Automata, Languages, and Iterated Function Systems

Lecture notes for the SIGGRAPH '91 course:
"Fractal Modeling in 3D Computer Graphics and Imagery"

Przemyslaw Prusinkiewicz and *Mark Hammel*

Department of Computer Science
University of Regina
Regina, Saskatchewan S4S 0A2
Canada

1 The topic

The original method for approximating the attractor $\mathcal{A}$ of an iterated function system $\mathcal{F} = \{F_1, \dots, F_n\}$ consists of selecting a starting point x_0 and applying sequences of transformations from $\mathcal{F}$ to it. These sequences may be chosen at random or in a deterministic fashion, but regardless of the approach used, any sequence applied to $x_0 \in \mathcal{A}$ yields a point y that also belongs to $\mathcal{A}$. In contrast, the generalization of IFS's discussed in these notes introduces a control mechanism that restricts the set of applicable transformations to a (regular) language L over the alphabet $\mathcal{F}$. Thus, in *language-restricted iterated function systems (LRIFS's)*, only some sequences of transformations applied to $x_0 \in \mathcal{A}$ yield points y in $\mathcal{A}$. LRIFS's are more powerful than the ordinary IFS's, in the sense that some attractors of LRIFS's cannot be generated by IFS's. In addition, restrictions on transformation application make it possible to avoid non-invertible affine mappings which are incompatible with the escape-time method for visualizing the attractors. These notes describe the concept of LRIFS, illustrate it using examples, and relate it to other methods for generating fractals, such as the Koch construction and L-systems. The bibliography lists recent papers on the topic.

2 Iterated function systems

Let X be a complete metric space with distance function d. The distance between point $a \in X$ and set $B \subset X$ is defined as:

$$d(a, B) = \inf_{b \in B} d(a, b).$$

The *half-distance* between set $A \subset X$ and set $B \subset X$ is equal to:

$$d'(A, B) = \sup_{a \in A} d(a, B).$$

Note that, in general, $d'(A, B) \neq d'(B, A)$. The *distance between sets* A and B is the greater of the two half-distances:

$$\rho(A, B) = \max\{d'(A, B), d'(B, A)\}.$$

The function $\rho(A, B)$ satisfies the distance axioms in the space $H(X)$ of all closed nonempty bounded subsets of the space X and is called the *Hausdorff metric* on this space.

A function (transformation) $F : X \to X$ is called a *contraction*, if there is a constant $r < 1$ such that

$$d(F_{\cancel{i}}(x), F_{\cancel{i}}(y)) \leq r d(x, y)$$

for all $x, y \in X$.

A transformation $F : X \to X$ is extended to the domain $H(X)$ of subsets of X:

$$F(A) = \{F(x) : x \in A\}.$$

Since the values of F are sets, it is possible to perform set-theoretic operations on them. Let $\mathcal{F} = \{F_1, \ldots, F_N\}$ be a set of functions $F_i : X \to X$, extended to the domain $H(X)$ as described above. The equation

$$\mathcal{S}(A) = \bigcup_{i=1}^{N} F_i(A)$$

defines a function $\mathcal{S} : H(X) \to H(X)$ associated with the set $\mathcal{F}$. Hutchinson [21, page 728] showed that if all functions F_i are contractions in space X with metric d, then $\mathcal{S}$ is a contraction in the space $H(X)$ with the Hausdorff metric ρ.

The space $H(X)$ and the transformation $\mathcal{S}$ satisfy conditions of *Banach's fixed point theorem* [10, page 778], presented here in a narrowed version.

Banach's fixed point theorem. Let M be a complete metric space, and suppose that $T : M \to M$ is a contractive transformation in M. Then for any initial element $x_0 \in M$, the iterative process $x_{n+1} = T(x_n)$, $n = 0, 1, 2, \ldots$, can be continued indefinitely, and the sequence $\{x_n\}$ converges to an element $x \in M$, which is the unique solution of the equation $x = T(x)$.

According to this theorem, there is a unique set $\mathcal{A} \in H(X)$, such that

$$\mathcal{A} = \mathcal{S}(\mathcal{A}) = \bigcup_{i=1}^{N} F_i(\mathcal{A}). \tag{1}$$

The set $\mathcal{F}$ of contractive mappings $F_1, F_2, \ldots, F_N$ is called an *iterated function system*, and the set $\mathcal{A}$ is called the *attractor* of $\mathcal{F}$.

Let $\mathcal{S}^n$ denote the n-fold *power* of the transformation $\mathcal{S}$, defined recursively by the formulae $\mathcal{S}^0(A) = A$ and $\mathcal{S}^n(A) = \mathcal{S}^{n-1}(\mathcal{S}(A))$, where $n = 1, 2, 3, \ldots$. The fixed point theorem states that, for any compact set A, the sequence $\mathcal{S}^n(A)$ converges to the attractor $\mathcal{A}$ in the space $(H(X), \rho)$,

$$\lim_{n\to\infty} \mathcal{S}^n(A) = \mathcal{A}. \tag{2}$$

Let $\mathcal{S}^\star$ denote the *iteration* of the transformation $\mathcal{S}$,

$$\mathcal{S}^\star(A) = \bigcup_{n=0}^{\infty} \mathcal{S}^n(A).$$

We will show that $\mathcal{S}^\star(A) = \mathcal{A}$ for any $A \subseteq \mathcal{A}$. Taking the definition of transformation $\mathcal{S}$ into account, the following inclusions hold:

$$\begin{aligned} A &\subseteq \mathcal{A} \\ \mathcal{S}(A) &\subseteq \mathcal{S}(\mathcal{A}) = \mathcal{A} \\ \mathcal{S}^n(A) &\subseteq \mathcal{A} \text{ for all } n = 0, 1, 2, \ldots \\ \mathcal{S}^\star(A) &\subseteq \mathcal{A}. \end{aligned}$$

On the other hand,

$$\mathcal{A} = \lim_{n\to\infty} \mathcal{S}^n(A) \subseteq \mathcal{S}^\star(A),$$

thus, in conclusion,

$$\mathcal{A} = \mathcal{S}^\star(A).$$

The above equation provides the basic method for constructing the attractor $\mathcal{A}$, by selecting a starting point $x_0 \in \mathcal{A}$ and applying all possible sequences of transformations from $\mathcal{F}$ to it. Of course, in practice it is impossible to consider the infinite number of sequences, and the construction ends after a finite number of steps. Various strategies for choosing subsequent transformations and terminating the approximation of the attractor are discussed in [16, 19].

In the scope of these notes, we are interested in iterated function systems consisting of linear functions F_i in the plane. A legible method for expressing them is important in practice, since it can make IFS's easier to specify and understand. We will express transformations by composing operations of translation, rotation, and scaling. The following notation is observed:

- $t(a,b)$ is a translation by vector (a,b):

$$\begin{aligned} x' &= x+a \\ y' &= x+b \end{aligned}$$

- $r(\alpha)$ is a rotation by angle α with respect to the origin of the coordinate system:

$$\begin{aligned} x' &= x\cos\alpha - y\sin\alpha \\ y' &= x\sin\alpha + y\cos\alpha \end{aligned}$$

- $s(r_x, r_y)$ is a scaling with respect to the origin of the coordinate system:

$$\begin{aligned} x' &= r_x x \\ y' &= r_y x \end{aligned}$$

  If $r = r_x = r_y$, we write $s(r)$ instead of $s(r,r)$.

We compose transformations from left to right. For instance, if $F_1 = t(a,b)$, $F_2 = r(\alpha)$, and $F_3 = s(r)$, then

$$x \circ t(a,b) \circ r(\alpha) \circ s(r) = x \circ F_1 \circ F_2 \circ F_3 = F_3(F_2(F_1(x))).$$

The operator of composition is often omitted without ambiguity,

$$x \circ F_1 \circ F_2 \circ F_3 = xF_1F_2F_3.$$

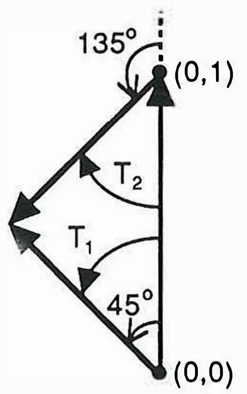
Whenever a transformation cannot be (conveniently) represented in terms of translations, rotations, and scalings, we can use matrix notation.

Example. Figure 1a shows transformations that take vector $\vec{A}$ to vectors $\vec{B}$ and $\vec{C}$. Using the notation introduced above, these transformations can be expressed as:

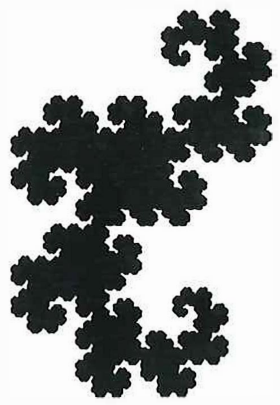
$$\begin{aligned} F_1 &= s(\frac{\sqrt{2}}{2})r(45) \\ F_2 &= s(\frac{\sqrt{2}}{2})r(135)t(0,1) \end{aligned}$$

The attractor of the IFS $\mathcal{F} = \{F_1, F_2\}$ is shown in Figure 1b.

We are now well positioned to characterize the modification of iterated function systems introduced in this paper. The main idea is to limit the set of allowable sequences of transformations to a subset of $\mathcal{S}^\star$. We proceed by identifying sequences of transformations from $\mathcal{F}$ using words over the alphabet of transformation labels V. In consequence, sets of sequences of transformations can be viewed as formal languages over the alphabet V. The necessary notions of the formal language theory are recalled in the next section.



(a)



(b)

Figure 1: Two contractions (a), and the attractor of the resulting IFS (b)

3 Formal languages

An *alphabet* V is a finite nonempty set of *symbols* or *letters*. A *string* or *word* over alphabet V is a finite sequence of zero or more letters of V, whereby the same letter may occur several times. The string consisting of zero letters is called the *empty word* and is denoted by ϵ. The *concatenation* of words $x = a_1a_2 \ldots a_m$ and $y = b_1b_2 \ldots b_n$ is the word formed by extending the sequence of symbols x with the sequence y, thus $xy = a_1a_2 \ldots a_mb_1b_2 \ldots b_n$.

The set of all words over V is denoted by V^*. A *formal language* over an alphabet V is any set L of words over V, hence $L \subset V^*$. Since formal languages are sets, the set-theoretic operations can be performed on languages. In addition, concatenation is extended to languages by the formula:

$$L_1L_2 = \{xy : x \in L_1 \;\&\; y \in L_2\}.$$

The n-th *power* of a language L is defined recursively as $L^0 = \epsilon$, $L^n = LL^{n-1}$ for $n = 1, 2, 3, \ldots$. The *iteration* of language L is the union of all its powers,

$$L^* = \bigcup_{n=0}^{\infty} L^n.$$

The problem of providing finite specifications for infinite languages takes a central place in formal language theory. The two main techniques are grammars and automata. In these notes we focus on *regular languages*, which can be specified using *finite-state automata.*

A **nondeterministic finite-state (Rabin-Scott) automaton** is a quintuplet:

$$\mathcal{M} = < V, S, s_0, E, I >,$$

where:

- V is a finite set of symbols, called the alphabet,
- S is a finite set of states,
- $s_0 \in S$ is a distinguished element of the set S, called the initial state,
- $E \subset S$ is a distinguished subset of S, called the set of final states,
- $I \subset V \times S \times S$ is a state transition relation.

Instead of $(a, s_i, s_k) \in I$, we often write $(a, s_i) \to s_k$.

At the beginning of the operation, the automaton is in initial state s_0. It processes a word $w = a_1 a_2, \ldots, a_n \in V^\star$ letter by letter, arriving at successive states $s_1, s_2, \ldots, s_n$, according to the state transition relation:

$$(a_1, s_0) \to s_1, \quad (a_2, s_1) \to s_2, \ldots \quad (a_n, s_{n-1}) \to s_n.$$

Processing stops when the string of symbols is exhausted. If the sequence of states $s_1, s_2, \ldots, s_n$ can be chosen in such a way that s_n is a final state, $s_n \in E$, then the word w is *accepted* by the automaton $\mathcal{M}$, otherwise it is *rejected.*

Formally, the state transition relation is extended to words as follows:

- $(\epsilon, s_i) \to^\star s_i$ for all $s_i \in S$,
- if there exists such an $s_j \in S$ that $(w, s_i) \to^\star s_j$ and $(a, s_j) \to s_k$, then $(wa, s_i) \to^\star s_k$.

The set

$$L(s_k) = \{w \in V^\star : (s_0, w) \to^\star s_k\}$$

is called the *language associated with the state* s_k. The *language accepted by the automaton* $\mathcal{M}$ is defined as:

$$L(\mathcal{M}) = \bigcup_{s_k \in E} L(s_k) = \{w \in V^\star : (\exists s_k \in E)\,(s_0, w) \to^\star s_k\}.$$

Let $I(s_k) \subset I$ denote the set of all transitions to the state s_k. For any state s_k other than s_0, the following equality holds:

$$L(s_k) = \bigcup_{(a, s_i, s_k) \in I(s_k)} L(s_i)a.$$

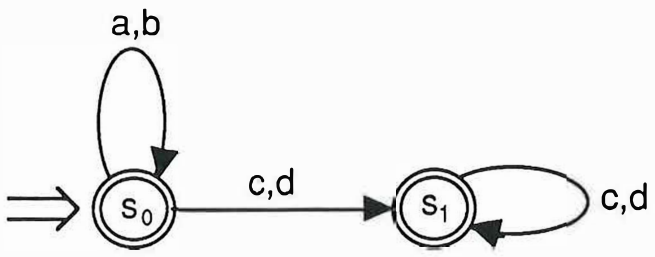


Figure 2: A sample finite automaton

The empty word must be added in the case of the initial state:

$$L(s_0) = \bigcup_{(a,s_i,s_0)\in I(s_0)} L(s_i)a \cup \epsilon.$$

Finite state automata are commonly represented as directed graphs, with the nodes representing states and arcs representing transitions. The initial state is indicated by a short arrow. If $E \neq S$, the final states are distinguished by double circles. A sample finite automaton is shown in Figure 2. The languages associated with the states s_0 and s_1 satisfy the following equations:

$$\begin{aligned} L(s_0) &= L(s_0)(a \cup b) \cup \epsilon \\ L(s_1) &= L(s_0)(c \cup d) \cup L(s_1)(c \cup d) \\ L(\mathcal{M}) &= L(s_0) \cup L(s_1) \end{aligned}$$

These equations can be solved, yielding an algebraic description of the language $L(\mathcal{M})$ in the form of a *regular expression*. The algebra of regular expressions furnishes the set of allowable operations on expressions. Intuitively, the solution proceeds by substituting expressions for variables and performing set-theoretic operations. These operations may lead to equations of the form $X = XA \cup B$, which have solutions $X = BA^\star$. For the example under consideration, we obtain:

$$\begin{aligned} L(s_0) &= (a \cup b)^\star \\ L(s_1) &= (a \cup b)^\star(c \cup d) \cup L(s_1)(c \cup d) = (a \cup b)^\star(c \cup d)(c \cup d)^\star \\ L(\mathcal{M}) &= (a \cup b)^\star \cup (a \cup b)^\star(c \cup d)(c \cup d)^\star = (a \cup b)^\star(c \cup d)^\star \end{aligned}$$

We will now apply these notions to iterated function systems.

4 Language-restricted iterated function systems

A *language-restricted* iterated function system (LRIFS) is a quintuplet $\mathcal{F}_L = \langle X, \mathcal{F}, V, h, L\rangle$, where:

- X is the underlying metric space,

- $\mathcal{F}$ is an iterated function system in space X,
- V is an alphabet,
- $h : V \to \mathcal{F}$ takes letters of the alphabet V to mappings of the IFS $\mathcal{F}$,
- $L \subset V^*$ is a language over the alphabet V.

The domain of the mapping h is extended from the set of letters V to the set of words V^* using the equation:

$$h(a_1 a_2 \ldots a_n) = h(a_1) \circ h(a_2) \circ \ldots \circ h(a_n).$$

According to this definition, the extended function h is a homomorphism from the monoid generated by the alphabet V with the operation of concatenation to the monoid generated by the iterated function system $\mathcal{F}$ with composition. The final extension of h from the domain of words to the domain of languages over V is given by the equation:

$$h(L) = \bigcup_{w \in L} h(w).$$

Thus, $H(L)$ is a function defined in the domain $H(X)$ of subsets of the space X.

If the language L consists of all words over alphabet V, then $h(L) = h(V^*)$ is the set of all sequences of transformations in the IFS $\mathcal{F}$, and

$$h(L) = (F_1 \cup F_2 \cup \ldots \cup F_n)^* = \mathcal{S}^*.$$

In contrast, if L is a subset of V^*, then only some composite transformation from $\mathcal{S}^*$ belong to $h(L)$, hence $h(L) \subset \mathcal{S}^*$. If x_0 belongs to the attractor $\mathcal{A}$ of the IFS $\mathcal{F}$, and $\mathcal{A}_L(x_0)$ denotes the image of the set $\{x_0\}$ with respect to the transformation $h(L)$, the following inclusion holds:

$$\mathcal{A}_L(x_0) = x_0 \circ h(L) \subset x_0 \circ h(V^*) = \mathcal{S}^*(x_0) = \mathcal{A}.$$

Thus, the set $\mathcal{A}_L(x_0)$ generated by the LRIFS $\mathcal{F}_L$ with the starting point $x_0 \in \mathcal{A}$ is a subset of the attractor $\mathcal{A}$.

Example. Consider an LRIFS $\mathcal{F}_L = \langle X, \mathcal{F}, V, h, L \rangle$, where:

- the space X is the plane,
- the IFS $\mathcal{F}$ consists of four transformations:

$$\begin{aligned} F_1 &= s(0.5) \\ F_2 &= s(0.5)t(0, 0.5) \\ F_3 &= s(0.5)r(45)t(0, 1) \\ F_4 &= s(0.5)r(-45)t(0, 1), \end{aligned}$$

- the alphabet V consists of four letters a, b, c, d,

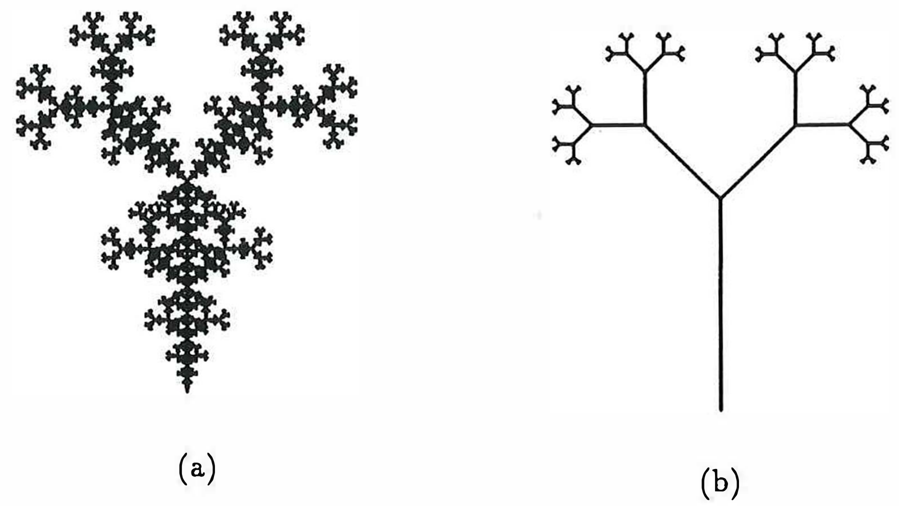


Figure 3: Attractor $\mathcal{A}$ and its subset $\mathcal{A}_L(x_0)$

- the homomorphism h is defined by:

$$h(a) = F_1, \quad h(b) = F_2, \quad h(c) = F_3, \quad h(d) = F_4,$$

- the language L is defined by the regular expression

$$L = (a \cup b)^*(c \cup d)^*$$

  which corresponds to the automaton from Figure 2.

Figure 3 compares the attractor $\mathcal{A}$ of the IFS $\mathcal{F}$ with the set $\mathcal{A}_L(x_0)$ generated by the LRIFS $\mathcal{F}_L$ using the starting point $x_0 = (0,0)$. Clearly, the branching structure of Figure (b) is a subset of the original attractor (a).

Example. Figure 4 shows six sets $\mathcal{A}_L(x_0)$ generated by LRIFS's with the same set of transformations $\mathcal{F}$, but with different languages L. It is assumed that:

- X is the plane,
- $\mathcal{F}$ consists of four transformations:

$$\begin{aligned} T_1 &= s(0.5)t(0.0, 0.5) \\ T_2 &= s(0.5)t(0.5, 0.5) \\ T_3 &= s(0.5) \\ T_4 &= s(0.5)t(0.5, 0.0) \end{aligned}$$

- $V = \{T_1, T_2, T_3, T_4\}$,
- $h(T_i) = T_i$ for $i = 1, 2, 3, 4$,

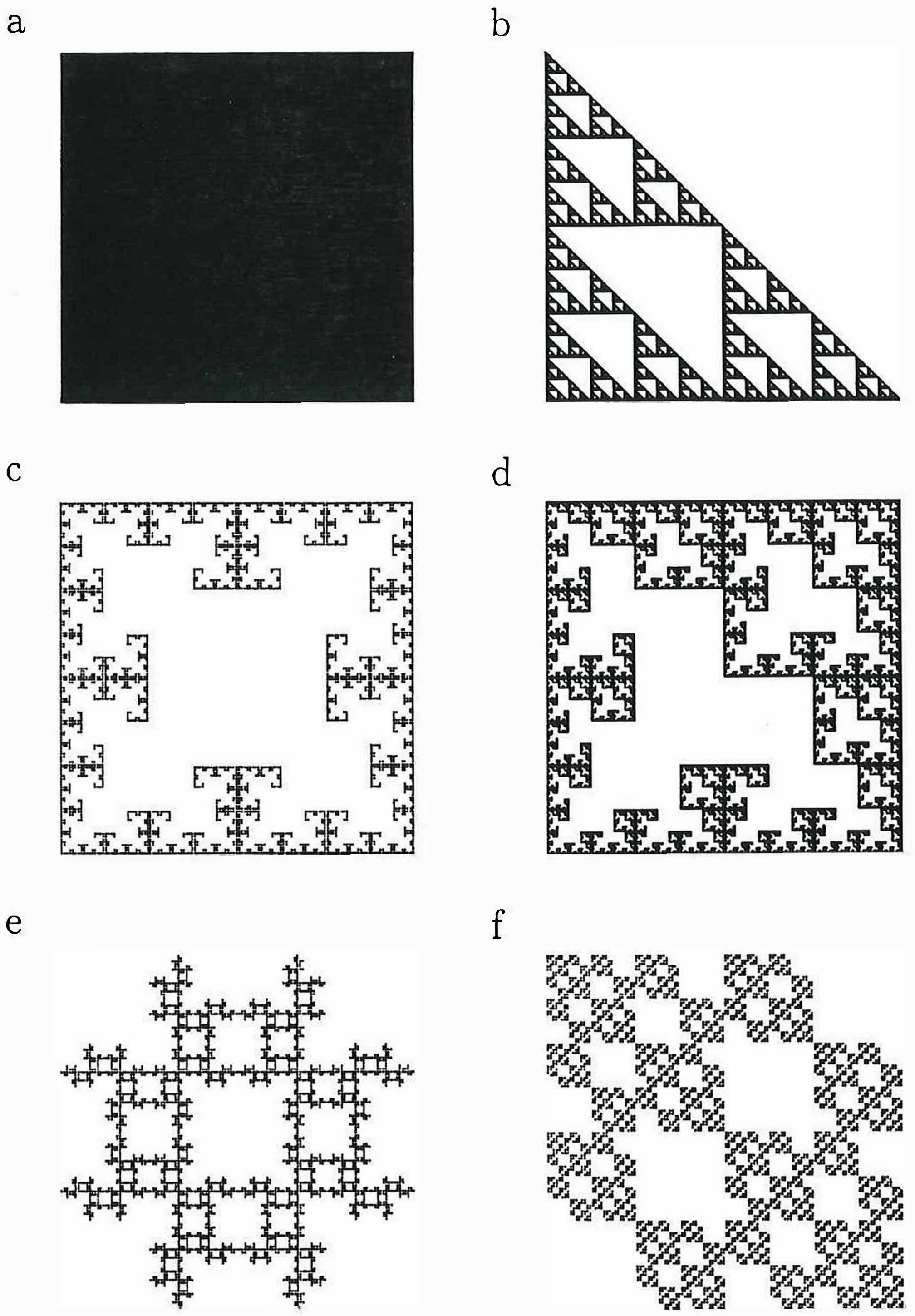


Figure 4: Sample sets $\mathcal{A}_L(x_0)$, generated using the same set of transformations $\mathcal{F}$. Figure (b) is the Sierpiński gasket [22], figures (c) and (d) are from [8], figure (e) is from [27].

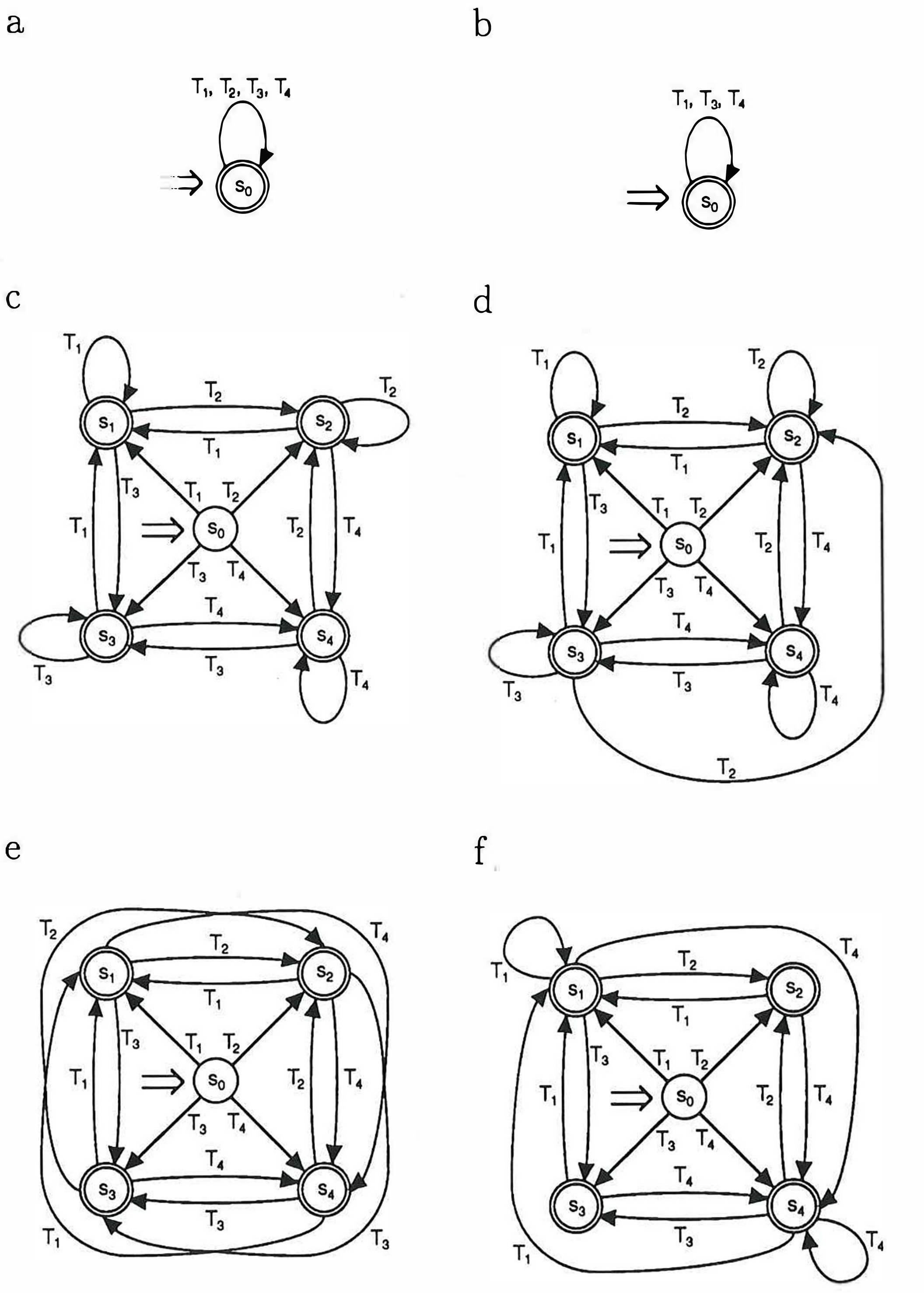


Figure 5: The automata used to create images in the previous figure

The languages L are defined using finite automata shown in Figure 5.

Examination of this example leads to the following observations:

- In contrast to the "ordinary" IFS's where the variety of generated figures is obtained by changing the set of transformations, LRIFS's make it possible to generate various figures using the same transformations, by changing the language L.
- The underlying IFS recursively subdivides a square into four smaller squares. Consequently, the languages L_1 to L_6 are related to the quad-tree representation of the figures shown. This relation is explored in [8, 9].

5 The recursive structure of the generated sets

Consider an LRIFS $\mathcal{F}_L = \langle X, \mathcal{F}, V, h, L\rangle$, where L is a regular language, and let $\mathcal{M} = \langle V, S, E, s_0, I\rangle$ be the finite automaton that accepts L. According to Section 3,

$$\begin{aligned} L(s_0) &= \bigcup_{(a,s_i,s_0)\in I(s_0)} L(s_i)a \cup \epsilon, \\ L(s_k) &= \bigcup_{(a,s_i,s_k)\in I(s_k)} L(s_i)a \text{ for } s_k \neq s_0. \end{aligned}$$

After applying homomorphism h to both sides of these equations, for an arbitrary $x_0 \in X$ we obtain:

$$\begin{aligned} x_0h(L(s_0)) &= \bigcup_{(a,s_i,s_0)\in I(s_0)} x_0h(L(s_i))h(a) \cup \{x_0\}, \\ x_0h(L(s_k)) &= \bigcup_{(a,s_i,s_k)\in I(s_k)} x_0h(L(s_i))h(a) \text{ for } s_k \neq s_0. \end{aligned}$$

Let $\mathcal{A}_k = x_0h(L(s_k))$ denote the image of the set $\{x_0\}$ with respect to the transformation $h(L(s_k))$, and suppose that $F_a = h(a)$ is the transformation of $\mathcal{F}$, represented by symbol a. We obtain:

$$\mathcal{A}_0 = \bigcup_{(a,s_i,s_0)\in I(s_0)} \mathcal{A}_i \circ F_a \cup \{x_0\} = \bigcup_{(a,s_i,s_0)\in I(s_0)} F_a(\mathcal{A}_i) \cup \{x_0\}, \tag{3}$$

$$\mathcal{A}_k = \bigcup_{(a,s_i,s_k)\in I(s_k)} \mathcal{A}_i \circ F_a = \bigcup_{(a,s_i,s_k)\in I(s_k)} F_a(\mathcal{A}_i) \text{ for } s_k \neq s_0. \tag{4}$$

This system of equations describes the recursive structure of the set $\mathcal{A}_L(x_0)$, generated by the LRIFS $\mathcal{F}_L$. It can be compared with the equation

$$\mathcal{A} = \bigcup_{F_i\in\mathcal{F}} F_i(\mathcal{A})$$

that characterizes the attractor $\mathcal{A}$ of an IFS $\mathcal{F}$. The attractor of an IFS is the union of its images with respect to all transformations in $\mathcal{F}$. In contrast, an LRIFS divides the set $\mathcal{A}$ into a finite number of subsets $\mathcal{A}_k$; each of them is the union of the images of selected subsets $\mathcal{A}_i$ sets with respect to selected transformations F_a.

Example. Figure 5 shows a structure that exhibits an alternating branching pattern along the main axis and an opposite branching pattern along the first-order lateral axes. Figure 5 partitions this structure into four subsets $\mathcal{A}_i$, $i = 1, 2, 3, 4$, and reveals their relationships. They are captured by the following equations:

$$\begin{aligned}
\mathcal{A}_0 &= \mathcal{A}_0(T_1 \cup T_2 \cup T_3 \cup T_4) \cup \{x_0\} \\
\mathcal{A}_1 &= \mathcal{A}_0(T_5 \cup T_6 \cup T_7) \cup \mathcal{A}_1 T_8 \cup \mathcal{A}_3(T_6 \cup T_7) \\
\mathcal{A}_2 &= \mathcal{A}_0(T_9 \cup T_{10} \cup T_{11}) \cup \mathcal{A}_2 T_{12} \cup \mathcal{A}_3(T_{10} \cup T_{11}) \\
\mathcal{A}_3 &= \mathcal{A}_0 T_{13} \cup \mathcal{A}_1 T_{13} \cup \mathcal{A}_2 T_{13} \cup \mathcal{A}_3 T_{13} \\
\mathcal{A} &= \mathcal{A}_0 \cup \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3.
\end{aligned}$$

The corresponding automaton is shown in Figure 5. The transformations used to generate structure Figure 5 are specified below:

$$\begin{aligned}
T_1 &= s(0.5)t(-0.002, 0) \\
T_2 &= s(0.5)t(0.002, 0) \\
T_3 &= s(0.5)t(-0.002, 0.13) \\
T_4 &= s(0.5)t(0.002, 0.13) \\
T_5 &= s(0.42)r(45) \\
T_6 &= s(0.2)r(90)t(-0.05, 0.05) \\
T_7 &= s(0.2)t(-0.05, 0.05) \\
T_8 &= t(0.3, -0.3)s(0.74)t(-0.3, 0.3) \\
T_9 &= s(0.37)r(-45)t(0, 0.14) \\
T_{10} &= s(0.172)r(-90)t(0.05, 0.19) \\
T_{11} &= s(0.172)t(0.05, 0.19) \\
T_{12} &= t(-0.265, -0.405)s(0.74)t(0.265, 0.405) \\
T_{13} &= t(0, -1)s(0.74)t(0, 1)
\end{aligned}$$

6 Attractors of LRIFS's

According to Section 2, the attractor $\mathcal{A}$ of an IFS $\mathcal{F}$ does not depend on the selection of the initial point x_0. Furthermore, the set $\mathcal{S}^*(x_0)$ is equal to the attractor $\mathcal{A}$, as long as $x_0 \in \mathcal{A}$. In contrast, the equations 3, which relate subsets $\mathcal{A}_k$ associated with an LRIFS $\mathcal{F}_L$, include a term representing the initial point. In general, its choice affects the set $\mathcal{A}_L(x_0)$.

Example. Consider an LRIFS $\mathcal{F}_L = \langle X, \mathcal{F}, V, h, L \rangle$, where X is the Euclidean plane and the alphabet V is equal to $\{a, b\}$. The transformations F_a and F_b are given below:

$$\begin{aligned}
F_a &= s(0.5) \\
F_b &= s(0.5)t(0.5, 0)
\end{aligned}$$

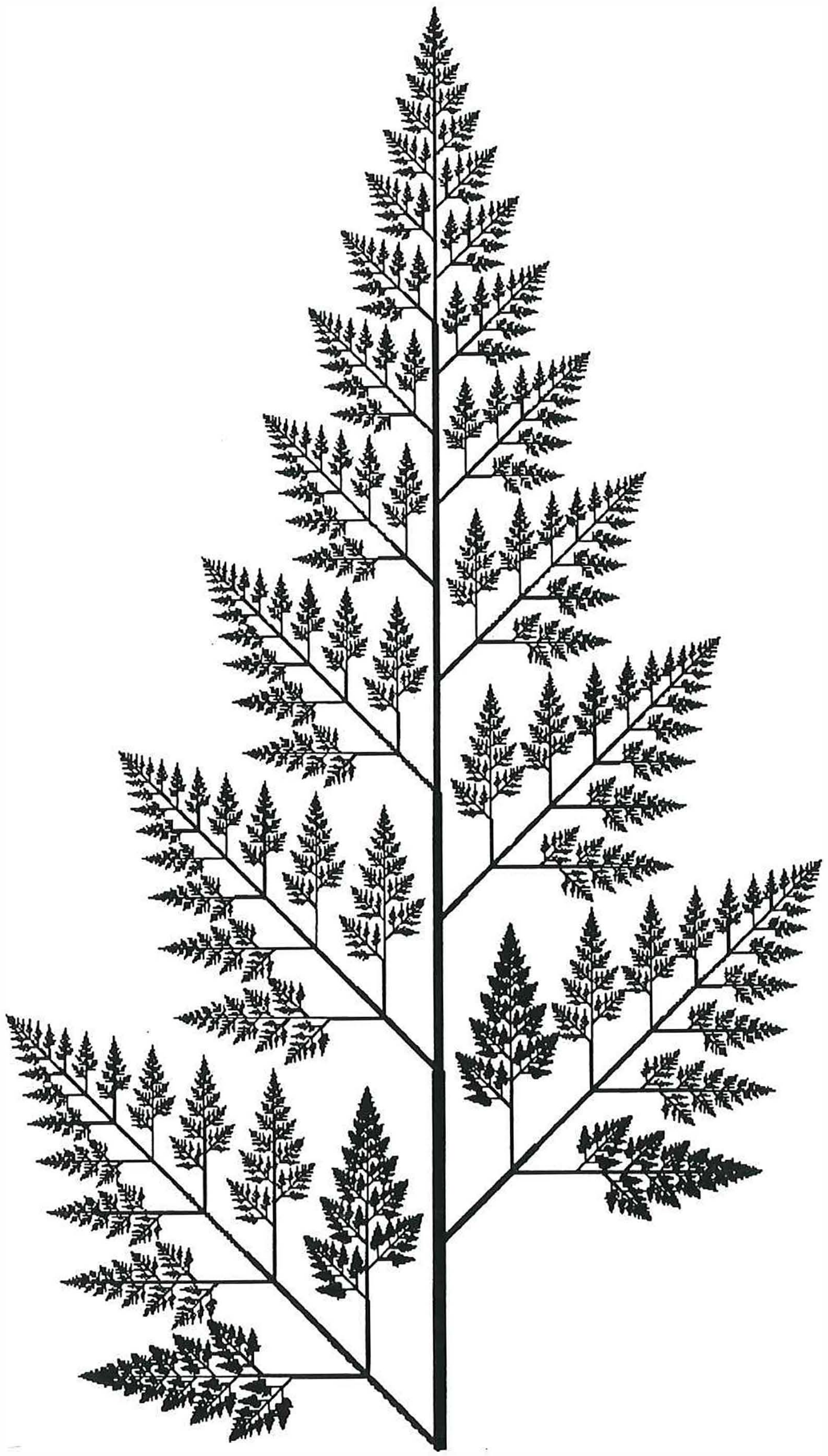


Figure 6: A structure $\mathcal{A}$ with an alternating–opposite branching pattern

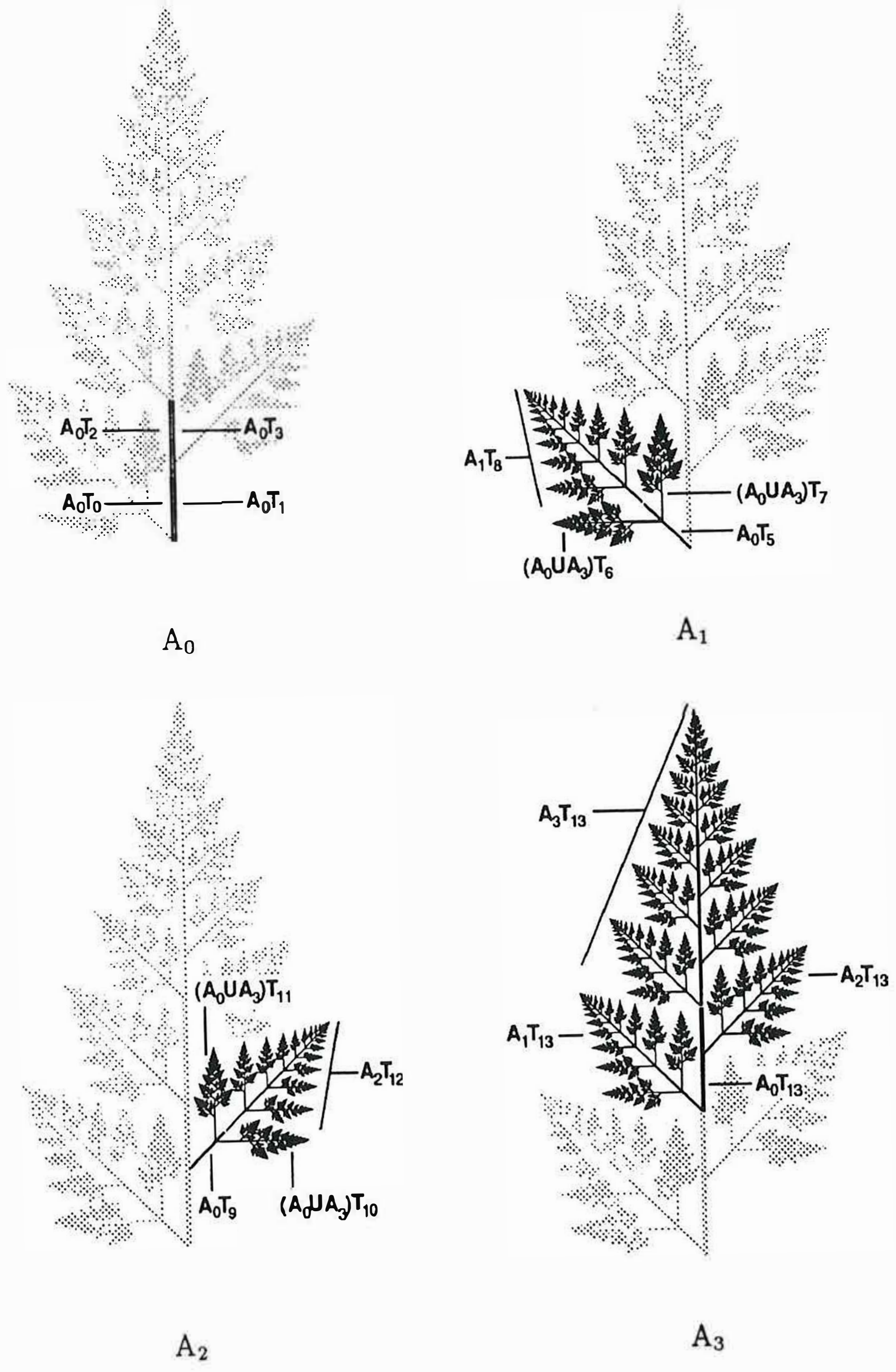


Figure 7: The recursive structure of the alternating–opposite branching pattern

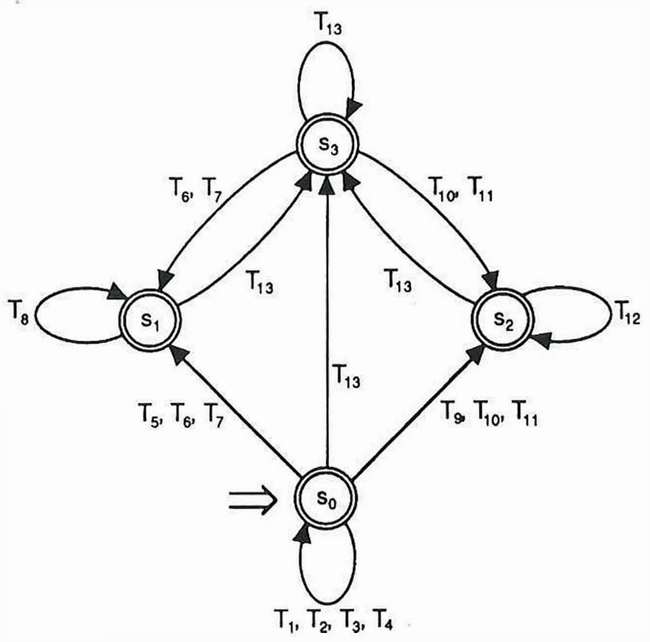


Figure 8: The automaton $\mathcal{M}$ capturing the recursive structure of the alternating–opposite branching pattern

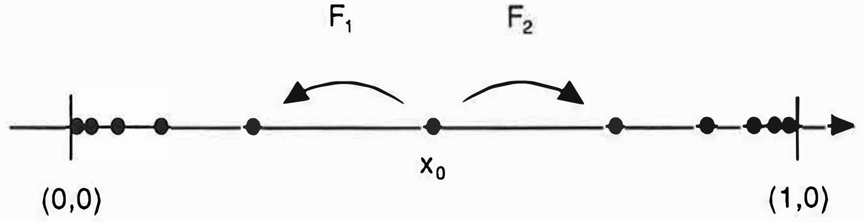


Figure 9: Example of a set generatd by an LRIFS. No smallest closed set exists in this case.

The language L consists of words of the form a^i or b^i:

$$L = a^{\star} \cup b^{\star}.$$

The sets generated by this LRIFS depend on the choice of the starting point x_0 (Figure 9).

Below we give a sufficient condition for the existence of the smallest compact set generated by an LRIFS. This is similar to the case of "ordinary" IFS's, where the attractor $\mathcal{A}$ can be defined as the smallest set satisfying the equation

$$\mathcal{A} = \bigcup_{F_i \in \mathcal{F}} F_i(\mathcal{A}).$$

Theorem. Consider an LRIFS $\mathcal{F}_L = \langle X, \mathcal{F}, V, h, L \rangle$, and assume that L has the *prefix property* (there exists a nonempty word $v \in V^*$ such that $vL \subset L$). Denote by x_0 the fixed point of the transformation $h(v)$, and let $\mathcal{A} = x_0 h(L)$. Then for any $x \in X$, the inclusion $\mathcal{A} \subset xF(L)$ holds.

Proof. From the inclusion $vL \subset L$ it follows that $v^n L \subset L$ for any $n = 0, 1, 2, \ldots$. Thus, for any x,

$$x_0 F(L) = \lim_{n\to\infty} xF(v^n)F(L) = \lim_{n\to\infty} xF(v^n L) \subset xF(L).$$

By analogy with "ordinary" IFS's, we call the set $\mathcal{A}$ the *attractor of the LRIFS* $\mathcal{F}_L$.

In the case of regular languages L, the prefix property means that the graph of the automaton $\mathcal{M}$ that defines L has at least one cycle including the initial state. If v is a word such that $(s_0, v) \rightarrow^* s_0$, the fixed point of the transformation $h(v)$ can be used as the starting point x_0. For example, in the case of automaton $\mathcal{M}$ shown in Figure 2, v can be chosen arbitrarily from the set $(a \cup b)^*$, thus the initial point $x_0 = h(v)$ for generating the branching structure shown in Figure 3b can be chosen anywhere on the main (vertical) stem.

A popular method for generating attractors of IFS's is Barnsley's *chaos game* [3]. While the fixed point theorem suggests the iterative construction of an attractor by exploring the tree of composite transformations and applying them to the initial point, the chaos game makes it possible to pursue only one path in this tree, provided that the transformation applied in each step is chosen at random. Using the terminology of Markov processes instead of automata, Barnsley, Elton, and Hardin showed that the chaos game can also be used to generate the attractors of LRIFS's, if the language L is defined by a *strongly connected* automaton $\mathcal{M}$ [5]. (An automaton is strongly connected if, for any pair of states $s_i, s_k \in S$, there exists a word $v \in V^*$ such that $(s_i, v) \rightarrow^* s_k$.) Given a state s_i, the transformation is chosen at random from the set of transformations $h(a)$ such that $(a, s_i) \rightarrow s_k$ for some state s_k. For example, Figure 6 shows two attractors of LRIFS's generated using the chaos game method. The images correspond to Figure 4, c and e. The automata defining languages L are similar to those shown in Figure 5, c and e, except that states s_0 with related transitions have been removed, and states s_1 are considered initial.

7 Iterated function systems, Koch construction, and L-systems

According to Section 5, construction of an LRIFS proceeds by partitioning the attractor $\mathcal{A}$ into subsets $\mathcal{A}_k$, and finding transformations T_i that relate them. Another possibility occurs if a description of the set $\mathcal{A}$ is already known, for example, in terms of a Koch construction or an L-system. A method for obtaining an iterated function system equivalent to a given Koch construction is described formally in [25]. The

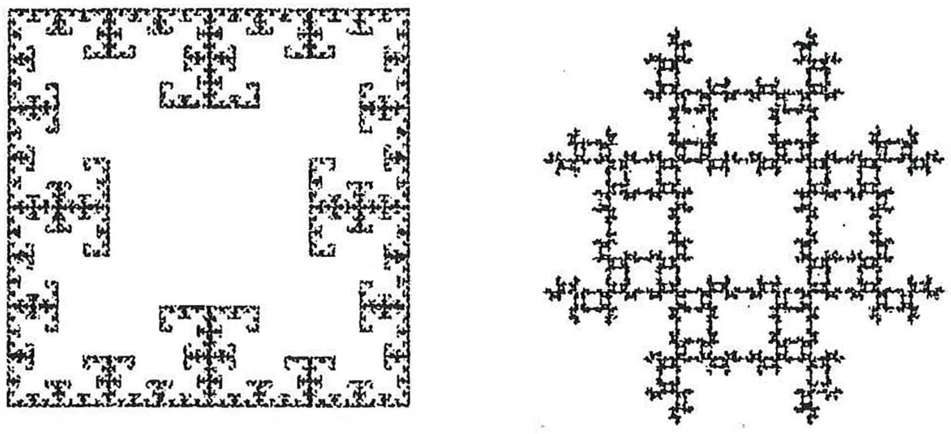
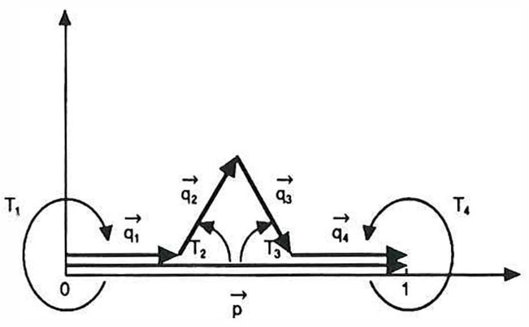


Figure 10: Sample attractors of LRIFS's, generated using the chaos game method.

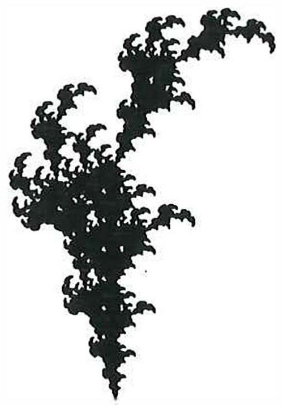


$$\begin{aligned}
\mathtt{T}_1 &= \mathtt{s(1/3)}\\
\mathtt{T}_2 &= \mathtt{s(1/3)r(60)t(1/3,0)}\\
\mathtt{T}_3 &= \mathtt{s(1/3)r(-60)t(1/2,}\sqrt{3}\mathtt{/6)}\\
\mathtt{T}_4 &= \mathtt{s(1/3)t(2/3,0)}
\end{aligned}$$

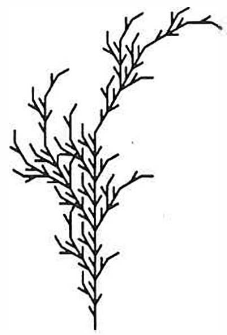
Figure 11: Koch construction of the snowflake curve, and the associated iterated function system

main idea is as follows. The Koch construction is viewed as a rewriting system that takes the predecessor vector $\vec{p}$ into a successor polygon $\vec{q}_1\vec{q}_2\ldots\vec{q}_n$. If F_i denotes the similarity that takes $\vec{p}$ to $\vec{q}_i$, then the attractor $\mathcal{A}$ of the iterated function system $\mathcal{F} = \{F_1, F_2, \ldots, F_n\}$ is the same as the curve resulting from the Koch construction. For example, Figure 11 shows the production $\vec{p} \rightarrow \vec{q}_1\vec{q}_2\vec{q}_3\vec{q}_4$ and the resulting IFS for the snowflake curve.

Figures 12, 13 and 14 compare branching structures generated using Koch construction with those generated using iterated function systems. Since the limit curves are the same, structures obtained in a small number of steps are shown as well. Koch constructions are specified and implemented using the formalism of L-systems with turtle interpretation [23, 24].

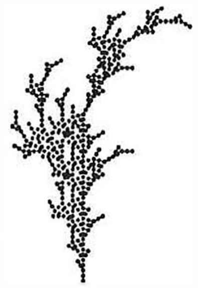


The limit structure



$$\begin{array}{rl} \omega: & \mathtt{F} \\ p_1: & \mathtt{F} \rightarrow \mathtt{F[+F]F[-F]} \end{array}$$

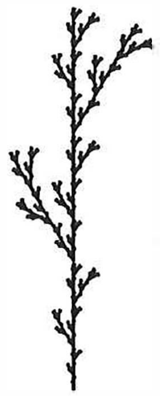
L-system



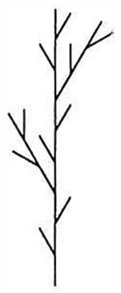
$$\begin{aligned} \mathtt{T}_1 &= \mathtt{s(0.5)r(30)t(0.0, 0.5)} \\ \mathtt{T}_2 &= \mathtt{s(0.5)r(-30)t(0.0, 1.0)} \\ \mathtt{T}_3 &= \mathtt{s(0.5)} \\ \mathtt{T}_4 &= \mathtt{s(0.5)t(0.0, 0.5)} \end{aligned}$$

IFS

Figure 12: A comparison of structures generated using an L-system and the equivalent IFS

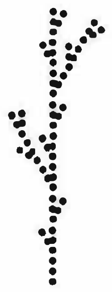


The limit structure



$$
\begin{aligned}
\omega &: \mathtt{F} \\
p_1 &: \mathtt{F} \rightarrow \mathtt{F[+F]F[-F]F}
\end{aligned}
$$

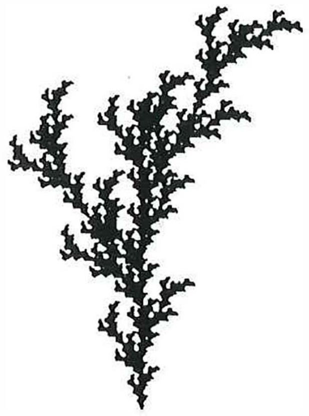
L-system

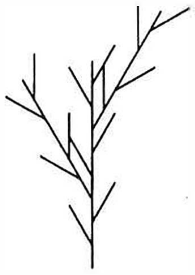


$$
\begin{aligned}
\mathtt{T}_1 &= \mathtt{s(0.33)r(30)t(0.0, 0.33)} \\
\mathtt{T}_2 &= \mathtt{s(0.33)r(-30)t(0.0, 0.67)} \\
\mathtt{T}_3 &= \mathtt{s(0.33)} \\
\mathtt{T}_4 &= \mathtt{s(0.33)t(0.0, 0.33)} \\
\mathtt{T}_5 &= \mathtt{s(0.33)t(0.0, 0.67)}
\end{aligned}
$$

IFS

Figure 13: A comparison of structures generated using an L-system and the equivalent IFS

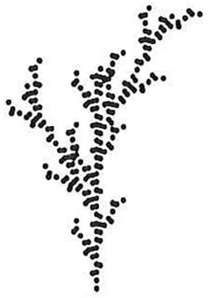

The limit structure



ω : F

p_1 : F $\rightarrow$ F[+FF]F[−FF]F

L-system



$$\begin{aligned}
T_1 &= s(0.33)r(30)t(0.0, 0.33)\\
T_2 &= s(0.33)r(30)t(-0.17, 0.62)\\
T_3 &= s(0.33)r(-30)t(0.0, 0.67)\\
T_4 &= s(0.33)r(-30)t(0.16, 0.62)\\
T_5 &= s(0.33)\\
T_6 &= s(0.33)t(0.0, 0.33)\\
T_7 &= s(0.33)t(0.0, 0.6
\end{aligned}$$

IFS

Figure 14: A comparison of structures generated using an L-system and the equivalent IFS

L-systems have capabilities extending beyond Koch constructions. For example, Figure 15 shows a structure obtained in a simple simulation of plant growth. In the L-system, letters X describe apices that yield new branches according to production p_1; production p_2 describes the exponential elongation of branch segments. Each letter X on the right side of production p_1 yields a branch similar to the entire "plant", but twice as small, since the branch starts developing one step after the plant. Thus, the production reveals self-similarity in the generated structure, which is the basis of the corresponding LRIFS. Figures 16 and 17 provide additional examples. Figure 18 uses parametric L-systems to capture situations where the delay $D+1$ in the development of lateral branches may be greater than one, and segments may elongate by a noninteger factor R in each derivation step. A formal analysis of the relationship between L-systems, applied to model simple growing structures, and iterated function systems is given in [24, Chapter 8]. Curves obtained by Koch constructions and models of growing structures are the only fractals for which the equivalence between L-systems with turtle interpretation and (language-restricted) IFS's has been shown. Future work should generalize these results.

8 A guide to the references

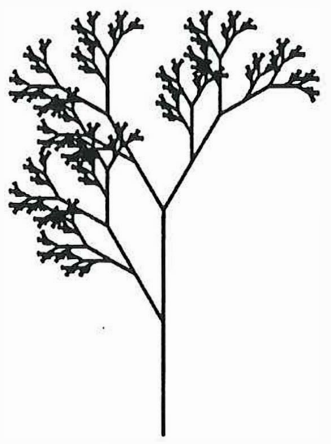
Iterated Function Systems (IFS's) are among the basic methods for generating fractals. The term itself was introduced by Barnsley and Demko [4], but the essential concept is usually attributed to Hutchinson [21]. Vrscay [27] traces the idea further back to Williams [28], who studied fixed points of finite compositions of contractive maps. A detailed introduction to IFS's is presented in [3].

Barnsley, Elton and Hardin [5] introduce recurrent iterated function systems (RIFS) using Markov chains to control the order in which (affine) transformations are applied. The relation to finite automata is not mentioned directly, but it is not necessary to use a stochastic algorithm (the chaos game) to traverse the graph representing the Markov chain; various deterministic algorithms can also be used. This paper is well known and often cited, but concepts similar to RIFS's were around before.

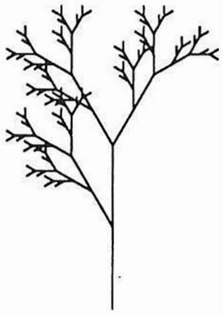
Berger [7] wrote a popular article in the same vein as Barnsley's paper. Apparently, Womack's M.S. thesis [29] is an independent (re)discovery of the concept.

In a paper which focuses on fractal generation by substitutions, Dekking provides a one-paragraph survey of the global construction of fractals, exemplified by recurrent IFS's [15], and credits Bedford [6] with the first results relating these two methods. Bedford's paper is rather difficult to follow. In contrast, Gilbert [18] and Bandt [1, 2] provide very legible presentations of the extensions of IFS's based on equations similar to those used in these notes:

$$\mathcal{A}_k = \bigcup_{(a,s_i,s_k)\in I(s_k)} F_a(\mathcal{A}_i).$$

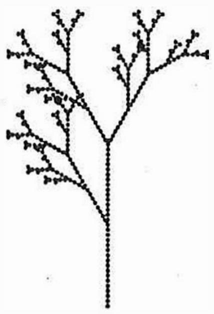


The limit structure



$$\begin{aligned} \omega &: \mathtt{X} \\ p_1 &: \mathtt{X} \rightarrow \mathtt{F[+X]F[-X]+X} \\ p_2 &: \mathtt{F} \rightarrow \mathtt{FF} \end{aligned}$$

L-system



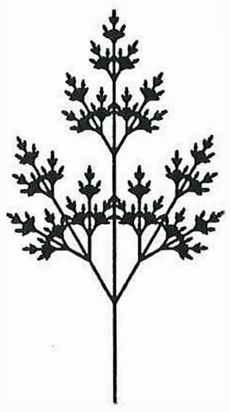
$$(\mathtt{Q}_1 \cup \mathtt{Q}_2)^*(\mathtt{T}_1 \cup \mathtt{T}_2 \cup \mathtt{T}_3)^*$$

$$\begin{aligned} \mathtt{Q}_1 &= \mathtt{s(0.5)} \\ \mathtt{Q}_2 &= \mathtt{s(0.5)t(0.0, 0.5)} \end{aligned}$$

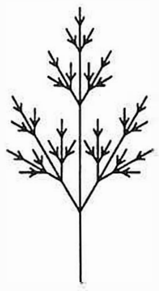
$$\begin{aligned} \mathtt{T}_1 &= \mathtt{s(0.5)r(30)t(0.0, 0.5)} \\ \mathtt{T}_2 &= \mathtt{s(0.5)r(-30)t(0.0, 0.5)} \\ \mathtt{T}_3 &= \mathtt{s(0.5)r(-30)t(0.0, 1.0)} \end{aligned}$$

LRIFS

Figure 15: A comparison of structures generated using an L-system and the equivalent LRIFS

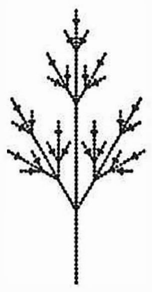


The limit structure



$$\begin{aligned} \omega &: \texttt{X} \\ p_1 &: \texttt{X} \rightarrow \texttt{F[+X][-X]FX} \\ p_2 &: \texttt{F} \rightarrow \texttt{FF} \end{aligned}$$

L-system



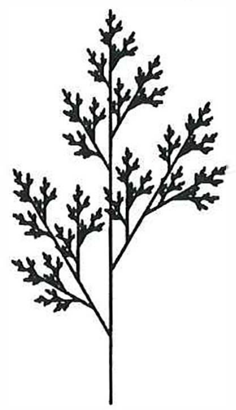
$$(\texttt{Q}_1 \cup \texttt{Q}_2)^*(\texttt{T}_1 \cup \texttt{T}_2 \cup \texttt{T}_3)^*$$

$$\begin{aligned} \texttt{Q}_1 &= \texttt{s(0.5)} \\ \texttt{Q}_2 &= \texttt{s(0.5)t(0.0,0.5)} \end{aligned}$$

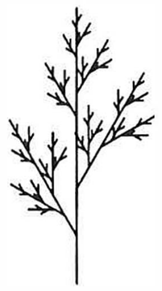
$$\begin{aligned} \texttt{T}_1 &= \texttt{s(0.5)r(30)t(0.0,0.5)} \\ \texttt{T}_2 &= \texttt{s(0.5)r(-30)t(0.0,0.5)} \\ \texttt{T}_3 &= \texttt{s(0.5)t(0.0,1.0)} \end{aligned}$$

LRIFS

Figure 16: A comparison of structures generated using an L-system and the equivalent LRIFS

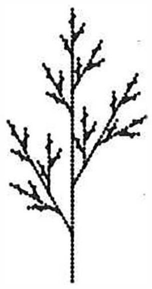


The limit structure



$$\begin{aligned} \omega &: \mathtt{X} \\ p_1 &: \mathtt{X} \rightarrow \mathtt{F[+X]F[-X]FX} \\ p_2 &: \mathtt{F} \rightarrow \mathtt{FF} \end{aligned}$$

L-system



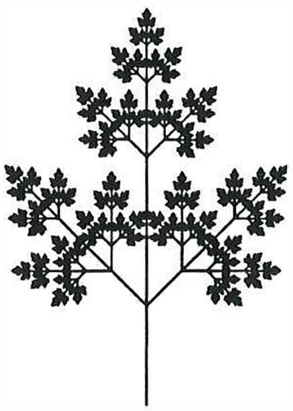
$$(\mathtt{Q}_1 \cup \mathtt{Q}_2)^*(\mathtt{T}_1 \cup \mathtt{T}_2 \cup \mathtt{T}_3)^*$$

$$\begin{aligned} \mathtt{Q}_1 &= \mathtt{s(0.5)} \\ \mathtt{Q}_2 &= \mathtt{s(0.5)t(0.0, 0.5)} \end{aligned}$$

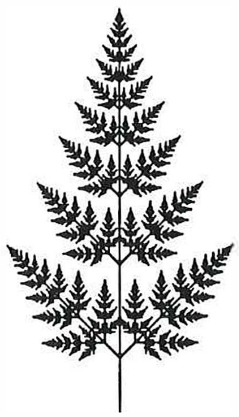
$$\begin{aligned} \mathtt{T}_1 &= \mathtt{s(0.5)r(30)t(0.0, 0.33)} \\ \mathtt{T}_2 &= \mathtt{s(0.5)r(-30)t(0.0, 0.67)} \\ \mathtt{T}_3 &= \mathtt{s(0.5)t(0.0, 1.0)} \end{aligned}$$

LRIFS

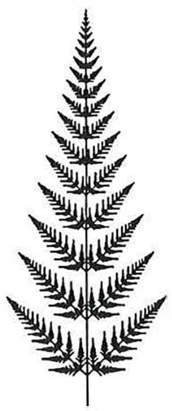
Figure 17: A comparison of structures generated using an L-system and the equivalent LRIFS



D = 0, R = 2



D = 2, R = 1.36



D = 7, R = 1.17

```
#define D 0              /* apical delay */
#define R 2.0            /* internode elongation rate */
```

$$
\begin{array}{lllcl}
\omega: & \mathtt{A(0)} & & & \\
p_1: & \mathtt{A(d)} & : \mathtt{d} > 0 & \rightarrow & \mathtt{A(d-1)} \\
p_2: & \mathtt{A(d)} & : \mathtt{d} = 0 & \rightarrow & \mathtt{F(1)[+A(D)]F(1)A(0)} \\
p_3: & \mathtt{F(a)} & : * & \rightarrow & \mathtt{F(a*R)}
\end{array}
$$

Parametric L-system

$$(\mathtt{Q}_1 \cup \mathtt{Q}_2)^*(\mathtt{T}_1 \cup \mathtt{T}_2 \cup \mathtt{T}_3)^*$$

$$
\begin{array}{lcl}
\mathtt{Q}_1 & = & \mathtt{s(0.5)} \\
\mathtt{Q}_2 & = & \mathtt{s(0.5)t(0.0, 0.5)}
\end{array}
$$

$$
\begin{array}{lcl}
\mathtt{T}_1 & = & \mathtt{s(1/(R \uparrow (D+1)))r(45)t(0.0, 0.5)} \\
\mathtt{T}_2 & = & \mathtt{s(1/(R \uparrow (D+1)))r(-45)t(0.0, 0.5)} \\
\mathtt{T}_3 & = & \mathtt{s(1/R)t(0.0, 1.0)}
\end{array}
$$

LRIFS

Figure 18: A comparison of structures generated using an L-system and the equivalent LRIFS. Numbers D and R are parameters of the model.

A paper by Feiste [17], closely related to those by Bandt, relaxes the assumption that all maps F_a must be contractions.

Although the ideas of Brandt could be described using the language of the theory of automata, he uses a notion of a "sofic system" instead. The relation between IFS's and formal languages is pursued explicitly by Culik and Dube [11, 13, 12, 14]. Berstel and his co-workers approach IFS's from the language-theoretic point of view as well, although papers [8, 9] are not deeply rooted in the existing literature of the topic. Examples of the application of finite automata to control the application of transformations are given in Master's theses by Sandness [26] and Hepting [20].

Vrscay [27] gives a very thorough survey of IFS's, then presents recurrent IFS's according to the definition by Barnsley, Elton and Hardin, gives collage theorems for recurrent IFS's, and considers the inverse problem (how to construct a recurrent IFS for a given image). The last problem also reoccurs on other papers by Vrscay.

References

[1] Ch. Bandt. Self-similar sets I. Topological Markov chains and mixed self-similar sets. *Math. Nachr.*, 142:107–123, 1989.

[2] Ch. Bandt. Self-similar sets III. Constructions with sofic systems. *Monatsh. Math*, 108:89–102, 1989.

[3] M. F. Barnsley. *Fractals everywhere.* Academic Press, 1988.

[4] M. F. Barnsley and S. Demko. Iterated function systems and the global construction of fractals. *Proceedings of the Royal Society of London Ser. A*, 399:243–275, 1985.

[5] M. F. Barnsley, J. H. Elton, and D. P. Hardin. Recurrent iterated function systems. *Constructive Approximation*, 5:3–31, 1989.

[6] T. Bedford. Dynamics and dimension for fractal recurrent sets. *J. London Math. Soc*, 2(33):89–100, 1986.

[7] Marc A. Berger. Images generated by orbits of 2-D Markov chains. *Chance*, 2(2):18–28, 1989.

[8] J. Berstel and A. Nait Abdallah. Tétrarbres engendrées par des automates finis. Technical Report 89-7, Laboratoire Informatique Théorique et Programmation, Université P. et M. Curie, 1989.

[9] J. Berstel and M. Morcrette. Compact representation of patterns by finite automata. Technical Report 89-66, Laboratoire Informatique Théorique et Programmation, Université P. et M. Curie, 1989.

[10] I. N. Bronshtein and K. A. Semendyayev. *Handbook of mathematics.* Van Nostrand Reinhold Co., 1979.

[11] Karel Culik II and Simant Dube. Affine automata and related techniques for generation of complex images. Manuscript, Department of Computer Science, University of South Carolina, Columbia, SC, 1990.

[12] Karel Culik II and Simant Dube. Balancing order and chaos in image generation. Manuscript, Department of Computer Science, University of South Carolina, Columbia, SC, 1991.

[13] Karel Culik II and Simant Dube. Encoding images as words and languages. Manuscript, Department of Computer Science, University of South Carolina, Columbia, SC, 1991.

[14] Karel Culik II and Simant Dube. Rational and affine expressions for image description. *Discrete Applied Mathematics*, 1991. To appear.

[15] F. M. Dekking. Substitutions, branching processes and fractal sets. In *Proceedings of the NATO Advanced Study Institute on Fractal Geometry, Montreal, July 1989*. Kluwer Academic Publishers, 1991. To appear.

[16] S. Dubuc and A. Elqortobi. Approximations of fractal sets. *Journal of Computational and Applied Mathematics*, 29:79–89, 1990.

[17] U. Feiste. A generalization of mixed invariant sets. *Monatsh. Math*, 108, 1989.

[18] William J. Gilbert. Complex bases and fractal similarity. *Ann. Sc. Math., Québec*, 11(1):65–77, 1987.

[19] D. Hepting, P. Prusinkiewicz, and D. Saupe. Rendering methods for iterated function systems. In *Proceedings of FRACTAL '90, the First IFIP Conference on Fractals, Lisbon, Portugal*, June 1990. To appear.

[20] D. H. Hepting. Approximation and visualization of sets defined by iterated function systems. Master's thesis, Department of Computer Science, University of Regina, Regina, Saskatchewan, 1991.

[21] J. E. Hutchinson. Fractals and self-similarity. *Indiana University Journal of Mathematics*, 30(5):713–747, 1981.

[22] B. B. Mandelbrot. *The Fractal Geometry of Nature.* W. H. Freeman, New York, 1982.

[23] P. Prusinkiewicz and J. Hanan. *Lindenmayer systems, fractals, and plants*, volume 79 of *Lecture Notes in Biomathematics.* Springer-Verlag, Berlin, 1989.

[24] P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants.* Springer-Verlag, New York, 1990. With J. Hanan, F. D. Fracchia, D. R. Fowler, M. J. M. de Boer and L. Mercer.

[25] P. Prusinkiewicz and G. Sandness. Koch curves as attractors and repellers. *IEEE Computer Graphics and Applications*, 8(6):26–40, November 1988.

[26] G. Sandness. Fractal generation by iterative geometric transformations. Master's thesis, Department of Computer Science, University of Regina, Regina, Saskatchewan, 1988.

[27] E. R. Vrscay. Iterated function systems: Theory, applications and the inverse problem. In *Proceedings of the NATO Advanced Study Institute on Fractal Geometry, Montreal, July 1989.* Kluwer Academic Pulishers, 1991. To appear.

[28] R. F. Williams. Composition of contractions. *Bol. Soc. Brasil. Mat.*, 2:55–59, 1971.

[29] T. E. Womack. Linear and Markov iterated function systems in fractal geometry. Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, Virginia, 1989.