RAIRO
INFORMATIQUE THÉORIQUE

M. A. NAIT ABDALLAH

Data types as algorithms

RAIRO – Informatique théorique, tome 18, nº 1 (1984), p. 3-22.
<http://www.numdam.org/item?id=ITA_1984__18_1_3_0>

R.A.I.R.O. Informatique théorique/Theoretical Informatics
(vol. 18, n° 1, 1984, p. 3 à 22)

DATA TYPES AS ALGORITHMS (*)

by M. A. NAIT ABDALLAH (1)

Communicated by J.-F. PERROT

Abstract. — *This paper presents a simplified construction of an algorithm space. The intuitive notion of an algorithm collection goes back to Nolin 1974. Our goal is here to formalize and 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 space where every object is computable.*

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

INTRODUCTION

This paper presents a simplified construction of an algorithm space [6, 3]. The idea and the terminology (*algorithm*) go back to Nolin 1974 [6], who wanted a semantics for a programming language with type declarations. (It must be pointed out here that Nolin's "algorithms" are not algorithms in the usual sense, i. e. a step-by-step description of a computation. They are in fact much closer to generalized types, or algebraic sorts.)

However, a proof of the conceptual reasonableness of the idea of an algorithm space, or in other words, a mathematical proof of the existence of such a space, was missing. Such a proof was first given in a restricted setting it [2]. It used bundle theory, which, as shown in [3], underlies a significant part of programming language semantics model theory.

(*) Received in April 1982, revised in February 1983.
Research supported by the University of Waterloo under grant 126-7124.

(1) Department of Computer Science, University of Waterloo, Waterloo, Ontario, Canada, N2L 3G1, and Laboratoire d'Informatique théorique et Programmation, Paris.

0399-0540/84/01 3 22/$ 4.20/ © AFCET-Bordas

In [8], which is a shorter version of [7], Nolin and Le Berre present a simplified version of this proof, where they use "solely elementary set theory properties" [7]. Technically, this amounts to abandon the upper bundle structure used in [3]. Unfortunately, we discovered [4, 5] an error in a crucial step of their 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 falsehood of their theorem follows from a careful analysis of the representation used for threshold functions:

$$f_{xy}: \quad z \to \textit{if } z \leqq x \textit{ then } y \textit{ else } T,$$

which turns out to be identical to the representation used in [2, 3].

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 *enumerable* algorithm collection. The enrichment is the introduction of *computability* notions: all our algorithms will be computable in some sense. And this makes the whole algorithm space itself effectively presentable.

1. BUNDLE STRUCTURES OVER $\mathscr{P}(\mathbb{N})$

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

$$\forall x \in E, \quad x = \bigcup_{m \in x} \{m\} = \bigcup_{a_n \subseteq x} a_n;$$

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

$$a: \quad n \to \{n-1\} \qquad \text{if} \quad n \neq 0;$$
$$a_0 = \emptyset \qquad \text{where} \quad \emptyset \text{ is the empty set.}$$

This gives an elementary monic ordered bundle structure over E [3].

The *spectra* are defined by:

$$s: \quad E \to \mathscr{P}(E);$$
$$x \to \{a_n : a_n \subseteqq x\}.$$

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:

$$\begin{aligned} b :\ & \mathbb{N} \to E, \\ & m \to b_m = \complement \{ k_0, k_1, \ldots, k_{p-1} \}, \end{aligned}$$

where:

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

(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:

$$b_m \supseteq \bigcap_i x_i \quad \Rightarrow \quad \exists\, i\; b_m \supseteq x_i.$$

The *spectrum* function:

$$\begin{aligned} s :\ & E \to \mathcal{P}(E); \\ & x \to s(x) = \{ b_m : b_m \supseteq x \}, \end{aligned}$$

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

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

This structure will be called the *upper bundle* 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.

2. COMPUTABILITY IN $E=\mathscr{P}(\mathbb{N})$

2.1. Computable elements of $\mathscr{P}(\mathbb{N})$

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 $\{n : a_n \subseteqq x\}$ 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 $\{m : x \subseteqq b_m\}$ 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:

$$\{m : x \subseteqq b_m\} = \{m : x \subseteqq \complement \{k_0, k_1, \ldots, k_{p-1}\}, m = \sum_{i<p} 2^{k_i}, k_i \text{ all different}\},$$

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

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

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

For:

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

$\mathscr{P}_{\text{fin}}(y)$ r. e. $\Rightarrow y$ r. e.: we take a recursive enumeration of $\mathscr{P}_{\text{fin}}(y)$ 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. □

DEFINITION: $x \in E$ is *computable* if and only if x is inf-computable and x is sup-computable. □

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

LEMMA 2: $\forall x \in E = \mathscr{P}(\mathbb{N})$:

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

(ii) *x is sup-computable* $\Leftrightarrow \complement x$ *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 $\emptyset$ 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. $\square$

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:

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

is primitive recursive and bijective. Its inverse:

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

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. $\square$

2.3. *C*-domains

Let X be a poset. An element $u \in X$ is *algebraic* iff for any decreasing (computable) chain $\{x_k\}_{k \in \mathbb{N}}$ which has a glb $\prod_k x_k$ in X:

$$u \geqq \prod_k x_k \quad \Rightarrow \quad \exists k \quad u \geqq x_k.$$

As an example, in the poset E ordered by inclusion, every b_m is algebraic. More generally, in any c. p. o. for the opposite order which has an enumerable set of compact elements, every compact element is algebraic.

DEFINITION: *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 algebraic elements is given, and this enumeration verifies:

$$\forall x \in X, \quad x = \Pi \{ a_n : a_n \geqq x \}$$

and $\{ n : a_n \geqq x \}$ is recursively enumerable. □

Examples of c-domains included in E are

$$X = \{ \{ n \} : n \in \mathbb{N} \} \cup \{ \mathbb{N} \}$$

and

$$X = Y \cup \{ \mathbb{N} \},$$

where $Y \subseteqq \mathscr{P}_{\text{fin}}(\mathbb{N})$ 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:

LEMMA 4: *The set E_s of sup-computable elements of E is the unique c-domain such that:*

(i) *It is included in $E = \mathscr{P}(\mathbb{N})$ as a poset.*

(ii) *It has the enumeration b and contains every recursive subset $x \subseteqq \mathbb{N}$ as an element.* □

Proof:

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

(ii) Let $X \subset \mathscr{P}(\mathbb{N})$ 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 $E_s \subseteqq X$. If $x \notin E_s$, then $\{ m : b_m \supseteqq x \}$ 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 $E = \mathscr{P}(\mathbb{N})$.

3. 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 $\{ (m, n) : f(a_n) \subseteqq b_m \}$ is recursively enumerable in the indices m, n.

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

$$f\leqq g \quad\Leftrightarrow\quad \forall x\; f(x)\subseteq g(x);$$

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

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

$$s(y)=\bigsqcup y=\max y;$$
$$r(x)=\downarrow x=\{z: z\leqq x\}.$$

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

$$A=\bigcap_{i\in I} FX_i Y_i;$$

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

LEMMA 5: *The threshold function:*

$$[x, y]=\lambda t\in E \quad \textit{if} \quad t\subseteq x \textit{ then } y \textit{ else } \mathbb{N};$$

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

Proof:

$$\{(m, n) : [x, y](a_n) \subseteqq b_m\}$$
$$= \{(m, n) : \textit{if } a_n \subseteqq x \textit{ then } y \textit{ else } \mathbb{N} \subseteqq b_m\}$$
$$= \{(m, n) : a_n \subseteqq x \text{ and } y \subseteqq b_m\} \cup \{(m, n) : a_n \nsubseteq x,\ \mathbb{N} = b_m\}$$
$$= \{m : y \subseteqq b_m\} \times \{n : a_n \subseteqq x\} \cup \{n : a_n \nsubseteq x\}.$$

Hence $\{(m, n) : [x, y](a_n) \subseteqq b_m\}$ is r. e. iff $\{m : y \subseteqq b_m\}$ is r. e. and $\{n : a_n \subseteqq x\}$ is recursive [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:

$$\text{graph}^*(f) = \{(m, n) \mid f(a_n) \subseteqq b_m\} = \{(m, n) \mid f(\{n-1\}) \subseteqq b_m\}.$$

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

$$\text{graph}(f) = \{(m, n \mid \{m\} \subseteqq f(e_n)\}, \qquad e_n = \complement b_n.$$

Any element $u \in \mathbf{P}\omega$ operates as a continuous function by:

$$\text{fun}(u)(x) = \{m \mid \exists\, e_n \subseteqq x : (n, m) \in u\}.$$

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

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

One easily verifies that:

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

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

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

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

$$\{(p, q) : \bigcap \{b_m : (p+1, m) \in u\} \subseteqq b_q\}$$
$$= \{(p, q) : \bigcup \{e_m : (p+1, m) \in u\} \supseteqq e_q\}$$
$$= \{(p, q) : \exists m \ \ (p+1, m) \in u \wedge e_q \subseteqq e_m\};$$

is recursively enumerable.

The relation $e_q \subseteqq e_m$ is recursive, therefore the set is recursively enumerable; i. e. every r. e. set $u \in E$ defines a computable function $\text{fun}^*(u) : E \to E$.

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

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

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

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

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

or:

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

The relation:

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

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:

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

with:

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

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

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 Π and every β_m verifies the following algebraicity property:

$$\forall \{ x_i \}_{i \in I}, \qquad \beta_m \geqq \prod_i x_i \quad \Rightarrow \quad \exists i \, \beta_m \geqq x_i.$$

Moreover, for the computability aspect, we have.

LEMMA 6: *The relations* $\beta_m \leqq \beta_n$, $[a_n, b_m] \geqq \beta_p$, $\beta_m = \beta_n \sqcap \beta_p$ *are all recursive in the indices* m, n, p.

Proof: We consider the proof for $\beta_m \leqq \beta_n$. First notice that:

$$[a, b] \leqq [c, d] \quad \Leftrightarrow \quad (c \subseteqq a \wedge b \subseteqq d) \qquad \text{or} \qquad d = \mathbb{N}.$$

Thus:

$$[a_n, b_m] \leqq [a_{n'}, b_{m'}] \quad \Leftrightarrow \quad a_{n'} \subseteqq a_n \,(\text{recursive}) \wedge b_m \subseteqq b_{m'} \,(\text{recursive});$$

or:

$$b_{m'} = \mathbb{N} \,(\text{recursive}).$$

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

$$\beta_m = e_{k_0} \sqcap \ldots \sqcap e_{k_{p-1}}, \qquad m = \sum_{i<p} 2^{k_i}; \qquad e_{k_i} = [a_{U^1_2(v(k_i))}, b_{U^2_2(v(k_i))}];$$

$$\beta_n = e_{l_0} \sqcap \ldots \sqcap e_{l_{q-1}}, \qquad n = \sum_{j<q} 2^{l_j}; \qquad e_{l_j} = [a_{U^1_2(v(l_j))}, b_{U^2_2(v(l_j))}].$$

Thus:

$$\beta_m \leqq \beta_n \quad \Leftrightarrow \quad e_{k_0} \sqcap \ldots \sqcap e_{k_{p-1}} \leqq e_{l_0} \sqcap \ldots \sqcap e_{l_{q-1}}$$

$$\Leftrightarrow \quad (\text{simplifying the notations})$$

$$[a_{k_0}, b_{k_0}] \sqcap \ldots \sqcap [a_{k_{p-1}}, b_{k_{p-1}}] \leqq [a_{l_0}, b_{l_0}] \sqcap \ldots \sqcap [a_{l_{q-1}}, b_{l_{q-1}}]$$

$$\Leftrightarrow \quad \bigcap \{ b_{k_i} : a_{l_0} \leqq a_{k_i} \} \subseteqq \bigcap \{ b_{l_j} : a_{l_0} \leqq a_{l_j} \}$$

$$\textit{and} \quad \bigcap \{ b_{k_i} : a_{l_1} \leqq a_{k_i} \} \subseteqq \bigcap \{ b_{l_j} : a_{l_1} \leqq a_{l_j} \};$$

$$\vdots$$

$$\textit{and} \quad \bigcap \{ b_{k_i} : a_{l_{q-1}} \leqq a_{k_i} \} \subseteqq \bigcap \{ b_{l_j} : a_{l_{q-1}} \leqq a_{l_j} \}.$$

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

We have a similar argument for the other relations. Whence the lemma. □

LEMMA 7: *The set of computable functions* $f : E \to E$ *which have the following* simplicity *property:*

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

is closed under composition.

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

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

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

The spectrum of $g \circ f$, which is a monotone function, is given by:

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

$$x \subseteqq a_n \Rightarrow f(x) \subseteqq b_m \qquad \text{and} \qquad g(b_m) = \bigcup \{g(a_p) : a_p \subseteqq b_m\}.$$

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

$$a_n \xrightarrow{f} b_m = \begin{array}{l} a_{p_0} \xrightarrow{g} b_{q_0}, \\ a_{p_1} \to b_{q_1}, \\ \quad \vdots \\ a_{p_i} \to b_{q_i}, \\ \quad \vdots \end{array}$$

This gives $[a_n, \bigcup_i b_{q_i}]$ as approximating $g \circ f$, i. e., $(g \circ f)(a_n) \subset \bigcup_i b_{q_i}$.

The set $b_m = \bigcup_i a_{p_i}$ is recursive, thus the sequence $\{b_{q_i}\}$ is recursively enumerable since $\{(p, q) : [a_p, b_q] \geqq g\}$ is recursively enumerable.

Therefore $\bigcup_i b_{q_i}$ is r. e., i. e., *inf-computable.* 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 $g \circ f$. Buf if we impose that $g \circ f$ be simple, i. e., $f(a_n) = a_m$ for some $m \in \mathbb{N}$, then, since:

$$g = \Pi \{[a_p, b_q] : g(a_p) \subseteqq b_q\},$$

then:

$$(g \circ f)(a_n) = g(a_m) = \bigcap_q \{b_q : a_m \subseteqq a_p\}$$

which is a sup-computable element of E, whence an r. e. spectrum for $g \circ f$:

$$s(g \circ f) = \{ [a_n, b_s] : f(a_n) = a_p, g(a_p) \subseteq b_s \}. \quad \square$$

LEMMA 8: *There is a canonical bijection between the simple computable functions $f: E \to E$ and the recursive functions from $\mathbb{N}$ to $\mathbb{N}$.*

Proof: Obvious. $\square$

LEMMA 9: *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, $f(x_p) = \bigcap \{b_m : x_p \subseteq a_n, f(a_n) \subseteq b_m\}$ 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. $\square$

3.2. Computable sequences of functions

DEFINITION: A sequence $\{f_p\}_{p \in \mathbb{N}}$ of regular functions from E to E 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. $\square$

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

LEMMA 10: *If $\{f_p\}_{p \in \mathbb{N}}$ is a computable sequence of functions, then its greatest lower bound $\prod_p f_p$ is a computable function.*

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

$$(\prod_p f_p)(x) = \bigcup \{(\prod_p f_p)(a_n) : a_n \subseteq x\}.$$

Now we just have to glue the spectra together:

$$s(\prod_p f_p)=\bigcup_p s(f_p).$$

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

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

LEMMA 11: *Given the enumeration* β *of the 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.*

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 $[E\to E]=X$. □

We obtain similar results by replacing E by E_s, and there is a canonical bijection between $[E\to E]$ and $[E_s\to E_s]$.

4. WADSWORTH SCHEME

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

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

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

$[\Delta_1\to\Delta_1]=\{f : f(\prod_k x_k)=\prod_k f(x_k)$ for every decreasing computable sequence(x_k) and $\{(m, n) : f(\beta_n)\leqq\beta_m\}$ 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, all have 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:

$$\Delta_1 = [E \to E],$$
$$A_1 = E + \Delta_1;$$
$$A_2 = E + \Delta_2 = E + [E + \Delta_1 \to E + \Delta_1];$$

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 $\{ \mathrm{Dom}\ (f), \mathrm{Dom}\ (g) \}$ 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:

$$\left.\begin{aligned} A_0 &= E_s; \\ A_1 &= E_s + \Delta_1 = E_s + [A_0 \to A_0]; \\ A_2 &= E_s + \Delta_2 = E_s + [A_1 \to A_1]. \end{aligned}\right\} \qquad (1)$$

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 if $G : \Delta_1 \to E + \Delta_1$ 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_s structure and the Δ_{n+1} structure as follows:

$$\begin{array}{ccc} & T & \\ / & & \backslash \\ E_s & & \Delta_{n+1} \end{array} = A_{n+1},$$

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

$$[x, y] : \quad z \to \textit{if } z \leqq x \textit{ then } y \textit{ else } T$$

is computable iff x is finitely *computable* and y is sup-computable. If $x \in E$, this can be checked at once (finite computability = recursiveness). If $x \in \Delta_n \cup \{T\}$, we use the definition.

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

$$\begin{array}{lll} \forall n \in \mathbb{N}; & & \\ i_0 : & A_0 \to A_1, & x \to x; \\ i_n : & A_n \to A_{n+1}, & x \to x \quad \text{if } x \in E_s \cup \{T\}; \\ & & i_{n-1} \circ x \circ j_{n-1} \quad \text{if } x \in \Delta_n; \\ j_0 : & A_1 \to A_0, & y \to y \quad \text{if } y \in E_s \cup \{T\}; \\ & & \mathbb{N} \quad \text{if } y \in \Delta_1; \\ j_n : & A_{n+1} \to A_n, & y \to y \quad \text{if } y \in E_s; \\ & & j_{n-1} \circ y \circ i_{n-1} \quad \text{if } y \in \Delta_{n+1}. \end{array}$$

This diagram will be called *Wadsworth scheme*. Notice that, for every n:

$$j_n \circ i_n = \mathrm{id}_{A_n};$$

$$i_n \circ j_n \geqq \mathrm{id}_{A_n};$$

i_n and j_n are distributive with respect to Π and $\cup$.

An element $x = (x_n)_{n \in \mathbb{N}} \in \bigtimes_{n \in \mathbb{N}} A_n$ belonging to the cartesian product of the A_n's will be called *computable* 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 $\bigtimes_{n \in \mathbb{N}} A_n$ are defined in the usual way. Notice that $\bigtimes_{n \in \mathbb{N}} A_n$ has a c-domain structure. Its kernel is the cartesian product of the kernels. Consider now the projective limit:

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

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:

$$A_\infty = \{ (x_n)_{n \in \mathbb{N}} \in \bigtimes_n A_n \mid x_n = x_0 \in E_s \} + \{ (x_n)_{n \in \mathbb{N}} \in \bigtimes_n A_n \mid x_0 = \mathbb{N},\ x_n = j_n(x_{n+1}) \}.$$

Thus $A_\infty = E_s + \Delta_\infty$, 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 lower 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* $\mathscr{A}$ is the smallest set containing A_ω and such that every decreasing computable sequence in Δ_ω has a greater lower bound. Elements of $\mathscr{A}$ are called *algorithms.* □

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

$$N(\mathscr{A}) = \bigcup_n N(A_n).$$

LEMMA 13: *Let $i_{n\infty} : A_n \to \mathscr{A}$ be the canonical injection of A_n into $\mathscr{A}$, and $j_{n\infty} : \mathscr{A} \to A_n$ the canonical projection of $\mathscr{A}$ 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.:

$$\{(p, q) : i_{n\infty}(a_p) \leqq b_q\} = (\text{if } b_q = (b_q^n)_{n \in \mathbb{N}}) \{(p, q) : a_p \leqq b_q^n \text{ in } A_n;\ b_q = i_{n\infty}(b_q^n)\}.$$

We know that in A_n the relation $a_p \leqq b_q^n$ 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) Projection $j_{n\infty}$: Similarly this set must be r. e.:

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

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

LEMMA 14: *Let $f : \mathcal{A} \to \mathcal{A}$ be a computable function. Then the sequence of* $\displaystyle\mathop{\times}_{n \in \mathbb{N}} A_n$ *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:

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

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 $(]f[_{n+1}) = j_{\infty n} \circ$ *spectrum* $(f) \circ i_{n\infty}$.

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 $\displaystyle\mathop{\times}_{n} A_n$. Therefore:

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

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

$$[x] : \mathscr{A} \to \mathscr{A};$$
$$y \to \prod_n x_{n+1}(y_n) \qquad \textit{if} \quad y \in \Delta_\omega \quad \textit{or} \quad y = a_p \in E;$$
$$\bigcup \{ [x](a_p) : a_p \subseteqq y \in E \} \ \textit{otherwise}.$$

Proof: Regularity here means regularity for the c-domain structure, and is obvious. The spectrum of $[x]$ is the cartesian product $\prod_n s(x_n)$, which is r. e. since 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 $f : \mathscr{A} \to \mathscr{A}$ 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* $z = (z_n)_{n \in \mathbb{N}} = \prod_n z_n \qquad f(z) = f(\prod_n z_n) = \prod_n f(z_n)$.

Proof: It suffices to see that:

$$[\quad]f[\quad](z) = \prod_n]f[_{n+1}(z_n) = \prod_n (\lambda y \in A_n . (f(y))_n)(z_n) = \prod_n (f(z_n))_n$$
$$= \prod_{n,k} (f(z_n))_k = (\text{property of } \Pi) = \prod_n \prod_k (f(z_n))_k = \prod_n f(z_n).$$

Thus $[\quad]f[\quad](z) = f(z)$ 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 $[.] : \Delta_\omega \to [\mathscr{A} \to \mathscr{A}]$ and since $\mathscr{A}$ is a c-domain.

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

$$\forall \{ x_k \}_{k \in \mathbb{N}} \text{ decreasing computable sequence } \ f(\prod_k x_k) = \prod_k f(x_k);$$

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.

THEOREM (existence of enumerable algorithm domains): *There exists an enumerable set $\mathscr{A}$, called algorithm space, such that:*

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

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

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

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

is represented as an element of $\mathscr{F}$ *which is:*

$$(\lambda z \in A_n . \coprod_p \text{ if } z_p \leqq x_p \text{ then } y_n \text{ else } T_n)_{n+1 \in \mathbb{N}}.$$

Proof: Results from the preceding lemmae.

The enumerability of $\mathscr{A}$ 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. □

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'État, Paris, 1980.
4. M. A. NAIT ABDALLAH, *Sur les espaces informatiques de Nolin et Le Berre*, C.R.A.S., t. 295, série I, pp. 711-714.
5. M. A. NAIT ABDALLAH, *The necessity of double bundle structure in sort theory*, University of Waterloo, Report CS-82-36.
6. L. NOLIN, *Algorithmes universels*, R.A.I.R.O. rouge, No. 2, 1974, pp. 5-18.
7. L. NOLIN and F. LE BERRE, *Les espaces informatiques, leur existence, leurs rapports avec la logique combinatoire et les λ-calculs*, Revue Technique Thomson/CSF, Vol. 13, No. 3, septembre 1981, pp. 599-633.
8. L. NOLIN and F. LE BERRE, *L'existence d'espaces informatiques*, C.R.A.S., t. 292, série I, pp. 499-502.
9. D. SCOTT, *Continuous lattices*, Springer LNM 274, 1972, pp. 97-136.
10. D. SCOTT, *Data types as lattices*, S.I.A.M. J. Comp., Vol. 5, 1976, pp. 522-587
11. C. WADSWORTH, *Semantics and pragmatics of the λ-calculus*, Ph. D. Thesis, Oxford, 1971.