Wellfounded and non-wellfounded types of continuous functionals

Dag Normann
The University of Oslo

1 *Introduction and summary*

When Kreisel [3] introduced the continuous functionals, the main motivation was to find an absolute notion of 'constructibly true' for statements of analysis. A statement of analysis can be represented as a type, which again can be seen as a set of functionals at a certain level, and the statement is constructibly true if this type has a recursive element.
It turns out that any statement of analysis proved in a constructive formal logic will be constructibly true in Kreisel's sense.

Kleene [2] introduced the equivalent hierarchy of countable functionals as a subhierarchy of the hierarchy of all functionals of finite types. The countable functionals are closed under Kleene's [1] notion of computability, and are thus natural from a recursion-theoretic point of view.

We see type theory as an attempt to extend constructive analysis to a constructive set theory. In type theory we deal with function spaces, infinite product spaces and variants of second order constructions. It is well known that in order to get a full set-theoretic model of transfinite first order type theory one need an inaccessible cardinal or some other high-power set-theoretical axioms. Naive models for type-theory with universes require even stronger principles.
Such results show that the set-theoretical principles on which type theory is based are very strong and that the constructivity-aspect of the theory must be reflected in the semantics if the semantics is going to be

manageable.
The question trigging off the writing of this paper was as follows:

> If we construct a model for transfinite type theory where each object is countable and where 'constructive' is replaced by 'continuous', what will the complexity of the full model be?

Since it is the nature of the underlying intuition and not of the formal theory we are investigating, it is important that the semantics must be based on a natural interpretation of the type-constructors themselves and not of the various formal theories for them.
In this paper we will use a simplified version of Kreisel's continuous functionals as the base for such investigations.
The many characterisateons of the continuous or countable functionals of finite types (see Normann [4] or [5] for a survey of such characterisateons) show that the concept is a natural one. It may then be expected that a continuation of the hierarchy based on the same idea of construction will form a possible source for semantics for type theories, and that the investigation of this hierarchy gives back some information about the nature of type theory.
Although we are not constructing models for specific theories, we expect that when a statement can be represented as a type, and if this statement is proved in any reasonable constructive logic, the type will have a recursive element.

We will first define the basic hierarchy, where we do not take matters as equality between objects or effectivity of the structures into consideration. Then we will discuss how to restrict the constructions in various ways giving more structure to the objects constructed.

Our hierarchy of 'types' will be based on the standard constructions of type theory, such as finite and infinite sums and products of continuously parameterised families of types. In addition we generalize the introduction of types via strict positive inductions. We will accept types with non-wellfounded trees of subtypes, but where each element will have a wellfounded basis. We call such objects *type-streams*.

The results are in three groups.

The first results show that there are natural recursion theorems in this setting. The advantage of type-streams to strictly positive operators is that the predecessor relation is not so rigid. Thus the recursion theorem will cover recursive definitions of functions f where the calculation of f(x) requires the knowledge of f(y) for objects y at some depth below x, and not necessarily at the immediate predecessors. Type-streams also makes it possible to represent more relevant sets of predecessors as types.
Using Baire-space (N^N) makes it easy to view inductively defined types as a special case of nonwellfounded types, but there are of course possible to model such types with domains, with categories or using any other standard way of modelling type theory.

The next group of results shows a close connection between the hierarchy at hand and the hierarchy for recursion in the functional ^3E. Forming dependent types is more or less a continuous collection, while recursion in ^3E is based on a highly discontinuous collection principle. The proof of the equivalence of those two hierarchies supports to some extent the view that the notion of a transfinite type is a complex and non-constructive one.

At the end we use ^3E to extend the hierarchy to a hierarchy accepting certain second order constructions, and we prove some basic properties of these constructions. This shows that we do not need to increase the complexity in order to model second order phenomena.

A full account of these results will require several long, tedious and unexciting proofs. We will leave out many details, in order to make the underlying ideas more transparent. Some proofs will be left entirely for the reader, while for others we outline the main steps. It may sometimes take some effort to fill in the details.

2 *The basic hierarchy*

We are going to define a hierarchy of types, where the elements of the types will be Kleene-type associates. At each level the hierarchy will be closed under strictly positive inductions. These inductively defined types will be special cases of what we may call *type-streams*.

Example 1
Consider the equation

$$X = \mathbf{N} \oplus \mathbf{N} \to X.$$

In our semantics we will let

$$X = \mathbf{N} \oplus (\mathbf{N} \to (\mathbf{N} \oplus (\mathbf{N} \to (\mathbf{N} \oplus \ldots_{\inf} \ldots))))$$

and for an object α to be an element of X we require that the *evaluation tree* of α is well founded.

Example 2
Consider the type stream

$$X = \mathbf{N} \oplus (\mathbf{N} \to (\mathbf{N}^{\mathbf{N}} \oplus (\mathbf{N} \to (\mathbf{N}^{(\mathbf{N}^{\mathbf{N}})} \oplus (\mathbf{N} \to \ldots \ ))))).$$

This does not correspond to a single type-equation. Still we may define the elements of X in a precise way. Our definition of types will capture types like X in example 2.
We are now ready to give the formal definitions. We will let FUN be a recursive set of code numbers of finite functions from $\mathbf{N}$ to $\mathbf{N}$, identifying them with the functions themselves. An *associate* will be a function from $\mathbf{N}$ to $\mathbf{N}$ that maps FUN monotonuously into FUN. Any associate γ will define a partial continuous function $F_\gamma: \mathbf{N}^{\mathbf{N}} \to \mathbf{N}^{\mathbf{N}}$ in the canonical way.

2.1 *Definition*
By induction on the ordinal α we define the class $B_\alpha \subseteq \mathbf{N}^{\mathbf{N}}$ together with the interpretation $I(f) \subseteq \mathbf{N}^{\mathbf{N}}$ for $f \in B_\alpha$. The elements of B_α will be *indices* for the basic α-types.
Assume that B_β is defined for $\beta < \alpha$ and that $I(f)$ is defined for $f \in B_\beta$.

i) B_α is the largest set satisfying
$h \in B_\alpha$ if and only if
$h = {}^1 0$ (the constant zero)
or
$h = <1, f, g>$ or $h = <2, f, g>$ for some $f, g \in B_\alpha$
or
$h = <3, f, g>$ or $h = <4, f, g>$
where $f \in B_\beta$ for some $\beta < \alpha$ and g is an associate such that
$F_g(x) \in B_\alpha$ whenever $x \in I(f)$

ii) We define $\{ I(h) \}_{h \in B_\alpha}$ as the minimal solution to the following equations in $\{ J(h) \}_{h \in B_\alpha}$:

$h = {}^1 0$: $J(h) = \{ {}^1 0 \}$

$h = <1,f,g>$: $J(h) = \{ <1,0,x> \mid x \in J(f) \} \cup \{ <1,1,x> \mid x \in J(g) \}$

$h = <2,f,g>$: $J(h) = \{ <2,x,y> \mid x \in J(f) \text{ and } y \in J(g) \}$

$h = <3,f,g>$: $J(h) = \{ <2,x,y> \mid x \in I(f) \text{ and } y \in J(F_g(x)) \}$

$h = <4,f,g>$: $J(h) = \{ <3,x> \mid x$ is an associate and for all $y \in I(f)$ we have that $F_x(y) \in J(F_g(y)) \}$

$h = <5,f>$: $J(h) = J(f)$.

2.2 *Remarks*

Using the existence of largest fixpoints and least fixpoints of monotone operators we see that the concepts of Definition 2.1 are well defined. The types constructed will contain a unit type and will be closed under disjoint unions, finite products and certain dependent infinite sums and products. The case $h = <5,f>$ is mainly a dummy case, but it will be of technical use besides providing us with the empty type as the least solution to the equation $X = X$.

We will show in some detail that these definitions capture the fixpoints of strictly positive inductive definitions.

2.3 *Definition*

We define the strictly positive operators $\Gamma : \mathcal{P}(\mathbf{N}^{\mathbf{N}}) \to \mathcal{P}(\mathbf{N}^{\mathbf{N}})$, and for each Γ the set of indices γ for Γ as follows:

$\Gamma(X) = X$ has index $<5,->$ where $-$ is the totally undefined function.

$\Gamma(X) = I(f)$ has index $<0,f>$ when $f \in B_\alpha$.

If Γ_1 and Γ_2 have indices γ_1 and γ_2 resp. then $\Gamma(X) = \Gamma_1(X) \oplus \Gamma_2(X)$ has index $<1,\gamma_1,\gamma_2>$ and $\Gamma(X) = \Gamma_1(X) \times \Gamma_2(X)$ has index $<2,\gamma_1,\gamma_2>$.

If $f \in B_\beta$ for some $\beta \in On$, if g is an associate and if for each $x \in I(f)$ we have that $F_g(x)$ is an index for Γ_x then $<3,f,g>$ is an index for $\Sigma(\Gamma_x(X) \mid x \in I(f))$ and $<4,f,g>$ is an index for $\Pi(\Gamma_x(X) \mid x \in I(f))$.

2.4 *Theorem*

If Γ is a strictly positive operator, then there is an ordinal α and an element $Fix_\Gamma \in B_\alpha$ such that $\Gamma(I(Fix_\Gamma)) = I(Fix_\Gamma)$ and this is the least

fixpoint of Γ.

Proof
Let Γ be given. In Γ there may be suboperators that are constant $I(f)$ for some $f \in B_\beta$ for some ordinal β. Let α be an upper bound for the set of such β.
Uniformly recursive in an index γ for Γ there is a monotone continuous operator G on the set of partial functions from $\mathbf{N}$ to $\mathbf{N}$ mapping total objects into total objects and such that if $f \in B_\alpha$ then $G(f) \in B_\alpha$ and $I(G(f)) = \Gamma(I(f))$ for sufficiently large α. G is constructed by a routine application of the recursion theorem in such a way that G has a total fixpoint. This will be our Fix_Γ. We leave the details for the reader.

2.5 *Remark*
In the definition of strictly positive inductive definitions we omitted one possible case

$$\Gamma(X) = \Sigma(\Gamma_2(x, X) \mid x \in \Gamma_1(X))$$

where both Γ_1 and Γ_2 are strictly positive.
There are several reasons for this. One reason is that we would have to restrict Γ_2 to cases where $\Gamma_2(x, X)$ will be meaningful for all $x \in \Gamma_1(X)$ throughout the induction. This is technically unpleasant, but possible.
Another reason is that Theorem 2.4 will fail. It is however possible to alter the definition of B_α in order to recover the theorem.
A third reason is that certain structural results concerning strictly positive inductive definitions will fail. We will mention one result (without proof) which will fail if we extend the definition in any nontrivial way:

2.6 *Theorem* (Functoriality)
Let Γ be strictly positive with index γ.
Uniformly in γ there is a recursive operator Γ^* such that if $g: X \to Y$ is continuous, then $\Gamma^*(g): \Gamma(X) \to \Gamma(Y)$ is continuous.
Moreover Γ^* commutes with composition of functions.

The functorial properties of strictly positive inductive operators are well known.

3 *Two alternative hierarchies*

In this section we will consider two restricted hierarchies with more structure to the individual types. In the E-hierarchy we take equality between objects of a given type into consideration. This leads to a restriction in the definition of dependent types and the two interpretations of one type-term will in general be quite incomparable.

3.1 *Definition*

We define the set E_α together with the interpretation $I(f)$ for each $f \in E_\alpha$ and the equivalence relation $\approx_f$ on $I(f)$ as follows, assuming the definitions given for $\beta < \alpha$:

i) E_α is the largest set satisfying

$h \in E_\alpha$ if and only if

$h = {}^1 0$ (the constant zero)

or

$h = <1, f, g>$ or $h = <2, f, g>$ for some $f, g \in E_\alpha$

or

$h = <3, f, g>$ or $h = <4, f, g>$

where $f \in E_\beta$ for some $\beta < \alpha$ and g is an associate such that

* $F_g(x) \in E_\alpha$ whenever $x \in I(f)$

** $I(F_g(x)), \approx_{F_g(x)} = I(F_g(y)), \approx_{F_g(y)}$ whenever $x, y \in I(f)$ and $x \approx_f y$

or

$h = <5, f>$ for some $f \in E_\alpha$.

ii) We define $\{I(h)\}_{h \in E_\alpha}$ in the same way as for the B-hierarchy, with one important difference

$h = <4, f, g>$: $J(h) = \{<3, x> \mid x$ is an associate, for all $y \in I(f)$ we have that $F_x(y) \in J(F_g(y))$ and for all $y, z \in I(f)$, if $y \approx_f z$ then $F_x(y) \approx_{F_g(y)} F_x(z)\}$

iii) We define $\approx_h$ on $I(h)$ in the obvious way.

This hierarchy is actually closer to the type structures defined by Kleene and Kreisel. There is however one main difference, we accept the empty set as a type. This is of course natural when we want to interpret propositions as types.

We will now describe another hierarchy, where we make sure that all types are nonempty. The objects of this hierarchy, called the D-hierarchy

will be similar to the objects of the B-hierarchy, we just happen to know in some effective way that they are nonempty. As we will see, this requires a restriction on the functions in the product type.
In formulating the hierarchy we need a concept about the B-hierarchy that will be equally meaningful for the D-hierarchy:

3.2 *Definition*
Let $f \in B_\alpha$ for some α.
We define the *tree* T_f of indexed subtypes as follows, using σ for an arbitrary sequence, using comma for concatenation and identifying an object f with the corresponding sequence of length one:
T_f is a set of finite sequences with $f \in T_f$.
If σ, g is in T_f then
i) if $g = <1, g_1, g_2>$ or $g = <2, g_1, g_2>$, then σ, g, g_1 and σ, g, g_2 are in T_f.
ii) if $g = <3, g_1, g_2>$ or $g = <4, g_1, g_2>$ then $\sigma, g, h, F_{g_2}(h) \in T_f$ for each $h \in I(g_1)$
iii) if $g = <5, g_1>$ then $\sigma, g, g_1 \in T_f$.

The tree T_f contains codes for all the subtypes of $I(f)$, and in case of dependent sums or products, also an information about the parameter of the node.

3.3 *Definition*
By simultaneous recursion on the ordinal α we define the set D_α of indices together with
The set $\Delta_f \subseteq FUN$ for each $f \in D_\alpha$
The function $e_{f,\delta}$ for each $f \in D_\alpha$ and $\delta \in \Delta_f$
The interpretation $I(f)$ for all $f \in D_\alpha$.

We define the relation $f \in D_\alpha$ from the set of D_β for $\beta < \alpha$ as we defined B_α from B_β with one restriction:
If $x_1, x_2, \ldots$ is an infinite branch in T_f then for arbitrarily large n we have that $x_n = <1, g_1, g_2>$ and $x_{n+1} = g_2$.

This is a restriction of the non-wellfoundednes accepted, any such non-wellfoundednes must arise from infinitely often going through the right part of a disjoint union. We will see that this ensures in an effective way

that all types will be nonempty.

Let $f \in D_\alpha$. Uniformly recursive in f we construct a set Δ_f of finite partial functions on $\mathbf{N}$ and for each $\delta \in \Delta_f$ a function extending δ. The point will be that $e_{\delta,f} \in I(f)$ and that for all $g \in I(f)$ and finite sets X in $\mathbf{N}$ there is a subfunction $\delta \in \Delta_f$ of g defined for all $n \in X$. Δ_f will always contain the empty function.

We will use the recursion theorem to compute Δ_f and the extension maps $e_{\delta,f}$ for $\delta \in \Delta_f$.
We consider the six possible cases:

o) $f = {}^1 0$. Then Δ_f contains all finite functions that are constant 0, and $e_{\delta,f} = {}^1 0$ for all $\delta \in \Delta_f$.

i) $f = <1, g, h>$. Let Δ_f be the empty function together with the functions $<1, 0, \delta>$ for $\delta \in \Delta_g$ and the functions $<1, 1, \delta>$ for $\delta \in \Delta_h$, where $<1, 0, \delta>(0) = 1, <1, 0, \delta>(1) = 0$ and $<1, 0, \delta>(n+2) = \delta(n)$ whenever $\delta(n)$ is defined. $<1, 1, \delta>$ is defined similarily. We will use this kind of notation freely later.
The definition of $e_{\delta,f}$ is obvious except when δ is the empty function. Then $e_{\delta,f} = <1, 0, e_{\delta,g}>$, i.e. we use the extension in the left adend.

ii) $f = <2, g, h>$. The construction is obvious and trivial in this case, and is left for the reader.

iii) $f = <3, g, h>$. Let Δ_f be the empty function together with all $<3, \delta_1, \delta_2>$ such that $\delta_1 \in \Delta_g$ and $\delta_2 \in \Delta_{h(\delta_1)}$ (i.e. the part of Δ that we can compute from $h(\delta_1)$).
If δ is the empty function then $e_{\delta,f} = <2, e_{\delta,g}, e_{\delta,F_h(e_{\delta,g})}>$
If $\delta = <3, \delta_1, \delta_2>$ let $e_{\delta,f} = <2, e_{\delta_1,g}, e_{\delta_2,F_h(e_{\delta_1,g})}>$

iv) $f = <4, g, h>$. Let Δ_f be the set of finite maps δ satisfying:
* if $\delta(\sigma) = \tau$, and $\sigma \in$ FUN then $\tau \in$ FUN
** δ is consistent and monotone as a partial function on FUN
*** for each $\sigma_1, \ldots, \sigma_k$ of the domain of δ, if $\sigma_1 \cup \ldots \cup \sigma_k \in \Delta_f$ then $\tau_1 \cup \ldots \cup \tau_k \in \Delta_{h(\sigma_1)} \cup \ldots \cup h(\sigma_k)$.
If $\delta \in \Delta_f$ we compute $e_{\delta,f}$ as follows:
Let $\delta = \{(\sigma_1, \tau_1), \ldots, (\sigma_n, \tau_n)\}$

We will construct a continuous function $E_{\delta,f}$ computable in f, and we will let $e_{\delta,f}$ be the corresponding associate.
Let $x \in N^N$. Let $X = \{ k \mid \sigma_i(k) = \sigma_j(k)$ for some $i, j \leq n \}$
Let $Y_x = \{ i \mid$ if $\sigma_i(n)$ is defined then $\sigma_i(n) = x(n) \}$
Let $\pi \in \Delta_{F_h(x)}$ be minimal w.r.t. code numbers such that $\tau_i \subseteq \pi$ for all $i \in Y_x$. Then let $E_{\delta,f}(x) = e_{\pi,F_h(x)}$. We let $e_{\delta,f}$ be an associate for $E_{\delta,f}$ extending δ.

v) $f = <5, g>$. Let Δ_f be the empty function together with all $<5, \delta>$ where $\delta \in \Delta_g$.
If δ is the empty function, we let $e_{\delta,f} = e_{\delta,g}$.
If $\delta = <5, \delta_1>$, we let $e_{\delta,f} = e_{\delta_1,g}$.

Finally we will define $I(f)$ as for the B-hierarchy with one important restriction.
If $f = <4, g, h>$ then $f_1 \in I(f)$ if f_1 is an associate, $F_{f_1}(x) \in I(F_h(x))$ whenever $x \in I(g)$ and finally for all $\delta_1, \ldots, \delta_k$, if $\delta_1 \cup \ldots \cup \delta_k \in \Delta_g$ then $f_1(\delta_1) \cup \ldots \cup f_1(\delta_k) \in \Delta_{h(\delta_1) \cup \ldots \cup h(\delta_k)}$

3.4 Theorem
Let D be the union of all D_α defined above. Then
i) Δ_f is uniformly recursive in f for each $f \in D$
ii) $e_{\delta,f} \in I(f)$ for each $f \in D$ and $\delta \in \Delta_f$
iii) If $f \in D$, if $g \in I(f)$ and if X is a finite set, then there is a subfunction δ of g in Δ_f with $X \subseteq$ domain(δ)

Proof
The proof is tedious, but in most cases trivial.
First we notice that the relation $\delta \in \Delta_f$ is uniformly recursive in f by recursion on the code number of δ. This is trivial.
Then we have to prove that if $f \in D$ and $\delta \in \Delta_f$ then $e_{\delta,f}$ extends δ and is an element of $I(f)$. If δ is the empty function, $e_{\delta,f}$ is defined by recursion over the subtype-ordering, using only the left branch in the case of disjoint unions. By the definition of D this ordering is well-founded, so $e_{\delta,f}$ is well defined. To verify that $e_{\delta,f} \in I(f)$ in this case is trivial.
The rest of the proof is by induction on the code number of δ still with six subcases. The proofs are easy and are left for the reader.

The proof of the density property iii) is primarily by induction on α, by subinduction on the cardinality of X and by subsubinduction on the largest element of X. The only involved case is case 4, where the definition of I is carefully manufactured in order to make the proof work.

In Normann [6] the notion of a Kleene-space was introduced. In a Kleene-space the virtues of the B and D-hierarchies are combined, we identify associates representing the same object and we have full recursive control over the nonempty neighborhoods. Unfortunately we had to give up to include dependent sums; together with recursively defined families of types they seem to break out of the class of Kleene-spaces.

4 *A recursion-scheme for* B_α

We see our type-streams as a minor generalization of types defined by strict positive inductive definitions. One advantage is that the connection between the type and the defining formula is loosened. This gives us the opportunity to view recursion in a more general way. It also makes it easier to define useful types.
In this section we will prove two recursion theorems on purely semantical grounds. It remains to see if they have some natural syntactical or logical counterparts. They seem however to be more general than recursion theorems proved or axiomatised within traditional type theory.
In defining recursion, the type of an object is essential. Thus the notion of recursion will be developed with respect to the set of pairs (f, g) with $g \in B$ and $f \in I(g)$

In this section we will use the symbols $\oplus, \times, \Sigma$ and Π for disjoint unions, cartesian products,dependent sums and dependent products as they were technically defined in section 2. We let N_0 be the initially defined one-point type.

4.1 *Definition*

a) Let P_α be the set of pairs (f, g) where $g \in B_\alpha$ and $f \in I(g)$. Let P be the union of the P_α's
b) Let $(f, g) \in P$.
We define the *support-type* $S(f, g)$ of (f, g) by the following set of equations, of which we take the minimal solution; and simultaneously, to each $x \in S(f, g)$ we define the *predecessor*

$(\rho(f,g,x),R(f,g,x))$

of (f,g) *with index* x giving recursion equations for ρ and R:

$I(g)=N_0$ and $f=^10$:

$S(f,g)=\emptyset$.

There is no need to define ρ and R.

$I(g)=I(g_1)\oplus I(g_2), f=<1,0,f_1>$:

$S(f,g)=N_0\oplus S(f_1,g_1)$.

$x=<1,0,^10>: \rho(f,g,x)=f_1$ and $R(f,g,x)=g_1$.

$x=<1,1,y>: \rho(f,g,x)=\rho(f_1,g_1,y)$ and $R(f,g,x)=R(f_1,g_1,y)$

$I(g)=I(g_1)\oplus I(g_2), f=<1,1,f_1>$:

$S(f,g)=S(f_1,g_2)$.

The definitions of ρ and R are in analogy with the definition above.

$I(g)=I(g_1)\times I(g_2), f=<2,f_1,f_2>$:

$S(f,g)=(N_0\oplus S(f_1,g_1))\oplus(N_0\oplus S(f_2,g_2))$.

The definitions of ρ and R are left for the reader.

$I(g)=\Sigma(I(F_{g_2}(x)\mid x\in I(g_1)), f=<2,f_1,f_2>$:

$S(f,g)=N_0\oplus S(f_2,F_{g_2}(f_1))$

$x=<1,0,^10>: \rho(f,g,x)=f_2$ and $R(f,g,x)=F_{g_2}(f_1)$.

$x=<1,1,y>: \rho(f,g,x)=\rho(f_2,F_{g_2}(f_1),y)$ and

$R(f,g,x)=R(f_2,F_{g_2}(f_1),y)$

$I(g)=\Pi(F_{g_2}(x)\mid x\in I(g_1)), f=<3,h>$:

$S(f,g)=I(g_1)\oplus\Sigma(S(F_h(x),F_{g_2}(x))\mid x\in I(g_1))$

The definitions of ρ and R are left for the reader.

$g=<5,g_1>, f\in I(g)$:

$S(f,g)=S(f,g_1)$, using the $<5,->$ - construction.

The definitions of ρ and R are left for the reader.

4.2 *Theorem*

a) Uniformly recursive in $g\in B_\alpha$ and $f\in I(g)$ there is a $h\in B_\alpha$ such that

$S(f,g)=I(h)$

will satisfy the equations above. Moreover the solution is unique up to

$I(h)$.

b) If $g \in B_\alpha, f \in I(g)$ and $x \in S(f,g)$ then $\rho(f,g,x)$ and $R(f,g,x)$ are well defined, $R(f,g,x) \in B_\alpha$ and $\rho(f,g,x) \in I(R(f,g,x))$.

Proof

a) The defining equations for $S(f,g)$ can be transfered to defining equations for a partial function $F:\mathbf{N}^\mathbf{N} \to \mathbf{N}^\mathbf{N}$ on the indices, and by the recursion theorem they have a solution H.
By induction on the rank of (f,g) as an element of P we prove that $H(f,g) \in B_\alpha$ and that $I(H(f,g))$ is the unique solution to the original equations.
In this construction we are using type-streams in constructing the types $S(f,g)$.

b) Both ρ and R are defined by recursion on the rank of $(f,g) \in P$.
The fact that $(\rho(f,g,x), R(f,g,x) \in P$ for all $x \in S(f,g)$ is proved by induction on the same rank.

4.3 *Remark*

The set P is inductively defined, so if $(f,g) \in P$ then (f,g) has a set of predecessors. This set of predecessors cannot in general be represented as the elements of some type. The type $S(f,g)$ is a type of the same "structure" as the set of predecessors, and $\lambda x(\rho(f,g,x), R(f,g,x))$ mapping $S(f,g)$ onto the true predecessors.

4.4 *Definition*

Let $F:P_\alpha \to B_\alpha$ be continuous and let $h:\mathbf{N}^\mathbf{N} \to \mathbf{N}^\mathbf{N}$ be a partial, continuous function.
We say that h *matches* F *on* $S(f,g)$ if
$h(x) \in I(F(\rho(f,g,x), R(f,g,x)))$ for all $x \in S(f,g)$.

4.5 *Theorem* (Semantical recursion scheme)

Let $F:P_\alpha \to B_\alpha$ be continuous.
Assume that G is continuous (of type 3, in the sense of e.g. Kleene [2]) such that if $(f,g) \in P_\alpha$ and h matches F on $S(f,g)$ then
$G(f,g,h) \in I(F(f,g))$

Then there is a unique continuous function H defined on P_α such that for each $(f,g) \in P_\alpha$:

$$H(f,g) \in I(F(f,g))$$

and

$$H(f,g) = G(f,g,H'(f,g))$$

where

$$H'(f,g)(x) = H(\rho(f,g,x), R(f,g,x))$$

Proof

H' is defined recursively from H so the equation

$$H(f,g) = G(f,g,H'(f,g))$$

has a solution by the recursion theorem.

By induction on the rank of $(f,g) \in P_\alpha$ we then show that $H(f,g) \in I(F(f,g))$ and that $H'(f,g)$ matches F on $S(f,g)$ (a consequence of the induction hypothesis).

Finally by a standard induction on the rank of $(f,g) \in P_\alpha$ we show that $H(f,g)$ is unique. The details are routine.

Theorem 4.5 is general in the sense that we may define H by induction not just knowing H on the immediate predecessors, but on any set of predecessors we may choose. The disadvantage of the result is that it aims at defining H on all of P_α, but for many applications of recursion this will be requiring too much. We will now prove a slightly more general result.

4.6 *Definition*

a) Let X be a subset of P_α, let $K:X \to B_\alpha$ be continuous and let $T:\Sigma(I(K(x)) \mid x \in X) \to \mathbf{N}^\mathbf{N}$ be continuous such that $T_x(f) = T(x,f)$ is a 1-1 map from $I(K(x))$ to $S(x)$.
We say that (X,K,T) is *an inductive system* if
$$(\rho(T_x(y)), R(T_x(y))) \in X$$
for all $x \in X$ and $y \in I(K(x))$

b) If (X,K,T) is an inductive system, $F : X \to B_\alpha$ is continuous, $h:\mathbf{N}^\mathbf{N} \to \mathbf{N}^\mathbf{N}$ is a partial, continuous function and $x \in X$, we say that h *matches* F *on* $K(x)$ if $h(y) \in I(F(\rho(T_x(y)), R(T_x(y))))$ for all $y \in K(x)$.

An inductive system is in a sense a substructure of $P, \{S(f,g)\}_{(f,g) \in P}$ where we replace P by a subset X and we permit a restricted set of

predecessors of x indexed by the type $K(x)$. Via T_x the type $K(x)$ can be embedded in the full set of predecessors of x. The predecessors of x in the view of the system will then both be in X and be true predecessors.

4.7 *Theorem* (Extended semantical recursion scheme)
Let (X, K, T) be an inductive system, $F:X \to B_\alpha$ be continuous.
Assume that G is continuous such that if $x \in X$ and h matches F on $K(x)$, then $G(x,h) \in I(F(x))$.
Then there is a unique continuous function H defined on X such that for each $x \in X$ we have

$$H(x) \in I(F(x))$$

and

$$H(x) = G(x, H'(x))$$

where

$$H'(x) = \lambda y H(\rho(T_x(y), R(T_x(y)))).$$

Proof
The proof is basicly the same as the proof of theorem 4.5, and is left for the reader.

We will give two important examples on how to use theorem 4.7, both concerned with strictly positive inductive definitions.
If $X = \Gamma(X)$ is the least fix-point of a strictly positive inductive definition, we have shown that there is a real g such that $X = I(g)$. We will in the two examples assume that g is the real obtained from this proof.

4.8 *Example*
If Γ is a strictly positive inductive operator, we define the canonical index-set $K_\Gamma(x)$ for the immediate predecessors of $x \in \Gamma(X)$ by recursion on Γ:
$\Gamma(X) = A : K_\Gamma(x) = \varnothing$ (A a constant)
$\Gamma(X) = X : K_\Gamma(x) = N_0$
$\Gamma(X) = \Gamma_1(X) \oplus \Gamma_2(X) : K_\Gamma(<1,0,y>) = K_{\Gamma_1}(y)$ etc.
$\Gamma(X) = \Gamma_1(X) \times \Gamma_2(X) : K_\Gamma(<2,x,y>) = K_{\Gamma_1}(x) \oplus K_{\Gamma_2}(y)$.
$\Gamma(X) = \Sigma(\Gamma_1(y)(X) \mid y \in A) : K_\Gamma(<2,x,y>) = K_{\Gamma_1(x)}(y)$
$\Gamma(X) = \Pi(\Gamma_1(y)(X) \mid y \in A)$:

$$K_\Gamma(<3,x,z>) = \Sigma(K_{\Gamma_1(y)}(F_z(y)) \mid y \in A)$$

We construct an inductive system using $\{ (x, g) \mid x$ is in the least fixpoint of $\Gamma \}$ as X and the function K_Γ. It is trivial to define the map T.

4.9 *Example*
We let Γ and X be as in example 4.8. By a similar construction as the one in 4.8 we may isolate the type of all predecessors of x in X, indexed by K'_Γ. The definition is as in 4.8 with the following changes:
We define $K'_{\Gamma'}$ for all subformulas Γ' of Γ.
In the case $\Gamma'(X) = X$ we let $K'_{\Gamma'}(x) = N_0 \oplus K'_\Gamma(x)$
This gives us a recursion-equation for K'_Γ that can be solved. It is then simple to define the corresponding function T'.

These two examples show that our notion of recursion covers recursion on a set defined by strictly positive induction, both with respect to immediate predecessors and with respect to general predecessors.

5 *The complexity of the hierarchy*

It is clear that there will be an ordinal α such that $B_{\alpha+1} = B_\alpha$.
In this section we will show that this ordinal is the first ordinal not recursive in ${}^3E$ and any $f \in \mathbf{N}^{\mathbf{N}}$.
It is remarkable that our hierarchy goes that far, since it is based on continuity and ${}^3E$ is a highly discontinuous object. ${}^3E$ was introduced by Kleene in [1] and is defined as the following object of type 3:

${}^3E(F) = 0$ if $F(f) = 0$ for all $f \in \mathbf{N}^{\mathbf{N}}$
${}^3E(F) = 1$ if $F(f) \neq 0$ for some $f \in \mathbf{N}^{\mathbf{N}}$.
We assume F to be total.

We will have to assume some familiarity with recursion in ${}^3E$ in this section. In Sacks[7] all properties of ${}^3E$ that we need are proved in the setting of E-recursion.

5.1 *Theorem*
There are two partial functions ρ_1 and ρ_2 both recursive in ${}^3E$ such that
i) $\rho_1(f) = 0 \Leftrightarrow f \in B$
ii) $\rho_2(f, x)\downarrow \Leftrightarrow f \in B$ (where $\downarrow$ means 'is defined')
If $f \in B$ then
$\rho_2(f, x) = 1$ if $x \in I(f)$

$\rho_2(f,x) = 0$ if $x \notin I(f)$

Proof
We define ρ_1 and ρ_2 simultaneously using the recursion theorem for ^3E. We describe informally the algorithms represented by ρ_1 and ρ_2 and leave for the reader to verify that these algorithms will work.

Computing ρ_1: Given f that looks like an index for a type A, we may start to develope the tree of subtypes of A. If we in that tree find a function that is not on one of the basic forms of the indices, we reject f, e.g. let $\rho_1(f) = 1$. If we in that tree find a function looking like the index of a dependent sum or a dependent product, we use ρ_1 to check if the domain really is one of our types, and then ρ_2 to compute the branching at the respective node.

Computing ρ_2: If ρ_1 accepts f, and x is given, we give a top-down construction of the tree of pairs (y,g) where y is a predecessor of x and g is the index of the corresponding subtype. If we somewhere in this tree find that y is not on the basic form of the elements of $I(g)$, we reject x. Otherwise we accept x as an element of $I(f)$ if and only if the tree constructed above is wellfounded. We may use ^3E to test this.

To complete the proof we must show by induction on α that if $f \in B_\alpha$ then $\rho_1(f) = 0$, and by induction on the length of the computation that if $\rho_1(f) = 0$ then $f \in B$ and $\rho_2(f,-)$ is as desired. There will be no surprises in any of the proofs.

Of course this theorem is valid for the two other hierarchies we defined in section 3. We will now prove the converse of theorem 5.1, i.e that our hierarchy captures the full complexity of recursion in ^3E. We will not need non-wellfounded types except for the empty type in this argument, and we may even avoid using dependent sums.
This proof can be adjusted to the E-hierarchy, but the empty type is essential, and the proof does not work for the D-hierarchy. We will leave an estimation of the complexity of the D-hierarchy as an open problem.
In fact the proof will work for many natural semantics for type theory. Of course the type corresponding to $\mathbf{N}^{\mathbf{N}}$ must essentially be interpreted as $\mathbf{N}^{\mathbf{N}}$. Moreover we will need the empty type, and we will need that there is a uniform function mapping the empty type represented by any term into

the empty type. Finally the semantics must accept that the constructions we perform are uniform. It is likely that if we are given a natural basis for a semantics for type theory, and if families of types constructed by recursion over some inductively defined type is a dependent family in the sense of the semantics, then the families constructed in this proof will also be dependent families in the sense of the semantics. This general claim can of course not be supported by a precise proof.

In the rest of this paper we will let $\mathbf{f}$ denote a finite sequence from $\mathbf{N}$ and $\mathbf{N}^{\mathbf{N}}$.

5.2 *Theorem*
Let $\{e\}(f)$ denote the Kleene-computation relative to ^3E with index e and input f.
Then uniformly recursive in e, f we can define
i) an object $T(e,f) \in \mathbf{N}^{\mathbf{N}}$
ii) a partial continuous function $t(e,f):\mathbf{N}^{\mathbf{N}} \to \mathbf{N}$
such that

$$\{e\}(f)\downarrow \Leftrightarrow T(e,f) \in B \text{ and } I(T(e,f)) \neq \varnothing$$

and when $\{e\}(f)\downarrow$ then $t(e,f)$ is defined and constant $\{e\}(f)$ on $I(T(e,f))$

Proof
We construct $T(e,f)$ and $t(e,f)$ using the recursion theorem.
There will be nine cases following Kleene's schemata S1 - S9. We will omit case 5, primitive recursion, since this can be reduced to the other cases. We will use terms for types in the proof, though it is to be understood that we use the coded versions in the B-hierarchy.

1. $\{e\}(x,f) = x+1$
   $T(e,x,f) = \mathbf{N}_0$ (The canonical one-point type)
   $t(e,x,f)(y) = x+1$

2. $\{e\}(f) = q$
   $T(e,f) = \mathbf{N}_0$
   $t(e,f)(y) = q$

3. $\{e\}\{x,f) = x$

$T(e,x,f) = \mathbf{N}_0$
$t(e,x,f)(y) = x$

4. $\{e\}(f) = \{e_1\}(\{e_2\}(f),f)$
$T(e,f) = \Sigma(T(e_1,t(e_2,f)(x),f) \mid x \in T(e_2,f))$
$t(e,f)(<x,y>) = t(e_1,t(e_2,f)(x),f)(y)$

6. $\{e\}(f) = \{e_1\}(\sigma(f))$ where σ is a permutation.
$T(e,f) = T(e_1,\sigma(f))$, where we use the $<5,->$-construction.
$t(e,f)(y) = t(e_1,\sigma(f))(y)$

7. $\{e\}(x,f,f) = f(x)$
$T(e,x,f,f) = \mathbf{N}_0$
$t(e,x,f,f)(y) = f(x)$

8. $\{e\}(f) = {}^3E(\lambda f\{e_1\}(f,f))$
Let $T_1(f,y)$ be $\mathbf{N}_0$ if $t(e_1,f,f)(y) > 0$, and let it be empty otherwise.
Let $T_2(f,y)$ be $\mathbf{N}_0$ if $t(e_1,f,f)(y) = 0$, and let it be empty otherwise.
Let
$T(e,f) = \Pi((T(e_1,f,f) \to \varnothing) \to \varnothing) \mid f \in \mathbf{N}^{\mathbf{N}}) \times$
$[(\Sigma(\Sigma(T_1(f,y) \mid y \in T(e_1,f,f)) \mid f \in \mathbf{N}^{\mathbf{N}}) \oplus$
$(\Pi(\Pi(T_2(f,y) \mid y \in T(e_1,f,f)) \mid f \in \mathbf{N}^{\mathbf{N}})]$
Let $t(e,f)(<x,r(y)>) = 1$
$t(e,f)(<x,l(y)>) = 0$

9. $\{e\}(e_1,f) = \{e_1\}(f)$
$T(e,e_1,f) = T(e_1,f)$, where we use the $<5,->$-construction.
$t(e,e_1,f)(y) = t(e_1,f)(y)$

Claim 1
If $\{e\}(f)\downarrow$ then $T(e,f) \in B$ and $I(T(e,f)) \neq \varnothing$.
Moreover, $t(e,f)$ is constant $\{e\}(f)$ on $I(T(e,f))$.

Proof
We use induction on the length of the computation $\{e\}(f)$. All cases but two are trivial.

In case 4 we observe that $I(T(e,f))$ will be the cartesian product of $I(T(e_1,f))$ and $I(T(e_2,\{e_1\}(f),f))$, which will be nonempty. Moreover $t(e,f)$ will be constant $t(e_2,\{e_1\}(f),f))$ on $I(T(e,f))$, which is the right value.

In case 8 we first show that $I(\Pi((T(e_1,f,f)\to\emptyset)\to\emptyset)\mid f\in \mathbf{N}^{\mathbf{N}}))$ is nonempty. But since $I(T(e_1,f,f))$ is nonempty by the induction hypothesis, $I(\Pi((T(e_1,f,f)\to\emptyset)\to\emptyset)\mid f\in \mathbf{N}^{\mathbf{N}}))$ will contain all associates for partial continuous functionals.
If $\{e\}(f)=1$ there is an $f\in \mathbf{N}^{\mathbf{N}}$ such that $\{e_1\}(f,f)>0$. For this f, $T(e_1,f,f)$ is nonempty and $T_1(f,y)=\mathbf{N}_0$ while $T_2(f,y)$ is empty for $y\in T(e_1,f,f)$. This shows that $\Sigma(\Sigma(T_1(f,y)\mid y\in T(e_1,f,f))\mid f\in \mathbf{N}^{\mathbf{N}})$ is nonempty while $\Pi(\Pi(T_2(f,y)\mid y\in T(e_1,f,f))\mid f\in \mathbf{N}^{\mathbf{N}})$ is empty. Thus the construction works in this case.
If $\{e\}(f)=0$ we see that $\Sigma(\Sigma(T_1(f,y)\mid y\in T(e_1,f,f))\mid f\in \mathbf{N}^{\mathbf{N}}$ is empty. On the other hand $T_2(f,y)=\mathbf{N}_0$ for all $f\in \mathbf{N}^{\mathbf{N}}$ and $y\in T(e_1,f,f)$. Then $\Pi(\Pi(T_2(f,y)\mid y\in T(e_1,f,f))\mid f\in \mathbf{N}^{\mathbf{N}})$ is nonempty, and the construction still works.

Claim 2
If $T(e,f)\in B_\alpha$ for some α and $y\in I(T(e,f))$, then $\{e\}(f)\downarrow$.

Proof
We use induction on α and subinduction on the rank of $y\in I(T(e,f))$ (in the inductive definition of $\{I(h)\}_{h\in B_\alpha}$)
Again there will be nine cases, following S1 - S9, where case 5 is omitted and cases 1 , 2 , 3 and 7 are trivial.
Cases 6 and 9: We use that the ranks of the elements in $I(<5,f>)$ are one higher than the ranks of the same objects as elements of $I(f)$.

Case 4: If $I(T(e,f))\neq\emptyset$, then $I(T(e_2,f))\neq\emptyset$. Moreover, if $T(e,f)\in B_\alpha$ then $T(e_2,f)\in B_\beta$ for some $\beta<\alpha$. By the induction hypothesis $\{e_2\}(f)\downarrow$ and then $\{e_2\}(f)=t(e_2,f)(y)$ for any $y\in I(T(e_2,f))$.
Now let $<y,x>\in I(T(e,f))$. Then $x\in(T(e_1,t(e_2,f)(y),f)$ with lower rank, and by the subinduction hypothesis $\{e_1\}(\{e_2\}(f),f)\downarrow$, so $\{e\}(f)\downarrow$.

Case 8 : Assume that $T(e,f)\in B_\alpha$. Then for all $f\in \mathbf{N}^{\mathbf{N}}$ we have that

$T(e_1, f, f) \in B_\beta$ for some $\beta < \alpha$, since it occurs negatively in the definition. Assume further that $I(T(e, f)) \neq \emptyset$.
Then in particular $I(\prod((T(e_1, f, f) \to \emptyset) \to \emptyset) \mid f \in N^N))$ is nonempty. It follows that for all $f \in N^N$, $I(T(e_1, f, f))$ is nonempty. By the induction hypothesis $\{e_1\}(f, f)\downarrow$ for all $f \in N^N$ so $\{e\}(f)\downarrow$.

This ends the proof of theorem 5.2.

In the proof of Theorem 5.2 we did not use the non-wellfounded types at all, it was sufficient to use a subhierarchy of well-founded types constructed by finite or continuous infinite sums and products. It is in fact possible to prove this theorem avoiding infinite sums. The reason is that in the proofs we did not use the sums themselves, only the facts that they were nonempty. The following observation shows that we everywhere in the construction can replace the use of infinite sums with a use of infinite products:

5.3 *Lemma*
Let $f \in B_\alpha$ and let for each $x \in I(f)$, let $g_x \in B_\alpha$ be continuous in x. Then there is a type P using only products such that

$$P \neq \emptyset \Leftrightarrow \Sigma(I(g_x) \mid x \in I(f)) \neq \emptyset$$

Proof
Let $P = \prod(I(g_x) \to \emptyset \mid x \in I(f)) \to \emptyset$

We leave the verification of the desired property of P for the reader.

6 *Second order type-theory, a possible semantics*

In section 2 we defined a model for types with induction, and in section 5 we showed the correspondence between this model and recursion in ^3E. In this section we will extend the hierarchy so that we also capture types constructed by universal quantification over a family of types parameterised over all types. The idea is to capture

$$\bigcap_{X \text{ type}} \Gamma(X)$$

for reasonably uniform Γ. Using the same methods it is possible to model other forms of second order constructions, e.g. the set of uniform functions which to X gives an element of $\Gamma(X)$.

First we notice that the constructions of section 2 and the proofs of section 5 can be relativised to any $X \subseteq \mathbf{N}^{\mathbf{N}}$, i.e. we may let X represent some ground type with fixed index and then uniformly recursive in X and ${}^3E$ we may generate types from X and compute the content of each type. It is even so that uniformly in X this hierarchy is equivalent to recursion in $X, {}^3E$. This is a consequence of the following lemma, left without proof:

6.1 *Lemma*
Let $X \subseteq \mathbf{N}^{\mathbf{N}}, T(n) \subseteq \mathbf{N}^{\mathbf{N}}$ be nonempty for each $n \in \mathbf{N}$, and let $t(n, y)$ be a continuous function such that $t_n(y) = t(n, y)$ is constant on each $T(n)$. Let $y_n \in T(n)$ for each n.
Then using a uniform type-constructor we find independently of the choice of y_n a set $S \neq \emptyset$ and a function s such that s is constant 1 on S if $\lambda nt(n, y_n) \in X$, and s is constant 0 on S otherwise.

Our objective is to isolate a class of type-operators Γ that is sufficiently general to be useful as a model of second order type theory, but where the relation 'X type $\Rightarrow \Gamma(X)$ type' is verified by some object recursive in ${}^3E$. Lemma 6.1 then indicates that operators of the form

$$\Gamma_f(X) = I(f, X)$$

for f where $f \in B(X)$ for all $X \subseteq \mathbf{N}^{\mathbf{N}}$, are not uniform enough, since the class of such Γ_f will not even be semirecursive in ${}^3E$.
If we try to restrict ourselves to those Γ_f where for some fixed α recursive in ${}^3E$ and for all X we have $f \in B_\alpha(X)$ we may still find that the intersection of the sets $I(f, X)$ will not be semirecursive in ${}^3E$.

Our solution will more or less be that we require $\Gamma_f(X)$ to be a type within any reasonable countable model $(\mathcal{A}; X)$, and for g to be in $\Lambda X \Gamma(X)$ we require that $\mathcal{A} \models g \in \Gamma(X)$ for the same class of models. We will essentially restrict ourselves to sets X which are recursive in ${}^3E$ and some $f \in \mathbf{N}^{\mathbf{N}}$.

6.2 *Definition*
A *model* is a structure

$$\mathcal{A} = \langle \mathbf{N}, A \rangle$$

where A is a countable subset of $\mathbf{N}^{\mathbf{N}}$, such that
i) $\mathcal{A} \models A$ is closed under recursion in ${}^3E$.
ii) If $X \subseteq A$ is recursive in ${}^3E^{\mathcal{A}}$ and some $f \in A$ and if

$\mathcal{A} \models$ X wellfounded, then X is wellfounded.

6.3 *Lemma*
We may code the models as elements of $\mathbf{N}^{\mathbf{N}}$ such that the relation

f codes a model $\mathcal{A}$

is recursive in ^3E.

Proof
The proof is trivial and the complexity can actually be seen to be Π^1_2.

6.4 *Definition*
Let $\mathcal{A}$ be a model.
We say that $\mathcal{A}$ is a ^3E *-model* if for all $e, m \in \mathbf{N}$ and for all $f_1, \ldots, f_k \in A$ we have

$\{e\}(^3E, f_1, \ldots, f_k) = m \Rightarrow \mathcal{A} \models \{e\}(^3E, f_1, \ldots, f_k) = m.$

If α is an ordinal, we say that $\mathcal{A}$ is a $^3E, \alpha$ *-model* if for all $e, m \in \mathbf{N}$ and for all $f_1, \ldots, f_k \in A$ we have

$\{e\}(^3E, f_1, \ldots, f_k) = m$ with the length of computation bounded by α
$\Rightarrow \mathcal{A} \models \{e\}(^3E, f_1, \ldots, f_k) = m.$

We observe that the relation

f is a code for a ^3E-model

is uniformly co-r.e. in ^3E, and that the relation

f is a code for a $^3E, \alpha$ - model

is uniformly recursive in ^3E and α.

6.5 *Lemma*
Let $\mathcal{A}$ be a ^3E - model
Then the ^3E - computations relative to elements f of A^n will form an initial segment of the $\mathcal{A}$ - computations relative to $^3E^{\mathcal{A}}$ under the 'length-of-computation'- prewellordering. We call this initial segment the *standard* computations in $\mathcal{A}$.

Proof
Assume that $\{e\}(^3E, f) = m$ and let $\{d\}(^3E, g)$ be any computation, where f and g are sequences from $\mathcal{A}$.
Since ^3E may decide if $\langle d, g \rangle \leq \langle e, f \rangle$ by a convergent computation, the answer will be the same in $\mathcal{A}$ as in the real world.

We are now ready to give the main definition:

6.6 *Definition*
Let $B \subseteq \mathbf{N}$ be finite, $B = \{ i_1 , \ldots , i_k \}$.
Let $\{ X_i \}_{i \in B}$ be a family of subsets of $\mathbf{N}^{\mathbf{N}}$, each X_i recursive in ${}^3E$ and some f_i.
We define the set $S(\{ X_i \}_{i \in B})$ and for each $f \in S(\{ X_i \}_{i \in B})$ we define the set $I(f ; \{ X_i \}_{i \in B})$.
Simultaneously we use the recursion theorem for ${}^3E$ to find indices for recursive functions ρ_1 and ρ_2 such that

$f \in S(\{ X_i \}_{i \in B}) \Leftrightarrow \rho_1({}^3E , f , \{ X_i \}_{i \in B} , \{ f_i \}_{i \in B}) = 0$

and if $f \in S(\{ X_i \}_{i \in B})$ then $\lambda g \rho_2(g , f , {}^3E , \{ X_i \}_{i \in B} , \{ f_i \}_{i \in B})$ is the characteristic function of $I(f ; \{ X_i \}_{i \in B})$.

Now for the definition:
The set $S(\{ X_i \}_{i \in B})$ has the closure properties of the B-hierarchy, and in addition two more:

$< 6 , i , {}^1 0 > \in S(\{ X_i \}_{i \in B})$ for $i \in B$ and then
$I(< 6 , i , {}^1 0 > ; \{ X_i \}_{i \in B}) = X_i$.
$< 7 , m , f > \in S(\{ X_i \}_{i \in B})$ if $m \notin B$ and for all ${}^3E$ - models $\mathcal{A}$ with $f \in A$ and $f_i \in A$ for all $i \in B$ we have that
$\mathcal{A} \models \forall X$(X recursive in ${}^3E$ and some $g \rightarrow f \in S(\{ X_i \}_{i \in B'})$, and the length of the computation $\rho_1({}^3E , \{ X_i \}_{i \in B'} , \{ f_i \}_{i \in B'})$ in $\mathcal{A}$ is bounded by some standard computation in $\mathcal{A}$), where $B' = B \cup \{ m \}$
$X_m = X$ and $f_m = g$.
(In order to be formally correct we should replace X_i by $X_i \cap A$ in the definition above. This will in a sense be the interpretation of X_i in $\mathcal{A}$.)

We let $h \in I(< 7 , m , f > ; \{ X_i \}_{i \in B})$ if there is a finite set $Y \subseteq \mathbf{N}^{\mathbf{N}}$ such that for all ${}^3E$ -models $\mathcal{A}$ with $Y \subseteq A$, $h \in A$, $f \in A$ and $f_i \in A$ for all $i \in B$ and for all $X \subseteq A$ recursive in ${}^3E$ and some $g \in A$ in the sense of $\mathcal{A}$ we have that
$\mathcal{A} \models \rho_2(h , f , X , \{ X_i \}_{i \in B} , \{ f_i \}_{i \in B}) = 1$ with a computation bounded by some standard computation in $\mathcal{A}$.
The definition of ρ_1 in this case is easy, we notice that the co-r.e. statement that $\mathcal{A}$ is a ${}^3E$-model occurs negatively and that the rest is trivially r. e. in ${}^3E$. In order to show how to compute ρ_2 we must notice that there is a uniform bound on the length of the standard parts we need in order to

compute the values $\rho_1(f, \{X_i\}_{i\in B'}, \{f_i\}_{i\in B'})$ in the various models $\mathcal{A}$ and for the various subsets X and elements g of A. Moreover we can compute an upper bound for the length of $\{\rho_2(h, f, \{X_i\}_{i\in B}, \{f_i\}_{i\in B})\}_{h\in N^N}$ when the length of $\rho_1(f, \{X_i\}_{i\in B}, \{f_i\}_{i\in B})$ is known. We use this to find an α such that we can replace the use of ^3E-models in the definition of $h \in I(<7, m, f>; \{X_i\}_{i\in B})$ with $^3E, \alpha$ - models.

We will not prove that we have constructed a model for any second order type theory, since we are not concerned with formal theories here. We will however formulate a few lemmas that might be helpful in such a venture.

First we should notice that although we have used the notation $S(X_1, \ldots, X_n)$, the set is dependent on the choice of $f_1, \ldots, f_n$. There seem to be no natural way to avoid this breach of elegancy. The definition of I in case 7 is however designed so that the value of $I(f; X_1, \ldots, X_n)$ will be independent of the choice of $f_1, \ldots, f_n$. We leave the verification of this for the reader.

6.7 Lemma

If $f \in S(\{X_i\}_{i\in B})$ then for all ^3E - models $\mathcal{A}$ with $f, f_i \in A$ we have that $\mathcal{A} \models f \in S(\{X_i\}_{i\in B})$ by a computation in the standard part, and we have $g \in I(f; \{X_i\}_{i\in B}) \Leftrightarrow \mathcal{A} \models g \in I(f; \{X_i\}_{i\in B})$ for all such $\mathcal{A}$ with $g \in A$.

Proof

Immediate from the definition

This means that if Γ is a type and X is a type variable not occurring in Γ then $\Lambda X.\Gamma$ is a type with the same elements as Γ.

6.8 Lemma

a) Let f and $\{X_i\}_{i\in B}$ be given, where X_i is recursive in ^3E and f_i. If for all ^3E - models $\mathcal{A}$ and for all $X_m \subseteq A$ recursive in $^3E^{\mathcal{A}}$ and some $g \in A$ we have $\mathcal{A} \models f \in S(\{X_i\}_{i\in B\cup\{m\}})$ then $f \in S(\{X_i\}_{i\in B\cup\{m\}})$ for all $X_m \subseteq N^N$ that are recursive in ^3E and some g.

b) If $f, \{X_i\}_{i\in B}$ are as above, if $Y \subseteq N^N$ is finite and if $\mathcal{A} \models g \in I(f; \{X_i\}_{i\in B\cup\{m\}})$ for all $\mathcal{A}, X_m$ as above with $Y \subseteq A$, then

$g \in I(f; \{X_i\}_{i \in B \cup \{m\}})$ for all $X_m \subseteq \mathbf{N}^{\mathbf{N}}$ that are recursive in ${}^3E$ and some h.

Proof

Both a) and b) are proved by a simple Löwenheim-Skolem argument.

6.9 *Lemma*

Let $\mathcal{A}$ be a ${}^3E$ - model.

Let $\mathcal{B}$ be a model with a code in $\mathcal{A}$.

If $\mathcal{A} \models \mathcal{B}$ is a ${}^3E$ - model, then $\mathcal{B}$ is a ${}^3E$ - model.

The proof is trivial.

6.10 *Lemma* (Substitution)

There is a recursive function ρ with the following property:

Let $B \subseteq \mathbf{N}$ and $C \subseteq \mathbf{N}$ be finite sets.

Let $\{Z_j\}_{j \in C}$ be a family of subsets of $\mathbf{N}^{\mathbf{N}}$, Z_j recursive in ${}^3E$ and u_j.

Assume that $f_i \in S(\{Z_j\}_{j \in C})$ for each $i \in B$, with $Y_i = I(f_i; \{Z_j\}_{j \in C})$.

Y_i will be recursive in f_i and $\{u_j\}_{j \in C}$, and this will be the representation we will use below.

For technical reasons we will assume that f_i is not the canonical code for the inductive type $\mu X(X = X)$.

Assume further that $g \in S(\{Y_i\}_{i \in B})$ with $X = I(g; \{Y_i\}_{i \in B})$.

Then $h = \rho(g, B, C, \{f_i\}_{i \in B}) \in S(\{Z_j\}_{j \in C})$ and
$X = I(h; \{Z_j\}_{j \in C})$

Proof

We will construct ρ using the recursion theorem. ρ will be constructed so that g will be recursive in $\rho(B, C, \{f_i\}_{i \in B}, g), {}^3E$ and if $<6, i, {}^10>$ occur in the subtype-tree of g, then f_i is recursive in $\rho(g, B, C, \{f_i\}_{i \in B}), {}^3E$.

What ρ does is rewriting g substituting f_i for $<6, i, {}^10>$. In order to recover g we must be able to tell where in the subtype-tree of $\rho(g, B, C, \{f_i\}_{i \in B})$ a substitution has been made. The trick used for that will be useful when we want to compute f_i from $\rho(g, B, C, \{f_i\}_{i \in B}), {}^3E$ as well.

We assumed that f_i is not the canonical code for the empty type, and we

may as well assume that f_i is not of the form $< 5 , f >$.
In the proof we will let $< 5^n , f >$ denote $< 5 , < 5 , \ldots , < 5 , f > \ldots > >$ with n occurrences of 5.
Let T_g be the tree of subtypes of g.
If $< 5^n , f >$ is a node in T_g we call it *maximal* if $< 5^{n+1} , f >$ is not a node and f is not on the form $< 5 , f' >$. We call n the *length* of this node, the length will in many cases be 0.
Using primitive recursion we may rewrite g to a recursively equivalent g' such that a maximal $< 5^n , f >$ is replaced by

a maximal $< 5^{2n+1} , f' >$ if $f = < 6 , i , {}^1 0 >$ for some $i \in B$

a maximal $< 5^{2n} , f' >$ if $f \neq < 6 , i \, {}^1 0 >$ for all $i \in B$,

and such that $I(g' ; \{ Y_i \}_{i \in B}) = I(g ; \{ Y_i \}_{i \in B})$.
Below we will assume that g is obtained from such a shift.
ρ is actually going to be primitive recursive. The definition is divided into cases, following the type-constructors. In all cases but two ρ will simply commute with the top type-constructor in g. This means in particular that ρ will preserve maximality and commute with 5^n.
The two remaining cases will be

$\rho(< 6 , i , {}^1 0 > , B , C , \{ f_i \}_{i \in B}) = f_i$ for $i \in B$.

$\rho(< 7 , m , f > , B , C , \{ f_i \}_{i \in B} > = < 7 , m_1 , \rho(f , B_1 , C_1 , \{ f_i \}_{i \in B_1}) >$
where m_1 is the least ordered pair $< m , m_2 >$ not in C, $B_1 = B \cup \{ m \}$, $C_1 = C \cup \{ m_1 \}$ and $f_m = < 6 , m_1 , {}^1 0 >$.

We can obtain g from $\rho(g , B , C , \{ f_i \}_{i \in B})$ by reversing the recursive definition of ρ. From the length of the maximal nodes we can tell when the recursion is to be continued and when we shall insert $< 6 , i , {}^1 0 >$.
In case 7 we defined m_1 so that m is uniformly primitive recursive in m_1. Thus g is actually primitive recursive in $\rho(g , B , C , \{ f_i \}_{i \in B})$.
To recover f_i is a bit more delicate. From ${}^3E$ and $\rho(g , B , C , \{ f_i \}_{i \in B})$ we may compute the subtype tree $T_{\rho(g , B , C , \{ f_i \}_{i \in B} ,)}$ of $\rho(g , B , C , \{ f_i \}_{i \in B} ,)$. We may also compute the set $\{ f \mid \exists n(< 5^{2n+1} , f >$ is a maximal node in $T_{\rho(g , B , C , \{ f_i \}_{i \in B} ,)})$ This set is finite, so the elements can be computed using ${}^3E$. This will be the f_i's.

We will now prove that if $g \in S(\{ Y_i \}_{i \in B})$ and $f_i \in S(\{ Z_j \}_{j \in C})$ then $\rho(g , B , C , \{ f_i \}_{i \in B}) \in S(\{ Z_j \}_{j \in C})$.

Simultaneously we will prove that $I(\rho(g, B, C, \{f_i\}_{i\in B}); \{Z_j\}_{j\in C}) = X$.
We will use induction on the length of the verification of $g \in S(\{Y_i\}_{i\in B})$.
The proof can also be carried out inside any ^3E-model $\mathcal{A}$ and a part of the induction hypothesis at α will be that the claim holds within $\mathcal{A}$ for the standard computations bounded by α in the real world.

For both, case 7 is the only nontrivial case.
So let $X = I(<7, m, f>; \{Y_i\}_{i\in B})$ and
$h = <7, m_1, \rho(f, B_1, C_1, \{f_i\}_{i\in B_1})>$.
We first show that $h \in S(\{Z_j\}_{j\in C})$. Let $\mathcal{A}$ be a ^3E - model with $h \in \mathcal{A}$, $u_j \in \mathcal{A}$ for $j \in C$, and let X_{m_1} be $\mathcal{A}$-recursive in ^3E and a $f' \in A$. Then f and the f_i's will be elements of $\mathcal{A}$ since they are recursive in h, ^3E at a finite level. We now use the induction hypothesis inside $\mathcal{A}$, so

$$\mathcal{A} \models \rho(f, B_1, C_1, \{f_i\}_{i\in B_1}) \in S(\{Z_j\}_{j\in C} \cup \{X_{m_1}\})$$

This is exactly what is required for h to be in $S(\{Z_j\}_{j\in C})$.

We can use the induction hypothesis inside $\mathcal{A}$ since we have to verify that $f \in S(\{Y_i\}_{i\in B_1})$ with $Y_m = X_{m_1}$ holds in $\mathcal{A}$ bounded by a standard computation.

We will now prove that

$$I(h; \{X_j\}_{j\in C}) = I(<7, m, {}^10>; \{Y_i\}_{i\in B})$$

by proving the inclusion both ways. We will use the special notation introduced above.

Let $g \in I(h; \{X_j\}_{j\in C})$. Let Y be finite such that for all ^3E-models $\mathcal{A}$ with $Y \subseteq A$, $u_j \in A$ for $j \in C$, $g \in A$ and $h \in A$, and for all relevant X_{m_1} we have

$$\mathcal{A} \models g \in I(\rho(f, B, C, \{f_i\}_{i\in B_1}); \{X_j\}_{j\in C_1})$$

By the induction hypothesis in $\mathcal{A}$, and letting $Y_m = X_{m_1}$, we have

$$\mathcal{A} \models g \in I(f; \{Y_i\}_{i\in B_1}).$$

Let $Y' = Y \cup \{u_j\}_{j\in C}$.
If $\mathcal{A}$ contains Y', g and $\{f_i\}_{i\in B}$ we are in the situation above. This shows that $g \in I(<7, m, f>; \{Y_i\}_{i\in B})$.

Conversely, let $g \in I(<7, m, f>; \{Y_i\}_{i\in B})$.
Let Y be finite such that for all models $\mathcal{A}$ containing Y, $\{f_i\}_{i\in B}$,

$\{u_j\}_{j\in C}$, g and f and for all relevant Y_m we have

$\mathcal{A} \models g \in I(f; \{Y_i\}_{i\in B_1})$.

Using the induction hypothesis inside $\mathcal{A}$ we have that

$\mathcal{A} \models g \in I(\rho(f, B_1, C_1, \{f_i\}_{i\in B_1}); \{X_j\}_{j\in C_1})$

with $X_{m_1} = Y_m$ and as above $f_m = <6, m_1, {}^1 0>$.

Using the same Y we see that this demonstrates that

$g \in I(\rho(f, B, C, \{f_i\}_{i\in B}); \{X_j\}_{j\in C})$.

This ends the proof of the substitution lemma.

REFERENCES

[1] Kleene, S. C.: Recursive functionals and Quantifiers of finite types, T.A.M.S. 91 (1959), 1 - 52

[2] Kleene, S. C.: Countable functionals, in A. Heyting (ed.) Constructivity in mathematics, North-Holland (1959), 81-100

[3] Kreisel, G. : Interpretation of analysis by means of functionals of finite type, in A. Heyting (ed.) Constructivity in mathematics, North-Holland (1959), 101 - 128.

[4] Normann, D. : Recursion on the Countable Functionals, Springer Lecture Notes in Mathematics 811, Springer-Verlag (1980).

[5] Normann, D. : Characterizing the continuous functionals, JSL 48 (1983), 965-969.

[6] Normann, D. : Kleene-spaces,In Ferro & al. Logic Colloquium '88 Elsevier (1989), 91-109.

[7] Sacks, G. E. : Higher Recursion Theory, Springer-Verlag (1990).