RAIRO. INFORMATIQUE THÉORIQUE ET APPLICATIONS

S. DUBE

Fractal geometry, Turing machines and divide-and-conquer recurrences

RAIRO. Informatique théorique et applications, tome 28, n° 3-4 (1994), p. 405-423

<http://www.numdam.org/item?id=ITA_1994__28_3-4_405_0>

Informatique théorique et Applications/Theoretical Informatics and Applications
(vol. 28, n° 3-4, 1994, p. 405 à 423)

FRACTAL GEOMETRY, TURING MACHINES AND DIVIDE-AND-CONQUER RECURRENCES

by S. DUBE ([1])

Abstract. – *In this paper, we survey two new results establishing relationships between the theory of computation and the fractal geometry. Iterated Function Systems (IFS) are used as the tools to define fractals. First we survey the results which show that simple questions about IFS and their attractors are undecidable. The proofs are simple and are obtained by reducing the Post Correspondence Problem and by interpreting strings as numbers and concatenation operation as composition of affine transformations. These results show that for every Turing machine there exists a fractal set which can be viewed, in a certain sense, as geometrically encoding the complement of the language accepted by the machine. One can build a fractal-based geometrical model of computation which is computationally universal. Secondly we survey the results which show how fractal geometry can be fruitfully used to solve divide-and-conquer recurrences. A recursive algorithm possesses temporal self-similarity and there is a natural connection with spatial self-similar objects (fractal images). This approach yields a new and general way of solving such divide-and-conquer recurrences.*

1. INTRODUCTION

The emerging science of complex (chaotic) systems has brought remarkable insights into the nature of universe and life. It breaks across disciplinary boundaries as complex systems abound in physical, biological, mathematical, ecological, economic, meteorological, and many other fields [10]. Geometrically one can display the working of such complex systems as fantastic fractal images. A fractal image can be often viewed as a chaotic set of a dynamical system. This link between fractal geometry and complex systems is truly remarkable.

In this paper, the intuitive fact that fractals are complex objects is made precise from a computational point of view by proving that there do not exist algorithms to answer simple questions about fractals.

([1]) Department of Mathematics, Statistics and Computing Science, University of New England, Armidale NSW 2351, Australia.

Informatique théorique et Applications/Theoretical Informatics and Applications
0988-3754/94/03-04/$ 4.00/

The notion of undecidability is important in the theory of computation. The Turing machine is a simple computing device but can have complicated behavior. Rice's Theorem states that any nontrivial property of the recursively enumerable languages which are the languages defined (generated) by Turing machines is undecidable. Similarly, fractals can be defined (generated) by rather simple iterative methods. But as we will show these iterative methods can simulate the working of the Turing machine and hence produce complicated sets. Here the Turing machine provides the role of a dynamical system and the fractal image associated with it encodes its "chaotic" set (the words which are not accepted by the Turing machine and may lead to an infinite behavior in which the machine never halts).

This paper is motivated by a conjecture by Penrose in his book [14] that the Mandelbrot set is undecidable under a reasonable model of computation which allows you to work with real numbers. He speculates that fractals might be graphical way of looking at non-recursive mathematics. But Penrose in his book is faced with the problem of dealing with real numbers and still talk about computability questions. What is a "computable" real number?

There are some inherent philisophical problems in dealing with these numbers which have infinite description and these problems come to surface under a theory of computation over reals when simple questions such as, if a given computable real number is 0, turn out to be undecidable. A natural and invariant theory of computation over reals does not exist. Under a recently developed model of computation over reals which seems to be more natural than the earlier models, fractal sets have been shown to be undecidable [3]. But what about answering these questions under the classical model of computation over integers (rationals)?

The advantage of working under the Turing machine model is that it is *the* model in which you can prove that some problem is computationally unsolvable. And this model is a natural, rigorous and invariant one, unlike any theory of computation over reals. Any such undecidable problem should remain undecidable over any satisfactory theory of computation over reals.

We show that even under the classical theory of computation over rational numbers, in which the Turing machine is the model of computation, one can prove some problems about fractals to be undecidable.

Fractal geometry was pioneered by Mandelbrot who showed that many natural objects and phenomena have fractal characteristics [13]. We use Iterated Function Systems (IFS) as our basic mechanisms to define fractals. IFS were introduced by Hutchinson [12] and have been further developed

and applied for image generation and compression by Barnsley [1]. An IFS is a collection of N contractive affine transformations. Of course we restrict the coefficients of these affine transformations to be rational numbers. An IFS defines a unique set which is the fixed point of a mapping obtained by applying these N affine transformations and then taking the union. Using an iterative method one can obtain this fixed point in the limit and therefore the fixed point is also called the attractor of the IFS. It is the fractal defined by the IFS. In 2-dimensional real space it would be an image. The iterative method is called the Deterministic Algorithm for IFS.

In this paper, a number of simple problems about IFS are shown to be undecidable. For example, given an IFS defined on the unit square $[0, 1]^2$, there is no algorithm to test if its fractal intersects the diagonal line segment between the 2-D points (0, 0) and (1, 1). This also proves the problem of testing the emptiness of intersection of fractals defined by two given IFSs is undecidable. Another undecidable problem is testing whether a given IFS is totally disconnected.

The proofs are quite straightforward. The basic idea is to interpret strings as numbers and to implement concatenation operation as composition of affine transformations. The proofs are obtained by reducing the Post Correspondence Problem (PCP) and its variants. These results allow one to build a simple "geometrical" model of computation based on IFS which is computationally universal.

The results in this paper show how the fractal geometry and the computability theory are inter-related. And since the fractal geometry has strong links with the theory of complex systems, one is pleased to see these three scientifically rich disciplines conveying the same basic message, that finitely describable and seemingly simple systems have complex and *provably* unpredicatable behavior and such systems occur everywhere.

The second aim of the paper is to surevy results which show how fractal geometry can be fruitfully applied to solve divide-and-conquer recurrences. The analysis of the time complexity of algorithms is of fundamental importance to computer scientists. A great number of useful algorithms use Divide-and-Conquer approach, in which the original problem is reduced to a number of smaller problems [4]. In this paper, we consider a new fractal geometry based approach to analyze such algorithms, in which the size of a smaller problem is related to that of the original problem by a multiplicative factor.

The problem of analysis of such recursive algorithms reduces to solving divide-and-conquer recurrence relations. A number of methods have been developed for solving such recurrence relations, and also for general recurrence relations.

In [4] the Master method to solve divide-and-conquer recurrences is discussed. The Master Method is based on the Master Theorem, which is adapted from [2]. We use fractal geometry to come up with a technique to solve the divide-and-conquer algorithms which generalizes the Master Theorem.

At a conceptual level, the notion of self-similarity is not limited solely to images but can be used to describe many natural phenomena like distribution of noise of a channel, Brownian motion of particles in air. In this paper, we show that a divide-and-conquer algorithm is also "self-similar" as it is made of its smaller "copies". Here self-similarity is temporal while in case of a natural object it is spacial.

This natural connection between fractal geometry and divide-and-conquer recurrences yields a new and general way of solving such recurrences. We use Mutually Recursive Function Systems (MRFS) for this purpose. MRFS are generalization of IFS. In this paper, we state a theorem which shows how divide-and-conquer recurrences can be modelled by MRFS, and the solution of the recurrences is closely related to the fractal dimension of the attractors of the MRFS.

For details of the results surveyed in this paper, *see* [7, 8].

2. PRELIMINARIES

2.1. Undecidability

We assume that the reader is familiar with the basic theory of computation [11].

We briefly describe the notations for ω-strings and ω-languages. An ω-language is a set of infinite strings (ω-strings). The symbol ω informally means infinite repetition. Thus the ω-string $001^\omega = 001111\ldots$ The ω-language $0^* 1^\omega$ is the set of all ω-strings which have finitely many 0's followed by infinitely many 1's (this language is also an ω-regular language). Similarly, one can work with ω-relations. For example, $\{(0, 1) + (1, 0)\}^\omega$ is an ω-relation which contains ordered pairs of ω-strings $(\{\sigma_1 \sigma_2 \sigma_3 \ldots, \gamma_1 \gamma_2 \gamma_2 \ldots\})$ where $\sigma_i, \gamma_i \in \{0, 1\}$ and $\sigma_i \neq \gamma_i$ for all i.

The sets of all finite words and ω-strings over an alphabet Σ are Σ^* and Σ^ω respectively and Σ^∞ denotes $\Sigma^* \cup \Sigma^\omega$.

A well-known undecidable problem is the Post Correspondence Problem (PCP):

Given lists A and B of k strings each from Σ^*, say

$$A = w_1, w_2, \ldots, w_k \qquad \text{and} \qquad B = x_1, x_2, \ldots, x_k,$$

does there exist any sequence of integers $i_1, i_2, \ldots, i_m$ with $m \geqq 1$ such that

$$w_{i_1} w_{i_2} \ldots w_{i_m} = x_{i_1} x_{i_2} \ldots x_{i_m}?$$

The sequence $i_1 i_2 \ldots i_m$ is called a *solution* of this instance of PCP.

To prove that PCP is undecidable one first proves that a modified version of the PCP (which we would refer to as M-PCP) can be reduced to PCP. M-PCP is stated as follows:

Given lists A and B of k strings each from Σ^*, say

$$A = w_1, w_2, \ldots, w_k \qquad \text{and} \qquad B = x_1, x_2, \ldots, x_k,$$

does there exist any sequence of integers $i_1, i_2, \ldots, i_m$ with $m \geqq 1$ such that

$$w_1 w_{i_1} w_{i_2} \ldots w_{i_m} = x_1 x_{i_1} x_{i_2} \ldots x_{i_m}$$

where one is required to start with the first string on each list and these strings are not used again *i. e.* $i_j \neq 1$ for all $1 \leqq j \leqq m$. Note that the condition $i_j \neq 1$ is not given in [11] but it can be implied from the same proof.

2.2. Iterated Function Systems

An IFS is given by N contractive transformations $w_1, w_2, \ldots, w_N$ on a complete metric space (X, d). Its notation is:

$$\{X; w_1, w_2, \ldots, w_N\}.$$

Typically, X is the k-dimensional Euclidean real space and the transformations are affine. We will restrict X to be $[0, 1]^k$.

A 2-dimensional *affine transformation* $w : R^2 \to R^2$ is defined by

$$w\begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}$$

where a_{ij}'s and b_i's are rational numbers (since we will be working under the classical theory of computation therefore we don't allow the coefficients to be reals). Similarly, a 1-dimensional affine transformation $w : R \to R$ is

defined by $w(x) = ax + b$. Likewise, we can define an affine transformation on R^n for all integers $n > 2$.

An IFS defines a unique set A which is the fixed point of the mapping $W : \mathcal{H}(X) \to \mathcal{H}(X)$ where $\mathcal{H}(X)$ is the set of nonempty compact subsets of the complete metric space (X, d) and W as defined as:

$$W(B) = w_1(B) \cup w_2(B) \cup \ldots \cup w_N(B)$$

for all $B \in \mathcal{H}(X)$. For the proof of existence and uniqueness of A *see* [1].

This invariant set A is called the *attractor* of the IFS and is the fractal set defined by it. It is a fractal since it is union of its own smaller copies:

$$A = w_1(A) \cup w_2(A) \cup \ldots \cup w_N(A).$$

This attractor A can be characterized as follows. Consider the alphabet

$$\Sigma = \{w_1, w_2, \ldots, w_N\}$$

by treating w_i as a "symbol". Consider a mapping $\phi : \Sigma^\omega \to X$ defined as:

$$\phi(\sigma) = \lim_{i \to \infty} \sigma_1(\sigma_2(\ldots(\sigma_i(x))\ldots))$$

where $\sigma = \sigma_1 \sigma_2 \ldots$ and $\sigma_1, \sigma_2, \ldots \in \Sigma$ and x is any point in X. The fact that the above limit is independent of x follows from the contractivity of the transformations [1]. If $\phi(\sigma) = a$ then σ is called an *address* of a. In [1] it is shown that ϕ maps Σ^ω onto A.

Consider the IFS $\{[0, 1]^2; w_1, w_2, \ldots, w_N\}$ where $w_1, w_2, \ldots, w_N$ are N contractive affine transformations mapping the unit square $[0, 1]^2$ into itself. The attractor A can be described as the output of the following iterative algorithm:

1. Initialize a set $S = \{U\}$ of parallelograms where U is the unit square $[0, 1]^2$.

2. Execute an infinite loop consisting of the following step:

• Apply the N affine transformations on all the parallelograms in S to get a new set which replaces the old one.

The limiting value of S is the fractal defined by the IFS. The above algorithm is called the Deterministic Algorithm for IFS. One could have initialized S to be $\{B\}$ where B is any nonempty compact subset of $[0, 1]^2$, *see* [1].

The attractor A can be equivalently defined as

$$A = \bigcap_{n=1}^{\infty} \text{The set } S \text{ at } n\text{-th iteration}$$

At the n-th iterative step we have all the parallelograms obtained by applying all the sequences of transformations of length n on the unit square *i. e.* $w_{i_1}(w_{i_2}(\ldots(w_{i_n}(U))\ldots))$ for all sequences $w_{i_1} w_{i_2} \ldots w_{i_n}$. Note that each parallelogram at an iterative step is a proper subset of some parallelogram at the previous step and a superset of some parallelogram at the next step. Every point in the attractor is the limit point of some sequence of such nested parallelogram [6].

Example 1: In Figure 1 we show the first 3 steps of the Deterministic Algorithm on the IFS $\{[0, 1]^2; w_1, w_2, w_3\}$ where

$$w_1(x, y) = (0.5\,x, 0.5\,y)$$
$$w_2(x, y) = (0.5\,x + 0.5, 0.5\,y)$$
$$w_3(x, y) = (0.5\,x, 0.5\,y + 0.5).$$

The attractor is also shown and is the well-known Sierpinski Triangle. For examples of IFS which generate images of ferns, trees, clouds, mountains, forests etc. *see* [1]. One can also generate grey or color images by associating probabilities with the affine transformations. □

3. UNDECIDABLE PROBLEMS ABOUT IFS

One can use PCP and M-PCP problems to show that a number of other variants are also undecidable.

THEOREM 1: *Given two lists* $\langle x_1, x_2, \ldots, x_N\rangle$ *and* $\langle y_1, y_2, \ldots, y_N\rangle$ *of strings in* Σ^*, *the following problems are undecidable:*

I-PCP: *It is undecidable to test if PCP has an infinite solution* i. e. *to test if there exists an infinite sequence* $i_1, i_2, \ldots$ *such that*

$$x_{i_1} x_{i_2} \ldots = y_{i_1} y_{i_2} \ldots$$

MI-PCP: *It is undecidable to test if PCP has an infinite solution 1,* $i_1, i_2, \ldots$ *and* $i_j \neq 1$ *for all* $j \geqq 1$.

TI-PCP: *It is undecidable to test if there exist two different infinite sequences* $i_1, i_2, \ldots$ *and* $j_1, j_2, \ldots$ *such that*

$$x_{i_1} x_{i_2} \ldots = x_{j_1} x_{j_1} \ldots$$
$$y_{i_1} y_{i_2} \ldots = y_{j_1} y_{j_1} \ldots$$

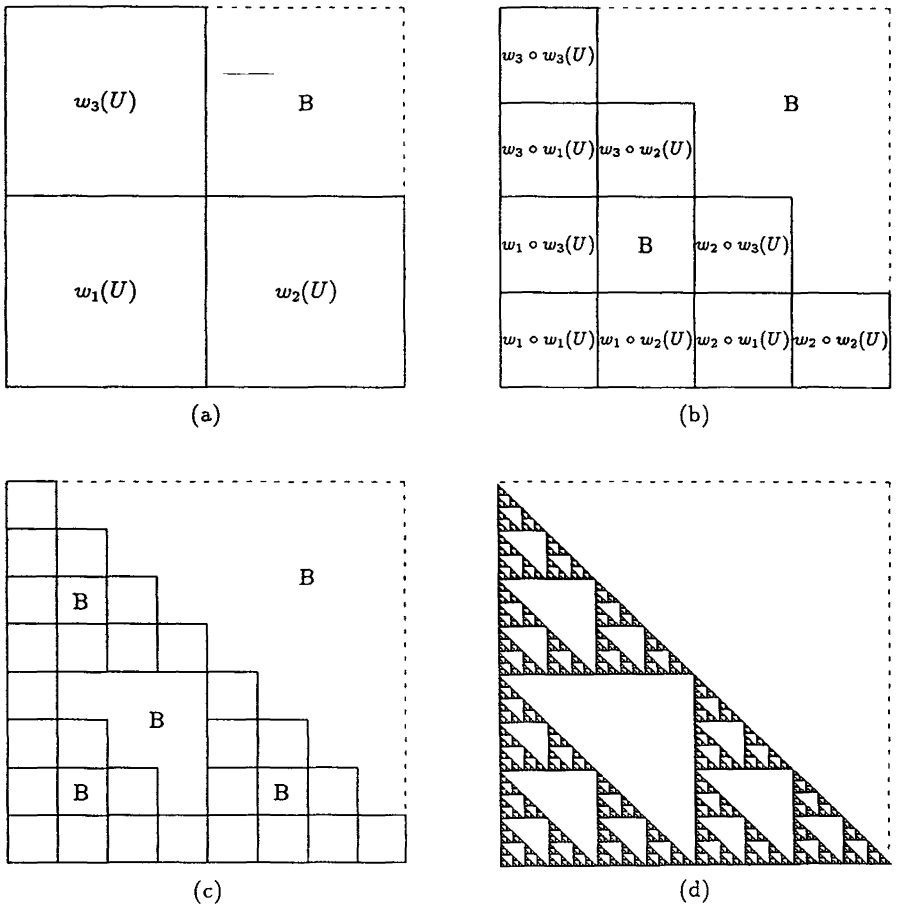

Figure 1. – Illustration of the IFS Deterministic Algorithm. (a), (b) and (c) show the parallelograms after the first, second and third iterations, respectively, and a B denotes a blank region which is not going to be a part of the attractor. Note how each parallelogram "splits" into three smaller parallelograms during each iteration. (d) shows the attractor which is the classic Sierpinski Triangle. By judiciously choosing transformations one can generate images of ferns, trees, mountains, clouds, flowers etc. By allowing transformations more general than affine transformations IFS can also generate Julia sets.

Proof: Both MI-PCP and I-PCP are well-known undecidable problems. For reduction of I-PCP to TI-PCP, *see* [7]. □

By reducing the Halting Problem for the Universal Turing machine one can prove the above problems to be undecidable for fixed lists.

THEOREM 2: *There exist two lists* $\langle x_2, x_3, \ldots, x_N \rangle$ *and* $\langle y_1, y_2, , y_N \rangle$ *such that for a given string* x_1, *PCP, M-PCP, I-PCP, MI-PCP, TI-PCP are undecidable.*

We would refer to the versions of MI-PCP and TI-PCP for the Universal Turing machine case as U-MI-PCP and U-TI-PCP.

The undecidability proofs for IFS are quite straightforward. The basic idea is to interpret strings as numbers in the unit interval [0, 1]. A string $w \in \Sigma = \{1, \ldots, d-1\}$ (note that 0 is not in Σ) is interpreted as a rational number by putting the radix point to the left and interpreting the string in base-d notation. Thus if $\Sigma = \{1, 2\}$ the string 2112 is interpreted as the ternary number 0.2112. Similarly we can interpret ω-strings as real numbers. We define $I(\varepsilon) = 0$ where ε is the null string. Formally we define an *interpretation function I:*

$$I: \ \Sigma^{\infty} \rightarrow [0, 1]$$

where Σ^{∞} denotes the set of finite and ω-strings over Σ.

The following lemma shows how concatenation of strings can be carried out as composition of affine transformations.

LEMMA 1: *Let* $\Sigma = \{1, 2, \ldots, d-1\}$. *Let* $I: \ \Sigma^{\infty} \rightarrow [0, 1]$ *be the interpretation function as defined above. Then I is injective and non-surjective.*

For every string $w \in \Sigma^*$ *define a 1-D affine transformation* $f^w: \ [0, 1] \rightarrow [0, 1]$ *as*

$$f^w(x) = \frac{1}{d^{|w|}} x + I(w).$$

Then the following hold:

1. *Let* $w \in \Sigma^*$ *and let* $x \in \Sigma^{\infty}$. *Then*

$$I(wx) = f^w(I(x)).$$

2. *Let* $u_1, u_2, \ldots, u_i \in \Sigma^*$ *where* $i \geqq 0$. *Then*

$$I(u_1 u_2 \ldots u_i) = f^{u_1}(f^{u_2}(\ldots(f^{u_i}(0))\ldots)).$$

3. *Let* $u_1, u_2, u_3, \ldots \in \Sigma^+$. *Then*

$$I(u_1 u_2 u_3 \ldots) = \lim_{i \rightarrow \infty} f^{u_1}(f^{u_2}(\ldots(f^{u_i}(x))\ldots))$$

where x is any real number.

4. *Let* $u_1, u_2, \ldots, u_N \in \Sigma^+$. *Let* $R = \{u_1 + u_2 + \ldots + u_N\}^\omega \subseteq \Sigma^\omega$ *for some* $N \geq 1$. *Let* A *be the attractor of the IFS* $\{[0, 1]; f^{u_1}, f^{u_2}, \ldots, f^{u_N}\}$. *Then* $I : \quad R \to A$ *is a bijection.*

Example 2: Let $\Sigma = \{1, 2, \ldots, 9\}$. Thus numbers are in decimal notation. Let $p = 124$, $q = 4$ and $r = 91845$ be three strings. Then,

$$f^p(x) = 10^{-3}x + 0.124$$
$$f^q(x) = 10^{-1}x + 0.4$$
$$f^r(x) = 10^{-5}x + 0.91845$$

One can check that Lemma 1 holds. For example, pick any string s in Σ^∞, say $s = 4276$. Then

$$I(ps) = I(1244276) = 0.1244276 = f^p(0.4276) = f^p(I(s)).$$

Look at IFS $\{[0, 1]; g^p, f^q, f^r\}$. Its attractor consists of precisely theose numbers which have representations of the form:

$$0 \cdot u_1 u_2 u_3 u_4 \ldots$$

where $u_i \in \{124, 4, 91845\}$ for all i. But this is exactly $I(R)$ where $R = \{124 + 4 + 91845\}^\omega$. □

We can generalize the above lemma to higher dimensions. In 2-D case, we have ordered pairs of strings which are interpreted as a point in the unit square *i. e.*

$$I : \quad \Sigma^\infty \times \Sigma^\infty \to [0, 1]^2$$

is defined as

$$I(u, v) = (I(u), I(v)) \qquad \text{for all} \quad u, v \in \Sigma^\infty.$$

For the sake of completeness we generalize Lemma 1 for 2-dimensions.

LEMMA 2: *Let* $\Sigma = \{1, 2, \ldots, d-1\}$. *Let* $I : \Sigma^\infty \times \Sigma^\infty \to [0, 1]^2$ *be the interpretation function as defined above. Then* I *is injective and non-surjective.*

Let $(x_1, y_1), (x_2, y_2), (x_3, y_3) \ldots \in \Sigma^+ \times \Sigma^+$. *Let* $g_i : [0, 1]^2 \to [0, 1]^2$ *be a 2-D affine transformation defined as*

$$g_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} d^{-|x_i|} & 0 \\ 0 & d^{-|y_i|} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{bmatrix} I(x_i) \\ I\, y_i \end{bmatrix}$$

Then the following hold:

1. *Let* $(r, s) \in \Sigma^\infty \times \Sigma^\infty$. *Then for all* $i \geqq 1$,

$$I((x_i, y_i) \cdot (r, s)) = g_i(I(r, s)).$$

2. *For all* $i \geqq 0$,

$$I\begin{pmatrix} x_1 x_2 \ldots x_i \\ y_1 y_2 \ldots y_i \end{pmatrix} = g_1\left(g_2\left(\ldots\left(g_i\begin{pmatrix}0\\0\end{pmatrix}\right)\ldots\right)\right).$$

3. *Let* z *be any point in the real plane. Then*

$$I\begin{pmatrix} x_1 x_2 x_3 \ldots \\ y_1 y_2 y_3 \ldots \end{pmatrix} = \lim_{i\to\infty} g_1(g_2(\ldots(g_i(z))\ldots))$$

4. *Let* $R = \{(x_1, y_1) + (x_2, y_2) + \ldots + (x_N, y_N)\}^\omega \subseteqq \Sigma^\omega \times \Sigma^\omega$ *where* $N \geqq 1$. *Let* A *be the attractor of the IFS* $\{[0, 1]^2; g_1, g_2, \ldots, g_N\}$. *Then* $I : R \to A$ *is a bijection.*

Now we can notice why certain problems on IFS would be undecidable. Lemmas 1 and 2 show that there is exact correspondence between the strings and the concatenation operation, and the application of the corresponding affine transformations. Thus one can reduce questions of strings obtained by concatenation to questions on numbers obtained by application of affine transformations. Since the attractor of an IFS is nothing but a set of numbers obtained by the application of affine transformations, one can reduce undecidable problems on strings to undecidable problems on IFS.

First we state that there is a semi-procedure to test if the attractor of an IFS does not intersect a given line segment.

LEMMA 3: *Let* $\{R^2; w_1, w_2, \ldots, w_N\}$ *be an IFS (affine transformations have rational coefficients). Let* L *be a line segment specified by its two end-points (specified by rational numbers). Then there is a procedure which will terminate iff the attractor* A *of the IFS does not intersect* L.

However there does not exist any semi-procedure which would terminate if the attractor intersects a line segment.

THEOREM 3: *Given an IFS, it is undecidable to test if its attractor intersects the diagonal line segment between points (0, 0) and (1, 1).*

Proof: Consider an instance of I-PCP. Let the two lists be

$$\langle x_1, x_2, \ldots, x_N\rangle \qquad \text{and} \qquad \langle y_1, y_2, \ldots, y_N\rangle.$$

Consider the IFS

$$\{[0, 1]^2; g_1, g_2, \ldots, g_N\}$$

Let $g_i : [0, 1]^2 \to [0, 1]^2$ be a 2-D affine transformation defined as

$$g_i \begin{pmatrix} x \\ y \end{pmatrix} = \begin{bmatrix} d^{-|x_i|} & 0 \\ 0 & d^{-|y_i|} \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{bmatrix} I(x_i) \\ I(y_i) \end{bmatrix}$$

Then our claim is that there exists an infinite solution of the I-PCP iff the attractor of the IFS intersects the diagonal line segment.

Let $R = \{(x_1, y_1) + (x_2, y_2) + \ldots + (x_N, y_N)\}^\omega$. Let A be the attractor of the IFS.

There exists an infinite solution of the I-PCP iff there exists $\sigma \in \Sigma^\omega$ such that $(\sigma, \sigma) \in R$. And A intersects the diagonal line segment iff there exists $a \in [0, 1]$ such that $(a, a) \in A$.

Since from Lemma 2 $I : R \to A$ is a bijection, therefore

$$(\exists \sigma)(\sigma, \sigma) \in R \quad \text{iff} \quad (\exists a)(a, a) \in A.$$

Thus there exists an infinite solution of I-PCP iff A intersects the diagonal line. This completes the proof. □

THEOREM 4: *Let M be a Turing machine. Then there exist an IFS with its attractor $A \subseteq [0, 1]^2$ and two fixed mappings $\psi, \theta : \Sigma^* \to \mathcal{Q} \cap [0, 1]$, such that for any word $w \in \Sigma^*$, the following holds*

$$w \in L(M) \quad \text{iff} \quad A \cap L = \phi$$

where L is a line segment in the unit square with slope $\psi(w)$ and intercept $\theta(w)$.

Proof: By considering how MI-PCP is reduced from the Halting Problem of Turing Machines, one can prove this theorem. *See* [7]. □

Theorem 3 and Theorem 4 imply that it is undecidable to test if the intersection of the attractors of two given IFSs is empty. This is because for every line segment there exists a simple IFS with 2 transformations which defines the line segment as its attractor.

Now we prove another interesting undecidability result about IFS. Let

$$\{[0, 1]^2; w_1, w_2, \ldots, w_N\}$$

be an IFS with attractor A. It is called *totally disconnected* iff each point in its attractor has a unique address. Equivalently, it is totally disconnected iff for all $i, j \in \{1, 2, \ldots, N\}$, $i \neq j$,

$$\omega_i(A) \cap w_j(A) = \phi.$$

Note that being totally disconnected is a property of the IFS and not its attractor. A set is called totally disconnected if its only nonempty connected subsets are singletons. A totally disconnected attractor of an IFS is also referred to as a Cantor set. Only when IFS has 2 affine transformations, then if it is totally disconnected it implies that its attractor is a Cantor set, and if it is not totally disconnected then it implies that its attractor is connected [1]. A Julia set is defined by an IFS with 2 transformations (which are not affine) and therefore a Julia set can be either a Cantor set or connected. However this is not true in general. An IFS with more than 2 transformations might have an attractor which is neither a Cantor set nor connected.

There exists a semi-procedure for testing if a given IFS is totally disconnected.

LEMMA 4: *There exists a procedure which given an IFS would terminate iff the IFS is totally disconnected.*

The problem of testing if a given IFS is totally disconnected is undecidable *i. e.* there does not exist a semi-procedure which will terminate if a given IFS is not totally disconnected.

THEOREM 5: *Given an IFS, it is undecidable to test if it is totally disconnected.*

The proof follows from reduction of TI-PCP, *see* [7]. One could have proved Theorems 4 and 5 by reducing U-MI-PCP and U-TI-PCP respectively.

Consider Theorem 4. Let the Turing Machine be the Universal Turing machine. One can call the IFS constructed to be "universal" IFS as it encodes working of the Universal Turing machine. Let its attractor denoted as A_U. Consider the set

$$H = \{[m, b] \in (\mathcal{Q} \times \mathcal{Q}) \cap [0, 1]^2 \mid A_U \text{ intersects the line } y = mx + b\}.$$

Clearly H is undecidable or otherwise the Halting Problem for the Universal Turing machine would be decidable.

```
Algorithm A (B [1 ... n])
Array B;
if n = 1 then ..., print (hello);
  else call A (B [1 ... n/2]),
       call A (B [n/4 ... 3 n/4]),
       call A (B [n/2 ... n]);
end A;
```

Figure 2. – Dynamic Structure of a Recursive Algorithm Captured by Sierpinski Triangle.

Similarly consider Theorem 5 in which we are reducing U-TI-PCP. Let the instance of U-TI-PCP be the string x. Let $\lambda = I(\omega)$ and denote the IFS constructed in the proof of the theorem as IFS (λ). Consider the set

$$M = \{\lambda \in \mathcal{Q} \cap [0, 1] \text{ IFS } (\lambda) \text{ is not totally disconnected}\}.$$

M is also undecidable. It can be viewed as a map of parameterized family of IFSs. Note that M is defined in the spirit of the Mandelbrot set:

$$\text{The Mandelbrot set} = \{c \in C \mid J(c) \text{ is connected}\}$$

where $J(c)$ is the Julia set for the dynamical system $f(z) = z^2 + c$. The above could have been written as:

$$\text{The Mandelbrot set} = \{c \in C \mid \text{IFS } (c) \text{ is not totally disconnected}\}$$

where

$$\text{IFS } (c) = \{C;\ \sqrt{z - c},\ -\sqrt{z - c}\}.$$

4. DIVIDE-AND-CONQUER RECURRENCES

Consider a recursive algorithm A as shown in Figure 2. The algorithm A calls itself 3 times and at each recursive call the input size is halved. The system of recurrence relation is:

$$T n = 3 T (n/2)$$
$$T (1) = O (1)$$

Note that we assume that all computation is done at the "trivial-case" $n = 1$. Therefore, $T(n) = \Theta(n^{\log_2 3})$.

Now A can be modeled by an IFS which generates the well-known Sierpinski Triangle, *see* Figure 1. Its fractal dimension is $\log_2 3$. This is no coincidence as can be intuitively deduced as follows:

Consider the intuitive definition of fractal dimension of a self-similar image $\mathcal{O}$ which implies that if $\mathcal{O}$ has fractal dimension D then

$$(1) \qquad (\text{number of self-similar copies}) \approx C\,(\text{magnification factor})^D$$

where C is some positive constant. Now consider the recursive algorithm A such that at each of its recursive calls the size of the input is reduced by a multiplicative factor. Each such calls creates a "copy" of A on smaller input. Since we assume that the only computation is done at the "trivial-case" when the size of the input is 1, the total time taken by A on an input of size n is the total number of recursive calls made with input size equal to 1. How many such trivial-case recursive calls are made? For this, we rewrite (1) as

$$(2) \qquad (\text{number of recursive calls}) \approx C\,(\text{magnification factor})^D$$

In our case

$$\text{magnification factor} = \frac{\text{size of the original input}}{\text{size of the trivial-case input}} = \frac{n}{1} = n.$$

Therefore, from (2) the time complexity of the algorithm A is

$$T(n) \approx Cn^D.$$

Now this interrelationship between divide-and-conquer recurrences and fractals can be generalized: to a system of recurrences and in which the multiplicative factor is any real number. Also one can easily handle the case in which computation is done at other recursion levels besides the trivial-case.

Consider a group $\mathcal{A}$ of n mutually recursive algorithms and let one algorithm be distinguished as the main algorithm (main "routine" in the terminology of programming languages) which is called first. An algorithm may call itself or any other algorithm. In such a recursive call the size of input is reduced by a multiplicative factor.

Now $\mathcal{A}$ can be modeled as an MRFS $\mathcal{M}$ defined on n variables, such that the execution of $\mathcal{A}$ corresponds "graphically" with the sequence of images generated while executing the Deterministic Algorithm on $\mathcal{M}$. Let the attractor of $\mathcal{M}$ (the fractal image defined by $\mathcal{M}$) be $\mathcal{O}$. Then the theorem states the mathematical relationship between the Hausdorff-Besikovitch dimension D and the Hausdorff D-dimensional measure of $\mathcal{O}$ and the time complexity of $\mathcal{A}$.

THEOREM 6: *Let $\mathcal{A}$ be a group of mutually recursive algorithms. Let $\mathcal{M}$ be a condensation MRFS modeling $\mathcal{A}$ and having $\mathcal{O}$ as its attractor. Let the Hausdorff-Besikovitch dimension of $\mathcal{O}$ be D and its Hausdorff D-dimensional measure be k. Let $T(n)$ be the time complexity of $\mathcal{A}$ on input of size n. Then,*

$$T(n) = \Theta(n^D \log^p n)$$

where p is the length of the longest path in the order structure DAG of $\mathcal{M}$. Furthermore, if $p = 0$ (equivalently, $k < \infty$) then k is the constant of proportionality by which two algorithms with same value of D can be compared.

This theorem is interesting because:

1. It provides a mathematically rigorous method to analyze mutually recursive algorithms.

2. It generalizes the known methods to solve recurrence relations.

3. It provides another link between discrete mathematics and continuous mathematics.

For details of the results mentioned in the section, *see* [8].

5. CONCLUSIONS

This paper showed a link between computability and fractal geometry motivated by some interesting speculative conjectures of Penrose. This paper suggests a number of challenging but seemingly difficult mathematical problems:

1. Are there algorithms for membership, inclusion and equivalence problems for IFS? Is there any algorithm to test if the attractor of a given IFS is totally disconnected *i. e.* is a Cantor set? Is there an algorithm to test if it is connected? Can we test if fractal dimension of the attractor of a given IFS is less than a given rational number? Can we test if the Hausdorff distance between the attractors of two given IFSs is less than a given rational number? The last problem would be trivially undecidable if equivalence problem for IFS turns out to be undecidable. Our conjecture is that all non-trivial properties of fractals should be undecidable.

2. For recreational mathematics, it would be interesting to actually generate actual pictures of the images of sets A_U, H and M discussed in this paper by judiciously choosing the transformations. Would such pictures express the complexity of the behavior of the Universal Turing Machine? It is also possible that they might turn out to be quite ordinary, since there is nothing

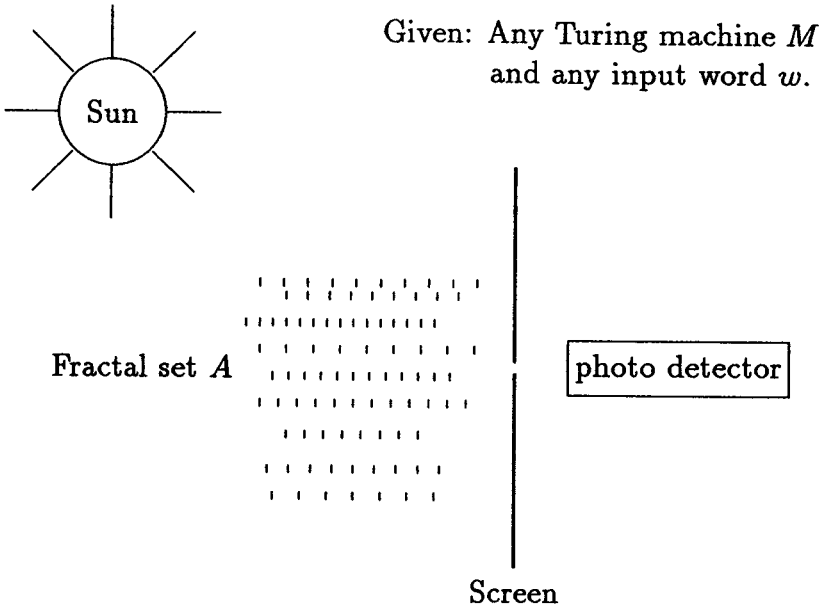

Figure 3. – A fractal-based "sun-computer"; for every Turing machine M there exists such a fractal set A which graphically "encodes" the behavior of M on all inputs. The slope of the sun rays and the position of the screen depend on the input word w.

special about the structure of the Universal Turing Machine. While generating these pictures, one may use the semi-procedures in Lemmas 3 and 4 to color a point depending on how fast the procedure terminates.

3. Finally can we characterize a fractal corresponding to a Turing machine in terms of the properties of the latter? Of particular interest would be to investigate the time-space complexity properties. How fractals encoding Turing machines solving NP-complete problems will be different from those encoding Turing machines solving P problems?

In conclusion, the three important disciplines – computability, fractal geometry and complex systems are closely related. This paper presents one more evidence in the support of this belief. Clearly there is an almost unlimited scope for research aimed at understanding complexity so commonly found in physical as well as mathematical world.

Finally, the author would like to take liberty in presenting the results of this paper in an informal and rather graphic way by proposing a model of computation in which IFS is the machine. Somehow the "sun-computer"

described below reminds one of the interesting suggestion of building a digital sun-dial by projecting shadows of a fractal set [9].

Let M be a Turing Machine. Consider a fractal set A, a source of light such as the sun, a screen and a photon detector as shown in Figure 3. The fractal set A has the following intriguing property. Take any word w. Compute two rational numbers $\psi(w)$ and $\theta(w)$ where ψ and θ are some fixed effective mappings. Now project parallel light rays onto the fractal set A where the slope of the light rays is $\psi(w)$. Make a hole in the screen at position $\theta(w)$. Near the hole place the photon detector. The interesting property of the set A is:

$$\text{A photon is detected iff } w \in L(M).$$

Furthermore suppose $\psi(w)$ depends only upon the length of the word w and approaches zero value as the length becomes arbitrarily large. Thus one can enumerate all the words in $L(M)$ (and in its complement) by letting the sun go down in appropriate steps. At each stage, we use the same screen but with hole at different locations (alternatively one may keep the hole fixed but move the screen up or down). When the sun will finally set, one would have enumerated all the words in lexicographic order!

Lemma 3 suggests a fractal-based model of computation. The classical models of computation proposed by Post, Turing, Church, Kleene, and the others, are all symbol manipulation systems. Under our model of computation one can work in a different abstract domain – of geometrical shapes and of geometrical operations on them. Absolutely no symbols are allowed.

We will build an IFS-like machine. Call such a machine M. Its program is nothing but N contractive geometrical operations which let M scale, translate and rotate an object (which will be a parallelogram). The machine just executes the IFS Deterministic Algorithm with these transformations and has an added capacity for receiving input and for "looking" at the arrangement of the objects:

1. The input to M is a line segment L.

2. M starts with a set $S = \{U\}$ of parallelograms where U is the unit square $[0, 1]^2$. It then enters an infinite loop of the following two steps:

Copying: M performs the N geometrical operations on all the parallelograms in S to get a new set which replaces the old one just as in the IFS Deterministic Algorithm.

Testing ("Looking"): M tests if any parallelogram in S intersects L. If none intersects, it accepts L and stops.

Of course, in order to show the equivalence of the above geometrical model of computation to the Turing Machine model, we need to interpret strings as geometrical entities and vice-versa and work with line segments and affine transformations which are specified by rational numbers. But the point is that *this IFS-based model of computation can be viewed in pure geometrical sense.* And each such machine defines a fractal in the limit of its infinite computation which is graphical encoding (in some sense) of what is not in the language.

REFERENCES

1. M. F. Barnsley, *Fractals Everywhere*, Academic Press, 1988.
2. J. L. Bentley, D. Haken and J. B. Saxe, A General Method for Solving Divide-and-Conquer Recurrences, *SIGACT News*, 1980, *12*, pp. 36-44.
3. 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, *Bulletin of American Mathematical Society*, 1989, *21*, pp. 1-46.
4. T. H. Cormen, C. E. Leiserson and R. L. Rivest, *Introduction to Algorithms*, MIT Press, 1990.
5. K. Culik II and S. Dube, Affine Automata and Related Techniques for Generation of Complex Images, *Theoretical Computer Science*, 1993, *116*, pp. 373-398.
6. K. Culik II and S. Dube, Encoding Images as Words and Languages, *International Journal of Algebra and Computation*, 1993, *3*, No. 2, pp. 211-236.
7. S. Dube, *Undecidable Problems in Fractal Geometry*, Technical Report 93-71, Dept. of Math. and Comp. Sci., University of New England at Armidale, Australia.
8. S. Dube, Using Fractal Geometry for Solving Divide-and-Conquer Recurrences, to appear in *Journal of Aust. Math. Soc., Applied Math.*, Preliminary version in Proc. of ISAAC'93, Hong Kong. *Lecture Notes in Computer Science*, Springer-Verlag, *762*, pp. 191-200.
9. K. J. Falconer, Digital Sun Dials, Paradoxical Sets and Vitushkin's Conjecture, *Math Intelligencer*, 1987, *9*, pp. 24-27.
10. J. Gleick, *Chaos-Making a New Science*, Penguin Books, 1988.
11. J. E. Hopcroft and J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.
12. J. Hutchinson, Fractals and Self-similarity, *Indiana University Journal of Mathematics*, 1981, *30*, pp. 713-747.
13. B. Mandelbrot, *The Fractal Geometry of Nature*, W. H. Freeman and Co., San Francisco, 1982.
14. R. Penrose, *The Emperor's New Mind,* Oxford University Press, Oxford, 1990.