Definedness

Solomon Feferman[1]

Dedicated to Alonzo Church

Abstract

Questions of definedness are ubiquitous in mathematics. Informally, these involve reasoning about expressions which may or may not have a value. This paper surveys work on logics in which such reasoning can be carried out directly, especially in computational contexts. It begins with a general logic of "partial terms", continues with partial combinatory and lambda calculi, and concludes with an expressively rich theory of partial functions and polymorphic types, where termination of functional programs can be established in a natural way.

1 Why logics of definedness?

Questions of definedness are ubiquitous in mathematics; they are to be distinguished from the equally ubiquitous questions of existence. I do not mean by the latter, (external) metaphysical questions about the nature and existence of mathematical objects, but rather (internal) mathematical questions such as whether there exist infinitely many twin primes or whether there exist any zeros of the Riemann zeta function in the critical strip off the line $x = 1/2$. Questions of definedness, on the other hand, have to do with whether some mathematical expression has a value, or "exists", according to some explanation of how it is to be evaluated, if at all. Answers to such questions require a definite semantic context. In algebra, for example, the definedness of $\sqrt{b^2 - 4ac}$ is sensitive to whether we are working in integers, rationals, reals or complex numbers. Much of the conceptual progress in mathematics can be laid to wrestling with awkward problems of undefinedness or ambiguity of definedness in some number system or space.

Questions of definedness are particularly ubiquitous in analysis: in daily practice one deals regularly with expressions such as

$$\lim_{n\to\infty} a_n\ ,\ \sum_{n=0}^{\infty} a_n\ ,\ \sum_{n=0}^{\infty} a_n x^n\ ,\ \sum_{n=-\infty}^{\infty} a_n e^{inx}\ ,\ \lim_{x\to a} f(x)\ ,\ f'(x)\ ,\ \int_a^b f(x)\,dx$$

etc., etc., without knowing in advance whether or not they are defined. Moreover, there is a substantial divergence (sic!) between the "official" account governing the use of such

[1]Lecture for mini-conference on Partial Functions and Programming: Foundational Questions, U.C. Irvine, 17 May 1995. I wish to thank Professor Karel Lambert for his work in organizing this conference. (Research supported by grants from the N. S. F.)

expressions according to modern logical and set-theoretical criteria, and their everyday usage. Take, for example, the expression $f'(x)$ for (partial) functions f of a real argument x. According to one official account, we first define an operation D as follows: the domain of D consists of all pairs (f, x) such that

(1)
$$\text{(i)}\quad \exists\delta\, \forall y(0 < |y-x| < \delta \rightarrow y \in \text{dom}(f)),\ \text{and}$$
$$\text{(ii)}\quad \exists L\, \forall \in > 0\, \exists\delta > 0\, \forall y \left(0 < |y-x| < \delta \rightarrow \left|\frac{f(y)-f(x)}{y-x} - L\right| <\in\right) .$$

Then it is proved that for each $(f, x) \in \text{dom}(D)$, the L in (ii) is unique if it exists, and $D(f, x)$ is defined in that case to be the unique such L. Finally, $f'(x)$ is said to be defined if $(f, x) \in \text{dom}(D)$, and for such f, x, $f'(x)$ is taken, by definition, to be $D(f, x)$. How much more cumbersome all this is than the "definition":

(2)
$$f'(x) = \lim_{y \to x} \frac{f(y)-f(x)}{y-x}\ ,\quad \text{if it exists .}$$

It is natural to make statements about definedness and reason about them directly, rather than push back to the official definition. For example, again with the derivative, one states such results as:

(3)
$$\begin{array}{rl} & \text{if } f'(x) \text{ and } g'(x) \text{ are defined then} \\ \text{(i)} & (f+g)'(x) \text{ is defined and equals } f'(x)+g'(x),\ \text{and} \\ \text{(ii)} & (f \cdot g)'(x) \text{ is defined and equals } f'(x)g(x)+f(x)g'(x)\ . \end{array}$$

Proofs of such statements may require us to return to the definition of D, but their application to compound expressions does not.

In recursion theory and computer science, questions of definedness usually have to deal with termination of an algorithm; typically, this is given recursively rather than explicitly, for example in Euclid's algorithm for $gcd(a, b)$ of natural numbers a, b:

(4)
$$gcd(a, b) = [a \text{ if } b = 0 \text{ else } gcd\,(b,\ \text{rem}\,(a, b))]\ .$$

One proves by complete induction on b that

(5)
$$\text{for all } a, gcd(a, b) \text{ is defined}\ .$$

Of course, justification of such reasoning presupposes a semantical account of recursive definitions. In particular, on the face of it, (4) requires definition of the functional $F(b) = \lambda a.gcd(a, b)$ by recursion of the form

(6)
$$F(b) = G(b, F \restriction b)\ ,$$

and justification of *that* requires proof by induction on the natural numbers of the statement that for each x there is a unique function(al) F_x with domain $[0, x]$ such that for all $y \leq x$

(7)
$$F_x(y) = G(y, F_x \restriction y)\ ;$$

then $F(b)$ is defined to be $F_b(b)$ for the unique such F_b with $\mathrm{dom}(F_b) = [0, b]$.

Granted that reasoning about definedness of one form or another of mathematical expression presupposes a definite semantics which determines the value of the expression in question when defined, one may ask whether it is not *then* possible to carry on sustained reasoning about definedness without hearkening back to the semantical basis.

Indeed, as will be explained in the next section, rather general logics for that purpose have been proposed and explored since the 1950s. In subsequent sections we shall see how one of these has been applied to reasoning about definedness in the specific area of the theory of computation.

2 Logics of existence vs. logics of definedness.

Within philosophical logic, two general kinds of formal systems for reasoning about existence or definedness have been formulated and studied since the 1950s under the heading of "free logics". To be noted among the contributors to this subject from that time on are Henry S. Leonard, Jaako Hintikka, Hugues Leblanc and Theodore Hailperin, Karel Lambert and Bas van Fraassen. The volume [Lambert 1991] is an excellent recent collection of essays on philosophical applications of free logic, to which Lambert has provided a useful introduction.[2] In it, he distinguishes broadly between *outer* (or *Meinongian*) and *inner* (or *Russellian*) logics of existence. Roughly speaking, in the former, free variables are conceived as ranging over a domain D of possibly non-existing entities; a predicate $E(x)$ holds for x in D if x actually exists. In the latter, all objects in D are regarded as existing, so $E(x)$ holds for all x, but $E(t)$ holds for a term or description t if it has a referent in D. I prefer to refer to these two kinds of systems here as *logics of existence* and *logics of definedness*, respectively.

Within mathematical logic, especially in connection with the study of constructive mathematics and the theory of computation, logics of these two basic kinds have also been introduced and studied since the mid 1970s. In this respect, logics of existence are due to Dana Scott and were presented in [Fourman 1977] and [Scott 1979] (with [Scott 1967] a precursor in some respects); however, the predicate $E(x)$ is not given the same informal interpretation in these as in Meinongian systems but rather is explained in terms of an idea of *partial elements.* Logics of definedness, on the other hand, were introduced by Michael Beeson [1981, 1985] under the rubric *logic of partial terms*; here [Feferman 1975, 1979], with their use of "pseudo" application terms, may be considered as precursors.[3]

A side-by-side comparison of these two kinds of logics is provided by [Troelstra and van Dalen 1988] Vol. I, in Ch. 2, sec. 2, referred to there as *E-logic* and *E^+-logic*, resp.[4] In both logics, the axiom of universal instantiation is modified to its restriction:

$$\forall x A(x) \wedge E(t) \rightarrow A(t) \ , \tag{1}$$

[2]I am indebted to Charles Parsons for bringing this work to my attention.

[3]Recently, Grigori Mints brought to my attention the still earlier formulation and use of such logics by R. A. Pljuškevičus [1968], in the Russian constructivist school of N. A. Shanin; regrettably, both Beeson and I were ignorant of that work.

[4]Troelstra and van Dalen present these in the framework of intuitionistic logic, but the comparison applies equally well to classical logic. Moreover, they work in a natural deduction style formalism, while the following is presented for a Hilbert-style formalism.

as is the dual axiom of existential instantiation. In accordance with the informal "outer" interpretation above, in E-logic one has the rule

(2) $$A(x)/A(t)$$

but *not* the rule

(3) $$A(x)/\forall x A(x)$$

while in E^+-("inner") logic, it is just the other way around. In place of (3) in E-logic one has

(4) $$[E(x) \to A(x)]/\forall x A(x) \ .$$

While free logic in general countenances the possibility that no objects exist, this seems irrelevant to mathematical applications. Thus if $\exists x E(x)$ is included in E-logic, one has:

(5) $$\vdash_{E^+} A(x_1, \ldots, x_n) \Longleftrightarrow \vdash_E E(x_1) \wedge \ldots \wedge E(x_n) \to A(x_1, \ldots, x_n) \ .$$

Hence for A closed,

(6) $$\vdash_{E^+} A \Longleftrightarrow \vdash_E A \ .$$

From the point of view of (6) there would seem to be no essential difference between logics of existence and logics of definedness for the purposes described in the introduction. However, I prefer the latter for the following reasons:

(i) *Philosophical.* I am troubled by the informal interpretation of the predicate E in the "outer" logics of existence, at least for mathematical applications. We do not ordinarily speak in mathematics of non-existing or partially existing objects. This is not to say that there is no coherent semantics for such logics; indeed, Scott and Fourman provide just that via the categorical theory of topoi. But that is very different from ordinary ways of saying what mathematics is "about".

(ii) *Everyday reasoning.* It is habitual to begin a proof of a statement of the form $\forall x A(x)$ holds – where 'x' is understood to range over a domain D – by saying "consider any x in D". This corresponds to the acceptance of rule (3) in definedness logic. There are no ordinary locutions which correspond to the rule (4) in existence logic.

(iii) *Models.* We shall see in the applications of definedness logic to the theory of computation, that one has for it a great variety of directly obtained natural models. (Again, this is not to deny that existence logic has other models of independent interest, such as indicated above.)

3 A basic logic of definedness: the logic of partial terms.

The syntax of the logic of partial terms (Cf. [Beeson 1985], pp. 97–99) is that of first-order predicate calculus with equality, extended by a definedness predicate, from now on written as $t \downarrow$ rather than $E(t)$. For simplicity, this is presented here for a single-sorted language, but the formulations apply *mutatis mutandis* to a many-sorted language.

The basic term-forming symbols are: *variables* ($a, b, c, \ldots, x, y, z$, with or without subscripts), *constant symbols* ($\mathbf{c}$), and *function symbols* ($\mathbf{f}$). There need not be any constant or function symbols; each of the latter that is used has a definite number of arguments. The *terms* $(s, t, \ldots)$ are generated as usual from the variables and constants by application of the function symbols. It will be seen that if there are no function symbols then the logic of partial terms simply reduces to ordinary logic.

The basic predicate-forming symbols are *the equality symbol* $(=)$, *relation symbols* $(\mathbf{R})$, and *the definedness symbol* $(\downarrow)$. There need not be any relation symbols, but each that is used has a definite number of arguments. The *atomic formulas* are all those of the form: $s = t$, $t\downarrow$, and $\mathbf{R}(t_1, \ldots, t_n)$, where $s, t, t_1, \ldots, t_n$ are terms and $\mathbf{R}$ is n-ary.

The basic logical symbols are: $\neg, \wedge, \vee, \rightarrow, \forall$ and $\exists$. Then the *compound formulas* $(A, B, \ldots)$ are generated by their application as usual, to form: $\neg A, A \wedge B, A \vee B, A \rightarrow B, \forall x A$ and $\exists x A$. The underlying logic below may be classical or intuitionistic, and it is only to admit the latter possibility that the full set of logical symbols is taken as basic; otherwise (say) $\neg, \wedge, \forall$ would suffice. The result of substituting a term t for a variable x in a formula A at all free occurrences of x in A is indicated by $A(t/x)$, or $A(t)$ when A is written as $A(x)$; it is assumed this is performed only when there are no collisions of free and bound variables in the resulting substitution.

The possible interpretations to be considered when the underlying logic is classical are given by *structures*

$$\mathcal{M} = \langle M, \mathbf{R}^{\mathcal{M}}, \ldots, \mathbf{f}^{\mathcal{M}}, \ldots, \mathbf{c}^{\mathcal{M}}, \ldots \rangle \ , \tag{1}$$

where (i) M is non-empty, (ii) for each n-ary $\mathbf{R}$ in the language, $\mathbf{R}^{\mathcal{M}} \subseteq M^n$ is an n-ary relation in M, (iii) for each n-ary $\mathbf{f}$ in the language, $\mathbf{f}^{\mathcal{M}}$ is an n-ary *partial function* from M to M (indicated by $\mathbf{f}^{\mathcal{M}} : M^n \underset{p}{\rightarrow} M$ or $\mathbf{f}^{\mathcal{M}} : M^n \overset{\sim}{\rightarrow} M$), and (iv) for each constant symbol $\mathbf{c}$ in the language, $\mathbf{c}^{\mathcal{M}} \in M$. Then for $\alpha : \mathrm{Var} \rightarrow M$ taken to be any *assignment* in M to the variables, one defines a partial map $[\![t]\!]_{\mathcal{M},\alpha}$ from terms t to M inductively as follows:

$$\begin{array}{rl} \text{(i)} & [\![x]\!]_{\mathcal{M},\alpha} = \alpha(x) \\ \text{(ii)} & [\![\mathbf{c}]\!]_{\mathcal{M},\alpha} = \mathbf{c}^{\mathcal{M}} \\ \text{(iii)} & [\![\mathbf{f}(t_1, \ldots, t_n)]\!]_{\mathcal{M},\alpha} \simeq \mathbf{f}^{\mathcal{M}} \left([\![t_1]\!]_{\mathcal{M},\alpha} \cdots, [\![t_n]\!]_{\mathcal{M},\alpha}\right), \end{array} \tag{2}$$

where '$\simeq$' means that the l.h.s. of (iii) is defined and equal to the r.h.s. just in case $\langle [\![t_1]\!]_{\mathcal{M},\alpha}, \ldots, [\![t_n]\!]_{\mathcal{M},\alpha} \rangle$ is in the domain of $\mathbf{f}^{\mathcal{M}}$. We then take

$$[\![t]\!]_{\mathcal{M},\alpha} \downarrow \Longleftrightarrow [\![t]\!]_{\mathcal{M},\alpha} \text{ has a value in } M \ . \tag{3}$$

Turning to the inductive definition of *satisfaction* for atomic formulas, one takes:

$$\begin{array}{rl} \text{(i)} & \mathcal{M} \models_\alpha (s = t) \Longleftrightarrow [\![s]\!]_{\mathcal{M},\alpha} \downarrow \& [\![t]\!]_{\mathcal{M},\alpha} \downarrow \& [\![s]\!]_{\mathcal{M},\alpha} = [\![t]\!]_{\mathcal{M},\alpha}, \\ \text{(ii)} & \mathcal{M} \models_\alpha (t\downarrow) \Longleftrightarrow [\![t]\!]_{\mathcal{M},\alpha} \downarrow, \text{ and} \\ \text{(iii)} & \mathcal{M} \models_\alpha \mathbf{R}(t_1, \ldots, t_n) \Longleftrightarrow \\ & [\![t_1]\!]_{\mathcal{M},\alpha} \downarrow \& \ldots \& [\![t_n]\!]_{\mathcal{M},\alpha} \downarrow \& \langle [\![t_1]\!]_{\mathcal{M},\alpha}, \ldots, [\![t_n]\!]_{\mathcal{M},\alpha} \rangle \in \mathbf{R}^{\mathcal{M}} \ . \end{array} \tag{4}$$

Finally, satisfaction for compound formulas is defined as usual.

The following is then a complete Hilbert-style system for this semantics:

I. *Propositional Axioms and Rules*

[see below]

II. *Quantificational Axioms and Rules*

(i) $\forall x A(x) \wedge (t\downarrow) \rightarrow A(t)$ (ii) $A(t) \wedge (t\downarrow) \rightarrow \exists x A(x)$

(iii) $\dfrac{B \rightarrow A(x)}{B \rightarrow \forall x A(x)}$ (iv) $\dfrac{A(x) \rightarrow B}{\exists x A(x) \rightarrow B}$,

'x' not free in B in (iii), (iv).

III. *Definedness Axioms*

(i) $x\downarrow$
(ii) $\mathbf{c}\downarrow$
(iii) $\mathbf{f}(t_1, \ldots, t_n)\downarrow \rightarrow (t_1\downarrow) \wedge \ldots \wedge (t_n\downarrow)$
(iv) $s = t \rightarrow (s\downarrow) \wedge (t\downarrow)$
(v) $\mathbf{R}(t_1, \ldots, t_n) \rightarrow (t_1\downarrow) \wedge \ldots \wedge (t_n\downarrow)$

IV. *Equality Axioms*

(i) $x = x$
(ii) $s = t \rightarrow t = s$
(iii) $r = s \wedge s = t \rightarrow r = t$
(iv) $\mathbf{R}(s_1, \ldots, s_n) \wedge \bigwedge_{1 \le i \le n} (s_i = t_i) \rightarrow \mathbf{R}(t_1, \ldots, t_n)$
(v) $\bigwedge_{1 \le i \le n} (s_i = t_i) \rightarrow \mathbf{f}(s_1, \ldots, s_n) \simeq \mathbf{f}(t_1, \ldots, t_n)$,

where, by definition,

$(s \simeq t) := (s\downarrow \vee\, t\downarrow \rightarrow s = t)$.

It is understood for the semantics presented in (1) – (4) above that the axioms and rules in I are complete for the classical propositional calculus and include the rule of Modus Ponens. If these are replaced by one or another standard version of the intuitionistic propositional calculus, one obtains with II – IV a complete system for a suitable Kripke-style semantics. In either case, the system is denoted LPT for the Logic of Partial Terms.

The Definedness Axioms III are worthy of special attention. III (i), (ii) reflect the conception of inner free logic, according to which all objects exist. III (iii) – (v) are called *strictness axioms*. In particular, in the theory of computation, III (iii) corresponds to *call-by-value semantics*; the alternative, *call-by-name semantics*, permits violations of this axiom.

Remark. Following [Lambert & van Fraassen 1967], one may add *partial description operators* $(\iota x)A(x)$ to LPT with the axioms

(DES) $(\iota x)A(x) = y \leftrightarrow \forall x[A(x) \leftrightarrow x = y]$.

Thus, by the strictness axiom III (iv), we have (equivalently)

$$\begin{array}{ll} \text{(i)} & (\iota x)A(x)\downarrow \leftrightarrow (\exists ! x)A(x), \text{ and} \\ \text{(ii)} & (\exists ! x)A(x) \rightarrow A\left((\iota x)\, A\,(x)\right). \end{array}$$

Similarly, one may add *partial selection operators* $(\varepsilon x)A(x)$ with the axioms:

$$\text{(SEL)} \qquad \begin{array}{ll} \text{(i)} & (\varepsilon x)A(x)\downarrow \leftrightarrow (\exists x)A(x), \text{and} \\ \text{(ii)} & (\exists x)A(x) \rightarrow A(\varepsilon x.A(x)) \end{array}$$

The axioms DES are of clear philosophical interest, providing one solution of the familiar puzzles as to how to handle non-referring definite descriptions, such as in 'the present king of France is bald'. But they also serve to regulate the use of mathematical definitions such as

$$\lim_{n\to\infty} x_n \;:=\; (\iota x)\forall\epsilon > 0 \exists n \forall m > n\,(|x_n - x| < \epsilon),$$

$$\text{and} \quad \{x : A(x)\} \;:=\; (\iota a)\forall x(x \in a \leftrightarrow A(x))$$

The SEL axioms also serve both philosophical and mathematical purposes. A simple example of the latter is provided by

$$\sqrt{a} \;:=\; (\epsilon x)(x^2 = a).$$

4 Logics of definedness for partial combinatory and lambda calculi; preliminary discussion.

We now turn to specializations of the Logic of Partial Terms to languages for untyped combinatory and lambda calculi, in which we can reason about definedness of their terms. The point of view here is different from the original conceptions of these calculi, for which see [Church 1941], [Curry & Feys 1968], as well as [Barendregt 1977, 1984], and [Hindley & Seldin 1986]. As originally conceived by Alonzo Church and Haskell B. Curry these were to be all-embracing systems for the most general concept of functions-as-rules. Every term s is thought of as representing a function; the formal application of s to t, written st, is supposed to represent the value of s at t. These calculi were intended to subsume theories of sets as a foundation for all of mathematics, with sets replaced by their characteristic functions. This can be done by using the identification of certain terms with the truth values T and F. Characteristic functions, then, are propositional functions in another guise, i.e. functions which evaluate for each object to T or F.

Not only were these calculi to encompass all of mathematics, they were also supposed to subsume logic as well. For, the propositional operations could be identified with suitable truth-value functions and the quantifiers with suitable functions to be applied to propositional functions. The discovery by Stephen C. Kleene and J. Barkley Rosser of paradoxes in Church's original system led him to retrench on these goals. Curry, on the other hand, continued to pursue the development of consistent logical (or "illative") systems within his combinatory calculi, albeit with limited success.

In both kinds of calculus, a variety of *reduction rules* are considered, leading to a syntactic relation $s \geq t$ of *reducibility* between terms. Of special interest are those terms which are *irreducible* or in *normal form*, i.e. for which $s \geq t$ holds only when s, t are identical. s is said to *have a normal form* if it is reducible to a term t in normal form. The fundamental theorem of Church and Rosser shows that t is unique in such a case (for suitable reduction rules). The possibility of self-application in the untyped combinatory and lambda calculi leads quickly to terms which have no normal form. The following quotations reveal that Church and Curry differed as to the significance of this situation.

> It is intended that, in any interpretation of the formal calculus [of lambda conversion], only those well-formed formulas [i.e. terms] which have a normal form shall be meaningful. ([Church 1941], pp. 14-15).
>
> If f denotes a particular function, we shall use the notation $(f\alpha)$ for the value of the function f for the argument α. If α does not belong to the range of arguments of f, the notation $(f\alpha)$ shall be meaningless ([Church 1941], p. 1).

In contrast, in the discussion of alternative explanations of the paradoxes in [Curry & Feys 1968] one reads:

> One alternative starts with the observation that [the "paradoxical" combinator] ... has no normal form. This suggests that the possession of a normal form should be a necessary condition for significance. This thesis appears to be the opinion of Church. ... From our standpoint every ob [term of the combinatory calculus] is significant in the broadest sense of being an object of thought; significance in a narrower sense means belonging to some semantical category. It seems to us unlikely ... that a single category of significant obs will suffice for the purposes of logic; rather, we suspect, there must be a whole hierarchy of them. ([Curry & Feys 1968], pp. 260-261)

In addition to reduction relations, one deals in these calculi with relations of *interconvertibility* or *interreducibility* $s = t$. For any given reduction relation, this is defined to hold if there is a finite sequence of terms $s_0, s_1, \ldots, s_n$ beginning with s and ending with t such that $s_i \geq s_{i+1}$ or $s_{i+1} \geq s_i$ for each i. Thus $=$ is the smallest equivalence relation which extends the $\geq$ relation. (For reduction relations for which the Church-Rosser result holds, one has $s = t$ if and only if there is an r with $s \geq r$ and $t \geq r$.)

In the spirit of the original conceptions, both Church and Curry treated the $\geq$ and $=$ relations as defined informally, outside of their calculi; equally, reasoning about them was carried out informally. A major step toward treating these calculi within the framework of mathematical logic was taken by Dana Scott in 1969 with his construction of the so-called D_∞ model for the λ-calculus and, thence, the combinatory calculus (cf. [Scott 1972] and the general references above). The very idea of discussing the semantics of these systems in terms of models was novel to the subject at that time. What these constituted were models of the λ- calculus and combinatory calculus as equational calculi. But the question of definedness of expressions in one sense or another was still regarded as external to the calculi. It was only with [Feferman 1975] that such calculi were first treated within the full symbolism of the first-order predicate calculus with equality in such a way as to reason about definedness internally. The approach there took as a basic relation between individuals a three-place relation $App(x, y, z)$ informally interpreted as: x applied to y evaluates to z, and written $xy \simeq z$. In the cited paper, terms of the combinatory calculus were not dealt with as first-class citizens, but only introduced contextually (as so-called "pseudo-terms"). One first defines $t \simeq z$ inductively; this is taken to be $t = z$ if t is a variable or constant, and then one takes

$$(st \simeq z) := \exists x \exists y (s \simeq x \wedge t \simeq y \wedge App(x, y, z)). \tag{1}$$

Thence one defines

$$s \simeq t := \forall x [s \simeq x \leftrightarrow t \simeq x]. \tag{2}$$

It was only with Beeson's introduction of the logic of partial terms (cf. [Beeson 1981, 1985]) that terms of the combinatory or λ-calculus could be reasoned about as first-class citizens. The shape this takes is described in the next two sections. One can alternatively carry out this kind of reasoning within the "outer" E-logic; that is not done here but the basis for such a development can be found (side-by-side with that in E^+-logic) in [Troelstra & van Dalen 1988] Vol. II, pp. 473 ff. In retrospect, it is tempting to identify the views of Church and Curry quoted above with one or the other of these approaches. This seems to me to be asking more of them than is fair, but if pressed I would say that the quote from Church puts him on the side of E-logic since on the face of it, the universe of discourse consists of all λ-terms, while that from Curry and Feys puts him (them) on the side of E^+-logic. But as there is no restriction on substitutivity of terms for variables in either system (except to avoid collisions of free and bound variables in the λ-calculus) one may have to assimilate both to E-logic.

5 The partial combinatory calculus.

The terms of the standard basic system of combinatory calculus are generated from the variables $a, b, c, \ldots, x, y, z$ and two constants $\mathbf{k}$ and $\mathbf{s}$ by closure under a binary function symbol $\mathbf{f}_{\text{app}}$ for application. We write st for $\mathbf{f}_{\text{app}}(s, t)$ and, as usual, $st_1 \cdots t_n$ for $(\cdots(st_1)\cdots)t_n$. On the face of it, the usual formulation of the combinatory calculus within equational logic treats application as a total operation. This system is denoted CL and is called the *total combinatory logic*. Its axioms are (i) $\mathbf{k}xy = x$ and (ii) $\mathbf{s}xyz = xz(yz)$. By contrast, the system CL_p of *partial combinatory logic* as formulated within LPT, has the following axioms:

(CL_p) (i) $\mathbf{k}xy = x$
(ii) $(\mathbf{s}xy)\downarrow \wedge \mathbf{s}xyz \simeq xz(yz)$.

The idea behind (CL_p) (ii) is that $\mathbf{s}xy$ always names a rule but its value at any given z may not be defined. In contrast, from (CL_p) (i) and the strictness axioms, $\mathbf{k}xy$ is always defined.

CL_p has a variety of interesting and natural models $\mathcal{M}$ (cf. also [Beeson 1985], Ch. VI and [Feferman 1979] sec. III for these).

1° *Recursion-theoretic models.*

(a) *Ordinary recursion theory.* Here the domain of $\mathcal{M}$ is the set $\omega = \{0, 1, 2, \ldots, \}$ of natural numbers. The interpretation of $\mathbf{f}_{\mathrm{app}}(x, y)$ in M is $\{x\}(y)$. There are suitable choices of $\mathbf{k}$, $\mathbf{s}$ in ω to satisfy (CL_p) (i), (ii). This model is also referred to as PRF for the partial recursive functions.

(b) *Generalized recursion theory.* Any enumerative recursion theory on an arbitrary structure, such as the notion of prime computability from [Moschovakis 1969] (cf. also [Fenstad 1980]) serves to determine a model of CL_p.

2° *Term models.* Both of these start from the set $Term$ of terms in the language of CL_p. The reduction relation $s \geq t$ is defined as usual in $Term$ (cf. [Barendregt 1984] or [Hindley & Seldin 1986]). Then one takes the convertibility relation $s \simeq t$ to hold if s, t have a common reduct. (equivalently if (s, t) belongs to the least equivalence relation extending the $\geq$ relation). A term t is said to be in normal form if t is irreducible; NT denotes the set of all terms in normal form. By the Church-Rosser theorem, for each $s \in Term$ there is at most one $t \in NT$ with $s \geq t$; we write $NF(s)$ for t in such a case.

(a) *The normal term model (NT).* Here $\mathcal{M}$ has domain the set of all terms in NT; we take $\mathbf{f}_{\mathrm{app}}(s, t) \simeq NF(st)$. The constants $\mathbf{k}$, $\mathbf{s}$ are interpreted by themselves. (The closed normal terms form a submodel CNT of NT.)

(b) *The total term model* (TT). Here $\mathcal{M}$ has as domain, the set of all (convertibility) equivalence classes $[s]$ for $s \in$ Term; we take $\mathbf{f}_{\mathrm{app}}([s], [t]) = [st]$. Application is thus total in this model. (Again, we can form a submodel CTT of TT by taking equivalence classes $[s]$ of closed terms.)

3° *Generated models.* Given any M with a pairing operation $P : M^2 \overset{1-1}{\to} M$, and any collection $\mathcal{F}$ of partial functions from M to M with $\mathrm{card}(\mathcal{F}) \leq \mathrm{card}(M)$, we can generate a model of CL_p which includes a code $\mathbf{c}_f$ for each element of $\mathcal{F}$. (The pairing operation is used to form codes.) $\mathbf{k}$, $\mathbf{s}$ are chosen to be any distinct members of M, and for any x, y we form further distinct codes $\mathbf{k}x$, $\mathbf{s}x$, $\mathbf{s}xy$. Then the $\simeq$ relation is generated by the inductive closure conditions:

(i) $\quad \mathbf{c}_f x \simeq f(x) \quad$ for each $f \in \mathcal{F}, x \in M$,

(ii) $\quad (\mathbf{k}x)y \simeq x,\;$ and

(iii) $$\frac{xz \simeq u,\; yz \simeq v,\; uv \simeq w}{(\mathbf{s}xy)z \simeq w}$$

4° *Set-theoretic models.* Given any transitive M in the cumulative hierarchy closed under pairings (in the set-theoretical sense), let $\mathcal{F}$ be the set of all functions (in the set-theoretical sense) which belong to M. Then use the procedure in 3° to form a model of CL_p which contains a code of each element of $\mathcal{F}$.

5° *Other models.* Particularly to be mentioned are Scott's D_∞ model and the graph model of Plotkin and Scott. These are more complicated to describe; cf. [Barendregt 1984] for details.

Remark on extensionality. Of the indicated models, only D_∞ is extensional though the graph model satisfies a weak form of extensionality. However, Henk Barendregt showed how one can construct extensional variants NTE and TTE of the term models in 2° above; cf. [Barendregt 1984] for their description.

6 The partial lambda calculus.

We now turn to a version of the lambda calculus in LPT. The terms are generated from variables as usual by application and abstraction, i.e. if s, t are terms then st is a term and $\lambda x.t$ is a term. The latter does not fit into the form of term builders in the first-order predicate calculus as described in sec. 3 above, though it can be subsumed under its extension by selection operators indicated there, i.e. by taking $\lambda x.t := (\varepsilon z)\forall x[zx \simeq t]$. Instead, we introduce the axioms λ_p for *Partial Lambda Calculus* directly, as follows:

(λ_p) (i) $\lambda x.t \downarrow$
(ii) $(\lambda x.t(x))y \simeq t(y)$.

The axiom (ii) corresponds to β-reduction in the ordinary ("total") lambda calculus, but the limitation on instantiation in LPT restricts its application to:

(1) $$s \downarrow \rightarrow (\lambda x.t(x))s \simeq t(s).$$

The usual translation of CL into the total λ-calculus works equally well for CL$_p$ into λ_p: one takes $\mathbf{k}$ to be $\lambda x \lambda y.x$ and $\mathbf{s}$ to be $\lambda x \lambda y \lambda z.xz(yz)$. In the other direction, the following translation of the total λ-calculus into CL works equally well for λ_p into CL$_p$:

Definition For each term t of CL$_p$ the term $\lambda^* x.t$ is a term of CL$_p$ defined as follows:

(i) $(\lambda^* x.x) := \mathbf{skk}$,

(ii) $(\lambda^* x.u) := \mathbf{k}u$ if u is a constant or variable distinct from x, and

(iii) $(\lambda^* x.t_1 t_2) := \mathbf{s}(\lambda^* x.t_1)(\lambda^* x.t_2)$.

This is easily seen to have the following properties:

Lemma If s, t are terms of CL$_p$ then

(i) the free variables of $(\lambda^* x.t)$ are just those of t distinct from x,

(ii) CL$_p \vdash (\lambda^* x.t)\downarrow$, and

(iii) CL$_p \vdash (\lambda^* x.t(x))y \simeq t(y)$.

One then infers the translation of (1):

(2) $$\mathrm{CL}_p \vdash s\downarrow \rightarrow\ (\lambda^* x.t(x))s \simeq t(s).$$

On the basis of this translation, I had taken it for granted since 1975 that every model of CL_p, in particular that in the partial recursive functions (PRF), induces a model of λ_p. However, not long ago, Thomas Strahm discovered that the situation is not so simple and requires reconsideration. The following reports briefly his work [Strahm 1994] in this respect. The problem is that substitution in the partial λ-calculus, defined by

(3) $$(\lambda x.t)[s/y] := \lambda x.t[s/y] \text{ if } x, y \text{ are distinct and } x \notin \text{ free } (s),$$

is not preserved under the $*$-translation. Namely,

(4) $$\text{there exist } s, t \text{ with } (\lambda^* x.t)[s/y] \neq \lambda^* x.(t[s/y]).$$

An example is provided by the terms y for tx and yy for s; in that case $(\lambda^* x.t)[s/y]$ is $\mathbf{k}(yy)$ while $\lambda^* x.(t[s/y])$ is $\mathbf{s}(\mathbf{k}y)(\mathbf{k}y)$.

One consequence of (3) in λ_p which fails in the recursion-theoretic model is weak extensionality, i.e.

(5) $$\lambda_p \vdash s = t \rightarrow \lambda x.s = \lambda x.t.$$

For, $s = t$ implies $(\lambda y \lambda x.y)s \simeq (\lambda y \lambda x.y)t$ so that by the "β-axiom" (λ_p) (ii), $\lambda x.s = (\lambda y \lambda x.y)s = (\lambda y \lambda x.y)t = \lambda x.t$. But the $*$- translation of (5) is false in PRF. One might ask whether there is another reasonable translation of λ_p into CL_p which converts PRF into a model of λ_p. In general, a model $\mathcal{M}$ for λ_p is supposed to determine a partial valuation $[\![t]\!]_\alpha$ for each term t and assignment α in $\mathcal{M}$ such that $[\![st]\!]_\alpha$ is the application in $\mathcal{M}$ of $[\![s]\!]_\alpha$ to $[\![t]\!]_\alpha$ and $[\![\lambda x.t]\!]_\alpha$ is a partial function on the domain M of $\mathcal{M}$ with

(6) $$[\![\lambda x.t]\!]_\alpha m \simeq [\![t]\!]_{\alpha_x^m} \text{ for each } m \in M,$$

where $\alpha_x^m(x) = m$, $\alpha_x^m(y) = \alpha(y)$ for $y \neq x$. As noted by Strahm, my student Elena Pezzoli proved the following:

Theorem *It is not possible to make PRF into a model of λ_p in such a way that the evaluation function $eval(t, \alpha) \simeq [\![t]\!]_\alpha$, for t a λ_p term and α : free $(t) \rightarrow \mathbb{N}$, is partial recursive.*

So not only does the $*$-translation fail to turn PRF into a model of CL_p, it appears from this that no reasonable translation will do so.

At any rate, in view of this situation, Strahm investigated whether a modified form of λ_p could be developed which *does* have a model in PRF. He succeeded in doing this via a calculus $\lambda_p\sigma$ which is a form of λ_p with *explicit substitutions*; a full report of that work is given in [Strahm 1994], and only the basic idea is described here. By a *substitution* θ is meant a set $\theta = \{t_1/x_1, \ldots, t_n/x_n\}$ where $n \geq 0$, the x_i are distinct variables and the t_i are terms of $\lambda_p\sigma$. Those terms are generated simultaneously with the substitutions by closing under application, abstraction, and $t\theta$ for any substitution θ. Then the β-axiom of λ_p is restated as

(7) $$(\lambda x.t)\theta y \simeq t(y/x \cdot \theta),$$

where $(y/x \cdot \theta)$ denotes the substitution $\{y/x\} \cup \theta^{-x}$ and θ^{-x} is θ with any binding for x deleted. There are a number of other axioms governing definedness in this calculus as well as properties of substitutions. Not only is PRF then restored as a model of $\lambda_p\sigma$ but also every model of CL_p described in the preceding section induces a corresponding model for $\lambda_p\sigma$.

Strahm's calculus $\lambda_p\sigma$ turns out to have interesting relations with work on explicit substitution for functional programming languages by [Abadi et al 1991] and [Curien 1991] among others. Moreover, Robert Stärk [1994] has found that there is a very close relationship between Strahm's axiomatization and the functional programming language SCHEME (in which explicit substitutions are called *environments*). So $\lambda_p\sigma$ constitutes a very natural extension of the partial λ-calculus which connects directly with work in theoretical computer science.

7 Logics for termination and correctness of functional programs.

About 10 years ago, I started exploring the application of my "explicit mathematics" approach ([Feferman 1975, 1979]) to logics for reasoning about functional programs, with either CL_p or λ_p at their base. For ready applications, one wants to have a rich language to talk about various *data type structures* involved in practice, e.g. Booleans, natural numbers, sets, lists, trees, records, streams, etc. Program constructs typically involve pairing, projections, definition by cases, recursive definition, etc. While these can be defined in CL_p or λ_p it is more convenient to take them as primitives. In these logics, *programs* are represented by terms t. *Termination* of a program t for arguments in a data type (or class) T is expressed by

$$\forall x \in T\ (tx\downarrow).$$

Specifications are properties $\phi(t)$ of programs; proof of *correctness* of a program according to a specification is then a proof of $\phi(t)$ in one of these logics.

The main aim of computer science is the construction of efficient, correct programs. For correctness one wants program design that makes the program construction follow an informal algorithm as closely as possible. Also large programs should be constructed in modular fashion, so the correctness problems can be broken into manageable pieces, each of which can be improved independently of the others.

There are various measures of the proof-theoretical strength of a logic. The one that is closest to the efficiency concerns of computer science is given by the characterization of its provably recursive functions. In the present framework, a function F is provably recursive if there is a term t such that for each n, tn defines $F(n)$ and is such that $\forall x \in \mathbb{N}(tx\downarrow \wedge tx \in \mathbb{N})$ is provable in the logic. In principle, if a logic is proof-theoretically weak then it is closer to guaranteeing efficiency of its provable recursive functions than one that is strong. On the other hand, the weaker the logic, the less maneuvering room do we have to reason about correctness of programs. The game then is to try to maximize expressive power and flexibility of reasoning while minimizing proof-theoretical strength. In my papers [1990], [1992], [l992a], I've explored a gamut of logics for these purposes ranging in strength from full second-order number theory (PA_2) down to primitive recursive arithmetic (PRA). As indicated by its name, the provably recursive functions of PRA are just the primitive recursive functions. Others have been working on pushing this down further to systems whose provably recursive functions are just the feasibly (polynomial time) computable functions.

For simplicity, I shall describe just one of these logics here, which is of strength PA_2; it is taken from [Feferman 1990], where it is denoted IOC. From the proof-theoretical point of view, PA_2 (and hence IOC) is very, very strong, and its provable recursive functions – for which no informative characterization is known – go far beyond any conceivable practical applications. Thus, I will describe in the next section a drastic formal weakening of IOC, yet one which retains many of its advantages as a place to reason freely about programs.

The system IOC is a variant of the system IOC_λ from [Feferman 1990]. It is two-sorted, in order to speak both about *individuals* and *classes* (or *types*). *Individual terms* $s, t, \ldots$ are generated from the individual variables $a, b, c, \ldots, x, y, z$ and the constant 0 by the operations: st, $\lambda x.t$, $P(s,t)$, $P_1(t)$, $P_2(t)$, $D(r,s,t,u)$, $Rec(t)$, $Sc(t)$, and $Pd(t)$. *Class terms* $S, T, \ldots$ are generated from the class variables $A, B, C, \ldots, X, Y, Z$ and the class constant $\mathbb{N}$ by the operation $\{x|\phi\}$ where ϕ is any formula. And *formulas* are generated simultaneously with the class terms from the atomic formulas $t\downarrow$, $s = t$ and $s \in T$ by the operations $\neg\phi$, $\phi \vee \psi$, $\phi \wedge \psi$, $\phi \rightarrow \psi$, $\forall x\phi$, $\exists x\phi$, $\forall X\phi$, $\exists X\phi$.

The logic of IOC is LPT with the strictness axioms extended to include:

$$s \in T \rightarrow s\downarrow . \tag{1}$$

(Thus '$s\downarrow$' is redundant with '$s \in T$'). The non-logical axioms of IOC are as follows

I. (λ_p)

 (i) $(\lambda x.t)\downarrow$

 (ii) $(\lambda x.t(x))y \simeq t(y)$

II. *Pairing, Projections*

 (i) $P(x,y)\downarrow \wedge P_1(z)\downarrow \wedge P_2(z)\downarrow$

 (ii) $P_1(P(x,y)) = x \wedge P_2(P(x,y)) = y$

III. *Definition by Cases*

 (i) $D(x,y,u,v)\downarrow$

 (ii) $x = y \rightarrow D(x,y,u,v) = u$

 (iii) $x \neq y \rightarrow D(x,y,u,v) = v$

IV. *Recursion*

 (i) $Rec(x)\downarrow$

 (ii) $y = Rec(x) \rightarrow yz \simeq xyz$

V. *Natural numbers*

(i) $0 \in \mathbb{N}$

(ii) $x \in \mathbb{N} \rightarrow Sc(x) \in \mathbb{N} \wedge Pd(x) \in \mathbb{N}$

(iii) $x \in \mathbb{N} \rightarrow Sc(x) \neq 0 \wedge Pd(Sc(x)) = x$

(iv) $0 \in X \wedge \forall x(x \in X \rightarrow Sc(x) \in X) \rightarrow \forall x(x \in \mathbb{N} \rightarrow x \in X)$

VI. *Comprehension*

$$y \in \{x|\phi(x)\} \leftrightarrow \phi(y)$$

Any individual x may serve to represent an *operation* (or *function*), with values xy when $xy \downarrow$; we tend to use the variables f, g, h when thinking of individuals as operations. As is common in the proof-theoretical literature, the comprehension axiom schema is said to be *impredicative*, since when ϕ contains bound class variables, it serves to define a class $A = \{x|\phi(x)\}$ by reference to the "totality" of classes, among which is A itself. These are the reasons for designating this system as IOC, for *Impredicative Theory of Operations and Classes.*

Remark. In view of the preceding section, in order to have a model for IOC in the partial recursive functions, we should modify the λ_p axioms to something like Strahm's $\lambda_p\sigma$.

The following abbreviations are made to simplify notation: (s,t) for $P(s,t), t'$ for $Sc(t)$ and $t \dot{-} 1$ for $Pd(t)$. $(t_1, \dots, t_n)$ is defined inductively by: $(t_1) = t_1, (t_1, \dots, t_{n+1}) = ((t_1, \dots, t_n), t_{n+1})$. We write $(\forall x \in A)\phi$ for $\forall x[x \in A \rightarrow \phi]$ and $(\exists x \in A)\phi$ for $\exists x(x \in A \wedge \phi)$. $A \subseteq B$ abbreviates $(\forall x \in A)(x \in B)$ and $A = B$ is written for $A \subseteq B \wedge B \subseteq A$. Note that implicit in our logic is instantiation of class variables by class terms:

(2)
$$\text{(i)} \quad (\forall X)\phi(X) \rightarrow \phi(T).$$
$$\text{(ii)} \quad \phi(T) \rightarrow (\exists X)\phi(X).$$

Hence we can derive induction on $\mathbb{N}$ for arbitrary formulas:

(3)
$$\phi(0) \wedge \forall x(\phi(x) \rightarrow \phi(x')) \rightarrow (\forall x \in \mathbb{N})\phi(x).$$

Now write

(4)
$$\text{(i)} \quad f : (A \rightarrow B) \text{ for } \forall x \in A(fx \in B), \text{ and}$$
$$\text{(ii)} \quad (A \rightarrow B) = \{z|z : (A \rightarrow B)\}.$$

For n-ary operations, we take $f : (A^n \rightarrow B)$ where

(5)
$$\text{(i)} \quad A \times B = \{z|\exists x \exists y(z = (x,y) \wedge x \in A \wedge y \in B)\},$$
$$\text{(ii)} \quad A^1 = A, A^{n+1} = A^n \times A.$$

Definition by primitive recursion on $\mathbb{N}$ is effected by the recursion axiom together with definition by cases to yield, for any a, g an f with

(6) $$fx \simeq \begin{cases} a & \text{if } x = 0 \\ g(x \dot{-} 1)(f(x \dot{-} 1)) & \text{if } x \neq 0 \end{cases}$$

so that

(7) $$f0 \simeq a \wedge \forall x \in \mathbb{N}(fx' \simeq gx(fx)).$$

From this it is proved by induction on $\mathbb{N}$ that if $a \in A$ and $g : \mathbb{N} \times A \to A$ then

(8) $$(f : \mathbb{N} \to A).$$

By carrying this out uniformly in n parameters one can associate with each $a : \mathbb{N}^n \to A$ and $g : \mathbb{N}^{n+1} \times A \to A$ an $f : \mathbb{N}^n \to A$ with

(9) $$\begin{array}{rl} \text{(i)} & f(x_1, \ldots, x_n, 0) = a(x_1, \ldots, x_n), \text{and} \\ \text{(ii)} & f(x_1, \ldots, x_n, y') = g(x_1, \ldots, x_n, y, f(x_1, \ldots, x_n, y)). \end{array}$$

In particular, all primitive recursive functions are defined in IOC. From this it follows that second-order number theory PA_2 can be translated into IOC. A translation of IOC into PA_2 is described in [Feferman 1990] sec. 6.3, so these systems are of the same strength.

To deal with partial recursive functions in IOC we need to define $(\mu x)(gx = 0)$ for any function g from $\mathbb{N}$ to $\mathbb{N}$. This is obtained by the following recursion:

(10) $$(\mu x)(gx = 0) \simeq f(x, 0) \text{ where } f(x, y) \simeq \begin{cases} y & \text{if } gy = 0 \\ f(x, y') & \text{if } gy \neq 0. \end{cases}$$

The provably recursive functions on $\mathbb{N}$ in IOC may then be identified with those recursive functions which are provably total on $\mathbb{N}$.

As an illustration of a proof of definedness and correctness in this system, let us return to the recursive definition of $gcd(x, y)$ in sec. 1. Here we assume the prior development of arithmetic and the definition of the primitive recursive functions $[x \div y]$ and $\text{rem}(x, y)$ with the properties, for $x, y \in \mathbb{N}$:

(11) $$y > 0 \to x = y \cdot [x \div y] + \text{ rem}(x, y) \wedge \text{ rem}(x, y) < y$$

The recursion for $gcd(x, y)$ defines it as fxy where:

(12) $$fxy \simeq \begin{cases} x & \text{for } y = 0 \\ fy(\text{rem}(x, y)) & \text{if } y \neq 0 \end{cases}$$

To show f is total, we prove by complete induction on $y \in \mathbb{N}$ that

(13) $$(\lambda x.fxy) : \mathbb{N} \to \mathbb{N}$$

That is, suppose $\forall z < y \forall u \in \mathbb{N}[fuz \in \mathbb{N}]$. To show for any $x \in \mathbb{N}$ that $fxy \in \mathbb{N}$, there are two cases to consider: (i) $y = 0$, in which case $fxy = x$, so $fxy \in \mathbb{N}$; (ii) $y \neq 0$, in which case $fxy \simeq fy \text{ rem}(x, y)$, so for $z = \text{ rem}(x, y) < y$ we can apply the induction hypothesis to conclude that $fyz \in \mathbb{N}$, hence again $fxy \in \mathbb{N}$.

To show that f meets the required specification, write $\phi(x, y, z)$ for the formula $[z|x \wedge z|y \wedge \forall u(u|x \wedge u|y \to u|z)]$. Thus what is to be proved is that $\phi(x, y, fxy)$ holds for all $x, y \in \mathbb{N}$. In parallel with the proof of (13), the following is established by complete induction on $y \in \mathbb{N}$:

$$(\forall x \in \mathbb{N})\phi(x, y, fxy). \tag{14}$$

Again there are two cases: (i) $y = 0$, in which case (14) holds by $\phi(x, 0, x)$, and (ii) $y > 0$, in which case one uses that $[z|x \wedge z|y \leftrightarrow z|y \wedge z|\text{rem}(x, y)]$ from (11).

8 Class constructions, polymorphism and virtual classes

A variety of standard class constructions are easily carried out in IOC. Besides $A \times B$, A^n and $A \to B$ already defined in sec. 7, we have a universal class V, finite classes $\{a_1, \ldots, a_n\}$, and the Boolean operations $A \cap B$, $A \cup B$ and $-A$. Sequences of classes $\langle B_x \rangle_{x \in A}$ are represented by a class C with $z \in C \leftrightarrow (\exists x \in A)(\exists y)(z = (x, y) \wedge y \in B_x)$. Then one defines $\bigcup\limits_{x \in A} B_x$ $\bigcap\limits_{x \in A} B_x$ as well as the disjoint sum $\sum\limits_{x \in A} B_x$ and Cartesian product $\prod\limits_{x \in A} B_x$. The sum $\sum\limits_{x \in A} B_x$ is in effect the class C above, while $\prod\limits_{x \in A} B_x$ is obtained as $\{f|(\forall x \in A)(fx \in B_x)\}$. If $T(x)$ is a class term with parameter x then by taking $B_x = T(x)$ one is led in this way to $\bigcup\limits_{x \in A} T_x$ $\bigcap\limits_{x \in A} T_x$, $\sum\limits_{x \in A} T(x)$ and $\prod\limits_{x \in A} T(x)$.

Now if $T(X)$ is a class term with class parameter X, we can also form

$$(1) \quad \begin{array}{ll} \text{(i)} & \bigcup\limits_X T(X) = \{z|\exists X(z \in T(X))\} \text{ and} \\ \text{(ii)} & \bigcap\limits_X T(X) = \{z|\forall X(z \in T(X))\}. \end{array}$$

In this system we cannot form $\sum\limits_X T(X)$ and $\prod\limits_X T(X)$, but it is shown in [Feferman 1990] how to define these in a conservative extension IOC_Λ of IOC where operations f can take classes as arguments; then $\prod\limits_X T(X) = \{f|\forall X(fX \in T(X)\}$. We shall return to this in the discussion of polymorphism, below.

The class constructions in (1) are the first examples where impredicative instances of the comprehension axiom VI of IOC are used. Further such are provided by the definition of abstract versions O_n of the constructive ordinal number classes, following [Feferman 1975] pp. 99-100. With $\underset{\cdot}{0} = (0, 0)$ as a code for the ordinal, 0, $\dot{S}c(x) = (1, x)$ as a code for the successor of the ordinal coded by x, and $\underset{\cdot}{\sup}(n, f) = (2, (n, f))$ as a code for the sup of fx for $x \in O_n$, we can define:

$$(2) \quad \begin{array}{ll} \text{(i)} & O_0 = \mathbb{N} \\ \text{(ii)} & O_{n+1} = \{a|\forall X[\underset{\cdot}{0} \in X \wedge \forall x(x \in X \to \dot{S}c(x) \in X) \\ & \quad \wedge \bigwedge_{k \le n} \forall f[(f : O_k \to X) \to \underset{\cdot}{\sup}(k, f) \in X] \\ & \quad \to a \in X]\}. \end{array}$$

The recursion axiom IV can then be used to establish forms of transfinite recursion on the classes O_n. The procedure (2) can be extended into the transfinite, but that will not be described further here.

The reader acquainted with proof-theoretical investigations will recognize that all these impredicative class constructions can already be carried out in a subsystem of IOC much weaker than PA_2, namely one of strength $\prod^1_1$-CA. As for the data types met in computational practice, still much less suffices. That is clear for the finitary ones such as lists, arrays, records, trees, graphs, etc. But it also holds for infinitary ones such as streams and "infinite-precision" reals. For example, the class of infinite streams of members of A may simply be identified with $\mathbb{N} \to A$; more generally, partial A-streams may be identified with $\{f|(\forall x \in \mathbb{N})[fx \downarrow \to \ fx \in A \land \forall y < x(fy \downarrow)]\}$. And the class of infinite precision reals may be identified with Cauchy sequences, an elementarily defined subclass of $\mathbb{N} \to \mathbb{Q}$, where $\mathbb{Q}$ is the class of rational numbers. Both of these are given by predicative definitions, i.e. are of the form $\{x|\phi\}$ where ϕ contains no bound class variables.

Thus it appears that if the impredicative class comprehension axiom VI of IOC is replaced by an axiom of the same form restricted to predicative (or elementary) formulas ϕ, then all the data types met in practice are readily constructed. The resulting system is called the *Elementary Theory of Operations and Classes*, denoted EOC. It was shown in [Feferman 1990] that EOC is of the same strength as the system PA of Peano Arithmetic, for which an informative classification by Georg Kreisel of its provably recursive functions has been known for many years (cf. [Schwichtenberg 1977]). What, then, is the value for the theory of computation of permitting impredicative class constructions of the kind described above? An advantage that has received considerable attention in the recent literature is in the use of *polymorphic types*. This grew out of work by Jean-Yves Girard and, independently, John C. Reynolds in the 1970's; for a recent introduction and survey, see [Mitchell 1990]. The idea is illuminated by the following examples:

$$(3) \qquad \begin{array}{rl} \text{(i)} & (\lambda x.x) \in \bigcap_X (X \to X) \\ \text{(ii)} & \lambda x \lambda y.x(xy) \in \bigcap_X [(X \to X) \to (X \to X)] \\ \text{(iii)} & \lambda x \lambda y.x(y,y) \in \bigcap_X [(X \times X \to X) \to (X \to X)] \end{array}$$

Each of these tells us something of the uniform (or polymorphic) behavior of an individual operation with respect to arbitrary types. In (i) this is the identity operation, in (ii) it is the operation of iterating any operation once, and in (iii) it is the operation of identification of arguments of any binary operation. In a system such as IOC_Λ (indicated above), where classes (types) can also serve as arguments of operations, (3) (i) would be expressed instead as

$$(4) \qquad \Lambda X(\lambda x : X)x \in \prod_X (X \to X),$$

and similarly for (3) (ii), (iii). The difference is that the term $\Lambda X(\lambda x : X)x$ wears its polymorphic behavior on its sleeve, while that of $\lambda x.x$ in (3) (i) must be deduced. The latter is called *implicit polymorphism* and the former *explicit polymorphism*. However it has long been known by application of a simple syntactic "erasing" operation that the latter can always be reduced to the former, so there is no difference in strength between these forms of polymorphism.

As it turns out, the essential impredicativity in such polymorphic type constructions comes not in their definition, but in their application. If it is known of a term t that

(5) $$t \in \bigcap_X T(X),$$

does it follow for any class term S that

(6) $$t \in T(S)?$$

Obviously, yes, if we admit the full axiom scheme of instantiation for class quantification,

(7) $$\forall X \phi(X) \rightarrow \phi(S).$$

But here, now, is a way (initiated in my [1990] paper, and explained more clearly in my papers [1992], [1992a]) to have one's cake and eat it too, i.e. to have the advantages of polymorphism while working in logics of restricted strength so as to bound the complexity of the provably recursive functions. Namely, one singles out a class of U-types ("user" types) S which includes the data types to be used in practice and *restricts* instantiation (7) to such S, while one *retains* the full impredicative comprehension axiom VI for arbitrary $\{x|\phi(x)\}$ (called the *G-types* or "general" types). Put in other terms, what is involved here is the idea of *virtual classes*, which is a kind of free logic for classes: every abstract $\{x|\phi(x)\}$ for ϕ a formula of IOC counts as a class term and satisfies

(8) $$y \in \{x|\phi(x)\} \leftrightarrow \phi(y),$$

and can be reasoned with on that basis. But only certain class terms selected according to some criterion or another serve to instantiate class quantifiers.

9 Further directions of work

(i) In parallel with the kind of definedness theory for untyped individual terms presented here, one would like to push the idea of virtual classes still further to develop a reasonable definedness theory for untyped (or unstratified) class terms, unlike those of IOC.

(ii) Continuing the pursuit of rich, flexible logics whose provably recursive functions are of bounded complexity, one would like to obtain such systems whose provable functions are just those which are polynomial-time computable. As essay in this direction was made several years ago by Sam Buss and Jeffrey Remmel (unpublished notes). More recently, Thomas Strahm [1995] has come up with a different and very promising solution.

(iii) Work on computation on abstract data types with applications to infinite streams and infinite precision reals initiated in [Feferman 1992b, 1992c] is continuing; that should be fit into the present framework.

(iv) The main thing that needs to be done is to see if the termination and correctness of substantial programs can be demonstrated in logics of the sort considered here. Though restricted logics are of interest from the proof-theoretical point of view of in-principle complexity bounds, there is no reason not to make use of the full resources of IOC for such purposes, at least to begin with.

Stanford University

References

M. Abadi, L. Cardelli, P.-L. Curien, and J.-J. Levy [1991], Explicit substitutions, *J. of Functional Programming*, 375–416.

H. P. Barendregt [1977], The type-free lambda calculus, in [Barwise 1977], 1091–1132.

_________ [1984], *The Lambda Calculus: Its Syntax and Semantics* (2nd. edn.), North-Holland (Amsterdam).

J. Barwise (ed.) [1977], *Handbook of Mathematical Logic*, North-Holland (Amsterdam).

M. Beeson [1981], Formalizing constructive mathematics: why and how? in *Constructive Mathematics*, Lecture Notes in Mathematics **873**, 146–190.

_________ [1985], *Foundations of Constructive Mathematics*, Springer-Verlag (Berlin).

A. Church [1941], *The Calculi of Lambda-Conversion*, Princeton University Press, reprinted 1963 by University Microfilms Inc. (Ann Arbor).

P. -L. Curien [1991], An abstract framework for environment machines, *Theoretical Computer Science* **82**, 389–402.

H. B. Curry and R. Feys [1958], *Combinatory Logic*, Vol. I, North-Holland (Amsterdam).

S. Feferman [1975], A language and axioms for explicit mathematics, in *Algebra and Logic*, Lecture Notes in Mathematics **450**, 87–139.

_________ [1979], Constructive theories of functions and classes, in *Logic Colloquium* '78, North-Holland (Amsterdam), 159–224.

_________ [1990], Polymorphic typed lambda-calculi in a type-free axiomatic framework, in *Logic and Computation*, Contemporary Mathematics **106**, 101–136.

_________ [1992], Logics for termination and correctness of functional programs, in *Logic from Computer Science*, MSRI Publications **21**, Springer-Verlag (New York), 95–127.

_________ [1992a], Logics for termination and correctness of functional programs, II. Logics of strength PRA, in *Proof Theory*, Cambridge Univ. Press (Cambridge).

_________ [1992b], A new approach to abstract data types, I. Informal development, *Mathematical Stuctures in Computer Science* **2**, 193–229.

_________ [1992c], A new approach to abstract data types, II. Computability on *ADTs* as ordinary computation, in *Computer Science Logic*, Lecture Notes in Computer Science **626**, 79-95.

J. E. Fenstad [1980], *General Recursion Theory. An Axiomatic Approach*, Springer-Verlag (Berlin).

M. P. Fourman [1977], The logic of topoi, in [Barwise 1977], 1053–1090.

J. R. Hindley and J. P. Seldin [1986], *Introduction to Combinators and λ-Calculus*, Cambridge Univ. Press (Cambridge).

K. Lambert (ed.) [1991], *Philosophical Applications of Free Logic*, Oxford University Press (New York).

K. Lambert and B. van Fraassen [1967], On free description theory, *Zeitschr. f. Math. Logik und Grundlagen der Math.* **13**, 225–40.

J. C. Mitchell [1990], Type systems for programming languages, in *Handbook of Theoretical Computer Science, Vol. B. Formal Models and Semantics*, Elsevier (Amsterdam), 365–458.

Y. N. Moschovakis [1969], Abstract first-order computability I, *Trans. Amer. Math. Soc.* **138**, 427–464.

R. A. Pljuškevičus [1968], A sequential variant of constructive logic calculi for normal formulas not containing structural rules, in *The Calculi of Symbolic Logic*, I, Proc. of the Steklov Inst. of Mathematics **98**, AMS Translations (1971), 175–229.

H. Schwichtenberg [1977], Proof theory: Some applications of cut-elimination, in [Barwise 1977], 867–895.

D. Scott [1967], Existence and description in formal logic, in *Bertrand Russell: Philosopher of the Century*, Little, Brown, and Co. (Boston), 181–200. (Reprinted in [Lambert 1991], 28-48].

__________ [1972], Continuous lattices, in *Toposes, Algebraic Geometry and Logic*, Lecture Notes in Mathematics **274** 97–136.

__________ [1979], Identity and existence in formal logic, in *Applications of Sheaves,* Lecture Notes in Mathematics **753**, 660—696.

R. Stärk [1994], Applicative theories and SCHEME. (in preparation).

T. Strahm [1994], Partial applicative theories and explicit substitutions, *J. Logic and Computation* (to appear).

__________ [1995], Polynomial time operations in applicative theories (in preparation).

A. S. Troelstra and D. van Dalen [1988], *Constructivism in Mathematics. An Introduction*, Vols. I and II, North-Holland (Amsterdam).