数理解析研究所講究録
918巻 1995年 224-237

Non-Standard Recursion Theory

Tohru NAOI
Faculty of Engineering, Gifu University
Yanagido, Gifu-shi, JAPAN
email: naoi@info.gifu-u.ac.jp

Abstract. This paper investigates 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 into a countable algebraic cpo by adding *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 *quasi-primitive recursion*, a slightly generalized version of primitive recursion. Also, *computability* of those functions is discussed through an order-theoretic semantics of term rewriting systems. Finally, the relation between quasi-primitive and primitive recursion is investigated.

1 Introduction

As shown by Herbrand, Gödel and Kleene [5], partial recursive functions can be defined by first-order equations. Once mathematical objects are described in such a formal way, the description itself generally admits non-standard interpretations. Namely, equations which are originally intended to define functions over the set N of natural numbers can be viewed as a description of functions over a set other than N. A non-standard interpretation of this kind has been proposed and studied by the authors' previous paper [9]. The present paper describes some new results on this framework.

An advantage of the new interpretation is that it allows the ordinary inference by equational axioms. For example, suppose that the conditional function if is defined as $if(0,x,y) = x$ and $if(s(x),y,z) = z$ by primitive recursion. Then, an equation $if(0,t_1,t_2) = t_1$ derived from the first axiom is valid for any terms t_1 and t_2 in our interpretation. On the other hand, in the conventional interpretation, it is valid only if the value of t_2 is *defined.* This inconvenience arises because one interprets if as a *partial* function. To compute these partial functions, we are forced to adopt *eager* evaluation strategies. That is, before we reduce $if(0,t_1,t_2)$ to t_1, we should evaluate the subterm t_2 to assure t_2 being defined. This does not seem to be a natural policy to computation.

Scott's mathematical tools for program semantics [11] enables us to formalize computable numerical functions without referring to evaluation or inference strategies. Our new interpretation, mainly based on Scott's approach,

claims that recursive functions described by equations are not partial functions over N but *continuous* functions over a superset of N, a complete partial order named N_∞. This framework generalizes the conventional one because partial functions can be treated as a special case of continuous functions.

The complete partial order N_∞ consists of *total* natural numbers 0, $s(0)$, $s(s(0))$, ..., *partial* natural numbers $\bot$, $s(\bot)$, $s(s(\bot))$, ..., and *infinity* ∞, where the least element $\bot$ denotes the state that a program does not terminate and *produces no outputs*.

Partial natural numbers appear since we obey a premise $\bot \neq s(\bot)$. These two elements should be distinguished with each other because $\bot$ carries entirely no information but $s(\bot)$ at least gives information that it is not equal to 0. In fact, the *sign* function defined by $sign(0) = 0$ and $sign(s(x)) = s(0)$ detects the difference between $\bot$ and $s(\bot)$ as $sign(\bot) = \bot$ but $sign(s(\bot)) = s(0)$, where the latter is obtained from the second axiom on *sign*.[1] Those partial numbers require the existence of ∞ for the order N_∞ to be complete: The infinity ∞ is the unique (and least) upper bound of the chain $\bot \sqsubseteq s(\bot) \sqsubseteq s(s(\bot)) \sqsubseteq \cdots$, and the unique (and least) fixed point of s.

We remark that this order structure is more complicated than the *flat* lattice discussed in Scott [11]: One can obtain a similar flat order from N_∞ by forcing $\bot = s(\bot)$ since it implies $\bot = s(\bot) = s(s(\bot)) = \cdots = \infty$. However, such a flat order is not suitable for our purpose since $\bot = s(\bot)$ also implies $\bot = sign(\bot) = sign(s(\bot)) = s(0)$, an infeasible equality.

Recall that bounded-minimization can be defined with primitive recursion. It is natural to conjecture that bounded-minimization works as minimization when ∞ is used as the bound of search. To prove this, we need to consider a slightly generalized version of primitive recursion, *quasi-primitive recursion*. It will also be shown that the descriptive powers of quasi-primitive and primitive recursions are the same in the conventional interpretation, while they differ in our interpretation.

Now, the following summarizes the contents of the present paper. Section 2 introduces basic definitions. In Section 3, we shall first illustrate the construction of N_∞ and several properties on continuous functions over this set. Then, quasi-primitive recursive functions are formalized and studied. Finally, the notion of recursive functions are defined in the way discussed above. In Section 4, we shall study the computational aspect of recursive functions using the notion of infinite term-rewriting. In Section 5, the descriptive powers of quasi-primitive and primitive recursions are compared.

[1] Another equality $sign(\bot) = \bot$ always holds in our interpretation.

2 Preliminary Definitions

2.1 CPOs and Continuous Functions

We introduce preliminary notions on CPOs and continuous functions. For more details of these subjects, see e.g., a tutorial in [1].

Let $\langle D, \sqsubseteq \rangle$ be a partial order. An n-ary function φ on D is said to be *monotone* if $c_1 \sqsubseteq d_1, \ldots, c_n \sqsubseteq d_n$ implies $\varphi(c_1, \ldots, c_n) \sqsubseteq \varphi(d_1, \ldots, d_n)$ for any $c_1, \ldots, c_n$ and $d_1, \ldots, d_n \in D$. A subset S of D is *directed* if it is nonempty and for any d_1 and $d_2 \in S$, 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 its l.u.b. denoted by $\sqcup S$.

Let $\langle D, \sqsubseteq \rangle$ be a CPO. A function φ on D is said *continuous* if for any directed subsets $S_1, \ldots, S_n$ of D, $\varphi(\sqcup S_1, \ldots, \sqcup S_n) = \sqcup \varphi(S_1, \ldots, S_n)$ holds. It is known that every continuous function is monotone and that continuous functions are closed under composition.

An element c of a CPO $\langle D, \sqsubseteq \rangle$ is *compact* if for every directed subset S of D, $c \sqsubseteq \sqcup S$ implies $c \sqsubseteq s$ for some $s \in S$. Let C be the set of all compact elements in D. We say that the CPO $\langle D, \sqsubseteq \rangle$ is an *algebraic CPO (with base C)* if for any d in D, $\{c \mid C \ni c \sqsubseteq d\}$ is directed and $d = \sqcup\{c \mid C \ni c \sqsubseteq d\}$ holds.

2.2 Terms and Infinite Trees

In this section we informally introduce the notion of infinite trees. A precise treatment of the notion is found in, e.g., [2].

Let F, B, and X be the set of *unknown function symbols*, *base function symbols*, and *variable symbols*, respectively. These three sets are countable and mutually disjoint. Each symbols are associated with a natural number called *arity*, and a variable symbol always has arity 0.

An $\langle F \cup B, X \rangle$-*tree* is a finite or infinite tree such that every node is labeled with a symbol in $F \cup B \cup X$, and for any node, the number of the *children* nodes is equal to the arity of the label on the node. 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 $T(F \cup B, X)$. A finite $\langle F \cup B, X \rangle$-tree is identified with a *well-formed term.*

Let T be a tree. We write $\mathrm{Node}(T)$ for the set of nodes of T. For p in $\mathrm{Node}(T)$, $T(p)$ is the label of p and $T[p \leftarrow T']$ denotes the tree obtained by replacing the subtree of T with root p by another tree T'. We define an order $\leq$ on $\mathrm{Node}(T)$ as follows: For all p and q in $\mathrm{Node}(T)$, $p \leq q$ iff p is an *ancestor* of q. For a subset P of $\mathrm{Node}(T)$, $\min(P)$ denotes the set of minimal nodes in P w.r.t. $\leq$ above.

For a tree T, $\mathrm{Var}(T)$ denotes the set of variable symbols occur in T as labels.

We now define the denotation of a term as a continuous function on a CPO $\langle D, \sqsubseteq \rangle$. 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 an n-ary function on D. We often write g_I for $I(g)$. A *continuous* interpretation I is an interpretation such that every $I(g)$ is continuous.

Let I be an interpretation of G and t a term in $T(G, X)$. For 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 for $d_1, ..., d_n$ in D,

$$t_I^{x_1,...,x_n}(d_1, ..., d_n) = d_i.$$

2. If $t = g(t_1, ..., t_m)$ for some g and some $t_1, ..., t_m$,

$$t_I^{x_1,...,x_n}(d_1, ..., d_n) = g_I(c_1, ..., c_m)$$

where $c_i = (t_i)_I^{x_1,...,x_n}(d_1, ..., d_n)$ for each i.

Since continuous functions are closed under composition, we get:

Proposition 2.2.1 *(Nivat [10]). For any continuous interpretation I of G and any term t in $T(G, \{x_1, ..., x_n\})$, a function $t_I^{x_1,...,x_n}$ is continuous.*

3 Recursive Functions

3.1 Completion of Natural Numbers

We shall now construct the ordered set $\langle N_\infty, \sqsubseteq \rangle$ mentioned in Section 1.

Let N be the set of natural numbers, and let

$$\begin{aligned} N_\infty \;=\; & \big\{\{n\} \mid n \in N\big\} \\ & \cup \big\{\{k \mid n \leq k \in N\} \mid n \in N\big\} \\ & \cup \{\emptyset\}. \end{aligned}$$

We define an order $\sqsubseteq$ on N_∞ by: $d \sqsubseteq d'$ iff $d \supseteq d'$ for d and d' in N_∞. A singleton $\{n\}$ will be identified with n and a set $\{k \mid n \leq k\}$ is denoted by $n\uparrow$; the latter is called a *partial* natural number. The set $\{n\uparrow \mid n \in N\}$ of all partial natural numbers is written as $N\uparrow$. We also write $\perp$ for $0\uparrow$ and ∞ for $\emptyset$. Under these conventions, we can write

$$N_\infty = N \cup N\uparrow \cup \{\infty\}$$

and draw Fig. 1 to illustrate the order.

Note that $n \leq m$ iff $n\uparrow \sqsubseteq m$ iff $n\uparrow \sqsubseteq m\uparrow$, and $\infty = \sqcup\{n\uparrow \mid n\uparrow \in N\uparrow\}$.

The construction of N_∞ is intuitively explained as follows. Consider a loop program to search a number x that satisfies a certain condition and

suppose that at the n-th iteration, the program tests n as a candidate of x. If the test for n is successful, an answer that $x \in \{n\}$, or $x = n$ is obtained. On the other hand, the state that the test has just failed for $n-1$ is represented by $n\uparrow = \{k \mid n \leq k\}$, which claims that if the number x exists, then x must be in $\{k \mid n \leq k\}$, or *at this moment* $x = n\uparrow$ holds in N_∞. The least element $\bot = 0\uparrow$, which gives no information about x, corresponds to the initial state that the search starts, and all the states of this successful execution will form an increasing chain $0\uparrow \sqsubseteq 1\uparrow \sqsubseteq \cdots \sqsubseteq n\uparrow \sqsubseteq n$. Contrastively, if there are no numbers that pass the test, the "result" of the non-terminating search is $x \in \emptyset$ or $x = \infty$. Thus, ∞ is the l.u.b. of an infinite chain $0\uparrow \sqsubseteq 1\uparrow \sqsubseteq \cdots \sqsubseteq n\uparrow \sqsubseteq \cdots$, which is the "limit" of the chain and actually, the program can not reach it by finite repetition.

Theorem 3.1.1 *$\langle N_\infty, \sqsubseteq \rangle$ is an algebraic CPO with base $N \cup N\uparrow$ and least element $\bot$.*

Corollary 3.1.2 *Let $\varphi : N_\infty \to N_\infty$ be a function such that $\varphi|_{N \cup N\uparrow}$ is monotone. Then, φ is continuous iff*

$$\varphi(\infty) = \sqcup\{\varphi(n\uparrow) \mid n\uparrow \in N\uparrow\}.$$

We say that an n-ary function φ on N_∞ is *stable* if $\varphi(N, \ldots, N) \subseteq N$: This notion corresponds to the totality of functions on N in the conventional framework. As well, φ is said *strict w.r.t. the i-th argument* if for any $d_1, \ldots, d_n$ in N_∞, $d_i \notin N$ implies $\varphi(d_1, \ldots, d_n) = \bot$. An n-ary function is *strict* if it is strict w.r.t. every argument. Also, in the conventional framework, if one of the argument to a function is undefined, then so is the value of the function. Remark that we can treat this situation by considering strict functions.

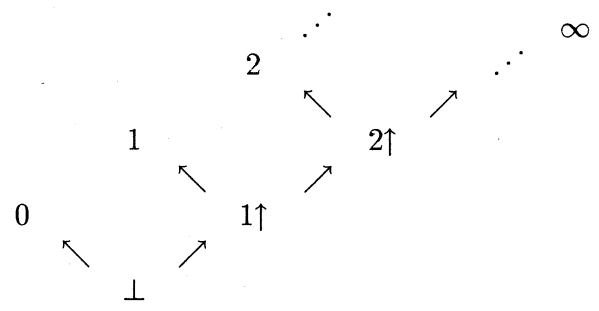

Figure 1: $\langle N_\infty, \sqsubseteq \rangle$

We shall now define the *successor* function s on N_∞ as follows:

$$\begin{aligned} s(n) &= n+1 \\ s(n\uparrow) &= (n+1)\uparrow \\ s(\infty) &= \infty. \end{aligned}$$

Note that N_∞ is partitioned into two well-founded s-chains, N and $N\uparrow$, and one singleton $\{\infty\}$. Clearly, ∞ is the unique fixedpoint of s.

Lemma 3.1.3 *The successor function s is stable, continuous, and not strict.*

The strict version of s can be obtained as the composition of s and a certain function that will be introduced later.

3.2 Quasi-Primitive Recursion

In this subsection, we shall formulate *quasi-primitive recursion*, together with *primitive recursion*, in a more formal way than the conventional framework.

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. For an interpretation I of $\mathrm{Fun}(E)$, we say that I *satisfies* E if for any $t = u$ in E, $t_I^{x_1,\ldots,x_n} = u_I^{x_1,\ldots,x_n}$ holds with $\{x_1, \ldots, x_n\} = \mathrm{Var}(t) \cup \mathrm{Var}(u)$.

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* that maps *zero* to 0 and *succ* to s.

Definition 3.2.1 A *quasi-primitive recursive definition (QPRD)* is a set of equations on $T(F \cup B, X)$ defined inductively as follows.

1. An empty set is a QPRD.
2. Let E_0 be a QPRD and $G_0 = \mathrm{Fun}(E_0) \cup B$. Also let $f \notin G_0$ be a function symbol and $x_1, \ldots, x_n$ mutually distinct variable symbols. Then, the union E of E_0 and one of the following systems of equations is a QPRD.

   (2a) **Constant functions.**

   $$f(x_1, \ldots, x_n) = t \in T(B, \emptyset).$$

   (2b) **Projection Functions.** For some $i \geq 0$,

   $$f(x_1, \ldots, x_n) = x_i.$$

   (2c) **Function Composition.** For some $g, h_1, \ldots, h_m \in G_0$,

   $$f(x_1, \ldots, x_n) = g(h_1(x_1, \ldots, x_n), \ldots, h_m(x_1, \ldots, x_n)).$$

(2d) **Quasi-Primitive Recursion.** For some $g, h, k_2, \ldots, k_n$ in G_0,

$$\begin{aligned} f(zero, x_2, \ldots, x_n) &= g(x_2, \ldots, x_n), \\ f(succ(x_1), x_2, \ldots, x_n) &= h(f(x_1, k_2(x_2), \ldots, k_n(x_n)), x_1, \ldots, x_n). \end{aligned}$$

A *primitive recursive definition* (*PRD*) is a QPRD such that whenever the case (2d) is applied, $k_2, \ldots, k_n$ are the identity function.

Lemma 3.2.2 *Let E_1 be a set of two equations given in (2d) above, G_0 a set of function symbols such that $f \notin G_0$ and $g, h, k_2, \ldots, k_n \in G_0$, and I_0 a continuous interpretation of G_0 on N_∞. Then, a continuous interpretation I of $G_0 \cup \{f\}$ that satisfies the following conditions uniquely exists.*

1. *I extends I_0 and satisfies E_1, and,*
2. *$f_I(\bot, d_2, \ldots, d_n) = \bot$ holds for any $d_2, \ldots, d_n$ in N_∞.*

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

Definition 3.2.3 For a QPRD E, the *denotational interpretation of E*, denoted by Δ_E, is an interpretation of $\mathrm{Fun}(E) \cup B$ defined inductively as below.

1. If $E = \phi$, then $\Delta_E = \Delta_B$.
2. Otherwise, there are a partition $\{E_0, E_1\}$ of E and a function symbol $f \notin \mathrm{Fun}(E_0) \cup B$ such that E_0 is a QPRD, and E_1 is one of the sets of equations from (2a – 2d) in Definition 3.2.1:

(2a – 2c) $\Delta_E = \Delta_{E_0} \cup \{f \mapsto t^{x_1,\ldots,x_n}_{\Delta_{E_0}}\}$, where $E_1 = \{f(x_1, \ldots, x_n) = t\}$

(2d) Δ_E is I in Lemma 3.2.2 with $G_0 = \mathrm{Fun}(E_0) \cup B$ and $I_0 = \Delta_{E_0}$.

Theorem 3.2.4 *For any QPRD E, the denotational interpretation Δ_E satisfies E.*

Definition 3.2.5 An n-ary function on N_∞ is *quasi-primitive recursive* if it is definable by some QPRD E, that is, it belgongs to the range of Δ_E.

Theorem 3.2.6 *Every quasi-primitive recursive function is stable and continuous.*

Let us see some examples of quasi-primitive recursive functions which will play important roles in the later of this paper. For notational simplicity, we shall leave each QPRDs anonymous and often identify a function symbol (say f) with its interpretation (say f_{Δ_E}).

Projection. From now on, we call the *i-th projection from* $(N_\infty)^n$ *to* N_∞ defined by (2b) as u_i^n. This function is, of course, not strict.

The strict "projection", e.g., $\tilde{u}_2^2$, can be defined by primitive recursion as:

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

Lemma 3.2.7 *The function* $\tilde{u}_2^2$ *is strict w.r.t. the first argument. That is, for any* d_1 *and* d_2 *in* N_∞, $d_1 \notin N$ *implies* $\tilde{u}_2^2(d_1, d_2) = \perp$.

Degeneration. The *degeneration* function defined below maps every $n \in N$ to n and every $d \notin N$ to $\perp$. This function will *project* any 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 minimization. Let π and β be $(k+1)$-ary functions on N_∞ and write $\bar{m}$ for a k-tuple $\langle m_1, \ldots, m_k \rangle$ in N^k. We say β is the *bounded minimization with respect to* π if for any $n \in N$ and $\bar{m}$ in N^k,

$$\beta(n, \bar{m}) = \begin{cases} \min\{i \in N \mid \pi(i, \bar{m}) = 0\} \text{ if } \pi(i, \bar{m}) = 0 \text{ for some } i \leq n, \\ n+1 \text{ otherwise.} \end{cases}$$

where min denotes the minimum with respect to the usual order on N.

Lemma 3.2.8 *If* π *is quasi-primitive recursive, then so is the bounded minimization* β *w.r.t.* π*: If* π *is represented by a function symbol* p *(i.e.,* $\pi = p_{\Delta_E}$ *for some* E*), then* β *represented by a symbol* b *is obtained by the following equations:*

$$\begin{aligned} b'(zero, \bar{x}, z) &= if(p(z, \bar{x}), zero, succ(zero)), \\ b'(succ(y), \bar{x}, z) &= if(p(z, \bar{x}), zero, succ(b'(y, \bar{x}, succ(z)))), \\ b(y, \bar{x}) &= b'(y, \bar{x}, zero). \end{aligned}$$

Suppose that there is $\bar{d}$ such that $\pi(i, \bar{d})$ is "false" (i.e., in $N_\infty - \{0, \perp\}$) for all i. Then, for any $n\uparrow$, $\beta(n\uparrow, \bar{d}) = (n+1)\uparrow$. We therefore obtain $\beta(\infty, \bar{d}) = \infty$. (See Corollary 3.1.2). The left-hand side virtually expresses "unbounded" search and the right-hand side means the failure of the search.

Note that the equational system in the lemma above is not a PRD but a QPRD. Although we can define bounded minimization by PRDs, the above definition is crucial for the result in the next section, where we shall see the minimization operation can be replaced by bounded minimization + ∞.

3.3 Minimization and Recursive Functions

In this section, we extend the class of functions dealt with in order to obtain the general class of *computable* functions. In the conventional framework, one introduces the operation of *minimization* for this purpose. In our framework, we do not require the operation as discussed below.

First we remark that in QPRDs there is no way to express ∞ since we only have *zero* as an constant symbol and quasi-primitive recursive functions (including s) are stable. Thus, we now extend our language by adding a new constant symbol *inf* to the set B of base function symbols: We let $B_{inf} = B \cup \{inf\}$ and define the *standard interpretation* $\Delta_{B_{inf}}$ of B_{inf} by $\Delta_{B_{inf}} = \Delta_B \cup \{inf \mapsto \infty\}$.

Definition 3.3.1 A *recursive definition (RD)* is a set E of equations on $T(F \cup B_{inf}, X)$, which is defined in the same way as SPRDs except that B_{inf} is used instead of B. 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 3.3.2 *For any RD E, the interpretation Δ_E satisfies E.*

Theorem 3.3.3 *Every recursive function is continuous.*

In general, recursive functions are not stable, and then, not quasi-primitive recursive: The following defines a representation *undef* of $\bot$, which is a non-stable constant.

$$undef = degen(inf).$$

Using *undef* and *inf*, every element in N_∞ is now representable by a term.

Let π be a $(k+1)$-ary function on N_∞. A *minimization with respect to* π is a k-ary function μ that satisfies: For any $\bar{m} \in N^k$,

$$\begin{cases} \mu(\bar{m}) = \min\{i \in N \mid \pi(i, \bar{m}) = 0\} \text{ if } \pi(i, \bar{m}) = 0 \text{ for some } i \in N, \\ \mu(\bar{m}) \notin N \text{ otherwise.} \end{cases}$$

Theorem 3.3.4 *For any recursive function π, there is a recursive minimization w.r.t π: The following equation gives it.*

$$m(x_1, \ldots, x_k) = b(inf, x_1, \ldots, x_k),$$

where b is defined as in Lemma 3.2.8

We note that, to show the theorem above, bounded minimization defined by a QPRD is used. As we discuss in Section 5, the closure property will not be held if bounded minimization is defined by a PRD.

4 Computability of Recursive Functions

In this section we discuss how our recursive functions are computed. The model of computation we use is term rewriting systems, which generalizes string rewriting systems and then, Turing machines.

4.1 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 $\mathrm{Var}(t) \supseteq \mathrm{Var}(u)$. 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 where $\langle t \to u \rangle = \langle t' \to u' \rangle$ and $t' = s$.

4.2 Algebraic Semantics of TRSs

To see that recursive functions are computable, we first introduce the algebraic semantics of left-linear and non-overlapping TRSs [6–8], which generalizes that of recursive program schemes originated by Nivat [10]. It will appear in a simpler form than the literature [6–8] since RDs have a quite restricted form as a TRS.

Let C be a subset of F. A TRS R is called a *constructor system on* C if for all $t \to u$ in R, $t = f(t_1, \ldots, t_k)$ holds for some $f \in F - C$ and $t_1, \ldots, t_k \in T(C, X)$. A constructor system R is *perfect* if for every f in $\mathrm{Fun}(R) - C$ and every $t_1, \ldots, t_k$ in $T(C, \emptyset)$, $f(t_1, \ldots, t_k)$ is a redex of R.

In what follows, we regard an RD E as a TRS and let $E_{inf} = E \cup \{inf \to succ(inf)\}$.

Lemma 4.2.1 *For any RD* E, E_{inf} *is a left-linear and non-overlapping TRS. Moreover, it is a perfect constructor system on* B.

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 $P \subseteq \mathrm{Node}(U)$, $T = U[p \leftarrow \Omega \mid p \in \min(P)]$. This ordering is due to [10].

Proposition 4.2.2 $\langle T^\infty(F \cup B_{\Omega,inf}), \preceq \rangle$ *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 a term* Ω.

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

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

where $P = \{p \in \mathrm{Node}(t) |\ t(p) \in F \cup \{inf\}\}$.[2] If $t \rightarrow^*_{E_{inf}} u$, 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:

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

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.

4.3 Computational Interpretation of RDs

Using the function $\mathrm{Val}_{E_{inf}}$, we formulate the functions on N_∞ *computed* by RDs.

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 $\langle T_\infty(B_\Omega), \preceq\rangle$.

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 easily be 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 $\langle T(B_\Omega) \cup \{inf\}, \lesssim_{E_{inf}}\rangle$ to $\langle N_\infty, \sqsubseteq\rangle$.

Definition 4.3.1 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,

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

for $d_1, ..., d_n$ in N_∞.

Theorem 4.3.2 *For any RD* E, $\Gamma_E = \Delta_E$.

[2] 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 $T(F \cup B_{\Omega,inf})$.

5 Quasi-Primitive Recursion vs. Primitive Recursion

In the earlier sections, we have introduced and mainly dealt with quasi-primitive recursion, a generalized version of primitive recursion. We shall now investigate the relationship between these two schemes of recursion.

5.1 Conventional Viewpoint

We can show that:

Theorem 5.1.1 *For any n-ary quasi-primitive recursive function σ, there is an n-ary primitive recursive function π such that $\sigma \cap (N^n \times N) = \pi \cap (N^n \times N)$.*

That is, they have the same descriptive power from the conventional point of view,

5.2 Detecting Bottom

Now, we shall see that quasi-primitive and primitive recursion are different with each other as schemes to describe functions on N_∞.

We have defined recursive functions using QPRDs with ∞. On the other hand, we can consider the class of functions described by PRDs with ∞. We call the latter as PR_∞-functions (Under this convention, we should call recursive functions as QPR_∞-functions). We shall show that the class of recursive functions is strictly larger than that of PR_∞-functions. For the purpose, we introduce a property that PR_∞-functions satisfy but recursive functions do generally not.

We extend the order $\sqsubseteq$ into an order over the set of n-tuples of elements in N_∞: For $\bar{d} = \langle d_1, \ldots, d_n \rangle$ and $\bar{d}' = \langle d'_1, \ldots, d'_n \rangle$, $\bar{d} \sqsubseteq \bar{d}'$ iff $d_i \sqsubseteq d'_i$ for all i. Let I be a subset of $\{1, \cdots, n\}$. We also define a strict order $\sqsubset_I$ over $(N_\infty)^n$ as follows: $\bar{d} \sqsubset_I \bar{d}'$ if $\bar{d} \sqsubseteq \bar{d}'$ and

1. for all $i \in I$, $d_i \sqsubset d'_i$ or d_i is maximal, and
2. for all $i \notin I$, $d_i = d'_i$.

Clearly, for any n-ary monotone function φ, $\bar{d} \sqsubset_I \bar{d}'$ implies $\varphi(\bar{d}) \sqsubseteq \varphi(\bar{d}')$.

For a monotone function φ, we say φ is *I-saturated at $\bar{d}$* if for all $\bar{e}$ such that $\bar{d} \sqsubset_I \bar{e}$ we have $\varphi(\bar{d}) = \varphi(\bar{e})$. A monotone function φ is said *simply monotone* if, for arbitrary $\bar{d}$, $\bar{d}'$ and I, $\bar{d} \sqsubset_I \bar{d}'$ and $\varphi(\bar{d}) = \varphi(\bar{d}')$ implies that φ is I-saturated at $\bar{d}'$.

Theorem 5.2.1 *Any PR_∞-function is simply monotone.*

It can easily be shown that recursive functions are generally not simply monotone. Thus, we have:

Corollary 5.2.2 *The class of recursive functions is strictly larger than that of PR_∞-functions.*

From the theorem above, we also obtain another interesting result. First remark that, in general, we can not decide whether $\varphi(\bar{d}) = \bot$ holds for a given recursive function φ and arguments $\bar{d} = \langle d_1, \ldots, d_n \rangle$, since this decision problem can be regarded as the halting problem from the following argument. Although the statement "$\varphi(\bar{d})$ is undefined" in the conventional formulation is parallel to the statement "$\varphi(\bar{d}) \notin N$" here, the latter is equivalent to "$degen(\varphi(\bar{d})) = \bot$" where *degen* is the degeneration operator. Since our class of recursive functions has been shown to include the class of *partial recursive functions* in the conventional framework, the decision of $degen(\varphi(\bar{d})) = \bot$ must be unsolvable.

Nevertheless, the unsolvability is not the case for PR_∞-functions. As we shall see below, these functions enjoy a sufficient condition which ensures that the halting problem is solvable.

Definition 5.2.3 An n-ary function φ is *lower-bound-verifiable* if for any $\bar{d} \in (N_\infty)^n$ and any $c \in N \cup N\uparrow$, it is decidable whether $\varphi(\bar{d}) \sqsupseteq c$ or not.

Note that if $c \in N$, then, the above test is just the test of $\varphi(\bar{d}) = c$.

The condition $\varphi(\bar{d}) = \bot$ is decidable for lower-bound-verifiable functions since this condition is true iff $\varphi(\bar{d}) \not\sqsupseteq 0$ and $\varphi(\bar{d}) \not\sqsupseteq s(\bot)$.

From the computational point of view, simply monotone functions enjoy a useful property.

Lemma 5.2.4 *Let φ be a simply monotone function and suppose that the lower-bound test $\varphi(\bar{d}) \sqsupseteq c$ is decidable for "finite" $\bar{d} \in (N \cup N\uparrow)^n$ and $c \in N \cup N\uparrow$. Then, φ is lower-bound verifiable.*

Using Theorem 5.2.1 and the above lemma, we get:

Theorem 5.2.5 *Any PR_∞-function is lower-bound-verifiable.*

The result suggests that the device PRDs + ∞ has not sufficient power to describe all computable functions.

6 Future Work

In our framework, the class of recursive function is proved to be closed under minimization, and clearly it includes all quasi-primitive recursive functions. However, it has not been shown that this class is the least one among such classes. We conjecture the positive answer to the question because recursive functions are "computable" in the sense of algebraic semantics of term rewriting systems (the function $\omega_{E_{inf}}$ and the relation $\rightarrow_{E_{inf}}$ is "recursive" in the conventional sense).

Acknowledgement
The author wishes to thank Prof. Muneo GOTO and Prof. Yasuyoshi INAGAKI for their encouragement, and Dr. Mizuhito OGAWA and Dr. Ken MANO for their helpful discussions.

References

[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] T. Naoi and Y. Inagaki, "Recursive Functions on a Completed Set of Natural Numbers," Technical Report COMP90-1, IEICE (1990).

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

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