Computability over the partial continuous functionals

Dag Normann [*]

February 5, 1998

Abstract

We show that to every recursive total continuous functional Φ there is a representative Ψ of Φ in the hierearchy of partial continuous functionals such that Ψ is $S1 - S9$ computable over the hierarchy of partial continuous functionals. Equivalently, the representative Ψ will be PCF-definable over the partial continuous functionals, where PCF is Plotkin's programming language for computable functionals.

1 Introduction

In [6] Kleene extended the concept of relative computations to computations relative to objects of arbitrarily finite pure types. He defined the relation $\{e\}(\phi_1, \ldots, \phi_n) \approx k$ inductively using 9 clauses generally known as $S1 - S9$. The functionals in Kleene's hiererchy are total, $Tp(0)$ is the set $\mathbb{N}$ of natural numbers, and $Tp(k+1)$ will consist of all total functions $\phi : Tp(k) \to \mathbb{N}$.
In [7] Kleene isolated some of these functionals as *countable.* $\phi : Tp(k) \to \mathbb{N}$ is countable if the global action of ϕ on countable inputs can be coded in a countable way via the *associates.*
Kleene showed that any functional $S1 - S9$-computable in countable functionals will itself be countable.

[*]Department of Mathematics, The University of Oslo, P.O. Box 1053, Blindern, N-0316 Oslo, Norway. e-mail: dnormann@math.uio.no

Simultanously and independently Kreisel [8] introduced the hierarchy of continuous functionals. The continuous functionals is a hierarchy (we restrict to the pure case here) $\{Ct(k)\}_{k\in\mathbb{N}}$ where ϕ is in $Ct(k+1)$ if $\phi : Ct(k) \to \mathbb{N}$ in some continuous way.
The main difference beteween Kleene's hierarchy and Kreisel's hierarchy is that Kleene's hierarchy is not extensional. If we take the hereditarily extensional collapse of the countable functionals we get the continuous functionals. From now on we will work with the extensional variant, i.e. the continuous functionals.

Both the original definitions had some ad hoc features, but experience has shown (via a number of characterisations) that the hierarchy constructed is the natural choice for a hierarchy of functionals of pure types where application is determined from finitary information.

Ershov [4] characterised the continuous functionals as the hereditarily total objects in a hierarchy of partial continuous functionals. In essence he used domains to define the continuous functionals. With the now established theory of algebraic domains, or Scott-Ershov domains, see Stoltenberg-Hansen & al. [16] for a general introduction, it seems natural to use domain theory as the framework for constructing the continuous functionals. This approach is taken in Normann [12], where $S1 - S9$ is also given. Berger [2, 3] use the smooth theory of domains to discuss the concept of totality in an abstract setting and to establish the Kreisel-Lacombe-Shoenfield theorem for higher types.

The introduction of the continuous functionals via domain theory actually gives us three hierarchies to consider, the partial continuous functionals $\{P(k)\}_{k\in\mathbb{N}}$, the hereditarily total objects in $P(k)$ forming the hierarchy $\{T(k)\}_{k\in\mathbb{N}}$ (which is not extentional) and the hierarchy $\{Ct(k)\}_{k\in\mathbb{N}}$ where we identify hereditarily extentionally equal objects. The Kleene Schemes $S1-S9$ make sense for both the hierarchies $\{Ct(k)\}_{n\in\mathbb{N}}$ and $\{P(k)\}_{n\in\mathbb{N}}$. The requirements for termination is however more restricted over the Ct-hierarchy compared to the P-hierarchy. In the scheme for functional application

$$\phi(\lambda\xi\{e\}(\xi,\phi,-))$$

we will require $\lambda\xi\{e\}(\xi,\phi,-)$ to be total when we compute in the Ct-hierarchy, while it may be partial when we compute in the P-hierarchy. We might consider $S1 - S9$ over the T-hierarchy as well, but since every computation in this case will respect extensional equality, this is essentially the same as computing over the Ct-hierarchy.

The higher type functionals computable over the Ct-hierarchy and the P-hierarchy will actually not be the same. Berger [2] observed that the fan functional, shown by Tait [17] not to be computable, is indeed computable if the computations take place in the P-hierarchy. By a straitforward application of the recursion theorem, we can show that the functional Γ introduced by Gandy and showed by Hyland [5] not to be computable in the fan functional, is computable over the P-hierarchy.
Berger [2] conjectured that any recursive functional is computable in the P-hierarchy. The main result of this paper is that this conjecture is true.

The recursion theory of $\{P(k)\}_{k\in\mathbb{N}}$ has also atracted the interest of theoretical computer scientists. Plotkin [14] defines a programming language PCF based on typed λ-calculus with local fixpoint operators at each type. His type structure has two ground types, the natural numbers and the boolean values, and it is closed under the formation of function types. Using algebraic domains he defines a semantics for PCF over an extension $\{P(\sigma)\}_{\sigma \text{ type}}$. He shows that every recursive object (see Section 2 for definitions) is PCF-definable relative to a parallel, continuous OR-operator and a parallel, continuous $\exists n$-operator. Plotkin [15] discuss totality in connection with denotational and operational semantics. Our theorems actually answer problems stated in [15].
The PCF-definable objects of pure types are exactly the P-computable objects, i.e. $S1 - S9$ over the partial, continuous functionals. This has been observed by Bellantoni [1] and Berger [2]. The proof follows a line of argument developed by Platek and used in Moldestad [10] establishing the conection between Platek's notion of computability and Kleene's notion.
Now, every mixed type is isomorphic to a retraction of a pure type, with primitive recursive projection- and inclusion maps. This is shown in detail in Moldestad [10] for total and partial objects using the same construction in the two cases. As a consequence, our results hold for mixed types as well.
In this setting our Main Theorem also reads:
To every recursive total $\Phi \in P(\sigma)$ there is an equivalent total $\hat{\Phi} \in P(\sigma)$ that is PCF-definable. Indeed, we may find $\hat{\Phi} \sqsubseteq \Phi$.

Acknowledgements I am grateful for useful discussions with U. Berger and enlightening comments from G. Plotkin and S. Bellantoni.

2 The Main Theorem

In this section we will give the basic definitions, state some standard facts from domain theory and state the main theorem. Any standard text, e.g [16], on domain theory can be used as background. We will consider an algebraic domain, or just a domain, to be the set of ideals in a set of partially ordered compacts closed under the least upper bounds of finite bounded sets, the ideals being ordered by inclusion, and we define our hierarchies in this setting.

Definition 1

a) Let $P(0)$ be the flat domain $\mathbb{N}_\perp = \{\perp\} \cup \mathbb{N}$.

b) Let $P(k+1)$ be the domain $P(k) \to \mathbb{N}_\perp$

where the details of the definition are given below.

The compacts in $P(0)$ will just be the domain elements, while the compacts in $P(k+1)$ will be given as sets $\{(\sigma_1, a_1), \ldots, (\sigma_s, a_s)\}$ where $\sigma_1, \ldots, \sigma_s$ are compacts in $P(k)$, $a_1, \ldots, a_s$ are numbers and $a_i = a_j$ whenever $\{\sigma_i, \sigma_j\}$ is bounded. We let $P_0(k)$ be the set of compacts in $P(k)$.
$P_0(k)$ will be ordered as follows:

1. We order $P_0(0)$ by $\perp \sqsubseteq \perp$, $\perp \sqsubseteq n$ and $n \sqsubseteq n$ for each $n \in \mathbb{N}$.

2. In $P_0(k+1)$ we let

$$\{(\sigma_1, a_1), \ldots, (\sigma_s, a_s)\} \sqsubseteq \{(\tau_1, b_1), \ldots, (\tau_t, b_t)\}$$

   if

$$\forall i \leq s \exists j \leq t(\tau_j \sqsubseteq \sigma_i \wedge b_j = a_i).$$

The domain elements are given as ideals of compacts, and we identify a compact with the ideal generated from it.

We organise $\{P(k)\}_{k \in \mathbb{N}}$ to a typed hierarchy by defining application as follows:

If $\alpha \in P(k+1)$ and $\beta \in P(k)$, we let $\alpha(\beta) = a$ if there is a compact τ in β such that the compact $\{(\tau, a)\}$ is in α. In this way $\{P(k)\}_{k \in \mathbb{N}}$ is viewed as a hierarchy of partial, continuous functionals.
We will call a pair (τ, a) a *basic compact*, and we will identify it with $\{(\tau, a)\}$.

Definition 2

a) Let $T(0) \subseteq P(0)$ be the set of natural numbers.

b) Let
$$T(k+1) = \{\alpha \in P(k+1) \mid \forall \beta \in T(k)(\alpha(\beta) \in \mathbb{N})\}.$$

$T(k)$ will be the set of hereditarily total objects in $P(k)$. Longo and Moggi [9] showed that the relation

$$\alpha \sim \beta \Leftrightarrow \alpha \sqcap \beta \in T(k)$$

is an equivalence relation on $T(k)$ and that each total object of type $k+1$ respects this relation.
By the Kleene-Kreisel density theorem this is also equivalent to $\{\alpha, \beta\}$ being bounded, or in other words, to α and β being consistent. This relation is further the same as hereditarily extentional equality. We will state the Kleene-Kreisel density theorem below.

Definition 3 Let $\{Ct(k)\}_{k \in \mathbb{N}}$ be the typed hierarchy isomorphic to the set of equivalence classes in $\{T(k)\}_{k \in \mathbb{N}}$ under the induced application operator.

It is easy to see that the compacts in $P_0(k)$ can be enumerated in such a way that all relevant relations and operations on compacts can be replaced by primitive recursive operations on the numbers. We will use this enumeration to define recursive sets of compacts and recursive enumerations of sets of compacts.

Definition 4

a) If $\phi \in Ct(k)$, a *representative* for ϕ will be an element in the corresponding equivalence-class in $T(k)$.

b) $\phi \in Ct(k)$ is *recursive* if it has one representative that is recursively enumerable. Using generally accepted terminology we will also call $\phi \in P(k)$ recursive when the set of compacts is recursively enumerable. This concept has also been called *Scott-computable*.

c) $\Psi \in P(k)$ is *P-computable* if it is $S1 - S9$-computable over the hierarchy $\{P(n)\}_{n \in \mathbb{N}}$.

d) $\phi \in Ct(k)$ is *P-computable* if it has one representative that is P-computable.

The enumerations of elements in $T(k)$ will to some extent correspond to Kleene associates. Our definition of *recursive* is equivalent to the classical one as a functional with a recursive associate. This is discussed in Normann [12]. It is also standard to view the recursively enumerable elements of an effective domain as the effective objects.

Theorem 1 Main Theorem
A functional in $Ct(k)$ is recursive if and only if it is P-computable.

Remark 1 This statement is weaker than the statement that every recursively enumerable element of $T(k)$ is $S1 - S9$-computable over the partial continuous functionals, a statement that is incorrect, as observed by Plotkin [14], for the equivalent PCF.

We will lead the reader to a proof of the main theorem in several steps. The theorem is trivially correct for $k = 0$ and $k = 1$ and easy for 2, so the first challenge comes at type 3. We will first consider a special case of type 3 functionals, where the key algorithm is transparent, before we adjust this algorithm to the general case of type 3.
We will then prove Theorem 2, a strengthening of the main theorem, by induction on the type. Finally we will prove Theorem 3, which will be the same as Theorem 2, but restricted to the effective operators. Unfortunately we seem to need a slightly more elaborate construction in order to obtain Theorem 3.

The induction hypothesis will require the theorem to be correct two types below. Theorem 2 is also easy (and well known) for types ≤ 2, and the general proof for type 3 is just a special case of the general proof for higher types. Thus the reason for splitting up between type 3 and type > 3 is to improve the readability of the argument, as is the reason for focusing on the special case in section 3.

We will use the Kleene-Kreisel density theorem, which can be stated as follows in our setting:

Proposition 1 *Uniformly recursive in any $\sigma \in P_0(k)$ there is an object $E(\sigma) \in T(k)$ with $\sigma \sqsubseteq E(\sigma)$.*

From any of the standard proofs of the density theorem, we can find an enumeration of the compacts in $E(\sigma)$ primitive recursive uniformly in σ. Kleene even shows that $E(\sigma)$ represents a primitive recursive functional in $Ct(k)$. It is however not correct that $E(\sigma)$ can be chosen to be primitive recursive in $P(k)$, we will give an example below.

3 A special case

Definition 5 Let $\{r_m\}_{m\in\mathbb{N}}$ be an enumeration of all functions $r : \mathbb{N} \to \mathbb{N}$ that is constant 0 exept on a finite set. This constitutes a dense set in the standard topology on $\mathbb{N}^{\mathbb{N}}$.
If σ is a finite sequence of natural numbers, and r is a function from the set of natural numbers to the natural numbers, we let σr denote the infinite sequence obtained by concatenation.

Definition 6 Let T be a recursive, non-wellfounded tree of sequences of natural numbers, with a recursive enumeration $T = \{\sigma_n \mid n \in \mathbb{N}\}$.
For $F \in T(2)$ let

$$\Phi_T(F) = \mu n \forall m \geq n \exists k \leq m (F \text{ is constant on } \{\sigma_k r_i \mid i \leq m\}).$$

Remark 2 At the face of it we have defined $\Phi_T(F)$ using the μ-operator over a Π^0_1-set and a priori Φ_T need not be everywhere defined or continuous. We will show that Φ_T is the restriction of a P-computable element in $T(3)$ to $T(2)$.
Functionals in $Ct(3)$ corresponding to Φ_T and similar functionals played an important rôle in showing the limitations of $S1 - S9$ computability in $\{Ct(k)\}_{k\in\mathbb{N}}$ itself, see Normann [13]. If T has no recursive infinite branch, Φ_T will not be $S1 - S9$-computable in the hierarchy $\{Ct(k)\}_{k\in\mathbb{N}}$. This is proved by an argument similar to the one used by Tait [17] to prove that the fan functional is not computable.

Lemma 1 is not really used in the proof of Lemma 2 or anywhere else in this paper, and is actually a consequence of Lemma 2. We state and prove it in order to give the reader a better intuition about the nature of the functional Φ_T:

Lemma 1 *Φ_T is the restriction of a recursive object $\tilde{\Phi}_T \in T(3)$ to $T(2)$.*

Proof
Let $F \in P(2)$. Let $\tilde{\Phi}_T(F) = n$ if for some $n_0 \geq n$ we have

1. $F(\sigma_{n_0}) \in \mathbb{N}$.
2. $\forall m \leq n_0 \forall i \leq n_0 F(\sigma_m r_i) \in \mathbb{N}$.
3. n is the least number $\leq n_0$ such that
   $\forall m(n \leq m \leq n_0 \rightarrow \exists k \leq m(F$ is constant on $\{\sigma_k r_i \mid i \leq m\}))$.

If we let F_0 vary over $P_0(2)$ we have that $\{(F_0, n) \mid \tilde{\Phi}(F_0) = n\}$ is r.e.

Now, let $F \in T(2)$. We can find a n_0 such that $F(\sigma_{n_0}) \in \mathbb{N}$ because the tree T is not well founded. Clearly 2. is satisfied by a compact $F_0 \sqsubseteq F$ for this n_0. Finally n_0 will be an upper bound for $\Phi_T(F)$ and then the n (that will exist) satisfying 3. will be the actual value of $\Phi_T(F)$.

Lemma 2 *Let Φ_T be as above.*
Φ_T is the restriction of a P-computable $\hat{\Phi}_T \in T(3)$ to $T(2)$.

Proof
Let $N \in \mathbb{N}$ and $F \in T(2)$. Let Φ_N be *the N'th approximation* to $\Phi_T(F)$:

$\Phi_N(F) =$
$\mu n \leq N \forall m(n \leq m \leq N \rightarrow \exists k \leq m(F$ is constant on $\{\sigma_k r_i \mid i \leq m\}))$.
where we let $\Phi_N(F) = N$ if we cannot find such $n \leq N$.
If $N \leq M$, then $\Phi_N(F) \leq \Phi_M(F) \leq \Phi_T(F)$, and $\Phi_T(F) = \lim_{N \rightarrow \infty} \Phi_N(F)$.
We consider the Φ_N defined above as the restriction of a uniformly P-computable $\Phi_N \in T(3)$ to $T(2)$, where we give the obvious algorithm for computing $\Phi_N(F)$ also for partial F. The one thing to notice is that we verify the quantifiers $\exists k \leq m$ by a sequential test, and not by a parallel test. In particular this means even for partial F that if $\tilde{\Phi}_T(F) \downarrow$ then the computation of $\Phi_N(F)$ terminates.

We introduce two auxilliary functions with F ranging over $P(2)$:

$$\phi(F, n) \text{ and } \psi(F, n)$$

which are defined simultaneously as follows:
If $F(\sigma_n r_{\psi(F,n)}) \neq F(\sigma_n r_0)$, let $\phi(F, n) = \phi(F, n + 1)$.

If $F(\sigma_n r_{\psi(F,n)}) = F(\sigma_n r_0)$, let $\phi(F,n) = \Phi_n(F)$.
We compute $\psi(F,n)$ as follows: Search for the least m such that $F(\sigma_n r_m) \neq F(\sigma_n r_0)$ or ($m > n$, $F(\sigma_n r_m) = F(\sigma_n r_0)$ and $F(\sigma_m r_0) = F(\sigma_m r_{\psi(F,m)})$).

By the recursion theorem, we have partial computable functionals ϕ and ψ satisfying these equations. We will show that whenever $F \in T(2)$ then $\phi(F,0) = \Phi_T(F)$ which means that we let $\hat{\Phi}(F) = \phi(F,0)$ for $F \in P(2)$.

Let $F \in T(2)$. Let n_0 and $a \in \mathbb{N}$ be such that (σ_{n_0}, a) is a basic compact in F. Then $\Phi_{n_0}(F) = \Phi_T(F)$. When we try to compute $\phi(F,n_0)$, we realise that $F(\sigma_{n_0} r_0) = F(\sigma_{n_0} r_{\psi(F,n_0)})$ without knowing anything about $\psi(F,n_0)$, so by our algorithm

$$\phi(F,n_0) = \Phi_{n_0}(F) = \Phi_T(F).$$

By reversed induction we will prove that for $n < n_0$ we have that $\phi(F,n) = \Phi_T(F)$ and that $\psi(F,n) \in \mathbb{N}$. Finally we will then reach our conclusion that $\phi(F,0) = \Phi_T(F)$ for $F \in T(2)$.
We will show that the computation of $\psi(F,n)$ terminates and that ϕ does what it is supposed to do.
By the induction hypothesis, and using that $F \in T(2)$, we see that for every step in the search up to n_0, the test will terminate. Since $F(\sigma_{n_0} r_0) = F(\sigma_{n_0} r_{\psi(F,n)})$, the test will terminate also at stage n_0 and the search will stop there if it ever gets that far. So, let $m \leq n_0$ be the point where the search stops. There are two cases:

Case 1
$F(\sigma_n r_m) \neq F(\sigma_n r_0)$.
Then $\psi(F,n) = m$ and by our algorithm and the induction hypothesis

$$\phi(F,n) = \phi(F,n+1) = \Phi_T(F).$$

Case 2
$F(\sigma_n r_m) = F(\sigma_n r_0)$.
Then $m > n$ and $F(\sigma_m r_0) = F(\sigma_m r_{\psi(F,m)})$ since we stopped the search here. Thus $\phi(F,m) = \Phi_m(F)$ by the algorithm and $\phi(F,m) = \Phi_T(F)$ by the induction hypothesis.
Moreover, $\phi(F,n) = \Phi_n(F)$. We will show that $\Phi_n(F) = \Phi_m(F)$, obtaining that $\phi(F,n) = \Phi_T(F)$.
We prove the nontrivial inequality $\Phi_m(F) \leq \Phi_n(F)$:
Choose m' with $\Phi_n(F) \leq m' \leq m$.

We show that there is a $k \leq m'$ such that F is constant on $\{\sigma_k r_i \mid i \leq m'\}$:
If $n \leq m'$ we may choose $k = n$ since we are in case 2. If $m' < n$ we have assumed that $\Phi_n(F) \leq m'$ so in particular $\Phi_n(F) < n$. Then by definition of $\Phi_n(F)$ there is a $k \leq m'$ such that F is constant on $\{\sigma_k r_i \mid i \leq m'\}$.
The proof is complete.

Remark 3 We have designed the algorithm for $\phi(F, n)$ (also for partial F) such that we get imediate termination exactly when $F(\sigma_n)) \in \mathbb{N}$, and then, provided that $\Phi_T(F)$ is defined, with the correct value $\phi(F, n) = \Phi_T(F)$. In the other case, we have to "climb" up to an $n_0 > n$ for which we have imediate termination.

Since we will refer to Lemma 2 in the proofs of Theorems 1 and 2, let us see why we consider this as a special case, and how it relates to the general case.

Corollary 1 *Let T be a recursive, non-wellfounded tree, and let $\Phi \in T(3)$ be defined by*

$$\Phi(F) = 0 \Leftrightarrow \exists \sigma \in T(F(\sigma) \in \mathbb{N})$$

for $F \in P(2)$.
Then there is a P-computable (PCF-definable) $\hat{\Phi} \in T(3)$ with $\hat{\Phi} \sqsubseteq \Phi$.

Proof
Let ϕ be as in the proof of Lemma 2 and let $\hat{\Phi}(F) = 0 \cdot \phi(F, 0)$.
$\hat{\Phi}$ will be P-computable,
Termination of $\hat{\Phi}(F)$ requires termination of $\phi(F, 0)$, which again will require imediate termination of some $\phi(F, n_0)$, i.e. that $F(\sigma_{n_0}) \in \mathbb{N}$.

4 Type 3

We will now prove the Main Theorem for type 3 in general. The key idea in the proof is as in the special case above, but the generality forces the proof to be technically more complicated, and the key algorithm may be hidden in these technicalities. In a sense we can say that we replace the tree T by the set of basic compacts for a $\Phi \in T(3)$.

In this section, we will let $\Phi \in T(3)$ be recursive, and we let

$$\{(\sigma_n, a_n)\}_{n \in \mathbb{N}}$$

be a recursive enumeration of the basic compacts in Φ. Then for every $F \in P(2)$ we have that

$$\Phi(F) = a \Leftrightarrow \exists n(\sigma_n \sqsubseteq F \wedge a = a_n)$$

We will construct a functional $\hat{\Phi}$ P-computable in $\{(\sigma_n, a_n)\}_{n \in \mathbb{N}}$ such that for all $F \in T(2)$ we have

$$\Phi(F) = \hat{\Phi}(F).$$

We may assume that each σ_n will be of the form

$$\sigma_n = \{(\tau_{n,1}, b_{n,1}), \ldots, (\tau_{n,s_n}, b_{n,s_n})\}$$

where each $\tau_{n,i}$ is a finite sequence of natural numbers, though this is not really essential for the argument to work. Let $\{r_i\}_{i \in \mathbb{N}}$ be the sequence of almost zero functions as in the previous section.

Now, if $a_i \neq a_j$ there will be $i' \leq s_i$ and $j' \leq s_j$ such that $\tau_{i,i'}$ and $\tau_{j,j'}$ are consistent while $b_{i,i'} \neq b_{j,j'}$. We say that *m is critical for (i,j)* if $\tau_{i,i'} \subseteq \tau_{j,j'} r_m$ or $\tau_{j,j'} \subseteq \tau_{i,i'} r_m$ for at least one choice of i' and j' as above. For each m, we let $\rho(m)$ be the least number $\geq m$ such that for each $i \leq m$ and $j \leq m$ with $a_i \neq a_j$ there is a critical $m' \leq \rho(m)$ for (i,j).
We say that $\sigma_n \sqsubseteq_m F$ if we for all $i \leq s_n$ and $j \leq \rho(m)$ have

$$F(\tau_{n,i} r_j) = b_{n,i}.$$

Finally, for $F \in T(2)$, let

$$\Psi(F) = \mu n \exists a \forall m \geq n \exists k \leq m(\sigma_k \sqsubseteq_m F \wedge a_k = a).$$

We will show that there is a P-computable functional $\hat{\Psi} \in T(3)$ that agrees with Ψ on $T(2)$. First we will show how to compute a total $\hat{\Phi}$ equivalent to Φ from such a $\hat{\Psi}$.

Lemma 3 *If Ψ agrees with $\hat{\Psi} \in T(3)$ on $T(2)$, there is a $\hat{\Phi} \in T(3)$ that is $S1-S9$-computable from $\hat{\Psi}$ and that is equivalent to Φ.*

Proof
For any m, $\{a_n \mid n \leq m \wedge \sigma_n \sqsubseteq_m F\}$ contains at most one object (this is the point of introducing $\rho(m)$).

We then let $\hat{\Phi}(F)$ be the unique a such that if $n \leq \hat{\Psi}(F)$ and $\sigma_n \sqsubseteq_{\hat{\Psi}(F)} F$ then $a_n = a$.

We will now use the argument from the special case to define $\hat{\Psi}$ as a P-computable function, and prove that it agrees with Ψ on $T(2)$.
For $N \in \mathbb{N}$ let

$$\Psi_N(F) = \mu n \leq N \exists a \forall m (n \leq m \leq N \to \exists k \leq m(\sigma_k \sqsubseteq_m F \wedge a_k = a))$$

where Ψ_N is a P-computable object in $T(3)$ in analogy with the Φ_N of the special case. We observe the same monotonicity properties as for the Φ_N's.

Now we define the two auxilliary functions ϕ and ψ by the recursion theorem in analogy with the construction in the special case:

1. If for all $i \leq s_n$ we have that $F(\tau_{n,i} r_{\psi(F,n)}) = b_{n,i}$ we let $\phi(F,n) = \Psi_n(F)$, otherwise we let $\phi(F,n) = \phi(F,n+1)$.

2. In computing $\psi(F,n)$ we search for the least m such that either

   $$F(\tau_{n,i} r_j) \neq b_{n,i} \text{ for some } i \leq s_n \text{ and some } j \leq \rho(m)$$

   (where termination requires that $F(\tau_{n,i} r_j)$ terminates for all $i \leq s_n$ and $j \leq \rho(m)$, this to avoid the need for non-deterministic parallellism) or

   $$m > n,\ F(\tau_{n,i} r_j) = b_{n,i} \text{ for all } i \leq s_n \text{ and } j \leq \rho(m) \text{ and } F(\tau_{m,i} r_{\psi(F,m)}) = b_{m,i} \text{ for all } i \leq s_m.$$

   In the first case we select $\psi(F,n)$ to be one $j \leq \rho(m)$ with $F(\tau_{n,i} r_j) \neq b_{n,i}$ for some $i \leq k_n$, while in the second case we let $\psi(F,n) = 0$.

We can now end the proof as in the special case. Let $F \in T(2)$. We choose some n_0 such that $\sigma_{n_0} \sqsubseteq F$. Then $\phi(F, n_0) = \Psi(F)$. Moreover, by reversed induction we show that for $n < n_0$ we have that $\psi(F,n)$ terminates and that $\phi(F,n) = \Psi(F)$. We do not repeat the details of this argument, just notice that some care has to be shown in case 2, the argument that shows that $\Psi_n(F) = \Psi_m(F)$ for $n < m \leq n_0$. There we observe that a_n will be the unique a used both in the definition of $\Psi_n(F)$ and $\Psi_m(F)$.

As an extra bonus we will obtain that if $\Phi \in P(3)$ is recursively enumerable and we construct $\hat{\Phi}$ from the enumeration as above, we get $\hat{\Phi} \sqsubseteq \Phi$. This is a consequence of

Lemma 4 *Let $\{(c_n, a_n)\}_{n\in\mathbb{N}}$ be a recursive enumeration of the basic compacts of an object $\Phi \in P(3)$ (not neccessarily total).*
For any $F \in P(2)$, let $\phi(F,n)$ and $\psi(F,n)$ be defined as in the construction. Then for any n.

$$\text{If } \forall m \geq n(\sigma_m \not\sqsubseteq F) \text{ then } \phi(F,n) = \bot.$$

Proof
In computing $\phi(F,n)$ we must rely on finding $\psi(F,n)$. There are four possibilities:

1. The search for $\psi(F,n)$ does not terminate.
   Then $\phi(F,n) = \bot$.
2. We find $\psi(F,n)$ but $F(\tau_{n,i} r_{\psi(F,n)}) = \bot$ for some i.
   Then $\phi(F,n) = \bot$.
3. We find $\psi(F,n)$ with some $F(\tau_{n,i} r_{\psi(F,n)}) \neq b_{n,i}$.
   Then we must compute $\phi(F,n+1)$ in order to compute $\phi(F,n)$.
4. Otherwise, i.e. we find $\psi(F,n)$ with $F(\tau_{n,i} r_{\psi(F,n)}) = b_{n,i}$ for all $i \leq s_n$.
   This, however, requires that we find some $n_1 > n$ such that $F(\tau_{n_1,i} r_{\psi(F,n_1)}) = b_{n_1,i}$ for all $i \leq s_{n_1}$, and in order to obtain this we need $n_2 > n_1$ with the same property etc.

Alltogether, we see that there is no way that our algorithm for $\phi(F,n)$ can terminate. Thus the lemma is proved.

Corollary 2 *Let $\Phi \in P(3)$ be recursively enumerable and let $\hat{\Phi}$ be as constructed. Let $F \in P(2)$.*
If $\hat{\Phi}(F) = a$, then $\Phi(F) = a$.

Proof
If $\hat{\Phi}(F) = a$, then $\phi(F,0)$ terminates. Then by Lemma 4 there is an n such that $\sigma_n \sqsubseteq F$ and $a = a_n$.
This means that $\Phi(F) = a$.

5 The general case for arbitrary types

The proof in type 3 does not imediatly adjust to the situation of higher types. Let us return to the special case in order to discuss the problem. In the algorithm we want to compute $F(\sigma_n r_{\psi(F,n)})$ and for the case $n = n_0$ we argue that this terminates if (σ_n, a) is a compact in F for some a. This is the case because we actually have an algorithm for $\sigma_n r_{\psi(F,n)}$ whose interpretation extends σ_n even if $\psi(F, n)$ does not terminate.

If we want to carry out a similar construction for types larger than three, we will somehow need to enumerate a dense set of total extensions of a compact δ of type $k - 2$ (even then we will need some more elaborate notation, but that is not a main obstacle). The problem is that the algorithm for computing the primitive recursive extensions of a compact δ will induce elements of $T(k - 2)$ not containing δ, just being consistent with δ. We will illustrate this at the lowest possible type.

Example 1 Let π_1, π_2 and π_3 be three pairwise inconsistent compacts of type 1, $\pi_1 = \{(1, 1), (2, 2)\}$, $\pi_2 = \{(2, 1), (3, 2)\}$ and $\pi_3 = \{(3, 1), (1, 2)\}$.
Let $\delta = \{(\pi_1, 1), (\pi_2, 2), (\pi_3, 3)\}$.
Let F be a total, recursive functional of type 2 consistent with δ. For any f, the computation of $F(f)$ can be seen as a sequential process, where we at a first stage independently of f ask "What is $f(x)$?" This in particular means that if (π, c) is a basic compact in F, then $\pi(x)$ must be in $\mathbb{N}$. Thus F cannot extend δ, because the intersection of the domains of π_1, π_2 and π_3 is empty.

The construction in section 4 can be seen as the construction of a uniform algorithm for computing $\hat{\Phi}$ from an enumeration of the basic compacts in Φ. We will use the fact that there is such a uniform algorithm at type $k - 2$ in order to prove the Main Theorem for type k.
We have produced such an algorithm for type 3, and it is trivial to produce such an algorithm for type 2:
Let $\{(\sigma_n, a_n)\}_{n\in\mathbb{N}}$ be an enumeration of the basic compacts in $F \in T(2)$.
We compute $\hat{F}(f)$ by the following algorithm: Search for the least n such that σ_n is consistent with f. Then $\hat{F}(f) = a_n$. We of course mean the sequential interpretation of this algorithm.

The Main Theorem will be a consequence of the more general

Theorem 2 *Let $k \geq 0$ and let $\Phi \in P(k)$.*
Let $\{\delta_n\}_{n\in\mathbb{N}}$ be an enumeration of the basic compacts in Φ.

Then uniformly P-computable in $\{\delta_n\}_{n\in\mathbb{N}}$ there is a $\hat{\Phi} \sqsubseteq \Phi$ such that if $\Phi \in T(k)$ then $\hat{\Phi} \in T(k)$

Proof
We will assume that $k > 3$ and that the theorem holds for $k-2$.
Let $\Phi \in P(k)$ be given, and let $\{(\sigma_n, a_n)\}_{n\in\mathbb{N}}$ be an enumeration of the basic compacts in Φ.
Let

$$\sigma_n = \{(\tau_{n,1}, b_{n,1}), \ldots, (\tau_{n,s_n}, b_{n,s_n})\}$$

for $n \in \mathbb{N}$.
Let $F \in P(k-1)$. We will give an algorithm for computing $\hat{\Phi}(F)$.

Let $\{(\pi_i, c_i)\}_{i\in\mathbb{N}}$ be a fixed, effective enumeration of all basic compacts of type $k-2$.
Then each compact $\tau \in P_0(k-2)$ has a unique finite enumeration

$$\tau = \{(\pi_{i_0}, c_{i_o}), \ldots, (\pi_{i_l}, c_{i_l})\}$$

where $i_0, \ldots, i_l$ is an increasing sequence.
Likewise, the enumeration $\{(\pi_i, c_i)\}_{i\in\mathbb{N}}$ induces an enumeration of each $g \in P(k-2)$. This enumeration will not neccessarily be effective even when g is recursive.
If $g \in T(k-2)$, let $\hat{g} \sqsubseteq g$ be the functional constructed from this enumeration. Here we use the induction hypothesis. Similarily, if $\tau \in P_0(k-2)$ we let $\hat{\tau}$ be the partial functional in $P(k-2)$ obtained from the enumeration of τ described above. $\hat{\tau}$ is P-computable uniformly in (an index for) τ, and by the induction hypothesis $\hat{\tau} \sqsubseteq \tau$.

Let

$$\{\xi_{n,i,j}\}_{n\in\mathbb{N}, i\leq s_n, j\in\mathbb{N}}$$

be a primitive recursive indexed family of primitive recursive enumerations of basic compacts of type $k-2$ such that

1. For all n, i and j, $\xi_{n,i,j}$ extends the enumeration of $\tau_{n,i}$.

2. For all n, i and l, $\xi_{n,i,j}$ enumerates the basic compacts in an element of $T(k-2)$.

3. For all n and i, $\{\xi_{n,i,j}\}_{j\in\mathbb{N}}$ is dense in the set of all enumerations of elements in $T(k-2)$ extending the given enumeration of $\tau_{n,i}$.

The existence of this enumeration is a consequence of the Kleene-Kreisel density theorem.

Claim 1
If $\sigma = \{(\tau_1, b_1), \ldots, (\tau_s, b_s)\}$, $F \in P(k-1)$ and $F(\hat{\tau}_j) = b_j$ for all $j \leq s$, then σ is consistent with F.

Proof
Assume not.
Then there is a j and a $g \in P(k-2)$ with $\tau_j \sqsubseteq g$ and with $b_j \neq F(g) \in \mathbb{N}$.
Choose an enumeration of g extending the fixed enumeration of τ_j. By the induction hypothesis, let $f \sqsubseteq g$ be constructed from this enumeration. Then f extends $\hat{\tau}_j$ so $F(f) = b_j$, contradicting that $f \sqsubseteq g$.

Claim 2
If $F \in T(k-1)$ and $\Phi \in T(k)$, there is an $n \in \mathbb{N}$ such that $F(\hat{\tau}_{n,i}) = b_{n,i}$ for all $i \leq s_n$.

Proof
We define $\tilde{F} \in P(k-1)$ by $(\tau, b) \in \tilde{F}$ if $F(\hat{\tau}') = b$ for some $\tau' \sqsubseteq \tau$.
We first show that $\tilde{F}$ is consistent:
Let (τ_1, b_1) and (τ_2, b_2) be in $\tilde{F}$ where τ_1 and τ_2 are consistent.
Choose $\tau_1' \sqsubseteq \tau_1$ and $\tau_2' \sqsubseteq \tau_2$ such that $F(\hat{\tau}_1') = b_1$ and $F(\hat{\tau}_2') = b_2$.
Let g be total extending $\tau_1 \sqcup \tau_2$ and choose two enumerations of g, one extending the fixed enumeraton of τ_1', the other extending the enumeration of τ_2'. Let f_1 and f_2 be the total objects equivalent to g obtained from the two enumerations of g. f_1 and f_2 will be equivalent, so

$$b_1 = F(f_1) = F(f_2) = b_2.$$

Now, let $g \in T(k-2)$. Then by the induction hypothesis, $\hat{g} \in T(k-2)$ so $F(\hat{g}) = b \in \mathbb{N}$.
There will be a compact $\tau \sqsubseteq g$ such that the enumeration of τ is an initial segment of the enumeration of g and such that $F(\hat{\tau}) = b$. We then have that $(\tau, b) \in \tilde{F}$.
This argument shows that $\tilde{F}$ is total and equivalent to F.
Then there is a $\sigma \sqsubseteq \tilde{F}$ and an $a \in \mathbb{N}$ such that $\Phi(\sigma) = a$, i.e. (σ, a) is a basic compact in Φ.
Let

$$\sigma = \{(\tau_1, b_1), \ldots, (\tau_s, b_s)\}.$$

By construction of $\tilde{F}$ there are $\tau'_j \sqsubseteq \tau_j$ for all $j \leq s$ such that $F(\hat{\tau}'_j) = b_j$. Then $\sigma \sqsubseteq \sigma' = \{(\tau'_1, b_1), \ldots, (\tau'_s, b_s)\}$ and in turn $(\sigma', a) \sqsubseteq (\sigma, a)$. Consequently (σ', a) is also a basic compact in Φ, and is of the form (σ_n, a_n) for some $n \in \mathbb{N}$. This ends the proof of the claim.

We are now ready to do the induction step. With some change of notation we may use the same proof as in the case of type 3.
We replace $\tau_{n,i} r_j$ by $\hat{\xi}_{n,i,j}$, where we mean the functional in $T(k-2)$ computed from the enumeration $\xi_{n,i,j}$.
We replace the definition of m being critical by:
m is critical for (i, j) if $\tau_{i,i'}$ is consistent with $\xi_{j,j',m}$ or vice versa for a relevant choice of i' and j'. (This will be decidable, and we can always find one m critical for (i, j) if $a_i \neq a_j$.)
Then the final change is to use a σ_{n_0} satisfying Claim 2 in showing that the algorithm terminates for all total F when Φ is total.

We end the proof by showing that

$$\hat{\Phi}(F) = a \Rightarrow \Phi(F) = a$$

for all $F \in P(k-1)$.
In the algorithm for $\hat{\Phi}$ we need the value $\phi(F, 0)$, so it is sufficient to show that

$$\phi(F, 0) \neq \bot \Rightarrow \Phi(F) \neq \bot.$$

For each $\sigma = \{(\tau_1, b_1), \ldots, (\tau_s, b_s)\}$ let

$$\check{\sigma} = \{(\hat{\tau}_1, b_1), \ldots, (\hat{\tau}_s, b_s)\}.$$

By the argument of section 4 we get

$$\phi(F, 0) \neq \bot \Rightarrow \exists n(\check{\sigma}_n \sqsubseteq F).$$

But $\sigma_n \sqsubseteq \check{\sigma}_n$ since $\hat{\tau}_{n,j} \sqsubseteq \tau_{n,j}$ so $\sigma_n \sqsubseteq F$. It follows that $\Phi(F) \neq \bot$, and the proof is complete.

Remark 4 When τ is a finite sequence of natural numbers, there is no problem in making an algorithm for τr_i uniformly in i such that we extend τ even for $i = \bot$.
The problem was that we could not do the same for the standard dense

set of extensions at higher types. However, the algorithm for $\hat{\xi}_{i,j,l}$ will give us $\hat{\tau}_{i,j}$ for $l = \perp$, simply because we are using enumerations of $\xi_{i,j,l}$ that extend the enumeration of $\tau_{i,j}$, and the partial enumeration of $\xi_{i,j,\perp}$ is indeed the enumeration of $\tau_{i,j}$. Thus we use the induction hypothesis, and the underlying construction on suitable associates of the extensions of a compact to get a P-computable version of the extension maps.

Remark 5 As an alternative to using complete domains as a basis for the semantics of programs one may use effective domains. This is standard and means that one restricts the attention to domain objects corresponding to r.e. ideals. This again leads to an alternative notion of totality, the hereditarily effective operators. It seems that we have to modify the construction of $\hat{\Phi}$ to a construction of a $\ddot{\Phi}$ in order to obtain Theorem 2 for the effective case. On the other hand, the construction $\ddot{\Phi}$ will also prove Theorem 2 in the complete case.

Definition 7 Let $R(0) = \mathbb{N}$.
Let $R(k+1) = \{\Phi \in P(k+1) \mid \Phi \text{ is recursive and } \Phi \in \mathbb{N} \text{ for all } F \in R(k)\}$.

Theorem 3 *Uniformly P-computable in a recursive enumeration of the basic compacts in $\Phi \in P(k)$, there is a $\ddot{\Phi} \sqsubseteq \Phi$ such that if $\Phi \in R(k)$ then $\ddot{\Phi} \in R(k)$.*

Proof
Most of the proof of Theorem 2 is directly valid for $R(k)$ as well as for $T(k)$. We will explain where the difficulty is and how the construction can be adjusted in order to avoid it. In the proof of Claim 2 in the general case, we need to show that $\tilde{F} \in R(k-1)$ when $F \in R(k-1)$.
By the construction, $\tilde{F}$ will be recursive when F is recursive. However, when we prove that $\tilde{F}$ is total we use the ineffective enumeration of a $g \in T(k-2)$. We do this because we use the given enumeration of all basic neighbourhoods in order to enumerate each $\tau \in P_0(k-2)$ and in turn to define $\hat{\tau}$.

The way we get around this obstacle is to consider all possible *versions* of basic compacts (σ_n, a_n), a version being one way to write down the compacts such that each $\tau_{n,i}$ is enumerated, but not neccessarily in the prefixed way. We then replace the enumeration $\{(\sigma_n, a_n)\}_{n\in\mathbb{N}}$ by an enumeration of all versions of each (σ_n, a_n).
We construct $\ddot{\Phi}$ based on this new enumeration $\{\sigma'_n, a'_n\}_{n\in\mathbb{N}}$ using the $\ddot{\tau}_{n,i}$ as

we previously used $\hat{\tau}_{n,i}$. The point is of course to use the enumeration of the $\tau_{n,i}$'s corresponding to the version.
We then define $\tilde{F}(\tau) = b$ if there is one $\tau_1 \sqsubseteq \tau$ with one enumeration giving rise to a $\ddot{\tau}$ with $F(\ddot{\tau}) = b$. This $\tilde{F}$ will still be recursive when F is, and we may use any recursive enumeration of a recursive g when we prove that $\tilde{F} \in R(k-1)$.
The rest of the argument will then be as in the proof of Theorem 2.

References

[1] Bellantoni, S. *Comments On Two Notions Of Higher Type Computability*, Unpublished notes, 1990.

[2] Berger, U. *Totale Objekte und Mengen in der Bereichtheorie* (in German), Thesis, München 1990.

[3] Berger, U. *Total sets and objects in domain theory*, Annals of Pure and Applied Logic 60 (1993) 91 - 117.

[4] Ershov, Yu. L. *Computable functionals of finite type*, Algebra and Logic 11 (1972) 203 - 277.

[5] Hyland. J.M.E. *Filterspaces and continuous functionals*, Annals of Mathematical Logic 16 (1979) 101 - 143.

[6] Kleene, S.C. *Recursive functionals and quantifiers of finite types I*, T.A.M.S. 91 (1959) 1 - 52.

[7] Kleene,S.C. *Countable functionals*, in A. Heyting (ed.) Constructivity in Mathematics, North-Holland (1959) 81 - 100.

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

[9] Longo, G. and Moggi, E. *The hereditarily partial effective functionals and recursion theory in higher types*, Jour. of Symbolic Logic 49 (1984), 1319 - 1332.

[10] Moldestad, J. *Computation in Higher Types* Springer Lecture Notes in Mathematics 574, (1977).

[11] Normann, D. *Recursion on the continuous functionals*, Springer Lecture Notes in Mathematics 811 (1980).

[12] Normann, D. *The continuous functionals*, to appear in E.R. Griffor (ed.) Handbook of computation theory, Elsevier (1998?).

[13] Normann, D. *The continuous functionals; computations, recursions and degrees*, Annals of Mathematical Logic 21 (1981) 1 - 26.

[14] Plotkin, G. *LCF considered as a programming language*, Theoretical Computer Science 5 (1977) 223 - 255.

[15] Plotkin, G. *Full Abstraction, Totality and PCF*, Draft.

[16] Stoltenberg-Hansen, V., Lindström, I and Griffor, E.R. *Mathematical Theory of Domains*, Cambridge Tracts in Theor. Comp. Science 22, Cambridge University Press (1994)

[17] Tait, W.W. *Continuity properties of partial recursive functionals of finite type*, unpublished notes.