted here: namely, Gödel's system T in his [1958]. It may suffice to say that the work in Girard[1972] may be viewed as an extension to Analysis (PA_2) of Gödel's work for PA. An introductory account to Gödel[1958] may be found in Hindley&Seldin[1986].*

5. Constructive Domains and Modest Sets as models for λ_2 .

Recall that our original motivation referred to the desire of representing higher type computations, for the purposes of Logic and of Computer Science. We defined Constructive Domains (**CD**; §.1) and Modests Sets (**M**; §.2) as a very natural framework for this; their naturality was clearly suggested by their relation to the category of enumerated sets (**EN**) and by the way they provided tools in order to extend pairings and gödel-numberings at higher types, following the HPEF. We then formalized the intended calculus, the typed λ-calculus, and characterized the class of models of that calculus (the CCC's; §.3). Logic and the practice of Mathematics suggested in turn an extended language and Type Theory suitable for the description of higher order constructs (λ_2 ; §.4).

In this section, we see how those structures, **CD** and **M**, yield also models for λ_2 .

As already mentioned, the crucial mathematical point is due to the second order, impredicative definition of λ_2 and the way types and terms mix up (rules C.2, C.3, C.4). In both models types will be interpreted as objects and terms as morphisms. In particular, one has to give a mathematical meaning to $(\forall x{:}Tp.B){:}Tp$, i.e. one has find an object wich interprets $(\forall x{:}Tp.B)$, where Tp is interpreted by a collection of objects, including the interpretation of $(\forall x{:}Tp.B)$ itself. This requires non trivial closure properties for the underlying structure. In particular, the interpretation of Tp must be closed under products indexed over Tp, i.e. under *dependent products* indexed over the structure itself, since elements (of the interpretation) of $(\forall x{:}Tp.B)$ interpret terms such as $(\lambda x{:}Tp.\ a)$, which are functions taking each element b of Tp to $[b/x]a$ of type $[b/x]B$ (recall C.2, C.3 and C.4).

The first model, over **CD**, will be given by turning the collection of all (interpretations of) types into an object of **CD**. Thus Tp itself will be interpreted as a type. This strong closure property will greatly simplify the interpretation of $(\forall x{:}Tp.B){:}Tp$.

The second model is based on early ideas in Girard[III-1972] and Troelstra[1973c] and on a recent unpublished result of Moggi, who proved unexpected closure properties of **M**, as a crucial

substructure of the Effective Topos in Hyland[II-1982].

The Constructive Finitary Projection Model

We start with a model in **CD** for the classical type-free λ-calculus (λβ), that is, by theorem 3.6.2, over an object U of **CD** such that $U^U < U$. The existence of such an U will be guarantied by, say, a constructive version of Scott's D_∞ construction (see Barendregt[1984], Hindley& Seldin[1986] and, for the effective counterpart, Smyth[I-1977], Kanda[I-1979], Giannini& Longo[I-1984]). To be precise, something more is required; this motivates the following definitions.

Recall that the morphisms in **CD** are continuous and computable maps, partially ordered pointwise. The following definition is a constructive version of the model in Amadio&al[II-1986] (which was inspired by MacCracken[II-1984] and Scott[II-1980b]). As usual, we identify a morphism space CD[A,B] with its representative B^A, when needed and unambiguous.

***5.1 Definition**. Let A , B be constructive domains. (θ,φ) is a **projection pair** on A, B iff θ∈CD[A,B] , φ ∈CD[B,A] and*

$$\phi \circ \theta = id\lceil A, \quad \theta \circ \phi \leq id\lceil B.$$

Write $A \underline{\ll} B$ iff there is a projection pair of A into B .

Note that a projection pair is more than a retraction pair in the sense of §.3.5.

***5.2 Definition**. Let A , B be constructive domains such that $A \subseteq B$ and $\leq_A = \leq_B \lceil A \times A$. $\phi \in CD[B,A]$ is a **projection** iff for all $b \in B$,*

$$\phi(b) \leq_B b \text{ , and for all } a \in \text{range}(\phi) \text{ , } \phi(a) = a .$$

We write $A \underline{\trianglelefteq} B$ if φ is onto.

Thus a projection is a retraction less or equal to the identity.

It is easy to show that $\underline{\ll}$ and $\underline{\trianglelefteq}$ are reflexive and transitive (although $\underline{\ll}$ is not antisymmetric). Moreover, if $A \underline{\ll} B$ then there is an $A' \cong A$ such that $A' \underline{\trianglelefteq} B$.

Next we show how to define a constructive domain which represents a collection of constructive domains. This will be done by taking as constructive domains the ranges of a particular class of projections.

***5.3 Definition**. A projection φ in **CD** is said to be **finitary** if the range of φ is a domain (and thus a constructive domain).*

We note here that essentially all projection pairs which normally arise are finitary. We are now ready to define the constructive domain which will represent the type of all types, that is the domain of the Constructive Finitary Projections.

5.4 Definition. *Let* U *be a constructive domain such that* $U^U \underline{\ll} U$. *Then let* $\mathbf{CFP_U} = \{ \phi \in CD[U,U] \mid \phi$ *is a finitary projection}.*

U as in 5.4 exists by the effective D_∞ construction recalled above, which actually gives $U^U \cong U$. One may also find an object U which strictly satisfies $U^U \underline{\ll} U$: take, say, the constructive part of the "filter model" in Barendregt&al[1983] and its variant in Coppo&al[1984].

If U is obvious from the context then we write simply CFP. Fix U as in the definition above.

5.5 Theorem. *(i) CFP is a constructive domain.*
(ii) $(\varphi \in CFP \Rightarrow range(\varphi) \underline{\trianglelefteq} U)$ *and* $(A \underline{\trianglelefteq} U \Rightarrow \exists \varphi \in CFP\ range(\varphi)=A)$.

The constructive domains we will be interested in are the subdomains of U. Notice that, by 5.5(ii), there is a one-to-one correspondence between elements of CFP and constructive domains $A \underline{\trianglelefteq} U$. Thus CFP represents the collection of subdomains of U. Somewhat surprisingly, CFP can be isomorphically embedded as a subdomain of U.

5.6 Lemma. $CFP \underline{\trianglelefteq} U^U$ *and hence* $CFP \underline{\ll} U$.
Proof hint: Define $\phi \in CD[U,U]$ by

$$\phi(g) = \sup_U\{f \le g \mid f \in CFP\}.$$

Clearly $\phi(g) \le g$ and if $g \in CFP$ then $\phi(g) = g$. Thus $range(\phi) \supseteq CFP$. Conversely, $CFP \supseteq range(\phi)$ since CFP is consistently complete w.r.t. $CD[U,U]$.
Therefore $CFP \underline{\trianglelefteq} U^U \underline{\ll} U$ and $CFP \underline{\ll} U$ by transitivity. Δ

Let now $\Psi \in CD[U^U,U]$, $\Phi \in CD[U, U^U]$ be the projection pair of U^U into U. Set $CFP = \{\Psi(f) / f \in CFP\}$: these are the canonical representative of CFP within U. When there is no ambiguity we identify CFP and CFP.

5.7 Corollary. *There exists* $p \in CFP$ *such that* $range(p) = CFP$.

Types will be interpreted by finitary projections. More precisely, types are *ranges* of

finitary projections. Note first that finitary projections are particular retractions, and that , if r is a retraction, then range(r) = { u∈ *U* / u = ru }, the set of its fixed points. Moreover, finitary projections and their ranges tidily relate, by the following fact.

5.8 Proposition. *Let f,g ∈ CFP. Then f ≤ g iff range(f) ⊆ range(g) .*

We are now ready to define our second order model. We sketch how to interpret types: details about the interpretation of terms may be found in Amadio&al[II-1986] or a simpler syntactic translation, sufficient for the guidelines of the interpretation, may be seen in Amadio&Longo[1986].

Recall that we interpret Tp by *CFP* or, equivalently, by p . Ground types (integer, booleans..., if given in the theory) are interpreted as subdomains of *U* , which is rich enough for this purpose, since it is a model of λβ . In order to give first an informal explanation on how to interpret higher types, we mix up syntax and semantics; λx.f(x) is the informal lambda notation for functions. The key point is that, in all interpretations of types as retractions, a:A is interpreted by a ∈ range(*A*) or, equivalently, by a = *A*a , where the retraction *A* interprets type A . The definitions of "→" and "∀" originate from elementary notions in Category Theory (see Scott[I-1976,II-1980b], Seely[II-1984,1986]]).

Recall that "→" is just a special case of "∀" , by 4.1. We discuss this simple case first. In a category **C**, if an object A is a retract of B via (i,j), then A , as a "subtype" of B , may be identified with (i,j) or, by some abuse of language, since categories do not need to have points or elements, it may be identified with the fixed points of i∘j (the range of i∘j, which is a retraction).

If **C** is a CCC, let C^A be the exponent of C and A in **C**; then, if A is a retract of B via (i',j') and C is a retract of D via (i,j), one has

C^A is a retract of D^B via (λx.i∘x∘j', λx.j∘x∘i') .

Indeed, (λx.i∘x∘j')∘(λx.j∘x∘i') is a retraction and its fixed points may be identified with C^A as a subtype of D^B . In other words, if one writes r = i∘j and s = i'∘j' , then C^A coincides with {x/ x = r∘x∘s } = range(λx.r∘x∘s) , where r , s and λx.r∘x∘s are all retractions.

In our case, over the type-free universe *U* , if (i,j) and (i',j') are projection pairs, then also (λx.i∘x∘j', λx.j∘x∘i') is so, and thus r, s and λx.r∘x∘s are all finitary projections, whose ranges are subdomains of *U* .

Thus, if types A and C are interpreted as finitary projections *A* and *C* , one has:

($\rightarrow$Interpret.) $A \rightarrow C$ is interpreted as $\lambda x.C \circ x \circ A$ (or its range).

As for "$\forall$", consider first rule C.3. This is a formation rule for terms; its meaning is that terms which have an applicative behaviour (λ-abstractions) can be only applied to terms of the intended imput type (A in the rule). The idea, in models where types are retractions, is to interpret those terms as functions which coerce each input to be of the right type. That is, $\lambda x{:}A.a$ wil be interpreted as $f \circ A$, where f depends on a .

As for rule C.4, the intuition is that a has type $\forall x{:}A.B$ iff a is a function which takes any b in (the range of) A into an element ab of (the range of) $[b/x]B$. Since types are particular retractions, this means ab is a fixed point of $[b/x]B$:

$$ab = ([b/x]B)(ab) .$$

Thus, $ab = (\lambda x.B)b(ab)$. Since $b = Ab$, then

$$ab = (\lambda x.B)(Ab)(a(Ab)) .$$

Observe now that a must be a λ-abstraction, by C.3, i.e. it is interpreted by $f \circ A$, for some f ; therefore

$$ab = (f \circ A)b = (\lambda x.B)(Ab)(f \circ A (Ab)) .$$

That is, a coerces *any* argument b to be in (the range of) the retraction A . Thus one may abstract (generalize) w.r.t. b :

$$a = \lambda t.(\lambda x.B)(At)(a(At)) .$$

Equivalently:

$$a = (\lambda zt.(\lambda x.B)(At)(z(At)))a .$$

Indeed, $(\lambda zt.(\lambda x.B)(At)(z(At)))$ turns out to be a retraction, when A and B are retractions. Thus, $a{:}(\forall x{:}A.B)$ gives, in the model, that a is a fixed point of the retraction $\lambda zt.(\lambda x.B)(At)(z(At))$.

The informal argument above shows that $\lambda zt.(\lambda x.B)(At)(z(At))$ soundly interprets $\forall x{:}A.B$, as we derived it exactly from the intended meaning of universal quantification as dependent product, i.e. as $\Pi_A([b/x]B)$. This may be summarized as follows:

5.9 Theorem. *CFP is a CCC. Moreover, for $A,B \in CFP$, one has*

$$range(\lambda zt.(\lambda x.B)(At)(z(At))) = \Pi_A([b/x]B) .$$

A category-theoretic understanding of this may be found in Seely[1986].

In conclusion:

(∀Interpret.) $\forall x{:}A.B$ is interpreted as $\lambda zt.(\lambda x.B)(At)(z(At))$ (or its range).

If B does not depend on x , then ($\rightarrow$ Interpret.) is a special case ($\forall$ Interpret.) , as one may easily check by β-reduction (cf. 3.1). As a side remark, for $p = pp$, observe that the *CFP* model also interprets an extension of the given language by Tp:Tp , i.e. the collection of types is a type (see Amadio&Longo[1986] for a discussion).

Problem. In Amadio&Longo[1986] a simple extension $\boldsymbol{\lambda\beta\eta p}$ of $\lambda\beta\eta$ is proposed, where $\boldsymbol{\lambda_2}$ can be easily "interpreted" (or translated). Add for this a constant symbol **p** to $\lambda\beta\eta$ jointly with the following two axioms and rule:

a.1) $pp = p$

a.2) $px \circ px = px$

R) $\dfrac{M \circ M = M}{pM = M}$

Clearly, $\lambda\beta\eta p$ is directly inspired by semantics and provides a very simple "model" for $\boldsymbol{\lambda_2}$, by the translation described above. However, **CFP** is not a mathematical model for $\lambda\beta\eta p$ and $\lambda\beta\eta p$ is not Church-Rosser (more precisely, there is no simple extension of $\lambda\beta\eta p$ to a CR reduction system, Amadio&Longo[1986]).

Conjecture: $\lambda\beta\eta p$ is consistent; it should even be conservative over $\lambda\beta\eta$.

The Modest or HEO^2 Model

The cartesian closed extension **M** of the category **EN** of numbered sets has been defined in §.2. In that category, types are interpreted as quotient subsets of ω .

In Girard[III-1972] and Troelstra[1973c] some hints are given on how to build a model for λ_2 in **M**: the **HEO^2** model. Second order terms are interpreted by erasing types from them and universally quantified types are interpreted as intersections. In that way, then, information is lost from terms and no apparent connection is given between intersection and the natural interpretation of universally quantified types as dependent products. This is not a criticism of that early work, first because of its pioneering role, second because some deep mathematical intuition was already present also in that sketchy model construction: for example, in Girard[1972], terms are more precisely interpreted as pairs (type-free term, its type). Surprisingly enough this is sufficient to recover the information preserved in the interpretation summarized below, since one

may easily prove that the interpretation of a typed term is the equivalence class of its type (a quotient set), which contains that term, as pointed out below. Recently, Moggi[II-1986Tp] suggested how to turn the **HEO^2** into a fully satisfying model of λ_2.

Remark *(Some more references)**.** We refer to Moggi's version of this result, which has not yet been fully written down (see the conclusion on how the discussion started on elettronic mail). However, since Moggi mentioned to have proved the "internal completeness" (i.e. the closure under all limits) of **M** within the topos in 5.12, several other relevant categories, in the same framework, have been shown to be internally complete by Martin Hyland, Pino Rosolini, Dana Scott. (Hyland's lecture in this volume should present a broader and less elementary account of this story: the reader is recommended to refer to it for a more category-theoretic oriented presentation. Also Freyd and Carboni recently devised a generalization of the results below).*

If $\underline{A} = (A,\pi_a)$ is an object in **M**, write $\underline{A}\{n\} = \{m \mid m\underline{A}n\}$ for the equivalence class of n w.r.t $\underline{A}$. In particular, then,

$$f \in M[\underline{A},\underline{B}] \quad \text{iff} \quad \exists n \;\; f = \underline{B}^{\underline{A}}\{n\} .$$

As types are interpreted by Ob_M (quotient sets), terms will be elements of types, i.e. equivalence classes: for $\underline{A}$ interpreting A,

a:A is interpreted as "$\underline{a}$ is an equivalence class in $\underline{A}$",

or, also, $\exists n \;\; \underline{a} = \underline{A}\{n\}$. (We write $\underline{x}$ for the interpretation of the term or type x).

More precisely, $A \to B$ is interpreted as $\underline{B}^{\underline{A}}$ and universally quantified types as follows. We discuss, for the sake of simplicity, $\forall x{:}A.B$ only when $A = Tp$, i.e. when A is the collection of types: the crucial, impredicative case. For the other case see 5.10.2.
We keep using an informal λ-notation for functions and $\to$-notation for function spaces (not necessarily formal terms and types).

Note first that $\forall x{:}Tp.B$ may be understood as $\forall(\lambda x{:}Tp.B)$ where $\lambda x{:}Tp.B$ is a function from types to types. As types are objects of **M**, $\forall : (Ob_M \to Ob_M) \to Ob_M$ turns each function $\lambda\underline{x}{:}Tp.\underline{B} : Ob_M \to Ob_M$ into a type, i.e. an object of **M**. Thus, all that we have to do is to find a meaning in **M** to $\forall(f)$ for at least each formally definable function f (by the notation for types).

In general, given $f : Ob_M \to Ob_M$, define $\forall : (Ob_M \to Ob_M) \to Ob_M$ by

$$\textit{for all } n, m \in \omega, \quad n \;\forall(f)\; m \quad \textit{iff} \quad \textit{for all } \underline{A}, \; n \; f(\underline{A})\; m .$$

Clearly, $\forall(f)$ is a partial equivalence relation. Thus,

(∀Interpret.) $\forall x{:}Tp.B$ is interpreted as $\forall(\lambda \underline{x}{:}Tp.\underline{B})$.

5.10 Remark. *1 -* $\forall(f) = \cap_{\underline{A}} f(\underline{A})$.

2 - In case $A{:}Tp$ *, define*

$$p(\forall \underline{x}{:}A.\underline{B})q \text{ iff for all } n, m \ (n \underline{A} m \Rightarrow \varphi_p(n) \ ([\underline{A}\{n\}/x] \ \underline{B}) \ \varphi_p(m))$$

which collapses to (Quot) of §.2, if $x \notin FV(B)$.

An other key point, formalized in rule C.4, is that terms of type $\forall x{:}Tp.B$ may be applied to a type: thus, elements of $\forall(f)$, that is equivalence classes, must be applicable to quotient sets, that is to collections of equivalence classes.

5.11 Definition *(Moggi; Polymorphic application). Let* $f : Ob_M \to Ob_M$. *Set*

$$APP_f(\forall(f)\{n\}, \underline{A}) = f(\underline{A})\{n\}.$$

Note that this is a good definition, since App_f depends on f : as $\forall$ is not injective, relevant information from f could be lost when trying to recover $f(\underline{A})$.

Our aim now is to prove that the interpretation of second order types based on the definition of $\forall(f)$ above, i.e. as intersection, preserves the naive mathematical meaning one would attach to them; namely, it corresponds to a product indexed over the interpretation Ob_M of Tp. In other words, a suitable framework, or category, must be found such that, for $f : Ob_M \to Ob_M$, in the category,

(Iso) $\forall(f)$ is (isomorphic to) $\prod_M f(\underline{A})$.

Clearly, (Iso) would give a strong closure property of **M**, as $\forall(f)$ is trivially in Ob_M .

The embedding in one direction is easy and true for every set-theoretic function $f : Ob_M \to Ob_M$. Indeed, by 5.11, one may injectively associate to each element $\forall(f)\{n\}$ of $\forall(f)$ a function $\lambda \underline{A}.(f(\underline{A})\{n\})$ in $\prod_M f(\underline{A})$, as $\lambda \underline{A}.(f(\underline{A})\{n\}) : Ob_M \to f(\underline{A})$.

Conversely, given $g \in \prod_M f(\underline{A})$, by definition of dependent product, one has

(1) $\forall \underline{A} \ \exists n \ g(\underline{A}) = f(\underline{A})\{n\}$.

And here is the key point. In order to prove that g has indeed the structure $\lambda \underline{A}.(f(\underline{A})\{n\})$, i.e. that, for some n , g is the same as an equivalence class $\forall(f)\{n\}$, one has to reverse the quantification in (1) and prove

(2) $\exists n \ \forall \underline{A} \ g(\underline{A}) = f(\underline{A})\{n\}$.

The observation that, under certain circumstances, one can actually go from (1) to (2), is independently due to Hyland and Moggi and is based on the use of a very

constructive framework, i.e. a particular model of Intuitionistic Set Theory (IZF), Hyland's **Effective Topos** (**Eff**; Hyland[II-1982], Hyland&al[II-1980]).

We give some hints for that structure and sketch how one can view at **M** at once as a full subCCC and an *object* of **Eff**, in the sense of 5.13 below.

***5.12 Definition**. (**Eff**) Objects: $(A, =_A)$, with $=_A : A \times A \to P\omega$ partial ;*

Morphisms: $Eff(A_1,A_2)$ is the set of the "total functions" w.r.t. $=_1$ and $=_2$

(i.e. total, single-valued , strict and substitutive relations).

Observe now that **M** is a small category; that is, Ob_M and the collection of all morphisms are objects of **Set** (they are just sets). More formally, this amounts to say that **M** is an **internal category** of **Set**, the classical category of sets (see Johnstone[1977]). The nice fact is that **M** may be also seen as an internal category of **Eff**, as there exist M_0, M_1 in Ob_{Eff} representing the objects and the morphisms of **M**, respectively.

***5.13 Theorem**. **M** is a full subCCC of **Eff** and an internal category of **Eff**.*

(Proof hint: for $(A,e_A)\in Ob_M$ set $(a =_A b) := \{n / e_A(n) = a \wedge a = b\}$; thus $(A, =_A) \in Ob_{\mathbf{Eff}}$. Note that $(A, =_A) \in Ob_{\mathbf{Eff}}$ is (isomorphic to an object) in $Ob_{\mathbf{M}}$ iff

$$(\exists n \in (a =_A b) \cap (c =_A d) \iff a = b = c = d) ,$$

in this case set $e_A(n) = a$ iff $n \in (a =_A a)$. Thus the embedding is full and faithful.

In order to turn **M** into an internal category, set $M_0 = (X_0, =_0)$ with $X_0 := Ob_M$ and $=_0: X_0 \times X_0 \to P\omega$ given by

$$(\underline{A} =_0 \underline{B}) := \text{ if } \underline{A} = \underline{B} \text{ then } \omega \text{ else } \varnothing.$$

M_0 represents Ob_M . As for the morphisms, take $M_1 = (X_1, =_1)$ with

$$X_1 := \{(\underline{A}, \underline{B}^{\underline{A}}\{n\}, \underline{B}) / \underline{A}, \underline{B} \in Ob_M,\ n\ \underline{B}^{\underline{A}}\ n \}$$

and $=_1: X_1 \times X_1 \to P\omega$ given by

$$((\underline{A}, \underline{B}^{\underline{A}}\{n\}, \underline{B}) =_1 (\underline{A}', \underline{B}'^{\underline{A}'}\{m\}, \underline{B}')) := \text{ if } \underline{A} = \underline{A}',\ \underline{B} = \underline{B}',\ \underline{B}^{\underline{A}}\{n\} = \underline{B}'^{\underline{A}'}\{m\},$$
$$\text{then } \underline{B}^{\underline{A}}\{n\} \text{ else } \varnothing .) \quad \Delta$$

Eff is a topos and, thus, a model of intuitionistic Logic. It also satisfies, among other properties, the Uniformity Principle (or König's Lemma), for M_0 is as in 5.13:

(UP) $\quad \forall \underline{A} \in M_0\ \exists n\ \Phi(\underline{A},n) \Rightarrow \exists n\ \forall \underline{A} \in M_0\ \Phi(\underline{A},n)$.

By applying (UP) on can go from (1) to (2). (This requires some work, within **Eff**; note also that the isomorphism in (Iso) above depends on f .)

In the model, then, the quantification over types (or Ob_M), is the same as over M_0 , an object, now, of **Eff**. Moreover, we can fix the class of functions for which (Iso) above holds: they are just the morphisms in **Eff** from M_0 to M_0 (i.e. the internal functions). By this and by the Uniformity Principle (UP), the isomorphism in **Eff** between $\forall(f)$ and $\Pi_M f(\underline{A})$ is proved.

The strong closure property for **M**, within **Eff**, we have sketched, that is its closure under products indexed over **M** itself, is the mathematical meaning of Girard's impredicative definition of second order types, over this very natural model, the "modest" or quotient subsets of ω .

5.14 Remark. *(Models and Intuitionistic Logic) The proof-theoretic connections between λ–calculus and Intuitionistic Logic were clear since Curry-Howard remark that the inhabited types of λ–terms are exactly the propositions of (positive) intuitionistic propositional calculus. This analogy was fruitfully extended at higher orders by Girard and Martin-Löf. The relevance of the intuitionistic perspective should now be clear also in the model theory of λ-calculus. As for the first order case, this is stressed by the constructive approach we followed here, which can be framed within a categorical approach to the semantics of Intuitionistic Logic, as shown in Rosolini[II-1986] (see also MItchell&Moggi[II-1987], for an elementary, elegant use of Kripke models). By the result just presented, this is even more striking in the second order case. As a matter of fact, in Reynolds[II-1984] it is shown that there is no non trivial model of Girard's second order language* $\boldsymbol{\lambda_2}$ *in the category of sets, if classical Set Theory (ZF) is taken. In Reynolds' words: polymorphism is not set-theoretic. By the discussion above, a set-theoretic model may be found, provided that a suitable model of IZF is taken; that is "polymorphism is* ***intuitionistically*** *set-theoretic".*

As pointed out before 3.5, the definition of **M** may be easily relativized to an arbitrary (partial) model of Combinatory Logic, CL; the same applies to the definition of **Eff**. Thus, instead of Kleene's $(\omega,.)$, take a model V of type-free $\lambda\beta$ and consider $\mathbf{M}_V$ and $\mathbf{Eff}_V$ over $(V,\cdot)$, see 3.6.

Given a second order term a , let **er(a)** be the erasure of a , i.e. the untyped term obtained from a by erasing all type information. Of course, er(a) may be soundly interpreted over V ; call $\underline{er(a)}$ its meaning. By induction one can easily establish the following tidy connection between $\underline{er(a)}$ and the interpretation of a in $\mathbf{M}_V$. Assume that $a:A$ in λ_2 , then

$$\underline{a} = \underline{A}\{\underline{er}(a)\} \ ,$$

i.e. the interpretation of the typed term a is the equivalence class of $\underline{er}(a)$ w.r.t. $\underline{A}$ (details may be found in Mitchell[1986]). Thus, the interpretation sketched in Girard [III-1972] contained enough information.

We conclude this section by relating the various constructions above by some useful category-theoretic embeddings. These will only relate, by cartesian functors, the first order structures of the various categories presented here, i.e. ordinary products and exponents, as products indexed over different categories do not seem to bear any relevant connection.

As worked out in Amadio&al[II-1986], the CFP construction in the first part of this section may be performed over any Scott domain U such that

$$U^U \underline{\ll} U \ .$$

Namely, over any λ-model where the retraction in 3.6 is indeed a finitary projection in the category of domains (see 5.3). Thus, when dropping the request on costructivity in 5.4, call $\mathbf{FP}_U$ the corresponding model of λ_2. In this general case, one has that

FP_U is a full subCCC of M_U, which is a full subCCC and an internal category of Eff_U.

This may be proved by putting toghether the previous work and Scott[I-1976], Hyland[II-1982], Longo&Moggi[II-1986].

Within the constructive appoach followed here, where we began with numbers and enumerations, analogue embeddings may be factorized through **CD** without relativizing the constructions of **M** and **Eff**. That is:

5.15 Theorem. ***CFP_U*** *full subCCC of*
CD *full subCCC of*
M *full subCCC and an internal category of*
Eff *.*

With some more work, one can look also at **CD** as an internal category of **Eff**; indeed, an internal subcategory of **M**.

Conclusion

The interest in Church's λ-calculus is mostly due, nowadays, to its relevance in Computer Science. We already quoted a few areas where this is explicit. As most of the problems and issues discussed in this lecture derive from the practice of computing, it is worth mentioning a few more references

which set a bridge between the Logic and the computing perspectives in λ-calculus.

As mentioned in the introduction, Scott's invention of models of the λ–calculus started denotational semantics of programming languages and brought into language design, via λ-calculus, the well-established tools of Tarskian semantics, i.e. the mathematical investigation of denotation and meaning. The analysis of the connections between theories and models received an important impulse by the results in Wadsworth[1976] and Hyland[1976] and was continued by several authors (e.g. Barendregt& Longo[1980], Longo[1983]). The interest in applications of this analysis is concerned with the comparison between operational and denotational semantics, as theories provide the operational description of languages. Further work in this direction led to the issue of "fully abstracness" in Computer Science, motivated by the desire of proving results on languages and operations by a direct analysis of models; this is possible by a full correspondence, in some cases, between denotation and meaning (see Mulmuley [1985] and Luke-Ong [1987] for recent work).

A relevant example of the influence of denotational semantics in language design is given by the continuously expanding Edinburgh programming language ML (Milner[1978], Gordon&al[1979], Damas[1985], Milner[1986]). Even compilers are nowadays built up with some use of model theoretic concepts of λ-calculus (Jones[1980]).

The higher order or explicity polymorphic languages provide a very interesting area for new applcations of λ-calculus. On one hand, the language invented by Girard for the purposes of proof-theory is the core of a several ways interaction among proof-theory, topos theory and the theory of functional languages. For example, Moggi's theorem above was given as an answer, raised by Albert Meyer, on the consistency of certain extensions of explicitly polymorphic languages. The result answered the question, by providing a model, and went further because it opened up a whole line of research in polymorphic model theory.

On the side of language design, the various theories of types have served to organize the study of type disciplines in programming and are now implemented in several languages (Nordstrom [III-1981], Burstall&Lampson[1984], Damas[1985], Constable&al[III-1986]). These investigations and their applications lead to new insights into polymorphism, modularity and abstraction, mostly since the work of Reynolds[1975] and Milner[1978] (see also Reynolds[1985], MacQueen[1986], Cardelli&Wegner[1985], Cardelli[1986]).

Under the motto "types as formulae" (see §4), Type Theory greatly influenced also automated theorem proving (deBruijn[III-1980], Constable&al[III-1984], Miller[III-1984], Coquand&Huet [III-1985]) and it even serves as a knowledge representation language for AI (Turner[1984], Constable&al [III-1986]). The other motto, "types as objects", summarizes instead the connections with Category Theory (see §3 and Lambek&Scott[II-1986]); surprisingly enough, even these very abstract studies influenced programming, since the equations mentioned in §3 have become the core

of a running machine (Cousinau&al[1985]).

Connections to other approaches in semantics (equational, algebraic) nicely come in, in the typed and higher order cases, by results an conservativity of extensions of equational theories (see Breazu-Tannen&Meyer[1987]) and by an original understanding of "abstract data types" (see Mitchell&Plotkin[1985]).

Acknowledgement. I am greatly endebted to my (former) student Eugenio Moggi: his deep insight into Mathematics and his curiosity for scientific knowledge meant much to me. In particular, he tought and keeps teaching me most of what I know on the categorical approach to life (and to the foundation of Mathematics).

Several people red an earlier version of this paper and suggested corrections and changes; these include Pierre Louis Curien, Pino Rosolini, Eugenio Moggi, Roger Hindley, Dieter Spreen, Kim Bruce.

Bibliography

This bibliography is organized according to the "line of thought" followed in the lecture. Thus **Part I** lists contributions to the areas broadly relating λ-calculus to higher type Recursion Theory, **Part II** is concerned with the connections to categories and **Part III** to Proof Theory.

The list is far from complete. Moreover, many entries should appear in more than one part and the classification is as arbitrary as any. For example, Scott[I-1976] had mostly relevance in Computer Science and uses several category-theoretic notions; however, the key ideas derive from type 2 Recursion Theory (classical Myhill-Shepherdson theorem). Similarly, Girard[II-1985] does not deal explicilely with Category Theory, but functorial notions play a major role. Ershov[I-1976] or Longo& Moggi[I-1984], say, do not mention λ-calculus at all; the underlying mathematical structures, though, are entirely borroughed from the model-theory of λ-calculus. Finally, the inclusion in Part III of some Computer Science papers is due to the increasing impact of theoretical ideas in automated deduction and program synthesis, which in turn stimulated the theory.

It should also be clear that papers in the "pure theory", both of types and of terms, are essentially not included: rather complete references to writings "within" λ-calculus may be found in Barendregt[1984] and Hindley&Seldin[1986]. The fourth and last part contains papers referred to in the previous pages, including a few papers in the pure theory.

Part I

Asperti A.[1987] "Stability, sequentiality and oracles" Dip. di Informatica, Pisa.

Barendregt H., Longo G. [1982] "Recursion Theoretic Operators and Morphisms of Numbered Sets," M.I.T., Lab. for Computer Science Tech. Mon. 194, **Fundamenta Mathematicae** CXIX (pp. 49-62).

Ershov Yu. L. [1976] "Model C of the partial continuous functionals," **Logic Colloquium 76** (Gandy, Hyland eds.) North Holland, 1977.

Friedman, H. [1975], "Equality between functionals," **Logic Colloquium** (Parikh ed.), LNM 453, Springer-Verlag.

Giannini P., Longo G. [1984] "Effectively given domains and lambda calculus semantics," **Information and Control** 62, 1 (36-63).

Kanda A. [1979] "Fully effective solutions of recursive domain equations" Proc. of MFCS'79, LNCS 74, Springer-Verlag.

Kleene S. C. [1936] "Lambda definability and recursiveness," **Duke Math. J.**, 2, (pp. 340-353).

Longo, G. [1982] "Hereditary partial effective operators in any finite type," Forshungsinstitut für Mathematik E.T.H. Zürich, 1982.

Longo, G. [1984p] "Limits, higher type computability and type free languages," **MFCS'84**, Prague (Chytil, Koubek eds.), LNCS 176 , Springer- Verlag, 1984 (pp. 96-114).

Longo G., Martini S. [1984] "Computability in higher types, Pω and the completeness of type assignement," **Theor. Comp. Sci.** 46, 2-3 (197-218).

Longo G., Moggi E. [1984] "The Hereditary Partial Recursive Functionals and Recursion Theory in higher types," **J. Symb. Logic**, vol. 49, 4 (pp. 1319-1332).

McCarty D. [1984] "Realizability and Recursive Mathematics" Ph. D. Thesis, Computer Sci. C.M.U..

Spreen D. [1984] "On r.e. inseparability of cpo indexed sets" (Börger et al. eds) LNCS 171, Springer- Verlag.

Spreen D., Young P. [1984] "Effective operators in a topological setting" **Logic Coll. 83** (Börger et al. eds) LNM, Springer- Verlag.

Scott D. [1970] "Outline of a mathematical theory of computation" **4th Ann. Princeton Conf. on Info. Syst. Sci.**(pp. 169-176).

Scott D. [1976] "Data types as lattices," **SIAM Journal of Computing**, 5 (pp. 522-587).

Scott, D. [1982] "Some ordered sets in Computer Science," in **Ordered Sets** (Rival Ed.), Reidel.

Scott D. [1982] "Domains for denotational semantics," (preliminary version), Proceedings ICALP 82, **LNCS** 140, Springer-Verlag).

Smyth M. [1977] "Effectively Given Domains", **Theoret. Comput. Sci. 5,** pp 255-272.

Weyrauch K. [1981] "Recursion and complexity theory on cpo's" (Deussen et al. eds) LNCS 104, Springer-Verlag.

Part II

Adachi T. [1983] "A categorical characterization of lambda-calculus models" Dept. of Info. Sci. Tokyo Inst. Tech., n. C-49.

Amadio R., Bruce K. B., Longo G. [1986] "The finitary projections model and the solution of higher order domain equations" **IEEE Conference on Logic in Computer Science**, Boston, June 1986.

Asperti A., Longo G. [1986] "Categories of partial morphisms and the relation between type-structures" Invited Lecture, **Semester on theory of Computation**, Banach mathematical Center, Warsaw, December 1985 (preliminary version: CAAP '86, LNCS 214, Spriger-Verlag).

Berry G. [1979] "Modèles complètament adequats et stables des lambda-calculus types", Thèse de Doctorat, Universite' Paris VII.

Berry G. [1979] "Some Syntactic and Categorial Constructions of lambda-calculus models" INRIA, Valbonne.

Breazu-Tannen V., Meyer A.[1985] "Lambda Calculus with Constrained Types" *in* **Logic of Programs**, Parikh (ed.), LNCS 193, Springer-Verlag.

Bruce K., Longo G. [1985] "Provable isomorphisms and domain equations in models of typed languages," (Preliminary version) **1985 A.C.M. Symposium on Theory of Computing (STOC 85)**, Providence (R.I.) , May 1985 (pp. 263-272).

Curien P.L., [1986] "Categorical Combinarors" **Info.&Contr.** (to appear).

Curien P.L., [1986] **Categorical Combinators and Functional Programming**, Pitman.

Dibjer P. [1983], "Category-theoretic logics and algebras of programs," Ph.D. Thesis, Chalmers University, Goeteborg.

Gunter C. [1985] "Profinite solutions for Recursive Domain Equations" Ph. D. Thesis, Comp. Sci. Dept., C.M.U..

Girard J.Y.[1985] "The system F of variable types, fifteen years later" **TCS**, to appear

Hayashi S. [1985] "Adjunction of semifunctors: categorical structures in non-extensional lambda-calculus" **Theor. Comp. Sci.** 4.

Hyland M. [1982] "The effective Topos," *in* **The Brouwer Symposium**, (Troelstra, Van Dalen eds.) North-Holland.

Hyland M., Johnstone P., Pitts A. [1980] "Tripos Theory," **Math. Proc. Camb. Phil. Soc.**, 88 (205-232).

Koymans K. [1982] "Models of the lambda calculus", **Information and Control,** 52, pp. 306-332.

Lambeck J. [1968] " Deductive systems and categories," **I. J. Math Systems Theory 2**, (pp.278-318).

Lambeck J. [1974] "Functional completeness of cartesian categories," **Ann. Math. Logic 6**, (pp.259-292).

Lambek J. [1980] "From lamba-calculus to cartesian closed categories," *in* Hindley andSeldin[1980], (pp. 375-402).

Lambek J., Scott P.J. [1974] "Aspects of higher order categorical logic" **Contemporary Mathematics** 30 (145-174).

Lambek J., Scott P.J. [1986] **Introduction to higher order Categorial Logic**, Cambridge University Press.

Longo G., Moggi E. [1984a] "Cartesian Closed Categories of Enumerations and effective Type Structures," **Symposium on Semantics of Data Types** (Khan, MacQueen, Plotkin eds.), LNCS 173, Springer-Verlag, (pp. 235-247).

Longo G., Moggi E. [1986] "Type-structures, principal morphisms, Combinatory Algebras: a category-theoretic characterization of functional completeness," Dip. di Informatica, Pisa *(prelim. version in* **Math. Found. Comp. Sci.**, Prague 1984 (Chytil, Koubek eds.) LNCS 176 , Springer-Verlag, 1984 (pp. 397-406)).

Martini S. [1986] "Categorical models for typed and type-free non-extensional lambda-calculus and Combinatory Logic" Dip. Informatica, Pisa.

McCarthy D.[1984] "Realizability and Recursive mathematics" Ph. D. Thesis, **Merton College**, Oxford.

McCracken N. [1984] "A finitary retract model for the polymorphic lambda-calculus," **Information and Control** (to appear).

Minc G.E. (G.E. Mints) [1977] "Closed categories and the theory of proofs," Translated from Zapiski Nauchnykh Seminarov Leningradskogo Otdeleniya Mat. Instituta im. V.A. Steklova AN SSSR **68** (pp. 83-114).

Mitchell J., Moggi E. [1987] "Kripke-style models for typed lambda caculus" **LICS 87**, Cornell.

Moggi E. [1986Tp] Message of Jan. 1986 on **Type e-mailing List**.

Moggi E.[1987] Ph. D. Thesis, Edinburgh, in preparation.

Moggi E. [1986] "Partial Morphisms in Categories of effective objects," **Info&Contr.**, to appear.

Obtulowicz A. [1986] **The logic of categories of partial functions and its applications** Dissertationes Mathematicae CCXLI, Warsaw.

Obtulowicz A. [1985] "Algebra of constructions I", **Information and Control** (to appear)

Obtulowicz A. , Wiweger A. [1982] "Categorial, Functorial and algebraic aspects of the type-free lambda-calculus" Universal Algebra and Applications. Banach Center Publications vol. 9, (399-422) **PWN**-Polish Scientific Publishers, Warszawa.

Panangaden P., Schwartzbach M.I. [1985] "Categorical Type Theory". Computer Science Department Technical Report, TR 85-716. Cornell University, Ithaca, NY, 1985.

Plotkin G. [1978] "Tω as a universal domain," **J. Comp. Syst. Sci.**, 17 (pp. 209-236).

Reynolds J. [1984], "Polymorphism is not set-theoretic," **Symposium on Semantics of Data Types**, (Kahn, MacQueen, Plotkin, eds.) LNCS 173, Springer-Verlag

Rosolini G. [1986] "Continuity and efffectiveness in Topoi" D. Phil. Thesis, **Oxford University**.

Robinson E., Rosolini P. [1987] "Categories of partial maps" Math. Dept., Univ. of Cambridge.

Scott D. [1972] "Continuous lattices" **Toposes, algebraic Geometry and Logic**, (Lawere ed.), SLNM 274, (pp.97-136) Springer-Verlag.

Scott D. [1980] "Relating theories of the lambda-calculus," *in* Hindley/Seldin[1980].

Scott D. [1980b] "A space of retracts" Manuscript, Bremen.

Scott P.J. [1978] "The "dialectica" interpretation and categories," **Zeitschr. f. Math. logik und Grundlagen d. Math.** 24 (pp. 553-575).

Seely R.A.G. [1984] "Locally cartesian closed Categories and type theory" **Math. proc. Cambridge Phil. Soc.**, 95, 33, pp. 33-48.

Seely R.A.G. [1986] "Categorical semantics for higher order polymorphic lambda calculus", **JSL** (to appear).

Smyth M., Plotkin G. [1982] "The category-theoretic solution of recursive domain equations" **SIAM Journal of Computing** 11, (pp.761-783).

Wand M. [1979] " Fixed-point Constructions in Order-enriched Categories," **Theoret. Comp.**

Sci. (pp. 13-30).

Wiweger A. [1984] "Pre-adjunctions and lambda-algebraic theories" **Coll. Math. XLVIII**, (153-165), Warszava.

Part III

Aczel P [1980] "Frege structures and the notions of proposition, truth and set." *In* **The Kleene Symposium**, ed. J. Barwise et al., Northe-Holland, pp. 31-59.

Allen S.F. [1987] "The Semantics of Type Theoretic Languages" Doctoral Dissertation, Computer Science Department, Cornell University March 1987, (expected).

Andrews P.B. [1971] "Resolution in type theory". **J. Symbolic Logic**, v.36 pages 414-432.

de Bruijn N.G. [1980] "A survey of the project AUTOMATH" In **Essays in Combinatory Logic, Lambda Calculus, and Formalism**", J.P. Seldin and J.R. Hindley, eds. Academic Press, New York, 1980, pages 589-606.

Bunder M.W.V. [1983] "A weak absolute consistency proof for some systems of illative combinatory logic," **JSL** 48, pp. 771-76

Church A. [1940] "A formalisation of the simple theory of types" **JSL** 5, pp. 56-58.

Constable R.L. [1980] "Programs and types" **21st I.E.E.E. Symposium on Foundations of Comput. Sci**., pp 118-128.

Constable R.L., Knoblock T.B., and Bates J.L. [1984] "Writing programs that construct proofs." **J. Automated Reasoning**, v.1 n.3, (1984), pages 285-3236.

Constable R. L. et al. [1986] **Implementing Mathematics with the Nuprl Proof Development System**. Prentice-Hall.

Coquand T. [1985] "Une théorie des constructions", Thèse de 3ème cycle, Université Paris VII.

Coquand T. [1986] "An analysis of Girard paradox", **LICS 86**, Boston.

Coquand T., Huet G. [1985] "Constructions: a higher order proof system for mechanizing mathematics" Report 401 INRIA, presented at **EUROCAL 85**.

Curry H. B., [1930] "Grundlagen der kombinatorischen Logik" **Amer. J. Math.** 52 (pp. 509-536,
789-834).

Girard J.Y. [1971] " Une extension de l'interpretation de Gödel a l'analyse, et son application a l'elimination des coupures dans l'analyse et la theorie des types". In **2nd Scandinavian Logic Simposium**, J.E. Festand, ed. North-Holland, Amsterdam, 1971, pages 63-92.

Girard, J. [1972] "Interpretation fonctionelle et elimination des coupure dans l'arithmetic d'ordre superieur," These de Doctorat d'Etat, Paris.

Gödel K. [1958] "Ueber eine bicher noch nicht benuetze Erweiterung des finiten Standpuntes," **Dialectica,** vol.12, pp.280-287.

Harper R., Honsell F., Plotkin G. [1987] "A framework for defining logics" **LICS 87**, Cornell.

Howard W. [1980] "The formulas-as-types notion of construction", in Hindley and Seldin 1980 pp.479-490. (MS written in 1969).

Huet G. [1986] "Formal Structures for Computation and Deduction" Lecture Notes, **C.M.U.**.

Knoblock T.B., Constable R.L. [1986] "Formalized metareasoning in Type Theory". In **Proceedings of Symposium on Logic in Computer Science**, 1986, pages 237-248.

Lambeck J., Scott P.J. [1980] "Intuitionist type theory and the free topos," **J. Pure Appl. Algebra 19** (215-257).

Martin-Löf P. [1971] "A theory of types," **Report 71-3**, Dept of Mathematics, Unibversity of Stockholm, February 1971, revised October 1971.

Martin-Löf P. [1975] "An intuitionistic theory of types" **Logic Colloqium 73**, Rose Shepherdson (Eds.), North-Holland (73-118).

Martin-Löf [1982] "Constructive logic and computer programming," *In* **Logic**, **Methodology and Philosophy of Science VI**, ed. L.J. Cohen et al. (eds.) North-Holland (pp. 153-175).

Martin-Löf P. [1984] **Intuitionistic Type Theory** Bibliopolis, Napoli.

McCarthy D.[1984] "Realizability and Recursive mathematics" Ph. D. Thesis, **Merton College**, Oxford.

Miller D. [1984] "Automating Higher Order Logic". PhD Thesis, Canergie-Mellon University.

Nederpelt R.P. [1980] "An approach to theorem proving on the basis of a typed lambda calculus," **SLNCS** 87, (pp.182-194).

Nordstrom B. [1981] "Programming in constructive set theory: some examples". In **Proceedings 1981 Conference on Functional Programming Languages and Computer Architecture**. Portsmouth, England, pages 141-153.

Petersson K. [1982] "A programming system for type theory" Chalmers University, Göteborg.

Rezus A. [1986] "Impredicative Type Theories" Informatica, Univ. Nijmegen.

Seldin J.P. [1978] "A sequent calculus formulation of type assignment with equality rules for the λβ-calculus," **JSL** 43, (pp.643-649).

Seldin J.P. [1980] " Curry's program," *In* Hindley and Seldin 1980, (pp.3-34).

Stenlund S. [1972] "Combinators, λ-terms, and Proof Theory". D. Reidel, Dordrecht, The Netherlands, 1972.

References

Amadio R., Longo G. [1986] "Type-free compiling of parametric Types" IFIP Conference on **Formal description of Programming Concepts** Ebberup (DK), North Holland, 1987 (to appear).

Backus J. (1978) "Can Programming be liberated of the Von Neumann style? A functional style and its algebra of programs" CACM 21, 8 (613-641).

Barendregt, H. [1984], **The lambda calculus; its syntax and semantics**, Revised and expanded edition, North Holland.

Barendregt H., Coppo M., Dezani M. [1983] "A filter lambda model and the completeness of type assignment," **J. Symb. Logic 48,** (931-940).

Barendregt H., Longo G. [1980] "Equality of lambda terms in the model T^{ω}," *in* **To H.B. Curry: Essays in Combinatory Logic, Lambda-calculus and Formalism**, (Seldin, Hindley eds.) Academic Press, London, (pp. 303-337).

Beeson M.[1980] **Foundations of Constructive Mathematics**, Springer- Verlag.

Berry G., [1978] "Stable models of typed λ–calculi". **SLNCS** 62, pp.72-89.

Berry G., Curien P.L. [1982] "Sequential Algorithms on Concrete Data Structures" **Theor. Comp. Sci. 20,** (265-321).

Breazu-Tannen V., Meyer A. [1987] "Polymorphism is conservative over simple types" **2nd IEEE LICS**, Cornell.

Burstall M.R., Lampson B. [1984] "A kernel language for abstract data types and modules". In **Semantics of Data Types, LNCS**, Vol. 173 Springer-Verlag, New York, 1-50.

Cardelli L. [1983] "The functional abstract machine" **Polymorphism: The ML/LCF/Hope Newsletter** I.

Cardelli L., Wegner P. [1985] "On understanding types, data abstraction, and polymorphism",

ACM Comp. Surveys, 17, 4.

Church A. [1932-3] "A set of partulates for the Foundation of Logic" **Annals of Math.** XXXIII (348-349) and XXXIV (839-864).

Church A., [1940] "A formalisation of the simple theory of types" **J. Symb. Logic** 5, pp. 56-58.

Church A. [1941] **The Calculi of Lambda Conversion**, Princeton Univ. Press, reprinted 1963 by University Microfilms Inc., Ann Arbor, Michigan, U.S.A.

Church A., Rosser J.[1936] "Some properties of conversion", **Trans. A.M.S.** 39 (472-482).

Constable R.L. and Zlatin D.R. [1984] "The type theory of PL/CV3". **ACM Trans. Prog. Lang. Sys.**, v.7, n.1 pages 72-93.

Coppo, M. [1984] "Completeness of type assignment in continuous lambda-models," **Theor. Comp. Sci**. 29 (309-324).

Coppo M., Dezani M. [1979] "A new type assignment for lambda terms," **Archiv Math. Logik 19,** (139-156).

Coppo M., Dezani M., Honsell F., Longo G. [1984] "Extended Type structures and filters lambda models," **Logic Colloquium 82** (Lolli, Longo, Marcja eds.), North-Holland, Studies in Logic 112, (241-262).

Cousineau G., Curein P.L., Mauny M. [1985] "The categorical Abstract Machine" LITP, CNRS-Paris 7, Janvier 1985.

Curry H. B., Hindley J. R., Seldin J. [1972] **Combinatory Logic vol. II**, North-Holland.

Damas A. [1985] "Tye disciplines in Programming Languages" Ph. D. Thesis, Comp. Sci. Dept., Edinburgh.

Girard J.Y.[1986] Book on Proof Theory in preparation for Bibliopolis.

Girard J.Y.[1987] "Linear Logic", **TCS**, to appear.

Gordon M., Milner R., Wadsworth C.[1979] **Edindurgh LCF**, LNCS 78, Springer-Verlag.

van Heijenoort J. (ed.) [1967] "From Frege to Gödel," Harvard University Press.

Hindley, R. [1969] "The principal type-scheme of an object in Combinatory Logic," **Trans. A.M.S.**, 146 (22-60).

Hindley, R. [1983] "The completeness theorem for typing lambda-terms," **Theor. Comp. Sci.** 22 1-17 (also **TCS** 22, pp. 127-133).

Hindley R., Longo G. [1980] "Lambda-calculus models and extensionality," **Zeit. Math. Logik Grund. Math.** n. 2, vol. 26 (289-310).

Hindley R., Seldin, J. (eds.) [1980] **To H.B. Curry: Essays in Combinatory Logic, Lambda calculus and formalism**, Academic Press.

Hindley R., Seldin J. [1986] **Introduction to Combinators and Lambda-Calculus**, London Mathematical Society.

Hyland M. [1976] "A syntactic characterization of the equality in some models of lambda calculus" **J. London Math. Soc. 2, 12.**

Jones N.[1980] (ed.) **Semantics-Directed Compiler Generation**, LNCS 94, Springer-Verlag.

Johnstone P. [1977] **Topos Theory**. Academic Press.

Leivant, D. [1983] "Polymorphic type inference", **1oth ACM Symposium on Principles of Prog. Langs.**, publ. ACM, (pp.88-98).

Longo, G. [1983] "Set-Theoretical Models of Lambda-calculus: Theories, Expansions, Isomorphisms," **Annals Pure Applied Logic** vol. 24, (153-188).

Longo, G. [1984] "Continuous structures and analytic methods in Computer Science," Lecture delivered at **Coll. on Trees in Algebra and Programming**, Bordeaux (Courcelle ed.) Cambridge University Press, 1984 (pp. 1-22).

Luke-Ong[1987] Ph. D. Thesis, Comp. Sci. Dept., Imperial College, London (in preparation).

MacQueen D.B. [1986] "Using Dependent Types to Express Modular Structure". In **13th ACM Symposium on Principles of Programming Languages**.

Meyer A. R., Mitchell J., Moggi E., Statman R. [1987] "Empty types in polymorphic lambda calculus" (ACM Conference on) **POPL '87** , Münich.

Meyer, A. R. Reinhold, M.B.[1986] "*Type* is not a type" , **Proc. Popl 86**, ACM.

Milner R. [1978] "A theory of type polymorphism in programming," **Journal of Computer and Systems Sci.**, 3 (348-375).

Milner, R. [1986] "Is computing an experimental Science?" Inaugural lecture for the LFCS, Edinburgh.

Mitchell, J. C. [1984] "Type inference and type containment", **Symposium on Semantics of Data Types** (Kahn, MacQueen, Plotkin eds.),SLNCS 173, Springer-Verlag (257-278).

Mitchell J. C., Plotkin G. [1985] "Abstract types have exixtential types" Proc. **POPL 85**, ACM (ACM - TOPLAS, to appear).

Mulmuley K. [1985] "Full abstraction and semantic equivalence" Ph. D. Thesis, Comp. Sci. Dept., C.M.U..

Odifredi P.G. [1986] "Church's Thesis: The extent of Recursiveness in The Universe and Man" Dip. di Informatica, Torino.

Odifredi P.G. [1987] **Classical Recursion Theory**, North-Holland.

Reynolds J. [1974], "Towards a theory of type structures," **Colloque sur la Programmation**, LNCS 19, Springer-Verlag (pp. 408-425).

Reynolds J.C. [1985] "Three approaches to type structure," **SLNCS** 185, (pp. 145-146).

Scott D. [1980a] "Lamda-calculus, some models, some philosophy," **The Kleene Symposium** (Barwise et al. eds.) North-Holland.

Statman R. [1979] "Intuitionistic propositional Logic is Polynomial time complete" **Theor. Comp. Sci.** 9, pp. 67-72.

Statman R. [1980] "Completeness, invariance and lambda-definability," **J. Symb. Logic** 47, 1 , (pp. 17-26).

Statman R. [1985] "Equality between functionals revisited" *in* **H. Friedman's research on the found. of Math.** (Harrington et al. eds), North Holland.

Troesltra A.S. [1973] " Notes in intutionistic second order atithmetic," Cambridge Summer School in Math Logic, Springer **LNM** 337 (pp. 171-203).

Troelstra A. [1973] **Metamathematical investigation of Intuitionistic Arithmetic and Analysis**. **LNM 344** , Springer-Verlag, Berlin.

Troelstra A.S. and van Dalen D. (eds) [1982] **The L.E.J. Brouwer Centenary Simposium**, Studies in Logic 110, North-Holland.

Turner R. [1984] **Logics for Artificial Intelligence**. Halsted Press (John Wiley & Sons), New York, 1984.

Wadsworth C.P. [1976] " The relation between computational and denotational properties for Scott's D_∞-models of the lambda-calculus," **S.I.A.M. J. Computing** 5 (pp.488-521).