Notre Dame Journal of Formal Logic
Volume 21, Number 4, October 1980

An Extension of the Basic Functionality Theory for the λ-Calculus

M. COPPO and M. DEZANI-CIANCAGLINI

1 Introduction The first works about the assignment of types to terms of the λ-calculus (or combinatory logic) arose in the context of logical theories of types.* Church [2] presented a first-order system with types based on λ-conversion, and since then many systems of this kind have been proposed; a review of them is given in the introduction of [15]. The problem of adjoining new objects and deduction rules to combinatory logic is faced in a more general way in [6] and [7], since Curry is interested in studying the properties of illative systems to analyze their suitability as a framework for the study of the foundations of logic. This leads him to introduce many different systems in which the new objects and rules are interesting for different aspects of combinatory logic. As Curry points out ([6], p. 256), the introduction of an illative system is always associated with some interpretation of terms.

The first and simplest of the illative systems proposed in [6] is the theory of basic functionality which is suggested by the very natural interpretation of terms as functions from terms to terms. In this system terms are associated (through a set of formal axioms and deduction rules) with functional characters or types. Types are built from a set of basic objects (which are left uninterpreted) by a composition operator F, which builds a new type $F\sigma\tau$ from two types or basic objects σ and τ. If a term has type $F\sigma\tau$ then we can interpret it as a function from terms of type σ to terms of type τ.

Functionality theory is interesting in the foundational program of Curry essentially since the constant F can very naturally be interpreted as implication. In this case a typed term can be seen as the representation of a proof in a natural deduction system.

*The authors are very grateful to Prof. J. P. Seldin for many useful suggestions to improve the results of this paper.

Received June 23, 1978; revised December 21, 1979

The notion of functionality is also interesting if we look at λ-calculus in the field of theoretical computer science. λ-calculus, in fact, can be considered as a programming language (see, for example, [9]) which supplies a natural formal framework for studying a very general class of programming concepts (such as the notion of a fixpoint of a functional or of self-application). Many of the most recent results on programming languages are more or less derived from the study of λ-calculus, especially the development of the mathematical base for denotational semantics, which started with the works of Scott [13], [14], Plotkin [10], and Wadsworth [17]. If we look at λ-calculus in this framework, the notion of the functional character of a term generalizes that of type in programming languages.

From this and, perhaps, other points of view, however, Curry's theory is not quite satisfactory since there are many terms which don't possess any functional character, even if we assume, with Church, that only terms with normal forms are suitable to possess some kind of interpretation. Types, in fact, are not preserved by conversion (unless we postulate it explicitly as Curry does in some systems) and not even all terms in normal form have a type. The reasons the theory of basic functionality fails to be truly satisfactory may be found both in the structure given to types and in the system of type assignment rules (which heavily depend on type structure).

Many recent theories (see the introduction of [15] for a review of them) have generalized functionality theory by modifying both type structure and type assignment rules, leading to more general systems in which, however, the functional characters assigned to terms often become quite complex. A feature of all these theories is that types are considered, more or less explicitly, as functions from terms to types.

In this paper we present a generalization of the basic functionality theory which is achieved simply by defining a more powerful rule for type formation and modifying accordingly the deduction rules. The main idea is to define from any arbitrary finite set of types $\{\tau_1, \ldots, \tau_n\}$ a "sequence" $[\tau_1, \ldots, \tau_n]$ whose underlying set of terms can be interpreted as the intersection of those of $\tau_1, \ldots, \tau_n$. This leads to a system in which, leaving the assignment rules substantially unchanged, the set of typed terms is much larger than in the basic functionality theory (in particular, all strongly normalizable terms possess at least one type). This fact can be intuitively explained by observing that the introduction of sequences leads to a more refined classification of terms which allows us to define, for any arbitrary term, a proper domain (if it exists) into which it is total. The consistency of the present system is given by the proof that every typed term reduces to a normal form (Section 3). In Section 4 we give a characterization of the set of typed terms which turns out to properly include the set of stratified terms (in the sense of [6], Ch. 9). As an immediate consequence of this characterization it turns out that, in the λ-I-calculus, all convertible terms have the same set of types. This restriction is not substantial if we are interested in studying functionality, since all Turing computable functions can be represented inside the λ-I-calculus.

The idea of extending basic functionality theory by allowing types to represent more refined categories was first introduced, implicitly, in the system proposed in [4]. There we introduce a partial-order relation between types

such that the class of terms with type τ includes any class of terms with type greater than τ. This inclusion can be seen as the weakest version of the intersection defined by sequences of types.

2 Formalization of the theory The set of types is defined inductively from a set of basic types $\{\varphi_1, \varphi_2, \ldots\}$ by an operator of composition which generalizes the object F of Curry's theory. In fact, while F builds a new type from two types, this operator (let us call it F' for the moment) builds a new type from a type and a set of types, i.e., if τ is a type and $[\sigma_1, \ldots, \sigma_n]$ $(n \geqslant 1)$ is a set of types (*sequence*) then $F'[\sigma_1, \ldots, \sigma_n]\tau$ is a type. $F'[\sigma_1, \ldots, \sigma_n]\tau$ is the type of a term which maps terms that have *all* types $\sigma_1, \ldots, \sigma_n$ into terms of type τ. We shall abbreviate this notation by omitting F' (without danger of confusion), i.e., we will write $[\sigma_1, \ldots, \sigma_n]\tau$ instead of $F'[\sigma_1, \ldots, \sigma_n]\tau$.

Definition 1 The set of types is defined as follows:

a. each basic type is a type
b. if $\sigma_1, \ldots, \sigma_n, \tau (n \geqslant 1)$ are types then $[\sigma_1, \ldots, \sigma_n]\tau$ is a type.

Example 1: $\tau \equiv [\varphi_1, \varphi_2, \varphi_3][[\varphi_1, \varphi_2]\varphi_2, \varphi_1]\varphi_3$ is a type.

We will write $[\bar{\sigma}]$ instead of $[\sigma_1, \ldots, \sigma_n]$ $(n \geqslant 1)$ whenever possible. Sequences are intended as sets of types, so they are independent of the order in which the types are written and of the number of occurrences of each type. More formally, we consider sequences modulo the following two equivalence relations:

$$[\sigma_1, \ldots, \sigma_i, \sigma_{i+1}, \ldots, \sigma_n] = [\sigma_1, \ldots, \sigma_{i+1}, \sigma_i, \ldots, \sigma_n]$$
$$[\sigma_1, \ldots, \sigma, \sigma, \ldots, \sigma_n] = [\sigma_1, \ldots, \sigma, \ldots, \sigma_n]$$
$$(0 < i \leqslant n).$$

Using set theoretic notation we write $\tau \in [\bar{\sigma}]$ if τ is a type in $[\bar{\sigma}]$ and $[\bar{\sigma}] \cup [\bar{\tau}]$ to denote the sequence which is the union of $[\bar{\sigma}]$ and $[\bar{\tau}]$. The composition operator builds a new type from a type and a set of n types $(n \geqslant 1)$. Obviously, when $n = 1$, $[\sigma_1]$ is the set of one type, and not a type.

We give now the T-formulation (see [6], p. 315) of our system of type assignment. A *basis* is a set of statements of the form $[\bar{\sigma}]x$, where $[\bar{\sigma}]$ is a sequence and x a variable (1), which functions as axioms for a deduction. This notation is very similar to that of [15] (appendix to Section A) except that here we assign sequences instead of types to variables. This allows the assignment of an arbitrary number of types (possibly only one) to the same variable. Moreover, it is not restrictive to assume, in a given basis, that only one sequence can be assigned to the same variable. We will do it in order to simplify notation.

If $\mathscr{B}$, $\mathscr{B}'$ are bases then $\mathscr{B} \cup \mathscr{B}'$ is the basis obtained by assigning to each variable x the sequence $[\bar{\sigma}] \cup [\bar{\sigma}']$, where $[\bar{\sigma}]$ and $[\bar{\sigma}']$ are the sequences assigned to x in $\mathscr{B}$ and $\mathscr{B}'$ respectively (obviously $[\bar{\sigma}]([\bar{\sigma}'])$ is empty if x does not occur in $\mathscr{B}(\mathscr{B}')$). Basis inclusion (denoted $\mathscr{B} \subseteq \mathscr{B}'$) is defined similarly.

$\mathscr{B} \vdash \sigma X$ ($\mathscr{B} \vdash [\bar{\sigma}]X$) means that from $\mathscr{B}$ we can deduce $\sigma X([\bar{\sigma}]X)$. In particular, if $\mathscr{B}$ is empty, we write $\vdash \sigma X(\vdash [\bar{\sigma}]X)$. Moreover if $\mathscr{B} \vdash \sigma X$ and $\mathscr{B} \subseteq \mathscr{B}'$, then $\mathscr{B}' \vdash \sigma X$.

The axiom scheme and rules of our system are:

Axiom (Ap′) *If $\mathcal{B}$ contains $[\bar{\sigma}]x$ and $\tau \in [\bar{\sigma}]$ then $\mathcal{B} \vdash \tau x$.*

Rule (Fe′). If $\mathcal{B} \vdash [\bar{\sigma}]\tau X$ and $\mathcal{B} \vdash [\bar{\sigma}]Y$ then $\mathcal{B} \vdash \tau(XY)$.

Rule (Fi′). If $\mathcal{B}, [\bar{\sigma}]x \vdash \tau X$ (so x does not occur in $\mathcal{B}$) then $\mathcal{B} \vdash [\bar{\sigma}]\tau\lambda x.X$.

Rule (Seq). If $\mathcal{B} \vdash \sigma_i X$ for $1 \leqslant i \leqslant n$ then $\mathcal{B} \vdash [\sigma_1, \ldots, \sigma_n]X$.

Axiom Ap′ and Rules Fe′ and Fi′ are straightforward generalizations of Ap, Fe, Fi of [6], p. 317. Rule Seq allows the assignment of sequences to terms (while Ap′, Fe′, and Fi′ allow only assignment of types to terms).

Notice that, owing to the presence of sequences, in a deduction many different types can be assigned to the *same* occurrence of a component of a term X. This is not true in classical functionality theories where each component of X has exactly one type.

Example 2: We show one deduction of $\vdash [\varphi]\varphi(\lambda x.xx)\lambda y.y$.

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{}{.\not\in [[[\varphi]\varphi][\varphi]\varphi,\ [\varphi]\varphi]x}}{\dfrac{a}{[[\varphi]\varphi][\varphi]\varphi x}\,\mathrm{Ap}'\quad \dfrac{\dfrac{a}{[\varphi]\varphi x}\,\mathrm{Ap}'}{[[\varphi]\varphi]x}\,\mathrm{Seq}}\,\mathrm{Fe}'}{[\varphi]\varphi(xx)}\,\mathrm{Fi}'-\{a\}}{[[[\varphi]\varphi][\varphi]\varphi,\ [\varphi]\varphi][\varphi]\varphi\lambda x.xx}\qquad \dfrac{\dfrac{\dfrac{}{.\not\in [[\varphi]\varphi]y}}{\dfrac{b}{[\varphi]\varphi y}\,\mathrm{Ap}'}\,\mathrm{Fi}'-\{b\}}{[[\varphi]\varphi][\varphi]\varphi\lambda y.y}\qquad \dfrac{\dfrac{}{.\not\in [\varphi]y}}{\dfrac{c}{\varphi y}\,\mathrm{Ap}'}\,\mathrm{Fi}'-\{c\}}{[\varphi]\varphi\lambda y.y}\,\mathrm{Seq}}{[[[\varphi]\varphi][\varphi]\varphi,\ [\varphi]\varphi]\lambda y.y}\,\mathrm{Fe}'}{[\varphi]\varphi(\lambda x.xx)\lambda y.y}$$

It is easy to verify that $(\lambda x.xx)\lambda y.y \geqslant \lambda y.y$. Notice that this term is not stratified in the sense of [6].

It is clear that when X is not a single variable $\mathcal{B} \vdash \tau X$ implies that some types can be assigned to the (immediate) components of X. More precisely:

if $X \equiv YZ$ then there exists a sequence $[\bar{\sigma}]$ such that $\mathcal{B} \vdash [\bar{\sigma}]\tau Y$ and $\mathcal{B} \vdash [\bar{\sigma}]Z$.

if $X \equiv \lambda x.Y$ and $\tau \equiv [\bar{\sigma}]\rho$ then $\mathcal{B}, [\bar{\sigma}]x \vdash \rho Y$.

Moreover, if $[\bar{\sigma}] \equiv [\sigma_1, \ldots, \sigma_n]$, $\mathcal{B} \vdash [\bar{\sigma}]X$ implies $\mathcal{B} \vdash \sigma_i X$ for all i, $1 \leqslant i \leqslant n$.

3 Main properties In this section we prove that types are preserved by reduction (Theorem 1) and that each term which possesses a type reduces to a normal form (Theorem 2). Lemmas 1 and 2 present more technical results. As a corollary of Theorem 2 we obtain a new proof of the Normal Form Theorem for the basic functionality theory of [6], p. 340. The results of this section will be proved for statements of the form $\mathcal{B} \vdash \sigma X$, where σ is a type. It is trivial to extend them to statements of the form $\mathcal{B} \vdash [\bar{\sigma}]X$, where $[\bar{\sigma}]$ is a sequence.

Lemma 1 *If R is a redex and R' its contractum, then $\mathcal{B} \vdash \tau R$ implies $\mathcal{B} \vdash \tau R'$.*

Proof: Let $R \equiv (\lambda x.Y)Z$. Then there exists a sequence $[\bar{\sigma}] \equiv [\sigma_1, \ldots, \sigma_n]$ such that we can deduce $\mathcal{B} \vdash \tau R$ from $\mathcal{B} \vdash [\bar{\sigma}]\tau\ \lambda x.Y$ and $\mathcal{B} \vdash [\bar{\sigma}]Z$ (x does not occur in $\mathcal{B}$). Moreover $\mathcal{B} \vdash [\bar{\sigma}]\tau\ \lambda x.Y$ only if $\mathcal{B}, [\bar{\sigma}]x \vdash \tau Y$. By definition $R' \equiv Y[x/Z]$. To obtain $\mathcal{B} \vdash \tau Y[x/Z]$ it is sufficient to replace, in the proof of

$\mathscr{B}$, $[\bar{\sigma}]x \vdash \tau Y$ the statements $\sigma_i x$ (introduced by Ap′) by the deductions of $\mathscr{B} \vdash \sigma_i Z$ for $1 \leqslant i \leqslant n$.

Theorem 1 *If $X \geqslant X'$ and $\mathscr{B} \vdash \sigma X$ then $\mathscr{B} \vdash \sigma X'$.*

Proof: It is clearly sufficient to prove the theorem when X' differs from X by the contraction of only one redex. Let R be such a redex and R' its contractum. The proof of $\mathscr{B} \vdash \sigma X'$ can be obtained from that of $\mathscr{B} \vdash \sigma X$ by replacing any deduction of a type for R by the corresponding deduction of the same type for R' (obtained as in the proof of Lemma 1).

To prove Theorem 2 we associate to each deduction $\mathscr{D}$ of a statement $\mathscr{B} \vdash \sigma X$ an element (the *measure* of $\mathscr{D}$) of a well-ordered set (in our case the set of pairs of integers). We then show (Lemma 2) that if X is not in normal form, there exists some suitable redex R such that $X \geqslant X'$ by the contraction of R and there is a deduction $\mathscr{D}'$ of $\mathscr{B} \vdash \sigma X'$ (obtained from $\mathscr{D}$ as in the proofs of Lemma 1 and Theorem 1) such that the measure of $\mathscr{D}'$ is strictly lower than that of $\mathscr{D}$. The proof of Theorem 2 is an immediate consequence of this result. To introduce the notion of measure we need some technical definitions.

The *length* $\|\tau\|$ of a type τ is defined as the number of occurrences of basic types in it. Now let $\mathscr{D}$ be a deduction of $\mathscr{B} \vdash \sigma X$ and $R \equiv (\lambda x.Y)Z$ the occurrence of a redex in X. The *characteristic set* of R in $\mathscr{D}$ is the set $\mathscr{C}(R)$ of all types $[\bar{\mu}]\nu$ assigned to the occurrences of $\lambda x.Y$ in $\mathscr{D}$, i.e., $\mathscr{C}(R) = \{[\bar{\mu}]\nu \mid [\bar{\mu}]\nu$ is a type of $\lambda x.Y$ in $\mathscr{D}\}$. The *height* $h(R)$ of R is the maximum length of the types of $\mathscr{C}(R)$, i.e., $h(R) = \max\{\|\tau\| \mid \tau \in \mathscr{C}(R)\}$. The *measure* of a deduction $\mathscr{D}$ is the pair of nonnegative integers $\langle m(\mathscr{D}), n(\mathscr{D})\rangle$ defined by:

$$m(\mathscr{D}) = \text{maximum height of redexes in } \mathscr{D}$$
$$n(\mathscr{D}) = \text{number of redexes with height } m(\mathscr{D}) \text{ in } \mathscr{D}.$$

We intend $m(\mathscr{D}) = 0$ if there are no redexes in X and $n(\mathscr{D}) = 0$ if $m(\mathscr{D}) = 0$. Integer pairs can be ordered by the usual lexicographic order relation ($\leqslant$):

$$\langle h', k'\rangle \leqslant \langle h, k\rangle \text{ iff } h' < h \text{ or, } h' = h \text{ and } k' \leqslant k.$$

Example 3: If $R \equiv (\lambda x.xx)\lambda y.y$ and we consider the deduction $\mathscr{D}$ of Example 2 then: $\mathscr{C}(R) = \{[[[\varphi]\varphi][\varphi]\varphi, [\varphi]\varphi][\varphi]\varphi\}$, $h(R) = 8$, $m(\mathscr{D}) = 8$, and $n(\mathscr{D}) = 1$.

Lemma 2 *Let $\mathscr{D}$ be any deduction of $\mathscr{B} \vdash \sigma X$ where X is a term not in normal form. Then there is a term X' and a deduction $\mathscr{D}'$ of $\mathscr{B} \vdash \sigma X'$ such that $X \geqslant X'$ and the measure of $\mathscr{D}'$ is lower than that of $\mathscr{D}$.*

Proof (constructive): Choose a redex $(\lambda x.Y)Z$ in X such that:

i. its height is $m(\mathscr{D})$
ii. no redex with height $m(\mathscr{D})$ occurs in it.

It is obvious that in X there exists always at least one redex which satisfies these conditions. Let $\mathscr{C}((\lambda x.Y)Z) = \{[\bar{\mu}_i]\nu_i \mid 0 < i \leqslant n\}$. Let X' be obtained from X by contracting $(\lambda x.Y)Z$. If we consider the deduction $\mathscr{D}'$ of $\mathscr{B} \vdash \sigma X'$ obtained as in the proofs of Lemma 1 and Theorem 1, we observe that:

(1) all redexes in X' which are residuals of some redex in X have, in $\mathscr{D}'$, the same characteristic sets as in $\mathscr{D}$. Moreover the redexes of X which have more than one residual in X' have heights lower than $m(\mathscr{D})$, by (ii).

(2) the new redexes in X' (i.e., those which are not residuals of any redex in X) have characteristic sets which contain either types belonging to $\bigcup_{i=1}^{n} [\bar{\mu}_i]$ (if they are redexes of the form ZV for some component V of $Y[x/Z]$) or types belonging to $\{\nu_1, \ldots, \nu_n\}$ (if $(\lambda x.Y)Z$ is applied, in X, to some component W and $Y[x/Z]W$ is a redex in X'). In both cases these redexes have heights lower than $m(\mathcal{D})$.

Therefore X contains exactly one redex of height $m(\mathcal{D})$ more than X'. This means that either $m(\mathcal{D}') < m(\mathcal{D})$ (if $(\lambda x.Y)Z$ was the only redex of height $m(\mathcal{D})$) or $m(\mathcal{D}') = m(\mathcal{D})$ and $n(\mathcal{D}') < n(\mathcal{D})$.

Theorem 2 *If X is an arbitrary term and $\mathcal{B} \vdash \sigma X$ (for some basis $\mathcal{B}$ and type σ) then X reduces to a normal form.*

Proof: By transfinite induction on the measure of $\mathcal{D}$, where $\mathcal{D}$ is a deduction of $\mathcal{B} \vdash \sigma X$, using Lemma 2. We notice that, if the measure of $\mathcal{D}$ is $\langle 0, 0 \rangle$, then X does not contain redexes, i.e., it is a normal form.

Notice that, thanks to the Church-Rosser property, the result of Theorem 2 is independent of the order in which we execute the reductions (although the proof is not). Moreover, since $\mathbf{B} \vdash \sigma X$ is deducible only if we can associate a type to each component of X, Theorem 2 implies that each component of X has a normal form too (i.e., X is strongly normalizable). Since the type assignment of basic functionality theory, when we force the bases to contain only assignment to variables, is a particular case of the present theory, we obtain immediately as a corollary of Theorem 2 the Normal Form Theorem of [6], p. 340, for bases which satisfy this restriction. It is not difficult, however, to use Theorem 2 to prove the Normal Form Theorem in its full formulation.[2]

4 Some characterization of typed terms In this section we prove some results which give a characterization of the set of terms which possess a type in our system. We do it first by proving (Theorem 3) that all terms in normal form possess a type and then by characterizing (Lemma 3) the class of all expansions which preserve types. Since all typed terms must possess a normal form (Theorem 2) this is enough for a characterization of typed terms. An immediate consequence of Lemma 3 is that, inside the λ-$\mathbf{I}$-calculus, all normalizable terms possess a type and types are preserved by convertibility (Theorems 4 and 5).

Theorem 3 *Each normal form N possesses at least one type for a suitable basis $\mathcal{B}$.*

Proof: By structural induction on N.
First step. If N is a free variable (say a) we have $[\bar{\sigma}]a \vdash \tau a$ for all sequences $[\bar{\sigma}]$ and types $\tau \in [\bar{\sigma}]$.
Inductive step. We split the proof according to two possible cases:

1. $N \equiv aN_1 \ldots N_m$ $(m > 0)$. By inductive hypothesis there exist some basis $\mathcal{B}_i$ and types σ_i such that $\mathcal{B}_i \vdash \sigma_i N_i (1 \leqslant i \leqslant m)$. Then if we choose $\mathcal{B} = \bigcup_{i=1}^{m} \mathcal{B}_i \cup \{[\sigma_1] \ldots [\sigma_m]\tau a\}$ where τ is any type, we obtain $\mathcal{B} \vdash \tau N$ by m applications of Fe′.

2. $N \equiv \lambda x.\overline{N}$. By inductive hypothesis there exist a basis $\mathcal{B}'$ and a type τ such that $\mathcal{B}' \vdash \tau\overline{N}$. If $\mathcal{B}' = \mathcal{B}, [\bar{\sigma}]x$, then we obtain, by Fi′: $\mathcal{B} \vdash [\bar{\sigma}]\tau N$. Otherwise, if $\mathcal{B}'$ does not assign a sequence to x, then x cannot occur in $\overline{N}$, so we have $\mathcal{B}', [\bar{\nu}]x \vdash \tau\overline{N}$ for any sequence $[\bar{\nu}]$ and by Fi′: $\mathcal{B}' \vdash [\bar{\nu}]\tau N$.

Lemma 3 *Let $R \equiv (\lambda x.Y)Z$ and $R' \equiv Y[x/Z]$. $\mathcal{B} \vdash \tau R'$ implies $\mathcal{B} \vdash \tau R$ iff there exists a sequence $[\bar{\sigma}]$ such that $\mathcal{B} \vdash [\bar{\sigma}]Z$.*

Proof: The "only if" part is obvious. In the "if" part we distinguish two cases, according to whether x occurs (Case 1) or not (Case 2) in Y (notice that Case 2 can occur only in the λ-**K**-calculus).

Case 1: We will consider only the occurrences of Z in $Y[x/Z]$ which replace occurrences of x in Y. Let $[\bar{\sigma}]$ be the collection of all types assigned to Z in one deduction of $\mathcal{B} \vdash \tau R'([\bar{\sigma}]$ is nonempty since x occurs in $Y)$. Then $\mathcal{B} \vdash [\bar{\sigma}]Z$. Moreover we can obtain $\mathcal{B}, [\bar{\sigma}]x \vdash \tau Y$ by replacing, in the proof of $\mathcal{B} \vdash \tau R'$, the deductions of $\mathcal{B} \vdash \rho Z$ (for each $\rho \in [\bar{\sigma}]$) by the statements ρx obtained by Ap′ from $[\bar{\sigma}]x$. So we have $\mathcal{B} \vdash [\bar{\sigma}]\tau\lambda x.Y$ and, by Fe′, $\mathcal{B} \vdash \tau R$.

Case 2: In this case $R' \equiv Y$. Then $\mathcal{B} \vdash [\bar{\nu}]\tau\lambda x.Y$ for all sequences $[\bar{\nu}]$ and from $\mathcal{B} \vdash [\bar{\sigma}]Z$ we obtain $\mathcal{B} \vdash \tau R$ by Fe′.

Theorem 4 *In the λ-**I**-calculus, if $X = X'$ and $\mathcal{B} \vdash \sigma X$ then $\mathcal{B} \vdash \sigma X'$.*

Proof: Immediate from Lemmas 1 and 3.

Theorem 5 *In the λ-**I**-calculus, if X is an arbitrary term then $\mathcal{B} \vdash \tau X$ (for some basis $\mathcal{B}$ and type τ) iff X reduces to a normal form.*

Proof: The "if" part is immediate from Theorems 3 and 4, and the "only if" part is immediate from Theorem 2.

Conclusion The present generalization of basic functionality theory turns out to be satisfactory in the λ-**I**-calculus in the sense that each solvable, i.e., normalizable (see [1]), term possesses a type. This is no longer true in the λ-**K**-calculus since, in this case, the class of solvable terms does not coincide with that of strongly normalizable terms. It is not at all clear, moreover, that the terms to which it can be interesting to assign a functional character are only the normalizable ones (for example, the fixed-point combinator Y plays an important role in the λ-definition of partial recursive functions). If we consider the λ-calculus as a language to describe functions, the results on models of the λ-calculus suggest that only unsolvable terms (in the sense of [17]) are completely meaningless. A further extension of the present theory, in which it is possible to assign meaningful functional characters to all solvable terms, can be obtained [5] by introducing a new object ω with the meaning of a universal category as the object E of Curry ([6], p. 240). This is done by Sallé [12], who generalizes the theory of [4]. Since ω is a universal category we have that, in the systems of [5] and [12], each term possesses at least type ω but only solvable terms turn out to possess types which carry information substantially greater than that of ω.

Lastly, let us examine briefly the behavior of types with respect to η-conversion. Owing to the existence of basic types, the present type assign-

ment is not invariant by η-expansion, since for example $\vdash[\varphi]\varphi\lambda x.x$ but $\nvdash[\varphi]\varphi\lambda xy.xy$ (when φ is any basic type). Moreover types are not invariant by η-reduction, and this is essentially due to the possibility of introducing, in sequences, types which are not "used" in the deduction. For example, $\vdash[[\sigma]\tau][\sigma,\rho]\tau\lambda xy.xy$ but $\nvdash[[\sigma]\tau][\sigma,\rho]\tau\lambda x.x$ for all types σ, ρ, τ. The invariance of types under η-convertibility is obtained, instead, in the systems presented in [4] and [12], thanks to some equivalence relations which are postulated between basic types.

NOTES

1. We make this last restriction since we are interesed in the λ-calculus without extra objects. It is not difficult, however, to introduce systems without this restriction.

2. J. P. Seldin [16] proves the Normal Form Theorem for the T-formulation of functionality theory taking into account the types which are assigned to redexes but in a way which is easier than that one done here. This is essentially because of the possibility of carrying over Prawitz' Normalization Theorem for classical logic [11]. This is done simply by thinking of types as formulas and $F\sigma\tau$ as $\sigma \supset \tau$ and by defining in a suitable way a reduction which corresponds to $\supset$-reduction.

REFERENCES

[1] Barendregt, H., "A characterization of terms of the λ-I-calculus having a normal form," *The Journal of Symbolic Logic,* vol. 38 (1973), pp. 441-445.

[2] Church, A., "A formulation of the simple theory of types," *The Journal of Symbolic Logic,* vol. 15 (1940), pp. 56-68.

[3] Church, A., *The Calculi of Lambda-Conversion,* Annals of Mathematical Studies, no. 6, Princeton University Press, Princeton, New Jersey, 1941.

[4] Coppo, M. and M. Dezani-Ciancaglini, "A new type assignment for λ-terms," *Archiv für Mathematische Logik und Grundlagenforschung,* vol. 19 (1978), pp. 139-156.

[5] Coppo, M., M. Dezani-Ciancaglini, and B. Venneri, "Functional characters of solvable terms," *Zeitschrift für Mathematische Logik und Grundlagen der Mathematik,* to appear.

[6] Curry, H. B. and R. Feys, *Combinatory Logic,* vol. 1, North-Holland, Amsterdam, 1958.

[7] Curry, H. B., R. Hindley, and J. P. Seldin, *Combinatory Logic,* vol. 2, North-Holland, Amsterdam, 1972.

[8] Kleene, S. C., "λ-definability and recursiveness," *Duke Mathematical Journal,* vol. 2 (1936), pp. 340-353.

[9] Morris, J. H., *λ-Calculus Models of Programming Languages,* Ph.D. thesis, MITMAC Reprint TR-57, Massachusetts Institute of Technology, Cambridge, Massachusetts, 1968.

[10] Plotkin, G., *A Set Theoretical Definition of Application,* Memo, MIP-R-85, Edinburgh, 1972.

[11] Prawitz, D., *Natural Deduction, a Proof-Theoretical Study,* Almqvist & Wiksell, Stockholm, 1965.

[12] Sallé, P., "Une extension de la théorie des types en λ-calcul," pp. 398-410 in *Automata, Languages and Programming,* eds., G. Ausiello and C. Böhm, Lecture Notes in Computer Science, Springer-Verlag, vol. 62, 1978.

[13] Scott, D., "Continuous lattices," pp. 97-136 in *Toposes, Algebraic Geometry and Logic,* ed., F. W. Lawvere, Lecture Notes in Mathematics, Springer-Verlag, vol. 274, 1972.

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

[15] Seldin, J. P., *The Theory of Generalized Functionality I,* Unpublished manuscript, Southern Illinois University at Carbondale, 1976.

[16] Seldin, J. P., "A sequent calculus for type assignment," *The Journal of Symbolic Logic,* vol. 42 (1977), pp. 11-28.

[17] Wadsworth, C. P., "The relation between computational and denotational properties for Scott's D_∞-models of the lambda calculus," *SIAM Journal on Computing,* vol. 5 (1976), pp. 488-521.

Istituto di Scienza dell'Informazione
dell'Università di Torino
Corso Massimo d'Azeglio 42-10125 Torino, Italy