BACHELOR

# Fractals

Mieras, W.

*Award date:*
2006

# Bachelorproject - Fractals

W. Mieras (0537945)

August 23, 2006

**Abstract**

In this report an introduction into the study of fractals is given. Some intuition is given of what fractals are and how to construct them. One can create fractals by looking at the attractor of a special type of contraction mapping, the so called Iterated Function System (IFS). In this report two algorithms are given that generate pictures of fractals using an IFS. Also a collage theorem is given that helps one to construct an IFS for a fractal when some target picture is given. A more general class of functions is also discussed, so called Iterated Random Functions, that will also converge to some image if they contract on the average. Lindenmayer systems give another way to look at fractals and construct them, some definitions, results and examples are discussed in this report. This report aims to show that fractals can be quite beautiful, also in a purely mathematical sense, and therefore an exiting topic to study.

# Contents

# Chapter 1

# Introduction

The main topic of study of this report are **fractals**. Different ways to look at fractals are explained and different ways to generate fractals are given. This report can be seen as an introduction into the field of study of fractals.

But what is a fractal exactly? The strict mathematical definition of a fractal is quite difficult and does not provide much intuition. To start with, a simple example of a fractal is the **Cantor set**, which is obtained by starting with the interval $[0, 1]$ and successively deleting the middle part of every subinterval. This gives:

$I_0 = [0, 1]$
$I_1 = [0, \frac{1}{3}] \cup [\frac{2}{3}, 1]$
$I_2 = [0, \frac{1}{9}] \cup [\frac{2}{9}, \frac{3}{9}] \cup [\frac{6}{9}, \frac{7}{9}] \cup [\frac{8}{9}, 1]$
$I_3 = ...$

And we define the Cantor set $\mathcal{C} = \bigcap_{n=0}^{\infty} I_n$, see also figure 1.1.

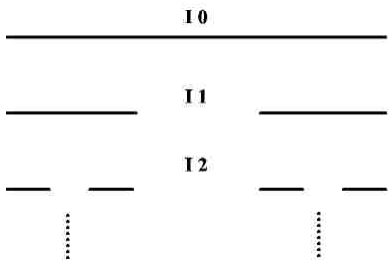


Figure 1.1: Construction of the Cantor set.

To provide some more insight in fractals, figures 1.2, 1.3 and 1.4 give a few more examples. Note the apparent *repetitive* structure in the pictures, on different scales the same patterns seem to emerge. This is a main property of fractals.

Fractal theory is an appealing field of study, not in the least bit because graphics of frac-

tals can be quite beautiful. Fractals turn up everywhere in nature and there are also many practical applications besides the artistic value of fractals, like modelling chaotic events, compressing pictures to very small sizes and modelling plant growth. Some examples of how one can use fractals to model plant growth will be given in this report. I hope you will enjoy reading this report and maybe get interested enough in fractals to read even more about them.

In chapter 2 we define a so called **Iterated Function System** (IFS) which can be used to study and to generate fractals. Two practical algorithms are given to generate pictures of fractals with an IFS: the deterministic algorithm and the random iteration algorithm. The collage theorem at the end of the chapter gives some insight in how to create an IFS given a picture of a fractal. Chapter 3 continues on the concept of an IFS and provides some more insights into fractals. Points of the fractal are given an address and dynamical systems are associated with fractals. The way the random iteration algorithm works is explained at the end of the chapter. In chapter 4 a generalization of the IFS is studied, in a more probabilistic setting. An IFS is a special kind of **iterated random function**, and the main theorem in this chapter says that if the random functions contract 'on the average' then the iterated images converge to some stationary probability distribution. Finally, in chapter 5 another way of looking at fractals and generating fractals is considered, the so called **Lindenmayer systems**. This way of looking at fractals is especially useful in a biological context, for plant modelling and studying the growth of a colony of bacteria for example.

Chapters 2 and 3 are loosely based on the book 'Fractals Everywhere' [1] by Michael Barnsley, chapter 4 contains mainly theory from the article 'Iterated Random Functions' [4] by Persi Diaconis and David Freedman, and chapter 5 makes frequent use of the book 'The Algorithmic Beauty Of Plants' [2] by Przemyslaw Prusinkiewics and Astrid Lindenmayer. Some references will be made to this material in the different chapters, and for some more insight in fractal theory this material is recommended. Most of the pictures in this report were generated using a practical implementation of the theory presented in this report. A description of this implementation is given in the relevant chapters.

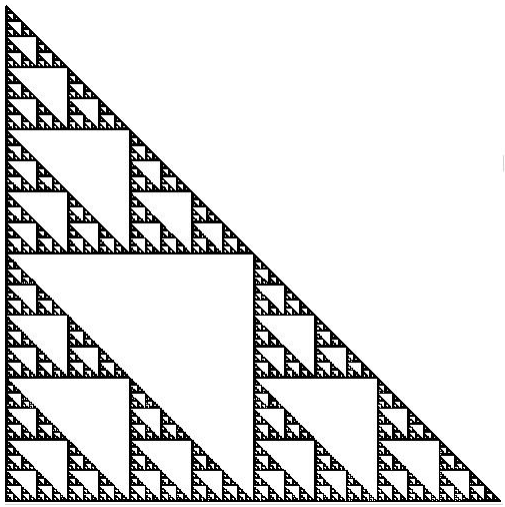

Figure 1.2: A Sierpinski fractal, generated using an IFS.

Figure 1.3: A fern fractal, generated using an IFS.

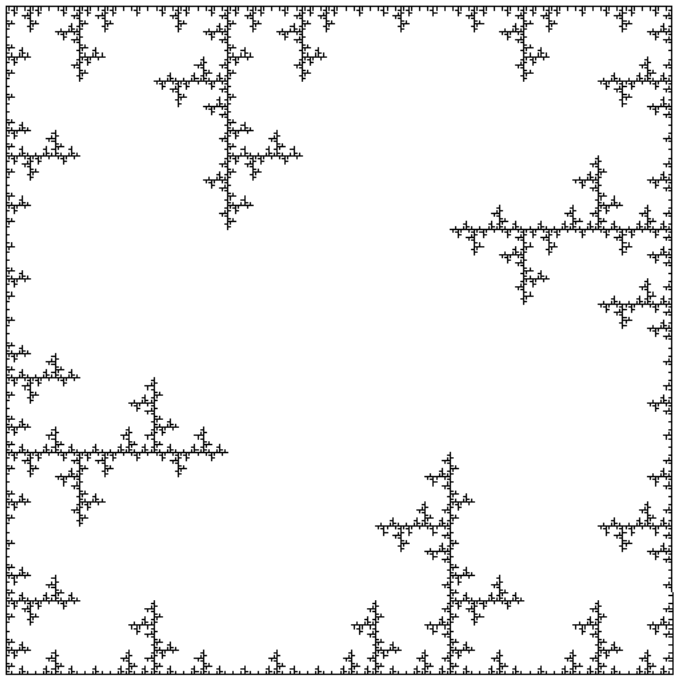

Figure 1.4: A Koch curve, generated using a L-System.

# Chapter 2

# Iterated Function Systems

## 2.1 Introduction

In this chapter a so called iterated function system (IFS) will be defined, which can be used to describe and to generate fractals.

An IFS is essentially a contraction mapping in the metric space $(\mathcal{H}(X), h)$. In section 2.2 metric spaces and several other mathematical structures are defined, some basic results are derived, and the space $(\mathcal{H}(X), h)$ is introduced, this is the space in which fractals 'live'. The main result is that this space is complete, which will be needed in later sections. In section 2.3 is defined what a contraction mapping is, and some properties of contraction mappings are derived. Finally, in section 2.4 the IFS is defined, and two algorithms are given to create pictures of fractals using an IFS. A collage theorem is given which allows one to construct an IFS based on a picture of a fractal. At the end of the section a practical implementation of the algorithms defined before is given. This implementation was used to generate most of the pictures in this chapter.

The main source of information for this chapter are chapters 2 and 3 from the book 'Fractals Everywhere' [1] by Michael Barnsley.

## 2.2 The space of fractals

In order to study fractals and fractal geometry, we need to define the space in which fractals live. First some general definitions and properties are stated which are used further on.

- A **space** $X$ is a set. The **points** of the space are the elements of the set. Examples of spaces are $\mathbb{R}^n$, $\mathbb{C}^n$, $\{1, 2, 3\}$.
- A **metric space** $(X, d)$ is a space $S$ with a real-valued function $d : X \times X \to \mathbb{R}$ which measures the distance between two points of $X$. $d$ needs to fulfil the following conditions:
  1. $\forall x, y \in X : d(x, y) = d(y, x)$
  2. $\forall x, y \in X, x \neq y : 0 < d(x, y) < \infty$
  3. $\forall x \in X : d(x, x) = 0$
  4. $\forall x, y, z \in X : d(x, y) \leq d(x, z) + d(z, y)$

An example of a metric space is $\mathbb{R}^2$ with the Euclidean metric $d(x,y) = \sqrt{x^2+y^2}$.

- A sequence $\{x_n\}$ of points in a metric space $(X,d)$ is a **Cauchy sequence** if for all $\epsilon > 0$ there exists an integer $N > 0$ such that $\forall n, m > N : d(x_n, x_m) < \epsilon$.
- A sequence $\{x_n\}$ of points in a metric space $(X,d)$ is called **convergent** to a point $x \in X$ if for all $\epsilon > 0$ there is an integer $N > 0$ such that $\forall n > N : d(x_n, x) < \epsilon$. This is written as $x = \lim_{n\to\infty} x_n$. Take for example $x_n = \frac{1}{n}$ in the metric space $(\mathbb{R}^2, \text{Euclidean})$, then $\lim_{n\to\infty} x_n = 0$.
- If a sequence $\{x_n\}$ of points in a metric space $(X,d)$ converges to a point $x \in X$ then it is a Cauchy sequence. But if $\{x_n\}$ is a Cauchy sequence it doesn't automatically follow that $\{x_n\}$ converges:

  A metric space $(X,d)$ is called **complete** if every Cauchy sequence $\{x_n\}$ converges to a point $x \in X$.
- Let $S \subset X$ be a subset of the metric space $(X,d)$. $x \in X$ is called a **limit point** of $S$ if there exists a sequence $x_n$ of $S \setminus \{x\}$ with $\lim_{n\to\infty} = x$. Note that the limit point does not need to lie in $S$: take for example $S = (0,1] \subset \mathbb{R}$ then the sequence $x_n = \frac{1}{n}$ converges to $0 \notin S$. Not also that not every element of $S$ is a limit point: take $S = \{1,2,3\}$ then $S$ does not contain any limit points.
- Let $S \subset X$ be a subset of the metric space $(X,d)$. The **closure** of $S$ is $\overline{S} = S \cup \{\text{Limit points of } S\}$. If $S = \overline{S}$ then $S$ is called **closed**.
- Let $S \subset X$ be a subset of the metric space $(X,d)$. $S$ is called **bounded** if there exists a point $a \in X$ and an $R > 0$ such that $d(a,x) < R$ for all $x \in S$.
- Let $S \subset X$ be a subset of the metric space $(X,d)$. $S$ is called **totally bounded** if for each $\epsilon > 0$ there exists a finite number of points $\{y_1, y_2, ..., y_n\} \subset S$ such that for every $x \in S$ there is a $y_i \in \{y_1, y_2, ..., y_n\}$ such that $d(x, y_i) < \epsilon$.
- Let $S \subset X$ be a subset of the metric space $(X,d)$. $S$ is called **compact** if every infinite sequence $(x_n)$ contains a subsequence with a limit in $S$. If $(X,d)$ is a complete metric space, then $S$ is compact if and only if $S$ is closed and totally bounded.

Now we can define the space $\mathcal{H}(X)$ in which fractals live:

Let $(X,d)$ be a complete metric space. $\mathcal{H}(X)$ **is the space whose points are the compact subsets of** $X$ **besides the empty set.**

Next, we need to define a useful metric on the space $\mathcal{H}(X)$ to be able to say when two fractals resemble each other, lie close to each other in some sense. First we define the distance from a point $x \in S$ to a set $B \in \mathcal{H}(X)$:

$$d(x,B) = \min\{d(x,y) : y \in B\}$$

So $d(x,B)$ is the distance from $x$ to the point $y \in B$ which is closest to $x$ under the metric $d$. Now we can define the distance from $A \in \mathcal{H}(X)$ to $B \in \mathcal{H}(X)$:

$$d(A,B) = \max\{d(x,B) : x \in A\}$$

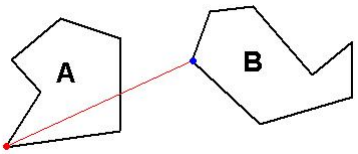


Figure 2.1: The distance from $A$ to $B$ is defined as the distance to the set $B$ from the point in $A$ that is the farthest away from the set $B$. Note that the distance from $A$ to $B$ is not the same as the distance from $B$ to $A$ in general.

The meaning of $d(A, B)$ is intuitively less obvious. It is the distance to the set $B$ of the point $x \in A$ which is the farthest from the set $B$, see figure 2.1. But one can imagine that if $A$ and $B$ lie closer to each other in $\mathcal{H}(X)$ that $d(A, B)$ becomes smaller. Note that $d(A, B)$ is not a metric, because $d(A, B) \neq d(B, A)$ in general. For example, let $A \subset B$ such that $B$ contains a point $x \notin A$. Then $d(A, B) = 0$, but $d(B, A) \neq 0$.

The concept of $d(A, B)$ can be extended to a metric: The **Hausdorff distance** between $A, B \in \mathcal{H}(X)$ is defined by $h(A, B) = \max\{d(A, B), d(B, A)\}$. For any $a \in A$, the distance from $a$ to the nearest point $b \in B$ is less than or equal to $h(A, B)$, and for any $b \in B$ the distance from $b$ to the nearest point $a \in A$ is also less than or equal to $h(A, B)$.

In order to show that the Hausdorff distance is a metric we need to show the following:

1. $h(A, B) = h(B, A)$: since $A$ and $B$ are symmetrical in the definition of the Hausdorff distance, this is trivial.

2. If $A \neq B$ then $0 < h(A, B) < \infty$: Since $h(A, B) = d(a, b)$ for some $a \in A$ and $b \in B$, $0 \leq h(A, B) < \infty$. Because $A \neq B$ we can assume without loss of generality that there exists an $a \in A$ with $a \notin B$, and $d(a, B) > 0$. So $h(A, B) > 0$.

3. $h(A, A) = 0$ is trivial, since $d(a, A) = 0$ for all $a \in A$.

4. $h(A, B) \leq h(A, C) + h(C, B)$: First show that $d(A, B) \leq d(A, C) + d(C, B)$:

$$\begin{aligned} d(a, B) &= \min\{d(a, b) : b \in B\} \\ &\leq \min\{d(a, c) + d(c, b) : b \in B\} \forall c \in C \\ &= d(a, c) + \min\{d(c, b) : b \in B\} \forall c \in C. \end{aligned}$$

   Thus

   $$d(a, b) \leq \min\{d(a, c) : c \in C\} + \max\{\min\{d(c, b) : b \in B\} : c \in C\} = d(a, C) + d(C, B).$$

   And $d(A, B) \leq d(A, C) + d(C, B)$ follows. Similarly it can be shown that $d(B, A) \leq d(B, C) + d(C, A)$. Then we have $h(A, B) = \max\{d(A, B), d(B, A)\} \leq \max\{d(B, C), d(C, B)\} + \max\{d(A, C), d(C, A)\} = h(B, C) + h(A, C) = h(A, C) + h(C, B)$.

So the Hausdorff distance is a metric and $(\mathcal{H}(X), h)$ is a metric space. Under certain circumstances this metric space is complete:

Let $(X, D)$ be a complete metric space. Then $(\mathcal{H}(X), h)$ is a **complete metric space**. Moreover, if $\{A_n \in (H(X))\}$ is a Cauchy sequence then $A = \lim_{n\to\infty} A_n \in \mathcal{H}(\mathcal{X})$ can be described as: $A = \{x \in X :$ there is a Cauchy sequence $\{x_n \in A_n\}$ which converges to $x\}$. The proof of this theorem is complex and will be omitted here, see [1] for the proof.

In the following section we will need a property of the Hausdorff distance:

- For all $A$, $B$, $C$ and $D$ in $\mathcal{H}(\mathcal{X})$ we have $h(A \cup B, C \cup D) \leq \max(h(A, C), h(B, D))$.

  **proof:** First note that $d(A \cup B, C) = \max(d(A, C), d(B, C))$ since

  $d(A \cup B, C) =$
  $\max\{d(x, C) : x \in A \cup B\} =$
  $\max(\max\{d(x, C) : x \in A\}, \max\{d(x, C) : x \in B\}) =$
  $\max(d(A, C), d(B, C))$.

  Similarly, $d(A, B \cup C) = \max(d(A, B), d(A, C)$. Now we have

  $h(A \cup B, C \cup D) =$
  $\max(d(A \cup B, C \cup D), d(C \cup D, A \cup B)) =$
  $\max(d(A, C \cup D), d(B, C \cup D), d(C \cup D, A), d(C \cup D, B)) =$
  $\max(h(A, C \cup D), h(B, C \cup D))$.

  Since $h(A, C \cup D) \leq h(A, C) + h(C, C \cup D) = h(A, C)$ and similarly $h(B, C \cup D) \leq h(B, D) + h(D, C \cup D) = h(B, D)$, it follows that $h(A \cup B, C \cup D) \leq \max(h(A, C), h(B, D))$.

## 2.3 Contraction Mappings

In this section we look at contraction mappings. First we define what a transformation is and give several definitions related to transformations which will be used later on. Then we define when a transformation is a contraction mapping, and give some properties of contraction mappings, with the Contraction Mapping Theorem as the main result. The IFS is essentially a special kind of contraction mapping on the space of fractals $(\mathcal{H}(X), h)$, and will be defined in the next section.

### 2.3.1 Transformations

Let $(X, d)$ be a metric space. A **transformation** on $X$ is a function $f : X \to X$ which assigns a point $f(x)$ to every point $x \in X$.

An example of a transformation is the **affine transformation**, $w : \mathbb{R}^2 \to \mathbb{R}^2$, defined by $w(x) = w\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}$, where $a, b, c, d, e, f \in \mathbb{R}$ and $x \in \mathbb{R}^2$. This transformation maps parallelograms into parallelograms, see figure 2.2.

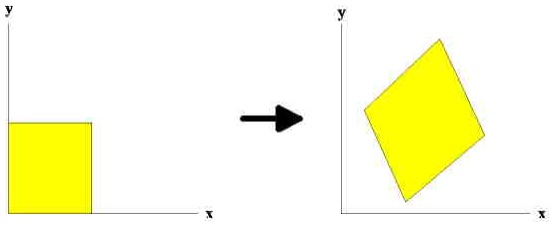


Figure 2.2: The affine transformation maps parallelograms into parallelograms.

There are several useful definitions for transformations which will be used later on. Let $f$ be a transformation on the metric space $(X, d)$:

- Let $S \subset X$, then $f(S) = \{f(x) : x \in S\}$.
- $f$ is **one-to-one** if for all $x, y \in X$ $f(x) = f(y)$ implies $x = y$.
- $f$ is **onto** if $f(X) = X$.
- $f$ is called **invertible** if it is one-to-one and onto, then it is possible to define the transformation $f^{-1} : X \to X$, the **inverse** of $f$, as $f^{-1}(y) = x$ where $x \in X$ is the unique point such that $y = f(x)$.
- The **forward iterates** of $f$ are transformations $f^{\circ n} : X \to X$ which are defined by $f^{\circ 0}(x) = x$,$f^{\circ 1}(x) = f(x)$, $f^{\circ n+1}(x) = f \circ f^{\circ n}(x)$.
- And the **backward iterates** of $f$ are transformations $f^{\circ(-n)} : X \to X$ which are defined by $f^{\circ(-1)}(x) = f^{-1}(x)$, $f^{\circ -n}(x) = (f^{\circ n})^{-1}(x)$.

### 2.3.2 Contraction mappings

A transformation $f : X \to X$ on a metric space $(X, d)$ is a **contraction mapping** if $d(f(x), f(y)) \leq s \cdot d(x, y)$ for a number $0 \leq s < 1$ and for all $x, y \in X$. $s$ is called a contractivity factor for $f$. Take for example $f(x) = 0.5x + 1$ in the metric space ($\mathbb{R}$,Euclidian). Then $d(f(x), f(y) = (0.5x + 1) - (0.5y + 1) = 0.5(x - y) = 0.5d(x, y)$. So $f$ is a contraction mapping with contractivity factor 0.5.

Let $f$ be a contraction maping on a metric space $(X, d)$. A point $x_f \in X$ such that $f(x_f) = x_f$ is a **fixed point** of $f$.

**Contraction Mapping Theorem**:
Let $(X, d)$ be a complete metric space and let $f : X \to X$ be a contraction mapping on $(X, d)$. Then $f$ has exactly one fixed point $x_f \in X$, and the sequence $\{f^{\circ n}(x) : n = 0, 1, 2, ...\}$ converges to $x_f$: $\lim_{n\to\infty} f^{\circ n}(x) = x_f$ for all $x \in X$.

**proof**:

Take an $x \in X$, let $f$ be a contraction mapping and let $0 \leq s < 1$ be a contractivity factor for $f$. It is easy to see that $d(f^{\circ n}(x), f^{\circ m}(x)) \leq s^{\min(n,m)} d(x, f^{\circ|n-m|}(x))$.
Furthermore, we have $d(x, f^{\circ k}(x)) \leq d(x, f(x)) + d(f(x), f^{\circ 2}(x)) + ... + d(f^{\circ(k-1)}(x), f^{\circ k}(x))$
$\leq (1 + s + s^2 + ... + s^{k-1}) d(x, f(x)) \leq \frac{1}{1-s} d(x, f(x))$, using the triangle inequality.
Substituting the second result in the first, we get $d(f^{\circ n}(x), f^{\circ m}(x)) \leq s^{\min(n,m)} \frac{1}{1-s} d(x, f(x))$.
Now it follows that $\{f^{\circ n}\}$ is a Chauchy sequence, and since $X$ is complete this sequence converges to a limit $x_f \in X$: $\lim_{n\to\infty} f^{\circ n}(x) = x_f$.
Because $f(x_f) = f(\lim_{n\to\infty} f^{\circ n}(x)) = \lim_{n\to\infty} f^{\circ(n+1)}(x) = x_f$ it follows that $x_f$ is a fixed point of $f$. Finally, we need to prove that $x_f$ is unique. Suppose there are two fixed points $x_f, y_f \in X$. Then $d(x_f, y_f) = d(f(x_f), f(y_f)) \leq s d(x_f, y_f)$. This implies $d(x_f, y_f) = 0$ and thus $x_f = y_f$.

Next, some properties of contraction mappings are derived. These properties will be used in the next section, where a special contraction mapping, the IFS, will be defined.

- Let $w : X \to X$ be a contraction mapping on the metric space $(X, d)$. Then $w$ is continuous.

  **proof**:
  Let $\epsilon > 0$ and let $s$ be the contractivity factor for $w$. Let $\delta = \frac{\epsilon}{s}$ and take $d(x, y) < \delta$. Then $d(w(x), w(y)) \leq s d(x, y) < \epsilon$, so $w$ is continuous.

- Let $w$ be a continuous mapping on $(X, d)$. Then $w$ maps $\mathcal{H}(X)$ into itself.

  **proof**:
  Let $S$ be a nonempty compact subset of $X$. It is easy to see that $w(S) = \{w(x) : x \in S\}$ is nonempty. We need to show that $w(S)$ is compact. Let $\{y_n = w(x_n)\}$ be an infinite sequence in $w(S)$. Then $\{x_n\}$ is an infinite sequence in $S$, and because $S$ is compact this sequence contains a convergent subsequence $\{x_{N_n}\}$ which converges to a point $\hat{x} \in S$. But since $w$ is continous it follows that $\{y_{N_n} = f(x_{N_n})\}$ converges to $\hat{y} = w(\hat{x}) \in w(S)$.

- Let $w : X \to X$ be a contraction mapping on the metric space $(X, d)$ with contractivity foctor $s$. Then $w : \mathcal{H}(\mathcal{X}) \to \mathcal{H}(\mathcal{X})$ defined by $w(B) = \{w(x) : x \in B\}$ for all $B \in \mathcal{H}(\mathcal{X})$ is a contraction mapping on $(\mathcal{H}(\mathcal{X}), h(d))$ with contractivity factor $s$.

  **proof**:
  From the fist lemma above we know that $w$ is continous. So by the second lemma $w$ is a map from $\mathcal{H}(\mathcal{X})$ to $\mathcal{H}(\mathcal{X})$. Let $B, C \in \mathcal{H}(\mathcal{X})$. Then $d(w(B), w(C)) = \max\{\min\{d(w(x), w(y)) : y \in C\} : x \in B\} \leq \max\{\min\{s \cdot d(x, y) : y \in C\} : x \in B\} = s \cdot d(B, C)$. Equivalently, $d(w(C), w(B)) \leq s \cdot d(C, B)$. So $h(w(B), w(C)) = \max(d(w(B), w(C)), d(w(C), w(B))) \leq s \cdot \max(d(B, C), d(C, B)) \leq s \cdot h(B, C)$.

- Let $(X, d)$ be a metric space and let $\{w_n : n = 1, 2, ..., N\}$ be contraction mappings on $(\mathcal{H}(\mathcal{X}), h)$. $s_n$ is the contractivity factor for $w_n$. Define $W : \mathcal{H}(\mathcal{X}) \to \mathcal{H}(\mathcal{X})$ by $W(B) = w_1(B) \cup w_2(B) \cup ... \cup w_N(B) = \bigcup_{n=1}^{N} w_n(B)$ for all $B \in \mathcal{H}(\mathcal{X})$. Then $W$ is a contraction mapping with contractivity factor $s = \max\{s_n : n = 1, 2, ..., N\}$.

  **proof**:

Let $N = 2$. Take $B, C \in \mathcal{H}(\mathcal{X})$. Then $h(W(B), W(C)) = h(w_1(B) \cup w_2(B), w_1(C) \cup w_2(C)) \leq \max(h(w_1(B), w_1(C)), h(w_2(B), w_2(C))) \leq \max(s_1 h(B, C), s_2 h(B, C)) \leq sh(B, C)$, where the first inequality follows from the last property of $h$ from the previous section. A general proof is obtained by induction on $N$.

## 2.4 The IFS

In this section a so called Iterated Function System (IFS) will be defined, which is essentially a contraction mapping in $(\mathcal{H}(X), h)$ and can be used to create pictures of fractals for example.

### 2.4.1 The IFS

An **iterated function system (IFS)** consists of a complete metric space $(X, d)$ and a finite set of contraction mappings $w_n : X \to X$ with contractivity factors $s_n$ for $n = 1, 2, ..., N$. The IFS is denoted by $\{X : w_n, n = 1, 2, ..., N\}$ and has contractivity factor $s = \max\{s_n : n = 1, 2, ..., N\}$.

From previous sections the following results about a IFS can be derived:

Let $\{X : w_n, n = 1, 2, ..., N\}$ be a IFS with contractivity factor $s$. Then the transformation $W : \mathcal{H}(\mathcal{X}) \to \mathcal{H}(\mathcal{X})$ defined by $W(B) = \bigcup_{n=1}^{N} w_n(B)$ for all $B \in \mathcal{H}(\mathcal{X})$ is a *contraction mapping* on the complete metric space $(\mathcal{H}(\mathcal{X}), h)$ with contractivity factor $s$, so $h(W(B), W(C)) \leq sh(B, C)$ for all $B, C \in \mathcal{H}(\mathcal{X})$. It has a unique *fixed point* $A \in \mathcal{H}(\mathcal{X})$ with $A = W(A) = \bigcup_{n=1}^{N} w_n(A)$, and is given by $A = \lim_{n\to\infty} W^{\circ n}(B)$ for any $B \in \mathcal{H}(\mathcal{X})$. This point is called the **attractor** of the IFS. This attractor is a *fractal*.

Before some examples of IFS's are given, we note that there is an extension possible to the definition of the IFS, namely the IFS with **condensation**:

- Let $(X, d)$ be a metric space and let $C \in \mathcal{H}(\mathcal{X})$. Define $w_0 : \mathcal{H}(\mathcal{X}) \to \mathcal{H}(\mathcal{X})$ by $w_0(B) = C$ for all $B \in \mathcal{H}(\mathcal{X})$. $w_0$ is called a **condensation transformation** and $C$ is the associated **condensation set**.
- Let $\{X, w_1, w_2, ..., w_N\}$ be an IFS with contractivity factor $s$ and let $w_0$ be a condensation transformation. Then $\{X, w_0, w_1, ..., w_N\}$ is called an **IFS with condensation** with contractivity factor $s$.

It is not difficult to check that the properties mentioned before for an IFS also hold for an IFS with condensation: it is a contraction mapping and has a unique fixed point $A \in \mathcal{H}(\mathcal{X})$. The IFS with condensation is another important way of making contraction mappings on $\mathcal{H}(\mathcal{A})$ and will be used in some theorems in the next chapter.

### 2.4.2 A graphical example

In this section pictures of the attractor of the IFS will be created using two different algorithms: the **deterministic algorithm** and the **random iteration algorithm**.

The deterministic algorithm is straightforward: we calculate a sequence of sets $\{A_n =$

$W^{\circ n}(A)\}$ where $A_{n+1} = \bigcup_{j=1}^{N} w_j(A_n)$, starting from a randomly chosen initial set $A_0$. Since $\lim_{n\to\infty} A_n = A$ where $A$ is the attractor of the IFS, for large $n$ $A_n$ will approximate the attractor $A$.

As a first example, consider the cantor set discussed in the introduction. We can use an IFS to get better and better approximations to this set. Consider the IFS $\{\mathbb{R} : w_1, w_2\}$ with:

$$w_1(x) = \frac{1}{3}x$$

$$w_2(x) = \frac{1}{3}x + \frac{2}{3}$$

And let $A_0$ be the interval $[0, 1]$. The first few iterations of the deterministic algorithm are displayed in figure 2.3.

For a more advanced and two-dimensional graph, consider the IFS $\{\mathbb{R}^2 : w_1, w_2, w_3\}$ with:

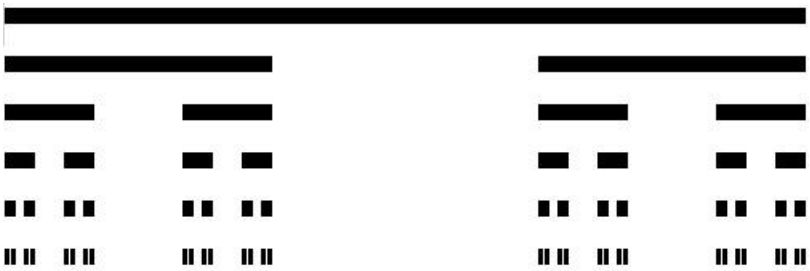

Figure 2.3: The first few iterations of the deterministic algorithm using an IFS for the cantor set.

$$w_1\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$w_2\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 50 \end{pmatrix}$$

$$w_2\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 50 \\ 50 \end{pmatrix}$$

The results of calculating $A_n$ for various values of $n$ and starting sets $A_0$ are displayed in figures 2.4 and 2.5. $A$ is a **Sierpinski triangle**, a famous fractal.

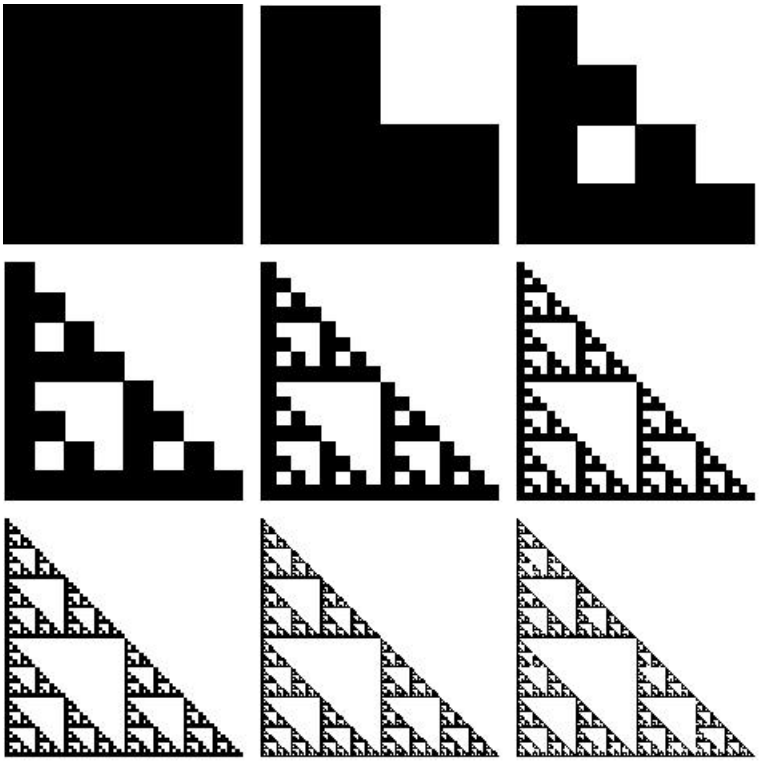

Figure 2.4: The first 9 sets $A_n$ calculated with the deterministic algorithm for the IFS defined above. $A_0$ is a filled square.

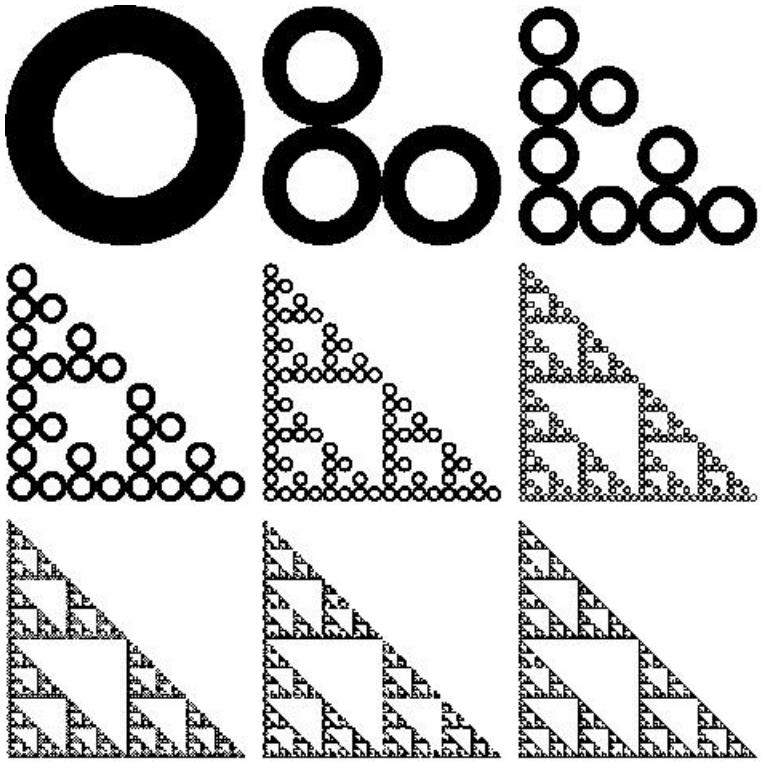

Figure 2.5: The first 9 sets $A_n$ calculated with the deterministic algorithm for the IFS defined above. $A_0$ is an open disc. Note that the figures converge to the same attractor as in figure 2.4 following the theorem that there is a unique attractor for the IFS.

A second algorithm to calculate pictures of the attractor of the IFS $\{X : w_n, n = 1, 2, ..., N\}$ is the random iteration algorithm. The algorithm works as follows: with every $w_i$ a probability $p_i$ is associated such that $p_i \geq 0$ and $\sum_{i=1}^{N} p_i = 1$. Take $x_0 \in X$ and $x_n \in \{w_1(x_{n-1}), w_2(x_{n-1}), ..., w_N(x_{n-1})\}$ where the probability that $x_n = w_i(x_{n-1})$ is $p_i$. This way a sequence $\{x_n : n = 0, 1, ...\}$ is constructed. Under certain conditions this sequence op points will represent the attractor of the IFS. The numbers $p_i$ do not influence the attractor, only how fast the sequence of points converges to the attractor. In figure 2.6 pictures of approximations of the Sierpinski triangle calculated with the random iteration algorithm are shown for increasing numbers of iterations.

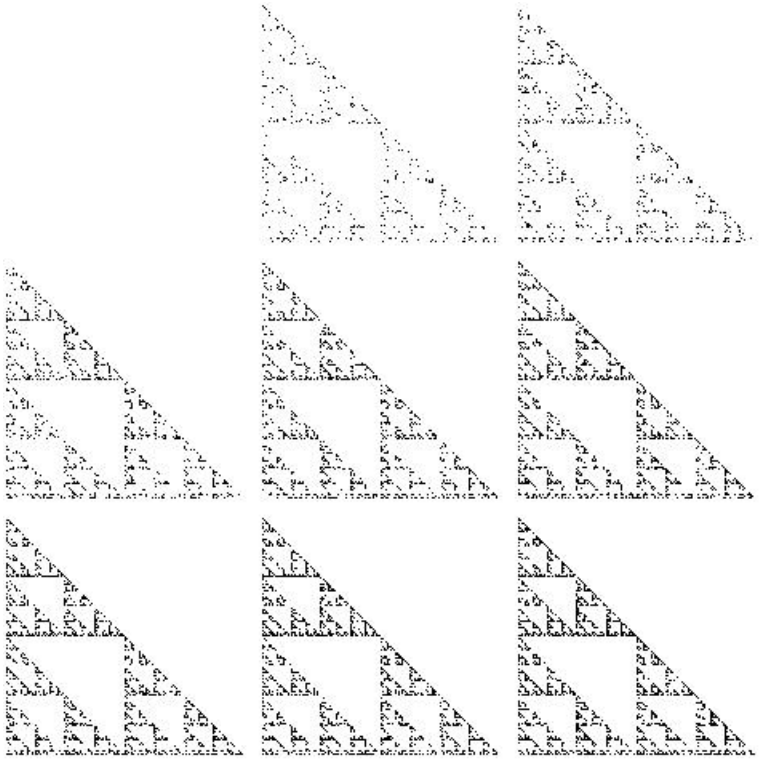

Figure 2.6: Graphical views of the Sierpinski triangle, calculated with the random iteration algorithm for increasing numbers of iterations.

### 2.4.3 The Collage Theorem

The next theorem if useful when one wants to find an IFS whose attractor looks like a given set $L \in \mathcal{H}(\mathcal{X})$. If one can find a set of contraction mappings for which the union of images of $L$ under the transformations is near the original $L$, under the Hausdorff metric, then the attractor of the IFS which consists of those contraction mappings will be close to $L$.

**The Collage Theorem**
Let $(X, d)$ be a complete metric space, let $L \in \mathcal{H}(\mathcal{X})$ and let $\epsilon \geq 0$. Choose an IFS

$\{X; w_1, w_2, ..., w_N\}$ with contractivity factor $0 \leq s < 1$, such that

$$h(L, \bigcup_{n=1}^{N} w_n(L)) \leq \epsilon.$$

Then $h(L, A) \leq \frac{\epsilon}{1-s}$, where $A$ is the attractor of the IFS. Equivalently,

$$h(L, A) \leq (1-s)^{-1} h(L, \bigcup_{n=1}^{N} w_n(L)) \text{ for all } L \in \mathcal{H}(\mathcal{X})$$

The proof follows immediately from the following lemma:

Let $(X, d)$ be a complete metric space. Let $f : X \to X$ be a contraction mapping with contractivity factor $0 \leq s < 1$ and let $x_f \in X$ be the fixed point of $f$. Then

$$d(x, x_f) \leq (1-s)^{-1} \cdot d(x, f(x)) \text{ for all } x \in X$$

**proof:**
$d(x, f) = d(x, \lim_{n\to\infty} f^{\circ n}(x)) = \lim_{n\to\infty} d(x, f^{\circ n}(x))$
$\leq \lim_{n\to\infty} \sum_{m=1}^{n} d(f^{\circ(m-1)}(x), f^{\circ(m)}(x))$
$\leq \lim_{n\to\infty} d(x, f(x))(1 + s + ... + s^{n-1}) \leq (1-s)^{-1} d(x, f(x)).$

Looking at figures 2.7, 2.8 and 2.9 might make this theorem more clear.

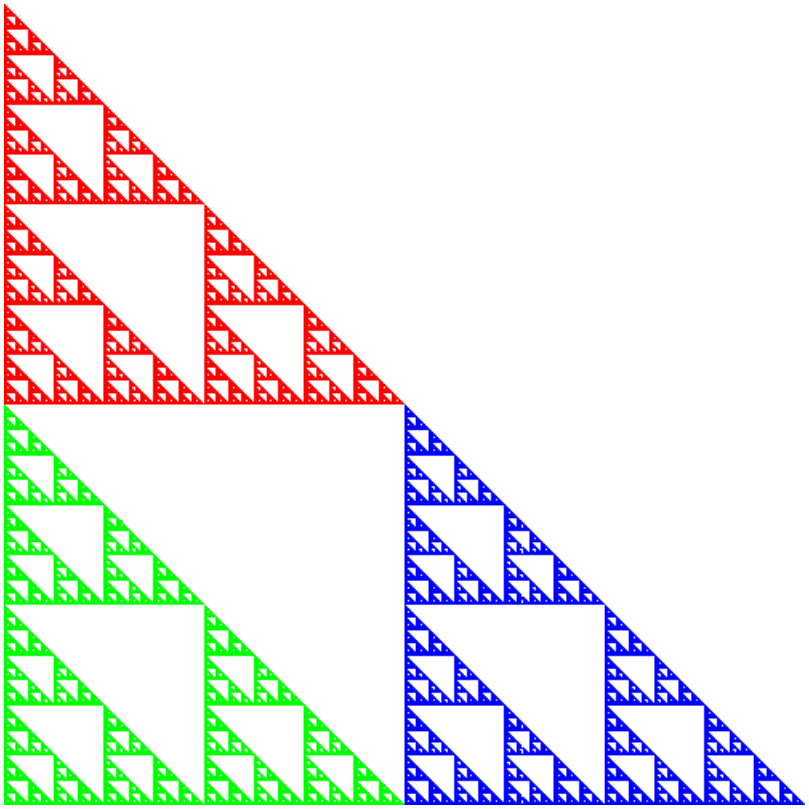

Figure 2.7: The collage of the three transformations of the original picture is equal to the original picture.

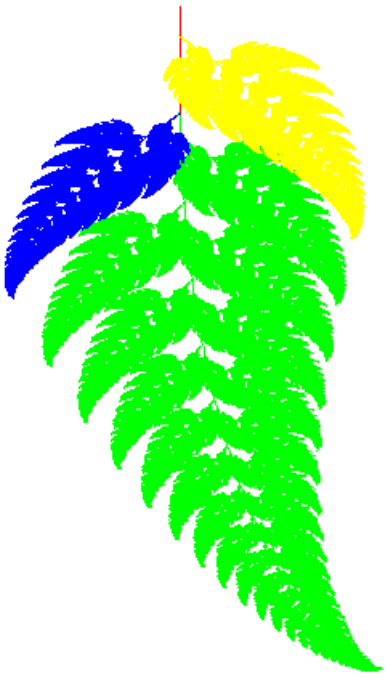

Figure 2.8: The collage of the four transformations of the original picture is equal to the original picture.

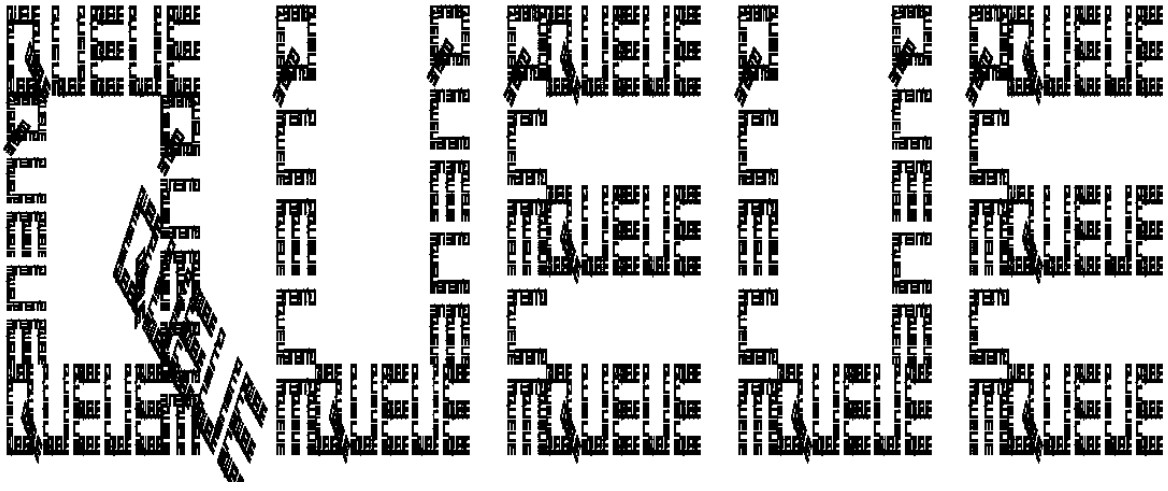

Figure 2.9: A more advanced collage, build from an IFS containing 19 transformations. See figure 2.10 for the specifics.

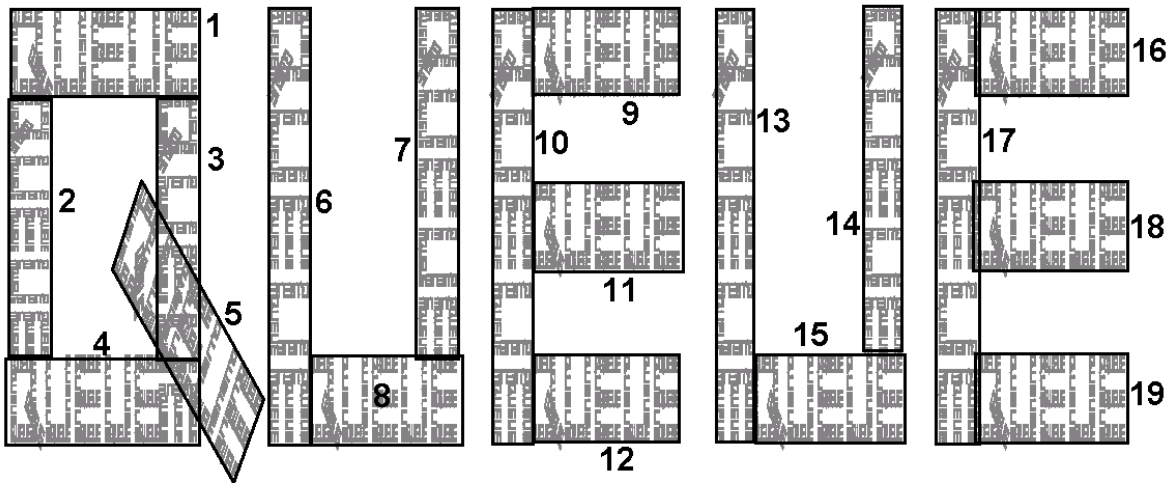


Figure 2.10: This figure shows the different transformations in the IFS that was used to build this fractal.

### 2.4.4 Implementation in Java

In this section an implementation in Java of the algorithms given before in Java is given. This is mostly straightforward from the definition of the algorithms, the main difficulty in the implementation is to take into account the different coordinate systems: the IFS's use arbitrary cartesian coordinates, while we will use some grid of size $x$ by $y$ with coordinates in the integer set $[0, x-1] \times [0, y-1]$ to represent the fractal on the computer.

In this section we only look at how to fill this grid and dispense with the actual drawing details. We define one abstract class `Fractal` which contains all functions for building the fractal, and the extensions of this class will contain the data needed to build the fractal. The following code gives the skeleton code for the class `Fractal`, the details will be shown later:

```
public abstract class Fractal
{
    public double[][] data;         // IFS transformations
    public double[] fs;             // Window size for transformations

    public byte[][] fractal;        // The fractal grid
    BufferedImage fractalImage;     // The image output

    public Fractal (int complexity)
    {
        ...
    }

    // deterministic algorithm
    private void buildFractal (int complexity)
    {
        ...
    }


    // random iteration algorithm
    private void buildFractalRIA (int complexity)
    {
        ...
    }

    // Build image of fractal
    private void buildFractalImage ()
    {
        ...
    }

    // extension classes need to define this
    public abstract void initData (int xRes, int yRes);

    // accessor function
    public BufferedImage get () { return fractalImage; }

}
```

The extension of this abstract class will fill the `data[][]` and `fs[]` fields and defines a starting point for the fractal ($A_0$):

```
public class SierpinskiFractal extends Fractal
{
    public SierpinskiFractal (int complexity) { super (complexity); }
```

```
    public void initData (int xRes, int yRes)
    {
        double[][] initData = { {0.5,   0,  0,  0.5,    0   ,0  ,0.33},
                                {0.5,   0,  0,  0.5,    0   ,50 ,0.33},
                                {0.5,   0,  0,  0.5,    50  ,50 ,0.34}};

        fs = new double[4];
        fs[0] = 0.0;
        fs[1] = 0.0;
        fs[2] = 100.0;
        fs[3] = 100.0;

        data = initData;

        fractal[xRes/2][yRes/2] = 1;
    }
}
```

The `data[][]` fields contain the transformations in the IFS. Only affine transformations are allowed: the first four coordinates in every row of `data[][]` describe the multiplication matrix, coordinates 5 and 6 describe the translation vector and the 7th coordinate gives the probability used for the random iteration algorithm. The `fs[]` fields give the size of the area in which the transformations are taking place: the coordinates in `fs[]` describe the left position, bottom position, width and height of the area respectively. `fractal[][]` will be used to store $A_0$, $A_1$, $A_2$ and so on in different iterations of the algorithm. In this case, $A_0$ is set to one point in the middle of the screen. Next, we will look at the actual implementation of the algorithms.

```
public Fractal (int complexity)
    {
        int xRes = 500;
        int yRes = 500;

        fractal = new byte[xRes][yRes];
        fractalImage = new BufferedImage (xRes, yRes, BufferedImage.TYPE_INT_RGB);

        initData (xRes, yRes);

        buildFractal (complexity);
        buildFractalImage ();

    }
```

The constructor first defines the size of the grid which will contain the fractal, in this case we use a 500 by 500 pixel grid. `fractal[][]` will store temporary data, the $A_0$, $A_1$ and so on, while `fractalImage` will contain the final image which will be used to put the picture on the screen. Next, the constructor calls `initdata ()` which initializes fractal specific data. Then `fractal[][]` is calculated by calling `buildFractal ()` which will iterate the algorithm `complexity` times. Finally a call to `buildFractalImage()` creates the output image. In case one wants to use the random iteration algorithm, the call to `buildFractal ()` is replaced by a call to `buildFractalRIA ()` and now `complexity` gives the number of points drawn. But first we discuss `buildFractal ()`:

```
    // deterministic algorithm
    private void buildFractal (int complexity)
    {
        double oldx, oldy, newx, newy;
```

```
    int xRes = fractal.length;
    int yRes = fractal[0].length;

    byte[][] s = new byte[xRes][yRes];

    for (int N = 0; N < complexity; N++)
    {
        for (int x = 0; x < xRes; x++)
        for (int y = 0; y < yRes; y++)
            if (fractal[x][y] == 1)
                for (int n = 0; n < data.length; n++)
                {
                    // convert to local coordinates
                    oldx = fs[0] + x * fs[2] / xRes;
                    oldy = fs[1] + y * fs[3] / yRes;

                    // apply transformations in local coordinates
                    newx = data[n][0] * oldx + data[n][1] * oldy + data[n][4];
                    newy = data[n][2] * oldx + data[n][3] * oldy + data[n][5];

                    // transform back to grid coordinates
                    s[(int)Math.floor ((newx - fs[0]) * xRes / fs[2])][(int)Math.floor ((newy - fs[1]) * yRes / fs[3])]
                        = 1;
                }

        // update the fractal and clear the temporary grid.
        for (int x = 0; x < xRes; x++)
            for (int y = 0; y < yRes; y++)
            {
                fractal[x][y] = s[x][y];
                s[x][y] = 0;
            }
    }
}
```

The algorithm works as follows: In every iteration we look at every point in `fractal[][]`. If there is a point on a specific position (`fractal[x][y] == 1`), then we apply all transformations to it and store the resulting points. Because the transformations are defined for an arbitrary cartesian coordinate system, the point $(x, y)$ is first transformed to a point in this coordinate system, using the constants in `fs[]` defined before. Then the transformations are applied and the coordinates are transformed back to their position in the fractal grid. The new positions are first saved in a temporary array, `s[][]`. After all points in `fractal[][]` are checked, `fractal[][]` is replaced by `s[][]` and `s[][]` is set to the zero array. The random iteration algorithm is more simple to implement:

```
// random iteration algorithm
private void buildFractalRIA (int complexity)
{
    double oldx, oldy, newx, newy, r;
    int x, y;
    int i;

    int xRes = fractal.length;
    int yRes = fractal[0].length;

    // start in the middle
    x = xRes / 2;
    y = yRes / 2;

    newx = x / (xRes / fs[2]) + fs[0];
    newy = y / (yRes / fs[3]) + fs[1];
```

```
    for (int N = 0; N < complexity; N++)
    {
        // select transformation
        r = Math.random (); i = -1;
        while (r > 0) { r -= data[i+1][6]; i++; }

        // apply transformation
        oldx = newx;
        oldy = newy;
        newx = (data[i][0] * oldx + data[i][1] * oldy + data[i][4]);
        newy = (data[i][2] * oldx + data[i][3] * oldy + data[i][5]);

        // add point to the fractal
        fractal[(int)((newx - fs[0]) * (xRes / fs[2]))][(int)((newy - fs[1]) * (yRes / fs[3]))] = 1;
    }
}
```

Starting with one point in the grid, here chosen to be the point in the middle, repeatedly a transformation is chosen at random and applied to that point. Because we only work with one point at a time, we always represent that point in the cartesian coordinate system of the IFS and we only need to transform the points to fractal grid coordinates when we add them to `fractal[][]`. The last function in the class `Fractal` is almost trivial:

```
// Build image of fractal
private void buildFractalImage ()
{
    int xRes = fractalImage.getWidth ();
    int yRes = fractalImage.getHeight ();

    for (int x = 0; x < xRes; x++)
        for (int y = 0; y < yRes; y++)
            if (fractal[x][y] == 1) fractalImage.setRGB (x, y, new Color (0, 0, 0).getRGB ());
            else fractalImage.setRGB (x, y, new Color (255, 255, 255).getRGB ());
}
```

Points on the fractal are just copied on `fractalImage`.

# Chapter 3

# A closer look at the IFS

## 3.1 Introduction

In this chapter we will try to understand why the random iteration algorithm works and look more closely at fractals and IFS's in general. We need to look at fractals from a different perspective, associate points on fractals with addresses in code space, and define so called **dynamical systems** on fractals.

The random iteration algorithm generates a certain **orbit** on the fractal. The so called **shadowing theorem** assures us that we can account for numerical errors during the computation of this orbit with a computer: there exists a real orbit on the fractal that is 'close to' the orbit generated by the random iteration algorithm with numerical errors. It can be shown that this orbit is very likely to be dense in the fractal hence a good approximation for the fractal.

This chapter is loosely based on chapter 4 of the book 'Fractals Everywhere' [1] by Michael Barnsley.

## 3.2 The addresses of points on fractals

In this section we look at a special way to view points in fractals, namely by associating addresses with points. Let $A$ be the attractor of an IFS, then we can associate an address with a point by looking at the sequence of transformations applied to $A$ which lead to it. But there are different types of IFS: for one type of IFS all points have only one address, this type of IFS is called totally disconnected, for another type of IFS 'a few' points have multiple addresses, this type of IFS is called just-touching, and for yet another type of IFS a large proportion of points have more than one address, this type of IFS is called overlapping.

As an example, look at the Sierpinski triangle generated with the IFS $\{\mathbb{R}^2; w_1, w_2, w_3\}$, where the $w_i$'s are defined as:

$$w_1 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.4 & 0 \\ 0 & 0.4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 50 \end{pmatrix}$$

$$w_2 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.4 & 0 \\ 0 & 0.4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 1 \\ 1 \end{pmatrix}$$

$$w_2 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.4 & 0 \\ 0 & 0.4 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 50 \\ 1 \end{pmatrix}$$

We can divide the attractor $A$ of this IFS in three parts: $w_1(A)$, $w_2(A)$ and $w_3(A)$. All points in part $w_1(A)$ have an address starting with '1', all points in part $w_2(A)$ have an address starting with '2' and all points in part $w_3(A)$ have an address starting with '3'. We can make a further subdivision of the attractor by noting that $w_1(A) = w_1(w_1(A) \cup w_2(A) \cup w_3(A))$. All points in $w_1(w_1(A))$ get an address starting with '11', all points in $w_1(w_2(A))$ get an address starting with '12' and so on. See figure 3.1. Note that in this case every point has a unique address, but this is not true in general.

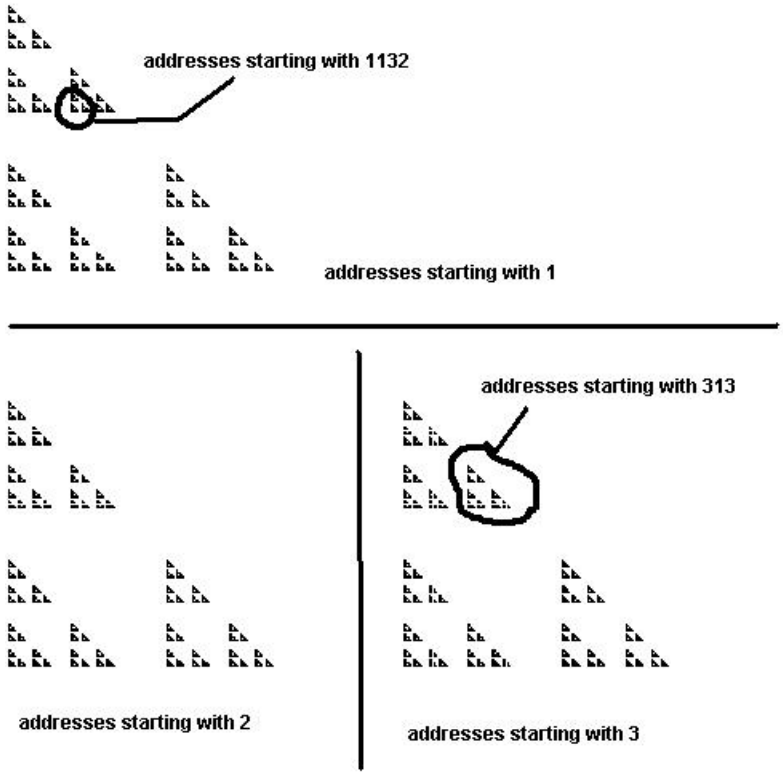


Figure 3.1: One can associate an address with each point on the attractor of the IFS that generates this Sierpinski triangle.

Now we first define the code space, a convenient space to describe addresses, and then associate it with the attractor of the IFS.

The **Code space** $\Sigma$ on $N$ symbols is a space on $N$ symbols $\{1, 2, ..., N\}$ such that for $\sigma \in \Sigma$ we can write $\sigma = \sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6...$

Let $\{X : w_1, w_2, ..., w_N\}$ be an IFS. The **Code space associated with this IFS**, $(\Sigma, d_C)$ is the code space on $N$ symbols $\{1, 2, ..., N\}$ with the metric $d_C$, where $d_C(\omega, \sigma) = \sum_{n=1}^{\infty} \frac{|\omega_n - \sigma_n|}{(N+1)^n}$ for all $\omega, \sigma \in \Sigma$.

Next, we will define a function $\phi : \Sigma \to A$, that maps an element from code space to a point of the attractor. This function is continuous and onto. To proof existence, continuity and surjectivity, we need the following results first:

- Let $(X, d)$ be a complete metric space. Let $\{X; w_1, w_2, ..., w_N\}$ be an IFS. Let $K \in \mathcal{H}(\mathcal{X})$. Then there exists a $\tilde{K} \in \mathcal{H}(\mathcal{X})$ such that $K \subset \tilde{K}$ and $w_n : \tilde{K} \to \tilde{K}$ for $n = 1, 2, ..., N$. So $\{\tilde{K}; w_1, w_2, ..., w_N\}$ is an IFS where the underlying space is **compact**.

  **proof**:
  Let $W : \mathcal{H}(\mathcal{X}) \to \mathcal{H}(\mathcal{X})$ be defined by $W(B) = \bigcup_{n=1}^{N} w_n(B)$ for all $B \in \mathcal{H}(\mathcal{X})$. To construct $\tilde{K}$ look at the IFS with condensation $\{X; w_0, w_1, ..., w_N\}$ where $w_0$ is associated with the condensation set $K$. The attractor of this IFS belongs to $\mathcal{H}(\mathcal{X})$. By writing $\tilde{K} = (K \cup W^{\circ 1}(K) \cup W^{\circ 2}(K) \cup W^{\circ 3}(K) \cup ...)$ it is easy to see that $K \subset \tilde{K}$ and $W(\tilde{K}) \subset \tilde{K}$.

- Let $(X, d)$ be a complete metric space. Let $\{X; w_1, w_2, ..., w_N\}$ be an IFS with contractivity factor $s$. Let $(\Sigma, d_C)$ denote the code space associated with the IFS. Define $\phi(\sigma, n, x) = w_{\sigma_1} \circ w_{\sigma_2} \circ ... \circ w_{\sigma_n}(x)$ for all $\sigma \in \Sigma$, $n \in \mathbb{N}$ and $x \in X$. Let $K$ be a compact nonempty subset of $X$. Then there is a real constant $D$ such that $d(\phi(\sigma, m, x_1), \phi(\sigma, n, x_2)) \leq Ds^{\min(m,n)}$ for all $\sigma \in \Sigma$, all $m, n \in \mathbb{N}$ and all $x_1, x_2 \in K$.

  **proof**:
  Construct $\tilde{K}$ as in the lemma above. Suppose $m < n$. Then $\phi(\sigma, n, x_2) = \phi(\sigma, m, \phi(\omega, n - m, x_2))$ where $\omega = \sigma_{m+1}\sigma_{m+2}... \in \Sigma$. Let $x_3 = \phi(\omega, n - m, x_2)$. Note that $x_3 \in \tilde{K}$. So $d(\phi(\sigma, m, x_1), \phi(\sigma, n, x_2)) = d(\phi(\sigma, m, x_1), \phi(\sigma, m, x_3)) \leq sd(w_{\sigma_2} \circ ... \circ w_{\sigma_m}(x_1), w_{\sigma_2} \circ ... \circ w_{\sigma_m}(x_3)) \leq s^2 d(w_{\sigma_3} \circ ... \circ w_{\sigma_m}(x_1), w_{\sigma_3} \circ ... \circ w_{\sigma_m}(x_3)) \leq s^m d(x_1, x_3) \leq s^m D$, where $D = \max\{d(x_1, x_3) : x_1, x_3 \in \tilde{K}\}$. Since $\tilde{K}$ is compact we know that $D$ is finite.

Now we can state the theorem linking code space to the attractor $A$ of the IFS:

**Theorem**: Let $(X, d)$ be a complete metric space. Let $\{X; w_1, w_2, ...w_N\}$ be a IFS with attractor $A$. Let $(\Sigma, d_C)$ be the code space associated with this IFS. For each $\sigma \in \Sigma$, $n \in \mathbb{N}$ and $x \in X$, let $\phi(\sigma, n, x) = w_{\sigma_1} \circ w_{\sigma_2} \circ ... \circ w_{\sigma_n}(x)$. Then $\phi(\sigma) = \lim_{n \to \infty} \phi(\sigma, n, x)$ exists, belongs to $A$, and is independent of $x \in X$. If $K$ is a compact subset of $X$ then the convergence is uniform over $x \in K$. The function $\phi$ is continuous and onto.

**proof**:

- $\phi$ exists: Let $x \in X$. Let $K \in \mathcal{H}(\mathcal{X})$ be such that $x \in K$. Construct $\tilde{K}$ as described above. Let $W : \mathcal{H}(\mathcal{X}) \to \mathcal{H}(\mathcal{X})$ be defined as $W(B) = \bigcup_{n=1}^{N} w_n(B)$ for all $B \in \mathcal{H}(\mathcal{X})$ as usual. $W$ is a contraction mapping on the metric space $(\mathcal{H}(\mathcal{X}), h)$ and $A = \lim_{n\to\infty}\{W^{\circ n}(K)\}$. So $\{W^{\circ n}(K)\}$ is a Cauchy sequence in $(\mathcal{H}(\mathcal{X}), h))$. Notice that $\phi(\sigma, n, x) \in W^{\circ n}(K)$, and from the previous theorem about the space $(\mathcal{H}(\mathcal{X}), h)$ it follows that if $\lim_{n\to\infty} \phi(\sigma, n, x)$ exists, it belongs to $A$. That this limit exists follows from the fact that $\{\phi(\sigma, n, x)\}$ is a Cauchy sequence: $d(\phi(\sigma, m, x), \phi(\sigma, n, x)) \leq Ds^{\min(n,m)}$ which goes to zero if $n, m$ go to infinity.

- $\phi$ is continuous: Let $\epsilon > 0$ and choose $n$ such that $s^n D < \epsilon$ and let $\sigma, \omega \in \Sigma$ be such that $d_C(\sigma, \omega) < \sum_{m=n+2}^{\infty} \frac{N}{(N+1)^m} = \frac{1}{(N+1)^{n+1}}$. Then it is easy to see that $\sigma$ and $\omega$ must be equal in the first $n$ terms: $\sigma_1 = \omega_1, \sigma_2 = \omega_2, ..., \sigma_n = \omega_n$. So for each $m \geq n$ we can write $d(\phi(\sigma, m, x), \phi(\omega, m, x)) = d(\phi(\sigma, n, x_1), \phi(\sigma, n, x_2))$ for some $x_1, x_2 \in \tilde{K}$. Now we have $d(\phi(\sigma, n, x_1), \phi(\sigma, n, x_2)) \leq s^n D < \epsilon$. Let $m \to \infty$, then $d(\phi(\sigma), \phi(\omega)) < \epsilon$.

- $\phi$ is onto: Let $a \in A$. Then there exists a sequence $\{\omega^{(n)} \in \Sigma : n = 1, 2, 3, ...\}$ such that $\lim_{n\to\infty} \phi(\omega^{(n)}, n, x) = a$. Since $(\Sigma, d_C)$ is compact, $\{\omega^{(n)}\}$ has a convergent subsequence with limit $\omega \in \Sigma$. Assume $\lim_{n\to\infty} \omega^{(n)} = \omega$ without loss of generality. Let $a(n)$ be the number of elements in $\{j \in \mathbb{N} : \omega_k^{(n)} = \omega_k \text{ for } 1 \leq k \leq j\}$, then $a(n) \to \infty$ as $n \to \infty$. So $d(\phi(\omega, n, x), \phi(\omega^{(n)}, x)) \leq s^{a(n)} D$. Let $n \to \infty$ then we have $d(\phi(\omega), a) = 0$ which implies $\phi(\omega) = a$.

Now we can formally define what the address of a point of an attractor is, and make a distinction between the totally disconnected, just-touching and overlapping type of IFS:

Let $\{X; w_1, w_2, ..., w_N\}$ be a IFS with code space $\Sigma$. Let $\phi : \Sigma \to A$ be the continuous function from code space onto the attractor of the IFS constructed before. An **address** of a point $a \in A$ is any member of the set $\phi^{-1}(a) = \{\omega \in \Sigma : \phi(\omega) = a\}$. This set is called the **set of addresses** of $a \in A$.

The IFS is called **totally disconnected** if each point of its attractor has exactly one address. The IFS is called **just-touching** if it is not totally disconnected but its attractor contains an open set $\mathcal{O}$ such that

- $w_i(\mathcal{O}) \cap w_j(\mathcal{O}) = \emptyset$ for all $i, j \in \{1, 2, ..., N\}$ with $i \neq j$.
- $\bigcup_{i=1}^{N} w_i(\mathcal{O}) \subset \mathcal{O}$.

The IFS is called **overlapping** if it is not totally disconnected and not just-touching.

Some further remarks:

- Let $\{X; w_1, w_2, ..., w_N\}$ be an IFS with invertible transformations and attractor $A$. Then the IFS is totally disconnected if and only if $w_i(A) \cap w_j(A) = \emptyset$ for all $i, j \in \{1, 2, ..., N\}$ with $i \neq j$.

- Let $\omega = \omega_1\omega_2\omega_3...$ be an address of a point $x \in A$. Then $\tilde{\omega} = j\omega_1, \omega_2, \omega_3...$ is an address of $w_j(x)$ for each $j \in \{1, 2, ..., N\}$.

Now we will define what a periodic point $a \in A$ is, and show that the set of periodic points is dense in $A$. Let $A$ be the attractor of a IFS $\{X; w_1, w_2, ..., w_N\}$. A point $a \in A$ is called a **periodic point** of the IFS if there exists a finite sequence of numbers $\{\sigma(n) \in \{1, 2, ..., N\})_{n=1}^{p}$ such that $a = w_{\sigma(P)} \circ w_{\sigma(P-1)} \circ ... \circ w_{\sigma(1)}(a)$. If $a$ is periodic, then the smallest number $P$ such that the relation above holds is called the **period** of $a$. Thus a point on the attractor is periodic if there is a finite sequence of transformations that one can apply on that point to get back to exactly the same point. Consider the point $\sigma$ in the associated code space, defined by $\sigma = \sigma(P)\sigma(P-1)...\sigma(1)\sigma(P)\sigma(P-1)...\sigma(1)....$ Then $\phi(\sigma) = a$.

A point in code space whose symbols are periodic is called a **periodic address**. A point in code space whose symbols are periodic after a finite set is omitted is called **eventually periodic**. We have the following result for periodic points:

- The attractor of an IFS is the closure of its periodic points

  **proof**:
  Code space is the closure of its periodic points. Since $\phi$ is a continuous mapping from the code space to $A$, it also holds that $A$ is the closure of its periodic points.

## 3.3 Dynamical systems

In this section we will introduce dynamical systems. The random iteration algorithm produces an orbit of a special type of dynamical system, the shift dynamical system.

### 3.3.1 Deterministic dynamical systems

First, some general definitions for dynamical systems are needed:

- A **dynamical system** is a transformation $f : X \to X$ on a metric space $(X, d)$ and is denoted by $\{X : f\}$. The **orbit** of a point $x \in X$ is the sequence $\{f^{\circ n}(x)\}$.

- Let $\{X : f\}$ be a dynamical system. A **periodic point** of $f$ is a point $x \in X$ such that $f^{\circ n}(x) = x$ for some $n \in \mathbb{N}$. This $n$ is called a **period** of $x$. The smallest period of $x$ is called the **minimal period** of $x$. The orbit of a periodic point of $f$ is called a **cycle** of $f$. The minimal period of a cycle is the number of different points which it contains, and a period of a cycle of $f$ is the period of a point in the cycle.

- Let $\{X : f\}$ be a dynamical system and let $x_f \in X$ be a fixed point of $f$. $x_f$ is called an **attractive** fixed point of $f$ is there exists an $\epsilon > 0$ such that $f$ maps the ball $B(x_f, \epsilon) = \{y \in X : d(x_f, y) \leq \epsilon\}$ into itself, and $f$ is a contraction mapping on $B(x_f, \epsilon)$. The point $x_f$ is called a **repulsive** fixed point of $f$ is there exist numbers $\epsilon > 0$ and $C > 1$) such that $d(f(x_f), f(y)) \geq Cd(x_f, y)$ for all $y \in B(x_f, \epsilon)$.

- A periodic point of $f$ of period $n$ is called **attractive** if it is an attractive fixed point of $f^{\circ n}$. A cycle of period $n$ is an **attractive cycle** of $f$ if it contains an attractive periodic point of $f$ of period $n$. A periodic point of $f$ of period $n$ is called **repulsive** if it is a repulsive fixed point of $f^{\circ n}$ and a cycle of period $n$ is called a **repulsive cycle** of $f$ if it contains a repulsive periodic point of $f$ of period $n$. A point $x \in X$ is called **eventually periodic** if there exists a $m \in \mathbb{N}$ such that $f^{\circ m}(x)$ is periodic.

After the following result, we can define a dynamical system on a (totally disconnected) fractal, the so called shift dynamical system.

- Let $\{X; w_1, w_2, ..., w_N\}$ be an IFS with attractor $A$. If this IFS is totally disconnected, then $w_n$ is one-to-one for each $n \in \{1, 2, ..., N\}$.

  **proof**:
  Suppose that there exist $a_1, a_2 \in A$ such that $w_n(a_1) = w_n(a_2) = a \in A$ for some $n \in \{1, 2, ..., N\}$. If $a_1$ has address $\omega$ and $a_2$ has address $\sigma$ than $a$ has the two addresses $n\omega$ and $n\sigma$. But this is impossible since the IFS is totally disconnected.

Let $\{X; w_1, w_2, ..., w_N\}$ be a totally disconnected IFS with attractor $A$. The associated **shift transformation** on $A$ is the transformation $S : A \to A$ defined by $S(a) = w_n^{-1}(a)$ for all $a \in w_n(A)$. This definition is good according to the lemma above. The dynamical system $\{A; S\}$ is called the **shift dynamical system** associated with the IFS. In figure 3.2 an orbit of a shift dynamical system on the Sierpinski fractal is shown as an example.

Next, we will associate the shift dynamical system on a fractal with a shift dynamical

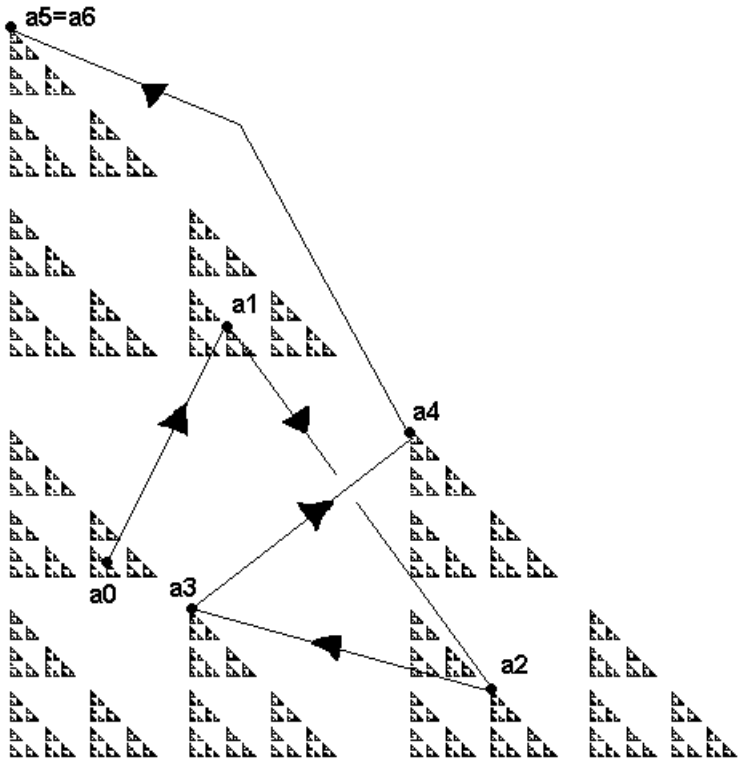


Figure 3.2: An orbit of a shift dynamical system on the Sierpinski fractal.

system on the associated code space. These systems are topologically equivalent using the following definitions:

- Two metric spaces $(X_1, d_1)$ and $(X_2, d_2)$ are called **topologically equivalent** if there exists a homeomorphism $f : X_1 \to X_2$. Two subsets $S_1 \subset X_1$ and $S_2 \subset X_2$ are topologically equivalent or **homeomorphic** if the metric spaces $(S_1, d_1)$ and $(S_2, d_2)$ are topologically equivalent. $S_1$ and $S_2$ are **metrically equivalent** if $(S_1, d_1)$ and $(S_2, d_2)$ are equivalent metric spaces.
- Two dynamical systems $\{X_1; f_1\}$ and $\{X_2; f_2\}$ are said to be **equivalent** or **topologically conjugate** if there exists a homeomorphism $\theta : X_1 \to X_2$ such that
  - $f_1(x_1) = \theta^{-1} \circ f_2 \circ \theta(x_1)$ for all $x_1 \in X_1$.
  - $f_2(x_2) = \theta \circ f_1 \circ \theta^{-1}(x_2)$ for all $x_2 \in X_2$.

Let $\{X; w_1, w_2, ..., w_N\}$ be a totally disconnected IFS and let $\{A; S\}$ be the associated shift dynamical system. Let $\Sigma$ be the associated code space of $N$ symbols and let $T : \Sigma \to \Sigma$ be defined by $T(\sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6...) = \sigma_2\sigma_3\sigma_4\sigma_5\sigma_6...$ for all $\sigma = \sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6... \in \Sigma$. Then the two dynamical systems $\{A; S\}$ and $\{\Sigma, T\}$ are equivalent. The homeomorphism which gives this equivalence is $\phi : \Sigma \to A$. Also, $\{a_1, a_2, ..., a_p\}$ is a repulsive cycle of period $p$ for $S$ if and only if $\{\phi^{-1}(a_1), \phi^{-1}(a_2), ..., \phi^{-1}(a_p)\}$ is a repulsive cycle of period $p$ of $T$.

### 3.3.2 Random dynamical systems

Let's extend the shift dynamical system for a totally disconnected IFS to the just-touching and overlapping cases: the random shift dynamical system.

Let $\{X; w_1, w_2\}$ be an IFS with attractor $A$. Assume that $w_1 : A \to A$ and $w_2 : A \to A$ are invertible. A sequence $\{x_n\}$ of points in $A$ is called an orbit of the **random shift dynamical system** associated with the IFS if:

$$x_{n+1} = \begin{cases} w_1^{-1}(x_n) \text{ when } x_n \in w_1(A), \text{and } x_n \notin w_1(A) \cap w_2(A), \\ w_2^{-1}(x_n) \text{ when } x_n \in w_2(A), \text{and } x_n \notin w_1(A) \cap w_2(A), \\ \text{one of } \{w_1^{-1}(x_n), w_2^{-1}(x_n), \text{when } x_n \in w_1(A) \cap w_2(A), \end{cases}$$

for all $n \in \{0, 1, 2, ...\}$. This is defined as $x_{n+1} = S(x_n)$. $\{A; S\}$ is the collection of possible orbits defined here, and $\{A; S\}$ is called the **random shift dynamical system** associated with the IFS.
If $w_1(A) \cap w_2(A) = \emptyset$ then the IFS is totally disconnected and the orbits defined for the random shift dynamical system for this IFS are equal to the orbits of the shift dynamical system $\{A; S\}$ defined earlier.

By adding an additional variable the random dynamics of the random shift dynamical system can be described by a totally deterministic dynamical system, the lifted shift dynamical system. This deterministic dynamical system is totally disconnected, and one can derive properties of the random dynamical system from this. A totally disconnected dynamical system is in a way 'nicer' then a just-touching or an overlapping dynamical system, this is the reason why we define the lifted shift dynamical system.

The **lifted** IFS associated with an IFS $\{X; w_1, w_2\}$ is the IFS $\{X \times \Sigma; \tilde{w}_1, \tilde{w}_2\}$ where $\Sigma$ is the code space on two symbols $\{1, 2\}$ and

$$\tilde{w}_1(x, \sigma) = (w_1(x), 1\sigma) \text{ for all } (x, \sigma) \in X \times \Sigma$$

$$\tilde{w}_2(x,\sigma) = (w_2(x), 2\sigma) \text{ for all } (x,\sigma) \in X \times \Sigma$$

For the attractor of the IFS $\{X \times \Sigma; \tilde{w}_1, \tilde{w}_2\}$ we have $\tilde{A} = \{(\phi(\sigma), \sigma) : \sigma \in \Sigma\}$, and $A = \{x \in X : (x,\sigma) \in \tilde{A} \text{ for some } \sigma \in \Sigma\} = \phi(\Sigma)$. So $A$ is simply the projection of the attractor of the lifted IFS into the original space $X$. And the projection of $\tilde{A}$ into $\Sigma$ is $\Sigma$. For the lifted IFS we now have the following result:

- Let $\{X; w_1, w_2\}$ be an IFS with attractor $A$ and let the transformations $w_1$ and $w_2$ be invertible. Then the associated lifted IFS is totally disconnected.

Let $\{X; w_1, w_2\}$ be an IFS and let the transformations $w_1$ and $w_2$ be invertible. Let $\tilde{A}$ be the attractor of the associated lifted IFS. Then the shift dynamical system $\{\tilde{A}, \tilde{S}\}$ associated with the lifted IFS is called the **lifted shift dynamical system** associated with the IFS. Note that $\tilde{S}(x,\sigma) = (w_{\sigma_1}^{-1}(x), T(\sigma))$ for all $(x,\sigma) \in \tilde{A}$, with $T(\sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6...) = \sigma_2\sigma_3\sigma_4\sigma_5\sigma_6...$ for all $\sigma = \sigma_1\sigma_2\sigma_3\sigma_4\sigma_5\sigma_6... \in \Sigma$.

### 3.3.3 The shadow theorems for dynamical systems

The following theorem shows why the lifted shift dynamical system is useful: orbits of the random shift dynamical system can be described by orbits in a higher dimensional deterministic shift dynamical system.

**The Shadow Theorem**: Let $\{X; w_1, w_2\}$ be an IFS of invertible transformations $w_1$ and $w_2$ and with attractor $A$. Let $\{x_n\}$ be any orbit of the associated random shift dynamical system $\{A; S\}$. Then there exists an orbit $\{\tilde{x}_n\}$ of the lifted dynamical system $\{\tilde{A}; \tilde{S}\}$ such that the first component of $\tilde{x}_n$ is $x_n$ for all $n$.

The next theorem shows that when one calculates an orbit of a (random) shift dynamical system with a computer, which will contain numerical errors, there still exists an orbit in the dynamical system which is 'close' to the calculated orbit. It also shows that if one manipulates an orbit by making small adjustments, for example to force the orbit to visit a large amount of different points, then there will be an orbit which is 'close' to this orbit.

**The Shadowing Theorem**: Let $\{X; w_1, w_2, ..., w_N\}$ be an IFS with contractivity factor $s$. Let $A$ be the attractor of the IFS and suppose that each of the transformations $w_n : A \to A$ is invertible. Let $\{A; S\}$ be the associated random shift dynamical system (or the associated shift dynamical system if the IFS is totally disconnected). Let $\{\tilde{x}_n\} \subset A$ be an approximate orbit of $S$, such that $d(\tilde{x}_{n+1}, S(\tilde{x}_n)) \leq \theta$ for all $n$ and for some fixed constant $\theta$, $0 \leq \theta \leq diam(A)$. Then there exists an exact orbit $\{x_n = S^{\circ n}(x_0)\}$ for some $x_0 \in A$, such that $(\tilde{x}_{n+1}, x_{n+1}) \leq \frac{s\theta}{1-s}$ for all $n$.

**proof**:
Choose $\sigma_n \in \{1, 2, ...N\}$ such that $w_{\sigma_1}^{-1}, w_{\sigma_2}^{-1}, ...$ is the sequence of inverse maps used to compute $S(\tilde{x}_0), S(\tilde{x}_1), ....$ Let $\phi : \Sigma \to A$ denote the code space map associated with the IFS. Define $x_0 = \phi(\sigma_1\sigma_2\sigma_3...)$. Then we can compute the exact orbit of $x_0$, $\{x_n = S^{\circ n}(x_0) = \phi(\sigma_{n+1}\sigma_{n+2}...)\}$, and compare this orbit with the orbit $\{\tilde{x}_n\}$.

Let $M$ be a large positive integer. Since $x_M$ and $S(\tilde{x}_{M-1})$ are both in $A$, we have $d(S(x_{M-1}), S(\tilde{x}_{M-1})) \leq \text{diam}(A) < \infty$. Because $S(x_{M-1})$ and $S(\tilde{x}_{M-1})$ are both computed with the same inverse map $w_{\sigma_M}^{-1}$ it follows that $d(x_{M-1}, \tilde{x}_{M-1}) \leq s \cdot \text{diam}(A)$. So $d(S(x_{M-2}), S(\tilde{x}_{M-2})) = d(x_{M-1}, S(\tilde{x}_{M-2})) \leq d(x_{M-1}, \tilde{x}_{M-1}) + d(\tilde{x}_{M-1}, S(\tilde{x}_{M-2})) \leq \theta + s \cdot \text{diam}(A)$. Analogously we can derive $d(x_{M-2}, \tilde{x}_{M-2}) \leq s(\theta + s \cdot \text{diam}(A))$. Repeating this argument $k$ times: $d(X_{M-k}, \tilde{x}_{M-k}) \leq s\theta + s^2\theta + ... + s^{k-1}\theta + s^k \cdot \text{diam}(A)$. So for any integers $M$ and $n$ with $0 < n < M$ we have $d(x_n, \tilde{x}_n) \leq s\theta + s^2\theta + ... + s^{M-n-1}\theta + s^{M-n} \cdot \text{diam}(A)$. Take the limit $M \to \infty$ to get $d(x_n, \tilde{x}_n) \leq s\theta(1 + s + s^2 + ...) = \frac{s\theta}{1-s}$ for all $n$.

To show how the Shadowing theorem works an example is given below. We look again at the following IFS for the Sierpinski triangle: $\{\mathbb{R}^2; w_1, w_2, w_3\}$, where the $w_i$'s are defined as:

$$w_1 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0.5 \end{pmatrix}$$

$$w_2 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

$$w_2 \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 0.5 & 0 \\ 0 & 0.5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} + \begin{pmatrix} 0.5 \\ 0 \end{pmatrix}$$

One can define the shift as $S(x_1, x_2) = (2x_1 \mod 1, 2x_2 \mod 1)$. Starting with the point $x_0 = (0.2147, 0.0353)$ we calculate an exact orbit and an errorful orbit. At the errorful orbit errors of size 0.0001 are introduced at each step. The shadowing theorem says that there exists an exact shadowing orbit that has distance less then $\frac{0.5*0.0001}{1-0.5} = 0.0001$ from the errorful orbit. The following table shows this:

| Errorful | Exact | Shadowing | distance |
|---|---|---|---|
| $(0.2147, 0.0353)$ | $(0.2147, 0.0353)$ | $(0.214773, 0.0353358)$ | 0.00010 |
| $(0.4295, 0.0706)$ | $(0.4294, 0.0706)$ | $(0.429546, 0.0706717)$ | 0.00011 |
| $(0.8591, 0.1413)$ | $(0.8588, 0.1412)$ | $(0.859091, 0.141343)$ | 0.00005 |
| $(0.7182, 0.2827)$ | $(0.7176, 0.2824)$ | $(0.718182, 0.282687)$ | 0.00003 |
| $(0.4363, 0.5653)$ | $(0.4352, 0.5648)$ | $(0.436364, 0.565373)$ | 0.00014 |
| $(0.8727, 0.1307)$ | $(0.8704, 0.1296)$ | $(0.872728, 0.130747)$ | 0.00008 |
| $(0.7454, 0.2614)$ | $(0.7408, 0.2592)$ | $(0.745456, 0.261494)$ | 0.00015 |
| $(0.4909, 0.5229)$ | $(0.4816, 0.5184)$ | $(0.490913, 0.522987)$ | 0.00010 |
| $(0.9818, 0.0459)$ | $(0.9632, 0.0368)$ | $(0.981825, 0.045975)$ | 0.00010 |
| $(0.9636, 0.0919)$ | $(0.9264, 0.0736)$ | $(0.96365, 0.09195)$ | 0.00010 |
| $(0.9273, 0.1839)$ | $(0.8528, 0.1472)$ | $(0.9273, 0.1839)$ | 0.00000 |

## 3.4 Why the Random Iteration Algorithm works

In this section some more intuition is given why the random iteration works.

Suppose we have an IFS $\{\mathbb{R}^2; w_1, w_2\}$, let $a \in A$ and let $\sigma \in \Sigma$ be the address of $a$: $\phi(\sigma) = a$.

Generate a large amount of random numbers: 112221...12, and compute the following sequence of points:
$a = \phi(\sigma)$
$w_1(a) = \phi(1\sigma)$
$w_1 \circ w_1(a) = \phi(11\sigma)$
$w_2 \circ w_1 \circ w_1(a) = \phi(211\sigma)$
$w_2 \circ w_2 \circ w_1 \circ w_1(a) = \phi(2211\sigma)$
$w_2 \circ w_2 \circ w_2 \circ w_1 \circ w_1(a) = \phi(22211\sigma)$
$w_1 \circ w_2 \circ w_2 \circ w_2 \circ w_1 \circ w_1(a) = \phi(122211\sigma)$
...
$w_2 \circ w_1 \circ ... \circ w_1 \circ w_2 \circ w_2 \circ w_2 \circ w_1 \circ w_1(a) = \phi(21...122211\sigma)$
This is roughly equivalent to what the random iteration algorithm does. Now look at the calculated points in reversed order, these points form an orbit of the shift dynamical system $\{A; S\}$, namely $\{S^{\circ n}(\phi(21...122211\sigma))\}$. We need to show that this orbit is very likely to be dense in the fractal, that it is a good approximation for the fractal. One way to get some intuition for this is to use the shadowing theorem. By making small adjustments in the orbit one can force the orbit to come close to every point in the attractor of the fractal. And the shadowing theorem now says that there **is** an actual orbit that is close to this adjusted orbit, and this orbit also comes close to every point in the attractor. This suggests that most orbits are **dense** in the attractor. To show that, we can use the following lemma for totally disconnected IFS's:

- Let $\{A; S\}$ be a shift dynamical system associated with a totally disconnected IFS $\{X; w_1, w_2, ..., w_N\}$. Let $\mathcal{N}(p)$ denote the number of distinct cycles of minimal period $p$ for $p \in \{1, 2, 3, ...\}$. Then

$$\mathcal{N}(p) = (N^p - \sum_{k=1, k \text{ divides } p}^{p-1} k\mathcal{N}(k))/p \text{ for } p = 1, 2, 3, ...$$

  **proof**:
  We show that the lemma holds for $N = 2$. For $p = 1$ the cycles of period 1 are the fixed points of $T$. $T(\sigma) = \sigma, \sigma \in \Sigma$ implies $\sigma = 11111...$ or $\sigma = 22222...$ so $\mathcal{N}(p) = 2$. For $p = 2$ we have $T^{\circ 2}\sigma = \sigma$, so $\sigma = 11111...., 121212...., 212121....$ or 22222. The only cycles that are not of period two must have minimal period 1. Also, we have two distinct points on a cycle of period 2, so $\mathcal{N}(2) = (2^2 - \mathcal{N}(1))/2 = 1$. Induction on $p$ gives the result for $N = 2$.

This lemma basically says that a large amount of cycles has a large period. Using $N = 2$ we get for example $\mathcal{N}(2) = 1$, $\mathcal{N}(3) = 2$, $\mathcal{N}(4) = 3$, $\mathcal{N}(5) = 6$, $\mathcal{N}(6) = 9$, $\mathcal{N}(7) = 18$, $\mathcal{N}(8) = 30$, $\mathcal{N}(9) = 56$, $\mathcal{N}(\infty\prime) = 99$, $\mathcal{N}(11) = 186$, $\mathcal{N}(12) = 335$, $\mathcal{N}(13) = 630$, $\mathcal{N}(14) = 1161$, $\mathcal{N}(15) = 2182$, $\mathcal{N}(16) = 4080$, $\mathcal{N}(17) = 7710$, $\mathcal{N}(18) = 14532$, $\mathcal{N}(19) = 27594$, $\mathcal{N}(20) = 52377$. In particular, $(1 - \frac{1+3+6+99}{52377}) * 100 = 99.8$ percent of all points lying on cycles of length 20 lie on cycles of minimal period 20.

We know that the set of all periodic cycles is dense in the attractor of the IFS. So the attractor can be approximated by cycles of a finite period, for example period 1000000. Then we can approximate $A$ with $\tilde{A}$, which consists of roughly $2^{1000000}$ points. If we pick a point on $\tilde{A}$ it is very likely that this point has minimal period 1000000, due to the lemma above, and

that the orbit of this point consists of 1000000 distinct points on $\tilde{A}$. It is possible to show that a statistically random sequence of symbols contains every possible finite subsequence. So it is to be expected that the set of 1000000 distinct points on $A$ is very likely to contain at least one representative of every part of the attractor. For the orbit produced by the random iteration algorithm we thus have found two important properties:

- It is very likely that all points of the orbit are different.
- It is very likely that the points of the orbit represent every part of the attractor.

And from this follows that it is very likely that the random iteration algorithm produces a good approximation to the attractor.

For just-touching and overlapping IFS the lemma used above doesn't hold, but since we then have a lifted dynamical system that is totally disconnected, we can still derive the same result.

# Chapter 4

# Iterated Random Functions

## 4.1 Introduction

In earlier chapters we studied the Iterated Function System (IFS), which can be seen as a contraction mapping in the metric space $(\mathcal{H}(X), h)$. It was proved that after repeatedly applying these contraction mappings on an element of $\mathcal{H}(X)$ the images will converge to an element of $\mathcal{H}(X)$ which can loosely be called a 'fractal'. In this section we study a more general class of mappings, the so called **iterated random functions**, of which the IFS's are a subclass. The main theorem of this chapter shows that if these iterated random functions contract 'on the average' then the sequence of images produced by repeatedly applying these random functions will converge to a unique stationary distribution. This theorem has many applications, some applications in stochastics will be discussed.

This chapter contains mainly theory from the article 'Iterated Random Functions' [4] by Persi Diaconis and David Freedman.

## 4.2 Iterated random functions

We will iterate random functions to construct a Markov chain. Let $\{f_\theta : \theta \in \Theta\}$ be a family of functions that map the state space $S$ into itself, $f_\theta : S \to S$ for all $\theta \in \Theta$. Let $\mu$ be a probability distribution on $\Theta$. If the current state is $x \in S$, we move to state $f_\theta(x)$, where $\theta$ is chosen at random from $\mu$. Let $\theta_1, \theta_2, ...$ be independent draws from $\mu$. We can write $X_0 = x_0$, $X_1 = f_{\theta_1}(x_0)$, $X_2 = (f_{\theta_2} \circ f_{\theta_1})(x_0)$,..., so $X_{n+1} = f_{\theta_{n+1}}(X_n)$. It is clear that the chain that is generated this way is a Markov chain, states $X_{n+1}, X_{n+2}, ...$ depend only on $X_n$.

We call this the **forward iteration** starting from $X_0 = x_0$ where $X_{n+1} = f_{\theta_{n+1}}(X_n) = (f_{\theta_{n+1}} \circ ... \circ f_{\theta_2} \circ f_{\theta_1})(x_0)$. Similarly, we can define the **backward iteration** as $Y_{n+1} = (f_{\theta_1} \circ f_{\theta_2} \circ ... \circ f_{\theta_{n+1}})(x_0)$. Although it is easy to see that $X_n$ has the same distribution as $Y_n$, the forward process $\{X_n; n = 0, 1, 2, ...\}$ behaves very differently from the backward process $\{Y_n; n = 0, 1, 2, ...\}$.

An important question is if there is a stationary probability distribution $\pi$ on $S$ such that $P\{X_n \in A\} \to \pi(A)$ as $n \to \infty$. The main theorem in the next section will give sufficient conditions for this to be true.

As a simple example of an iterated random function, let $S$ be $\mathbb{R}$ and define $f_+(x) = ax+1$ and $f_-(x) = ax - 1$, where $0 < a < 1$. Let $\Theta = \{+, -\}$ and suppose $\mu(+) = \mu(-) = \frac{1}{2}$. In figures 4.1 and 4.2 a typical output is shown for the forward and backward processes respectively, where $a$ is set to 0.5. The forward process appears to move almost randomly between -2 and 2, while the backward process appears to converge to a limit. Let $\xi_n = \pm 1$ with probability $\frac{1}{2}$. Then $X_{n+1} = aX_n + \xi_{n+1}$. So we have $X_n = \sum_{k=0}^{n} a^k \xi_{n-k}$ and $Y_n = \sum_{k=0}^{n} a^k \xi_k$, where we define $\xi_0 = X_0$. Using this notation the stationary distribution can be written as $X_\infty = \xi_1 + a\xi_2 + a^2\xi_3 + ....$ This series converges because $0 < a < 1$. Multiplying this distribution with $a$ and adding a new $\xi$ does not change the distribution, so the distribution of $X_n$ with $X_0 =^d X_\infty$ is stationary.

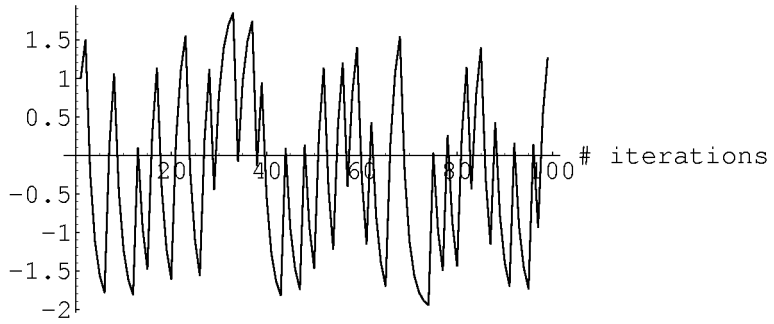


Figure 4.1: Output of the forward process $X_n$ of the iterated random function $f(x) = ax \pm 1$.

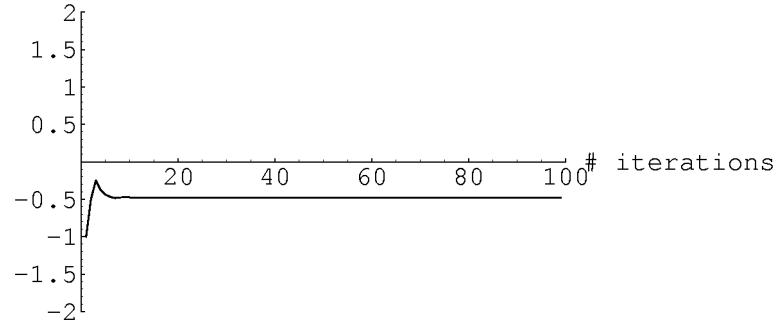


Figure 4.2: Output of the backward process $Y_n$ of the iterated random function $f(x) = ax \pm 1$.

## 4.3 Main theorem

Before the main theorem is stated we define a metric on probabilities and give some other definitions.

- Let P and Q be probabilities on the state space $S$. Define $\rho(P, Q)$, the **Prokhorov metric**, to be the infimum of the $\delta > 0$ such that $P(C) < Q(C_\delta) + \delta$ and $Q(C) < P(C_\delta) + \delta$, for all compact $C \subset S$ and $C_\delta$ the set of all points on $S$ with distance less than $\delta$ from $C$.

Note that from the definition it is clear that $0 \leq \rho(P, Q) \leq 1$.

- Let $S$ be a complete separable metric space with metric $\rho$. A function $f_\theta$ is called **Lipschitz** if there exists a $K_\theta$ such that for all $x, y \in S$ we have $\rho(f_\theta(x), f_\theta(y)) \leq K_\theta \rho(x, y)$

As an example, take $f_+(x) = ax + 1$ from the previous section with the standard Euclidian metric on $\mathbb{R}$. We have $|f_+(x) - f_+(y)| = |ax + 1 - (ay + 1)| = |a(x - y)| = |a||x - y|$, so $K_\theta = |a|$ works. Note that if we can find a $|K_\theta| < 1$ for a Lipschitz function, the function is a contraction: the images of the function are closer together then their originals. In this section, we are looking for families of Lipschitz functions that are contractions 'on the average'.

A final notation we need before stating the main theorem is the kernel $P_n(x, dy)$, which is the distribution of $X_n$ given that $X_0 = x$. This distribution will be compared with the stationary distribution under the Prokhorov metric to be able to say something about the rate of convergence to the stationary distribution.

**Main Theorem:** Let $(S, \rho)$ be a complete separable metric space. Let $\{f_\theta : \theta \in \Theta\}$ be a family of Lipschitz functions on $S$. Let $\mu$ be a probability distribution on $\Theta$. Assume $\int K_\theta \mu(d\theta) < \infty$, $\int \rho[f_\theta(x_0), x_0]\mu(d\theta) < \infty$ for some $x_0 \in S$ and $\int \log K_\theta \mu(d\theta) < 0$. Then

1. The Markov chain generated by $\{f_\theta : \theta \in \Theta\}$ and $\mu$ has a unique stationary distribution $\pi$.
2. $\rho[P_n(x, .), \pi] \leq A_x r^n$ for constants $A_x$ and $r$ with $0 < A_x < \infty$ and $0 < r < 1$, this bound holds for all times $n$ and starting states $x$.
3. The constant $r$ does not depend on $n$ or $x$, the constant $A_x$ does not depend on $n$ and $A_x < a + b\rho(x, x_0)$ where $0 < a, b < \infty$.

The condition $\int \log K_\theta \mu(d\theta) < 0$ makes sure that $K_\theta < 1$ for typical $\theta$ and makes 'contractions on the average' formal. Under this condition and several other regularity conditions, the theorem states that there exists a stationary distribution $\pi$ for the Markov chain generated by $\{f_\theta : \theta \in \Theta\}$, and that convergence is exponential.

We give a sketch of the proof, leaving out much of the technical details. The main step is to prove that the backward iterations converge almost surely to a limit which has the stationary distribution $\pi$. But first we derive some results on the forward proces:

- $\rho[X_n(x), X_n(y)] \leq [\prod_{j=1}^n K_{f_j}]\rho(x, y)$: This can easily be proven by induction: for $n = 0$ and $n = 1$ it is clear, and for $n > 1$ we have $\rho[f_{n+1}(X_n(x)), f_{n+1}(X_n(y))] \leq K_{f_{n+1}} \rho[X_n(x), X_n(y)]$.

- If $\epsilon > 0$ is small enough, there exist positive and finite constants $A$ and $r < 1$ such that $P\{\sum_{i=1}^{n} \log K_{f_i} > -n\epsilon\} < Ar^n$ for all $n = 1, 2, ....$ $A$ and $r$ depend on $\epsilon$ but not on $n$. The proof of this uses mainly some probability theory.

- For small enough $\epsilon > 0$ we have $\rho[X_n(x), X_n(y)] \leq \exp(-n\epsilon)\rho(x, y)$ for all $x, y \in S$, except for a set of $f_1, f_2, ..., f_n$ of probability less than $Ar^n$. $A$ and $r$ depend on $\epsilon$ but not on $n$. This statement follows from the previous two statements.

Now it follows that if there exists an invariant probability, it is unique: Suppose that $\pi$ and $\pi'$ are invariant. Pick $x$ from $\pi$ and $x'$ from $\pi'$ independently. Let $Y_n = X_n(x)$ and $Y'_n = X_n(x')$. We have $\rho(Y_n, Y'_n) \leq \exp(-n\epsilon)\rho(Y_0, Y'_0)$ except for a set of exponentially small probability. So the distributions of $Y_n$ and $Y'_n$ will be almost the same for large $n$, but $Y_n$ has distribution $\pi$ and $Y'_n$ has distribution $\pi'$. Therefore, $\pi$ and $\pi'$ must have the same distribution.

Now we come to the proof that the backward iterations converge to a limit in a stationary distribution. First note that we have $\rho[(f \circ g)(x), x] \leq \rho[f(x), x] + K_f\rho[g(x), g]$, using the triangle inequality: $\rho[(f \circ g)(x), x] \leq \rho[f(x), x] + \rho[(f \circ g)x, f(x)] \leq \rho[f(x), x] + K_f\rho[g(x), g]$. We can repeatedly apply this argument to get:

- Let $\{g_i\}$ be mappings of $S$ into itself and let $x \in S$. Then $\rho[(g_1 \circ g_2 \circ ... \circ g_m)(x), x)] \leq \rho[g_1(x), x] + K_{g_1}\rho[g_2(x), x] + K_{g_1}K_{g_2}\rho[g_3(x), x] + ... + K_{g_1}K_{g_2}...K_{g_{m-1}}\rho[g_m(x), x]$.

Now consider the backward iterations $Y_n(x) = (f_1 \circ f_2 \circ ... \circ f_n)(x)$. We have $\rho[Y_{n+m}(x), Y_n(x)] \leq K_{f_1}...K_{f_n}\rho[(f_{n+1} \circ f_{n+2} \circ ... \circ f_{n+m})(x), x]$. Using the previous result, we get $\rho[Y_{n+m}(x), Y_n(x)] \leq \sum_{i=0}^{\infty}(\prod_{j=1}^{n+i} K_{f_j}\rho[f_{n+i+1}(x), x])$. By using the previous result, $\prod_{j=1}^{n+i} K_{f_j} \leq e^{-(n+i)\epsilon}$ except for a set of small probability, and the initial conditions of the theorem we can now find finite positive constants $c_0, r_0 < 1, r_1 < 1$ such that for all $n_0$ and all $n \geq n_0$ and $m = 0, 1, ...$ we have $\rho[Y_{n+m}(x), Y_n(x)] \leq r_1^n$ except for a set of probability $c_0 r_0^{n_0}$. This says that $Y_n(x)$ is a Cauchy sequence and thus converges to a limit in $S$.

Since the backward iterations converge to a limit in a stationary distribution, the forward iterations will converge to a stationary distribution since the marginal distributions are the same. For more details see [4]. In the next section some examples are given that use this theorem.

## 4.4 Applications

### 4.4.1 A simple example

Consider a Markov chain with state space $S = (0, 1)$, the open unit interval. If the chain is at state $x$ the next state is in $(0, x)$ or $(x, 1)$ with equal probability $1/2$ and it assumes a random position in the chosen interval. So $k(x, y) = P(X_{n+1} = y | X_n = x) = \frac{1}{2}\frac{1}{x}\chi_{(0,x)}(y) + \frac{1}{2}\frac{1}{1-x}\chi_{(x,1)}(y)$ where $\chi_A(y) = 1$ if $y \in A$ and $\chi_A(y) = 0$ if $y \notin A$.

This Markov chain can be constructed using random iterated functions: Let $\phi_u(x) = ux, \psi_u(x) = x + u(1 - x)$ with $u$ chosen uniformly on $(0, 1)$ and $\phi, \psi$ chosen with equal probability $1/2$. The assumptions of the theorem apply: $\int \log K_\theta \mu(d\theta) < 0$ is clear since $K_\phi = u < 1$ and $K_\psi = 1 - u < 1$, and the conditions $\int K_\theta \mu(d\theta) < \infty$ and $\int \rho[f_\theta(x_0), x_0]\mu(d\theta) < \infty$ for some

$x_0 \in S$ also apply since we are working with nice non-degenerate functions that occur with probability $\frac{1}{2}$. Thus the theorem shows that there is a unique stationary distribution. After some calculations, one can find its density. First assume that the stationary distribution has a density $f(x)$. Then we have $f(y) = \int_0^1 k(x,y)f(x)dx = \frac{1}{2}\int_y^1 \frac{f(x)}{x}dx + \frac{1}{2}\int_0^y \frac{f(x)}{1-x}dx$. Differentiating this gives $f'(y) = -\frac{1}{2}\frac{f(y)}{y} + \frac{1}{2}\frac{f(y)}{1-y}$ so $\frac{f'(y)}{f(y)} = \frac{1}{2}(-\frac{1}{y} + \frac{1}{1-y})$, so $f(y) = \frac{1}{\pi\sqrt{x(1-x)}}$.

### 4.4.2 G/G/1 queue

Consider a $G/G/1$ queue with customers arriving at a queue with independently and identically distributed interarrival times $U_1, U_2, ....$ So the arrival times are the sums $U_1, U_1+U_2, ....$ Customer $j$ has service time $V_j$, where the $V_j$'s are also independent and identically distributed and independent of the arrival times. Let $W_j$ be the waiting time of customer $j$, where the waiting time denotes the time until service starts. We have $W_0 = 0$. For $j > 0$ we have $W_{j+1} = (W_j + V_j - U_{j+1})^+$. This is easy to see: If customer $j$ arrives at time $T_j = U_1 + ... + U_j$ and waits time $W_j$, his service is finished at time $T_j + W_j + V_j$. Customer $j+1$ arrives at time $T_j + U_{j+1}$. If $T_j + U_{j+1} > T_j + W_j + V_j$ then $W_{j+1} = 0$, else $W_{j+1} = W_j + V_j - U_{j+1}$.

In order to generate a sequence of waiting times $\{W_j : j = 0, 1, 2, ...\}$ we can the iterate random functions $f_\theta(x) = (x + \theta)^+$, where $\theta$ should be chosen from the distribution of $V - U$. For the functions $f_\theta$ the Lipschitz constant is 1, so the main theorem does not apply. But in this case, the use of random iterated functions is still useful: the backward iteration still gives the stationary distribution. For this distribution we have: $(f_{\theta_1} \circ ... \circ f_{\theta_n})(0) = (\theta_1 + (\theta_2 + ... + (\theta_{n-1} + \theta_n^+)^+)^+)^+ = \max_{1 \le j \le n}(\theta_1 + ... + \theta_j)^+$. The last equality can be proven easily using induction. Now set $X_j = V_j - U_{j+1}$, then we have an expression for the stationary distribution: $\lim_{n\to\infty} \max_{1 \le j \le n}(X_1 + ... + X_j)^+$, provided that this limit exists almost surely. Under certain regularity conditions it can be shown that this stationary distribution indeed exists, for example when $E(X_1) < 0$.

# Chapter 5

# Lindenmayer Systems

## 5.1 Introduction

In this section an alternative way to look at fractals is presented, including how to make pictures of fractals. This alternative way is to use **rewriting** as a way to define complex objects, in this case fractals. Starting with a simple object, parts of it are successively replaced using a set of rewrite rules or productions. A simple example is the snowflake curve, see figures 5.1 and 5.2. Starting with a simple triangle consisting of three edges ,the **initiator**, one replaces in each iteration every edge with a curve called the **generator**.

There are many ways to define and use rewriting. One way is to look at the rewriting of character strings, and in this section we look at one special type of rewriting character strings, the so called Lindenmayer systems, developed by the biologist Lindenmayer. Characteristic for this type of rewriting is to replace all characters in the character strings simultaneously. This makes this type of rewriting ideal for modelling biological phenomena, for example plant development or bacteria growth.

In the next section some basic definitions are given. Section 5.3 gives some extensions on Lindenmayer systems. In section 5.4 a practical implementation of Lindenmayer systems is given, and this implementation was used to generate most of the pictures in this chapter. The last section looks at the growth function of stochastic lindenmayer systems. Sections 5.2 and 5.3 are based on the book 'The Algorithmic Beauty Of Plants' [2] by Przemyslaw Prusinkiewics and Astrid Lindenmayer. The last section makes use of the article 'Growth Functions of Stochastic Lindenmayer Systems' [3] by Peter Eichhorst.

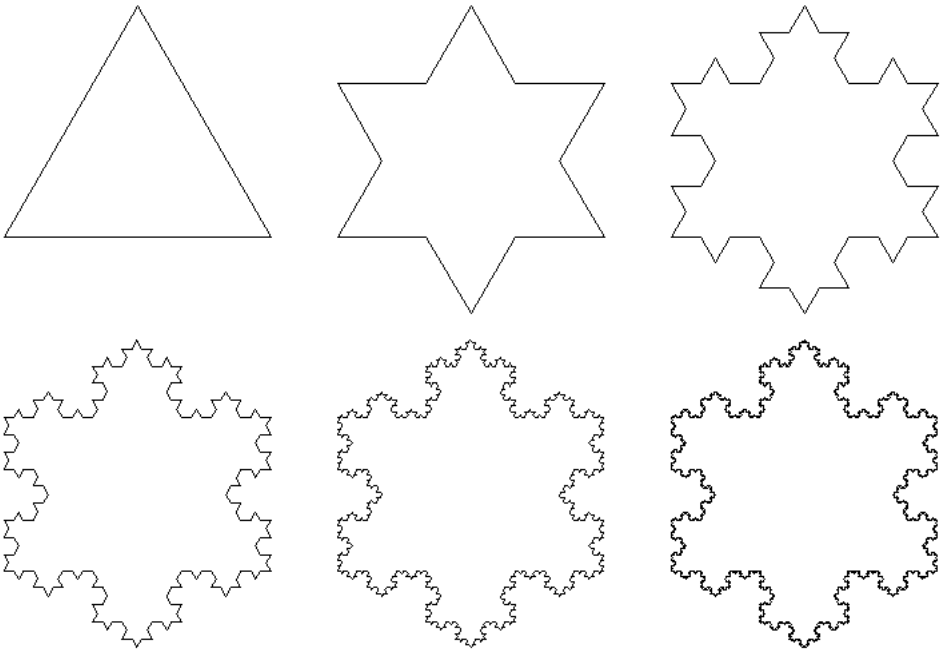

Figure 5.1: Successive iterations of the snowflake curve.

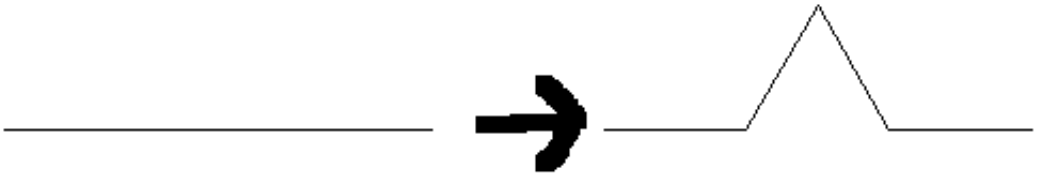

Figure 5.2: The rewrite rule of the snowflake curve. The curve on the right is called the generator.

## 5.2 Basic Lindenmayer

In this section the basic definition of a Lindenmayer system is given, including some examples. Lindenmayer systems will be called **L-systems**, and in this section we look at a simple type of L-system: the (Deterministic) OL-system which is a context-free version of a Lindenmayer system. Context-free means here that in every iteration of the system every character in the string is treated individually without considering the influence of neighbour characters.

- **OL-system**: An OL-system is a triple $(\Sigma, R, \omega)$, with $\Sigma$ a finite set of symbols called the **alphabet**, $R$ a finite set of rewrite rules called **productions** of the form $\Sigma \to \Sigma^*$, and $\omega \in \Sigma^+$ called the **axiom**. For every $a \in \Sigma$ there must be at least one production in $R$ which maps $a$ to $\alpha \in \Sigma^*$. If there is exactly one production in $R$ for every $a$, then the system is **deterministic**, and is called a **DOL-system**

As a first simple example of a DOL-System, let $\Sigma = \{a, b\}$, $R = \{a \to ab, b \to a\}$ and let the axiom be $b$. Now one can iteratively replace the axiom with a new string by applying the productions on every character in the string:
$b \to a \to ab \to aba \to abaab \to abaababa \to ...$

Some more definitions are needed:

- **derivation in an 0L system**: Let $x$ and $y$ be in $\Sigma^*$. $y$ **directly derives** $x$ ($y \Rightarrow x$) if $y = a_1a_2...a_n, a_i \in \Sigma$ for all $i = 1, .., n$ and $x = \alpha_1\alpha_2...\alpha_n, \alpha_i \in \Sigma^*$ for all$i = 1, ..., n$ and $a_i \to \alpha_i$ is a production in $R$ for all $i = 1, ..., n$.

  A derivation of $x$ from $y$ of length $n$ is an ordered pair $d = (T, \sigma)$. $T$, called the **trace** of $d$, is a sequence $\{w_i\}_{i=0}^n$ of $n + 1$ words in $\Sigma$ with $w_0 = y$, $w_n = x$ and $w_i \Rightarrow w_{i+1}, i = 1, ..., n$. $\sigma$ is a function from $\{(i, j)|0 \leq i < n, 1 \leq j \leq |w_i|\}$ to $R$. If $w_i = a_1a_2...a_m$ and $w_{i+1} = \alpha_1\alpha_2...\alpha_m$ then $\sigma(i, j) = a_j \to \alpha_j$ for $j = 1, ..., m$. $|x|$ denotes the number of symbols in $x$, so for example if $w = a_1a_2...a_n, a_i \in \Sigma, i = 1, ..., n$ then $|w| = n$.

For example, in the previous example *abaab* directly derives *abaababa*. Already these simple definitions have some practical applications, they are for example useful for modelling bacteria growth.

There are many ways to interpretate the strings generated by L-systems. A graphical interpretation is to look at the string as controlling a **turtle**, a kind of drawing tool. The most basic idea of the turtle is that it has a **state** $(x, y, \alpha)$, where $(x, y)$ represents the position of the turtle in cartesian coordinates and the angle $\alpha$ represents the heading of the turtle, the direction the turtle is facing. Let a step size $d$ and an angle increment $\delta$ be given, then we can associate commands for the turtle with the following symbols:

- `F`: Move forward with step $d$ and draw a line. The state of the turtle is changed from $(x_1, y_1, \alpha)$ to $(x_2, y_2, \alpha)$ with $x_2 = x_1 + d\cos\alpha$ and $y_2 = y_1 + d\sin\alpha$. A line is drawn between $(x_1, y_1)$ and $(x_2, y_2)$.
- `f`: Move forward with step $d$ without drawing a line.
- `+`: Turn left with angle $\delta$. The state of the turtle is changed from $(x, y, \alpha)$ to $(x, y, \alpha+\delta)$.

- `-`: Turn right with angle $\delta$.

Using only these four symbols it is already possible to draw some nice fractal objects. As a first example, consider the following L-System, called a Koch island:
$\omega$: `F-F-F-F`
$R$: `F->F-F+F+FF-F-F+F`
The trivial productions `+ -> +` and `- -> -` which are technically part of $R$ are omitted here. $\delta$ is set to 90 degrees. Successive iterations of this L-System are shown in figure 5.3. Several other examples of Koch islands are shown in figures 5.4, 5.5 and 5.6. The pictures might give some intuition about the power of L-system: using some simple rules very complex objects can be created.

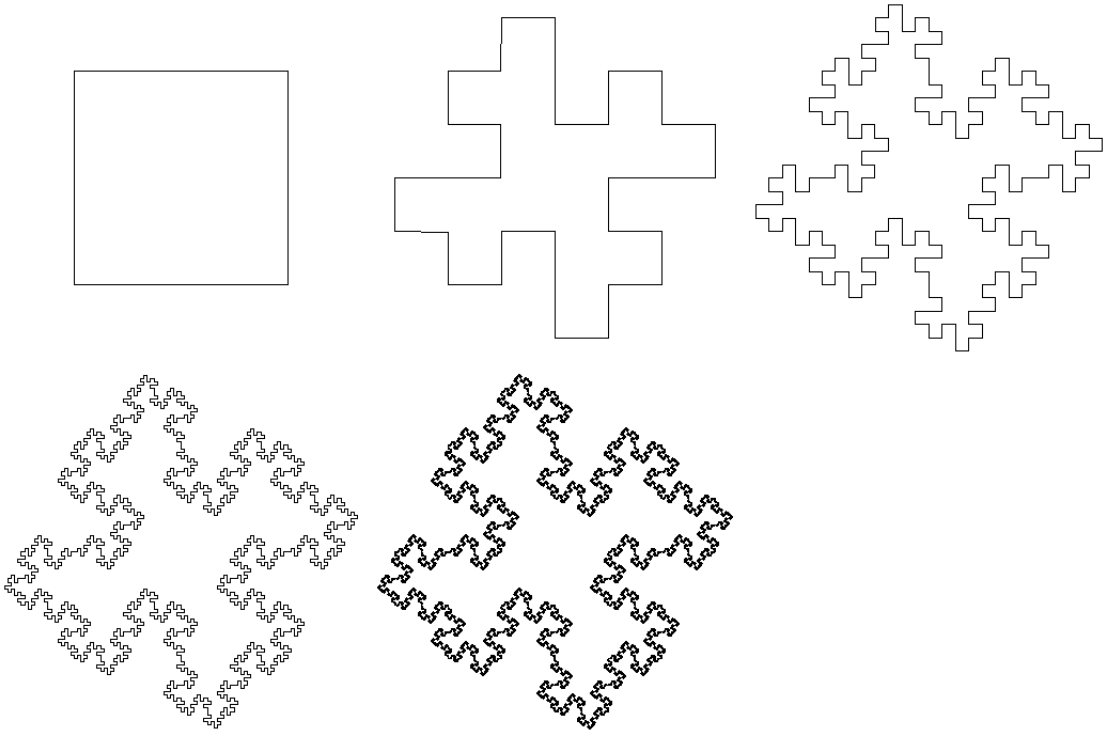

Figure 5.3: Successive iterations of the Koch island generated using the following L-System:
$\omega : F - F - F - F$,
$R : F- > F - F + F + FF - F - F + F$

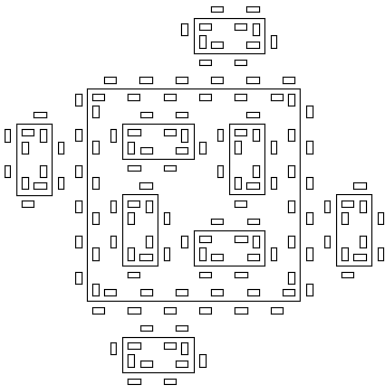

Figure 5.4: The second iteration of the Koch island generated using the following L-System:
$\omega : F + F + F + F$,
$R : F- > F + f - FF + F + FF + Ff + FF - f + FF - F - FF - Ff - FFF, f- > ffffff$

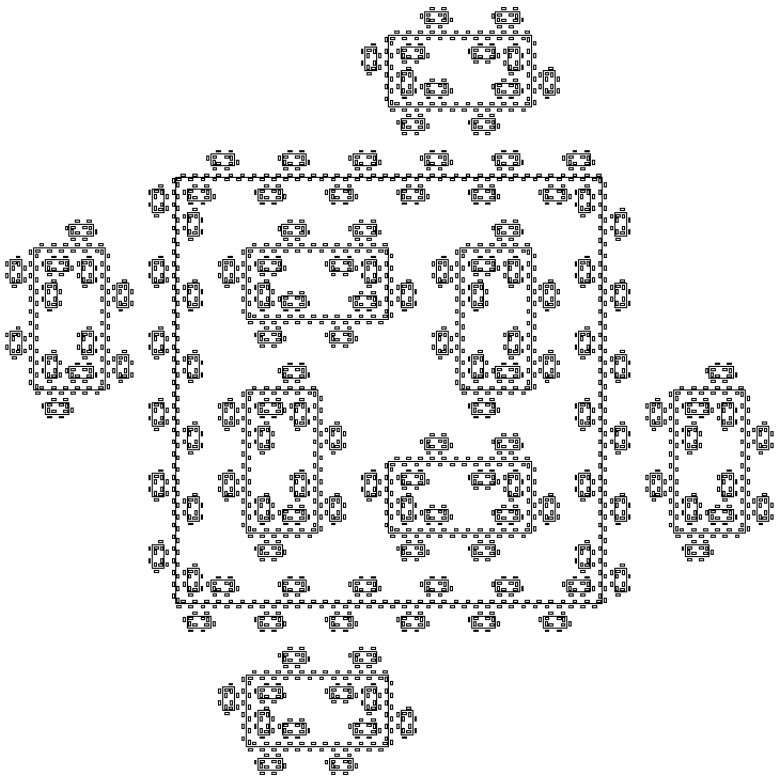

Figure 5.5: The third iteration of the Koch island generated using the following L-System:
$\omega : F + F + F + F$,
$R : F- > F + f - FF + F + FF + Ff + FF - f + FF - F - FF - Ff - FFF, f- > ffffff$
Note the enormous increase in complexity compared to the second iteration shown in the previous figure.

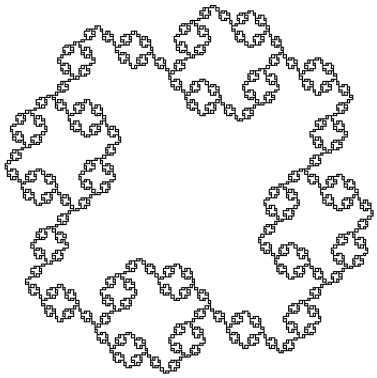

Figure 5.6: The fourth iteration of the Koch island generated using the following L-System:
$\omega : F - F - F - F$,
$R : F- > FF - F - F - F - F - F + F$

L-Systems are also ideal to represent branching structures like trees. By adding two symbols to the L-system which represent new commands for the turtle it is possible to generate nice pictures of trees:

- `[`: Push the current state of the turtle on a pushdown stack.
- `]`: Pop a state from the pushdown stack and make this the current state of the turtle.

This allows the turtle to draw a branch of the tree and then return to the original position on the tree. For example, with `F[+F]F[-F]F` the turtle first draws a line of length $d$, then turns left and draws a second line at the endpoint of the first line, then the turtle returns to the endpoint of the first line and draws a third line extending the first line, then the turtle turns right and draws a fourth line at the endpoint of the third line, and finally returns to endpoint of the third line and draws a fifth line extending the third line, see figure 5.7.

Using the bracket extension, many nice pictures of trees can be generated, see figures

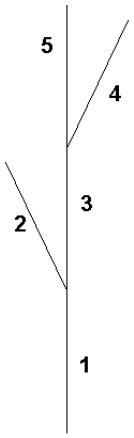


Figure 5.7: Example of using the bracket symbols: this result is obtained from the command string $F[+F]F[-F]F$. The numbering shows the order in which the lines are drawn by the turtle.

5.8 and 5.9. There are many more extensions possible to this turtle model. One can define the turtle in 3d-space to generate three dimensional pictures. The turtle model is also used to generate space filling curves.

Figure 5.8: This tree is generated using the following L-System: $\omega : F$,
$R : F- > F[+F]F[-F]F$. $\delta$ is 25 degrees and this is the fifth iteration.

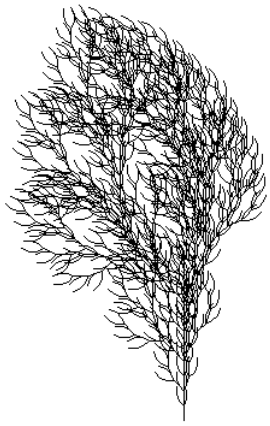

Figure 5.9: This tree is generated using the following L-System: $\omega : F$,
$R : F- > FF-[-F+F+F]+[+F-F-F]$. $\delta$ is 22 degrees and this is the fourth iteration.

## 5.3 Extensions

In this section we mention some further extensions of the L-system defined in the previous section. We start by defining a stochastic L-system, whose definition is analogous to the previous definition of the L-system.

- **S0L system**: A stochastic 0L system, S0L system, is a four tuple $(\Sigma, R, \rho, \omega)$. $\Sigma$ is a finite alphabet, $R$ is a finite set of productions as defined for 0L systems, $\rho$ is a mapping from $\Sigma$ to the interval $(0, 1]$ of real numbers with for all $a \in \Sigma$ that $\sum_{\alpha \in \Sigma^*} \rho(a \to \alpha) = 1$, and $\omega$, called the axiom distribution, is a mapping from $\Sigma^+$ to the interval $[0, 1]$ of real numbers with only finitely many $x \in \Sigma^+$ with $\omega(x) > 0$ and such that $\sum_{x \in \Sigma^+} \omega(x) = 1$. $\omega(x)$ is the probability that $x$ is the axiom for the system and $\rho(a \to \alpha)$ is the probability that $a$ is rewritten by $\alpha$.

- **derivation in a S0L system**: A derivation in a S0L system is defined exactly the same as in 0L system with the addition that a probability is assigned to every possible derivation. Let $S = (\Sigma, R, \rho, \omega)$ be a S0L system and let $d = (T, \sigma)$ be a derivation as defined for 0L systems. $P(\omega_i \Rightarrow \omega_{i+1})$ is the probability that $\omega_i \Rightarrow \omega_{i+1}$. Let $\omega_i = a_1 a_2 ... a_n$, then $P(\omega_i \Rightarrow \omega_{i+1}) = \prod_{j=1}^{m} \rho(\sigma(i, j))$. Now one can define the probability of the derivation $d$, $P(d)$, as $P(d) = \prod_{i=1}^{n-1} P(\omega_i \Rightarrow \omega_{i+1})$.

  Let $x, y \in \Sigma^*$ and let $T_y^n(x)$ be the set of all derivations of $x$ from $y$ of length $n$. Now one can define the probability of deriving $x$ from $y$ in $n$ steps as $P_x^y(x) = \sum_{d \in T_y^n(x)} P(d)$. Let $y_1, y_2, ..., y_m$ be an enumeration of all $y_i \in \Sigma^+$ with $\omega(y_i) > 0$ and let $x \in \Sigma^*$. The probability of deriving $x$ from $\omega$ in $n$ steps is defined by $P_\omega^n(x) = \sum_{i=1}^{m} \omega(y_i) P_{y_i}^n(x)$.

A simple example of a stochastic L-system is given by an extension of the tree example given before:
$\omega$: `F`
$R$: `F->F[+F]F[-F]F (0.33)`, `F->F[+F]F (0.33)`, `F->F[-F]F (0.34)`
Where the number behind each production gives the probability for that production. The result is shown in figure 5.10 for 5 different applications of the algorithm. Note that the trees all look different, but appear to come from the same family of trees.

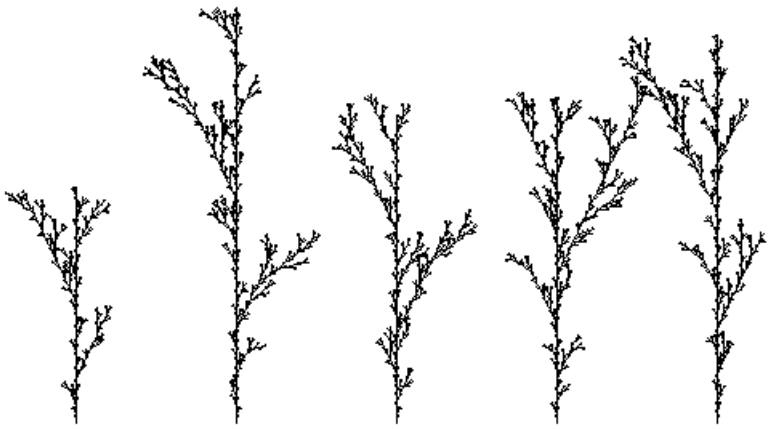

Figure 5.10: These trees are generated using the following stochastic L-System: $\omega : F$, $R : F- > F[+F]F[-F]F(0.33), F- > F[+F]F(0.33), F- > F[-F]F(0.34)$. $\delta$ is 22 degrees and this is the fifth iteration. The trees look different, but appear to come from the same family of trees.

There are many other extensions of the L-system introduced before. Up till now, we have considered context-free L-systems. But to model some types of plant development, the growth of one part of the tree might depend on its surroundings. One could also introduce parameters in L-systems, for example to model the growth of flowers or the transport of fluids in a plant. [2] is a good place to start exploring these extensions.

## 5.4 Implementation in Java

In this section a way to implement the L-system introduced in previous sections is described. This is actually quite straightforward from the definition of an L-system. A problem is how to choose the parameter $d$, since this depends exponentially on the number of iterations. As with the implementation of IFS's defined in a previous chapter, we start with an abstract class, although in this case the most code will be in the extended classes:

```
import java.awt.*;
import java.util.*;


public abstract class LSystem
{
    public String fractalDescriptor;        // a string describing the fractal
    public double d;                         // length of each line drawn by the turtle


    public LSystem (int complexity)          // complexity is the number of iterations
    {
        d = 1.0;

        init (complexity);

        for (int i = 0; i < complexity; i++)
            fractalDescriptor = rewrite (fractalDescriptor);
    }

    // set an appropriate value for d and set the initial value of the
    // string fractalDescriptor.
    public abstract void init (int complexity);

    // describes the productions applied to the string fractalDescriptor
    public abstract String rewrite (String input);

    // generate output by interpreting the string fractalDescriptor as
    // commands for a turtle.
    public abstract void render (Graphics g, int xPos, int yPos);
}
```

This class is pretty much self-explanatory. First the function `init()` is called, which sets values for the length of each line drawn by the turtle (`d`) and the initial value of the string describing the fractal, which is the axiom (`fractalDescriptor`). Then the function `rewrite()` is called a number of times which applies the productions to (`fractalDescriptor`). The resulting descriptor of a fractal, `fractalDescriptor`, is interpreted and drawn by the function `render()`. Next we look at an extended class to see how this works in practice. The class `KochIsland` generates the Koch island discussed before, see also figure 5.3:

```
import java.awt.*;
import java.util.*;

public class KochIsland extends LSystem
{
```

```
public KochIsland (int complexity) { super (complexity); }

public void init (int complexity)
{
    d = 200.0 / (Math.pow (4.0, complexity));
    fractalDescriptor = "F-F-F-F";
}
```

Of the function `init()` only the assignment of `d` needs explanation. For the Koch island, in one iteration of the algorithm the number of lines drawn (the number of "F"'s in the string) grows by a factor 4 so the value of `d` is decreased by a factor 4 for each iteration. Note that in general it is not trivial to find the growth factor, there could be complicated productions and the L-system could even be stochastic. The function `rewrite` is defined next:

```
// implements the productions on the string.
public String rewrite (String input)
{
    String output = "";

    for (int i = 0; i < input.length (); i++)
    {
        switch (input.charAt (i))
        {
            case 'F': output += "F-F+F+FF-F-F+F";
            break;

            case '+': output += "+";
            break;

            case '-': output += "-";
            break;

            default:
            break;
        }
    }

    return output;
}
```

Also this function is straightforward. Every character in the string is replaced by its unique production. Finally, the function `render` is defined:

```
// interpretate the string as commands for a turtle.
public void render (Graphics g, int xPos, int yPos)
{
    double[] turtle = {0.0, 0.0, 0.0};
    double t = Math.PI / 2.0;
    double[] dTurtle = {0.0, 0.0, 0.0};

    g.setColor (Color.BLACK);

    for (int i = 0; i < fractalDescriptor.length (); i++)
        switch (fractalDescriptor.charAt (i))
        {
            case 'F':
                dTurtle[0] = turtle[0] + d * Math.cos (turtle[2]);
                dTurtle[1] = turtle[1] + d * Math.sin (turtle[2]);
                g.drawLine (xPos + (int)turtle[0], yPos + (int)turtle[1], xPos + (int)dTurtle[0], yPos + (int)dTurtle[1]);
                turtle[0] = dTurtle[0];
                turtle[1] = dTurtle[1];
```

```
                break;

                case '+':
                    turtle[2] += t;
                break;

                case '-':
                    turtle[2] -= t;
                break;

                default:
                break;
            }
    }
}
```

The state of the turtle is stored in three `double` variables. Again, every character in the string is analyzed, and the turtle executes a command according to the value of the character. A temporary variable `dTurtle[]` is used to draw the lines.

The discussion above shows that the code to generate graphical output of L-systems is quite easy and follows almost immediately from the definitions. Also more complicated extensions like stochastic L-systems are easy to implement. One problem is which value must be chosen for `d`. Another problem not mentioned before is the speed: After only a few iterations the string will become quite large, and it will take quite some time to draw the resulting picture.

## 5.5 Growth functions of stochastic L-systems

In the theory on L-systems a lot of results are known on **growth functions**, functions that describe the number of symbols in a word in terms of its derivation length. These functions can describe for example how fast a plant grows or how fast a colony of bacteria is expanding. Another question is how much symbols there are of a certain type. Note that in OL-systems the growth functions are independent of the ordering of the symbols in the words. In this section we look at growth functions on stochastic L-systems. Here the growth functions give the **expected** length of a word after a certain derivation length. Without going into too much detail, some theorems are stated about growth functions in stochastic L-systems to give an impression of the work in this field. For more proofs and references to work in this field, see [3].

### 5.5.1 Some definitions

In this subsection we define the growth function and some useful vectors and matrices that will be used later.

**Growth function**: The growth function $f_S$ of a S0L system $S$ is defined by $f_S = \sum_{x \in \Sigma^*} |x| P_\omega^n(x), n \geq 0$. So $f_S(n)$ is the expected length of words that are generated after $n$ derivations.

For the following four definitions, let $S = (\Sigma, R, \rho, \omega)$ be a S0L system and let $a_1, a_2, ..., a_t$ with $a_i \in \Sigma$ be an enumeration of the elements of $\Sigma$.

**Parikh vector**: Let $x \in \Sigma^*$. Let $\#_{a_j}(x)$ denote the number of occurrences of $a_j$ in $x$.

Now the Parikh vector of $x$ is defined as $\pi_x = (\#_{a_1}(x), \#_{a_2}(x), ..., \#_{a_t}(x))$.

**Growth matrix**: The growth matrix $G_S$ of $S$ is a $t$ by $t$ matrix where element $(i, j)$ equals the expected number of occurrences of $a_j$ when $a_i$ is rewritten by a production in $R$: $G_S = (g_{ij})$ with $g_{ij} = \sum_\alpha \rho(a_i \to \alpha)\#_{a_j}(\alpha)$, where the sum is taken over all $\alpha$ such that $a_i \to \alpha \in R$.

**Initial vector**: Let $y_1, y_2, ..., y_m$ be an enumeration of all $y_i \in \Sigma^+$ with $\omega(y_i) > 0$. The initial vector of S is $\pi_\omega = \sum_{i=1}^{m} \omega(y_i)\pi_{y_i}$. Therefore, the initial vector can be seen as a special Parikh vector which denotes the expected number of occurences of $a_j$.

**Final vector**: The final vector $\eta_S$ is a $t$-dimensional column vector with all components equal to 1.

As a simple example to make these definitions more clear, we consider a deterministic L-system discussed earlier: let $\Sigma = \{a, b\}$, $R = \{a \to ab, b \to a\}$ and let the axiom be $b$. This gives the following sequence of derivations: $b \to a \to ab \to aba \to abaab \to abaababa \to ...$

For the growth function we have $f_S(0) = 1$, $f_S(1) = 1$, $f_S(2) = 2$, $f_S(3) = 3$, $f_S(4) = 5$, $f_S(5) = 8$. The Parikh vector of $x = abaababa$ is $\pi_x = (5, 3)$. The growth matrix is equal to $G_S = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}$. The initial vector is $\pi_\omega = (0, 1)$.

### 5.5.2 Properties of Growth Functions

Several theorems describing useful properties of the growth function can be derived and are stated below.

- For all S0L systems $S = (\Sigma, R, \rho, \omega)$ and all $n \in \mathbb{N}$ one can calculate $f_S(n)$ with $f_S(n) = \pi_\omega \cdot G_S^n \cdot n_S$. Note that this theorem is clear for the deterministic example discussed before: $\pi_\omega \cdot G_S^n$ is the Parikh vector of the word derived after $n$ derivations, and multiplying with $n_S$ just sums every coordinate in this vector, giving the total length of the word.
- Let $f$ be a growth function of a S0L system with at most $t$ symbols. Then there exists an algorithm which, given the first $2t$ values of $f$, can compute a lineair recurrence equation for $f$ such that all values of $f$ can be computed from these initial values and the recurrence formula.

Some more definitions on growth functions of S0L systems are given below.

Let $f_S$ be a growth function of a S0L system. The growth rate of $S$ is called **terminating** if there is a $N \in \mathbb{N}$ such that $f_S(n) = 0$ for all $n > N$. The growth rate of $S$ is called **limited** if there is a polynomial $p(n)$ such that $p(n) \geq f_S(n)$ for all $n$. The growth rate of $S$ is called **explosive** if it is not limited. A S0L **schema** is a S0L system without an axiom distribution, $S = (\Sigma, R, \rho)$. A S0L schema has **limited growth rate** if for all axiom distributions $\omega$ over $\Sigma^+$ the S0L system $(\Sigma, R, \rho, \omega)$ has limited growth. Analogue, a S0L schema has **explosive growth rate** if for all axiom distributions $\omega$ over $\Sigma^+$ the S0L system $(\Sigma, R, \rho, \omega)$ has explosive growth. If there exists axiom distributions $\omega_1$ and $\omega_2$ over $\Sigma^+$ such

that $(\Sigma, R, \rho, \omega_1)$ has limited growth rate and $(\Sigma, R, \rho, \omega_2)$ has explosive growth rate, then the S0L schema has **mixed growth rate**.

One can determine the growth rate of an S0L system by considering the roots of the characteristic polynomial of the growth matrix $G_S$: Let $p(z) = \det(G_S - zI)$ so $p(z)$ is the characteristic polynomial of $G_S$. If all roots of $p(z)$ are zero then $S$ has terminating growth rate. If all roots of $p(z)$ are less than or equal to one in absolute value, then $S$ has limited growth rate. If at least one of the roots of $p(z)$ is greater then one then $S$ has explosive growth rate.

### 5.5.3 Growth equivalence

In this subsection we compare the growth functions of different stochastic L-systems.

Let $S = (\Sigma, R, \rho)$ be a S0L schema and let $\omega_1$ and $\omega_2$ be axiom distributions for $S$. Let $f_{S_1}$ and $f_{S_2}$ be growth functions for the S0L systems $S_1 = (\Sigma, R, \rho, \omega_1)$ and $S_2 = (\Sigma, R, \rho, \omega_1)$ respectively. $\omega_1$ is $k$ **growth equivalent** with $\omega_2$, $\omega_1 \sim^k \omega_2$, if $f_{S_1}(n) = f_{S_2}(n)$ for $0 \leq n \leq k$. $\omega_1$ is **growth equivalent** with $\omega_2$, $\omega_1 \sim \omega_2$ if $f_{S_1}(n) = f_{S_2}(n)$ for all $n \geq 0$. The following theorem makes it easier to determine of two axiom distributions are growth equivalent:

- Let $S$ be a S0L schema with $t$ symbols and let $\omega_1$ and $\omega_2$ be two axiom distributions for $S$, then $\omega_1$ and $\omega_2$ are growth equivalent if and only if they are $t$ growth equivalent.

Now we define when two S0L systems are growth equivalent:

Two S0L systems $S_1$ and $S_2$ are **growth equivalent**, $S_1 \sim S_2$ if $f_{S_1}(n) = f_{S_2}(n)$ for all $n \geq 0$.

Let $S_1 = (\Sigma_1, R_1, \rho_1)$ and $S_2 = (\Sigma_2, R_2, \rho_2)$ be S0L schemas. $S_1$ **covers** $S_2$, $S_1 \geq S_2$, if for every axiom distribution $\omega_2$ over $\Sigma_2^+$ there is an axiom distribution $\omega_1$ over $\Sigma_1^+$ such that $S_1 = (\Sigma_1, R_1, \rho_1, \omega_1)$ and $S_2 = (\Sigma_2, R_2, \rho_2, \omega_2)$ are growth equivalent. $S_1$ and $S_2$ are **axiom growth equivalent**, $S_1 \sim^\omega S_2$, if $S_1 \leq S_2$ and $S_2 \leq S_1$. $S_1$ and $S_2$ are **symbol growth equivalent**, $S_1 \sim^s S_2$, if

1. For every $a \in \Sigma_1$ there is an $b \in \Sigma_2$ such that $\overline{S_1} = (\Sigma_1, R_1, \rho_1, \omega_1)$ and $\overline{S_2} = (\Sigma_2, R_2, \rho_2, \omega_2)$ are growth equivalent with $\omega_1(a) = \omega_2(b) = 1$ and $\omega_1$ and $\omega_2$ are zero elsewhere.
2. For every $b \in \Sigma_2$ there is an $a \in \Sigma_1$ such that $\overline{S_1}$ and $\overline{S_2}$ defined as above are growth equivalent.

To complete this section, the following result is stated: there exist algorithms to decide for S0L schemas $S_1 = (\Sigma_1, R_1, \rho_1)$ and $S_2 = (\Sigma_2, R_2, \rho_2)$:

- Let $\omega_1$ and $\omega_2$ be two axiom distributions over $\Sigma_1^+$. Determine if $\omega_1 \sim \omega_2$ for $S_1$.
- Let $S$ and $S'$ be two S0L systems. Determine if $S \sim S'$.
- Determine if $S_1 \sim^s S_2$.
- Determine if $S_1 \geq S_2$.
- Determine if $S_1 \sim^\omega S_2$.

### 5.5.4 Symbol Reduction

Let $S = (\Sigma, R, \rho)$ be a S0L schema. For all $a \in \Sigma$, $\omega_a$ is the axiom distribution with $\omega_a(a) = 1$ and $\omega_a(x) = 0$ for all $x \neq a$. $S_a = (\Sigma, R, \rho, \omega_a)$. Two symbols $a, b \in \Sigma$ are **growth equivalent** for $S$, $a \sim b$, if $S_a \sim S_b$. $S$ is **reduced** if there are no distinct symbols in $\Sigma$ which are growth equivalent for $S$. $S$ is **reduced** to the S0L system $S'$ if $S'$ is reduced and $S' \sim^s S$. The following theorem can now be formulated: For every S0L schema $S$ there exists an algorithm which gives a S0L schema $S'$ such that $S$ is reduced to $S'$. This result says that every S0L system is symbol growth equivalent to another S0L system that is reduced in the way that all symbols have a different growth rate.

# Chapter 6

# Conclusion

In this report an introduction into the study of fractals was given. There are many different ways to look at fractals, one can for example consider them as attractors of Iterated Function Systems, or more generally as a stationary probability distribution of an Iterated Random Function, but also as products of Lindenmayer Systems.

It turns out that there exist algorithms to generate fractals that are quite easy to implement. Surprisingly, although fractals can look extremely complex, it is often possible to describe them in a very simple way. For example, an initial axiom and a set of reproduction rules for a Lindenmayer System can sometimes be written in only one line but can generate trees and plants that resemble nature quite good. But the mathematical background of fractals is quite extensive, understanding why the random iteration algorithm for IFS's works is not easy for example.

In short, fractal theory is an interesting field of study and not only for the artistic value of fractals. One can model plant growth using fractals for example, but there are many other applications. Since fractals appear quite often in nature, even in unexpected places, they will probably get an increasingly important role in mathematics in the future.

# Bibliography

[1] Michael Barnsley, *Fractals Everywhere*, Academic Press, Inc, 1998

[2] Przemyslaw Prusinkiewics and Astrid Lindenmayer, *The Algorithmic Beauty Of Plants*, Springer-Verlag, 1996

[3] Peter Eichhorst, *Growth Functions of Stochastic Lindenmayer Systems*, Information and control 45, p.217-228, 1980

[4] Persi Diaconis and David Freedman, *Iterated Random Functions*, SIAM Review, Vol 41, No.1, p45-76, 1999