Data Types as Algorithms

by

M.A. Nait Abdallah

Department of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
N2L 3G1

and

Laboratoire d'Informatique
Théorique et Programmation, Paris

Research Report CS-82-10
March, 1982

Research supported by the University of Waterloo under grant 126-7124-99

DATA TYPES AS ALGORITHMS

Areski Nait Abdallah

Résumé:

Le présent mémoire décrit une construction simplifiée d'un espace d'algorithmes. La notion de collection d'algorithmes est dûe à Nolin (1974). Le but a été ici de simplifier sa présentation et de l'enrichir de la notion de calculabilité par le biais de l'utilisation des faisceaux. On construit une collection d'algorithmes dénombrable dans laquelle tous les objets sont calculables.

Abstract:

This paper presents a simplified construction of an algorithm space. The notion of an algorithm collection goes back to Nolin 1974. Our goal is here to simplify this presentation of algorithm theory. The introduction of a computability notion was induced by the utilization of bundle theory. We build an enumerable algorithm collection where every object is computable.

Introduction

This paper presents a simplified construction of an algorithm space [4, 3]. The idea and the terminology go back to Nolin 1974, who wanted a semantics for a programming language with type declarations. However, a proof of the conceptual reasonableness of the idea, or in other words, a mathematical proof of the existence of such a space, was missing. Such a proof was first given in [2], using bundle theory, which, as shown in [3], underlies a significant part of programming language semantics model theory.

In [6], which is a shorter version of [5], Nolin and Le Berre present a simplified version of this proof, where they use "solely elementary set theory properties" [6]. Technically, this amounts to abandon the upper bundle structure used in [3]. Unfortunately the paper has an error in a crucial step of the argument: the representation of every normal self-function $f : E \to E$ as an element of the domain $E^{(*)}$ which makes the entire proof unsound.

The argument in [5,6] follows, in a "set of subsets" setting, the one given in [3], which referred to previous work by D. Scott 72 and C. Wadsworth 71. The goal is to obtain a domain E such that

$$E = D + [E \to E] = D + F$$

where $F = [E \to E] = \{f : E \to E \mid f \text{ is normal}\}$. The representation of normal functions as elements of E can be reduced to the representation

(*) For this discussion, the reader is referred to Cras, Litp for notation and definition details.

of threshold functions:

$$f_{xy} : z \to \underline{\text{if}} \;\; z \leq x \;\; \underline{\text{then}} \;\; y \;\; \underline{\text{else}} \;\; T$$

which we shall also denote $[x,y]$. Since in the projective limit E any element x verifies:

$$x = (x_p)_{p \in \mathbb{N}} = \bigsqcap_p x_p$$

one easily sees that

$$[x,y] = [\bigsqcap_p x_p \, , \, y] = \bigsqcup_p [x_p \, , \, y]$$

Thus it is sufficient to represent the threshold functions $[x_p \, , \, y]$.

The representation used by Nolin and Le Berre amounts to define

$$\forall\, f \in F \, , \;\; \forall\, x \in E$$

$$[f] : x \to [f](x) = \bigsqcap_n f_{n+1}(x_n) \quad \text{denoted} \;\; f[x] \;\text{in}\; [5] \;\text{p.22}$$

The element $f \in F$ intended to represent the functions $[x_p \, , \, y]$ is the projective sequence

$$g^p = (r^p(f_{x_p y_p}) \, , \ldots , \, r(f_{x_p y_p}) \, , \, f_{x_p y_p} \, , \, f_{e(x_p) y_{p+1}} \, , \, f_{e^2(x_p) y_{p+2}} \, , \ldots)$$

(cf. [6] pp. 501, [5] pp. 25).

Lemma 6 of [6], whose main theorem is a consequence, states that

$$[x_p , y] = [g^p]$$

This lemma does not hold. Indeed, for any two projective sequences $(a_n)_{n\in\mathbb{N}}$, $(b_n)_{n\in\mathbb{N}}$ we have either $(a_n) \leq (b_n)$ or $(a_n) \nleq (b_n)$. Or in other words:

$$\text{either} \quad \forall_n \quad a_n \leq b_n$$

$$\text{or} \qquad \exists\, N \in \mathbb{N} \quad n > N \Rightarrow a_n \nleq b_n ,$$

since we have projective sequences and the projections r are monotone. The first case applied to $[g^p](z)$ gives:

$$z \leq x_p \Rightarrow [x_p , y](z) = y \quad \text{and}$$

$$[g^p](z) = \sqcap\{\ldots\ldots, y_p, y_{p+1} , y_{p+2} , \ldots\} = y$$

Here the first p elements of the sequence $((g^p)_{n+1}(z_n))_{n\in\mathbb{N}}$ are of no importance since the rest of the sequence is projective.

The second case applied to $[g^p](z)$ gives:

$$z \nleq x_p \Leftrightarrow z_p \nleq x_p \Rightarrow [x_p , y](z) = T \quad \text{and for any} \quad n \geq p$$

$$(g^p)_{n+1}(z_n) = [e^{n-p}(x_p), y_n](z_n) = T_n$$

Thus:

$$[g^p](z) = \sqcap\{\ldots, T_p , T_{p+1} , T_{p+2} , \ldots\}$$

But here the first p elements of the sequence $((g^p)_{n+1}(z_n))_{n\in\mathbb{N}}$ are very important since the others are top elements, and we are taking a greatest lower bound. Thus

$$(r[x_p\ ,\ y_p])(z_{p-1}) = \underline{\text{if}}\ \ e(z_{p-1}) \le x_p\ \ \underline{\text{then}}\ \ y_p\ \ \underline{\text{else}}\ \ T_p$$

Unfortunately we know nothing about

$$e(z_{p-1}) \le x_p$$

which may or may not be true. As an example if $z \in D$ and $x_p \in F$, then

$$r^p(x_p) = (x_p)_0 = x_0 = \gamma$$

$$[x_p\ ,\ y](z) = T$$

whereas $\quad [g^p](z) = y_0 = e_{0\infty}\ (y_0)$.

Which means that outside of the lowerset

$$\downarrow x_p = \{z \in E : z \le x_p\}$$

the two functions $[x_p\ ,\ y]$ and $[g^p]$ are unequal. Therefore the projective sequence g^p does not represent the threshold function $[x_p\ ,\ y]$. As a side remark, one may notice that the correctness of Nolin and Le Berre's proof would have given a positive answer to the following question:

> Is there a poset X, which has more than one element, such that the set $(X \to X)$ of monotone self functions has the same cardinality as X ?

In this paper a simplified construction of an algorithm space is presented. The intent of this construction is to enrich and simplify Nolin's original presentation of algorithm theory.

The simplification we propose is a better mastering of algorithm collections cardinality: in fact we describe an <u>enumerable</u> algorithm collection. The enrichment is the introduction of <u>computability</u> notions: all our algorithms will be computable in some sense. And this makes the whole algorithm collection itself effectively presentable.

I. Bundle Structures over $\mathcal{P}(\mathbb{N})$:

Let $\mathbb{N}$ be the set of integers and $E = \mathcal{P}(\mathbb{N})$ be the set of subsets of $\mathbb{N}$. We have

$$\forall x \in E \qquad x = \bigcup_{m \in x} \{m\} = \bigcup_{a_n \subseteq x} a_n$$

if we define an enumeration $a : \mathbb{N} \to E$,

$$a : n \to \{n - 1\} \qquad \text{if} \quad n \neq 0$$

$$a_0 = \phi \qquad \text{where} \quad \phi \quad \text{is the empty set}$$

This gives an elementary monic ordered bundle structure over E [3] . The spectra are defined by:

$$s : E \to \mathcal{P}(E)$$
$$x \to \{a_n : a_n \subseteq x\}$$

The elements a_n belong to their own spectrum and are therefore called rationals. They constitute the kernel of the bundle, which means that every $x \in E$ can be obtained as the union of some well-chosen rationals (in fact the elements of the spectrum of x, $s(x)$). The function $a : \mathbb{N} \to E$, $n \to a_n$ gives an enumeration of the kernel. The limit function is simply the set-theoretical union of elements of E. This bundle structure will be called the lower bundle structure of E.

On the other hand we also have:

$$\forall x \in E \quad x = \bigcap_{x \subseteq b_m} b_m$$

if we define an enumeration

$$b : \mathbb{N} \to E$$
$$m \to b_m = \complement\{k_0 , k_1 ,\dots, k_{p-1}\}$$

where $m = \sum_{i<p} 2^{k_i}$, $k_0 < k_1 < \ . \ . \ . < k_{p-1}$

(Thus we use the dyadic expansion of integer m) . Notice that $b_0 = \mathbb{N}$. The set of b_m's is closed under finite intersection, and each b_m verifies the following algebraicity property:

For any descending chain $\{x_i\}_{i\in I}$ of elements of E , $b_m \supseteq \bigcap_i x_i \Rightarrow \exists i \ b_m \supseteq x_i$. The <u>spectrum</u> function

$$s : E \to P(E)$$
$$x \to s(x) = \{b_m : b_m \supseteq x\}$$

defines an algebraic monic ordered bundle for the inverse inclusion over E . Every b_m is <u>rational</u>, and the <u>kernel</u> of the bundle is exactly the set of all b_m's .

The spectra are obviously closed under finite intersection, and the <u>limit</u> function is simply the set-theoretical intersection of elements of E .

This structure will be called the <u>upper bundle</u> structure of E.

Indeed the set of b_m's is exactly the set of all co-finite subsets of $\mathbb{N}$. Every co-finite set is recursive, thus recursively enumerable. Furthermore the relations $a_n \subseteq a_m$, $b_n \subseteq b_m$, $a_n \subseteq b_m$, $b_m = b_n \cap b_p$ are all recursive in the indices m, n, p .

II. Computability in $E = \mathcal{P}(\mathbb{N})$:

2.1 Computable Elements of $\mathcal{P}(\mathbb{N})$:

We say that an element $x \in E$ is inf-computable, i.e., computable when considered inside the lower bundle structure, if and only if the set $\{n : a_n \subseteq x\}$ is recursively enumerable in the index n. We see at once that:

$$x \in E \text{ is inf-computable} \Leftrightarrow x \text{ is r.e.}$$

We also define: $x \in E$ is sup-computable if and only if the set $\{m : x \subseteq b_m\}$ is recursively enumerable in the index m.

Fact 1:

$x \in E$ is sup-computable $\Leftrightarrow \complement x$ is inf computable $\Leftrightarrow \complement x$ is r.e.

Proof:

$$\{m : x \subseteq b_m\} = \{m : x \subseteq \complement\{k_0, k_1, \ldots, k_{p-1}\},$$

$$m = \sum_{i<p} 2^{k_i}, \; k_i \text{ all different}\}$$

$$x \subseteq \complement\{k_0, k_1, \ldots, k_{p-1}\} \Leftrightarrow \{k_0, k_1, \ldots, k_{p-1}\} \subseteq \complement x$$

i.e., we have to enumerate all finite subsets of $\complement x$. In fact one can show that $\forall y \subseteq \mathbb{N}$ if $\mathcal{P}_{fin}(y)$ is the set of finite subsets of y, then

$$y \text{ is r.e.} \Leftrightarrow \mathcal{P}_{fin}(y) \text{ is r.e.}$$

For:

y r.e. $\Rightarrow \mathcal{P}_{fin}(y)$ r.e. : we take a recursive enumeration of y , and we enumerate all finite subsets of y we can construct.

$\mathcal{P}_{fin}(y)$ r.e. $\Rightarrow y$ r.e. : we take a recursive enumeration of $\mathcal{P}_{fin}(y)$ and we evaluate, in an effective manner, the cardinal of each (finite) subset of y . If this cardinal is one, the subset is added to the enumeration of y , otherwise we discard the subset and consider the next one.

Whence the three equivalences. $\square$

<u>Definition</u>:

$x \in E$ is <u>computable</u> if and only if x is inf-computable and x is sup-computable. $\square$

In a programming language, the availability of a ground data type, say <u>integer</u>, amounts to the availability of a procedure with one variable, <u>int</u>(x) , such that, for any input data a , the call <u>int</u>(a) returns the value <u>true</u> if $a \in \mathbb{N}$ and <u>false</u> otherwise. This is exactly realized by the recursive subsets of $\mathbb{N}$. More explicitly:

<u>Lemma 2</u>: $\forall x \in E = \mathcal{P}(\mathbb{N})$

(i) x is inf-computable $\Leftrightarrow x$ is r.e.

(ii) x is sup-computable $\Leftrightarrow \mathcal{C}x$ is r.e.

(iii) x is computable $\Leftrightarrow x$ is recursive $\square$

As a summary:

1. The computable elements of E are exactly the recursive subsets of $\mathbb{N}$.
2. E is an elementary monic ordered bundle for $\subseteq$, with ϕ and the singletons as rational elements and the r.e. sets included in $\mathbb{N}$ as inf-computable elements. The set of inf-computable (resp. computable) elements forms a sub-bundle of E .
3. E is an algebraic bundle for $\supseteq$, with the cofinite subsets as rational elements, and has as sup-computable elements (i.e., computable for this bundle) all subsets $\subseteq \mathbb{N}$ whose complementary is r.e. . The set of sup-computable (resp. computable) elements of E forms an algebraic sub-bundle of E .

2.2 Computable Sequences:

We define computations in E . We call them computable sequences.

Definition:

A sequence $\{x_p\}_{p\in\mathbb{N}}$ of elements of E is said to be sup computable (resp. inf-computable) if and only if there exists a recursive function $\psi : \mathbb{N}^2 \to \mathbb{N}$, such that for any $p \in \mathbb{N}$, $\psi(p.,)$ is an enumeration of the (indices of the) spectrum of x_p for the upper bundle (resp. the lower bundle) structure of E . □

As a consequence, every term x_p of a computable sequence $\{x_p\}_{p\in\mathbb{N}}$ is computable according to the bundle structure considered (i.e., x_p is inf-computable if the sequence is inf-computable; and x_p is sup-computable if the sequence is).

There is an important fact about the compatibility of the computability and bundle notions in E .

Lemma 3: For any sequence $\{x_p\}_{p\in\mathbb{N}}$ in E ,

(i) if $\{x_p\}_{p\in\mathbb{N}}$ is inf-computable, then its limit in the lower bundle $\bigcup_p x_p$ is inf-computable.

(ii) if $\{x_p\}_{p\in\mathbb{N}}$ is sup-computable, then its limit in the upper bundle $\bigcap_p x_p$ is sup-computable.

Proof:

(i) We know that inf-computability amounts to recursive enumerability. Let $\psi : \mathbb{N}^2 \to \mathbb{N}$ be the function associated with the sequence $\{x_p\}_{p\in\mathbb{N}}$. The function

$$u : \mathbb{N}^2 \to \mathbb{N} \ , \ (m,n) \to \frac{1}{2}(n+m)(n+m+1) + m$$

is primitive recursive and bijective. Its inverse

$$v : \mathbb{N} \to \mathbb{N}^2 \ , \ \ p \to (p_1, p_2)$$

is also primitive recursive and it enumerates $\mathbb{N}^2$ along the "little diagonals", going from left to right:

$$(0,0),\ (1,0),\ (0,1),\ (2,0),\ (1,1),\ (0,2),\ (3,0),\ \ldots$$

Therefore:

$$g(p) = \psi(p_1,\ p_2) = \psi(U_2^1(v(p))\ ,\ \ U_2^2(v(p)))$$

is a recursive enumeration of the (indices of the) spectrum of $\bigcup_p x_p$. Hence $\{n : a_n \subseteq \bigcup_p x_p\}$ is r.e. , i.e., $\bigcup_p x_p$ is inf-computable.

(ii) Similarly, we have

$$g(p) = \psi(p_1,\ p_2) = \psi(U_2^1(v(p))\ ,\ \ U_2^2(v(p)))$$

is a recursive enumeration of the spectrum of $\bigcap_p x_p$. Therefore $\{m : b_m \supseteq \bigcap_p x_p\}$ is recursively enumerable, i.e., $\bigcap_p x_p$ is sup-computable. □

2.3 C-domains:

Let X be a poset. An element $u \in X$ is c-algebraic iff for any decreasing computable chain $\{x_k\}_{k\in\mathbb{N}}$ which has a glb in X , $u \geq \bigcap_k x_k \Rightarrow \exists k \;\; u \geq x_k$. As an example, in the poset E ordered by inclusion, every b_m is c-algebraic. More generally, in any c.p.o. for the opposite order which has an enumerable set of compact elements, every compact element is c-algebraic.

<u>Definition</u>: C-domains

A poset X is a c-domain iff

(i) X has a largest element

(ii) Every decreasing computable sequence has a greatest lower bound in X .

(iii) An enumeration $a : \mathbb{N} \to X$ of the set of c-algebraic elements is given, and this enumeration verifies:

$$\forall\, x \in X \quad x = \sqcap\{a_n : a_n \geq x\} \quad \text{and}$$

$$\{n : a_n \geq x\} \text{ is recursively enumerable .} \qquad \square$$

Examples of c-domains included in E are $X = \{\{n\} : n \in \mathbb{N}\} \cup \{\mathbb{N}\}$, and $X = Y \cup \{\mathbb{N}\}$ where $Y \subseteq \mathcal{P}_{fin}(\mathbb{N})$ is any set of finite subsets of $\mathbb{N}$ which is closed under finite intersection. However some c-domain included in E plays a special role:

<u>Lemma 4</u>: The set E_s of sup-computable elements of E is the unique c-domain such that:

(i) it is included in $E = \mathcal{P}(\mathbb{N})$ as a poset

(ii) it has the enumeration b and contains every recursive subset $x \subseteq \mathbb{N}$ as an element. $\square$

Proof:

(i) E_s is a c-domain by lemma 3.

(ii) Let $X \subset P(\mathbb{N})$ be a c-domain containing every computable (recursive) element of E. Then $b_m \in X$ for every m, whence every sup-computable element is in X. Thus $E_s \subseteq X$. If $x \notin E_s$, then $\{m : b_m \supseteq x\}$ is not r.e., i.e., $x \notin X$. Therefore $X = E_s$. □

In fact it appears that, once $b : \mathbb{N} \to E$ is chosen, E_s is, by construction, the largest c-domain contained in $E = P(\mathbb{N})$

III. Computable Functions:

3.1 Computable functions:

A function $f : E \to E$ is said to be regular iff it is regular for the lower bundle i.e., iff

$$\forall\, x \in E \quad f(x) = \bigcup_{a_n \subseteq x} f(a_n) = \bigcup_{n \in x} f(\{n\})$$

A regular function $f : E \to E$ is computable iff the set $\{(m,n) : f(a_n) \subseteq b_m\}$ is recursively enumerable in the indices m, n.

Since $f = \sqcap\{[a_n, b_m] : f(a_n) \subseteq b_m\}$ [3], the definition of a computable function amounts to the definition of a recursive enumeration of the set of threshold functions

$$\{[a_n, b_m] : f(a_n) \subseteq b_m\}$$

approximating f. If we define

$$(E \to E) = \{f : E \to E \mid f \ \text{regular}\}$$
$$[E \to E] = \{f : E \to E \mid f \ \text{computable}\}$$

supplied with the extensional order

$$f \leq g \Leftrightarrow \forall\, x \quad f(x) \subseteq y(x)$$

there is a canonical bijection between $[E \to E]$ (resp. $(E \to E)$) and the set of principal lowersets of $[E \to E]$ (resp. of $(E \to E)$):

$$\downarrow[E \to E] = \{\downarrow f : f \in [E \to E]\}$$

where $\downarrow f = \{g \in [E \to E] : y \leq f\}$ (resp ...), defined by:

$$[E \to E] \underset{r}{\overset{s}{\leftrightarrows}} \downarrow[E \to E]$$

$$s(y) = \bigsqcup y = \max y$$

$$r(x) = \downarrow x = \{z : z \leq x\}$$

Thus the above definition of computability gives a better insight in Nolin's definition of an algorithm (= a principal lowerset) through intersections

$$A = \bigcap_{i \in I} F\, X_i\, Y_i$$

where $X_i = \downarrow x_i$, $Y_i = \downarrow y_i$, $F\, X_i\, Y_i = \downarrow[x_i, y_i]$. Here what is requested by the definition is a recursive enumeration of the family $\{F\, X_i\, Y_i\}_{i \in I}$ i.e., $\{[x_i, y_i]\}_{i \in I}$. This is made precise by the following lemma.

<u>Lemma 5</u>: The threshold function

$[x, y] = \lambda t \in E.$ <u>if</u> $t \subseteq x$ <u>then</u> y <u>else</u> $\mathbb{N}$

is computable if and only if x is inf-computable and y is sup-computable.

<u>Proof</u>:

$\{(m,n) : [x, y](a_n) \subseteq b_m\} =$

$\{(m,n) :$ <u>if</u> $a_n \subseteq x$ <u>then</u> y <u>else</u> $\mathbb{N} \subseteq b_m\} =$

$\{(m,n) : a_n \subseteq x$ and $y \subseteq b_m\} =$

$\{m : y \subseteq b_m\} \times \{n : a_n \subseteq x\}$

Hence $\{(m,n) : [x, y](a_n) \subseteq b_m\}$ is r.e. iff $\{m : y \subseteq b_m\}$ is r.e. and $\{n : a_n \subseteq x\}$ is r.e. [1]. Whence the lemma by Lemma 1. □

It is worthwhile to notice here that computable functions $f : E \to E$ may be coded, via the bijection (u,v) as r.e. subsets of $\mathbb{N}$, i.e., $\forall f : E \to E$ computable we define

$$\mathrm{graph}^*(f) = \{(m,n) \mid f(a_n) \subseteq b_m\} =$$
$$\{(m,n) \mid f(\{n-1\}) \subseteq b_m\}$$

The corresponding definition in $P\omega$ [8] is:

$$\mathrm{graph}(f) = \{(m,n) \mid \{m\} \subseteq f(e_n)\}\ ,\ e_n = Cb_n$$

Any element $u \in P\omega$ operates as a continuous function by

$$\mathrm{fun}(u)(x) = \{m \mid \exists\, e_n \subseteq x : (n,m) \in u\}$$

The corresponding definition for our functions is for any $u \in E = P(\mathbb{N})$

$$\mathrm{fun}^*(u)(x) = \bigcup_{n \in x} (\bigcap\{b_m : (n+1,\ m) \in u\})$$

One easily verifies that:

(i) for any regular $f : E \to E$, $f = \mathrm{fun}^*(\mathrm{graph}^*(f))$

(ii) for any $u \in E$, $u \subseteq \mathrm{graph}^*(\mathrm{fun}^*(u))$, where the equality holds iff

$$\{(p,q) : \bigcap\{b_m : (p+1,\ m) \in u\} \subset b_q\} \subseteq u$$

(iii) Any computable $f : E \to E$ yields, by definition, a recursively enumerable $\mathrm{graph}^*(f)$. However, a r.e. set $u \in E$ defines a computable function $\mathrm{fun}^*(u) : E \to E$ iff

$\{(p,q) : \bigcap\{b_m : (p+1,\ m) \in u\} \subseteq b_q\}$ is recursively enumerable.

However, this analogy between $P\omega$ and our construction will become fuzzier at higher levels of functionality, due to our use of Wadsworth scheme (cf. infra) .

Actually we can define $F_{ab} = \downarrow[a,b]$

$$\downarrow[a,b] = \{f \in [E \to E] : f(a) \subseteq b\} = \{f \in [E \to E] : f \leq [a,b]\}$$

Obviously $\forall f \in [E \to E]$,

$$\downarrow f = \bigcap\{\downarrow[a,b] : f \leq [a,b]\}$$

or $\quad \uparrow f = \bigcup\{\uparrow[a,b] : f(a) \subseteq b\}$

The relation

$$\forall\ f \in (E \to E) \quad f = \sqcap\{[a_n, b_m] : f(a_n) \subset b_m\}$$

gives a bundle structure over the set of regular functions $(E \to E)$; more precisely this makes $(E \to E)$ an elementary monic ordered bundle, with

$$s(f) = \{[a_n, b_m] : f(a_n) \subset b_m\}$$

as a spectrum function. This structure may be made algebraic by taking the closure of the spectra for finite greatest lower bounds. The set $[E \to E]$ forms a subbundle for this structure.

An enumeration of the kernel is given by:

$$\beta : m \to \beta_m = e_{k_0} \sqcap \ldots . \sqcap\ e_{k_{p-1}}$$

with $\qquad m = \sum_{i<p} 2^{k_i} \qquad k_0 < k_1 < \ldots < k_{p-1}$

$$e_{k_i} = [a_{U_2^1(v(k_i))}, b_{U_2^2(v(k_i))}]$$

where function v , here applied to k_i , is the inverse of function u (recursive enumeration of $\mathbb{N}^2$, cf supra) .

The set of β_m's is closed under finite $\sqcap$ and every β_m verifies the following algebraicity property:

$$\forall \{x_i\}_{i \in I} \quad \beta_m \geq \underset{i}{\sqcap}\, x_i \Rightarrow \exists\, i \quad \beta_m \geq x_i$$

Moreover, for the computability aspect, we have

Lemma 6:

The relations $\beta_m \leq \beta_n$, $[a_n, b_m] \geq \beta_p$, $\beta_m = \beta_n \sqcap \beta_p$ are all recursive in the indices m, n, p .

Proof:

We consider the proof for $\beta_m \leq \beta_n$. First notice that

$$[a, b] \leq [c, d] \Leftrightarrow (c \subseteq a \wedge b \subseteq d) \text{ or } d = \mathbb{N}$$

Thus $\quad [a_n, b_m] \leq [a_{n'}, b_{m'}] \Leftrightarrow$

$$a_{n'} \subseteq a_n \text{ (recursive)} \wedge b_m \subseteq b_{m'} \text{ (recursive)}$$

or $\quad b_{m'} = \mathbb{N}$ (recursive)

Therefore the relation $[a_n, b_m] \leq [a_{n'}, b_{m'}]$ is recursive in the indices n, m, n', m' . Now consider $\beta_m \leq \beta_n$. We have:

$$\beta_m = e_{k_0} \sqcap \ldots \sqcap e_{k_{p-1}} \qquad m = \sum_{i<p} 2^{k_i}$$

$$e_{k_i} = [a_{U_2^1(v(k_i))}, b_{U_2^2(v(k_i))}]$$

$$\beta_n = e_{\ell_0} \sqcap \ldots \sqcap e_{\ell_{q-1}} \qquad n = \sum_{j<q} 2^{\ell j}$$

$$e_{\ell_j} = [a_{U_2^1(v(\ell j))}, b_{U_2^2(v(\ell j))}]$$

Thus $\beta_m \leq \beta_n \Leftrightarrow$

$$e_{k_0} \sqcap \ldots \sqcap e_{k_{p-1}} \leq e_{\ell_0} \sqcap \ldots \sqcap e_{\ell_{q-1}}$$

$\Leftrightarrow$ (simplifying the notations)

$$[a_{k_0}, b_{k_0}] \sqcap \ldots \sqcap [a_{k_{p-1}}, b_{k_{p-1}}] \leq [a_{\ell_0}, b_{\ell_0}] \sqcap \ldots \sqcap [a_{\ell_{q-1}}, b_{\ell_{q-1}}]$$

$\Leftrightarrow$

$$\cap\{b_{k_i} : a_{\ell_0} \leq a_{k_i}\} \subseteq \cap\{b_{\ell_j} : a_{\ell_0} \leq a_{\ell_j}\}$$

<u>and</u> $$\cap\{b_{k_i} : a_{\ell_1} \leq a_{k_i}\} \subseteq \cap\{b_{\ell_j} : a_{\ell_1} \leq a_{\ell_j}\}$$

$$\vdots$$

<u>and</u> $$\cap\{b_{k_i} : a_{\ell_{q-1}} \leq a_{k_i}\} \subseteq \cap\{b_{\ell_j} : a_{\ell_{q-1}} \leq a_{\ell_j}\}$$

Each of these inequalities is decidable. Thus the inequality $\beta_m \leq \beta_n$ is decidable (recursive) in the indices m, n.

We have a similar argument for the other relations. Whence the lemma. $\square$

<u>Lemma 7</u>:

The set of computable functions $f : E \to E$ which have the following <u>simplicity</u> property

$$\forall\, a_n\ \exists\, m \quad f(a_n) = a_m$$

are closed under composition.

Proof:

Let $E \xrightarrow{f} E \xrightarrow{g} E$ be two computable functions.

f computable $\Leftrightarrow \{(m,n) : [a_n, b_m] \geq f\}$ is r.e.

g computable $\Leftrightarrow \{(p,q) : [a_p, b_q] \geq g\}$ is r.e.

The spectrum of gof, which is a monotone function, is given by:

$$s(gof) = \{[a_n, b_s] : \exists\, m\ [a_n, b_m] \in s(f)\ ,$$
$$\exists\, p\ \ a_p \leq b_m \text{ and } [a_p, b_s] \in s(g)\}$$

$$x \subseteq a_n \Rightarrow f(x) \subseteq b_m \quad \text{and} \quad g(b_m) = \cup\{g(a_p) : a_p \subseteq b_m\}$$

The relation $a_p \subseteq b_m$ is recursive, thus it is enough to transform through g the elements $a_p \subseteq b_m$ in order to find out $s(gof)$. If $b_m = \bigcup_i a_{p_i}$, then we have the diagram:

$$a_n \xrightarrow{f} b_m = \left[\begin{array}{lcl} a_{p_0} & \xrightarrow{g} & b_{q_0} \\ a_{p_1} & \rightarrow & b_{q_1} \\ \vdots & & \\ a_{p_i} & \rightarrow & b_{q_i} \\ \vdots & & \end{array}\right.$$

This gives $[a_n, \bigcup_i b_{q_i}]$ as approximating gof, i.e., $(gof)(a_n) \subset \bigcup_i b_{q_i}$. The set $b_m = \bigcup_i a_{p_i}$ is recursive, thus the sequence $\{b_{q_i}\}$ is recursively enumerable since $\{(p,q) : [a_p, b_q] \geq g\}$ is recursively enumerable. Therefore $\bigcup_i b_{q_i}$ is r.e., i.e., <u>inf-computable</u>. What we need is the sup-computability of $\bigcup_i b_{q_i}$ in order to have a r.e. decomposition

$$[a_n, \bigcup_i b_{q_i}] = \prod_k [a_n, b_k]$$

from which we would deduce a r.e. spectrum for gof. Buf if we impose that gof be simple, i.e., $f(a_n) = a_m$ for some $p \in \mathbb{N}$, then, since $g = \prod\{[a_p, b_q] : g(a_p) \subseteq b_q\}$, $(gof)(a_n) = g(a_m) = \bigcap_q \{ b_q : a_m \subseteq a_p\}$ which is a sup-computable element of E , whence an r.e. spectrum for gof:

$$s(gof) = \{[a_n, b_s] : f(a_n) = a_p \text{ , } g(a_p) \subseteq b_s\} \qquad \square$$

<u>Lemma 8</u>: There is a canonical bijection between the simple computable functions $f : E \to E$ and the recursive functions from $\mathbb{N}$ to $\mathbb{N}$.

<u>Proof</u>:

Obvious.

<u>Lemma 9</u>: Let $f : E \to E$ be a computable function and $\{x_p\}_{p\in\mathbb{N}}$ be an inf-computable sequence of rational elements of E . Then the image $\{f(x_p)\}_{p\in\mathbb{N}}$ of the sequence is a sup-computable sequence of E .

Proof:

$(x_p)_{p\in\mathbb{N}}$ inf computable sequence $\Leftrightarrow \exists\, \psi: \mathbb{N}^2 \to \mathbb{N}$ recursive such that $\psi(p,.)$ enumerates $s(x_p)$. f computable $\Leftrightarrow \{(m,n) : f(a_n) \subseteq b_m\}$ is r.e. . Since every x_p is rational, $\{f(x_p) = \cap\{b_m : x_p \subseteq a_n$, $f(a_n) \subseteq b_m\}$ the relation $x_p \subseteq a_n$ is recursive, the relation $f(a_n) \subseteq b_m$ is recursively enumerable, thus the set $\{m : x_p \subseteq a_n,\ f(a_n) \subseteq b_m\}$ is r.e. which implies that $\{m : b_m \supseteq f(x_p)\}$ is r.e. . Whence a recursive function $\psi : \mathbb{N}^2 \to \mathbb{N}$ such that $\psi(p,.)$ enumerates the spectrum of $f(x_p)$ for the upper bundle:

$$\psi(p,q) = (\text{enumeration of } \{m : b_m \supseteq f(x_p)\})(q)$$

Thus $\{f(x_p)\}_{p\in\mathbb{N}}$ is sup-computable. □

3.2 Computable sequences of functions:

Definition:

A sequence $\{f_p\}_{p\in\mathbb{N}}$ of regular functions from E to Ė is a computable sequence iff $\exists\, \psi: \mathbb{N}^2 \to \mathbb{N}$ recursive such that $\forall\, p \in \mathbb{N}$ $\psi(p,.)$ is an enumeration of the spectrum of f_p. □

In particular every f_p will be a computable function. We have an analogous of Lemma 3 (i) for computable sequences of functions:

<u>Lemma 10</u>: If $\{f_p\}_{p\in\mathbb{N}}$ is a computable sequence of functions, then its greatest lower bound $\underset{p}{\sqcap} f_p$ is a computable function.

<u>Proof</u>:

The function $\underset{p}{\sqcap} f_p$ is regular since $(\underset{p}{\sqcap} f_p)(a_n) = \underset{p}{\sqcap} f(a_n)$ for every rational a_n, and we take the least regular extension of this to non rational elements:

$$(\underset{p}{\sqcap} f_p)(x) = \bigcup\{(\underset{p}{\sqcap} f_p)(a_n) : a_n \subseteq x\}$$

Now we just have to glue the spectra together:

$$s(\underset{p}{\sqcap} f_p) = \underset{p}{\bigcup} s(f_p)$$

By means of the indices, the sequence $\{s(f_p)\}_{p\in\mathbb{N}}$ defines an inf-computable sequence of E. Hence $\underset{p}{\bigcup} s(f_p)$ is inf-computable, therefore recursively enumerable. $\square$

Let us define for any regular $f \in (E \to E)$, f is <u>finitely computable</u> iff the set $\{(m,n) : f(a_n) \subseteq b_n\}$ is <u>recursive</u>. Then we have the analogous of lemma 4:

<u>Lemma 11</u>: Given the enumeration β of the c-algebraic elements, the set of computable functions $[E \to E]$ is the unique c-domain such that

(i) it is included in $(E \to E)$ as a poset

(ii) it contains every finitely computable function. $\square$

Proof:

(i) $[E \to E]$ is a c-domain: The largest element of $[E \to E]$ is the constant function $x \to \mathbb{N}$. Every decreasing computable sequence has a glb in $[E \to E]$ by lemma 10. The other requirements are trivially fulfilled. (ii) Let $X \subseteq (E \to E)$ be a c-domain containing the finitely computable functions. Then every β_m is in X , therefore E_s which is the closure of $\{\beta_m : m \in \mathbb{N}\}$ for the glb's of decreasing computable sequence is included in X . Thus $[E \to E] \subseteq X$ and one easily sees that $[E \to E] = X$.

IV. Wadsworth scheme:

We have a c-domain structure over $[E \to E] = \Delta_1$. We can define the partial computable function spaces $[E \to \Delta_1]$, $[\Delta_1 \to E]$, $[\Delta_1 \to \Delta_1]$

$$[E \to \Delta_1] = \{f : f(x) = \bigcup_{a_n \subseteq x} f(a_n) \quad \text{and} \quad \{(m,n) : f(a_n) \leq \beta_m\} \text{ is r.e.}\}$$

$$[\Delta_1 \to E] = \{f : f(\sqcap_k x_k) = \sqcap_k f(x_k) \text{ for every decreasing computable sequence } (x_k) \text{ and } \{(m,n) : f(\beta_n) \subseteq b_m\} \text{ is r.e.}\}$$

$$[\Delta_1 \to \Delta_1] = \{f : f(\sqcap_k x_k) = \sqcap_k f(x_k) \text{ for every decreasing computable sequence } (x_k) \text{ and } \{(m,n) : f(\beta_n) \leq \beta_m\} \text{ is r.e.}\}$$

We must be careful here, we are dealing with partial functions in the set-theoretical sense.

Lemma 12: The spaces $[E \to \Delta_1]$, $[\Delta_1 \to E]$, $[\Delta_1 \to \Delta_1]$, when supplied with the extensional order, have all a c-domain structure.

Proof:

Analogous to Lemma 11. Only the rational elements change, and we use the same enumeration technique as for $[E \to E]$.

Then we can define

$$\Delta_1 = [E \to E] \quad , \quad A_1 = E + \Delta_1$$

$$A_2 = E + \Delta_2 = E + [E + \Delta_1 \to E + \Delta_1]$$

where

$$\Delta_2 = [E + \Delta_1 \to E + \Delta_1] =$$

$$[E \to E + \Delta_1] \times [\Delta_1 \to E + \Delta_1]$$

with

$$[E \to E + \Delta_1] = [E \to E] + [E \to \Delta_1]$$

$$[\Delta_1 \to E + \Delta_1] = [\Delta_1 \to E] + [\Delta_1 \to \Delta_1]$$

$$[E \to E] + [E \to \Delta_1] = \{f + g : f \in [E \to E],\ g \in [E \to \Delta_1]\}$$

$f + g$ being defined as the canonical extension of f and g, if $\{Dom(f)\ ,\ Dom(g)\}$ is a partition of E. (Here, as has been said earlier, we use partial functions.) The space $[\Delta_1 \to E] + [\Delta_1 \to \Delta]$ is defined in a similar way.

This defines the partial sequence of spaces:

$$A_0 = E$$

$$A_1 = E + \Delta_1 = E + [A_0 \to A_0] \qquad (1)$$

$$A_2 = E_2 + \Delta_2 = E + [A_1 \to A_1]$$

As may be seen from the construction, Δ_2 has a c-domain structure by using Lemma 12. We also have the following property: if $\{f_p\}_{p\in\mathbb{N}}$ is a computable sequence of Δ_1 and $G : \Delta_1 \to E + \Delta_1$ is a computable function, then $\{G(f_p)\}_{p\in\mathbb{N}}$ is a computable sequence, by an argument similar to the one used for Lemma 9.

The finite sequence of (1) can be extended by:

$$A_0 = E_s$$

$$A_{n+1} = E_s + \Delta_{n+1} = E_s + [A_n \to A_n]$$

where for any $n \in \mathbb{N}$, $\Delta_{n+1} = [A_n \to A_n]$ has a c-domain structure by construction, and A_{n+1} is supplied with a bundle structure obtained by gluing together the E-structure and the Δ_{n+1} structure as follows:

$$\begin{tikzcd} & T \arrow[dl, dash] \arrow[dr, dash] & \\ E_s & & \Delta_{n+1} \end{tikzcd} \quad = A_{n+1}$$

Every A_n is a c-domain. Now the threshold function

$$[x,y] : z \to \underline{\text{if}} \;\; z \le x \;\; \underline{\text{then}} \;\; y \;\; \underline{\text{else}} \;\; T$$

is computable iff x is <u>inf-computable</u> and y is sup-computable. If $x \in E$, this can be checked at once. If $x \in \Delta_n \cup \{T\}$, we make x inf-computable, since $\Delta_n \cup \{T\}$ is enumerable, by setting the lower bundle as follows:

$$\forall \;\; x \in \Delta_n \cup \{T\} \quad s_{low}(x) = \{x\}$$

which implies that every $x \in \Delta_n \cup \{T\}$ is inf-computable. Because of its triviality this lower bundle structure will be somewhat overshadowed in the sequel.

We now make the above sequence a diagram, by defining, following Wadsworth 71,

$\forall\ n \in \mathbb{N}$

$i_0 : A_0 \to A_1 \ , \ x \to x$

$i_n : A_n \to A_{n+1} \ , \ x \to x \quad \text{if} \quad x \in E \ \cup \{T\}$

$\qquad\qquad i_{n-1} \ 0 \ x \ 0 \ j_{n-1} \quad \text{if} \quad x \in \Delta_n$

$j_0 : A_1 \to A_0 \ , \ y \to y \quad \text{if} \quad y \in E \ \cup \{T\}$

$\qquad\qquad \mathbb{N} \quad \text{if} \quad y \in \Delta_1$

$j_n : A_{n+1} \to A_n \ , \ y \to y \quad \text{if} \quad y \in E$

$\qquad\qquad j_{n-1} \ 0 \ y \ 0 \ i_{n-1} \quad \text{if} \quad y \in \Delta_{n+1}$

This diagram will be called <u>Wadsworth scheme</u>. Notice that, for every n ,

$j_n \ 0 \ i_n = id_{A_n}$

$i_n \ 0 \ j_n \geq id_{A_n}$

i_n and j_n are distributive with respect to $\sqcap$ and $\sqcup$

An element $x = (x_n)_{n\in\mathbb{N}} \in \prod_{n\in\mathbb{N}} A_n$ belonging to the cartesian product of the A_n's will be called <u>computable</u> if and only if there exists $\psi : \mathbb{N}^2 \to \mathbb{N}$ recursive such that for every $n \in \mathbb{N}$ $\psi(n,.)$ is an enumeration of (the indices of) the spectrum of x_n . If $x_n \in E$, then we consider the spectrum of x_n for the upper bundle. Computable sequences of elements of $\prod_{n\in\mathbb{N}} A_n$ are defined in the usual way. Notice that $\prod_{n\in\mathbb{N}} A_n$ has a c-domain structure. Its kernel is the cartesian product of the kernels. Consider now the projective limit

$$A_\infty = \{(x_n)_{n\in\mathbb{N}} \in \prod_n A_n \mid x_n = j_n(x_{n+1})\}$$

Then for any $\{(x_n)_{n\in\mathbb{N}} \in A_\infty$, we have:

- either x_n belongs to the E-part of A_∞ and $x_n = x_{n+1}$
- or x_n belongs to the functional part of A_∞ and $x_n = j_{n-1} \circ x_{n+1} \circ i_{n-1}$

Therefore we have the equality of sets:

$$A_\infty = \{(x_n)_{n\in\mathbb{N}} \in \prod_n A_n \mid x_n = x_0 \in E\} +$$

$$\{(x_n)_{n\in\mathbb{N}} \in \prod_n A_n \mid x_0 = \mathbb{N}\,,\ x_n = j_n(x_{n+1})\}$$

Thus $A_\infty = E + \Delta_\infty$, where Δ_∞ is the functional part of A_∞.

Now both notions of computability and projective limit are put together in order to define the set of computable projective sequences:

$$A_\omega = \{(x_n)_{n\in\mathbb{N}} \in A_\infty \mid (x_n)_{n\in\mathbb{N}} \text{ is computable}\}$$

One easily sees that $A_\omega = E_s + \Delta_\omega$ The closure of A_ω for decreasing computable sequences will be called an 'algorithm space'. Here again, we disregard the power bundle structure of A_ω, as far as functional elements are concerned, because of the triviality of this bundle structure. Thus computability here means computability for the upper bundle structure.

Definition:

The algorithm space A is the smallest set containing A_ω and such that every decreasing computable sequence in Δ_ω has a greater lower bound. Elements of A are called algorithms □

In other words, A is the smallest c-domain containing A_ω . Its kernel is canonically isomorphic to the union of the kernels of the A_n's :

$$N(A) = \bigcup_n N(A_n)$$

Lemma 13: Let $i_{n\infty} : A_n \to A$ be the canonical injection of A_n into A , and $j_{n\infty} : A \to A_n$ the canonical projection of A onto A_n . Then both $i_{n\infty}$ and $j_{\infty n}$ are computable.

Proof:

The regularity comes from the distributivity of functions i and j .

(i) injection $i_{n\infty}$: this set must be r.e. $\{(p,q) : i_{n\infty}(ap) \leq b_q\} =$ (if $b_q = (b_q^n)_{n\in\mathbb{N}}$)

$\{(p,q) : a_p \leq b_q^n \text{ in } A_n \ ; \ b_q = i_{n\infty}(b_q^n)\}$

We know that in A_n the relation $a_p \leq b_q^n$ is recursive, and the set of rational elements of A_n is enumerable (this set contains all the rationals of E_s for the upper bundle, and all the rationals of Δ_n), which completes the proof.

(ii) Similarly: this set must be r.e.:

$$\{(p,q) : j_{\infty n}(a_p) \leq b_q\} = (\text{if } a_p = (a_p^n)_{n\in\mathbb{N}})$$
$$= \{(p,q) : a_p^n \leq b_q\} = \{(p,q) : a_p^n \leq b_q^n\}$$

which is r.e. since the relation $a_p^n \leq b_q^n$ is recursive in A_n by the same argument as for Lemma 6. Whence the lemma. □

Lemma 14: Let $f : A \to A$ be a computable function. Then the sequence of $\prod_{n\in\mathbb{N}} A_n$ defined by

$$]f[_0 = \mathbb{N}$$
$$]f[_{n+1} = \lambda y \in A_n.\ (f(y))_n = \lambda y \in A_{n.j_{\infty n}}\ (f(y))$$ is projective and computable, and thus an element of A_ω □

Proof:

(i) the sequence $(]f[_p)_{p\in\mathbb{N}}$ is projective since:

$$j_n(]f[_{n+1}) = j_{n-1}\ 0\ ]f[_{n+1}\ 0\ i_{n-1} =$$
$$(j_{n-1}\ 0\ j_{\infty n})\ 0\ f\ 0\ (i_{n\infty}\ 0\ i_{n-1}) =$$
$$j_{\infty,n-1}\ 0\ f\ 0\ i_{n-1,\infty} =]f[_n$$

Thus $(]f[_p)_{p\in\mathbb{N}} \in A_\infty$

(ii) the sequence $(]f[_p)_{p\in\mathbb{N}}$ is computable: spectrum $(]f[_0) = \{\mathbb{N}\}$

spectrum $(]f[_{n+1}) = j_{\infty_n}$ 0 spectrum (f) 0 $i_{n\infty}$

Let ψ be a function defined as follows:

1. $\psi(0,.) = \lambda q.$ index of $\mathbb{N}$ in the enumerating of the kernel of A_ω

2. $\psi(p,q) = j_{\infty,p-1} \circ s(q) \circ i_{p-1,\infty}$ where $s : q \to s(q)$ is a recursive enumeration of the spectrum of f.

Thus $\psi : \mathbb{N}^2 \to \mathbb{N}$ seen as a function into the indices is recursive and $\psi(p,.)$ is an enumeration of the spectrum of $]f[_p$. Thus $(]f[_p)_{p\in\mathbb{N}}$ is a computable element of $\prod_n A_n$. Therefore

$$(]f[_p)_{p\in\mathbb{N}} \in A_\omega \quad . \qquad \square$$

Lemma 15: Any $x \in \Delta_\omega$ defines a computable function

$$[x] : A \to A$$

$$y \to \prod_n x_{n+1}(y_n) \quad \text{if } y \in \Delta_\omega \text{ or } y = a_p \in E$$

$$\bigsqcup\{[x](a_p) : a_p \subseteq y \in E\} \quad \text{otherwise.}$$

Proof:

Regularity here means regularity for the c-domain structure, and is obvious. The spectrum of $[x]$ is $\bigcup_n s(x_n)$, which is r.e. since the sequence $x = (x_n)_{n\in\mathbb{N}}$ is computable.

More importantly, we have a representation property, which justifies the use of the c-domain notion:

Lemma 16: Let $f : A \to A$ be a computable function. Then for any z in the kernel of Δ_ω or such that $z = a_p \in E$, the following are equivalent:

(i) $f(z) = []f[](z)$

(ii) if $z = (z_n)_{n\in\mathbb{N}} = \prod_n z_n$,

$$f(z) = f(\prod_n z_n) = \prod_n f(z_n)$$

Proof:

It suffices to see that

$$[\,]f[\,](z) = \prod_n \,]f[_{n+1}\,(z_n) =$$

$$\prod_n (\lambda y \in A_n \,.\, (f(y))_n)(z_n) = \prod_n (f(z_n))_n$$

$$= \prod_{n,k} (f(z_n))_k = \quad \text{(property of } \sqcap)$$

$$= \prod_n \prod_k (f(z_n))_k = \prod_n f(z_n)$$

Thus $[\,]f[\,](z) = f(z)$ is equivalent to

$$\prod_n f(z_n) = f(z)$$

Whence the lemma. □

Lemma 16 characterizes completely the representation of computable functions as elements of Δ_ω, because of the definition of the operation $[.] : \Delta_\omega \to [A \to A]$ and since A is a c-domain.

Thus, basically, we see that our functions must satisfy the following "continuity" property:

$\forall\ \{x_k\}_{k\in\mathbb{N}}$ decreasing computable sequence,

$$f(\prod_k x_k) = \prod_k f(x_k)$$

which is analogous to Scott-continuity, but for two modifications: the inversion of the order and the introduction of computability. In the present setting, the above property is what we need when computing with procedures.

<u>Theorem (existence of algorithm collections)</u>:

There exists an enumerable set $\mathcal{A}$, called algorithm space, such that

(i) $\mathcal{A} = E_s + \mathcal{F}$, and every element is computable.

(ii) Every decreasing computable sequence has a greatest lower bound in $\mathcal{A}$

(iii) if $x,y \in \mathcal{A}$ then any computable threshold function

$$[x,y] : z \to \underline{\text{if}} \;\; z \leq x \;\; \underline{\text{then}} \;\; y \;\; \underline{\text{else}} \;\; T$$

is represented as an element of $\mathcal{F}$ which is

$$(\lambda z \in A_n \,.\, \bigsqcup_p \;\; \underline{\text{if}} \;\; z_p \leq x_p \;\; \underline{\text{then}} \;\; y_n \;\; \underline{\text{else}} \;\; T_n)_{n+1 \in \mathbb{N}}$$

<u>Proof</u>:

Results from the preceding lemmae.

The enumerability of $\mathcal{A}$ comes from the fact that the set of recursive functions $\psi : \mathbb{N}^2 \to \mathbb{N}$ is enumerable, and all our objects are computable. $\square$

References

[1] J.D. Monk: Mathematical Logic, Springer (1976).

[2] M.A. Nait Abdallah: Types and approximating calculi in programming languages semantics, 3rd Workshop on Continuous Lattices, Riverside, California, (1979).

[3] M.A. Nait Abdallah: Faisceaux et Sémantique des programmes, Thèse d'Etat, Paris (1980).

[4] L. Nolin: Algorithmes universels, RAIRO rouge 2 (1974), pp. 5 - 18.

[5] L. Nolin, F. Le Berre: Les espaces informatiques, leurs existence, leurs rapports avec la logique combinatoire et les λ-calculs, Rapport LITP, # 81 - 21, Paris (1981).

[6] L. Nolin, F. Le Berre: L'existence d'espaces informatiques, C.R.A.S. t. 292, série I, pp. 499 - 502.

[7] D. Scott: Continuous lattices, Springer LNM 274 (1972), pp. 97 - 136.

[8] D. Scott: Data types as lattices, SIAM J. Comp. 5 (1976), pp. 522 - 587.

[9] C. Wadsworth: Semantics and pragmatics of the λ-calculus, Ph.D. thesis, Oxford (1971).