ソフトウェア基礎論　35－1
(1990. 5. 24)

Recursive Functions on a Completed Set of Natural Numbers

Tohru NAOI and Yasuyoshi INAGAKI
School of Engineering, Nagoya University
Furo-cho, Chikusa-ku, Nagoya, JAPAN

Abstract: This paper proposes a new approach to the recursive function theory, which is suitable for dealing with computational models based on term-rewriting. The set of natural numbers is completed to a countable algebraic cpo with *partial* numbers and ∞. *Primitive recursion* is interpreted as a way to define continuous functions on this "non-standard" structure of numerals. It is shown that *recursive functions* can be formalized as functions definable by the constant ∞ and *semi-primitive recursion*, a slightly generalized primitive recursion. Finally, *computability* of those functions is discussed through an order-theoretic semantics of non-overlapping linear term rewriting systems.

完備な自然数集合における帰納関数論

直井 徹　稲垣 康善
名古屋大学 工学部

あらまし: 本論文では、Scott の方法に基づいた、帰納関数論の新しい枠組みを提案する。この枠組みは、項の書換えに基づく計算モデルにおける遅延評価の概念を自然に説明できるという特徴を持つ。

本論文では、まず、「自然数」の構造を「部分的」数と「無限大」とを含む可算な代数的完備順序集合として定式化する。次に、原始帰納法をこの順序集合上の連続関数の記述とみなすことによって、原始帰納法の超準的な解釈を与える。さらに、原始帰納法を準原始帰納法とよばれる形へわずかに拡張し、次のことを導く。すなわち、準原始帰納的な関数定義において定数「無限大」の使用を許すことにより、最小化演算が記述可能である。したがって、ここに帰納的関数の超準的な定式化が得られたことになる。この帰納的関数の計算可能性は、重なりのない線形項書換え系を計算モデルとし、その近似正規形に基づく意味論を考えることで示される。

0. Introduction

As widely known, the recursion theory has been developed on the basis of the following formulation for the semantics of programs:

> *The input-output relation of a program determines a partial function, which is defined for an input if and only if the program terminates for the input.*

It should be noted, however, that this formulation is too restrictive to deal with computational models based on rewriting, such as λ-calculus or term rewriting systems. Under so-called *lazy* evaluation strategy, some non-terminating rewriting processes should be understood as processes of producing an *infinite* object, e.g. an *infinite list*. As far as this strategy is employed, it implies an inconsistency concerns the composition of programs to interpret the output of such a process as undefined (see [1] for the case of λ-calculus). Thus, the conventional theory is forced to prohibit either lazy evaluation, or rewriting schemes such that the choice of strategies can affect the termination of computation; The former is the case of Herbrand-Gödel-Kleene system of equations [5], and the latter is the case of λI-calculus [3]. These settings, however, restrict the peculiar feature of computation by rewriting.

In the present paper, we therefore propose a new approach to the theory of recursive function, which is based on Scott's order-theoretic framework [10] and able to treat lazy evaluation processes easily. Our study is sketched as below.

We first construct a directed-complete partial order $\langle N^\infty, \sqsubseteq \rangle$, as our set of *data*, which includes the set N of natural numbers. N is completed to N^∞ by being added (1) *partial* natural numbers $\bot$, $s(\bot)$, $s(s(\bot))$, ... generated from an *least defined element* $\bot$ by the *successor* function s, and (2) an *infinity* ∞ defined as the *limit* (l.u.b.) of those partial objects. We shall study the computability of *total* functions on the structure N^∞. Note that if an extra axiom $s(\bot) = \bot$ is placed on, then, we get $s^n(\bot) = \bot$ for every n and also have $\infty = \bot$. Consequently, any theory on *total* functions on N^∞ would be just a copy of an theory on *partial* functions on N. Nevertheless, our *lazy* successor function never satisfies $s(\bot) = \bot$, which is a crucial difference between the starting points of ours and of the conventional theory.

Next, we shall define *primitive recursion* on N^∞. Primitive recursive definitions of functions are formulated as syntactical objects, i.e., sets of *equations*. These systems of equations could be interpreted as descriptions of functions on N. However, we interpret them, in a *non-standard* way, as descriptions of continuous functions on N^∞. More accurately, we shall proceed with *semi-primitive recursion* that generalizes primitive recursion; it has the same descriptive power as primitive recursion when it is interpreted on N.

We can obtain a more general class of continuous functions, i.e., the class of *recursive* functions, by allowing a constant symbol *inf*, a representation of ∞, to occur in semi-primitive recursive definitions. We shall see that "*semi-primitive recursion* + ∞" is the counterpart of "*primitive recursion* + *minimalization*" in the conventional theory, by showing that minimalization can be described by the new scheme.

To see that those functions on N^∞ are *computable*, we utilize term rewriting systems (TRSs) as computational devices. A recursive definition is now viewed as a TRS. The input-output relation of the TRS program is defined through an order-

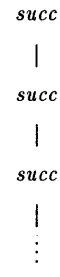

Fig. 1. An infinite tree.

theoretic semantics, which has been investigated in authors' papers [6,7,8]. By the semantics, the output of computation is defined as the *limit* of a (possibly infinite) rewriting sequence of terms; the limit is an infinite tree in general. Hence, nontermination need not involve undefinedness.

To finish this chapter, let us see a few examples that illustrate the notion of the limit of nonterminating rewriting sequence, and the existence of ∞ as the behavior of TRS programs.

Consider a rewrite rule

$$inf \rightarrow succ(inf),$$

where the function symbol *succ* represents the successor function s. This rule causes an infinite rewriting sequence that starts from *inf*:

$$inf \rightarrow succ(inf) \rightarrow succ(succ(inf)) \rightarrow \cdots.$$

The limit of the above sequence is defined as an infinite tree shown in Fig. 1, which will be interpreted as ∞ in N^{∞}. The sequence itself can be interpreted as an increasing sequence of partial numbers:

$$\bot \sqsubseteq s(\bot) \sqsubseteq s(s(\bot)) \sqsubseteq \cdots.$$

The next example suggests how *inf* can be used in our programming. The following three rules computes *less-than-or-equal-to* predicate:

$$\begin{aligned} lteq(zero, x) &\rightarrow true, \\ lteq(succ(x), zero) &\rightarrow false, \\ lteq(succ(x), succ(y)) &\rightarrow lteq(x, y). \end{aligned}$$

With these rules, we obtain a rewriting sequence

$$\begin{aligned} & lteq(succ^n(zero), inf) \\ & \quad \rightarrow lteq(succ^n(zero), succ(inf)) \\ & \quad \rightarrow lteq(succ^{n-1}(zero), inf) \\ & \quad \vdots \\ & \quad \rightarrow lteq(zero, inf) \\ & \quad \rightarrow true, \end{aligned}$$

which claims that $n \leq \infty$ holds for n in N. Also in arithmetic operations, *inf* behaves as we expect. For example, consider the following well-known definition of addition:

$$\begin{aligned} add(zero, x) &\rightarrow x, \\ add(succ(x), y) &\rightarrow succ(add(x, y)). \end{aligned}$$

It will be seen that the limits of the two rewriting sequences start from $add(inf, succ^n(zero))$ and $add(succ^n(zero), inf)$ are both identical with the tree in Fig. 1. In other words, $\infty + n = n + \infty = \infty$ holds for every n.

We shall develop our study of computable functions on this non-standard structure $\langle N^{\infty}, \sqsubseteq \rangle$ of numerals. Although it may seem somewhat curious, it is tightly supported by the behavior of programs.

1. Preliminary Definitions.

1.1. CPOs and Continuous Functions.

We introduce preliminary notions on CPOs and continuous functions, originated from [10]. For more details of the subjects, see e.g., an accurate tutorial in [1].

Let $\langle D, \sqsubseteq \rangle$ and $\langle D', \sqsubseteq' \rangle$ be partial orders. A function $\varphi: D^n \rightarrow D'$ is said to be *monotone* if $d_1 \sqsubseteq e_1, ..., d_n \sqsubseteq e_n$ implies $\varphi(d_1, ..., d_n) \sqsubseteq' \varphi(e_1, ..., e_n)$ for any $d_1, ..., d_n$, and $e_1, ..., e_n \in D$. A

subset S of D is *directed* if it is nonempty and for any $d_1, d_2 \in S$, there exists $d \in S$ such that $d_1 \sqsubseteq d$ and $d_2 \sqsubseteq d$. A *complete partial order* (*CPO*) is a partial order $\langle D, \sqsubseteq \rangle$ such that there is a least element in D, and every directed subset S of D has an l.u.b. denoted by $\sqcup S$.

Let $\langle D, \sqsubseteq \rangle$ and $\langle D', \sqsubseteq' \rangle$ be CPOs. A function $\varphi: D^n \to D'$ is said *continuous* if for any directed subsets $S_1, ..., S_n$ of D, $\varphi(\sqcup S_1, ..., \sqcup S_n) = \sqcup \varphi(S_1, ..., S_n)$ holds. It is known that every continuous function is monotone.

An element d of a CPO $\langle D, \sqsubseteq \rangle$ is *compact* if for every directed subset S of D, $d \sqsubseteq \sqcup S$ implies $d \sqsubseteq d'$ for some d' in S. Let E be the set of all compact elements in D. We say that the CPO $\langle D, \sqsubseteq \rangle$ is *algebraic* (*with base* E) if for any d in D, we get $d = \sqcup \{e | E \ni e \sqsubseteq d\}$.

For a function $\varphi: D \to D'$ and a subset S of D, $\varphi|_D$ denotes the restriction $\varphi \cap (S \times D')$ of φ to S.

1.2. Terms and Infinite Trees.

The notion of infinite trees is introduced informally. A precise treatment of the notion is found in [2].

Let F, B, and X be the set of *unknown function symbols*, *base function symbols*, and *variable symbols*, respectively. These three sets are mutually disjoint, and each symbols are associated with a natural number, called *arity*. The set F is given by $F = \bigcup_n F_n$ where each F_n is a countable set of n-ary symbols, and X is a countable set of nullary symbols.

An $\langle F \cup B, X \rangle$-*tree* is a finite or infinite tree such that its nodes are labeled with symbols in $F \cup B \cup X$, and the number of *sons* of a node is equal to the arity of the label. We denote the set of $\langle F \cup B, X \rangle$-trees by $T^\infty(F \cup B, X)$, and the set of finite $\langle F \cup B, X \rangle$-trees by $T(F \cup B, X)$. We shall write $T^\infty(F \cup B)$ for $T^\infty(F \cup B, \phi)$, and $T(F \cup B)$ for $T(F \cup B, \phi)$. A finite $\langle F \cup B, X \rangle$-tree is identified with a *well-formed term*.

For a tree T, $\mathrm{Var}(T)$ denotes the set of variable symbols occur in a tree T as labels. We denote the set of nodes of T by $\mathrm{Node}(T)$. For a tree T, we define an order $\leq_T$ on $\mathrm{Node}(T)$ by: for all p and q in $\mathrm{Node}(T)$, $p \leq_T q$ iff p is an *ancestor* of q. For a subset P of $\mathrm{Node}(T)$, we write $\min_T(P)$ for the set of minimal nodes in P under $\leq_T$. For p in $\mathrm{Node}(T)$, $T(p)$ and $T[p \leftarrow T']$ denote the label of p and the tree one obtain by replacing the subtree of T with root p by T', respectively. Finally, $T[T_1/x_1, ..., T_n/x_n]$ is the tree such that for all i, each x_i in T has been substituted by T_i, simultaneously.

We now define the denotation of a term as a continuous function on a CPO $\langle D, \sqsubseteq \rangle$. For simplicity, we shall treat elements in D as nullary functions on D. For a subset G of $F \cup B$, an *interpretation of* G (*on* D) is a map $I: G \to \bigcup_n \{D^n \to D\}$ such that for any g in G with arity n, $I(g)$ is a continuous function from D^n to D. We often write g_I for $I(g)$. Let I be an interpretation of G. For any distinct variable symbols $x_1, ..., x_n$ such that $\mathrm{Var}(t) \subseteq \{x_1, ..., x_n\}$, a function $t_I^{x_1, ..., x_n}: D^n \to D$ is defined inductively by:

(1) If $t = x_i$ for some i, then, $t_I^{x_1, ..., x_n}(d_1, ..., d_n) = d_i$ for any $d_1, ..., d_n$.

(2) If $t = g(t_1, ..., t_m)$ for some g and $t_1, ..., t_m$, $t_I^{x_1, ..., x_n}(d_1, ..., d_n) = g_I(c_1, ..., c_m)$ where we let $c_i = (t_i)_I^{x_1, ..., x_n}(d_1, ..., d_n)$ for each i.

Proposition 1.2.2. *For any interpretation* I *of* G *and any* t *in* $T(G, \{x_1, ..., x_n\})$, *the function* $t_I^{x_1, ..., x_n}$ *is continuous.* □

2. Recursive Functions.

2.1. Completion of Natural Numbers.

We now construct the ordered set $\langle N^\infty, \sqsubseteq \rangle$ mentioned in Introduction.

Let N be the set of natural numbers, and let

$$N^\infty = \{\{n\} | \ n \in N\} \cup \{\{k | \ n \leq k \in N\} | \ n \in N\} \cup \{\phi\}.$$

We define an order $\sqsubseteq$ on N^∞ by: $d \sqsubseteq d'$ iff $d \supseteq d'$ for d and d' in N^∞. We shall identify a singleton $\{n\}$ with n and denote a set $\{k | \ n \leq k\}$ by $n\uparrow$; the latter is called a *partial* natural number, which can be understood as a possible output from a program *running*. The set $\{n\uparrow \mid n \in N\}$ of all partial natural numbers is denoted by $N\uparrow$. We also write $\perp$ for $0\uparrow$ and ∞ for ϕ. Under the convention, we can write $N^\infty = N \cup N\uparrow \cup \{\infty\}$ and draw Fig. 2 to illustrate the order. Note that $n \leq m$ iff $n\uparrow \sqsubseteq m$ iff $n\uparrow \sqsubseteq m\uparrow$, and $\infty = \sqcup\{n\uparrow \mid n\uparrow \in N\uparrow\}$.

Theorem 2.1.1. *$\langle N^\infty, \sqsubseteq \rangle$ is an algebraic CPO with base $N \cup N\uparrow$ and a least element $\perp$.* □

Lemma 2.1.2. *Let $\varphi : N^\infty \to N^\infty$ be a function such that $\varphi|_{N \cup N\uparrow}$ is monotone. Then, φ is continuous iff $\varphi(\infty) = \sqcup\{\varphi(n\uparrow) | \ n\uparrow \in N\uparrow\}$.* □

There are two properties on functions on N^∞ to be defined. We say that an n-ary function φ is *stable* if $\varphi(N, ..., N) \subseteq N$. This notion corresponds to the totalness of functions in the conventional theory. An n-ary function φ is *strict (w.r.t. the i-th argument)* if $\varphi(d_1, ..., d_{i-1}, \perp, d_{i+1}, , ..., d_n) = \perp$ holds for any $d_1, ..., d_{i-1}, d_{i+1}, ..., d_n$ in N^∞.

Let us define the *successor* function s on N^∞ as follows: $s(n) = n + 1$ and $s(n\uparrow) = (n+1)\uparrow$ for all n in N, and $s(\infty) = \infty$. Note that N^∞ is partitioned into two well-founded s-chains N and $N\uparrow$, and a singleton $\{\infty\}$. Clearly, ∞ is a unique fixedpoint of s. It is the successor of itself and can not be the successor of any other elements. The function s plays an essential role together with $\sqsubseteq$.

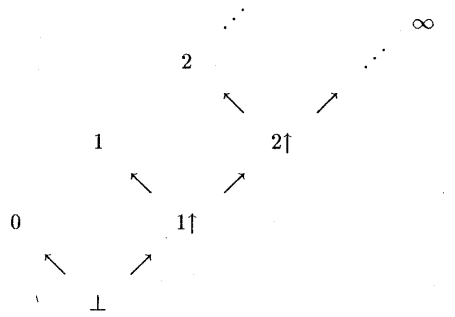

Fig. 2. An CPO $\langle N^\infty, \sqsubseteq \rangle$.

Theorem 2.1.3. *The successor function s is stable, continuous, and not strict.* □

2.2. Semi-Primitive Recursion.

We shall now formulate *semi-primitive recursion*, together with *primitive recursion*, in a more formal way than the conventional framework; it will be interpreted as a method to define continuous functions on N^∞.

An *equation (on $T(F \cup B, X)$)* is a pair $\langle t, u \rangle$ of terms in $T(F \cup B, X)$, which will be written as $t = u$. For a set E of equations, we denote, by $\mathrm{Fun}(E)$, the set of unknown or base function symbols that occur in equations of E. Let I be an interpretation of a set G of function symbols such that $G \supseteq \mathrm{Fun}(E)$. We say that I *satisfies* E if for any $t = u$ in E, $t_I^{x_1,...,x_n} = u_I^{x_1,...,x_n}$ holds with $\{x_1, ..., x_n\} = \mathrm{Var}(t) \cup \mathrm{Var}(u)$.

From now on, we suppose that the set B consists of *zero* and *succ* with arities 0 and 1, respectively. We denote, by Δ_B, the *standard interpretation of B* such that *zero* $\mapsto 0$ and *succ* $\mapsto s$.

Definition 2.2.1. *A semi-primitive recursive definition (SPRD) is a set of equations on $T(F \cup B, X)$ defined inductively as:*

(1) An empty set is an SPRD.

(2) Let E_0 be an SPRD and $G = \mathrm{Fun}(E_0) \cup B$. For a set E_1 of equations, $E_0 \cup E_1$ is an SPRD iff for some function symbol $f \in F - \mathrm{Fun}(E_0)$ and mutually distinct variable symbols $x_1, ..., x_n$, one of the following conditions holds:

(2a) for some t in $T(G, \{x_1, ..., x_n\})$,

$$E_1 = \{f(x_1, ..., x_n) = t\},$$

(2b) for some $t, t_2, ..., t_n$ in $T(G, \{x_2, ..., x_n\})$ and u in $T(G, \{y, x_1, ..., x_n\})$,

$$E_1 = \{f(zero, x_2, ..., x_n) = t,$$
$$f(succ(x_1), x_2, ..., x_n) \quad (*)$$
$$= u[f(x_1, t_2, ..., t_n)/y]\}.$$

A primitive recursive definition (PRD) is an SPRD such that whenever the case (2b) is applied, $t_2 = x_2, ..., t_n = x_n$ are satisfied. □

The next lemma claims, roughly, that the system of equations in the form $(*)$ uniquely determines a continuous function represented by f whenever $t, t_2, ..., t_n$ and u represent continuous functions.

Lemma 2.2.2. *Let E_1 be a set of equations given by $(*)$, H be any set of function symbols such that $f \notin H$ and $u, t, t_1, ..., t_n \in T(H, X)$, and I_0 be any interpretation of H on N^∞. Then, an interpretation I of $H \cup \{f\}$ that satisfies the following conditions uniquely exists.*

(1) I extends I_0 and satisfies E_1, and,

(2) $f_I(\perp, d_2, ..., d_n) = \perp$ holds for any $d_2, ..., d_n$ in N^∞. □

This lemma can be shown by induction on N and on $N\uparrow$, and Tarski's fixedpoint theorem. It help us to formulate an interpretation *specified* by an SPRD:

Definition 2.2.3. *For an SPRD E, the denotational interpretation of E, denoted by Δ_E, is an interpretation of $\mathrm{Fun}(E) \cup B$ defined inductively by:*

(1) $\Delta_E = \Delta_B$ if $E = \phi$.

(2) Otherwise, there are a partition $\{E_0, E_1\}$ of E such that E_0 is an SPRD, and $f \in F - \mathrm{Fun}(E_0)$. There are two subcases:

(2a) If $E_1 = \{f(x_1, ..., x_n) = t\}$, then, $\Delta_E = \Delta_{E_0} \cup \{f \mapsto t_{\Delta_{E_0}}^{x_1, ..., x_n}\}$,

(2b) If E_1 is of the form $(*)$, then, Δ_E is I in Lemma 2.2.2 with $H = \mathrm{Fun}(E_0) \cup B$ and $I_0 = \Delta_{E_0}$.

□

Theorem 2.2.4. *For any SPRD E, the denotational interpretation Δ_E satisfies E.* □

Definition 2.2.5. *An n-ary function on N^∞ is semi-primitive (resp. primitive) recursive if it is in the range of Δ_E for an SPRD (resp. PRD) E.* □

Theorem 2.2.6. *Every semi-primitive recursive function is continuous.* □

It can also be shown that:

Theorem 2.2.7. *Every semi-primitive recursive function is stable.* □

About the relation between semi-primitive and primitive recursion, we can show that they have the same descriptive power as far as we compare them from the conventional point of view. That is,

Theorem 2.2.8. *For any n-ary semi-primitive recursive function σ, there is an n-ary primitive recursive function π such that $\sigma \cap (N^n \times N) = \pi \cap (N^n \times N)$.* □

Nevertheless, we conjecture that: *The class of*

semi-primitive recursive functions properly includes the class of primitive recursive functions.

The reason to treat semi-primitive recursion will be seen later in this section and the next section.

Let us see some important examples of semi-primitive recursive functions. In those examples, we shall not name an SPRD and write merely f_Δ for a function symbol f, instead of e.g. f_{Δ_E}; each function symbol will be understood as a representation of a particular function on N^∞.

Projection. The *k-th projection* u_k^n *from* $(N^\infty)^n$ is simply defined by

$$u_k^n(x_1, ..., x_k, ..., x_n) = x_k.$$

This projection function is not strict (w.r.t. any argument except the k-th). One who needs a strict "projection", say $\tilde{u}_2^2$, should define it by primitive recursion such as

$$\begin{aligned} \tilde{u}_2^2(zero, x_2) &= x_2, \\ \tilde{u}_2^2(succ(x_1), x_2) &= \tilde{u}_2^2(x_1, x_2). \end{aligned}$$

From Lemma 2.2.2, we have $(\tilde{u}_2^2)_\Delta(\perp, d) = \perp$ for any $d \in N^\infty$. Then, by induction on $N\uparrow$ and Theorem 2.2.4, we obtain $(\tilde{u}_2^2)_\Delta(n\uparrow, d) = \perp$ for any $n\uparrow \in N\uparrow$. Finally, by the continuity and Lemma 2.1.2, we get $(\tilde{u}_2^2)_\Delta(\infty, d) = \perp$. †

Degeneration. The *degeneration* function defined below maps every $n \in N$ to n and every $d \notin N$ to $\perp$. This function will *project* our results on the functions on N^∞ to the conventional world.

$$degen(x) = \tilde{u}_2^2(x, x).$$

Conditional. The following defines the *conditional* function with *true* represented by 0 and *false* represented by any $d \in N^\infty - \{0, \perp\}$.

$$\begin{aligned} if(zero, x_1, x_2) &= x_1, \\ if(succ(x), x_1, x_2) &= x_2. \end{aligned}$$

Bounded minimalization. Let π and β be $(k+1)$-ary functions on N^∞ and write $\vec{n}$ for a k-tuple $\langle n_1, ..., n_k \rangle$ in N^k. We say β is a *bounded minimalization with respect to* π if for any $m \in N$ and $\vec{n}$ in N^k,

$$\beta(m, \vec{n}) = \begin{cases} \min_{\leq}\{i \in N \mid \pi(i, \vec{n}) = 0\} \\ \qquad \text{if } \exists i \leq m \ \ \pi(i, \vec{n}) = 0, \\ m+1 \quad \text{otherwise.} \end{cases}$$

Lemma 2.2.9. *For any semi-primitive recursive function π, there is a bounded minimalization w.r.t. π which is semi-primitive recursive. If π is represented by a function symbol p, then the following two SPRDs define the bounded minimalizations represented by symbols $\bar{b}_p$ and b_p:*

$$\begin{aligned} \bar{b}_p^*(zero, \vec{x}, y) &= if(p(y, \vec{x}), y, succ(y)), \\ \bar{b}_p^*(succ(x), \vec{x}, y) &= if(p(y, \vec{x}), y, \\ &\qquad \bar{b}_p^*(x, \vec{x}, succ(y)))), \\ \bar{b}_p(x, \vec{x}) &= \bar{b}_p^*(x, \vec{x}, zero), \end{aligned}$$

and

$$\begin{aligned} b_p^*(zero, \vec{x}, y) &= if(p(y, \vec{x}), zero, succ(zero)), \\ b_p^*(succ(x), \vec{x}, y) &= if(p(y, \vec{x}), zero, \\ &\qquad succ(b_p^*(x, \vec{x}, succ(y)))), \\ b_p(x, \vec{x}) &= b_p^*(x, \vec{x}, zero). \end{aligned}$$

□

As seen below, $(\bar{b}_p)_\Delta$ and $(b_p)_\Delta$ are not identical in general. Suppose that there is $\vec{d}$ such that $\pi(m, \vec{d})$ is in $N^\infty - \{0, \perp\}$ for all m. Then, for any $n\uparrow$, $(b_p)_\Delta(n\uparrow, \vec{d}) = (n+1)\uparrow$ although $(\bar{b}_p)_\Delta(n\uparrow, \vec{d}) = \perp$. Moreover, we have $(b_p)_\Delta(\infty, \vec{d}) = \infty$ and $(\bar{b}_p)_\Delta(\infty, \vec{d}) = \perp$ for the $\vec{d}$.

† Strictly Speaking, $\tilde{u}_2^2$ is not a projection.

Note that the above definitions are not PRDs but SPRDs. Although we can define bounded minimalization by PRDs, we shall need the above definitions in the next section.

2.3. Minimalization and Recursion.

In this section, we extend the class of functions on N^∞ dealt with, to obtain the general class of *computable* functions. In the conventional theory, one introduces the operation of *minimalization* for the purpose. In our theory, however, we do not require it since we can describe it by semi-primitive recursion with a constant ∞.

To the end, we first extend our equational language by adding a new constant symbol (nullary function symbol) *inf* to the set B of base function symbols. Let $B_{inf} = B \cup \{inf\}$; the *standard interpretation* $\Delta_{B_{inf}}$ of B_{inf} extends Δ_B and maps *inf* to ∞.

Definition 2.3.1. *A recursive definition (RD) is a set E of equations on $T(F \cup B_{inf}, X)$ defined in the same way as SPRDs. For an RD E, the denotational interpretation Δ_E of E is also defined similarly. An n-ary function on N^∞ is said to be recursive if it belongs to the range of Δ_E for some RD E.* □

Theorem 2.3.2. *For any RD E, the interpretation Δ_E satisfies E.* □

Theorem 2.3.3. *Every recursive function is continuous.* □

There exist recursive functions which are not semi-primitive recursive, because non-stable functions can be defined by RDs. The following RD defines a representation *undef* of $\perp$:

$$undef = degen(inf).$$

Let π be a $(k+1)$-ary function on N^∞. A *minimalization with respect to* π is a k-ary function μ that satisfies: For any $\vec{n} \in N^k$ such that $\pi(m, \vec{n}) = 0$ holds for some $m \in N$,

$$\mu(\vec{n}) = \min_{\leq}\{m \in N |\ \pi(m, \vec{n}) = 0\}.$$

Theorem 2.3.4. *For any recursive function π, there is a recursive minimalization w.r.t π. Both of the following RDs give it if π is represented by p:*

$$\bar{m}_p(x_1, ..., x_k) = \bar{b}_p(\infty, x_1, ..., x_k),$$

and

$$m_p(x_1, ..., x_k) = b_p(\infty, x_1, ..., x_k).$$

□

Hence our class of recursive function is closed under minimalization, and clearly it contains every primitive recursive functions. Although it is still not known if this class of functions is the least one among such classes, the result in the next chapter suggests the positive answer.

3. Computability of Recursive Functions.

3.1. Term Rewriting Systems.

We briefly introduce some definitions and notations on term rewriting systems. See e.g. [4] for more details.

A *rewrite rule* is a pair $\langle t, u \rangle$ of terms in $T(F \cup B_{inf}, X)$ such that $t \notin X$ and $Var(t) \supseteq Var(u)$ holds. We write $t \to u$ for a rewrite rule $\langle t, u \rangle$. A *term rewriting system* (TRS) R is a set of rewrite rules. A term s is called a *redex of* R if for some $t \to u$ in R, t matches s. We denote the *reduction relation in* R (on $T(F \cup B_{inf}, X)$) by $\to_R$, and the reflexive-transitive closure of $\to_R$ by $\to_R^*$.

A TRS R is said *left-linear* if for each $t \to u$ in R, no variable symbol occurs in t twice or more. R is *non-overlapping* (*non-ambiguous*) if for every $t \to u$ and $t' \to u'$ in R, t does not unify with any non-variable subterm s of t' except the trivial case that $\langle t \to u \rangle = \langle t' \to u' \rangle$ and $t' = s$.

3.2. Algebraic Semantics.

To see that recursive functions are *computable*, we first introduce the algebraic semantics of left-linear and non-overlapping TRSs [6,7,8]. It will appear in a much simpler form than the literature since RDs have a quite restricted form as a TRS.

Let C be a set of function symbols. A TRS R is called a *constructor system on* C if for all $t \to u$ in R, $t = f(t_1, ..., t_n)$ is satisfied for some $f \notin C$ and $t_1, ..., t_n \in T(C, X)$. A constructor system R is *perfect* if for every f in $\mathrm{Fun}(R) - C$ and every $t_1, ..., t_n$ in $T(C)$, $f(t_1, ..., t_n)$ is a redex of R.

Lemma 3.2.1. *For any RD E, E_{inf} is a left-linear and non-overlapping TRS. Moreover, it is a perfect constructor system on B.* □

In what follows, we let E be an RD and $E_{inf} = E \cup \{inf \to succ(inf)\}$. To formalize the computational semantics of E using a TRS E_{inf}, we again extend the set of base function symbols. Let Ω be a new constant symbol, $B_\Omega = B \cup \{\Omega\}$ and $B_{\Omega,inf} = B \cup \{\Omega, inf\}$. Let us define an order $\preceq$ on $T^\infty(F \cup B_{\Omega,inf})$ by: $T \preceq U$ iff for some $P \subseteq \mathrm{Node}(U)$, $T = U[p \leftarrow \Omega |\ p \in \min_U(P)]$. This ordering is due to [9].

Proposition 3.2.2. *$\langle T^\infty(F \cup B_{\Omega,inf}), \preceq \rangle$ and $\langle T^\infty(B_\Omega), \preceq \rangle$ are algebraic CPOs with bases $T(F \cup B_{\Omega,inf})$ and $T(B_\Omega)$, respectively. In both cases, the least element is Ω.* □

Define a function $\omega_{E_{inf}}$ from $T(F \cup B_{\Omega,inf})$ to $T(B_\Omega)$ as below: For each $t \in T(F \cup B_{\Omega,inf})$,

$$\omega_{E_{inf}}(t) = t[p \leftarrow \Omega |\ p \in \min_t(P)],$$

where $P = \{p \in \mathrm{Node}(t) |\ t(p) \in F \cup \{inf\}\}$.† If $t \to^*_{E_{inf}} u$, then, we call $\omega_{E_{inf}}(u)$ an *approximate normal form of* t.

Next, we define the *symbolic value of t in E_{inf}*, denoted by $\mathrm{Val}_{E_{inf}}(t)$, by:

$$\mathrm{Val}_{E_{inf}}(t) = \sqcup\{\omega_{E_{inf}}(u) |\ t \to^*_{E_{inf}} u\}.$$

We can show that $\mathrm{Val}_{E_{inf}}$ is a total monotone function from $T(F \cup B_{inf,\Omega})$ to $T^\infty(B_\Omega)$. This function defines the *evaluator* for a program, i.e., a term in $T(F \cup B_{inf,\Omega})$. Approximate normal forms are *partial* outputs from the evaluator running and their l.u.b. is the *eventual* output as the limit.

3.3. Computational Interpretation of RDs.

Using the function $\mathrm{Val}_{E_{inf}}$, we formulate the functions on N^∞ *computed* with an RD E.

The CPO $\langle T^\infty(B_\Omega), \preceq \rangle$ is considered as the *output domain* of our evaluator. It is easy to see that the CPO is isomorphic to $\langle N^\infty, \sqsubseteq \rangle$; We let γ_o be the unique continuous bijection from $\langle N^\infty, \sqsubseteq \rangle$ to $\langle T^\infty(B_\Omega), \preceq \rangle$, which is a B_Ω-isomorphism in the sense of [2,8].

Next, we shall construct the *input domain*. Remark that $\mathrm{Val}_{E_{inf}}(T(B_\Omega) \cup \{inf\}) = T^\infty(B_\Omega)$ holds for every RD E. Let us define a preorder $\lesssim_{E_{inf}}$ on $T(B_\Omega) \cup \{inf\}$ by: $t \lesssim_{E_{inf}} u$ iff $\mathrm{Val}_{E_{inf}}(t) \preceq \mathrm{Val}_{E_{inf}}(u)$. Then, it can be easily shown that $\langle T(B_\Omega) \cup \{inf\}, \lesssim_{E_{inf}} \rangle$ is a CPO isomorphic to $\langle N^\infty, \sqsubseteq \rangle$; We denote, by γ_i, the unique continuous bijection from $\langle N^\infty, \sqsubseteq \rangle$ to $\langle T(B_\Omega) \cup \{inf\}, \lesssim_{E_{inf}} \rangle$.

† It can be shown that $\omega_{E_{inf}}$ is exactly one appeared in [6,7,8] using the fact that E_{inf} is a perfect constructor system and t is in $T(F \cup B_{\Omega,inf})$.

Definition 3.3.1. *For an RD E, the computational interpretation Γ_E of E is an interpretation of* $\mathrm{Fun}(E) \cup B_{inf}$ *defined as follows: For* $f \in \mathrm{Fun}(E) \cup B_{inf}$ *with arity* n,

$$f_{\Gamma_E}(d_1, ..., d_n) = \gamma_o^{-1}(\mathrm{Val}_{E_{inf}}(f(\gamma_i(d_1), ..., \gamma_i(d_n)))),$$

for $d_1, ..., d_n$ *in* N^∞. □

Theorem 3.3.2. *For any RD* E, $\Gamma_E = \Delta_E$. □

4. Concluding Remarks.

From the Theorem 3.3.2, we can conclude that recursive functions are computable; Leaving the term "computable" to mean *intuitively effective*, we shall verify the above. An immediate consequence of the theorem is that a recursive functions is computable if the corresponding $\mathrm{Val}_{E_{inf}}|_{T(F \cup B_{inf}, \Omega)}$ is. The latter would be, however, revealed true from the following observation. First, one-step rewriting and $\omega_{E_{inf}}$ is clearly effective since E is a finite set. Second, there is a rewriting strategy (i.e., an effective and deterministic restriction of the relation $\rightarrow_{E_{inf}}$) that prevents non-terminating enumeration of $\rightarrow^*_{E_{inf}}$ by the evaluator approaching the l.u.b. of $\{\omega_{E_{inf}}(u) |\ t \rightarrow^*_{E_{inf}} u\}$ (see [8]); it is a kind of lazy evaluation methods.

Acknowlidgements.

The authors wish to thank Prof. Namio HONDA and Prof. Teruo FUKUMURA for their encouragement, Prof. Toshiki SAKABE and Prof. Tomio HIRATA for their helpful discussions, and Dr. Ken MANO, Dr. Mizuhito OGAWA and Dr. Satoshi ONO for their stimulative comments and help to survey related work. Their work is partly supported by the Grants from Ministry of the Education, Science and Culture of Japan (Scientific Research on Priority Areas No. 63633008, Grant-in-Aid for Encouragement of Young Scientists (A) No. 01750329)

Referrences.

[1] H.P. Barendregt, "The lambda Calculus - Its Syntax and Semantics," 2nd ed., North-Holland (1984).

[2] B. Courcelle, "Fundamental Properties of Infinite Trees," Theoretical Computer Science 25, pp. 95-169 (1983).

[3] A. Church, "The Calculi of Lambda Conversion," Annals of Math. 6, Princeton University Press (1941).

[4] G. Huet, "Confluent Reductions: Abstract Properties and Applications to Term Rewriting Systems," J. ACM 27, pp. 797-821 (1980).

[5] S. Kleene, "General Recursive Functions of Natural Numbers," Mathematische Annalen 112, pp. 727-742 (1936)

[6] T. Naoi and Y. Inagaki, "The Conservative Extension and Behavioral Semantics of Term Rewriting Systems," Technical Research Report No.8604, Department of Information Science, Nagoya University (1986).

[7] T. Naoi and Y. Inagaki, "The Relation between Algebraic and Fixedpoint Semantics of Term Rewriting Systems," Technical Report COMP86-37, IEICE (1986).

[8] T. Naoi and Y. Inagaki, "Algebraic Semantics and Complexity of Term Rewriting Systems," Proc. of 3rd RTA89, Lecture Notes in Computer Science 355, Springer, pp. 312-325 (1989).

[9] M. Nivat, "On the Interpretation of Recursive Polyadic Program Schemes," Symposia Mathematica 15, Rome, pp. 255-281(1975).

[10] D. Scott, "Continuous Lattices," in: F. Lawvere, ed., "Toposes, Algebraic Geometry and Logic," Lecture Notes in Mathematics 274, Springer, pp. 97-136(1972).