On the Structure of Solutions of Computable Real Functions

Juris Hartmanis[1]
Lane A. Hemachandra[2]

May, 1988

CUCS-344-88

[1]Department of Computer Science, Cornell University, Ithaca, NY 14853
[2]Department of Computer Science, Columbia University, New York, NY 10027

On the Structure of Solutions of Computable Real Functions

Juris Hartmanis[*]
Department of Computer Science
Cornell University
Ithaca, NY 14853

Lane A. Hemachandra[†]
Department of Computer Science
Columbia University
New York, NY 10027

May, 1988

Abstract

The relationship between the structure of a domain and the complexity of computing over that domain is a fundamental question of computer science. This paper studies how the structure of the real numbers constrains the behavior of computable real functions. In particular, we uncover a close correlation between the structure of the zero set of a computable real function, and the complexity of the zeros. We show that computable real functions with hard solutions perforce have many solutions. Furthermore, as the complexity of solutions increases, the number of solutions increases. We prove that computable real functions with nonrecursive, nonarithmetical, or random zeros have solution sets that are, respectively, infinite, uncountable, or of positive measure. In addition, we show that the *computational* complexity of the zero set of a computable real function is limited by its *topological* complexity.

These results suggest an emerging paradigm—the inability of machines to name complex strings can serve as the basis of powerful proof techniques in computational complexity theory.

[*]Research supported by NSF grant DCR-8520597.

[†]Research supported in part by a Hewlett-Packard Corporation equipment grant, a Fannie and John Hertz Foundation Fellowship, and NSF Research Grant DCR-8520597.

1 Overview and Background

1.1 Overview

In this paper, we seek to connect the behavior of a computable function with the properties of the points in its domain. The real numbers, a domain of fundamental importance, are our focus.

Informally, a computable real function is a function $f : \Re \rightarrow \Re$ such that there is a Turing machine which, given an oracle that provides an arbitrarily accurate version of its input, computes arbitrarily close approximations of the function value at that input (see part 1 of Definition 2.1). As is standard, we assume throughout this paper that all our computable real functions are over the compact domain $[0, 1]$, and are total on this domain.

A *zero* of a function f is a solution of the equation $f(x) = 0$. It is tempting to say that a computable real function cannot have a random zero—as the machine computing the function would itself name the zero. The function $f(x) = 0$ provides a quick counterexample to that temptation. This function has many random zeros, however they are so well hidden in a crowd of zeros that they are not easily named.

This paper is a study of how dense a crowd various types of zeros need in order to hide successfully. We show that:

1. a computable real function with a nonrecursive zero must have infinitely many zeros,
2. a computable real function with a zero that falls out of a numerical analog of the arithmetical hierarchy must have an uncountable number of zeros, and
3. a computable real function that has a Chaitin random zero must have a zero set of positive measure.

In fact, a finer look at result 2 shows that the number of times one need apply a concentration point operator to a zero set to reduce it to a finite set yields an upper bound on the complexity of the members of that zero set within the arithmetical number hierarchy. Thus, the computational complexity of the zero set of a computable real function is limited by the zero set's point-set topological complexity.

Finally, we give examples showing that some of these results are near-optimal. In result 3 above one can neither strengthen the conclusion to state that there is an open interval of zeros, nor use weaker types of randomness and still conclude that there is positive measure. In result 2 above, the conclusion cannot be strengthened to state that the zero set is of positive measure.

This abstract is organized as follows. Section 2 presents definitions and notations. Section 3 presents theorems relating the complexity and number of zeros of a computable real function. Section 4 explores the optimality of these results. Section 5 presents some open problems and areas for further research.

1.2 Background

Discrete structures play a central role in computer science. Yet science and the real world often deal in functions whose domain and range are the reals. Turing, in the first paragraph of his famous 1936 paper initiated the search for a computational theory of real numbers and functions:

> Although the subject of this paper is ostensibly the computable *numbers*, it is almost equally easy to define and investigate computable functions of... a real or computable variable... I hope shortly to give an account of the relations of the computable numbers, functions, and so forth to one another. This will include a development of the theory of functions of a real variable expressed in terms of computable numbers [Tur36, p. 230].

Turing never fulfilled his hope. Nevertheless, the field that his words launched, recursive analysis, has generated many interesting results over the years [Rob51,Ric54,TW80,KF82, Ko83,NY83,Fri84,Ko86,PT86]. These results study various models of real computation and real numbers [Ko83], the analogs of classical analysis for computable functions [KF82], the class of operators under which the computable real functions are closed [PR83], the connections between the structure of feasible complexity classes and integration and maximization of certain functions [Mul86,Fri84], and the connections between discrete and continuous computation [PT86].

This paper takes a different approach—exploring the correlation between the structure of the reals and the solutions of computable real functions—that sheds new light on the behavior of computable real functions.

2 Definitions

In this section, we review the definitions of computable real functions, recursive numbers, and randomness, and present an analog, for real numbers, of the arithmetical hierarchy.

2.1 Computable Real Functions

The definition of a computable real function captures the idea that close domain points must map to close values. The domain points are expressed in terms of an oracle, which can be queried to get a good approximation of a real value.

Definition 2.1 [Ko86]

1. A computable real function is a function $f : \Re \rightarrow \Re$ such that there is an oracle Turing machine M so for each $x \in \Re$ and each function ϕ for which $|\phi(j) - x| \leq$

2^{-j}, the function ψ computed by M with oracle ϕ (i.e., $\psi(n) = M^{\phi}(n)$) satisfies: $|\psi(n) - f(x)| \leq 2^{-n}$.

2. Let f be a continuous function on $[a, b]$. A function $m : \mathrm{N} \rightarrow \mathrm{N}$ is said to be a *modulus function* of f on $[a, b]$ if $(\forall n \in \mathrm{N})(\forall x, y \in [a, b])[|x - y| \leq 2^{-m(n)} \Rightarrow |f(x) - f(y)| \leq 2^{-n}]$.

The definition of computable real functions is rather robust, and is equivalent to other definitions [Grz55][Ko86, p. 11]. For the rest of this paper, we'll take computable real function to mean total computable real functions on the domain $[0, 1]$. Many properties of such functions follow from the definition. In particular, any computable real function (on a compact domain) is uniformly continuous and has a computable modulus of uniform continuity [Grz55, p. 55][KF82, p. 331].

2.2 An Arithmetical Number Hierarchy

The definition of recursive numbers has become standard [Ko83], and is robust. Many definitions have been shown to yield the same class [Rob51].

Definition 2.2 α is a *recursive*, or *computable*, *number* (written $\alpha \in \Sigma_0^{num}$) if there is a total Turing machine M so $|M(j) - \alpha| \leq 2^{-j}$.

Definition 2.3 α is a *recursively enumerable* number if there is a total Turing machine M so that $\lim_{j \rightarrow \infty} M(j) = \alpha$.

Ko [Ko83] presents three definitions of recursively enumerable real numbers, and proves that they are not all equivalent. It is easy to show that a recursively enumerable number under any of Ko's definitions is recursively enumerable under ours. The difference is that Ko requires the Turing machine that converges to α to be monotonic.

We generalize the concepts of a recursive number and recursively enumerable number, and define a hierarchy of numbers computable via machines in the arithmetical hierarchy.

Definition 2.4 α is a Σ_k-*number* (written $\alpha \in \Sigma_k^{num}$), $k \geq 1$, if there is a Turing machine M and a set L in Σ_{k-1}^0 (of the classical arithmetical hierarchy of Kleene [Rog67]) such that M^L is total and $\lim_{j \rightarrow \infty} M^L(j) = \alpha$. A number is an *arithmetical number* if for some k it is a Σ_k-number.

In Section 3, this hierarchy will link the *topological* complexity of a zero set to the *computational* complexity of its members. If the zero set of a computable real function is simple, in terms of the number of times one has to apply a concentration point operator to obtain a finite set, then its zeros fall in low levels of the arithmetical number hierarchy.

2.3 Randomness

The theory of randomness appears in two major formulations, which occurred respectively in the middle of the 1960s and 1970s. The second formulation—due to Chaitin and Levin [Cha74,Lev74,Cha82]—was a rather radical departure from the first formulation—due to Chaitin, Kolmogorov, and Solomonoff [Cha66,Kol65,Sol64]. The first formulation defined the complexity of a string as the size of the smallest program computing the string. In contrast, the second formulation defined the complexity of a string to be the size of the smallest *self-delimiting* program—a program such that the head of the universal machine does not run off the program's edge during the decoding—computing the string. The motivation for the change was to allow the joint information of two strings to satisfy the same equation as the entropy function of information theory [Cha82]. Essentially, with self-delimiting programs you can concatenate subroutines without having to spend a logarithmic number of bits to describe the location of the boundary. Though the first formulation still is often used, we adopt the elegant Chaitin-Levin formulation.

Definition 2.5 A number α is *Chaitin random* if $\exists c \forall n \; H(\alpha_n) \geq n - c$, where α_n denotes the first n bits of α. Here, $H(\cdot)$ is the self-delimiting complexity of Chaitin [Cha82].

Definition 2.6 A number α is *weak-random* if $\exists c \forall n \; H(\alpha_n) \geq \frac{n}{c}$.

3 Connecting the Complexity and Cardinality of Zeros

The theorems of this section show that computable real functions with complex zeros necessarily have many zeros.

It is easy to see that a function with a finite number of zeros has only recursive zeros—each zero is the *only* zero on some interval, which gives a procedure for finding it. This is a slight strengthening of an old result of Grzegorczky [Grz55] that the zero of a monotonically increasing computable real function is recursive. Theorem 3.1, however, applies to nonmonotonic computable real functions and to computable real functions that touch the x-axis without crossing it.

Theorem 3.1 Let f be a computable real function and let $Z_f = \{\alpha \mid \alpha \in [0,1] \text{ and } f(\alpha) = 0\}$. If Z_f contains a nonrecursive number ($Z_f \not\subseteq \Sigma_0^{num}$) then Z_f is an infinite set.

Proof Sketch Assume Z_f is finite. A *dyadic* value is a number of the form $k/2^l$, where $k, l \in \{0, 1, 2, \ldots\}$ and $0 \leq k \leq 2^l$, i.e., a fraction with a finite binary representation. Dyadic points are useful as they are easily and exactly described. Let z be an arbitrary element of Z_f. Since Z_f is finite, there are dyadic values a and b so $z \in (a, b)$, and z is the only zero of f on $[a, b]$. The modulus of uniform continuity of a computable function on a compact

domain is computable [Grz55,KF82]. So by sampling f at sufficiently fine dyadic points of (a, b), we effectively can trap the zero in an arbitrarily small interval, and thus z is recursive. ∎

Carrying this further, a computable real function with nonarithmetical zeros must have uncountably many zeros. We prove this by showing that if the zero set could be thinned via a topological "concentration point" operator, then the zeros would be forced into the arithmetical number hierarchy, as quantification has the power to describe the existence of concentration points of computable real functions.

A point α is a concentration point of set A if every open ball around α contains an infinite number of points in A [Rud74]. Given a closed set A, one can speak of the sequence of sets: $A \supseteq$ the set of concentration points of $A \supseteq$ the set of concentration points of the set of concentration points of $A \supseteq \cdots$. By observing that when this sequence eventually vanishes the points in A belong to the arithmetical number hierarchy, Theorem 3.2 proves that nonarithmetical numbers only appear in uncountable zero sets.

Theorem 3.2 Let f be a computable real function and let $Z_f = \{\alpha \mid \alpha \in [0,1] \text{ and } f(\alpha) = 0\}$. If some member of Z_f is a nonarithmetical number then Z_f is an uncountable set.

Proof Sketch If Z_f were countable, it might have concentration points (points so that each open ball around them has an infinite number of points of Z_f), and might have concentration points of concentration points and so on, but only finitely far. That is, for each computable f with a countable number of zeros, there exists an i so that $|Cp^{(i)}(Z_f)| < \infty$, where $Cp^{(j)}$ means the concentration point operator iterated j times and $Cp(A) = \{x \mid x \text{ is a concentration point of } A\}$. Lemma 3.3 shows that the zero's of f are all in the arithmetical number hierarchy. ∎

Lemma 3.3 Let f be a computable real function such that $|Cp^{(i)}(Z_f)| < \infty$. Then $Z_f \subseteq \Sigma^{num}_{2i+2}$.

This lemma is of independent interest. We interpret it to mean that, for computable real functions:

> *The computational complexity of the zero set is limited by the topological complexity of the zero set.*

The approach of the lemma's proof is simply to observe that $L_0 = \{< a, b > \mid Z_f \bigcap [a, b] = \emptyset$ and a, b are dyadic$\}$ is recursively enumerable (via the modulus function—the reason we don't claim that it is recursive is that the function might meet the x axis without crossing it), and that $L_1 = \{< a, b > \mid Cp(Z_f \bigcap [a, b]) \neq \emptyset$ and a, b are dyadic$\}$ is in Π^0_3 (since it is all (a, b) such that for all j there exists a sequence of $2j$ dyadic points $a < d_1 < \cdots < d_{2j} < b$ such that $< d_1, d_2 >, < d_3, d_4 >, \ldots, < d_{2j-1}, d_{2j} >$ are each in $\overline{L_0}$), etc. Using an L_1 oracle,

we can quickly get good approximations to the location of a concentration point that is the only concentration point in some open interval. This is the $i = 1$ case of Lemma 3.3—the other cases are similar. Please note that Lemma 3.3 is not tight. Indeed, the proof just alluded to actually gives a "Δ_{2i+2}^{num}" bound, and tighter bounds may be possible.

Truly random zeros force a computable real function to have uncountably many zeros. The proof shows that by using dyadic decomposition and the computable modulus of uniform continuity of a computable real function, we can find short names for every zero of a computable real function that has $\mu(Z_f) = 0$—thus such zeros are nonrandom. The crucial point is that we can not afford to spend the bits needed to describe the fineness of the dyadic decomposition. Thus, we structure our short names to implicitly incorporate this information.

Theorem 3.4 Let f be a computable real function and let $Z_f = \{\alpha \mid \alpha \in [0,1] \text{ and } f(\alpha) = 0\}$. If Z_f contains a Chaitin random number, then $\mu(Z_f) > 0$.

Proof Sketch Imagine breaking $[0,1]$ into 2^j length 2^{-j} segments and using the modulus function of f to declare each as "may contain a zero" or "clearly does not contain a zero." If we have $\mu(Z_f) = 0$ we claim there is a short name for each zero, and thus the zeros are not random. The short name is:

> "The i_0^{th} dyadic segment that is not known to be nonzero in the T_j (dyadic tree with 2^j length 2^{-j} intervals) that has the smallest j for which at most $\frac{2^j}{c}$ segments still threaten to contain a zero,"

where c is a highly compressible number. The trick involved is that this avoids explicitly including the value j.

For this approach to work, we need to know that the complement of Z_f can be "covered from inside," i.e., that $\overline{Z_f}$ contains sets of dyadic intervals of measure arbitrarily close to one, which is proven by Lemma 3.5. ∎

Lemma 3.5 Let T_j be the tree of 2^j length 2^{-j} intervals dividing $[0,1]$. If f is a computable real function and $\mu(Z_f) = 0$, then as $j \to \infty$ the length of the segments of T_j known (via the modulus function) to have no zero approaches 1.

The lemma is proven as follows. f is a computable real function, so f is continuous, so Z_f is closed, so $\overline{Z_f}$ is open. A standard result from analysis is that:
Fact [Rud74, Result 2.19(d)] Every nonempty open set in $\Re$ is a countable union of disjoint half-open dyadic segments.

Thus, $\overline{Z_f}$ (when it is non-empty) is a countable union of disjoint dyadic half-open intervals whose measure totals 1, and each of these intervals itself is estimated arbitrarily closely by dyadic intervals that the modulus function proves to contain no zero.

4 Bounding Examples

Theorems 3.2 and 3.4 show that if a computable real function has zeros with a certain degree of complexity, then the function has many zeros. It is natural to ask the following question. Can we strengthen the theorems by reducing the degree of complexity we assume to prove a given abundance of zeros, or by increasing the abundance of zeros that we can prove from a given complexity of the zero set?

This section proves that the theorems cannot be substantially strengthened in this way. We do so by providing counterexamples to strengthened versions of Theorems 3.2 and 3.4.

The following theorem shows that the use of Chaitin randomness in Theorem 3.4, is crucial. If we use weak-randomness, we loose our guarantee of measure. Theorem 4.1 also shows that the conclusion of Theorem 3.2 cannot be strengthened to show positive measure.

Theorem 4.1 There is a computable real function that has weak-random and nonarithmetical zeros, yet has a zero set of measure zero.

Proof Sketch Follow the Cantor set construction, but construct a function by making each subtracted segment into a "$\wedge$" (a bump of nonzeros) with the bumps at stage n of height 2^{-n}. Our function is the infinite limit of this process. It is a computable real function, and has all the Cantor points as zeros. Thus, by the nature of the Cantor set, our function has a zero set of measure zero, an uncountable number of zeros (and thus must have nonarithmetical zeros), and all its zeros can be compressed by a small multiplicative factor (as each point in the Cantor set, written in ternary, chooses only from two of three values for each ternary digit), even within Chaitin's second formulation of randomness. ∎

On the other hand, one might ask if Theorem 3.4 can be strengthened to conclude that random zeros imply not only a zero set of positive measure, but also an open interval of zeros. Theorem 4.2 below shows that we can not so strengthen Theorem 3.4. The proof of this is by Cantor's "extended" construction, which leaves measure but leaves no open intervals, and which, as above, can be converted into a computable real function.

Theorem 4.2 There is a computable real function f that has Chaitin random zeros, but whose zero set contains no open interval.

5 Conclusions, Related Results, and Open Problems

This paper proposed and applied the paradigm of studying the behavior of computable functions in light of the complexity structure of their domains. We studied the zero sets of computable real functions and proved that computable functions with zeros of high computational complexity have many zeros. Thus, though in much of mathematics all points are treated as equals, for computational purposes this is not the case. The complexity of real points profoundly constrains the behavior of computable real functions.

In fact, this paper reflects an emerging paradigm—the inability of machines to succinctly name complex strings provides a powerful proof technique in computational complexity theory [Nat85,CL86,Sip86,Hem86].

We have also proven independence results about the complexity of pairing computable real functions with modulus functions, and of recognizing computable real functions.

There are some open problems on the optimality of the theorems. Can Theorem 3.1 have its conclusion strengthened to "uncountable"? How tight can Lemma 3.3, which connects computational and topological complexity, be made? Would the theorems still hold with relaxed definitions of computable real functions that, for example, may not be uniformly continuous?

More generally, what other problems and areas can benefit from the approach of studying the influence of the complexity of domain points on computable functions over those points? Note, for example, that all the results of this paper carry over intact to the study of fixed points of computable real functions.

The study of computable functions from integers to integers, the standard model of computation, seems an excellent application area for these techniques. Clearly, a computable function cannot increase the randomness of an input by more than the information content of the machine. However, further and finer observations about the connection between domain complexity and machine behavior may shed light on the constraints implicit in computation by Turing machines.

Acknowledgements

For helpful conversations and references to the rich literature on computable real functions, we are very grateful to Terry Boult, Stephen Cook, Paris Kanellakis, Daniel Leivant, Yoram Moses, Christos Papadimitriou, and Joe Traub. Bruce Esrig kindly led us past the demons of point-set topology. All errors are solely the responsibility of the authors and those demons.

References

[Cha66] G. Chaitin. On the length of programs for computing finite binary sequences. *Journal of the ACM*, 13:547–569, 1966.

[Cha74] G. Chaitin. Information-theoretic limitations of formal systems. *Journal of the ACM*, 21:403–424, 1974.

[Cha82] G. Chaitin. Algorithmic information theory. In *Encyclopedia of Statistical Sciences*, pages 38–41. Wiley, 1982. Volume 1.

[CL86] M. Chrobak and M. Li. $k+1$ heads are better than k for PDA's. In *Proceedings 27th IEEE Symposium on Foundations of Computer Science*, pages 361–367, October 1986.

[Fri84] H. Friedman. The computational complexity of maximization and integration. *Advances in Mathematics*, 53:80–98, 1984.

[Grz55] A Grzegorczyk. Computable functionals. *Fundamenta Mathematicae*, 42:168–202, 1955.

[Hem86] L. Hemachandra. *Can P and NP Manufacture Randomness?* Technical Report TR86-795, Cornell Computer Science Department, Ithaca, NY, December 1986.

[KF82] K. Ko and H. Friedman. Computational complexity of real functions. *Theoretical Computer Science*, 20:323–352, 1982.

[Ko83] K. Ko. On the definitions of some complexity classes of real numbers. *Mathematical Systems Theory*, 16:95–109, 1983.

[Ko86] K. Ko. Applying techniques of discrete complexity theory to numerical computation. In R. Book, editor, *Studies in Complexity Theory*, pages 1–62, John Wiley and Sons, 1986.

[Kol65] A. Kolmogorov. Three approaches for defining the concept of information quantity. *Prob. Inform. Trans.*, 1:1–7, 1965.

[Lev74] L. Levin. Laws of information conservation and aspects of the foundation of probability theory. *Prob. Inf. Transmission*. 10:206–210, 1974.

[Mul86] N. Müller. Subpolynomial complexity classes of real functions and real numbers. In *Automata, Languages, and Programming (ICALP 1986)*, pages 284–293, Springer-Verlag *Lecture Notes in Computer Science*, 1986.

[Nat85] B. Natarajan. *The Homogeneous Capture of Random Strings.* Technical Report TR 85-672, Department of Computer Science, Cornell University, Ithaca, NY, April 1985.

[NY83] A. Nemirovsky and D. Yudin. *Problem Complexity and Method Efficiency in Optimization.* John Wiley, 1983.

[PR83] M. Pour-El and I. Richards. Noncomputability in analysis and physics: a complete determination of the class of noncomputable linear operators. *Advances in Mathematics*, 48:44–74, 1983.

[PT86] C. Papadimitriou and J. Tsitsiklis. Intractable problems in control theory. *SIAM J. Control and Optimization*, 24(4):639–654, 1986.

[Ric54] H. Rice. Recursive real numbers. *Proc. Amer. Math. Soc.*, 5:784–791, 1954.

[Rob51] R. Robinson. Review of R. Peter's book *Rekursive Funktionen. J. Symbolic Logic*, 16:280–282, 1951.

[Rog67] H. Rogers, Jr. *The Theory of Recursive Functions and Effective Computability.* McGraw-Hill, 1967.

[Rud74] W. Rudin. *Real and Complex Analysis.* McGraw-Hill, second edition edition, 1974.

[Sip86] M. Sipser. Expanders, randomness, or time versus space. In *Proceedings 1st Structure in Complexity Theory Conference*, pages 325–329, Springer Verlag *Lecture Notes in Computer Science #223*, June 1986.

[Sol64] R. Solomonoff. A formal theory of inductive inference. *Information and Control*, 7:224–254, 1964.

[Tur36] A. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, ser. 2*, 42:230–265, 1936. Correction, *ibid*, vol. 43, pp. 544–546, 1937.

[TW80] J. Traub and H. Woźniakowski. *A General Theory of Optimal Algorithms.* Academic Press, 1980.

A The Relationship Between Measure and Machine Size

If we fix a Chaitin random fixed point R, and ask for a machine (computing a real function f) M so R is a zero of f but Z_f is of very small measure, the machine M will have to be very large (Theorem A.1). For this theorem, we require machines with thier modulus function attached to them (though in general, it is not decidable if a pair is a machine modulus pair).

Theorem A.1 Fix R, Chaitin random. For all sufficiently small μ, if f is a computable function, $\mu = \mu(Z_f)$, computed by self-delimiting machine-modulus pair M whose zero set contains R, then

$$\mu(Z_f)(\log \frac{1}{\mu(Z_f)})(\log\log \frac{1}{\mu(Z_f)})^2 \geq \frac{1}{2^{|M|}}.$$

(Note: By using better bounds for $H(\cdot)$, we can get a sequence of stronger and stronger versions of this theorem. For example, the next version would replace $(\log\log \frac{1}{\mu(Z_f)})^2$ with $(\log\log \frac{1}{\mu(Z_f)})(\log\log\log \frac{1}{\mu(Z_f)})^2$.)

This holds for all sufficiently small μ: it is an almost everywhere flavor result. A stronger bound can be proven infinitely often.

Theorem A.2 Fix R, Chaitin random. For any monotone increasing recursive function $g(\cdot)$ (e.g. $\log\log\log n$) there is an infinite sequence of μ_i's that converge to zero and if f is a computable function with $\mu(Z_f) < \mu_i$ and self-delimiting machine-modulus pair M computes f and Z_f contains R, then

$$\mu_i\, g(\frac{1}{\mu}) \geq \frac{1}{2^{|M|}}.$$

On the other hand, there is no recursive relation between how fast the nonzero measure of machine's zero sets (necessarily containing Chaitin random points) goes to zero and machine size. This does not contradict the previous paragraph and Theorem A.1; in these there was a FIXED fixed point, R, shared by the increasingly small intervals. But in showing there is no recursive relation between measure and size, we'll have no single fixed point occurring in all our zero sets of decreasing measure (Theorem A.3).

Theorem A.3 There is no recursive relation between how the measure of Z_f goes to zero and the machine size needed to achieve this small measure.

This theorem is just a disguised version of the undecidability of the halting problem. The halting problem's undecidability implies that there is no recursive relationship between machine size and runtime. Let $M'(\epsilon)$ be a relatively small machine with a terribly long runtime. Then

$$f(x) = \max(0,\, x - \frac{1}{\text{Runtime}(M'(\epsilon))})$$

is computable by a small machine, yet has terribly small measure.