Recursion Theory on the Reals and Continuous-time Computation

Cristopher Moore
Santa Fe Institute

September 26, 1995

Abstract

We define a class of recursive functions on the reals analogous to the classical recursive functions on the natural numbers, corresponding to a conceptual analog computer that operates in continuous time. This class turns out to be surprisingly large, and includes many functions which are uncomputable in the traditional sense.

We stratify this class of functions into a hierarchy, according to the number of uses of the zero-finding operator μ. At the lowest level are continuous functions that are differentially algebraic, and computable by Shannon's General Purpose Analog Computer. At higher levels are increasingly discontinuous and complex functions. We relate this μ-hierarchy to the Arithmetical and Analytical Hierarchies of classical recursion theory.

1 Introduction

Classical computation theory deals with sets of bit strings, or equivalently functions on the natural numbers $f : \mathbf{N} \to \mathbf{N}$; it allows us to discuss computation and complexity in a discrete, digital world. But to discuss the physical world (or at least its classical limit) in which the states of things are described by real numbers and processes take place in continuous time, we need a different theory: a theory of analog computation, where states and processes are inherently continuous, and which treats real numbers not as sequences of digits but as quantities in themselves.

In this paper, we define a set of functions on the reals $\mathbf{R}$ analogous to the classical recursive functions on $\mathbf{N}$. We start by showing that standard mathematical functions are computable in this system, as well as numbers such as e and π; we then show that this definition of computability corresponds to a conceptual programming language in which for and while loops run in continuous time.

This conceptual computer is almost certainly unphysical, since energy or other quantities related to the variables and their derivatives would go to infinity during the course of a computation. To address the degree of unphysicality, we stratify our class of functions according to the number of uses of the zero-finding operator μ. The lowest level of this hierarchy coincides with Shannon's General Purpose Analog Computer [7], while the upper levels reach into the Arithmetical and Analytical Hierarchies of classical recursion theory [1] and include many classically uncomputable functions.

The surprising power of this system comes from the fact that, unlike **N**, **R** can be mapped into a compact subset of itself. This allows us to construct an operator η, which searches over all of **R** "in finite time", rendering the Halting Problem decidable.

2 Comparison with Other Models of Computation on the Reals

The best-known recent model of analog computation is Blum-Shub-Smale's [4]. They define flowchart machines whose states are finite-dimensional vectors of real or complex numbers, whose elementary operations are rational functions, and which can branch on polynomial inequalities.

Our definition differs from this one in several ways. First, if we're going to discuss computation with continuous states, it seems more appropriate to define a model with continuous time, rather than discrete steps. Secondly, their choice of rational functions as elementary operations may make sense for rings in general, but it seems rather arbitrary for a model of computation on **R**. For instance, e^x takes an infinite number of steps to compute, while it is one of the easiest functions to define in our system.

Several authors have looked at the BSS theory when linear or trigonometric functions are used [5], or from the point of view of recursive functions and descriptive complexity [6], but still with discrete recursion rather than continuous integration.

Our system is closer in spirit to Shannon's "General Purpose Analog Computer" [7], which has addition, multiplication and integration as fundamental operations, and Rubel's "Extended Analog Computer" [11] which also has operations which solve boundary-value problems and take certain infinite limits. Our system differs from Rubel's EAC in that we have a zero-finding operator μ, analogous to the μ of standard recursion theory; the lowest level of our hierarchy corresponds almost exactly to Shannon's GPAC.

3 Classical Recursion Theory

Traditional recursion theory [1] defines a set of *computable* or *recursive* functions on the natural numbers $\mathbf{N}$ in the sense that they can be generated from a set of elementary functions using certain reasonable rules. The initial functions are

$$\mathcal{O}(x) = 0$$

$$\mathcal{S}(x) = x + 1$$

$$\mathcal{I}_i^n(x_1, \ldots, x_n) = x_i \ \ (1 \leq i \leq n)$$

namely the constant zero, the successor function, and projection functions from vectors $\vec{x}$ to their components (of which the identity function $\mathcal{I}_1^1(x) = x$ is a special case). We are then allowed to define new functions h in terms of existing ones f and g through three operators:

1. *Composition:* $h(\vec{x}) = f(g(\vec{x}))$.

2. *Primitive Recursion:* $h(\vec{x}, 0) = f(\vec{x})$, $h(\vec{x}, y+1) = g(\vec{x}, y, h(\vec{x}, y))$. That is, let $h = f$ when $y = 0$, and then inductively define $h(y+1)$ in terms of $h(y)$, y, and the other variables $\vec{x}$.

3. *μ-Recursion:* $h(\vec{x}) = \mu_y f(\vec{x}, y) = \min\{y | f(\vec{x}, y) = 0\}$, i.e. $h(\vec{x})$ is the smallest y such that $f(\vec{x}, y) = 0$.

For instance, using primitive recursion we can define addition as

$$h(x, 0) = \mathcal{I}_1^1(x), \ \ h(x, y+1) = \mathcal{S}(h(x, y))$$

or, in plainer language,

$$x + 0 = x, \ \ x + (y+1) = (x+y) + 1$$

Functions that can be generated from $\mathcal{O}$, $\mathcal{S}$ and $\mathcal{I}$ with just composition and primitive recursion are called *primitive recursive.* Most simple arithmetic functions, such as multiplication, exponentiation, and a characteristic function for primeness ($\chi_p(x) = 1$ if x is prime and 0 otherwise) are primitive recursive; they correspond to programs with only `for` loops, which always halt [1].

The μ-operator, on the other hand, corresponds to a `while` loop, since it searches arbitrarily large values of y to find one such that $f(\vec{x}, y) = 0$. Of course, there may be no such y, and the `while` loop might never halt. Then f is undefined on that value of $\vec{x}$, and is only a partial function. The functions that can be generated with all three rules are called *partial recursive*; if a function is total, i.e. defined for all $\vec{x}$, it is simply *recursive.*

The recursive functions turn out to correspond exactly to many other definitions of computability, including Turing machines [2], λ-calculus, flowchart programs, Post's tag system, and so on [1]. For this reason this class of functions is considered a deep and universal definition of computability; this is the *Church-Turing Thesis.*

To get larger classes of functions, we can extend our model of computation with *oracles*, giving an infinite hierarchy of increasingly uncomputable functions. The *Arithmetical and Analytical Hierarchies* will appear in our discussion below.

We also speak of computable *sets*: for every set S we define a *characteristic function*, $\chi_S(x) = 1$ if $x \in S$ and 0 otherwise. We say S is a computable set if χ_S is a computable function.

4 Recursion on the Reals

In analogy with the recursive functions on $\mathbf{N}$, we define the following set of functions on $\mathbf{R}$:

Definition. A function $f\colon \mathbf{R}^m \to \mathbf{R}^n$ is $\mathbf{R}$-*recursive* if it can be generated from the constants 0 and 1 with the following operators: if f and g are already defined, then so is

1. *Composition:* $h(\vec{x}) = f(g(\vec{x}))$

2. *Differential Recursion* or simply *Integration:* $h(\vec{x}, 0) = f(\vec{x})$, $\partial_y h(\vec{x}, y) = g(\vec{x}, y, h(\vec{x}, y))$. In other words, let $h = f$ at $y = 0$, and then let the derivative of h with respect to y depend on $h(y)$, y, and $\vec{x}$. We also write

$$h(\vec{x}, y) = f(\vec{x}) + \int_0^y g(\vec{x}, y', h(\vec{x}, y'))\, dy'$$

or $h = f + \int g$ for short.

3. μ-*Recursion* or *Zero-Finding:* $h(\vec{x}) = \mu_y f(\vec{x}, y) = \inf\{y | f(\vec{x}, y) = 0\}$, where the infimum chooses the y with smallest absolute value, and (by convention) the negative one if there are two y with the same absolute value.

4. Vector-valued functions can be defined by defining their components.

Several comments are in order. First, integration seems to be the closest continuous analog to primitive recursion; defining $h(\vec{x}, y+1)$ in terms of $h(y)$, y, and $\vec{x}$ becomes defining $\partial h/\partial y$ in terms of $h(y)$, y and $\vec{x}$. There are two problems with this: a solution to the differential equation need not be unique, as in $f(0) = 0$, $df/dx = 2f/x$, which is solved by $f(x) = ax^2$ for any a. In addition, the function can diverge, such as $g(0) = 0$, $dg/dx = g^2 + 1$, for which $g = \tan x$ is only defined on the interval $(-\pi/2, \pi/2)$. For simplicity, we will say that h is only defined where a finite and unique solution (that includes the point $h(\vec{x}, 0) = f(\vec{x})$) exists.

Secondly, as before, the μ operator finds the "smallest" y such that $f(\vec{x}, y) = 0$, but with two modifications. First, since the real line extends in two directions, we search outward from 0 and take the y closest to the origin (with smallest $|y|$); if there are two with the same $|y|$, we take the negative one by convention. Secondly, if an infinite number of zeroes accumulate just above some y (or just below some negative y) then μ returns that y even if it itself is not a zero. Equivalently, let $[a, b]$ be the largest closed interval containing zero in which

$f(y) \neq 0$; then μ returns whichever of a and b has the least absolute value, and a if $a = -b < 0$.

Finally, we have chosen to explicitly allow vector-valued functions. This isn't necessary in the integer case, since we can easily encode 2 or more integers into a single one with one-to-one recursive functions like $f\colon \mathbf{N} \times \mathbf{N} \rightarrow \mathbf{N} : f(x,y) = 2^x(2y+1)$. There are $\mathbf{R}$-recursive one-to-one maps from $\mathbf{R}^n$ to $\mathbf{R}$, but they are of course not differentiable. Since this interferes with our integration operation, and they take several uses of our operators to construct, we prefer to allow vector-valued functions at the outset.

We can now begin generating functions. First we derive the projection functions, since we only included the constants 0 and 1 in our initial set:

Proposition 1. *The projection functions $\mathcal{I}_i^n$ are $\mathbf{R}$-recursive.*

Proof. For $i = n$, let

$$\mathcal{I}_n^n(x_1, x_2, \ldots, x_{n-1}, 0) = 0, \;\; \partial_{x_n}\mathcal{I}_n^n(x_1, \ldots, x_n) = 1$$

Then for $n > i$ we proceed by induction on n,

$$\mathcal{I}_i^n(x_1, \ldots, x_{n-1}, 0) = \mathcal{I}_i^{n-1}(x_1, \ldots, x_{n-1}), \;\; \partial_{x_n}\mathcal{I}_i^n(x_1, \ldots, x_n) = 0$$

∎

Next we derive basic arithmetic and trigonometric functions.

Proposition 2. *The functions $f_+(x,y) = x+y$, $f_\times(x,y) = xy$, $1/x$, $f_\div(x,y) = x/y$, e^x, $\ln x$, x^y, $\sin x$, $\cos x$, $\tan x$, $x \bmod y$, and the constants -1, e, and π are all $\mathbf{R}$-recursive.*

Proof. Let

$$f_+(x,0) = x, \;\; \partial_y f_+(x,y) = 1$$

(where of course by x we mean $\mathcal{I}_1^1(x)$)

$$f_\times(x,0) = 0, \;\; \partial_y f_\times(x,y) = x$$

$$-1 = \mu_y(y+1)$$

(where by $y+1$ we mean $f_+(y,1)$)

$$1/x = g(x-1) \text{ where } g(0) = 1, \;\; \partial_y g(y) = -g^2(y)$$

(where by $x-1$ we mean $f_+(x,-1)$, and by $-g^2$ we mean $f_\times(-1, f_\times(g,g))$)

$$f_\div(x,y) = f_\times(1/y, x)$$

$$e^0 = 1, \;\; \partial_y e^y = e^y, \;\; e = e^1$$

$$\ln x = h(x-1) \text{ where } h(0) = 0, \;\; \partial_y h(y) = 1/(y+1)$$

$$x^0 = 1, \;\; \partial_y x^y = x^y \ln x$$

$$\tan 0 = 0, \;\; \partial_y \tan y = \tan^2 y + 1, \;\; \pi = 4\mu_y(\tan y - 1)$$

$$\begin{pmatrix} \sin \\ \cos \end{pmatrix}(0) = \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \quad \partial_y \begin{pmatrix} \sin \\ \cos \end{pmatrix}(y) = \begin{pmatrix} \cos \\ -\sin \end{pmatrix}(y)$$

$$x \bmod y = \mu_z [\cos 2\pi \left(\frac{x-z}{y} - 1/2 \right) - 1] + y/2$$

This rather ungainly definition for $x \bmod y$ is built to make the proper z the one closest to the origin, according to our conventions for μ. $\blacksquare$

Several of these are partial functions. For instance, there are two ways we can define $1/x$: the integration given above, and

$$1/x = \mu_y(xy - 1)$$

Both of these will be undefined for $x = 0$: in the first case because $g(x)$ diverges at $x = -1$ and so is undefined for $x \leq -1$, and in the second case because μ will not find a suitable y. We will see below how to patch these partial functions to produce a reciprocal function $\widehat{1/x}$ that is everywhere defined.

5 R-recursive Reals

From Proposition 2, we have that e and π are computable by the following definition:

Definition. A number $x \in \mathbf{R}$ is **R**-*recursive* if $x = f(0)$ for some **R**-recursive function f. We will call the set of **R**-recursive reals $\mathcal{K}$.

(Note that $x = f(y)$ is also **R**-recursive if f and y are, since $x = g(0)$ where $g(y) = f(x + y)$.)

Now what we mean here is not that e and π have digit sequences which can be computed sequentially (the traditional definition of computable real) but rather that they can be generated as exact, inherently analog quantities.

Clearly $\mathcal{K}$ is countable, since we generate functions from 0 and 1 with a finite number of applications of the three operators. So most reals are not in $\mathcal{K}$; which ones are?

Proposition 3. *$\mathcal{K}$ contains the integers* $\mathbf{Z}$*, the rationals* $\mathbf{R}$*, and the algebraic numbers.*

Proof. Any integer is the sum of finitely many 1s or -1s. Any rational p/q is $\mu_y(qy - p)$. Any polynomial $P(x)$ with rational coefficients is **R**-recursive since f_+, $f_\times$, and x^y are, so any algebraic root r of P is $\mu_y P(y-x) + x$ for some rational x which is closer to r than to any other root. $\blacksquare$

Obviously $\mathcal{K}$ contains many other numbers, such as the roots of transcendental equations like $2x - \tan x = 0$, transcendental numbers such as $\sqrt{2}^{\sqrt{2}}$, and so on. Later, we will find out that $\mathcal{K}$ also contains many numbers which are not computable in the usual sense.

6 δ, $|x|$, and Θ

So far, we have only generated continuous functions (at least on the domain on which they are defined). In this section, we introduce some useful discontinuous ones.

Consider the absolute value $|x|$, the step function $\Theta(x) = 1$ if $x \geq 0$ and 0 if $x < 0$, $\max(x, y) = x$ if $x \geq y$ and y otherwise, and the Kronecker δ-function $\delta(x) = 1$ if $x = 0$ and 0 otherwise. Then we have

Proposition 4. *The functions $\delta(x)$, $|x|$, $\Theta(x)$ and $\max(x, y)$ are all **R**-recursive.*

Proof. Generating δ is an interesting exercise:

$$\delta(x) = 1 - \mu_y((x^2 + y^2)(y - 1))$$

There is always a root at $y = 1$, but if $x = 0$ then $y = 0$ is also a root, μ returns it since it is closer to 0, and $\delta = 1 - 0 = 1$. If $x \neq 0$, μ returns $y = 1$ as the only (real) root and $\delta(x) = 1 - 1 = 0$.

Furthermore, let $|x| = -\mu_y(x^2 - y^2)$, and $\operatorname{sgn}(x) = \mu_y(yx - |x|) = 0$ if $x = 0$, 1 if $x > 0$, and -1 if $x < 0$. Then $\Theta(x) = ((\operatorname{sgn}(x) + \delta(x)) + 1)/2$ and $\max(x, y) = x\,\Theta(x - y) + y\,\Theta(y - x)$. ∎

Θ allows us to do bounded searches. For instance, we can define

$$\mu_{y \geq 0} f(\vec{x}, y) = \mu_y \left(\Theta(y)(1 - f(\vec{x}, y)) - 1\right)$$

or

$$\mu_{y \in [a,b)} f(\vec{x}, y) = \mu_y \left(\Theta_{[a,b)}(y)(1 - f(\vec{x}, y)) - 1\right)$$

where

$$\Theta_{[a,b)}(x) = \Theta(x - a)(1 - \Theta(x - b))$$

and similarly for other closed or open intervals.

With δ, we can now patch our reciprocal function to make it defined everywhere. Consider the function

$$\widehat{1/x} = f_\times(x^{-1}, 1 - \delta(x))$$

We claim that this is always defined: even though the integral for x^{-1} diverges when $x = 0$, we don't need to evaluate it since $f_\times(y, 0)$ is defined to be 0 regardless of what y is. So if $x = 0$ this function simply returns 0.

To make this rigorous, we need to introduce a semantics to the system, which we will do through a programming model.

7 A Programming Model: Continuous-Time Flowchart Programs

Consider the following (rather fanciful) **PASCAL**-like program, which operates with a continuous time parameter t:

```
function $h(\vec{x}, y : \text{real}) : \text{real}$
var $y'$ : real
   $h := f(\vec{x})$;
   for $y' = 0$ to $y$ do
      $\partial_t h := g(\vec{x}, y', h)$;
   return($h$);
```

This program calculates the function $h = f + \int g$ (we can make the loop run backwards if $y < 0$). As in conventional for loops, let us say that if $y = 0$ the loop never executes; then, as we claimed above, $f_\times(y, 0) = 0$ even if y is undefined.

Similarly, the program

```
function $h(\vec{x} : \text{real}) : \text{real}$
var $y$ : real
   $y := 0$;
   while $f(\vec{x}, y) \neq 0$ and $f(\vec{x}, -y) \neq 0$ do
      $\partial_t y := 1$;
   if $f(\vec{x}, -y) = 0$ then return($-y$)
      else return($y$);
```

calculates $h = \mu_y f$. As in discrete recursion theory, if there is no y such that $f(\vec{x}, y) = 0$, the while loop never halts and h is undefined.

Clearly, programs with these kinds of loops, with semantics as we've defined, calculate precisely the **R**-recursive functions. However, we are not proposing this "programming language" as a physically realistic model of continuous-time computation, for several reasons.

First, what happens if we try (non-rigorously) to define a computation time for these programs? The for loop in $h(y) = f + \int g(y')dy'$ runs "y times", and the while loop in $h(x) = \mu_y f(x, y)$ runs "h times" (or ∞ if μ is undefined). It seems reasonable to think, then, that an analog computer could calculate the first in time proportional to y, and the second in time proportional to h. But in fact both these loops call on the functions g and f an uncountable number of times; so if it takes nonzero time to calculate each value of f and g, h's computation time will be uncountably infinite.

Secondly, and perhaps equivalently, in the next section we will see examples where the variables and their derivatives go to infinity during the course of a computation. Any physical realization of such a computer would presumably run out of resources or explode.

Finally, the μ-operator itself is fundamentally unrealistic, since it finds zeroes even when they are completely isolated and discontinuous, such as a function which is 1 everywhere except at a particular y. Any noise or coarse-graining, inevitable in a real physical computer, would make this impossible.

For now, let us simply explore the set of **R**-recursive functions for its own sake, and see what else it contains.

8 $\chi_{\mathbf{Z}}$, $\chi_{\mathbf{Q}}$, η, and the Compression Trick

The fact that $\mathbf{Z} \subset \mathcal{K}$, i.e. integers are recursive reals, should not be confused with the statement that the set of integers $\mathbf{Z}$ is a recursive set. The first means that any particular integer is recursive, while the second means there is a recursive function, $\chi_{\mathbf{Z}}$, which tells whether a given real is an integer or not. We now show that $\mathbf{Z}$ and $\mathbf{Q}$ are in fact **R**-recursive sets; in the process, we will use a trick that has no analog in discrete recursion theory.

Proposition 5. *$\mathbf{Z}$ and $\mathbf{Q}$ are* ***R**-recursive sets.*

Proof. For $\mathbf{Z}$, let $\chi_{\mathbf{Z}}(x) = \delta(\sin(\pi x))$.

Deriving $\chi_{\mathbf{Q}}$ is somewhat harder. Our first attempt might be

$$\mu_q(\chi_{\mathbf{Z}}(q)\chi_{\mathbf{Z}}(xq) - 1)$$

that is, find a q such that q and xq are both integers. This returns the denominator q if one exists, i.e. if $x = p/q$; but the question is whether or not one exists. More precisely, x is rational if this μ is well-defined.

Let $g(y) = 1 - 1/y$ and $g^{-1}(y) = 1/(1-y)$ map the interval $[1, \infty)$ to $[0, 1)$ and back. Now write

$$\chi_{\mathbf{Q}}(x) = 1 - \delta\big(\mu_{y \geq 0}[(\chi_{\mathbf{Z}}(g^{-1}(y))\,\chi_{\mathbf{Z}}(xg^{-1}(y)) - 1)(y-1)] - 1\big)$$

We will now show this is a correct formula for $\chi_{\mathbf{Q}}$. Suppose x is rational with denominator q. Then μ will find a zero at $y = g(q)$ since both q and xq are integers. Since $y < 1$, $\chi_{\mathbf{Q}} = 1 - \delta(y-1) = 1$. If on the other hand x is irrational, there are no zeroes less than $y = 1$ — but $y = 1$ is always a zero, because of the factor of $y - 1$. So μ returns 1, and $\chi_{\mathbf{Q}} = 1 - \delta(1-1) = 0$. ■

Here we have compressed an infinite interval into a finite one, and placed an additional zero at the end of it; if there is no solution within the interval, μ finds the one at the end and halts. Thus we learn "within finite time" that no solution exists. We have done something that recursion on $\mathbf{N}$ cannot do, namely transform a search over all of $\mathbf{R}$ into a search over a compact subset.

In general, let us define a new operator, η:

Definition. For any function $f(\vec{x}, y)$, let

$$\eta_y f(\vec{x}, y) = \begin{cases} 1 \text{ if } \exists y : f(\vec{x}, y) = 0 \\ 0 \text{ if } \forall y : f(\vec{x}, y) \neq 0 \end{cases}$$

In other words, $\eta_y f = 1$ is a characteristic function for the set of $\vec{x}$ on which $\mu_y f$ is well-defined. We can also consider a bounded search $\eta_{y \in I} f$ for some interval I; in the proof above, $I = [1, \infty)$.

Then by generalizing Proposition 5, we have

Proposition 6. *If $f(\vec{x}, y)$ is* **R***-recursive, so is $\eta_y f(\vec{x}, y)$.*

Proof. Let

$$\eta_y f(\vec{x}, y) = 1 - \delta\big(\mu_y [f(\vec{x}, g^{-1}(y))(y - g(\infty))] - g(\infty)\big)$$

where g is a compression function that maps the real line $(-\infty, \infty)$ to a finite interval $(g^{-1}(-\infty), g^{-1}(\infty))$; for instance, if $g(x) = \tan^{-1}(x)$, then

$$\eta_y f(\vec{x}, y) = 1 - \delta\big(\mu_y (f(\vec{x}, \tan y)(y - \pi/2)) - \pi/2\big)$$

Then if μ is well-defined, we find a $y \in (-\pi/2, \pi/2)$ and $\eta = 1 - \delta(y + \pi/2) = 1$. If no y exists, then μ finds the zero at $\pi/2$ and $\eta = 1 - \delta(\pi/2 - \pi/2) = 0$. ∎

We can use η to patch any partial function $h = \mu_y f$ so that it becomes a total one: $h_{\text{tot}} = (\mu_y f)(\eta_y f)$ is defined everywhere, and $h = h_{\text{tot}}$ wherever h is defined.

This "Compression Trick" should certainly raise the reader's eyebrows, and in the next sections we will see that the existence of η makes the class of **R**-recursive functions rather powerful. But since $g^{-1}(y)$ goes to infinity during the course of the computation, η is almost certainly unphysical; so let us define a hierarchy to stratify the **R**-recursive functions according to their "unphysicality" — the number of uses of μ.

9 The μ-hierarchy

Definition. For a given **R**-recursive expression $s(\vec{x})$, let the *μ-number with respect to x_i* $M_{x_i}(s)$ be defined recursively as follows:

$$M_x(0) = M_x(1) = M_x(-1) = 0$$

$$M_x(f(g_1, g_2, \ldots)) = \max_j (M_{x_j}(f) + M_x(g_j))$$

$$M_x\big(h = f + \int_0^y g(\vec{x}, y', h)dy'\big) = \max(M_x(f), M_x(g), M_h(g))$$

$$M_y\big(h = f + \int_0^y g(\vec{x}, y', h)dy'\big) = \max(M_{y'}(g), M_h(g))$$

$$M_x(\mu_y f(\vec{x}, y)) = \max(M_x(f), M_y(f)) + 1$$

In other words, $M_x(s)$ is the depth of nested μs surrounding x in s. Then for an **R**-recursive function f, let $M(f) = \max_i M_{x_i}(s)$, minimized over all expressions s that define f. Finally, define the *μ-hierarchy* as the sets $M_j = \{f | M(f) \leq j\}$;

clearly $\cup_j M_j$ is the set of **R**-recursive functions. We also say that a set s is in M_j if χ_s is.

For instance, all the functions in Proposition 2 are in M_0 except $x \bmod y$. Both e and π are in M_0, although the definition of π given above is M_1; we can use $4\tan^{-1} 1$ instead, as we will see below. The functions $|x|$, Θ, and δ are in M_1. (We define $M(-1) = 0$ so that all of **Z** and **Q** will be in M_0; otherwise $-1 = \mu_x(x+1)$ would be in M_1 and so would all negative integers.)

We add

Lemma 7. *If f is in M_j, then $\eta_y f$ is in M_{j+2}.*

Proof. Our expression for η in Proposition 6 nests f inside a δ and a μ, so M increases by 2 (assuming we use a M_0 compression function g such as $\tan^{-1}$.) $\blacksquare$

We can prove a few things about the first few levels of this hierarchy.

9.1 M_0, Differentially Algebraic functions, and Shannon's GPAC

Definition. A function $f(x)$ is *differentially algebraic* [9] or DA if its derivatives satisfy a polynomial equation $P(x, f(x), f'(x), \ldots, f^{(k)}(x)) = 0$ for some polynomial P with rational coefficients. A function of several variables is DA if it is a DA function of each variable when the others are held fixed. Non-DA functions are called *transcendentally transcendental.*

Differentially algebraic functions are closed under addition, multiplication, composition, inversion (where the inverse is continuous), differentiation, and integration. Most commonly used functions in mathematics are DA, including trigonometric and Bessel functions; exceptions are the ?-function and Riemann's ζ-function [8], and solutions to the Dirichlet problem on a disk [9], even when the boundary conditions are DA.

Shannon [7] defined a *General Purpose Analog Computer* (GPAC), which consists of circuits containing "black boxes" which add, multiply, integrate, and provide constants and the running time t. The outputs of GPACs turn out to be precisely the DA functions [10].

The lowest level of our hierarchy, M_0, consists of the "primitive **R**-recursive" functions; those definable from 0 and 1 by composition and integration alone, without μ. Then

Lemma 8. *M_0 is closed under differentiation and inversion, with f^{-1} defined on an interval where f is continuous.*

Proof. For differentiation, we proceed inductively, starting from $0' = 1' = 0$. For composition, $f(g(x))' = f'(g(x))g'(x)$. For integration, if $h(x,y) = f(x) + \int_0^y g(x,y',h)dy'$, then $\partial_y h = g(x,y,h)$ and $\partial_x h = \partial_x f + \int_0^y \partial_x g(x,y',h)dy'$. So if f is in M_0, f' is too.

Now suppose $g = f^{-1}$ where f is **R**-recursive. Then $g'(x) = 1/f'(g(x))$ is **R**-recursive since f' and $1/x$ are, and $g(x) = 0 + \int_0^{x-f(0)} g'(x' + f(0))dx'$ is in

M_0 (if $f(0)$ is undefined, we can choose a different base point to integrate from). ∎

Corollary. $\pi = 4\tan^{-1} 1$ *is in* M_0.

Proposition 9. *Functions in* M_0 *are differentially algebraic and are analytic on the domain in which they are defined. Therefore, functions in* M_0 *are also computable by Shannon's GPAC. Conversely, any function computable by Shannon's GPAC from initial conditions in* M_0 *is also in* M_0.

Proof. Since 0 and 1 are DA and DA functions are closed under composition and integration, $M_0 \subset DA$. Since analyticity is also preserved by composition, and by integration when we only allow unique solutions, functions in M_0 are also analytic.

Conversely, we show that the unique solution of any algebraic differential equation (ADE) is in M_0. Polynomials are in M_0, and by Lemma 8 so are their inverses; so we can invert any ADE to solve for the highest derivative $f^{(k)}$ in terms of the lower ones. Then we can transform this to a first-order system of equations in k variables in the usual way, and integrate it; if the initial values $f(0), f'(0), \ldots, f^{(k-1)}(0)$ are in M_0, clearly the integral is too. ∎

9.2 M_1 and Discontinuities

At least one μ is needed, then, to get a discontinuous function. We can set an upper limit on the amount of discontinuity that can appear in M_1:

Proposition 10. *A function in* M_1 *can have at most a countable number of discontinuities, and a countable number of discontinuities in its derivatives, along any one variable. In between these continuities, the function must be analytic.*

Proof. Let $h(\vec{x}) = \mu_y f(\vec{x}, y)$ where $f \in M_0$. If we fix all components of $\vec{x}$ but one, call it x, the set $C = \{(x,y) | f(\vec{x}, y) = 0\}$ is an analytic curve in two dimensions. As shown in figure 1, discontinuities in $h(x) = \min\{y | (x,y) \in C\}$ occur where $dx/d\tau = 0$ where τ is arc length, and discontinuities in $\partial h'/\partial x$ occur where C self-intersects. Since C is analytic curve, both of these can only happen a countable number of times; in between, h coincides with a section of C and so is analytic.

Furthermore, if h_1 and h_2 are two such functions with sets of discontinuities S_1 and S_2, the discontinuities of their composition $h_1 h_2$ are contained in $S_1 \cup h_1^{-1}(S_2)$. The preimages $h_1^{-1}(s)$ occur where C intersects the line $y = s$; since C is analytic there are a countable number of such intersections for each $s \in S_2$, resulting in a countable total number of discontinuities.

Finally, integrating such a function only introduces discontinuities in its higher derivatives. ∎

Corollary. $\chi_{\mathbf{Q}}$, the characteristic function of the rationals, is in $M_2 - M_1$.

Proof. We actually don't need the discontinuous $\chi_{\mathbf{Z}}$ we used for $\chi_{\mathbf{Q}}$ in

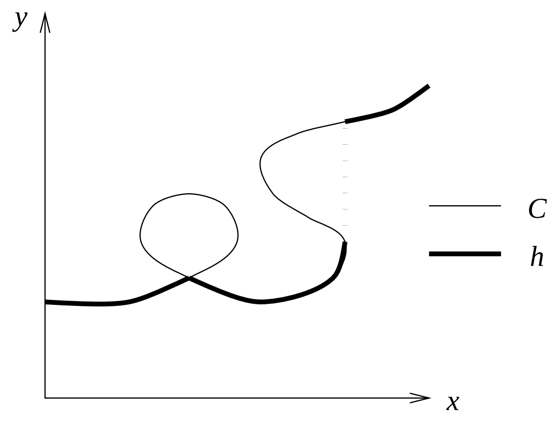

Figure 1: If $h(x) = \mu_y f(x, y)$ where f is in M_0, h and its derivatives can only have a countable number of discontinuities since the curve $C = \{(x, y)|f(x, y) = 0\}$ is analytic.

Proposition 5; we can write it as

$$\chi_{\mathbf{Q}}(x) = \eta_y[(1 - \sin^2 \pi xy)(1 - \delta_{\mathbf{Z}}(y)) - 1]$$

where

$$\delta_{\mathbf{Z}}(y) = \frac{\sin^2 \pi y}{(\pi y)^2}$$

$\delta_{\mathbf{Z}}(y) = 0$ if and only if y is a nonzero integer. Since $\delta_{\mathbf{Z}}$ is in M_0, by Lemma 7 $\chi_{\mathbf{Q}}$ is in M_2; but it can't be in M_1 since it's everywhere discontinuous. ∎

We also see here another difference between **N**-recursion and **R**-recursion; any partial **N**-recursive function can be expressed with just one μ, using a function that embodies a universal Turing machine [1]. On **R**, we have a function that requires at least two μs; we will discuss below whether a universal **R**-recursive function exists.

10 Iterated Maps, Turing Machines, and Halting Problems

We next show that an iterated **R**-recursive function is **R**-recursive:

Proposition 11. *For a function $F(\vec{x})$, define $F_{\text{iter}}(t, \vec{x}_0) = F^t(\vec{x}_0)$, the tth iteration of F on $\vec{x}_0$. Then if F is* **R***-recursive, so is F_{iter}, and if $F \in M_j$, then $F_{\text{iter}} \in M_{\max(j,1)}$.*

Proof. First we define two "clock" functions, r and s:

$$s(t) = \Theta(\sin(\pi t))$$
$$r(0) = 0, \;\; \partial_t r(t) = 2s(t) - 1$$

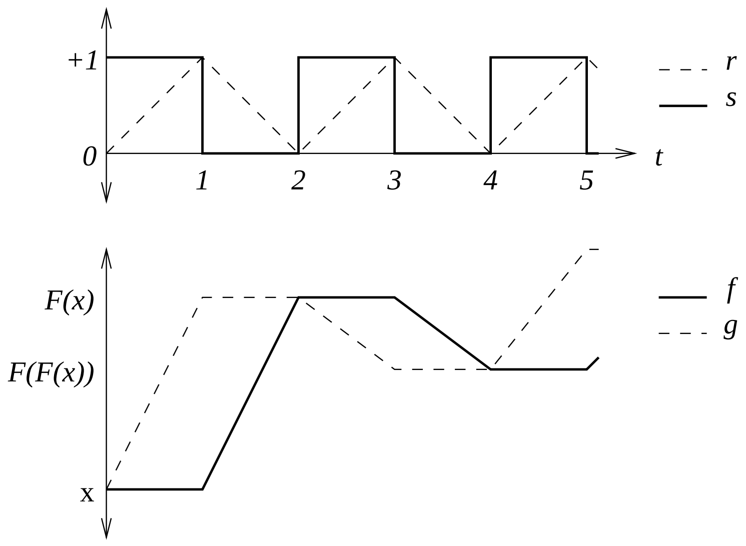

Figure 2: The iteration scheme of Proposition 11, using r and s as clock functions. $F^n(x) = f(2n)$.

Here $s(t)$ is a square wave, 1 if $t \in [2m, 2m+1]$ and 0 if $t \in (2m+1, 2m+2)$ for integer m, and $r(t)$ is a sawtooth, going between 0 at even integer t and 1 at odd t. Then let

$$\vec{g}(0) = \vec{f}(0) = \vec{x}_0$$

$$\partial_t \vec{g}(t) = \big(F(\vec{f}(t)) - \vec{f}(t)\big)s(t)$$

$$\partial_t \vec{f}(t) = \frac{\vec{g}(t) - \vec{f}(t)}{r(t)}(1 - s(t))$$

At $t = 0$, both $\vec{f}$ and $\vec{g}$ start at $\vec{x}_0$. As t ranges from 0 to 1, $s = 1$, so that $\vec{f}$ is held fixed and $\vec{g}$ closes the distance from $\vec{x}_0$ to $F(\vec{x}_0)$. Then as t continues from 1 to 2, $s = 0$, $\vec{g}$ stays constant and $\vec{f}$ catches up, and at $t = 2$ both are at $F(\vec{x}_0)$. Then the cycle begins again.

For integer n, then, $F_{\text{iter}}(n, \vec{x}_0) = \vec{f}(2n)$ as shown in figure 2. And if $F(\vec{x})$ is in M_j, then $F_{\text{iter}}(t, \vec{x})$ is in $M_{\max(1,j)}$ since r and s are in M_1. ■

We can also make F_{iter} into a step function on t, by defining

$$\lfloor n \rfloor = n - n \bmod 1$$

with $n \bmod 1$ defined as above. Then $\lfloor n \rfloor$ is the integer part of n, and we can define $F_{\text{iter}}(n, \vec{x}_0) = \vec{f}(2\lfloor n \rfloor)$ (but this is now in M_2). We will refer to $F_{\text{iter}}(n, \vec{x})$ simply as $F^n(\vec{x})$ below.

Since we really just need integer values of t, we have implicitly used the following:

Definition. A function with some integer arguments is **R**-recursive if there is an **R**-recursive function equal to it when restricted to the integers.

We now refer to the fact [12, 13] that there exist maps in one or two dimensions whose actions simulate an arbitrary Turing machine, one step per iteration. These maps are in fact **R**-recursive:

Proposition 12. *There are M_1 functions of the unit square, and M_0 functions on the real line, which can simulate any Turing machine; the characteristic functions χ_{halt} of their halt states are also M_1 and M_0 respectively.*

Proof. In [12, 14] a piecewise linear map of the unit square is shown. It uses the digit sequences of a point's x and y coordinates as the left and right halves of a Turing machine's tape. To shift the tape to the left, for instance, we can use the commonly known Baker's Transformation

$$\vec{F}(x, y) = \big(2x \bmod 1,\ y/2 + (1/2)\,\Theta(x - 1/2)\big)$$

doubling x, halving y, and shifting the most significant digit of x to y. This map has two rectangular components, each mapped in an affine way; with a finite number of such components, corresponding to different combinations of Turing machine states and state symbols, we can simulate any Turing machine. If the ith rectangle is $[x_{i1}, x_{i2}] \times [y_{i1}, y_{i2}]$, we can write

$$\vec{F}(x, y) = \sum_{i=1}^{n} \Theta_{[x_{i1},x_{i2}]}(x)\,\Theta_{[y_{i1},y_{i2}]}(y)\,\vec{F}_i(x, y)$$

where $\Theta_{[a,b]}(x) = \Theta(x - a)\,\Theta(b - x)$ and

$$\vec{F}_i(x, y) = (a_i x + b_i, c_i y + d_i)$$

A halt state corresponds to a particular rectangle, so χ_{halt} is just another product of Θs. Both F and χ_{halt} are clearly M_1 in this case.

The analytic maps on the real line in [13] are sums of a finite number of trigonometric terms, and so are in M_0. They are in a halt state if $x \bmod p = s$ for a certain p and s, so $\chi_{\text{halt}}(x) = \delta(x \bmod p - s)$. Even better, write $\chi_{\text{halt}}(x) = (\sin(\pi x)/\sin(\pi(x - s)/p))^2$, which is in M_0 (we can get rid of the quotient by using the multiple angle formula). ∎

Proposition 13. *Any* **N**-*recursive set is in M_2.*

Proof. For any recursive set s, there is a machine M that always halts and reports whether the input x is in s or not. We can easily arrange for M to halt after an odd or even number of steps for $x \in s$ or $x \notin s$ respectively. Then

$$\chi_s(x) = (1 - \sin \pi\mu_t[\chi_{\text{halt}}(F_M^t(x)) - 1])/2$$

Using the M_0 maps for F_M and χ_{halt}, the μ is in M_2 since F_M^t is M_1. ∎

Proposition 14. *Any elementary* **N**-*recursive function [3], i.e. whose computation time is bounded by a function of the form* $2^{2^{\cdot^{\cdot^{\cdot^{2^x}}}}}$ *for some finite number of exponentials, is in* M_1.

Proof. Some machine M calculates f in bounded time. If we define $f_0(x) = x$ and $f_{k+1}(x) = 2^{f_k(x)}$, then $f_k(x)$ is in M_0 for any given k (although we conjecture that as a function of k and x it is not). Then

$$f(x) = F_M^{f_k(x)}(x)$$

is in M_1. ∎

Now we use η to solve the Halting Problem.

Proposition 15. *Any partial* **N**-*recursive set is in* M_3.

Proof. For a partial recursive set s, there is a machine M such that s is the set of inputs x on which M halts; but this is simply

$$\chi_s(x) = \eta_t(\chi_{\text{halt}}(F_M^t(x)) - 1)$$

F_M^t is M_1 and by Lemma 7 the η increases M by 2. ∎

To review: we can simulate Turing machines with **R**-recursive maps. We can use the μ operator to search for times t at which the machine halts. But with the "Compression Trick" and η, we can map all of **R** into a compact interval, scan it for all t, and render the Halting Problem decidable.

In the next sections, we will see that the η operator makes the class of **R**-recursive functions quite powerful.

11 The Arithmetical Hierarchy

Recall [1] that recursive and partial recursive functions are the bottom two levels of an infinite hierarchy of increasingly uncomputable sets, the *Arithmetical Hierarchy.* Sets at the jth level of this hierarchy can be specified with j alternating quantifiers, $\exists$ and $\forall$, applied to a recursive predicate. If the outermost quantifier is an $\exists$, the set is in Σ_j^0; complements of these sets, whose outermost quantifier is an $\forall$, are in Π_j^0; and Δ_j^0 is the intersection of these two. The union of all of these is written Δ_ω^0.

In particular, $\Delta_1^0 = \Delta_0^0$ is the recursive functions, and Σ_1^0 is the partial recursive functions, for which $\exists t$ such that M halts after t steps. The set of Turing machines M that halt on all inputs is in Π_2^0, since $\forall x\, \exists t$ such that M halts on input x after t steps.

We can also think of the hierarchy as adding successive oracles. A Turing machine with an oracle for solving the Halting Problem, which is in Σ_1^0, can compute recursive functions in Δ_2^0; that Turing machine has a halting problem of its own in Σ_2^0, and so on.

We now show that this entire hierarchy is in fact **R**-recursive, and lines up nicely with the μ-hierarchy:

Proposition 16. *The Arithmetical Hierarchy Δ^0_ω is **R**-recursive. In particular, $\Delta^0_j \subset M_{2j}$ and $\Sigma^0_j, \Pi^0_j \subset M_{2j+1}$ for all $j > 0$.*

Proof. By Proposition 15, $\Sigma^0_1 \subset M_3$; since $\chi_{\overline{s}} = 1 - \chi_s$, $\Pi^0_1 \subset M_3$ also. Then adding quantifiers to a characteristic function χ_P for some predicate P can be done by adding ηs:

$$\chi_{\exists a:P}(b, c, \ldots) = \eta_a(1 - \chi_P(a, b, c, \ldots))$$

$$\chi_{\forall a:P}(b, c, \ldots) = 1 - \eta_a \chi_P(a, b, c, \ldots)$$

or "is there an a such that $\chi_P = 1$?" and "is there no a such that $\chi_P = 0$?" respectively. Each η increases M by 2 according to Lemma 7, so $\Sigma^0_j, \Pi^0_j \subset M_{2j+1}$.

For instance, for a set S in Σ^0_3

$$S = \{x | \exists a : \forall b : \exists c : P(a, b, c, x)\}$$

where P is some property with recursive characteristic function χ_P, we would write

$$\chi_S(x) = \eta_a \eta_b \eta_c(\chi_P(a, b, c, x) - 1)$$

Then recall that Δ^0_j sets are recursive for a Turing machine that has an oracle for a Σ^0_{j-1}-complete problem, such as the halting problem for Turing machines at the Δ^0_{j-1} level. This oracle is a M_{2j-1} function, which can be patched into our Turing machine function in a M_1 way by calling this function whenever the Turing machine is in a certain state. Then we repeat Proposition 13 with F_M and F^t_M in M_{2j-1}, and $\Delta^0_j \subset M_{2j}$. ∎

Corollary. *$\mathcal{K}$ contains many real numbers which are not computable in the sense of Turing.*

Proof. Turing defined a real number as computable if its digit sequence is: specifically, if the nth digit can be calculated by a recursive function $g(n)$ which halts in recursive time. But for any **R**-recursive function f that maps integers to binary values, we can construct

$$x_f = \mu_x \eta_n \delta(d(n, x) - f(n))$$

where $d(n, x) = \lfloor 2^n x \rfloor \bmod 2$ gives x's nth binary digit; in other words, x_f has f as its digit sequence, since there is no n for which $d(n, x) \neq f(n)$. Then for any f in Δ^0_j for $j > 1$, x_f is uncomputable. ∎

12 Continued Fractions, Functionals, and the Analytical Hierarchy

The Arithmetical Hierarchy consists of sets described by a finite number of integer quantifiers. But we can also quantify over functions (or equivalently sets) of integers, which gives us the *Analytical Hierarchy* [1]. Now Σ^1_j and Π^1_j consist of the sets describable with j function quantifiers applied to an arithmetical predicate, the outermost of which is an $\exists$ or and $\forall$ respectively, and Δ^1_j is their intersection. The entire hierarchy is called Δ^1_ω.

For instance, suppose we have a *recursive partial ordering* $\succ_M$, i.e. a recursive function $\phi_{\succ_M} : \mathbf{N} \times \mathbf{N} \to \{0,1\}$ calculated by a Turing machine M such that $\phi_{\succ_M}(a,b) = 1$ if $a \succ_M b$ and 0 otherwise. Then the set of machines M such that $\succ_M$ is *well-founded*, i.e. there is no infinite sequence of descending integers $a_0 \succ_M a_1 \succ_M a_2 \succ_M \cdots$, is

$$\{M | \forall f : \exists n : f(n) \not\succ_M f(n+1)\}$$

where f ranges over all functions mapping $\mathbf{N}$ to $\mathbf{N}$. This set is in Π^1_1, since there is one function quantifier $\forall f$ (the $\exists n$ is just a number quantifier, and part of the arithmetic predicate to which $\forall f$ applies). In fact, this set is Π^1_1-*complete,* in that every Π^1_1 set can be recursively reduced to it [1].

Function quantifiers are much more powerful than number quantifiers; while number quantifiers search over $\mathbf{N}$, function quantifiers search the space of functions $\mathbf{N}^\mathbf{N}$ which is uncountably large. But this is the cardinality of the reals, and in fact we can express such a search $\mathbf{R}$-recursively by mapping reals to functions on $\mathbf{N}$:

Proposition 17. *The function $\phi(x,n)$ that gives the nth place in x's continued fraction expansion is* $\mathbf{R}$*-recursive and in M_2. Conversely, given any* $\mathbf{R}$*-recursive function f that maps* $\mathbf{N}$ *to* $\mathbf{N}$*, the number $x[f]$ such that $\phi(x,n) = f(n)$ is* $\mathbf{R}$*-recursive.*

Proof. Let

$$x = x_0 + \cfrac{1}{x_1 + \cfrac{1}{x_2 + \cfrac{1}{\ddots}}}$$

Then $x_n = \lfloor g^n(x) \rfloor$ where $g(x) = f_\times\big((x \bmod 1)^{-1}, 1 - \chi_\mathbf{Z}(x)\big)$. Since $x \bmod 1$ and $\chi_\mathbf{Z}(x)$ are in M_1, so is $g(x)$; since $\lfloor x \rfloor$ is M_1 as well, $\phi(x,n) = x_n$ is in M_2.

Conversely, if f is $\mathbf{R}$-recursive, we can write

$$x[f] = \mu_x \eta_n \delta(x_n - f(n))$$

which will return an x such that $x_n = f(n)$ for all n. ∎

Next we define $\mathbf{R}$-recursive functionals, since function quantifiers need predicates with functions as variables.

Definition. An $\mathbf{R}$-*recursive functional* $s[f,g,\ldots](\vec{x})$, with function variables $f, g, \ldots$ and number variables $\vec{x}$, is an $\mathbf{R}$-recursive expression generated from the initial functions $0, 1, f, g, \ldots$ by composition, integration, and μ.

This is in complete analogy with the notion of partial recursive functional [1]. For instance, the "application operator" $\mathrm{Ap}[f](x) = f(x)$ is $\mathbf{R}$-recursive, as are the operators $\mathrm{Mu}[f](\vec{x}) = \mu_y f(\vec{x}, y)$, $\mathrm{Eta}[f](\vec{x}) = \eta_y f(\vec{x}, y)$, $\mathrm{Iter}[f](t, \vec{x}) = f^t(\vec{x})$, and the continued fraction encoding $x[f]$ from Proposition 17.

Lemma 18. *If $s[f, g, h, \ldots](\vec{x})$ is an $\mathbf{R}$-recursive functional and f_0 is an $\mathbf{R}$-recursive function, then $t[g, h, \ldots](\vec{x}) = s[f_0, g, h, \ldots](\vec{x})$ is an $\mathbf{R}$-recursive functional. If $f_0, g_0, h_0, \ldots$ are $\mathbf{R}$-recursive functions, then $u(\vec{x}) = s[f_0, g_0, h_0, \ldots](\vec{x})$ is an $\mathbf{R}$-recursive function of $\vec{x}$.*

Proof. This is clear from the definition: if we replace one of s's function variables with a particular $\mathbf{R}$-recursive function f_0, we get a functional t that can be generated from 0,1, and the remaining function variables. If all of s's function variables are replaced, u can be generated from 0 and 1 and so is simply an $\mathbf{R}$-recursive function. ∎

Proposition 19. *The Analytical Hierarchy Δ^1_ω is $\mathbf{R}$-recursive. In particular, $\Sigma^1_j, \Pi^1_j \subset M_{3+4j}$.*

Proof. Let $\chi_P[f, g, h, \ldots](\vec{x})$ be an $\mathbf{R}$-recursive characteristic functional for a property P of functions and integers, where $f, g, h, \ldots$ are functions which send $\mathbf{N}$ to $\mathbf{N}$ and where P only depends on their values on $\mathbf{N}$. Then, as in Proposition 16, we define

$$\chi_{\exists f:P}[g, h, \ldots](\vec{x}) = \eta_y \left(1 - \chi_P[\overline{y}, g, h, \ldots](\vec{x})\right)$$

$$\chi_{\forall f:P}[g, h, \ldots](\vec{x}) = 1 - \eta_y \chi_P[\overline{y}, g, h, \ldots](\vec{x})$$

where $\overline{y}(x) = y_x$, the xth term in y's continued fraction expansion. By Proposition 17 and Lemma 18, these functionals are $\mathbf{R}$-recursive; when all the function variables have been spoken for by function quantifiers, the result will be an $\mathbf{R}$-recursive function on $\vec{x}$.

A classic theorem [1] states that the arithmetical predicate can always be reduced to just one quantifier, of opposite type to the innermost function quantifier; so by Proposition 16 we start with a Functional in M_3. Then each function quantifier increases M by four: two for the η on the outside, and two for the y_x on the inside. Thus we get a total of M_{3+4j}. ∎

For example, the Π^1_1-complete set defined above, of Turing machines M that calculate a well-founded ordering, has the characteristic function

$$\chi_{\mathrm{wfo}}(M) = 1 - \eta_y \eta_n \phi_{\succ_M}(y_n, y_{n+1})$$

where $\succ_M$ can be calculated by a universal Turing machine given M as input. The set $\{(M, a, b) | a \succ_M b)\}$ is partial recursive (not recursive, since some of the M never halt) and so in M_3 by Proposition 15; adding two for y_n and two for η_y, we find this set is in M_7.

Corollary. *The Arithmetical Hierarchy is contained in M_7.*

Proof. It is contained in Δ^1_1. ∎

This seems wrong at first, given our earlier proof that $\Delta^0_j \subset M_{2j}$ and the fact [1] that Δ^0_j is a proper subset of Δ^0_{j+1} for all j. But multiple number quantifiers (in fact, a countably infinite number of them) can be subsumed into function quantifiers, and by using the continued fraction expansion to encode multiple integer variables into a single real instead of giving each integer its own real, we get a more efficient use of real variables.

We can also consider the hierarchy Δ^2_ω generated by third-order quantifiers; but the space of functionals that these quantifiers search over has cardinality larger than $\mathbf{R}$, so we expect this to be beyond the reach of the $\mathbf{R}$-recursive class. In fact, we suggest the following:

Conjecture. *If an* $\mathbf{R}$*-recursive function* f *maps integers to Boolean values* $\{0,1\}$*, then* $f|_{\mathbf{Z}}$ *restricted to the integers is* χ_S *for some Analytical set* S*.*

On the other hand, the reader can verify that we can $\mathbf{R}$-recursively encode functions from $\mathbf{Z}$ to $\mathbf{R}$ as single reals (although not from $\mathbf{R}$ to $\mathbf{R}$) so sets midway between Δ^1_ω and Δ^2_ω may be representable.

13 Directions for Further Work

In a future paper, I hope to prove the following conjectures.

1. Just as η patches μ to make functions everywhere defined, there is an $\mathbf{R}$-recursive operator ζ which senses when a unique solution to $h = f + \int g$ exists. Thus every $\mathbf{R}$-recursive function can be patched to make a total function, since μ and $\int$ are the only ways partial functions can arise.

2. It would then follow that the *universal function* $u(n,x) = f_n(x)$, where n is an integer index labelling each $\mathbf{R}$-recursive function f_n, is not $\mathbf{R}$-recursive. The proof would go like this: let $\mathrm{Def}(n,x) = 1$ if $f_n(x)$ is defined and 0 otherwise (Def can be defined from u, η and ζ, so it's $\mathbf{R}$-recursive if they are). Then let

$$v(x) = \delta\big(u(x,x)\mathrm{Def}(x,x)\big)$$

This is always defined, since $f_\times(u(n,x), \mathrm{Def}(n,x))$ always is. But $v = f_{\tilde{v}}$ for some $\tilde{v}$, and letting $x = \tilde{v}$ we get

$$v(\tilde{v}) = \delta\big(u(\tilde{v},\tilde{v})\mathrm{Def}(\tilde{v},\tilde{v})\big) = \delta(v(\tilde{v}))$$

But this is a contradiction since $x \neq \delta(x)$ for all x. So v must not be $\mathbf{R}$-recursive; so then neither is u.

In classical recursion theory, u is computable but Def is not; if we make the same diagonalization $v(x) = 1 - u(x,x)$ we simply find that v is undefined. But this system's ability to patch partial functions into total ones makes the universal function contradictory.

3. This would be strong evidence that the μ-hierarchy doesn't collapse, i.e. M_j is properly contained in M_{j+1} for all j. If the universal function were $\mathbf{R}$-recursive, it would be in some M_j, and all higher $M_{k>j}$ would collapse to it.

4. Is $\mathcal{K}$, the set of $\mathbf{R}$-recursive reals, $\mathbf{R}$-recursive? It is tempting to say no, since a diagonalization over members of $\mathcal{K}$ between 0 and 1, say, could give us a non-$\mathbf{R}$-recursive real. But to do this we need an $\mathbf{R}$-recursive map from

N to $\mathcal{K}$; and this is a version of the universal function u, which we believe is not **R**-recursive! So it does not appear this diagonalization can be carried out **R**-recursively. I leave this as an open question.

5. Infinite limits can be expressed in terms of μ; if the reverse is true, we can define a *limit hierarchy* and relate the μ-hierarchy to it. This would be analogous to Schoenfield's theorem [1] that functions in Δ^0_{n+1} are infinite limits of functions in Δ^0_n, and would connect the μ-hierarchy with the levels of Rubel's Extended Analog Computer. It would also suggest the M_j as a kind of constructive version of the Baire classes [15, 16], since the fact that functions in M_0 and M_1 are continuous and countably discontinuous respectively is reminescent of Baire's theorem on functions of the first class.

14 Conclusion

We have defined a class of functions on the reals generated with simple rules of recursion, as a proposed model of idealized computation in continuous time. The zero-finding operator μ seems unphysical, so we have stratified our class into a hierarchy based on the number of uses of it. This μ-hierarchy ranges from continuous functions calculable by Shannon's circuit model of analog computation, up through the Arithmetical and Analytical hierarchies of increasingly uncomputable functions — showing that a continuous model of computation is, in principle, far more powerful than a discrete one.

Does the *Physical Church-Turing Thesis*, that the physical world is computable, still hold? In a perfect, classical world where the μ operator can be implemented, no. But in a world with noise, quantum effects, finite accuracy, and limited resources, even $\delta(x)$ isn't physically realizable: how can we tell precisely whether $x = 0$ or not? If x is a velocity, we need to wait an infinite time to see if it moves; if x is a probability, we need an infinite number of ensembles to see if it happens; if x is a position, we need light of infinite frequency to locate it; if x is $T - T_c$ where T_c is a critical temperature, we need an infinite number of particles for the thermodynamic limit to be meaningful. So in the world we live in, only the lowest level of the μ-hierarchy seems to be realizable, and the Physical Church-Turing thesis seems safe.

But these infinite limits are precisely the ones in which many physical quantities are defined. The critical exponent of a spin system, for instance, requires infinite time and a thermodynamic limit, as well as an infinite series of systems closer and closer to the critical temperature. The μ-hierarchy may be a good tool to classify the various quantities about the world we want to measure, and tell us how many infinite limits they are away from being physically computable.

Acknowledgements. I am grateful to Elizabeth Hunke, Nelson Minar, and Tino Gramß for careful readings of the manuscript, to Alan Horwitz for helpful comments and an introduction to Lee A. Rubel's work on analog computation, and to Spootie the Cat for companionship.

References

[1] P. Odifreddi, *Classical Recursion Theory.* Elsevier, 1989.

[2] Marvin Minsky. *Computation: Finite and Infinite Machines.* Prentice-Hall, 1967.

[3] C.H. Papadimitriou, *Computational Complexity.* Addison-Wesley, 1994.

[4] L. Blum, M. Shub, and S. Smale, "On a Theory of Computation and Complexity over the Real Numbers: NP-Completeness, Recursive Functions and Universal Machines." *Bull. Amer. Math. Soc.* **21** (1989) 1-46.

[5] K. Meer, "A Note on a $P \neq NP$ Result for a Restricted Class of Real Machines." *Journal of Complexity* **8** (1992) 451-453, and "Real Number Models under Various Sets of Operations." *Journal of Complexity* **9** (1993) 366-372.

[6] E. Grädel and K. Meer, "Descriptive Complexity Theory over the Real Numbers." NeuroCOLT Technical Report NC-TR-95-040.

[7] C. Shannon, "Mathematical Theory of the Differential Analyzer." *J. Math. Phys. MIT* **20** (1941) 337-354.

[8] M.B. Pour-El, "Abstract Computability and its Relation to the General-Purpose Analog Computer." *Trans. Amer. Math. Soc.* **199** (1974) 1-28.

[9] L.A. Rubel, "Some Mathematical Limitations of the General-Purpose Analog Computer." *Advances in Applied Mathematics* **9** (1988) 22-34.

[10] L. Lipshitz and L.A. Rubel, "A Differentially Algebraic Replacement Theorem, and Analog Computability." *Proc. Amer. Math. Soc.* **99** (1987) 367-372.

[11] L.A. Rubel, "The Extended Analog Computer." *Advances in Applied Mathematics* **14** (1993) 39-50.

[12] C. Moore, "Unpredictability and Undecidability in Dynamical Systems." *Phys. Rev. Lett.* **64**, 2354 (1990), and *Nonlinearity* **4** (1991) 199.

[13] C. Moore, "Smooth One-dimensional Maps of the Interval and the Real Line Capable of Universal Computation." Santa Fe Institute preprint 93-01-001, submitted to *Theor. Comp. Sci.*

[14] M. Cosnard, M. Garzon, and P. Koiran, "Computability Properties of Low-dimensional Dynamical Systems." *Theor. Comp. Sci.* **132** (1994) 113.

[15] K. Kuratowski, *Topology.* Academic Press, 1966.

[16] J.C. Oxtoby, *Measure and Category.* Springer-Verlag, 1980.