Available online at www.sciencedirect.com
SCIENCE DIRECT
ELSEVIER
Electronic Notes in Theoretical Computer Science 120 (2005) 83–95
Electronic Notes in Theoretical Computer Science
www.elsevier.com/locate/entcs

Type-2 Computability and Moore's Recursive Functions

Akitoshi Kawamura[1]

Department of Computer Science,
Graduate School of Information Science and Technology,
The University of Tokyo, Japan

Abstract

Regarding effectivity of functions on the reals, there have been several proposed models of analog, continuous-time computation, as opposed to the digital, discrete nature of the type-2 computability. We study one of them, Moore's *real (primitive) recursive* functions, whose definition mimics the classical characterization of recursive functions on $\mathbb{N}$ by the closure properties. We show that the class of type-2 computable real functions falls between Moore's classes of primitive recursive and recursive functions.

Keywords: type-2 computability, real recursive functions, analog computation, integral equations.

1 Introduction

This paper investigates links between the two theories in the title which deal with effectivity of functions from and to the real numbers.

There has been increasing interest in analog computation models in which continuous quantities are treated as an entity in themselves and computation takes place in continuous time. As a formulation of effectivity in such an analog world we consider the class of *recursive functions* and its subclass of *primitive recursive functions*, both introduced by Moore [4]. These classes are defined by closure properties that bear a flavor of Kleene's classical characterization of the discrete recursive functions on the natural numbers [5].

[1] Email: akitoshi@is.s.u-tokyo.ac.jp

doi:10.1016/j.entcs.2004.06.036

Although Moore's classes have been compared [4] against the classical recursive functions over $\mathbb{N}$ under the usual embedding $\mathbb{N} \subseteq \mathbb{R}$, their relation with the well-developed theory of type-2 computability [7] seems to have drawn little attention so far. In this paper, we study Moore's classes in relationship to type-2 computability, and show inclusions between the classes from the two theories by simulating each model by the other.

We introduce Moore's classes in Section 2 (with slight modification on the definition that seems necessary to give it rigor), and explore them with examples in Section 3. We show in Section 4 that Moore's primitive recursiveness implies type-2 computability, and in Section 5 that type-2 computability in turn implies Moore's recursiveness.

In what follows, a function may be partial, unless explicitly stated as total. We occasionally annotate a function by superscripts to show its arity, thus writing $f^{m\to n}$ for a function f taking m arguments and yielding n values. A superscript on an operation on functions, as in $h = \mathrm{C\textsc{m}}^{m\to n}[f;g]$, denotes the arity of the resulting function h. For terminology pertaining to type-2 computability we follow Weihrauch [7].

2 Moore's Real Recursive Functions

Recall that the classical discrete recursive functions on the natural numbers are characterized by the closure property under certain operations on functions:

Juxtaposition. For n functions $f_1^{m\to 1}, \ldots, f_n^{m\to 1}$, define

$$\mathrm{J\textsc{x}}^{m\to n}[f_1,\ldots,f_n](\boldsymbol{v}) = \big(f_1(\boldsymbol{v}),\ldots,f_n(\boldsymbol{v})\big).$$

Composition. For functions $f^{m\to n}$ and $g^{l\to m}$, define

$$\mathrm{C\textsc{m}}^{l\to n}[f;g](\boldsymbol{v}) = f\big(g(\boldsymbol{v})\big).$$

Primitive Recursion on $\mathbb{N}$. For functions $f^{m\to n}$ and $g^{m+1+n\to n}$ on $\mathbb{N}$, define

$$\begin{aligned}\mathrm{P\textsc{r}}_{\mathbb{N}}^{m+1\to n}[f;g](\boldsymbol{v},0) &= f(\boldsymbol{v}),\\ \mathrm{P\textsc{r}}_{\mathbb{N}}^{m+1\to n}[f;g](\boldsymbol{v},t+1) &= g\big(\boldsymbol{v},t,\mathrm{P\textsc{r}}_{\mathbb{N}}^{m+1\to 1}[f;g](\boldsymbol{v},t)\big).\end{aligned}$$

Minimization on $\mathbb{N}$. For a function $f^{m+1\to 1}$ on $\mathbb{N}$, define

$$\mathrm{M\textsc{n}}_{\mathbb{N}}^{m\to 1}[f](\boldsymbol{v}) = (\text{the least } t\in\mathbb{N} \text{ with } f(\boldsymbol{v},t)=0, \text{ if any}).$$

The recursive functions on $\mathbb{N}$ are then defined to be those in the smallest class that contains some basic functions and is closed under the above operations.

Moore's real (primitive) recursive functions are defined by replacing the last two operations by their real versions:

Primitive Recursion on $\mathbb{R}$. For real functions $f^{m\to n}$ and $g^{m+1+n\to n}$, define $\textsc{Pr}^{m+1\to n}[f;g]$ to be the solution h of the integral equation

$$h(\boldsymbol{v},t) = f(\boldsymbol{v}) + \int_0^t g\big(\boldsymbol{v},\tau,h(\boldsymbol{v},\tau)\big)\,\mathrm{d}\tau,$$

where $\textsc{Pr}[f;g]$ is defined at all and only those $(\boldsymbol{v},t)$ such that for all τ between 0 and t inclusive,

(i) the integral equation uniquely determines $h(\boldsymbol{v},\tau)$, and
(ii) $g\big(\boldsymbol{v},\tau,h(\boldsymbol{v},\tau)\big)$ is defined.

Minimization on $\mathbb{R}$. For a real function $f^{m+1\to 1}$, define

$$\textsc{Mn}^{m\to 1}[f](\boldsymbol{v}) = \mu t.\, f(\boldsymbol{v},t) = \inf\{\, t\in\mathbb{R} \mid f(\boldsymbol{v},t) = 0\,\},$$

where inf chooses the t with least absolute value. If two such bounds have the same absolute value, then by convention the negative one is chosen.

Definition 2.1 The *primitive recursive functions* are those in the smallest class that contains the 0-ary constant functions $0^{0\to 1}$, $1^{0\to 1}$, $-1^{0\to 1}$ and is closed under $\textsc{Jx}$, $\textsc{Cm}$ and $\textsc{Pr}$. The *recursive functions* are those in the smallest class that contains $0^{0\to 1}$, $1^{0\to 1}$, $-1^{0\to 1}$ and is closed under $\textsc{Jx}$, $\textsc{Cm}$, $\textsc{Pr}$ and $\textsc{Mn}$.

The operation $\textsc{Pr}$ needs some comments. The real $\textsc{Pr}$, unlike its discrete counterpart, can produce partial functions from total ones, as in the equation

$$\tan t = 0 + \int_0^t (1+\tan^2\tau)\,\mathrm{d}\tau,$$

whose solution is defined only on the open interval $(-\pi/2;\pi/2)$. We therefore specified the domain of the produced function to be maximal under the two conditions above. The condition (i) forbids producing meaningful functions from e.g. the equation

$$h(t) = 0 + \int_0^t \mathrm{zero?}\big(h(\tau)\big)\,\mathrm{d}\tau \quad \text{where} \quad \mathrm{zero?}(x) = \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise} \end{cases}$$

of which both $f(t) = t$ and $f(t) = 0$ (as well as two other functions) are solutions. The condition (ii), though not clear in Moore's original definition [4], seems necessary in order to make true his claim that all primitive recursive functions are analytic. To see this, consider the function g defined

by $g(t,x) = 1/\sqrt{|t|}$ for $t \in \mathbb{R} \setminus \{0\}$, which is primitive recursive (and analytic on its domain) as will be clear shortly. But the function h defined by

$$h(t) = \begin{cases} 2\sqrt{t} & \text{if } t \geqslant 0 \\ -2\sqrt{-t} & \text{if } t < 0 \end{cases}$$

is not differentiable at $t = 0$, although it "nearly" solves the equation

$$h(t) = 0 + \int_0^t g\big(\tau, h(\tau)\big)\,\mathrm{d}\tau.$$

This is why we impose (ii), a condition stricter than the usual notion of integrability; according to our formulation, where the integrand must be defined everywhere on the interval in order for the integral to be defined, $\textsc{Pr}[0^{0\to1}; g]$ is not the above h but simply a nowhere defined function. Variants of $\textsc{Pr}$ are compared by Campagnolo [1] and Graça [2].

3 Examples

This section gives examples of real recursive functions. The following Lemmata 3.1, 3.2 and 3.3 originate partly from Moore [4]. We begin with some primitive recursive functions:

Lemma 3.1 *The following functions are primitive recursive: for each $n \in \mathbb{N}$, the n-ary constants $0^{n\to1}$, $1^{n\to1}$, $-1^{n\to1}$; for positive integers n and $i \leq n$, the n-ary projection $\mathrm{id}_i^{n\to1}$ to the i-th component; binary addition $\mathrm{add}^{2\to1}$ and multiplication $\mathrm{mul}^{2\to1}$; reciprocal $\mathrm{inv}_+^{1\to1}\colon x \mapsto 1/x$ and natural logarithm $\ln^{1\to1}$ defined for positive numbers; the trigonometric and exponential functions $\sin^{1\to1}$, $\cos^{1\to1}$, $\exp^{1\to1}$ and constants $\mathrm{e}^{0\to1}$, $\pi^{0\to1}$.*

Proof. The n-ary constant $0^{n\to1}$ is built by $0^{n\to1} = \textsc{Cm}^{n\to1}\big[0^{0\to1}; \textsc{Jx}^{n\to0}[\,]\big]$; similarly for $1^{n\to1}$ and $-1^{n\to1}$. Projections are defined inductively on $n - i$ by $\mathrm{id}_i^{i\to1} = \textsc{Pr}^{i\to1}[0^{i-1\to1}; 1^{i+1\to1}]$ and $\mathrm{id}_i^{n+1\to1} = \textsc{Pr}^{n+1\to1}[\mathrm{id}_i^{n\to1}; 0^{n+2\to1}]$. For addition and multiplication, use the usual recursive definition $\mathrm{add}^{2\to1} = \textsc{Pr}^{2\to1}\big[\mathrm{id}_1^{1\to1}; 1^{3\to1}\big]$ and $\mathrm{mul}^{2\to1} = \textsc{Pr}^{2\to1}\big[0^{1\to1}; \mathrm{id}_1^{3\to1}\big]$. For $\mathrm{inv}_+^{1\to1}$, define

$$\begin{aligned} \mathrm{square}^{1\to1} &= \textsc{Cm}\big[\mathrm{mul}^{2\to1}; \textsc{Jx}\big[\mathrm{id}_1^{1\to1}, \mathrm{id}_1^{1\to1}\big]\big], \\ g^{1\to1} &= \textsc{Pr}\big[1^{0\to1}; \textsc{Cm}\big[\textsc{Cm}\big[\mathrm{mul}^{2\to1}; \textsc{Jx}\big[-1^{1\to1}, \mathrm{square}^{1\to1}\big]\big]; \mathrm{id}_2^{2\to1}\big]\big], \\ \mathrm{inv}_+^{1\to1} &= \textsc{Cm}\big[g^{1\to1}; \textsc{Cm}\big[\mathrm{add}^{2\to1}; \textsc{Jx}\big[\mathrm{id}_1^{1\to1}, -1^{1\to1}\big]\big]\big], \end{aligned}$$

or, in a more familiar language,

$$\mathrm{square}(z) = z^2, \qquad g(y) = 1 - \int_0^y \mathrm{square}\big(g(\eta)\big)\,\mathrm{d}\eta, \qquad \mathrm{inv}_+ = g(x-1).$$

Logarithm and exponentiation are analogous, using suitable integral equations. Trigonometric functions are defined by

$$\mathrm{trig}^{1\to 2} = \textsc{Pr}\big[\textsc{Jx}\big[0^{0\to 1}, 1^{0\to 1}\big]; \textsc{Jx}\big[\mathrm{id}_3^{3\to 1}, \textsc{Cm}\big[\mathrm{mul}^{2\to 1}; \textsc{Jx}\big[-1^{3\to 1}, \mathrm{id}_2^{3\to 1}\big]\big]\big]\big],$$
$$\sin^{1\to 1} = \textsc{Cm}\big[\mathrm{id}_1^{2\to 1}; \mathrm{trig}^{1\to 2}\big], \qquad \cos^{1\to 1} = \textsc{Cm}\big[\mathrm{id}_2^{2\to 1}; \mathrm{trig}^{1\to 2}\big],$$

or equivalently, by a system of equations

$$\begin{pmatrix}\sin(t)\\ \cos(t)\end{pmatrix} = \begin{pmatrix}0\\ 1\end{pmatrix} + \int_0^t \begin{pmatrix}\cos(\tau)\\ -\sin(\tau)\end{pmatrix}\mathrm{d}\tau.$$

Euler's constant is $\mathrm{e}^{0\to 1}(\,) = \exp(1)$. The circle ratio is $\pi^{0\to 1}(\,) = 4\,\mathrm{Arctan}(1)$, with Arctan defined by a suitable integral equation. □

Without Mn, we only obtained continuous functions. We next construct functions with discontinuities by using Mn:

Lemma 3.2 *The following functions are recursive: total tests*

$$\mathrm{zero?}\colon x \mapsto \begin{cases} 0 & \text{if } x = 0 \\ 1 & \text{otherwise,}\end{cases} \qquad \mathrm{integer?}\colon x \mapsto \begin{cases} 0 & \text{if } x \text{ is an integer} \\ 1 & \text{otherwise;}\end{cases}$$

the conditional $\mathrm{ifthenelse}^{1+2m\to m}$ *that maps* $(p, \boldsymbol{a}, \boldsymbol{b})$ *to* $\boldsymbol{a}$ *if* $p = 0$ *and* $\boldsymbol{b}$ *otherwise; the total reciprocal* inv *(extending* inv_+ *in Lemma 3.1) with* $\mathrm{inv}(0) = 0$ *and* $\mathrm{inv}(t) = 1/t$ *for each* $t \neq 0$*; the function* round *mapping* x *to the unique integer in* $(x-1/2; x+1/2]$*; the function* digit *mapping each* (x, b, i) *with* $b > 1$ *and* $i \in \mathbb{Z}$ *to the digit in* b^i*'s place when* x *is written in base-*b *notation.*

Note that $\mathrm{digit}(x, b, i)$ is unspecified for $b \leq 1$ or $i \notin \mathbb{Z}$: it can be undefined, or defined to be any value, for such b or i. We adopt this convention throughout, namely that when we claim a function to be recursive whose value is specified only on some subset of the input space, we mean the function has a recursive extension, whose values on unexpected arguments we do not care.

Proof. Define

$$\mathrm{zero?}(x) = \mu y.\,(x^2 + y^2)(1-y),$$
$$\mathrm{integer?}(x) = \mathrm{zero?}\big(\sin(\pi x)\big),$$
$$\mathrm{ifthenelse}(p, \boldsymbol{a}, \boldsymbol{b}) = \big(1 - \mathrm{zero?}(p)\big)\cdot \boldsymbol{a} + \mathrm{zero?}(p)\cdot \boldsymbol{b},$$

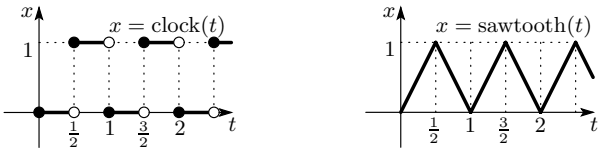

Fig. 1. The functions clock and sawtooth.

with the usual multiplication on vectors built in the obvious way, and

$$\begin{aligned}\mathrm{inv}(x) &= \mu y.\, x(xy-1),\\ \mathrm{round}(x) &= x - \mu r.\, \mathrm{integer?}(x-r),\\ \mathrm{digit}(x,b,i) &= \mathrm{round}\left(\frac{x}{b^i}-\frac{1}{2}\right) - b\cdot\mathrm{round}\left(\frac{x}{b^{i+1}}-\frac{1}{2}\right),\end{aligned}$$

where exponentiation b^i is built by $b^i = \exp(i\cdot\ln(b))$. □

We prepare some general ways to create a recursive function from another, which we will use in Section 5 to show several complicated functions recursive.

Lemma 3.3 *If $f^{m\to m}$ is recursive, so is the function*

$$g^{m+1\to m}\colon (\boldsymbol{v},k)\mapsto f^k(\boldsymbol{v}) = \underbrace{f(f(\cdots(f}_{k}(\boldsymbol{v}))\cdots)) \qquad (k\in\mathbb{N}\setminus\{0\}).$$

Proof. Define $F^{m+m+1\to m}$ by

$$F(\boldsymbol{b},\boldsymbol{v},t) = f\big(\mathrm{ifthenelse}^{1+2m\to m}\big(1-\mathrm{integer?}(t)+\mathrm{digit}(t,2,-1),\boldsymbol{b},\boldsymbol{v}\big)\big),$$

so that $\mathrm{dom}\,F \supseteq \mathbb{R}^m\times\mathrm{dom}\,f\times\mathbb{R}$ and $F(\boldsymbol{b},\boldsymbol{v},t) = f(\boldsymbol{b})$ if $t\in(n;n+1/2)$ for some $n\in\mathbb{Z}$. Define $\mathrm{clock}^{1\to1}$ and $\mathrm{sawtooth}^{1\to1}$ by

$$\mathrm{clock}(t) = \mathrm{digit}(t,2,-1),\quad \mathrm{sawtooth}(t) = 0+\int_0^t \big(2-4\,\mathrm{clock}(\tau)\big)\,\mathrm{d}\tau$$

using digit from Lemma 3.2 (Fig. 1). Let

$$\begin{pmatrix} g(\boldsymbol{v},t)\\ h(\boldsymbol{v},t)\end{pmatrix} = \begin{pmatrix}\boldsymbol{v}\\ \boldsymbol{v}\end{pmatrix} + \int_0^t \begin{pmatrix} 2\big(F\big(h(\boldsymbol{v},\tau),\boldsymbol{v},\tau\big)-h(\boldsymbol{v},\tau)\big)\big(1-\mathrm{clock}(\tau)\big)\\ 2\big(h(\boldsymbol{v},\tau)-g(\boldsymbol{v},\tau)\big)\,\mathrm{inv}\big(\mathrm{sawtooth}(\tau)\big)\,\mathrm{clock}(\tau)\end{pmatrix}\mathrm{d}\tau,$$

using inv from Lemma 3.2 (Fig. 2). We have $g(\boldsymbol{v},k) = h(\boldsymbol{v},k) = f^k(\boldsymbol{v})$ for $k\in\mathbb{N}\setminus\{0\}$, as desired, with g computing the next value during the first half of each clock time, and h then catching up during the second half. □

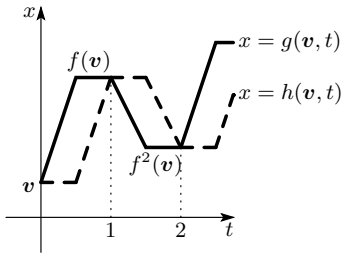

Fig. 2. Simulating iteration $f^k(\boldsymbol{v})$ by real recursive functions.

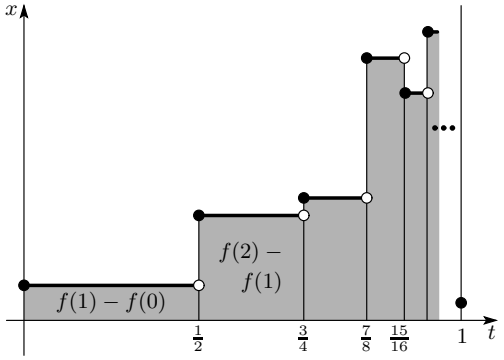

Fig. 3. Simulating limitation $\lim_{k\to\infty} f(k)$ by real recursive functions.

As an application, if $f^{m+1\to 1}$ is recursive, so is

$$g\colon (\boldsymbol{v}, n) \mapsto \sum_{i=0}^{n-1} f(\boldsymbol{v}, i) \qquad (n \in \mathbb{N} \setminus \{0\}),$$

since $g(\boldsymbol{v}, n) = \mathrm{id}_{m+2}^{m+2\to 1}\big(F^n(\boldsymbol{v}, 0, 0)\big)$ where $F(\boldsymbol{v}, t, S) = \big(\boldsymbol{v}, t+1, f(\boldsymbol{v}, t) + S\big)$.

Lemma 3.4 *If $f^{m+1\to n}$ is recursive, so is the function*

$$g^{m\to n}\colon \boldsymbol{v} \mapsto \lim_{k\to\infty} f(\boldsymbol{v}, k) \qquad (\{\boldsymbol{v}\} \times \mathbb{N} \subseteq \operatorname{dom} f),$$

where k runs through $\mathbb{N}$.

Proof. Define $\lg\colon x \mapsto \mu y.\, x(2^y - x)$ and $k\colon t \mapsto \mathrm{round}\big(-\lg(1-t) - 1/2\big)$ using round from Lemma 3.2. Then k is recursive and $\operatorname{dom} k = (-\infty; 1]$. Hence

$$g(\boldsymbol{v}) = f(\boldsymbol{v}, 0) + \int_0^1 2^{k(t)+1}\big(f\big(\boldsymbol{v}, k(t)+1\big) - f\big(\boldsymbol{v}, k(t)\big)\big)\,\mathrm{d}t$$

gives the desired g (Fig. 3). □

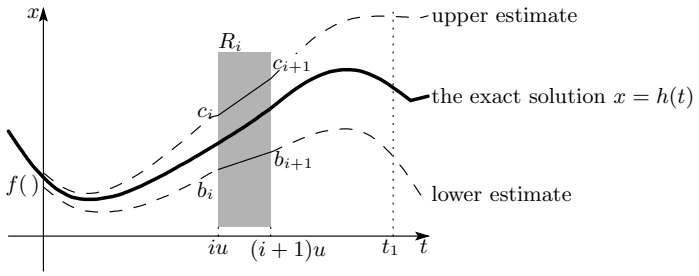

Fig. 4. Solving an integral equation. Since $b_{i+1} - b_i < ug(t,x) < c_{i+1} - c_i$ for all $(t,x) \in R_i$, we have $b_{i+1} < h\big((i+1)u\big) < c_{i+1}$ if $b_i < h(iu) < c_i$.

4 Primitive Recursive Functions Are Computable

Given our clarification made in Section 2 about the definition of PR, the theory of differential equations tells us [6,3] that if f and g are analytic functions with open domain, so is $\mathrm{PR}[f;g]$, whence so are all primitive recursive functions, as suggested by Moore [4]. In this section we further show that they are type-2 computable. Other things being obvious, we are to show:

Theorem 4.1 *If $f^{m\to n}$ and $g^{m+1+n\to n}$ are type-2 computable analytic functions with open domain, so is $h = \mathrm{PR}^{m+1\to n}[f;g]$.*

Proof. We only show computability, referring to the theory of differential equations [3, Chap. 14] for analyticity and open domain. We sketch a proof for $(m,n) = (0,1)$, the general case being analogous. Suppose we want the machine to compute an interval of length $< \varepsilon$ containing $h(t_1)$. We assume $t_1 > 0$, the negative case being analogous. If $h(t_1)$ is undefined there is nothing left to prove. Therefore suppose that $h(t_1)$ is defined, i.e. that the integral equation

$$h(t) = f(\,) + \int_0^t g\big(\tau, h(\tau)\big)\,\mathrm{d}\tau$$

has a unique solution defined on all of $[0;t_1]$. Then there are $n \in \mathbb{N}$ and rational numbers β, u, d, $b_0, b_1, \ldots, b_n$, $c_0, c_1, \ldots, c_n$ such that

(i) $(n-1)u < t_1 \leqslant nu$;

(ii) $b_0 < f(\,) < c_0$ and $c_0 - b_0 = d$;

(iii) $c_{n-1} - b_{n-1} + 2d < \beta < \varepsilon$;

(iv) For each $i = 0, \ldots, n-1$, all points $(t,x) \in R_i = \big[ui; u(i+1)\big] \times [b_i - d; c_i + d]$ satisfy $-d < b_{i+1} - b_i < ug(t,x) < c_{i+1} - c_i < d$.

To see why, first note that since g has open domain and is analytic, there are numbers $M \in \mathbb{N} \setminus \{0\}$ and $\eta > 0$ such that g and all its derivatives exist

and their absolute values are bounded by M on the open set $\{\,(t,x) \in \mathbb{R}\times\mathbb{R} \mid -\eta < t < t_1+\eta \text{ and } |x-h(t)| < \eta\,\}$ enclosing the true orbit.

Let $\beta < \min\{\varepsilon,\eta\}$ and $d < \mathrm{e}^{-6t_1M}\beta$ be positive, so that the second inequality of (iii) holds. Let u and n satisfy $u(M+d) < d$ and (i). Let b_0 and c_0 satisfy (ii). For $i = 0,\ldots,n-1$, let b_{i+1} and c_{i+1} be such that

$$b_i + u\min_{(t,x)\in R_i} g(t,x) - ud < b_{i+1} < b_i + u\min_{(t,x)\in R_i} g(t,x)$$
$$c_i + u\max_{(t,x)\in R_i} g(t,x) < c_{i+1} < c_i + u\max_{(t,x)\in R_i} g(t,x) + ud$$

Then (iv) follows inductively because $|b_{i+1}-b_i| < u|\min_{(t,x)\in R_i} g(t,x)| + ud < uM + ud < d$ and similarly for $|c_{i+1}-c_i|$. We have

$$\begin{aligned}
c_{i+1} - b_{i+1} &< c_i - b_i + u\max_{(t,x)\in R_i} g(t,x) - u\min_{(t,x)\in R_i} g(t,x) + 2ud \\
&< c_i - b_i + uM(c_i - b_i + 2d + u) + 2ud \\
&= (c_i-b_i)\left(1 + uM\left(1 + \frac{2d+u+2d/M}{c_i-b_i}\right)\right) \\
&< (c_i-b_i)\left(1 + uM\left(1 + \frac{2d+d+2d}{d}\right)\right) = (c_i-b_i)(1+6uM),
\end{aligned}$$

where the second inequality uses the fact that g's derivatives are bounded by M and that no two points in R_i are more than $c_i - b_i + 2d + u$ apart. Hence $c_{n-1} - b_{n-1} < d(1+6uM)^{n-1} < d\mathrm{e}^{6u(n-1)M} < d\mathrm{e}^{6t_1M} < \beta$, as required by (iii).

Under the conditions above, the true orbit h on $[0;t_1]$ passes between the polygons $b_0\cdots b_n$ and $c_0\cdots c_n$ (Fig. 4), so that $\big(t_1, h(t_1)\big) \in R_{n-1}$.

Our machine tries all tuples $\big(\beta,u,d,(b_0,c_0),\ldots,(b_n,c_n)\big) \in \mathbb{Q}\times\mathbb{Q}\times\mathbb{Q}\times(\mathbb{Q}\times\mathbb{Q})^*$, until it happens to find one that meets the above conditions, when it outputs the interval $[b_{n-1}-d; c_{n-1}+d]$ of length $< \varepsilon$. Note that if a tuple does meet the conditions with $\beta < \eta$, the machine can tell it, since g is then defined on all over R_i, making (iv) verifiable. Also note that by good job scheduling, every tuple is eventually taken care of, even if the machine, not knowing η, gets stymied forever on other tuples because of undefinedness. □

Thus, every primitive recursive function is type-2 computable.

5 Computable Functions Are Recursive

Moore shows [4] that recursive real functions can simulate Turing machines, so all discrete recursive functions are real recursive (under the canonical embedding $\mathbb{N} \subseteq \mathbb{R}$). It is natural then to ask the same question for type-2 machines,

which share the digital, discrete nature with the Turing machine. This section sketches a way to simulate type-2 machines by real recursive functions.

First of all, we need to specify the representation ρ by which the machine to be simulated denotes real numbers. We define

$$\rho(\mathtt{0}^n, a_0 a_1 \cdots) = n + \sum_{i=0}^{\infty} \overline{a_i} \left(\frac{1}{2}\right)^i \quad (a_i \in \{\mathtt{0}, \mathtt{1}, \mathtt{2}\}),$$

where $\overline{\mathtt{0}} = 0$, $\overline{\mathtt{1}} = 1$ and $\overline{\mathtt{2}} = 2$. Thus, one real number is expressed by two strings, one finite and one infinite, denoting respectively the integer part and the fractional part (though this "fractional part" ranges from 0 to 4). The integer part uses the tally notation; the fractional part uses the encoding that resembles the infinite decimal expansion in that it writes the number as a power series, but differs in that for each digit, the regions covered by different digits overlap each other. This redundancy makes ρ define the standard computability [7]. So a function $f^{m\to n}$ is computable if, when viewed as stream conversion between ρ-names, it is realized by some type-2 machine with $2m$ input tapes and $2n$ output tapes.

We next describe how to represent a configuration of the machine by real numbers. A configuration consists of:

- the state of the machine, which we encode by a natural number.
- m input tapes containing an infinite string $a_{-l}a_{-l+1}\cdots a_0 a_1 \cdots$ with the tape head pointing to a_0, which we encode by
$$\sum_{i=-l}^{\infty} (\overline{a_i} + 1) \left(\frac{1}{5}\right)^i;$$
- m input tapes and several work tapes containing a finite string $a_{-l}a_{-l+1}\cdots a_0 a_1 \cdots a_k$ with the tape head pointing to a_0, which we encode by
$$\sum_{i=-l}^{k} (\overline{a_i} + 1) \left(\frac{1}{5}\right)^i;$$
- $2n$ output tapes containing a finite string $a_0 a_1 \cdots a_k$, which we encode by
$$\sum_{i=0}^{k} (\overline{a_i} + 1) \left(\frac{1}{5}\right)^i.$$

A program is encoded in a natural number in any reasonable manner.

Having specified the encoding, we now consider how to simulate each step of the machine. We need a recursive real function that, when given (the real

numbers standing for) a configuration and a program, returns the configuration resulting after executing one appropriate instruction in the program. For this purpose we need to move the head to the left or right, change the state according to the symbol read, and write a symbol onto a tape, all of which can be carried out by functions from Lemmata 3.1 and 3.2. For example, to move the tape head of a work tape to the right, simply multiply the corresponding real number by 5; to write a 2 on a cell, change the appropriate digit of the corresponding real number to 3 using digit from Lemma 3.2.

By Lemma 3.3 on iteration we have a recursive function that, given the initial configuration and a number $k \in \mathbb{N}$, computes the string written on an output tape after k steps. By Lemma 3.4 we can take its limit at $k \to \infty$. Note that the encoding for output tapes is so designed that this limit value indeed stands for the appropriate infinite string under the same encoding as infinite input strings.

So it remains to show that we have recursive functions for input and output. For the integer part, the functions we need to show recursive are

$$\mathrm{iinput}\colon n \mapsto \sum_{i=0}^{n-1} \left(\frac{1}{5}\right)^i \qquad \text{and} \qquad \mathrm{ioutput}\colon \sum_{i=0}^{n-1} \left(\frac{1}{5}\right)^i \mapsto n \qquad (n \in \mathbb{N}).$$

For iinput, see the example after Lemma 3.3. For ioutput, let $\mathrm{ioutput}(t) = \mu n.\,\big(\mathrm{integer?}(n) + \mathrm{nonnegative?}(n) + \mathrm{digit}(t, 5, -n)\big)$ where $\mathrm{nonnegative?}(n) = 1 - \mathrm{zero?}\big(\pi/2 - \mu\theta.\,(\tan^2\theta - n)(\theta - \pi/2)\big)$. To input the fractional part, we need a function finput such that for any $t \in [0;1]$ there are $a_0, a_1, \ldots \in \{0,1,2\}$ with

$$t = \sum_{i=0}^{\infty} \overline{a_i} \left(\frac{1}{2}\right)^i \qquad \text{and} \qquad \mathrm{finput}(t) = \sum_{i=0}^{\infty} (\overline{a_i} + 1) \left(\frac{1}{5}\right)^i.$$

To output the fractional part, we need

$$\mathrm{foutput}\colon \sum_{i=0}^{\infty} (\overline{a_i} + 1) \left(\frac{1}{5}\right)^i \mapsto \sum_{i=0}^{\infty} \overline{a_i} \left(\frac{1}{2}\right)^i \qquad (a_i \in \{0,1,2\}).$$

To see these functions are recursive, define

$$\mathrm{finput}(t) = \sum_{i=0}^{\infty} \left(\frac{1}{5}\right)^i \big(\mathrm{digit}(t, 2, -i) + 1\big),$$
$$\mathrm{foutput}(t) = \sum_{i=0}^{\infty} \left(\frac{1}{2}\right)^i \big(\mathrm{digit}(t, 5, -i) - 1\big),$$

using Lemma 3.4 for infinite summations.

Now that we are done with our simulation, we have shown that every type-2 computable real function is recursive.

6 Summary and Future Work

We studied Moore's classes of primitive recursive and recursive functions, which are characterized by simple closure properties under a few operations. They may be considered as an analog, continuous-time computation model, as opposed to the type-2 computable functions which can be characterized as those computed by digital stream conversion between names.

We have seen two-way inclusion results that link between these concepts from digital and analog models: all primitive recursive functions are computable, and all computable functions are recursive.

Unfortunately, the three classes for which we have established the order of inclusion are not so close to each other. As we have noted, all primitive recursive functions are analytic, while in general computable functions are not; all computable functions are continuous, while in general recursive functions are not. Thus, we believe that there is still much to be done before we can relate these computation models in a more satisfactory way. One direction would be to somehow restrict the demonic zero-finding operator that makes the class of recursive functions fairly large, in order to get a more down-to-earth concept of recursiveness. Another possibility is to pursue a characterization of the primitive recursive functions in the framework of type-2 computability.

Acknowledgement

The author is grateful to Masami Hagiya, Izumi Takeuti, Hideki Tsuiki and the anonymous referee for insightful comments and suggestions.

References

[1] Campagnolo, M. L., *The complexity of real recursive functions*, in: C. S. Calude et al., editors, *Unconventional Models of Computation*, Lecture Notes in Computer Science **2509** (2002), pp. 1–14.

[2] Graça, D., "The General Purpose Analog Comoputer and Recursive Functions Over the Reals," Master's thesis, Instituto Superior Técnico (2002).

[3] Lang, S., "Real and Functional Analysis," Graduate Texts in Mathematics **142**, Springer-Verlag, 1993, third edition.

[4] Moore, C., *Recursion theory on the reals and continuous-time computation*, Theoretical Computer Science **162** (1996), pp. 23–44.

[5] Odifreddi, P., "Classical Recursion Theory," Studies in logic and the foundations of mathematics **125**, Elsevier Science Publishers B.V., 1989.

[6] Pontryagin, L. S., "Ordinary Differential Equations," Addison-Wesley, 1962.

[7] Weihrauch, K., "Computable Analysis: An Introduction," Texts in Theoretical Computer Science, Springer-Verlag, 2000.