

LOGICAL RELATIONS AND FEASIBILITY IN HIGHER TYPES

ULRICH BERGER

Abstract. We apply logical relations to define two hierarchies of functionals in all finite types as new candidates for higher type feasibility. The first hierarchy uses logical relations to generalize Cobham's Limited Recursion on Notation to all finite types. The hierarchy coincides with Cook and Kapron's Basic feasible Functionals (BFF) up to type level two, but at higher types our hierarchy might be strictly larger than BFF. The second hierarchy is defined by a generalized Kripke logical relation. The new point here is that the starting relations are not defined at base types, i.e. type level 0, but at types of any chosen level k . This requires a new construction of the relation for function types that in general differs from the usual one. The second hierarchy coincides with BFF up to type level k and is probably strictly larger at type level $k + 1$. We prove that if we work in the model of partial continuous functionals, all elements of our hierarchies are computable (which is not obvious for the second hierarchy). Both our hierarchies are closed under λ -definability. We neither know whether our hierarchies can be effectively generated nor how they are related in general.

§1. Introduction. Given a system of functionals of type level 1 (or of type levels \leq some fixed k) which is λ -closed, that is, closed under definability by simply typed λ -terms, it is natural to ask how this can be extended to a λ -closed system of functionals in all finite types. Longo and Moggi [11] investigated this question for the case of the system of partial functions on the natural numbers. They presented a method of extending this system to all finite types using a construction reminiscent of the definition of Banach-Mazur functionals and showed that the resulting hierarchy coincides with the partial continuous functionals. A related construction was investigated by Sazonov and Voronkov [13]. In both papers it is stated that this construction, when based on polynomial time computable functions of type 1, gives rise to a natural notion higher type

University of Wales Swansea.

feasible functional, but this claim has never been substantiated, and so far no one seems to have pursued this idea any further.

In this paper we have a fresh look at this approach from a more general perspective. We will show that indeed one obtains a very simple, but useful notion of higher type computational complexity, in fact we show that it is as general as possible.

§2. Type structures. Given sets A, B we let $A \rightarrow B$ denote the set of all functions from A to B . As usual, we write $f: A \rightarrow B$ for $f \in A \rightarrow B$ and $A \rightarrow B \rightarrow C$ for $A \rightarrow (B \rightarrow C)$. Application of a function $f: A \rightarrow B$ to an argument $a \in A$ will be written fa . Application associates to the left, i.e. fab means $(fa)b$. We define the basic combinators as usual by

$$\begin{aligned} \mathsf{K}_{AB}: A \rightarrow B \rightarrow A, & \quad \mathsf{K}_{AB}ab = a \\ \mathsf{S}_{ABC}: (A \rightarrow B \rightarrow C) \rightarrow (A \rightarrow B) \rightarrow A \rightarrow C, & \quad \mathsf{S}_{ABC}fga = fa(ga) \end{aligned}$$

We fix a set (of symbols) T_0 of *base types* and define the set T of *simple types* as the closure of the base types under the formal function type construction, $\rho \rightarrow \sigma$. As usual, $\rho \rightarrow \sigma \rightarrow \eta$ means $\rho \rightarrow (\sigma \rightarrow \eta)$, and $\vec{\rho} \rightarrow \sigma$ means $\rho_1 \rightarrow \dots \rightarrow \rho_k \rightarrow \sigma$. A *type structure* is a family of sets $D = \{D(\rho) \mid \rho \in T\}$ such that $D(\rho \rightarrow \sigma) \subseteq D(\rho) \rightarrow D(\sigma)$ for all $\rho, \sigma \in T$, and D is *combinatorially complete*, i.e. $\mathsf{K}_{D(\rho)D(\sigma)} \in D(\rho \rightarrow \sigma \rightarrow \rho)$ and $\mathsf{S}_{D(\rho)D(\sigma)D(\eta)} \in D((\rho \rightarrow \sigma \rightarrow \eta) \rightarrow (\rho \rightarrow \sigma) \rightarrow \rho \rightarrow \eta)$ for all $\rho, \sigma, \eta \in T$.

Typed λ -terms, are constructed from typed variables, x^ρ, y^ρ, \dots , by abstraction, $(\lambda x^\rho M^\sigma)^{\rho \rightarrow \sigma}$, and application, $(M^{\rho \rightarrow \sigma} N^\rho)^\sigma$. We will often omit type information as long as this does not cause ambiguities, and when we speak of *terms* we mean typed λ -terms. The set of all terms (of type ρ) is denoted $\Lambda(\rho)$. $\mathsf{FV}(M)$ denotes the set of free variables of a term M . We adopt the usual notational conventions for terms, e.g. we write $\lambda \vec{x}. M \vec{N}$ for $\lambda x_1 \dots \lambda x_n (\dots (M N_1) \dots N_k)$. As is well-known, terms can be interpreted in any type structure D . More precisely, there is a map $\llbracket \cdot \rrbracket [\cdot \mapsto \cdot]$ such that for each term M^ρ with $\mathsf{FV}(M) \subseteq \vec{x}^{\vec{\sigma}}$ and $\vec{a} \in D(\vec{\sigma})$ (the latter stands for $a_i \in D(\sigma_i)$ for $i = 1, \dots, k$) we have $\llbracket M \rrbracket [\vec{x} \mapsto \vec{a}] \in D(\rho)$ and the following equations hold

$$\begin{aligned} \llbracket x_i \rrbracket [\vec{x} \mapsto \vec{a}] &= a_i \\ \llbracket \lambda y M \rrbracket [\vec{x} \mapsto \vec{a}] b &= \llbracket M \rrbracket [y, \vec{x} \mapsto b, \vec{a}] \\ \llbracket MN \rrbracket [\vec{x} \mapsto \vec{a}] &= \llbracket M \rrbracket [\vec{x} \mapsto \vec{a}] \llbracket N \rrbracket [\vec{x} \mapsto \vec{a}] \end{aligned}$$

An element b of a type structure D is λ -definable over a set $A \subseteq D$ (more precisely $A \subseteq \bigcup\{D(\rho) \mid \rho \in T\}$) if there is a term M and $\vec{a} \in A$ such that $b = \llbracket M \rrbracket[\vec{x} \mapsto \vec{a}]$. A term is in *long normal form* if it is of the form $\lambda\vec{x}.y\vec{M}$ where the term $y\vec{M}$ is of base type and each of the M_i is again in long normal form. All we need to know about long normal forms is the well-known fact that to every term M there exists a term N of the same type in long normal form such that $\text{FV}(N) \subseteq \text{FV}(M)$ and for all \vec{a} of appropriate types $\llbracket M \rrbracket[\vec{x} \mapsto \vec{a}] = \llbracket N \rrbracket[\vec{x} \mapsto \vec{a}]$.

§3. Substructures and their largest extensions. Throughout this section we fix a type structure D .

The *level* of a type is defined by $\text{level}(\tau) = 0$, for $\tau \in T_0$, and $\text{level}(\rho \rightarrow \sigma) = \max(\text{level}(\rho) + 1, \text{level}(\sigma))$. We set $T(\leq n) := \{\rho \in T \mid \text{level}(\rho) \leq n\}$ and $D(\leq k) := \bigcup\{D(\rho) \mid \rho \in T(\leq k)\}$. We will often write $\rho \leq k$ for $\rho \in T(\leq k)$. The notations $T(< k)$, $D(< k)$, $\rho < k$, $\Lambda(\leq k)$, e.t.c. have similar meanings.

A k -substructure of D is a family $P = \{P(\rho) \mid \rho \leq k\}$ such that $P(\rho) \subseteq D(\rho)$ for each $\rho \leq k$. Throughout the paper we will only consider k -substructures for $k \geq 1$. A k -substructure P is λ -closed if it contains all elements in $D(\leq k)$ which are λ -definable over P (in particular P is closed under application). A $k+1$ -substructure Q is an *extension* of a k -substructure P if $Q(\rho) = P(\rho)$ for all $\rho \leq k$.

For the rest of this section we fix a number $k \geq 1$ and a λ -closed k -substructure P of D .

We are interested in the question of how to extend P to a λ -closed $k+1$ -substructure of D . It is easy to see that there is a smallest such extension: Just take all elements of $D(\leq k+1)$ that are λ -definable over P . More interesting is the fact that there is also a largest extension. We will prove this now.

An element $a \in D$ is *conservative over P* if all elements of $D(\leq k)$ which are λ -definable over $P \cup \{a\}$ are in P . We set

$$P^+ := \{a \in D(\leq k+1) \mid a \text{ is conservative over } P\}$$

PROPOSITION 3.1. P^+ is a λ -closed $k+1$ -substructure which extends P . P^+ is largest in the sense that for any other λ -closed $k+1$ -substructure Q extending P we have $Q \subseteq P^+$.

It is clear that P^+ is an extension of P and that any λ -closed $k+1$ -substructure extending P must be contained in P^+ . It remains to be proven that P^+ is λ -closed.

Let $\rho \leq k+1$, say $\rho = \vec{\rho} \rightarrow \tau$ where $\rho = \rho_1, \dots, \rho_n$ and τ is a base type. Hence $\vec{\rho} \leq n$ i.e. $\rho_i \leq k$ for all i . We call $a \in D(\rho)$ *Banach-Mazur over P* if for all tuples of types $\vec{\sigma} < n$ and \vec{g} where $g_i \in P(\vec{\sigma} \rightarrow \rho_i)$ we have $a \circ (\vec{g}) \in P(\vec{\sigma} \rightarrow \tau)$ (note that the expressions involved make sense since $\vec{\sigma} \rightarrow \rho_i \leq k$ and $\vec{\sigma} \rightarrow \tau \leq k$). The meaning of $a \circ (\vec{g})$ is the obvious one, namely $(a \circ (\vec{g}))\vec{b} = a(g_1\vec{b}) \dots (g_n\vec{b})$.

LEMMA 3.2. *Let P be a λ -closed k -substructure of D and let $a \in D(\leq k+1)$. Then a is conservative over P iff a is Banach-Mazur over P .*

PROOF. If a is conservative over P , then a is also Banach-Mazur over P , since for any $\vec{g} \in P$ of appropriate types, $a \circ (\vec{g})$ is λ -definable over $P \cup \{a\}$ and hence in P .

For the converse, assume a is Banach-Mazur over P . We show, by induction on the lengths of terms $M \in \Lambda(\leq k)$ in long normal form with free variables among u, \vec{x} of appropriate types that for any $\vec{b} \in P$ of appropriate types we have $c := \llbracket M \rrbracket[u, \vec{x} \mapsto a, \vec{b}] \in P$. Let $M = \lambda \vec{y}. z \vec{M}$ and set $d_i := \llbracket \lambda \vec{y}. M_i \rrbracket[u, \vec{x} \mapsto a, \vec{b}]$. By induction hypothesis, we have $d_i \in P$. If the head variable z is u , then $c = a \circ (\vec{d}) \in P$ because a was assumed to be Banach-Mazur over P . Otherwise, i.e. if z is one of the variables \vec{x}, \vec{y} , then c is λ -definable over P and hence in P because P is assumed to be λ -closed. \dashv

We are now in a position to complete the proof of proposition 3.1. We show, by induction on the lengths of terms $M \in \Lambda(\leq k+1)$ in long normal form with free variables among \vec{x} of types $\vec{\rho} \in T(\leq k+1)$, that for any tuple $\vec{a} \in P^+(\vec{\rho})$ we have $c := \llbracket M \rrbracket[\vec{x} \mapsto \vec{a}] \in P^+$. By lemma 3.2 it suffices to show that c is Banach-Mazur over P . To this end let $g_i \in P(\vec{\sigma} \rightarrow \rho_i)$ where $\vec{\sigma} < k$. We need to show $c \circ (\vec{g}) \in P$. Let $M = \lambda \vec{y}. z \vec{M}$ and set $d_i := \llbracket \lambda \vec{y}. M_i \rrbracket[\vec{x} \mapsto \vec{a}]$. By induction hypothesis, we have $d_i \in P^+(\leq k)$ and therefore $e_i := d_i \circ (\vec{g}) \in P$. Now, if $z = x_i$, then $c = a_i \circ (\vec{d})$ and therefore $c \circ (\vec{g}) = a_i \circ (\vec{e}) \in P$ because $a_i \in P^+$. If $z = y_i$, then $c \circ (\vec{g})$ clearly is λ -definable from \vec{g} and \vec{e} and therefore in P , since we assumed P to be λ -closed.

We define the the notion of a *substructure* of D analogous to k -substructures, but without the level restriction. Hence, a substructure

of D is a family $Q = \{Q(\rho) \mid \rho \in T\}$ such that $Q(\rho) \subseteq D(\rho)$ for each type ρ . A substructure Q is λ -closed if it contains all elements in D which are λ -definable over Q . A substructure Q is an *extension* of a k -substructure P if $Q(\rho) = P(\rho)$ for all $\rho \leq k$.

Given a λ -closed k -substructure P we define n -substructures P^n , for $n \geq k$, by $P^k = P$, $P^{n+1} = (P^n)^+$. We let P^ω be the substructure of D obtained by joining all P^n , i.e. for a type ρ of level n we set $P^\omega(\rho) := P(\rho)$ if $n \leq k$ and $P^\omega(\rho) := P^n(\rho)$ if $n > k$. Hence $P^\omega(\rho) = P^m(\rho)$ for all $m \geq \max(k, \text{level}(\rho))$.

COROLLARY 3.3. *Let P be a λ -closed k -substructure of a type structure D . Then P^ω is a λ -closed substructure of D which is maximal in the sense that there is no λ -closed substructure of D that extends P and properly contains P^ω .*

§4. Feasible higher type functionals. We briefly review the some concepts and results on feasible higher type functionals and relate them to our work in the previous section. An excellent introduction into higher type feasibility is Cook's article [2]. For simplicity we work within the type type structure of all (set-theoretic) hereditarily total functions over the base type $HT(\text{nat}) = \mathbb{N} = \{0, 1, 2, \dots\}$. We could equally well work with Kleene and Kreisel's model \mathcal{C} of *total continuous functionals* [9], [10] or the Scott/Ershov model $\hat{\mathcal{C}}$ of *partial continuous functionals* [5]. In fact we will investigate $\hat{\mathcal{C}}$ in the next section more closely. Let BFF be the substructure of *Basic Feasible Functionals* consisting of the denotations of the closed terms of Cook and Urquhart's system PV^ω [4]. Hence a functional is basic feasible iff it is λ -definable from polynomial time functions (of type level one) and the type 2 functional $\mathcal{R}: \text{nat} \rightarrow (\text{nat} \rightarrow \text{nat} \rightarrow \text{nat}) \rightarrow (\text{nat} \rightarrow \text{nat}) \rightarrow \text{nat} \rightarrow \text{nat}$

$$\mathcal{R}yghx = \begin{cases} y & \text{if } x = 0 \\ gx(\mathcal{R}ygh\lfloor x/2 \rfloor) \equiv: t & \text{if } x > 0 \text{ and } |t| \leq |hx| \\ hx & \text{otherwise} \end{cases}$$

that corresponds to Cobham's *limited recursion on notation* [1]. By definition BFF is λ -closed and, according to Cobham [1], an extension (in the sense of our section 3) of $PTIME$, the set of polynomial time computable functions. At least up to type 2, BFF is widely regarded a serious candidate for a good notion of feasibility in higher types. This is partly

due to various characterizations of BFF in terms of other (idealized) programming languages [3], [7] and, up to type 2, Oracle Turing Machines (OTMs) [3],[8].

EXAMPLE (Cook [2]). Define a relation \preceq on $\mathbb{N} \times \mathbb{N}$ by $(a, b) \preceq (a', b')$ iff $|a| < |b|$ or $|a| = |b|$ and $|a'| = |b'|$. By using a standard pairing function, we can assume \preceq is defined in \mathbb{N} . Define $L: (\mathbf{nat} \rightarrow \mathbf{nat}) \rightarrow \mathbf{nat}$ by $L(g) = \text{least } i > 0 \text{ such that } g(i-1) \preceq g(i)$.

§5. Systems of substructures.

§6. Substructures of the domain hierarchy. In this section we build the ambient type structure D within the cartesian closed category of effective Scott domains. By an *effective Scott-domain*, *domain* for short, we mean a bounded complete algebraic dcpo with an effective base, i.e. the compacts are numbered such that, with respect to the numbering, order and consistency between compacts is decidable and existing finite suprema of compacts can be effectively computed. For basic information on effective Scott domains we refer to Stoltenberg-Hansen, Griffor and Lindström [6]. We write $a \sqsubseteq b$ if a and b are related in the domain ordering. By \perp we denote the least element and by X_0 the set of compact elements of a domain X . As usual we call an element of a domain *computable* if the set (of codes) of its compact approximations are recursively enumerable.

We consider simple types built over the single base type \mathbf{nat} , i.e. $T_0 = \{\mathbf{nat}\}$. We define $D = \{D(\rho) \mid \rho \in T\}$ by $D(\mathbf{nat}) = \mathbb{N}_\perp =$ the flat domain $\{\perp\} \cup \mathbb{N}$, and $D(\rho \rightarrow \sigma) =$ the domain of all continuous functions from $D(\rho)$ to $D(\sigma)$, i.e. the exponential in the category of domains. Let $\nu_0^\rho: \mathbb{N} \rightarrow D_0(\rho)$ be an *effective base*, i.e. an effective enumeration of the compacts in $D(\rho)$. We may regard each effective base ν_0^ρ as an element of $D(\mathbf{nat} \rightarrow \rho)$ by identifying it with its strict extension, i.e. $\nu_0^\rho(\perp) := \perp$. Later in this section we will examine effective bases in greater detail.

By definition, an element $a \in D(\rho)$ is computable iff the set $\{n \mid \nu_0^\rho n \sqsubseteq a\}$ is recursively enumerable. There is a useful alternative characterization of computability: Let $\rho = \vec{\rho} \rightarrow \mathbf{nat}$. Then $a \in D(\rho)$ is computable iff $\lambda \vec{n}. a(\nu_0^{\rho_1} n_1) \dots (\nu_0^{\rho_k} n_k) \in D(\vec{\mathbf{nat}} \rightarrow \mathbf{nat})$ is computable.

Let P be a (k -)substructure of D . We call P *computable* if all elements of P are computable. We are interested in finding extra conditions on P which ensure that if P is computable, then P^ω is computable.

We say a k -substructure P has all effective bases if it contains all ν_0^ρ with $\rho \leq k$.

From the discussion above we immediately get:

LEMMA 6.1. *Let P be a λ -closed k -substructure of D which is computable and has all effective bases. Then P^+ is computable.*

Therefore, our question on the computability of P^ω can be reduced to the problem of finding conditions on P that ensure that if P has all effective bases, then P^+ has all effective bases.

In order to solve the latter problem we examine the structure of compact elements and their enumerations. Every element of $D(\text{nat})$ is compact and we set $\nu_0^{\text{nat}}0 = \perp$, $\nu_0^{\text{nat}}(n+1) = n$. How do the compacts of a function type $\rho \rightarrow \sigma$ look like? For compacts $v \in D(\rho)$ and $w \in D(\sigma)$ we define –following Plotkin’s notation [12]– the *step function* $[v \Rightarrow w] \in D(\rho \rightarrow \sigma)$ by

$$[v \Rightarrow w]a = \begin{cases} w & \text{if } v \sqsubseteq a \\ \perp & \text{otherwise} \end{cases}$$

The compact elements of $D(\rho \rightarrow \sigma)$ are exactly the existing suprema of finite sets of step functions. From a complexity theoretic point of view it is important that in any of our domains $D(\eta)$ the supremum of a set exists provided its elements are pairwise *consistent*, i.e. have a supremum. Two step functions $[v \Rightarrow w], [v' \Rightarrow w'] \in D(\rho \rightarrow \sigma)$ are consistent iff whenever v and v' are consistent, then w and w' are consistent.

From the considerations above it should be clear that for all types ρ, σ the effective base $\nu_0^{\rho \rightarrow \sigma}$ can be defined in such a way that there are polynomial time computable functions $\text{len}^{\rho, \sigma}: \mathbb{N} \rightarrow \mathbb{N}$, $\text{left}^{\rho, \sigma}: \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$, $\text{right}^{\rho, \sigma}: \mathbb{N} \rightarrow \mathbb{N} \rightarrow \mathbb{N}$ such that for all $n \in \mathbb{N}$, $\text{len}^{\rho, \sigma}n \leq |n|$ (the binary length of n) and $\nu_0^{\rho \rightarrow \sigma}n$ is the supremum of the step functions $[\nu_0^\rho(\text{left}^{\rho, \sigma}ni) \Rightarrow \nu_0^\sigma(\text{right}^{\rho, \sigma}ni)]$ with $i \leq \text{len}^{\rho, \sigma}n$.

Let $\mu \in D(\text{nat} \rightarrow \text{nat} \rightarrow \rho)$ and $g \in D(\text{nat} \rightarrow \text{nat})$ such that for all $n \in \mathbb{N}$ the supremum $\sup\{\mu nr \mid r \leq gn\}$ exists (in $D(\rho)$). Then we can define a function $\text{sup}(\mu, g) \in D(\text{nat} \rightarrow \rho)$, by

$$\text{sup}(\mu, g) := \sup\{\mu nr \mid r \leq gn\}$$

We call a set $A \subseteq D$ *closed under sharply bounded suprema* if whenever μ, g of the types above are in A and for all $n \in \mathbb{N}$, $gn \leq |n|$ and $\sup\{\mu nr \mid r \leq gn\}$ exists, then $\text{sup}(\mu, g) \in A$. Similarly we define

$$\text{inf}(\mu, g) := \inf\{\mu nr \mid r \leq gn\}$$

which exists without restriction on μ and g , because in a domain any nonempty set has an infimum. We call $A \subseteq D$ *closed under sharply bounded infima* if whenever μ, g of the types above are in A and $gn \leq |n|$ for all $n \in \mathbb{N}$, then $\inf(\mu, g) \in A$.

LEMMA 6.2. *Let P be a λ -closed k -substructure of D . Then, if P is closed under sharply bounded suprema (infima), so is P^+ .*

PROOF. Let $\mu \in P^+(\text{nat} \rightarrow \text{nat} \rightarrow \rho)$ and $g \in P^+(\text{nat} \rightarrow \text{nat})$ such that for all $n \in \mathbb{N}$ the supremum $\sup\{\mu nr \mid r \leq gn\}$ exists and $gn \leq |n|$. We need to show $\sup(\mu, g) \in P^+$. We have $\rho \leq k+1$, $g \in P$, and μ is Banach-Mazur over P . We need to show that $\sup(\mu, g)$ is Banach-Mazur over P . Let $\vec{\sigma} < k$, $\vec{\rho} = \rho_1, \dots, \rho_n$, $h \in P(\vec{\sigma} \rightarrow \text{nat})$, $h_i \in P(\vec{\sigma} \rightarrow \rho_i)$. We must show $\sup(\mu, g) \circ (h, \vec{h}) \in P(\vec{\sigma} \rightarrow \text{nat})$. Define $\tilde{\mu} \in D(\text{nat} \rightarrow \vec{\sigma} \rightarrow \text{nat})$ by $\tilde{\mu} n \vec{b} := \mu n (h_1 \vec{b}) \dots (h_n \vec{b})$. The function $\tilde{\mu}$ is λ -defined from μ and $\vec{h} \in P$ and its type level is $\leq k$. Hence $\tilde{\mu} \in P$ because μ is Banach-Mazur over P . Furthermore, $\sup\{\tilde{\mu} nr \mid r \leq gn\}$ exists. Therefore $\sup(\tilde{\mu}, g) \in P$ since P is closed under sharply bounded suprema. One easily verifies that for all $\vec{b} \in D(\vec{\sigma})$, $\sup(\tilde{\mu}, g)(h \vec{b})(h_1 \vec{b}) \dots (h_n \vec{b}) = (\sup(\mu, g) \circ (h, \vec{h})) \vec{b}$. This shows that $\sup(\mu, g) \circ (h, \vec{h})$ is λ -definable over P and hence in P since P is λ -closed.

For sharply bounded infima the proof is similar. \dashv

We call a function $f \in D(\text{nat} \rightarrow \text{nat})$ *partial polynomial time computable* if the corresponding function on codes, $g: \vec{\mathbb{N}} \rightarrow \mathbb{N}$, defined by $g \vec{n} = (\nu_0^{\text{nat}})^{-1}(f(\nu_0^{\text{nat}} n_1)) \dots (f(\nu_0^{\text{nat}} n_k))$ is polynomial time computable.

LEMMA 6.3. *Let P be a λ -closed (k -)substructure of D which is closed under sharply bounded suprema and infima and contains all partial polynomial time computable functions (of level 1). Then P has all effective bases.*

PROOF. For any type ρ we define $\sqsubseteq_0^\rho \in D(\text{nat} \rightarrow \rho \rightarrow \text{nat})$ by

$$\sqsubseteq_0^\rho na = \begin{cases} 0 & \text{if } \nu_0^\rho n \sqsubseteq a \\ \perp & \text{otherwise} \end{cases}$$

We prove that ν_0^ρ and \sqsubseteq_0^ρ are in P , by induction on the (syntactic) length of types ρ (of level $\leq k$). The case nat is trivial. For the case $\rho \rightarrow \sigma$ define first $\text{if}^\sigma \in D(\text{nat} \rightarrow \sigma \rightarrow \sigma)$ by $\text{if}^\sigma 0b = b$, $\text{if}^\sigma nb = \perp$ if $n \neq 0$, which is λ -definable from the corresponding $\text{if}^{\text{nat}} \in P(\text{nat})$ and hence in P . Now

define $\mu \in D(\text{nat} \rightarrow \text{nat} \rightarrow \rho \rightarrow \sigma)$ by

$$\mu ni = [\nu_0^\rho(\text{left}^{\rho,\sigma} ni) \Rightarrow \nu_0^\sigma(\text{right}^{\rho,\sigma} ni)]$$

We have $\mu nia = \text{if}^\sigma(\sqsubseteq_0^{\rho,\sigma}(\text{left}^{\rho,\sigma} ni)a)(\nu_0^\sigma(\text{right}^{\rho,\sigma} ni))$. Therefore $\mu \in P$, by induction hypothesis. Clearly $\nu_0^{\rho \rightarrow \sigma} = \text{sup}(\mu, \text{len}^{\rho,\sigma})$. In order to see that $\sqsubseteq_0^{\rho \rightarrow \sigma} \in P(\rho \rightarrow \sigma)$, define $\mu' \in D(\text{nat} \rightarrow \text{nat} \rightarrow (\rho \rightarrow \sigma) \rightarrow \text{nat})$ by

$$\mu' nif = \begin{cases} 0 & \text{if } [\nu_0^\rho(\text{left}^{\rho,\sigma} ni) \Rightarrow \nu_0^\sigma(\text{right}^{\rho,\sigma} ni)] \sqsubseteq f \\ \perp & \text{otherwise} \end{cases}$$

We have $\mu' \in P$ since $\mu' nif = \text{if}^{\text{nat}}(\sqsubseteq_0^\sigma(\text{right}^{\rho,\sigma} ni)(f(\nu_0^\rho(\text{left}^{\rho,\sigma} ni))))0$. Clearly $\sqsubseteq_0^{\rho \rightarrow \sigma} = \text{inf}(\mu', \text{len}^{\rho,\sigma})$. \dashv

By the corollary 3.3 and lemmas 6.2, 6.3, 6.1 we obtain

COROLLARY 6.4. *Let P be a computable λ -closed k -substructure of D which is closed under sharply bounded suprema and infima and contains all partial polynomial time computable functions. Then P^ω is computable and has all effective bases.*

PROPOSITION 6.5. *Let P be the set of all partial polynomial time computable functions. Then P^ω is a computable λ -closed substructure of D which extends P and has all effective bases.*

PROOF. Since P is a 1-substructure of D it suffices to show that P satisfies the conditions in corollary 6.4. Clearly P is computable and λ -closed (the latter just means, because $k = 1$, that P contains all projections and is closed under adding dummy arguments). P is also closed under sharply bounded suprema and infima since $\text{sup}(\mu, g)$ and $\text{inf}(\mu, g)$ can be defined from μ and g by Cobham's bounded recursion on notation [1]. \dashv

REFERENCES

§7. Extensionality.

[1] A. COBHAM, *The intrinsic computational difficulty of functions*, **Logic, Methodology and Philosophy of Science II** (Y. Bar-Hillel, editor), North-Holland, Amsterdam, 1965, pp. 24–30.

[2] S. COOK, *Computability and complexity of higher type functions*, **Logic from Computer Science, Proceedings of a Workshop held November 13–17, 1989** (Y.N. Moschovakis, editor), MSRI Publications, no. 21, Springer Verlag, Berlin, Heidelberg, New York, 1992, pp. 51–72.

- [3] S. COOK and B. KAPRON, *Characterizations of the basic feasible functionals of finite type*, **Feasible mathematics** (S. Buss and P. Scott, editors), Birkhäuser, 1990, pp. 71–96.
- [4] S. COOK and A. URQUHART, *Functional interpretations of feasibly constructive arithmetic*, **Annals of Pure and Applied Logic**, vol. 63 (1993), pp. 103–200.
- [5] Y.L. ERSHOV, *Model C of partial continuous functionals*, **Logic colloquium 1976** (R. Gandy and M. Hyland, editors), North Holland, Amsterdam, 1977, pp. 455–467.
- [6] E. GRIFFOR, I. LINDSTRÖM, and V. STOLTENBERG-HANSEN, **Mathematical theory of domains**, Cambridge University Press, 1993.
- [7] R. IRWIN, B. KAPRON, and J. ROYER, *On characterizations of the basic feasible functionals, Part I*, **Journal of Functional Programming**, vol. 11 (2001), no. 1, pp. 117–153.
- [8] B. KAPRON and S. COOK, *A new characterization of type 2 feasibility*, **SIAM Journal on Computing**, vol. 25 (1996), pp. 117–132.
- [9] S. C. KLEENE, *Countable functionals*, **Constructivity in mathematics** (A. Heyting, editor), North-Holland, Amsterdam, 1959, pp. 81–100.
- [10] G. KREISEL, *Interpretation of analysis by means of constructive functionals of finite types*, **Constructivity in Mathematics**, (1959), pp. 101–128.
- [11] G. LONGO and E. MOGGI, *The hereditarily partial effective functionals and recursion theory in higher types*, **The Journal of Symbolic Logic**, vol. 49 (1984), no. 4, pp. 1319–1332.
- [12] G. D. PLOTKIN, *LCF considered as a programming language*, **Theoretical Computer Science**, vol. 5 (1977), pp. 223–255.
- [13] V. SAZONOV and A. VORONKOV, *A construction of typed lambda models related to feasible computability*, LNCS, vol. 713, Springer Verlag, Berlin, Heidelberg, New York, 1993.

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF WALES SWANSEA
SINGLETON PARK
SWANSEA
SA2 8PP, UNITED KINGDOM
E-mail: u.berger@swan.ac.uk