Effectively Given Continuous Domain: A Computable Interval Analysis

Benjamín René Callejas Bedregal
Departamento de Informática e Matemática Aplicada - DIMAp
Universidade Federal do Rio Grande do Norte - UFRN
Campus Universitário - Lagoa Nova,
50.072-970, Natal-RN, Brasil
bedregal@dimap.ufrn.br

Benedito Melo Acióly
Universidade Estadual do Sudoeste da Bahia - UESB
Estrada do Bem Querer, Km 4
45.000-000, Vitória da Conquista-BA, Brasil
bma@uesb.br

Abstract

The notion of effectively given continuous domain is used with the objetive of characterizing the computable functions (as opposed to the merely continuous ones) and the computable elements of types represented by continuous domains. We show that computability on arbitrary effectively given continuous domain strongly depends upon the Church-Turing computability on a countable base.

In this work the notion of effectively given continuous domains is used in order to show how to define a notion of computability on the real intervals and from it we derive a natural computability notion on the real numbers. In order to show how the computable functions on the continuum are really computed we introduce an extended Turing machine,

KeyWords: Computability on the Continuum, Domain Theory, Interval Analysis, Turing machines.

1 Introduction

The notion of effective procedure is intuitive and not limited to countable sets, while the notion of computability on Turing machines and partial recursive functions limits the computable functions to countable ones. The theory of computability on countable sets is called *type 1 computability* by Weirhauch [Wei95] whereas the computability on sets with the cardinality of continuum is called by him *type 2 computability*.

The computability on sets with cardinality of the continuum has been more directioned to computability on the real numbers. Very different approaches have been used to investigate, from a constructive standpoint, main concepts arising in analysis such as real numbers, limits, derivatives and measure. An important difference among these approaches lies in the way real numbers are represented [Gia93]. There are basically four definitions of computable real numbers which was shown equivalent [PR88]. For each of these the definition of computable real number is an effectivization of the usual real numbers representation (Cauchy sequences on rational numbers, Dedekind cuts, nested intervals and infinite decimal fractions). The Cauchy representation notion is the most accepted of real numbers representations [PR88, Wei95]. The notion of computable real functions was give by Grzegorczyk's in [Grz57] where he uses the notion of computable operators on the natural numbers sequence set. This notion was investigated and generalized by [PR88, Wei95]. In this approach (called here as Cauchy computability) an approximation of the output with arbitrary precision is computed from a suitable approximation of the input [Bra95].

In ordinary practice the computation on real numbers is performed by approximating a real number by float point numbers or by rational intervals. In this way the computations are obtained approximating the arithmetic on real numbers by a limited precision arithmetic on the rationals. In exact real computation instead the result of a computation can be obtained with arbitrary precision, getting rid of the unfortunate phenomenon of the "round-off-error" [Gia93]. Using Moore interval arithmetic [Moo66] the round-off-error is known.

Our approach on real numbers computability uses the nested interval representation, i.e. a real number x is computable if there exists a r.e. nested set of rational intervals such that the limit is the degenerate real interval $[x, x]$. Clearly, we can work with computable real intervals instead of computable real numbers. Since every nested interval sequence alway converges to some real interval, we can to omite the notion of "converges effectively" as in the Cauchy representation. Finally we can define a natural extention of Turing machine (called **C**-Turing machine by Bedregal in [Bed96]) based upon this approach to compute on real intervals and show how to compute in a **C**-Turing machine the interval arithmetic by Moore [Moo66] and some other interval functions.

A **C**-Turing machine is basically a Turing machine extended with an infinite quantity of infinite tapes such that in each one of these infinite tapes our **C**-Turing machine computes the partial result of the computation of the infinite object in such way that in the limit the Turing machine computes the function to the infinite object. We think that each tape contains a finite approximation of the ideal object in such a way that the approximation in the tape i is a refinement of the approximation in the tape $i-1$ converging, in this way, to the ideal object, in a similar way to a sequence of nested rational intervals converging to a real interval.

2 Classical Real Computability

The notion of Church-Turing computability is defined on countable sets [Rog67]. However the original notion of effective procedure does not limit itself to countable sets. For example, it is natural one thinks that the identity function, in $\mathbb{R}$, is "computable". But, because the real numbers can not be represented in a finite alphabet, it is not computable by a classical Turing machine.

When there is a mechanical way of obtaining a real number we say that it is a computable real number. For example, the real number e is the limit when n go to ∞ of the sequence

$$S_n = 1 + \frac{1}{1!} + \frac{1}{2!} + \cdots + \frac{1}{n!}$$

Since the sum, product and division on rational numbers are computable we can assert that there is a Turing machine which computes e. Turing in 1937 [Tur37] introduced the first notion of computable real number and from them several other notions have emerged. However, the most similar with the notion of computable real number which will be seen in this work was given by Martin-Löf in [Lof70], where a real number is computable if the set of approximations (rational intervals) is r.e. We can reduce the definition of real computable number by consider a r.e increasing sequence and other r.e. decreasing sequence which converges to the real (the nested interval definition).

Proposition 1 *A real number x is computable if, and only if, there are r.e. sequences $p_0 < p_1 < p_2 < \cdots$ and $q_0 > q_1 > q_2 > \cdots$ of rational numbers such that*

$$\lim_{i\to\infty} p_i = \lim_{i\to\infty} q_i = x \quad \blacksquare$$

There exists severals approach to define computability for reals numbers, but the more accept of them is the Cauchy computability [Grz57, PR88, Wei95, Bra95]. A

function $f :\subseteq \mathbb{R} \longrightarrow \mathbb{R}$[1] is **Cauchy-computable** if there exist a computable operator
$F :\subseteq \mathbb{Q}^\omega \longrightarrow \mathbb{Q}^\omega$ which sends Cauchy sequence in Cauchy sequence in such a way that

$$f(x) = \lim_{i\to\infty} F((q_n)_{n\in\mathbb{N}})$$

when $x = \lim_{n\to\infty} q_n$ and $(\forall m > n) \mid q_n - q_m \mid < 2^{-n}$. Furthemore an operator $F :\subseteq \mathbb{Q}^\omega \longrightarrow \mathbb{Q}^\omega$ is called *computable* if the corresponding functional $F' :\subseteq \mathbb{Q}^\omega \times \mathbb{N} \longrightarrow \mathbb{Q}^\omega$ defined by $F'((q_n)_{n\in\mathbb{N}}, k) = F((q_n)_{n\in\mathbb{N}})(k)$ is computed by some Turing machine (F' is Church-Turing computable) [Grz57].

3 Scott-Computability

Domain theory was established in order to have appropriate spaces on which to define semantic functions for the denotational approach to programming language semantics which assigns to each piece of a program an element in a domain. Due to the possible self applicative nature of some programs, the domains must have sometimes some special properties. For example, it may be required that a domain be isomorphic to its own function space (i.e., $D \cong [D \longrightarrow D]$). Dana Scott found many spaces with these properties which can be classified in algebraic and continuous.

One of the most natural condition for a category of semantic domains for computer programming languages, in particular, and for any mathematical model of computing, is to have a good notion of computability, i.e. we need a flexible and intuitive theory of computability through finitary approximations which must agree with the Church-Turing theory of computability [Gun85]. Thus, in our case we need to say what we mean by a continuous domain to be effectively given, and what we mean by a continuous function between effectively given continuous domains to be computable. A definition of a computable element of an effectively given continuous domain would, one hopes, have the following property: a continuous function $f : D_1 \longrightarrow D_2$ is computable if, and only if, it is computable as element of the domain $\mathbf{D}_1 \to \mathbf{D}_2$. We hope that the usual constructors defined on continuous domains will turn out to be computables.

Since the Scott computability notion considers computability on the domains with the cardinality of the continuum and each element of a domain is obtained as limit (supremum) of a countable chain of basic elements, we can extend the Church-Turing computability notion in such a way that it coincides with the Scott-computability one when the domain is countable and the computability of continuous functions depends on the computability a la Church-Turing of a function on the

[1] $f :\subseteq \mathbb{R} \longrightarrow \mathbb{R}$ denote that the f is a partial function

basis. In this work, we will show that the role of a base to determine computability of functions is more essential than one given until now. It allows us to reduce, in some sense, the computability in the domain to a classical computability in a countable base.

One important application of this general theory is in computable analysis. For example, we can obtain a notion of computability on the real intervals which depends on the computability of a function on the rational intervals. We will show as this theory can be useful to develop a interval computability theory (as generalization of a real computability theory), which can have a similar role in the computability on the continuum as the computability on the natural number has with respect to the computability on countable sets.

3.1 Continuous domains

Let $\mathbf{D} = \langle D, \leq \rangle$ be a partially ordered set (poset). A set $\Delta \subseteq D$ is called *directed* if each of its finite subset has an upper bound, or equivalently, $\forall a, b \in \Delta \;\; \exists c \in \Delta$ such that $a \leq c$ and $b \leq c$. A poset $\mathbf{D}$ is *directed complete* (*dcpo* for short) if each directed set Δ has a least upper bound or supremum (denoted by $\bigsqcup \Delta$) and a *cpo* if it has a least element. We say that a is *way below* b (denoted by $a \ll b$) if for every directed set Δ such that $b \leq \bigsqcup \Delta$ then $a \leq x$ for some $x \in \Delta$. We let $\twoheaddownarrow x = \{y \in D \;/\; y \ll x\}$[2]. A dcpo $\mathbf{D}$ is called *continuous* if, for all $x \in D$, the set $\twoheaddownarrow x$ is directed and $x = \bigsqcup \twoheaddownarrow x$. A set $B \subseteq D$ is a *base* of $\mathbf{D}$ if for each $x \in D$, $\twoheaddownarrow x \cap B$ is a directed set with $x = \bigsqcup \twoheaddownarrow x \cap B$. A dcpo is continuous if, and only if, it has a base [AJ94].

An element $a \in D$ is *compact* if $a \ll a$. We will denote the set of all compact elements of $\mathbf{D}$ by $\mathbf{D}^0$. A continuous dcpo $\mathbf{D}$ is said *algebraic* if $\mathbf{D}^0$ is a base. A subset of a poset is said consistent if it has an upper bound in the poset. A continuous dcpo is a *continuous domain* if it has a countable base and each consistent set has a supremum. An algebraic dcpo is a *Scott domain* if the set of compact elements is countable and each consistent set has a supremum.

An element x of a poset $\mathbf{D}$ is *total* if $x \sqsubseteq y$ implies that $x = y$. $Tot(\mathbf{D})$ denote the set of total elements of $\mathbf{D}$.

There are several constructions on continuous domains, such as lifting, Cartesian product, function space, etc. For example, the *Cartesian product* of two continuous domains $\mathbf{D} = \langle D, \sqsubseteq_D \rangle$ and $\mathbf{E} = \langle E, \sqsubseteq_E \rangle$ is defined as the continuous domain $\mathbf{D} \times \mathbf{E} = \langle D \times E, \sqsubseteq \rangle$, where $(x, y) \sqsubseteq (x', y')$ if, and only if, $x \sqsubseteq_D x'$ and $y \sqsubseteq_E y'$. We will abreviate $\mathbf{D} \times \cdots \times \mathbf{D}$ by $\mathbf{D}^n$.

Let $\mathbf{D}$ and $\mathbf{E}$ be dcpo's. A function $f : E \longrightarrow D$ is called *continuous* if it is monotonic (i.e. if $x \leq y$ then $f(x) \leq f(y)$) and preserves least upper bounds (i.e. $f(\bigsqcup \Delta) = \bigsqcup f(\Delta)$).

[2]Analogously, we let $\twoheaduparrow x = \{y \in D \;/\; x \ll y\}$

The partial orders are endowed with a topology, called Scott topology where the notions of continuous functions coincide, i.e. a function is continuous w.r.t the order if and only if is continuous w.r.t. it topology.

Definition 2 *Let* $\mathbf{D} = \langle D, \sqsubseteq \rangle$ *be a dcpo.* $\mathcal{O} \subseteq D$ *is taken as open in the* **Scott topology** *of* D *if*

1. $x \in \mathcal{O}$ *and* $x \sqsubseteq y$ *then* $y \in \mathcal{O}$
2. *If* $\Delta \subseteq D$ *is a directed set and* $\bigsqcup \Delta \in O$ *then* $\Delta \cap \mathcal{O} \neq \emptyset$

The Scott topology on the dcpo $\mathbf{D}$ *will be denoted by* $\mathcal{T}_{\mathbf{D}}$.

The idea of this topology is as follows:

1. if information x suffices to indicate that test $\mathcal{O}$ has succeded, then any greater informations is sufficient a fortiori;
2. if the limit of a sequence of better and better approximations passes a test $\mathcal{O}$, then some of the approximants already pass. The last requirement is connected with the idea that open sets correspond to finitary tests.

Proposition 3 *[Law87] Let* $\mathbf{D}$ *be a continuous cpo and* $B \subseteq D$ *a base of* $\mathbf{D}$*. Then, the set*

$$\overline{B} = \{\Uparrow x \ / \ x \in B\}$$

is a base to $\mathcal{T}_{\mathbf{D}}$. ■

Proposition 4 *[Smy92] Let* $\mathbf{D}$ *be a dcpo. Then* $\mathcal{T}_{\mathbf{D}}$ *is a* T_0 *topology.* ■

Proposition 5 *Let* $\mathbf{D}$ *be a continuous cpo and* $B \subseteq D$ *a base of* $\mathbf{D}$*. If* $E \subseteq D$*, then, the set*

$$\overline{B_{Tot(\mathbf{D})}} = \{\Uparrow x \cap Tot(\mathbf{D}) \ / \ x \in B\}$$

is a base to $\mathcal{T}_{\mathbf{D}}/Tot(\mathbf{D})$. ■

Theorem 6 *[Smy92] Let* $\mathbf{D}_1 = \langle D_1, \sqsubseteq_1 \rangle$ *and* $\mathbf{D}_2 = \langle D_2, \sqsubseteq_2 \rangle$ *be dcpo's. A function* $f : D_1 \longrightarrow D_2$ *is continuous (as partial order) between* $\mathbf{D}_1$ *into* $\mathbf{D}_2$ *if, and only if,* f *is continuous topologically between* $\langle D_1, \mathcal{T}_{\mathbf{D}_1} \rangle$ *into* $\langle D_2, \mathcal{T}_{\mathbf{D}_2} \rangle$. ■

Let $\mathcal{T}$ a topology over a set X and $Y \subseteq X$. The subspace topology on Y induced by $\mathcal{T}$, denoted by $\mathcal{T}/Y$, is the relativization of $\mathcal{T}$ to Y, i.e. the sets open in the subspace are precisely those of the form $Y \cap \mathcal{O}$, where $\mathcal{O} \in \mathcal{T}$. Clearly, $\mathcal{T}/Y$ constitutes a topology on Y. Moreover, if we consider the totals elements of the cpo with it topology, will have a Hausdorff space.

Theorem 7 *[KT84] Let* $\mathbf{D}$ *be a dcpo. Then* $\mathcal{T}/Tot(\mathbf{D})$ *is metrizable.* ■

3.2 Computability on continuous domains

The original notion of effectively given domain was presented by Smyth [Smy77] for continuous cpo's.

Definition 8 *Let* $\mathbf{D} = \langle D, \sqsubseteq \rangle$ *be a continuous domain,* B *a countable basis. The pair* $\langle \mathbf{D}, B \rangle$ *is an* **effectively given continuous domain**, *EGCO in short, if* $\{(a,b) \in B \times B \;/\; a \ll b\}$ *is a r.e. set.*

In the original definition of effectively given domains was used an enumeration of a base [Smy77, WS83]. But, using the informal notion of r.e. set [3] [Rog67] we can to avoid this enumerations, simplifying the notion of effectively given continuous domains.

It is possible to extend, naturally, all the constructors on continuous domains to effectively given continuous domains. For example, the Cartesian product of the effectively given continuous domains $\langle \mathbf{D}, B_D \rangle$ and $\langle \mathbf{E}, B_E \rangle$ is the effectively given continuous domain $\langle \mathbf{D} \times \mathbf{E}, B_D \times B_E \rangle$.

The notion of EGCD brings up discussions on computability. First to the elements of the domain and then to the continuous functions between domains.

Definition 9 *Let* $\langle \mathbf{D}, B \rangle$ *be an EGCD. An element* $x \in D$ *is* **computable** *if the set* $\{b \in B \;/\; b \ll x\}$ *is r.e.*

Definition 10 *Let* $\langle \mathbf{D}_1, B_1 \rangle$ *and* $\langle \mathbf{D}_2, B_2 \rangle$ *be EGCD's. A continuous function* $f : D_1 \longrightarrow D_2$ *is* **computable** *if the set* $G(f) = \{(a,b) \in B_1 \times B_2 \;/\; b \ll f(a)\}$ *is r.e.*

Proposition 11 *Let* $\langle \mathbf{D}_1, B_1 \rangle$ *and* $\langle \mathbf{D}_2, B_2 \rangle$ *be EGCD's. If* $f : D_1 \longrightarrow D_2$ *is a continuous function such that* $graph(f) = \{(a,b) \in B_1 \times B_2 \;/\; b = f(a)\}$ *is r.e. then* f *is Scott-computable.*

PROOF: Because $\langle \mathbf{D}_2, B_2 \rangle$ is an EGCD there exists a program ϕ_2 that list the set $\{(a,b) \in B_2 \times B_2 \;/\; a \ll b\}$. If $graph(f) = \{(a,b) \in B_1 \times B_2 \;/\; b = f(a)\}$ is r.e. then there is a program ρ which lists element to element the set $B \times B$. Denote by $\rho(i)$ the $i^{\underline{th}}$ element listed by the program ρ.

The following program gives an effective enumeration of $G(f)$.

[3] Informally, a set is recursively enumerable (r.e. in short) if there exists an effective procedure which lists each element of the set (also it are allowed repetitions).

```
i := 1;
n := 1;
while i ≠ 0 do
        k := i;
        while k > 0 do
                a := π1(ρ(i));
                b := π2(ρ(i));
                j := 1;
                l := 0;
                while l ≠ k do
                        while b ≠ π2(φ2(j)) do j := j + 1; end − while
                        l := l + 1;
                end − while
                print "G(f)(" n ") = (" π1(φ2(j)), b ")";
                n := n + 1;
                k := k − 1;
        end − while
        i := i + 1;
end − while;
```

where π_1 and π_2 are the first projection and second projection, respectively. Since the while sentence always halts, the set $G(f)$ is r.e. Therefore, f is computable. ■

In the following we will show that our notion of computable function agrees with that of the Church-Turing-computable function, in the sense that the computability of a function depend on the Church-Turing computability of a function between their respectives basis (which are countables).

Theorem 12 *Let $\langle \mathbf{D}_1, B_1\rangle$ and $\langle \mathbf{D}_2, B_2\rangle$ be EGCD's. If $f : D_1 \longrightarrow D_2$ is a computable function then there exists a Church-Turing computable monotonic function $g : \mathbb{N} \times B_1 \longrightarrow B_2$ such that*

$$f(x) = \bigsqcup\{b \in B_2 \ / \ g(n, a) = b \textit{ for some } a \ll x \textit{ and } n \in \mathbb{N}\}.$$

PROOF: Define the function $g : \mathbb{N} \times B_1 \longrightarrow B_2$ as following $g(n, x) = \pi_2(a, b) = b$ where (a, b) is the $n^{\underline{th}}$ pair in the set $G(f)$ such that $f(a) \ll f(x)$. Since $G(f)$ is a r.e. set, the function g is Church-Turing computable. So,

$$\begin{aligned} f(x) &= \bigsqcup\{b \in B_2 \ / \ b \ll f(x)\} && \text{by def. of base} \\ &= \bigsqcup\{b \in B_2 \ / \ b \ll f(x) \text{ for some } (a,b) \in G(f)\} && \text{by def. of } G(f) \\ &= \bigsqcup\{b \in B_2 \ / \ g(n,a) = b \text{ for some } a \ll x \text{ and } n \in \mathbb{N}\} && \text{by def. of } g \ \blacksquare \end{aligned}$$

Theorem 13 *[Bed96] Let $\langle \mathbf{D}_1, B_1\rangle$ and $\langle \mathbf{D}_2, B_2\rangle$ be EGCD's. If $g : B_1 \longrightarrow B_2$ is a Church-Turing computable monotonic function then there exists a computable function such that*

$$f(x) = \bigsqcup\{g(y) \;/\; y \ll x\}. \;\blacksquare$$

Theorem 14 *Let* $\langle \mathbf{D}_1, B_1\rangle$ *and* $\langle \mathbf{D}_2, B_2\rangle$ *be EGCD's. If a function* $f : D_1 \longrightarrow D_2$ *is computable, then* f *is continuous w.r.t. the Scott topology.*

PROOF: By definition all computable function is continuous w.r.t. the order. By theorem 6, All continuous function w.r.t. the order is continuous w.r.t. the Scott Topology. So, all computable function is continuous w.r.t. the Scott topology. $\blacksquare$

Clearly, as consequence of this theorem the converse of the theorem 12 holds, i.e. if $g : \mathbb{N} \times B_1 \longrightarrow B_2$ is a Church-Turing computable monotonic function then $f(x) = \bigsqcup\{g(n, a) \;/\; n \in \mathbb{N} \text{ and } a \ll x\}$ is a computable function from D_1 into D_2. Thus the computability notion on continuous domains agrees strongly with the notion of Church-Turing computability, in the sense that computability on continuous domains extend the notion of Church-Turing computability to computability on sets with the continuum cardinality. So, we can generalize the Church thesis to

Generalized Church-Thesis: A function $f : A \longrightarrow B$ is computable informally or by a physical device, if and only if, there exist EGCD's $\langle\langle A, \sqsubseteq_1\rangle, B_1\rangle$ and $\langle\langle B, \sqsubseteq_2\rangle, B_2\rangle$, such that f is computable in the sense of the definition 10.

4 C-Turing machines

In order to computing with infinite objects, we can conceive a kind of universal Turing machine which considers an infinite quantity of infinite tapes, such that in each one of these infinite tapes our **C**-Turing machine computes the partial result of the computation of the infinite object in such way that in the limit the Turing machine computes the function for the infinite object. Each tape contains a finite approximation of the ideal object in such a way that the approximation in the tape i is a refinement of the approximation in the tape $i-1$ converging, in this way, to the ideal object. **C** denote the fact that we can compute in a **C**-Turing machine on sets with the cardinality of continuum.

A **C**-Turing machine consists of an infinite quantity of infinite tapes divided in cells, a finite control and a tape head that scans one cell of the current tape at a time. In one move the Turing machine, depending upon the symbol scanned in the current tape by the tape head and on the current state of the finite machine, the **C**-Turing machine changes the state, prints a symbol on the tape cell scanned, replacing what was written there and moves its head left (L) or right (R) one cell or moving the head tape flagged the current cell in the current tape and go to flagged cell of the next tape (N) or the previous tape (P) (initially the flagged cell is the most left cell of each tape). Formally an extended Turing machine program is described as a tuple $CTM = \langle Q, \Sigma, \Gamma, \delta, q_0, B \rangle$ where

- Q is a finite set of states,
- Γ is the finite set of allowable tape symbols,
- $B \in \Gamma$ is the special symbol denoting the blank,
- $\Sigma \subset \Gamma$ such that $B \notin \Sigma$, is the set of input symbols,
- δ is the next move (partial) function, a mapping from $Q \times \Gamma$ into $Q \times \Gamma \times \{L, R, N, P\}$
- $q_0 \in Q$ is the start state,

Since our model of Turing machine has not final states the computation in the **C**-Turing machine, are essentially infinite. Thus, we need distinguish two kind of non-halt: the computable and the loop. The **C**-Turing machine considers infinite quantity of infinites tapes which does not exist physically and infinite computations which cannot be completed in reality. However, for a computation of a **C**-Turing machine in any finite time we can observe the partial result of these computations. In order to guarante that a partial result of the computation is not going to be changed at some later time, we need to add a mechanism which cohibits the back two at consecutive times. This mechanism can be an interruptor of three states: green, yellow and red. The green state is the initial state of the interrupt, this state changes for the yellow state only when a command P is executed by the **C**-Turing machine. The yellow state returns to the green state only when the command N is executed by the machine. Finally, the yellow state changes to the red state only when the command P is again executed by the machine. This state causes an interruption of the execution of the **C**-Turing machine.

Definition 15 *A* **C**-*Turing machine* CTM *compute a partial function* $f_M : \omega\!-\!Seq(\Sigma^*) \longrightarrow \omega\!-\!Seq(\Gamma^*)$[4] *if the machine* CTM *with input* $X = x_1, x_2, \ldots$ *computes forever writing the sequence* Y *and* $f_M(X) = Y$.

Example 16 *We will consider that each rational number is represented as fraction. So, a rational number is represented by a pair of natural numbers separate by the symbol "/" and each natural number are represented in their unary notation. Thus, a rational interval* $[p, q]$ *is represented by a string* "["w_1"/"v_1","w_2"/"v_2"]", *such that* $w_1, v_1, w_2, v_2 \in \{1\}^*$, $p = |w_1| \,/\, |v_1|$ *and* $q = |w_2| \,/\, |v_2|$, *where* $|w|$ *is the length of the string* w.

By simplicity, only we consider $\mathbb{I}(\mathbb{Q})^+$, *the set of positive rational intervals.*

[4] $\omega\!-\!Seq(A)$ denote the set of infinite sequences the element of A.

	Σ							
Q	$[$	1	$/$	$;$	$]$	$\&$	B	X
q_0	$(q_0,[,R)$	$(q_0,1,R)$	$(q_0,/,R)$	$(q_0,;,R)$	$(q_0,],R)$		$(q_1,\&,R)$	
q_1							$(q_2,[,L)$	
q_2	$(q_3,[,R)$	$(q_2,1,L)$	$(q_2,/,L)$	$(q_2,;,L)$	$(q_2,],L)$	$(q_2,\&,L)$		
q_3		(q_3,X,R)	$(q_3,/,R)$	$(q_4,;,L)$				
q_4	$(q_7,[,R)$	$(q_4,1,L)$	$(q_4,/,L)$					$(q_5,1,R)$
q_5	$(q_5,[,R)$	$(q_5,1,R)$	$(q_5,/,R)$	$(q_5,;,R)$	$(q_5,],R)$	$(q_5,\&,R)$	$(q_6,1,L)$	
q_6	$(q_6,[,L)$	$(q_6,1,L)$	$(q_6,/,L)$	$(q_4,;,L)$	$(q_6,],L)$	$(q_6,\&,L)$		
q_7	$(q_7,[,R)$	$(q_7,1,R)$	$(q_8,/,R)$					
q_8		(q_8,X,R)		$(q_9,;,R)$				
q_9	$(q_9,[,R)$	$(q_9,1,R)$	$(q_9,/,R)$		$(q_9,],R)$	$(q_9,\&,R)$	$(q_{10},/,L)$	
q_{10}	$(q_{10},[,L)$	$(q_{10},1,L)$	$(q_{10},/,L)$	$(q_{11},;,L)$	$(q_{10},],L)$	$(q_{10},\&,L)$		
q_{11}		$(q_{11},1,L)$	$(q_{13},/,R)$					$(q_{12},1,R)$
q_{12}	$(q_{12},[,R)$	$(q_{12},1,R)$	$(q_{12},/,R)$	$(q_{12},;,R)$	$(q_{12},],R)$	$(q_{12},\&,R)$	$(q_{10},1,L)$	
q_{13}	$(q_{13},[,R)$	$(q_{13},1,R)$	$(q_{13},/,R)$	$(q_{13},;,R)$	$(q_{13},],R)$	$(q_{13},\&,R)$	$(q_{14},;,L)$	
q_{14}	$(q_{14},[,L)$	$(q_{14},1,L)$	$(q_{14},/,L)$	$(q_{15},;,L)$	$(q_{14},],L)$	$(q_{14},\&,L)$		
q_{15}		(q_{15},X,L)	$(q_{15},/,R)$		$(q_{16},],L)$			
q_{16}		$(q_{16},1,L)$	$(q_{16},/,L)$					$(q_{17},1,R)$
q_{17}	$(q_{17},[,R)$	$(q_{17},1,R)$	$(q_{17},/,R)$		$(q_{17},],R)$	$(q_{17},\&,R)$	$(q_{18},1,L)$	
q_{18}	$(q_{18},[,L)$	$(q_{18},1,L)$	$(q_{18},/,L)$	$(q_{19},;,R)$	$(q_{18},],L)$	$(q_{18},\&,L)$		$(q_{17},1,D,N))$
q_{19}		$(q_{19},1,R)$	$(q_{20},/,R)$					
q_{20}		(q_{20},X,R)			$(q_{21},],R)$			
q_{21}	$(q_{21},[,R)$	$(q_{21},1,R)$	$(q_{21},/,R)$	$(q_{21},;,R)$		$(q_{21},\&,R)$	$(q_{22},/,L)$	
q_{22}	$(q_{22},[,L)$	$(q_{22},1,L)$	$(q_{22},/,L)$	$(q_{22},;,R)$	$(q_{23},],L)$	$(q_{22},\&,L)$		
q_{23}		$(q_{23},1,L)$	$(q_{25},/,R)$					$(q_{24},1,R)$
q_{24}	$(q_{24},[,R)$	$(q_{24},1,R)$	$(q_{24},/,R)$	$(q_{24},;,R)$	$(q_{24},],R)$	$(q_{24},\&,R)$	$(q_{22},1,L)$	
q_{25}	$(q_{25},[,R)$	$(q_{25},1,R)$	$(q_{25},/,R)$	$(q_{25},;,R)$	$(q_{25},],R)$	$(q_{25},\&,R)$	$(q_0,1,N)$	

Figure 1: **C**-Turing machine computing the function $f([r,s]) = [r+1, s+1]$

1. *Let $f : \mathbb{I}(\mathbb{R})^+ \longrightarrow \mathbb{I}(\mathbb{R})^+$ be the function $f([r,s]) = [r+1, s+1]$. It can be obtained as the supremum of their restrictions to $\mathbb{I}(\mathbb{Q})$. Since, $g : \mathbb{I}(\mathbb{Q}) \longrightarrow \mathbb{I}(\mathbb{Q})$, definided by $g([p,q]) = [p+1, q+1]$, is computable, we can define a* **C**-*Turing machine computing g in each tape. If $[p_1, q_1] \ll [p_2, q_2] \ll \ldots \ll [p_n, q_n] \ll \ldots$ and $\lim_{i\to\infty}[p_i, q_i] = [r, s]$ then $f([r,s]) = \bigsqcup\{g(1, [p_1, q_1]), \ldots, g([p_n, q_n]), \ldots\}$. Thus, the following* **C**-*Turing machine compute g in each tape, and therefore compute f in the limit.*

   - $Q = \{q_0, q_1, \ldots, q_{25}\}$
   - $\Sigma = \{[, 1, /, ;,]\}$
   - $\Gamma = \{[, 1, /, ;,], \&, B, X\}$
   - δ *defined in figure 1*

	Σ										
Q	$[$	$-$	1	$/$	$;$	$]$	$\&$	B	X	Y	Z
q_0	$(q_0, [, R)$	$(q_0, -, R)$	$(q_0, 1, R)$	$(q_0, /, R)$	$(q_0, ;, R)$	$(q_0,], R)$	$(q_0, \&, R)$	$(q_1, [, L)$			
q_1	$(q_2, [, R)$	$(q_1, -, L)$	$(q_1, 1, L)$	$(q_1, /, L)$	$(q_1, ;, L)$	$(q_1,], L)$	$(q_1, \&, L)$				
q_2		(q_5, Y, R)	(q_3, X, R)	(q_6, Z, R)	(q_7, Z, R)	$(q_8,], R)$					
q_3	$(q_3, [, R)$	$(q_3, -, R)$	$(q_3, 1, R)$	$(q_3, /, R)$	$(q_3, ;, R)$	$(q_3,], R)$	$(q_3, \&, R)$	$(q_4, 1, L)$			
q_4	$(q_4, [, L)$	$(q_4, -, L)$	$(q_4, 1, L)$	$(q_4, /, L)$	$(q_4, ;, L)$	$(q_4,], L)$	$(q_4, \&, L)$		(q_2, X, R)	(q_2, Y, R)	(q_2, Z, R)
q_5	$(q_5, [, R)$	$(q_5, -, R)$	$(q_5, 1, R)$	$(q_5, /, R)$	$(q_5, ;, R)$	$(q_5,], R)$	$(q_5, \&, R)$	$(q_4, -, L)$			
q_6	$(q_6, [, R)$	$(q_6, -, R)$	$(q_6, 1, R)$	$(q_6, /, R)$	$(q_6, ;, R)$	$(q_6,], R)$	$(q_6, \&, R)$	$(q_4, /, L)$			
q_7	$(q_7, [, R)$	$(q_7, -, R)$	$(q_7, 1, R)$	$(q_7, /, R)$	$(q_7, ;, R)$	$(q_7,], R)$	$(q_7, \&, R)$	$(q_4, ;, L)$			
q_8	$(q_8, [, R)$	$(q_8, -, R)$	$(q_8, 1, R)$	$(q_8, /, R)$	$(q_8, ;, R)$	$(q_8,], R)$	$(q_8, \&, R)$	$(q_0,], N)$			

Figure 2: **C**-Turing machine computing the function $f([r, s]) = [-s, -r]$

2. Let $f : \mathbb{I}(\mathbb{R}) \longrightarrow \mathbb{I}(\mathbb{R})$ be the function $f([r, s]) = [-s, -r]$. If the interval $[r, s]$ can be considered the supremum of a chain of rational intervals, $[p_i, q_i]$, then, clearly, the interval $[-s, -r]$ is the supremum of the chain $[-q_i, -p_i]$. Therefore, if the function $g : \mathbb{I}(\mathbb{Q}) \longrightarrow \mathbb{I}(\mathbb{Q})$ is the function f restrict to $\mathbb{I}(\mathbb{Q})$, then $f(x) = \bigsqcup\{g(y) : y \ll x\}$, i.e. it satisfies the theorem 13. The following **C***-Turing machine compute g in each tape and, therefore, compute f in the limit.*

- $Q = \{q_0, q_1, \ldots, q_8\}$
- $\Sigma = \{[, -, 1, /, ;,]\}$
- $\Gamma = \{[, -, 1, /, ;,], \&, B, X, Y, Z\}$
- *δ definided in figure 2*

Notice that the n in $g(n, \emptyset)$ is generated by the machine according to the current tape. Thus, it is clear that there exists a strong relationship between Scott-computable functions and computable functions by **C**-Turing machines.

Theorem 17 *Let $\langle \mathbf{D}_1, B_1\rangle$ and $\langle \mathbf{D}_2, B_2\rangle$ be effectively given continuous domains. $f : D_1 \longrightarrow D_2$ is Scott-computable function if and only if there is a* **C***-Turing machine computing f_M and injective functions $\phi : B_1 \longrightarrow \Sigma^*$ and $\psi : \Gamma^* \longrightarrow B_2$ such that to each ω-chain C in B_1, $f(\bigsqcup C) = \bigsqcup \psi(f_M(\phi(C)))$.* ■

Thus, we have stated that **C**-Turing machines are adequate machine models for the Scott-computability, and therefore, we have a new evidence that our extend Church tesis is correct.

5 Computability on the real intervals continuous domain

In one of his early papers on domain theory, D. Scott [Sco70] suggested that a cpo consisting of intervals with real numbers as end points, real intervals in short, can be used to study computability on the real numbers. Previously, Martin-Löf gave a notion of computability on real numbers based on the recursive enumerability of their set of approximations (rational intervals). In both cases, the real line is embedded in spaces of approximations where a notion of computability can be defined in a natural way [Gia93]. On the other hand, Acióly in [Aci91] conjectured that the computability on the real intervals will have a similar role in the computability on the continuum as the computability on the natural numbers has with respect to the computability on countable sets. In this section we will show that the continuous domain of real intervals used by Acióly [Aci91] to give a computacional foundation of interval mathematics is effectively given and allows us to define a computability notion on the interval analysis, called here **Moore-computability**, as well as, on real analysis. So, define the continuous domain of real intervals by $\mathcal{R} = \langle \mathbb{I}(\mathbb{R}), \sqsubseteq \rangle$ where

- $\mathbb{I}(\mathbb{R}) = \{[r,s] : \ r,s \in \mathbb{R} \text{ and } r \leq s\} \cup \{[-\infty, +\infty]\}$
- $[r,s] \sqsubseteq [t,u]$ if, and only if, $r \leq t$ and $u \leq s$

with $\leq$ being the usual "lesser or equal" order on the extended real numbers (i.e. $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$). Observe, that $[r,s] \sqsubseteq [t,u]$ if, and only if, $[t,u] \subseteq [r,s]$. However, we opted by the first definition, because for interval mathematics the use of extreme is essential. The way below relation associated to this continuous domain is defined by

$$[r,s] \ll [t,u] \text{ if, and only if, } r < t \text{ and } u < s$$

The total elements of $\mathcal{R}$ is the set of all degenerate intervals, i.e.

$$Tot(\mathcal{R}) = \{[r,r] \ / \ r \in \mathbb{R}\}.$$

Since, to each degenerate interval $[r,r]$ has associated the real number r. Thus, the total elements of the continuous domain $\mathcal{R}$ can be thinked as the set of the real numbers.

A countable base for this continuous domain are the rational intervals, i.e. the set $\mathbb{I}(\mathbb{Q}) = \{[p,q] \ / \ p,q \in \mathbb{Q} \text{ and } p < q\}$. Since the base $\mathbb{I}(\mathbb{Q})$ is r.e. [BA93] and $\ll$ is clearly decidible, the set $\{(I,J) \in \mathbb{I}(\mathbb{Q}) \times \mathbb{I}(\mathbb{Q}) \ / \ I \ll J\}$ also is r.e. So, $\langle \mathcal{R}, \mathbb{I}(\mathbb{Q}) \rangle$ is an EGCD and therefore we can have a notion of computable real interval and computability of interval functions.

The *Moore-computable elements* (the computable elements of $< \mathcal{R}, \mathbb{I}(\mathbb{Q}) >$) are the real intervals which are supremum of r.e. chains of rational intervals. Clearly, from the definition of computable real number given by [Lof70], the Moore-computable elements are exactly the Turing-computable real numbers.

Proposition 18 *Let x and y be real numbers such that $x \leq y$. $[x, y]$ is Moore-computable if, and only if, x and y are Turing-computables.*

PROOF: If $[x, y]$ is Moore-computable then $\{[p, q] \in \mathbb{I}(\mathbb{Q})\ p/\ p < x \leq y < q\}$ is r.e. and therefore the sets $\{p \in \mathbb{Q}\ /\ p < x\}$ and $\{q \in \mathbb{Q}\ /\ y < q\}$ are r.e. So, we can obtain a r.e. sequence $p_0, p_1, \ldots, p_n, \ldots$ of rational numbers such that p_0 being the integer part of x, $p_0 < p_1 < \ldots < p_n < \cdots$ and $x = \lim_{n\to\infty} p_n$. Thus, defining $p'_0 = p_0 + 1$ and $p'_{i+1} = 2p_{i+1} - p_i = p_{i+1} + p_{i+1} - p_i$ we obtain a r.e. sequence such that $p'_0 > p'_1 > \ldots > p'_n > \ldots$

$$\begin{aligned} \lim_{n\to\infty} p'_n &= \lim_{n\to\infty} p'_{n+1} \\ &= \lim_{n\to\infty} 2p_{n+1} - p_n \\ &= 2\lim_{n\to\infty} p_{n+1} - \lim_{n\to\infty} p_n \\ &= 2x - x \\ &= x \end{aligned}$$

So, by the proposition 1, the real x is Turing-computable. Dually, the real number y is Turing-computable.

On the other hand, if x and y are Turing-computable, then by the definition of computable real number, given by Martin-Löf in [Lof70], the sets $\Downarrow_{\mathbb{I}(\mathbb{Q})} x = \{[p, q] \in \mathbb{I}(\mathbb{Q})\ p/\ p < x < q\}$ and $\Downarrow_{\mathbb{I}(\mathbb{Q})} y = \{[p, q] \in \mathbb{I}(\mathbb{Q})\ /\ p < y < q\}$ are r.e. Thus, $\Downarrow_{\mathbb{I}(\mathbb{Q})} x \cap \Downarrow_{\mathbb{I}(\mathbb{Q})} y$ also is r.e. So, because clearly $\Downarrow_{\mathbb{I}(\mathbb{Q})} x \cap \Downarrow_{\mathbb{I}(\mathbb{Q})} y = \{[p, q] \in \mathbb{I}(\mathbb{Q})\ /\ p < x \leq y < q\}$, the real interval $[x, y]$ is Moore-computable. ■

Clearly, from this proposition, the computable degenerate real intervals are exactly the intervals $[r, r]$ where r is a Turing-computable number.

By the theorems 13 and 12, the computable interval functions on the EGCD $\mathcal{R}$ (*Moore-computable functions*) depends on the Church-Turing computability on the rational interval functions. In the following we will enumerate some Moore-computable functions.

Proposition 19 *The following functions from $\mathbb{I}(\mathbb{R})^n$ in $\mathbb{I}(\mathbb{R})$ are Moore-computables.*

1. $[r, s] + [t, u] = [r + t, s + u]$
2. $[r, s] - [t, u] = [r - u, s - t]$
3. $[r, s] \cdot [t, u] = [min\{rt, ru, st, su\}, max\{rt, ru, st, su\}]$

4.

$$[r,s]/[t,u] = \begin{cases} [r,s]\cdot[\frac{1}{u},\frac{1}{t}] & , \textit{if } 0 \notin [t,u] \\ [-\infty,+\infty] & , \textit{otherwise} \end{cases}$$

5. *$f_c(I) = c$ where $c \in \mathbb{I}(\mathbb{R})_e$, i.e. c is a Moore-computable interval.*

6.

$$|\,[r,s]\,| = \begin{cases} [0, max\{|\,r\,|, s\}] & , \textit{if } 0 \in [r,s] \\ [r,s] & , \textit{if } 0 < r \\ [-s,-r] & , \textit{otherwise} \end{cases}$$

7. *$exp\,[r,s] = [e^r, e^s]$*

8. *$log\,[r,s] = [log\ r, log\ s]$*[5]

9.

$$sin\,[r,s] = \begin{cases} [-1,1] & , \textit{if } 2\pi \leq s-r \\ [-1,1] & \pi \leq s-r < 2\pi \textit{ and } sin\ r + \frac{s-r}{2} = 0 \\ [minS, 1] & , \textit{if } \pi \leq s-r < 2\pi \textit{ and } sin\ r + \frac{s-r}{2} > 0 \\ [minS, 1] & , \textit{if } s-r < \pi \textit{ and } s \leq (k+1)\frac{\pi}{2} \leq r \textit{ for some } k \textit{ multiple of } 4 \\ [-1, maxS] & , \textit{if } \pi \leq s-r < 2\pi \textit{ and } sin\ r + \frac{s-r}{2} < 0 \\ [-1, maxS] & , \textit{if } s-r < \pi \textit{ and } s \leq (k-1)\frac{\pi}{2} \leq r \textit{ for some } k \textit{ multiple of } 4 \\ [minS, maxS] & , \textit{otherwise} \end{cases}$$

where $S = \{sin\ r, sin\ s\}$.

10. *$cos\,[r,s] = sin\,[r+\frac{\pi}{2}, s+\frac{\pi}{2}]$, $tg\ I = \frac{sin\ I}{cos\ I}$ and $sec\ I = \frac{1}{cos\ I}$*

11. *Each polinomial interval function $p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n$ where the a_i are Moore-computable intervals.*

PROOF: (Draft) The first four of these functions, called interval arithmetic, can be recovered from their rational interval restriction, which are, clearly, computable. The constant function to a computable real interval c can be recovered from a Church-Turing computable monotonic function $g(n,I) = [p_n, q_n]$ where $[p_i, q_i]$ is a r.e. sequence of nested rational intervals converges to c. The functions absolute value, exponent, logarithm and sin are all recovered from their respective rational interval restrictions which are trivially Church-Turing computables. The remain functions are based on the sin, arithmetic operations and the constante function π,

[5] Since the real logarithms is a function from $\mathbb{R}^+$ (positive real numbers set) into $\mathbb{R}$, their interval version is a function from $\mathbb{I}(\mathbb{R})^+$ to $\mathbb{I}(\mathbb{R})$, where $\mathbb{I}(\mathbb{R})^+$ is the natural subdomain of $\mathbb{I}(\mathbb{R})$.

which are all computables. Finally, because the sum, product and the coefficient are computables, the polinomial interval functions with computable coefficient can be recovered from the function $q : \mathbb{N} \times \mathbb{I}(\mathbb{Q}) \longrightarrow \mathbb{I}(\mathbb{Q})$ defined by

$$q(m, I) = a_{0,m} + a_{1,m}I + \ldots + a_{n,m}I^n,$$

where $a_{i,j}$ is the j-th rational interval in the effective enumeration of a_i. So, because q is clearly Church-Turing computable, p is Moore-computable. ■

An example of non Moore-computable function is the equality with zero, i.e. the function $eq0 : \mathbb{I}(\mathbb{R}) \longrightarrow \mathbb{I}(\mathbb{R})$ defined by

$$eq0([r, s]) = \begin{cases} [1, 1] & \text{, if [r,s]=[0,0]} \\ [0, 0] & \text{, otherwise} \end{cases}$$

It happens because, trivially this function is non monotonic (For example, $[-1, 1] \sqsubseteq [0, 0]$, nevertheless, $eq0([-1, 1]) = [0, 0] \not\sqsubseteq eq0([0, 0]) = [1, 1]$) and therefore is non continuous. So, $eq0$ is non Moore-computable. We also can think $\{0, 1\}$, i.e. $\{[0, 0], [1, 1]\}$, as a domain with two elements, but independently of how we order them, this function remains be non continuous.

6 Computability on the real numbers

If we use the continuous domain $\langle \overline{\mathbb{R}}, \leq \rangle$ where $\overline{\mathbb{R}} = \mathbb{R} \cup \{-\infty, +\infty\}$ to introduce a computability theory on the real numbers, we have intuitively computable functions, such as the invertion of sign and division, noncomputables, because a requirement would be that the function be continuous in this domain, but trivially, these functions are not monotonics. These unpleasant facts do not hold on domain theory where the computability on the real numbers is derived from a computability on the real interval domain, as suggested by Scott [Sco70].

Before we will see a mechanism to transform a real function into a interval function.

Definition 20 *[Moo66] Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$. The* **interval extension** *of f is defined as the interval function $F : \mathbb{I}(\mathbb{R})^n \longrightarrow \mathbb{I}(\mathbb{R})$ such that*

$$F(I_1, \cdots, I_n) = \bigcap \{I \in \mathbb{I}(\mathbb{R}) \ / \ f(A) \in I \text{ for each } A \in I_1 \times \ldots \times I_n\}$$

Since, the nonempty intersection of real intervals is an interval, this function is well defined. In other words, the computability of a real function depend upon the computability of its interval extension. However, its reverse does not is hold, i.e. not all Moore-computable function is the interval extension of some computable real function (for example, the interval function $f(I) = [e, \pi]$).

Proposition 21 *Let $f : \mathbb{R}^n \longrightarrow \mathbb{R}$. The interval extension $F : \mathbb{I}(\mathbb{R})^n \longrightarrow \mathbb{I}(\mathbb{R})$ of f satisfies*

$$F(I_1, \cdots, I_n) = [\bigsqcap\{f(A) \;/\; A \in I_1 \times \ldots \times I_n\}, \bigsqcup\{f(A) \;/\; A \in I_1 \times \ldots \times I_n\}]$$

where the least upper bound ($\bigsqcup$) and the great lower bound ($\bigsqcap$) are w.r.t. the usual order on the real numbers.

Definition 22 *A function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is computable if its interval extension $F : \mathbb{I}(\mathbb{R})^n \longrightarrow \mathbb{I}(\mathbb{R})$ is Moore-computable.*

Proposition 23 *If a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is computable then there exists a Moore-computable function $F : \mathbb{I}(\mathbb{R})^n \longrightarrow \mathbb{I}(\mathbb{R})$ such that*

$$F([x_1, x_1], \ldots, [x_n, x_n]) = [f(x_1, \ldots, x_n), f(x_1, \ldots, x_n)]$$

PROOF: Straighforward from the proposition 21 and the definition 22. ■

Observe that, because we identified the degenerate intervals with the real numbers, this proposition declare that a real function f is computable if there exists a Moore-computable function F such that $F/tot(\mathcal{R}^n) = f$.

Theorem 24 *A function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is computable if, and only if, there exists a Church-Turing computable function $g : \mathbb{N} \times \mathbb{I}(\mathbb{Q})^n \longrightarrow \mathbb{I}(\mathbb{Q})$ such that*

$$f(x_1, \ldots, x_n) = \bigcap\{g(k, I_1, \ldots, I_n) \;/\; x_i \in I_i \text{ for each } i = 1, \ldots n \text{ and some } k \in \mathbb{N}\}$$

PROOF: Straighforward from the definition 22 and theorem 12. ■

Corollary 25 *A function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$ is computable if, and only if, there exists a Church-Turing computable function $g : \mathbb{N} \times \mathbb{Q}^n \longrightarrow \mathbb{Q}$ such that*

$$f(x_1, \ldots, x_n) = \bigsqcup\{g(k, y_1, \ldots, y_n) \;/\; y_i < x_i \text{ for each } i = 1, \ldots n \text{ and some } k \in \mathbb{N}\}. \;\blacksquare$$

The following proposition, says that our notion of computability is contained but not necessarelly properly contained, in the notion of Cauchy computability.

Theorem 26 *If a function $f : \mathbb{R} \longrightarrow \mathbb{R}$ is computable then f is Cauchy computable.*

PROOF: If f is computable then, by the corollary 25, there exists a function $g : \mathbb{N} \times \mathbb{Q} \longrightarrow \mathbb{Q}$ such that $f(x) = \bigsqcup\{g(k, y) \;/\; y < x \text{ for some } k \in \mathbb{N}\}$. Define, $F : \mathbb{Q}^\omega \longrightarrow \mathbb{Q}^\omega$ by $F((q_n)_{n\in\mathbb{N}}) = (g(n, q_n))_{n\in\mathbb{N}}$. So, because g is Church-Turing computable, the function F'$((q_n)_{n\in\mathbb{N}}, k) = F((q_n)_{n\in\mathbb{N}})(k) = g(k, q_k)$ also is Church-Turing computable. Therefore, f is Cauchy computable. ■

But, our notion of real computability preserves several important properties of Cauchy computability.

Proposition 27 *The following real functions are computables:*

1. *Let $x, y \in \mathbb{R}$. $x + y$, $x - y$ and $x \cdot y$*
2. *$f(x) = c$ for each computable real number c*
3. *$|x|$*
4. *e^x and log x*
5. *sin x, cos x, tan x and sec x*

PROOF: Clearly the interval extension of these functions are the functions described in the proposition 19, which are all Moore-computable. Therefore, these functions are computable. ■

7 Computability and Continuity

In [Bed96] is shown that the Scott topology on the continuous domain of real interval is compatible with the canonical real line topology in the sense that the subtopology for the total elements of $\mathcal{R}$ is homeomorphic to the real line. Thus, we hope that if the interval extension of a real function is continuous then the real function is also continuous in the real line topology.

Theorem 28 $\langle Tot(\mathcal{R}), \mathcal{T}/Tot(\mathcal{R})\rangle$ *is homeomorphic to* $\langle \mathbb{R}, \sigma\rangle$*, where* σ *is the usual real line topology.*

PROOF: Consider $\mathbb{I}(\mathbb{Q})$ as a base of continuous domain $\mathcal{R}$. By proposition 3, the set

$$\overline{\mathbb{I}(\mathbb{Q})} = \{\Uparrow[p, q] \;/\; [p, q] \in \mathbb{I}(\mathbb{Q})\}$$

is a base to Scott topology of $\mathcal{R}$, $\mathcal{T}_\mathcal{R}$. So, by proposition 5, for each open basic set $\Uparrow[p, q]$ the set

$$\Uparrow[p,q] \cap Tot(\mathcal{R}) = \{[r,r] \;/\; p < r < q\}$$

is an open basic set of topology $\mathcal{T}/Tot(\mathcal{R})$. Let $B_{Tot(\mathcal{R})}$ the set of basic open sets of $Tot(\mathcal{R})$.

Since is known from real analysis, the set

$$B_{\mathbb{R}} = \{(p,q) \;/\; p < q \text{ and } p,q \in \mathbb{Q}\},$$

where (p,q) is open rational interval, is a countable base of σ. Thus, clearly, $\mathcal{T}/Tot(\mathcal{R})$ and σ have a "same" base. Therefore, the function $f : \sigma \longrightarrow Tot(\mathcal{R})$ defined by

$$f(\mathcal{O}) = \{[r,r] \in \mathbb{I}(\mathbb{R}) \;/\; r \in \mathcal{O}\}$$

is an homeomorphism. ■

Clearly, we can to extend it theorem to cartesian products of $\mathcal{R}$.

Corollary 29 *$\langle Tot(\mathcal{R}^n), \mathcal{T}/Tot(\mathcal{R}^n)\rangle$ is homeomorphic to $\langle \mathbb{R}^n, \sigma^n\rangle$, where σ^n is the Euclidean topology of $\mathbb{R}^n$.* ■

From theorem 14 we have that each computable function is continuous w.r.t. the Scott topology. Thus, trivially, we have that each Moore-computable interval function $f : \mathbb{I}(\mathbb{R})^n \longrightarrow \mathbb{I}(\mathbb{R})$ is continuous w.r.t. the Scott Topology. It continuity property of Moore-computable interval functions can be extended to the computable real function as follow:

Theorem 30 *Let $F : \mathbb{I}(\mathbb{R})^n \longrightarrow \mathbb{I}(\mathbb{R})$ be the interval extension of a function $f : \mathbb{R}^n \longrightarrow \mathbb{R}$. If F is a Scott-continuous function then f is continuous w.r.t. the Euclidian topology.* ■

Corollary 31 *If $f : \mathbb{R}^n \longrightarrow \mathbb{R}^m$ is computable then f is continuous w.r.t. the Euclidian topology.* ■

In this way we obtain a proof that our approach to real computability also holds to classical result in computable analysis: every computable functions on real numbers is continuous with respect to the canonical real line topology.

Another clasical result is the test of equality between a real number and zero is not computable. Here, because their interval extension (the function $eq0$ of the previous section) is not Moore-computable, we also obtain the non computability of the equality with zero.

8 Conclusion

Our main contribution, in domain theory, is to give to the countable basis of a continuous domain a more preponderant role to show the computability of functions. We saw that a function on sets with the cardinality of the continuum is computable if, and only if, there exists a function on their respective countable bases which goes computing finite aproximations of infinite object in such way that in the limit, the function computes the ideal object. Thus, it is reasonable to introduce an extended Turing machine which considers an infinite quantity of infinite tapes for computating on the continuum [Bed96]. An important contribution of the **C**-Turing machine is that we show computations on the real intervals (and therefore on the real numbers) as universal Turing machines which compute on the rational interval.

In [Aci91] was shown the adequacity of continuous domain theory to give a foundation to interval mathematics. Thus, the Moore-computability notion given here (computability on the real interval continuous domain) can be useful to proporcionate an apropriate computability theory for interval analisys.

In the early paper by Scott [Sco70], he suggest the use of real interval domain to study the computability on the real numbers. In this work, we use the well known interval analysis technique for extending real functions to interval functions [Moo66] to make our goal possible. It was shown, that the computability on the real numbers can be studied from a computability theory on the real intervals.

The Aciòly conjecture [Aci91] is that the computability on the real intervals will have a similar role in the computability on the continuum as the computability on the natural numbers has with respect to the computability on countable sets. Thus, we believe that the Moore-computability notion will be a start point in the construction of a theory of computable analysis which will be more simple, because it is based on the computability on the rational intervals which is more intuitive and stronger notion than that based on Cauchy sequence on rational numbers as in [Wei95, PR88].

References

[AJ94] Samson Abramsky and Achim Jung. *Domain Theory.* In Handbook of Logic in Computer Science, Oxford University Press, 1994.

[Aci91] Benedito Melo Aciòly. *Computational Foundation of Interval Mathematics* (in portuguese). PhD thesis, CPGCC da UFRGS, Porto Alegre, 1991.

[BA93] Benjamín Callejas Bedregal and Benedito Melo Acióly. *Rational Intervals: An Effectively Given Information System for the real numbers.* In S.M. Markov editor. Scientific Computation and Mathematical Modelling, pages 159-169, Sofia-Bulgaria, DATECS Publishing, 1993.

[BA97] Benjamín Callejas Bedregal and Benedito Melo Acióly. *Computability on the Interval Space: A Domain Approach.* In XXIII Conferencia Latinoamericana de Informática - CLEI, Valparaíso-Chile, November 1997, pages 107-116.

[Bed96] Benjamín R. Callejas Bedregal. *Continuous Information Systems: A Computational and Logical Approach to Interval Mathematics* (in portuguese). PhD thesis, UFPE-Depto. de Informática, Recife, 1996.

[Bra95] Vasco Brattka. *Recursive Characterization of Computable Real-valued Functions and Relations.* Preprint submitted to Elsiever Science, 1995.

[Gia93] Pietro Di Gianantonio. *A Functional Approach to Computability on Real Numbers.* PhD thesis, Università di Pisa-Genova-Udine, Italy, march 1993.

[Grz57] A. Grzegorczyk. *On the definitions of computable real continuous functions.* Fundamenta Mathematicae, 44(1), 1957.

[Gun85] Carl A. Gunter. *Comparing categories of domains.* In LNCS 239, pages 101-121. Springer-Verlag, 1985.

[KT84] Tsutomu Kamimura and Adrian Tang. *Total Objects of Domains.* In Theoretical Computer Science 34, Pages 275-288, 1984.

[Law87] Jimmie D. Lawson. The versatile of Continuous Order. In M.Mislon, M.Main, A. Melton and D.Schmidt, eds., *Mathematical Foundations of Programming Languages Semantics*, New Orleans, 1987. Proceeding 3rd Workshop Tulane University, Springer-Verlag, LNCS 298.

[Lof70] Per Martin-Löf. *Notes on Constructive Mathematics.* Almqvist & Wiksell, Stockholm, 1970.

[Moo66] Ramon E. Moore. *Interval Analysis.* Englewoods Cliffs: Prentice Hall, 1966.

[PR88] Marian Boykan Pour-El and Jonathan Ian Richards. em Computability in Analysis and Physics. Perspectives in Mathematical Logic. Springer Verlag, Berlin, 1988.

[Rog67] Hartley Rogers Jr. *Theory of Recursive Functions and Effectively Computability.* McGraw-Hill Book Company, New York, 1967.

[Sco70] Dana Scott. *Outline of a Mathematical Theory of Computation.* In $4^{\underline{th}}$ annu Princeton Conference on Inf. Science and Systems, pages 65-106, 1970

[Smy77] Myke Smyth. *Effectively Given Domains.* In Theoretical Computer Science, pages 257-274, 1977.

[Smy92] Myke Smyth. *Topology.* In S.Abramsky, D.M. Gabbay and T.S.E. Maimbaum, editors, Handbook of Logic in Computer Science. Vol. 1. Oxford University Press, 1992.

[Tur37] Alan M. Turing. *On Computable Numbers, with applications to the entscheidungsproblem.* Proceeding of the London Mathematical Society, Vol. 42 pages 230-265 and Vol. 43 pages 544-546, 1936-1937.

[Wei95] Klaus Weihrauch. *A simple Introduction to Computable Analysis.* Technical Report 171-2/1995, Fern Universitat, 1995.

[WS83] Klaus Weihrauch and G. SchÄfer. *Admissible representations of effective cpo's.* Theoretical Computer Science, 26:131-147, 1983.