

Constructive Fractal Geometry : constructive approach to fractal modeling using language operations

J. THOLLOT and E. TOSAN

LIGIA-LISPI, bât 710

43, boulevard du 11 Novembre 1918

69622 VILLEURBANNE Cedex – France

{jthollot,et}@ligia.univ-lyon1.fr

Abstract

Using a generalization of the IFS model, we try to establish relations between operations over languages and operations over attractors. This leads us to a constructive approach of fractal geometry.

Keywords : Iterated Function Systems, Formal Languages, Transition Systems, Constructive Solid Geometry.

1 Introduction

The Iterated Function Systems (IFS) model has been introduced by BARNSEY [Bar88] for fractal geometry. This model is particularly interesting due to its rigorous formalism and its simplicity : a fractal is encoded by a finite number of contractive transformations.

Several authors have generalized the IFS model : Equations systems [CD92][CD93a][Har92] and Matrices [PJS92] are used to define attractor vectors. Languages accepted by a finite-state automaton [PH91][PH92][BM89][BNA89], directed multigraphs [MW88][Edg90] and affine expressions [CD93b] are used to define a subset of an IFS attractor. These models have one common point : they are all related to formal languages theory.

We have tried to define a constructive approach to fractal geometry adapted from Constructive Solid Geometry (CSG) [Req80][Rot82] in solid modeling. This can be done using languages. Indeed, there are relations between operations over languages and operations over attractors. For this purpose we have chosen to use the Language-Restricted Iterated Function Systems approach [PH91] because it seems to be the most general approach among those cited above. However, we present a slightly different definition of the attractor based on a transition system.

2 Definitions

The LRIFS model has been introduced by PRUSINKIEWICZ and HAMMEL [PH91][PH92]. It provides tools for restricting the sequences of applicable transformations of an IFS using a formal language. The definition of an attractor we use is adapted from LRIFS's using a transition system instead of a language. Thus, we will give the definitions of an IFS and a LRIFS and present our definition.

2.1 Iterated Function Systems

The IFS model is based on an application of the fixed point theorem in the set of the compact sets of a metric space.

Definition 2.1 *An IFS is a set $T = \{T_1, \dots, T_N\}$ of contractive transformations on \mathcal{X} .*

Notation : Let K be a compact set, we denote

$$T \circ K = \bigcup_{T \in T} T(K) = \{T(p) | T \in T, p \in K\}$$

Moreover, we have chosen to compose transformations from right to left because it is the classical notation for matrix products on which our approach is based.

Theorem 2.1 *Denote $\mathcal{H}(\mathcal{X})$ the set of all non-empty compact sets of \mathcal{X} . $(\mathcal{H}(\mathcal{X}), d_H)$ is a complete metric space, where d_H is the HAUSDORFF distance. The HUTCHINSON operator defined by :*

$$\begin{array}{ccc} F : \mathcal{H}(\mathcal{X}) & \longrightarrow & \mathcal{H}(\mathcal{X}) \\ K & \longmapsto & T \circ K \end{array}$$

is a contraction on $\mathcal{H}(\mathcal{X})$. Thus this operator has a unique fixed point :

$$A = F(A) = T \circ A$$

A is called the attractor of T and is denoted $\mathcal{A}(T)$.

Proof : the proof can be found in [Bar88].

2.2 Language-Restricted Iterated Function Systems

PRUSINKIEWICZ and HAMMEL [PH91][PH92] define an IFS as a tuple of functions. This enables them to define an alphabet of contraction labels and a language over this alphabet.

Definition 2.2 A *LRIFS* is a 4-tuple $\mathcal{I}_L = (T, \Sigma, h, L)$ where :

- $T = (T_1, \dots, T_N)$ is a tuple of contractions on \mathcal{X} .
- $\Sigma = \{1, \dots, N\}$ is an alphabet of contraction labels.
- h is a labeling function : $h(i) = T_i$ for $i \in \Sigma$.
- $L \subset \Sigma^*$ is a language over Σ .

The function h is generalized to languages over Σ using the following equations :

$$\begin{aligned} h(\epsilon) &= \text{Identity} \\ h(u_1 u_2 \dots u_k) &= T_{u_1} \circ T_{u_2} \circ \dots \circ T_{u_k} \\ h(L) &= \{h(w) | w \in L\} \end{aligned}$$

where ϵ is the empty word.

2.3 Attractor associated with a transition system

We have chosen to work only with languages accepted by a transition system, that is regular languages. This approach allows us to define an attractor associated to a transition system.

This definition [TT93] is based on the equivalence between a transition system [Har78], the graph of this transition system and the matrix associated with this graph [GM86]. Using this matrix we can apply the fixed point theorem in $\mathcal{H}(\mathcal{X})^n$ as it has been done in [PJS92] in order to produce an attractor vector. The attractor associated with the transition system will then be a projection of this vector.

More precisely :

Definition 2.3 A transition system is a 5-tuple $\mathcal{M} = (Q, \Sigma, \delta, Q_I, Q_F)$ where :

- Q is a finite nonempty set of states.
- Σ is an alphabet.
- δ is a function from $Q \times \Sigma$ into Q called the direct transition function.
- $Q_I \subset Q$ is the set of initial states.
- $Q_F \subset Q$ is the set of final states.

The language accepted by \mathcal{M} is given by

$$L(\mathcal{M}) = \{w \in \Sigma^*, \delta(q_i, w) = q_f, q_i \in Q_I, q_f \in Q_F\}$$

with $\delta(q, aw) = \delta(\delta(q, a), w)$ if $a \in \Sigma$ and $w \in \Sigma^*$.

$L(\mathcal{M})$ is a regular language.

\mathcal{M} can be viewed as a graph in which vertices represent the states and edges the transition function. The initial (resp. final) states are pointed by a short entering (resp. outgoing) arrow.

Definition 2.4 Let \mathcal{M} be a transition system. The matrix associated with \mathcal{M} is a $n \times n$ matrix $\Delta = (\Delta_{ij})$ where n is the number of states of \mathcal{M} and :

$$\forall i, j = 1, \dots, n \quad \Delta_{ij} = \{a \in \Sigma | \delta(q_i, a) = q_j\}$$

The function h is generalized to $n \times n$ matrices using the following equations :

$$\begin{aligned} h(\Delta)_{ij} &= h(\Delta_{ij}) \\ h(\Delta) \circ V &= \left(\sum_{j=1}^n h(\Delta_{ij}) \circ V_j \right) \end{aligned}$$

with $V \in \mathcal{H}(\mathcal{X})^n$.

Proposition 2.1 $h(\Delta)$ is a contractive operator. Thus it has a unique fixed point \mathcal{V} in $\mathcal{H}(\mathcal{X})^n$. The attractor associated with \mathcal{M} is then defined by :

$$\mathcal{A}(\mathcal{M}) = \bigcup_{q_i \in Q_I} \mathcal{V}_i$$

Proof : see [PJS92][TT93].

This approach enables us to visualize the attractor using the deterministic algorithm. Indeed, we can construct the sequence $(h(\Delta)^n \circ V)_{n \in \mathbb{N}}$, where $V \in \mathcal{H}(\mathcal{X})^n$. This sequence converges to \mathcal{V} .

3 Languages operations and attractors

We have started developing a constructive approach of fractals by using operations over IFS's [Gen92][GT91]. This approach enables the creation "step by step" of an image. Using simple shapes (not necessarily fractals) we are able to construct complex fractal shapes. Thus it gives a control on the image construction.

The use of languages theory allows us to extend this approach by using operations over languages. Indeed, the set of regular languages is closed under the union, concatenation, intersection and shuffle operations [Har78] and thus, it is possible to construct complex languages using simple ones. Moreover, these operations over languages have simple equivalent operations over transition systems. Using definition 2.1, we will be able to produce the attractor corresponding to the result of these operations. Thus we will be able to built an attractor by composing simple attractors.

Our question is : given an arbitrary operation $*$ and two transition systems $\mathcal{M} = (Q, \Sigma, \delta, Q_I, Q_F)$ and $\mathcal{M}' = (Q', \Sigma, \delta', Q'_I, Q'_F)$ over the same alphabet, what is the relation between $\mathcal{A}(\mathcal{M} * \mathcal{M}')$, $\mathcal{A}(\mathcal{M})$ and $\mathcal{A}(\mathcal{M}')$? We will present for each operation the results we obtain. Proofs can be found in [TT94].

3.1 Union

We present here what the attractor associated with the union of two languages is. The union of two languages L and L' is defined by :

$$L \cup L' = \{w \in \Sigma^* / w \in L \text{ or } w \in L'\}$$

The transition system \mathcal{M}'' that accepts $L(\mathcal{M}'') = L(\mathcal{M}) \cup L(\mathcal{M}')$ is :

$$\mathcal{M}'' = (Q'', \Sigma, \delta'', Q''_I, Q''_F)$$

where

$$\begin{aligned} Q'' &= Q \cup Q' \\ Q''_I &= Q_I \cup Q'_I \\ Q''_F &= Q_F \cup Q'_F \\ \delta''(q, a) &= \begin{cases} \delta(q, a) & \text{if } q \in Q \\ \delta'(q, a) & \text{if } q \in Q' \end{cases} \end{aligned}$$

Proposition 3.1

$$\mathcal{A}(\mathcal{M}'') = \mathcal{A}(\mathcal{M}) \cup \mathcal{A}(\mathcal{M}')$$

3.2 Concatenation

We now present what the attractor associated with the concatenation of two languages is. It is given by :

$$L.L' = \{w \in \Sigma^* / w = uv, u \in L, v \in L'\}$$

Let \mathcal{M}'' be the transition system that accepts $L(\mathcal{M}'') = L(\mathcal{M}).L(\mathcal{M}')$. \mathcal{M}'' is obtained by connecting each final state of \mathcal{M} to each initial state of \mathcal{M}' by an ϵ -transition.

Proposition 3.2

$$\mathcal{A}(\mathcal{M}'') = \mathcal{A}(\mathcal{M}) \cup h(L(\mathcal{M})) \circ \mathcal{A}(\mathcal{M}')$$

3.3 Intersection

The intersection of two languages is given by :

$$L \cap L' = \{w \in \Sigma^* / w \in L \text{ and } w \in L'\}$$

The transition system \mathcal{M}'' that accepts $L(\mathcal{M}'') = L(\mathcal{M}) \cap L(\mathcal{M}')$ is :

$$\mathcal{M}'' = (Q'', \Sigma, \delta'', Q''_I, Q''_F)$$

where

$$\begin{aligned} Q'' &= Q \times Q' \\ Q''_I &= Q_I \times Q'_I \\ Q''_F &= Q_F \times Q'_F \\ \delta''((q_1, q_2), a) &= (q'_1, q'_2) \text{ if } \begin{cases} \delta(q_1, a) = q'_1 \\ \text{and} \\ \delta'(q_2, a) = q'_2 \end{cases} \end{aligned}$$

Proposition 3.3 If $L(\mathcal{M}) \cap L(\mathcal{M}') \neq \emptyset$ then

$$\mathcal{A}(\mathcal{M}'') \subseteq \mathcal{A}(\mathcal{M}) \cap \mathcal{A}(\mathcal{M}')$$

3.4 Shuffle

The shuffle of two languages is given by :

$$\begin{aligned} L \sqcup L' &= \{w = u_1 v_1 u_2 v_2 \dots u_m v_m \in \Sigma^* \mid \\ &\quad u_1 u_2 \dots u_m \in L, v_1 v_2 \dots v_m \in L' \\ &\quad u_i, v_i \in \Sigma \cup \{\epsilon\}\} \end{aligned}$$

The transition system \mathcal{M}'' that accepts $L(\mathcal{M}'') = L(\mathcal{M}) \sqcup L(\mathcal{M}')$ is :

$$\mathcal{M}'' = (Q'', \Sigma, \delta'', Q''_I, Q''_F)$$

where

$$\begin{aligned} Q'' &= Q \times Q' \\ Q''_I &= Q_I \times Q'_I \cup Q \times Q'_I \\ Q''_F &= Q_F \times Q'_I \cup Q \times Q'_F \\ \delta''((q, q'), a) &= \begin{cases} (q'', q') & \text{if } \delta(q, a) = q'' \\ (q, q'') & \text{if } \delta'(q', a) = q'' \end{cases} \end{aligned}$$

\mathcal{M}'' is not necessarily determinist.

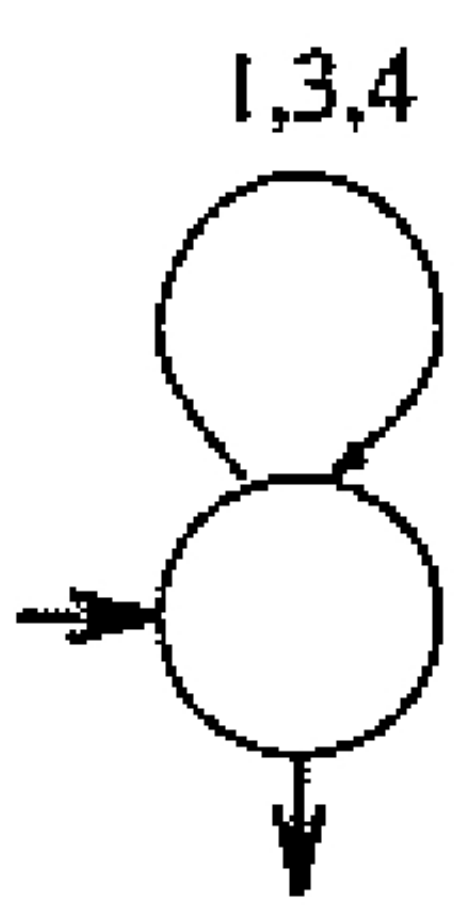
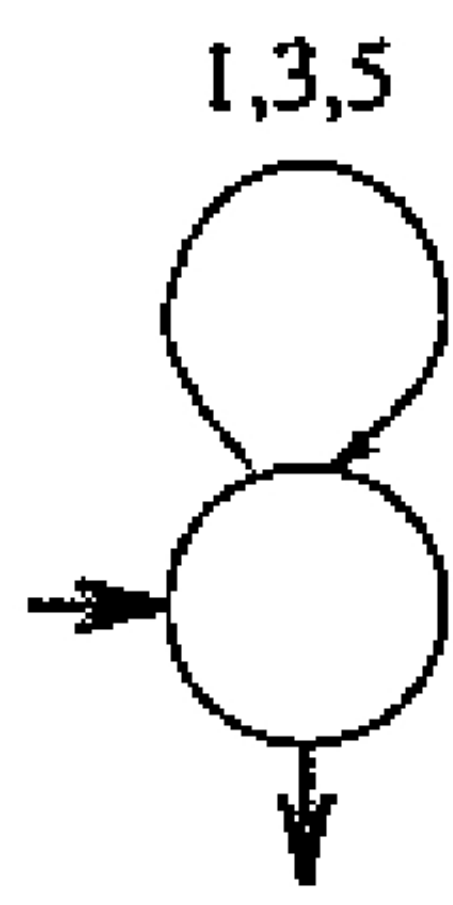
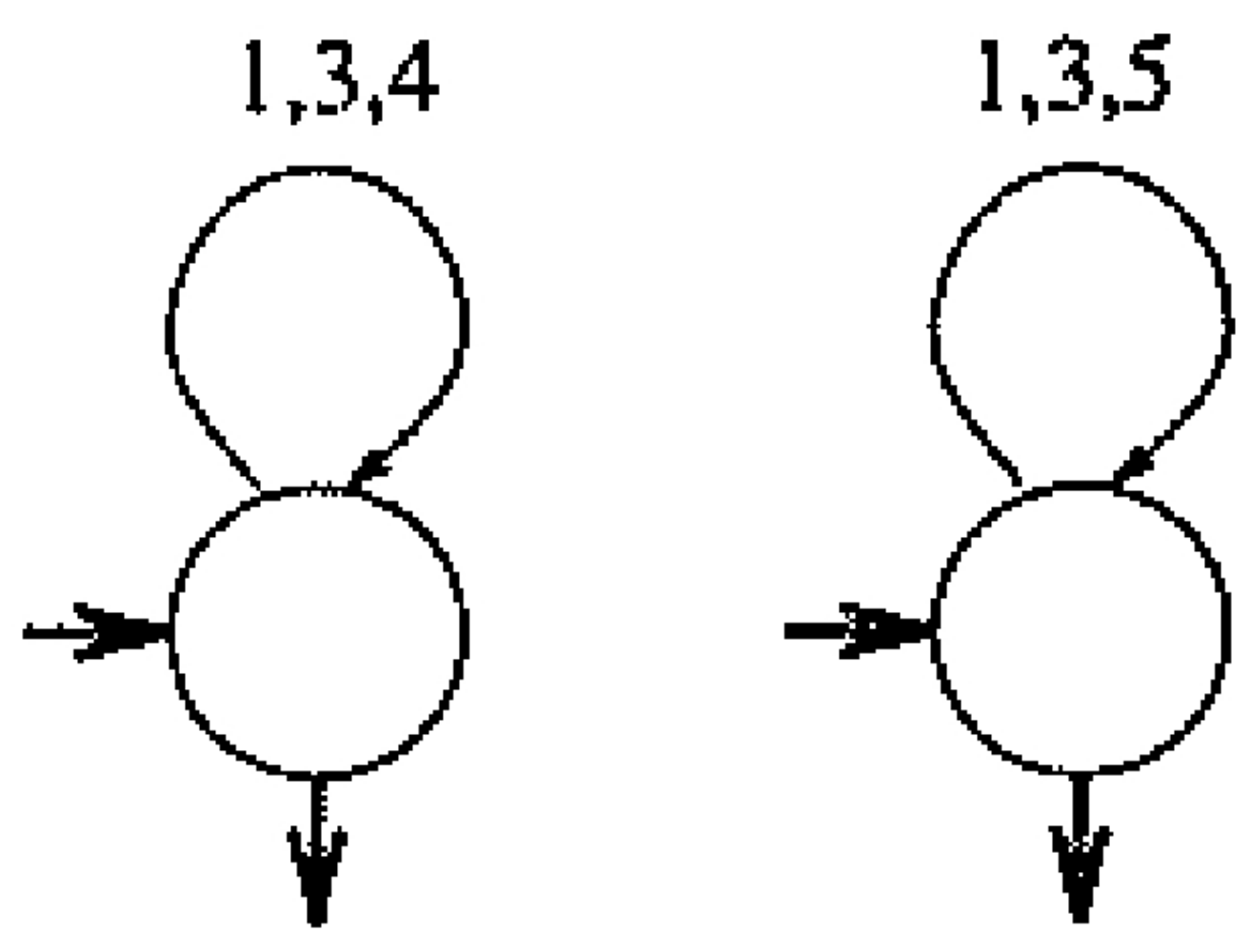

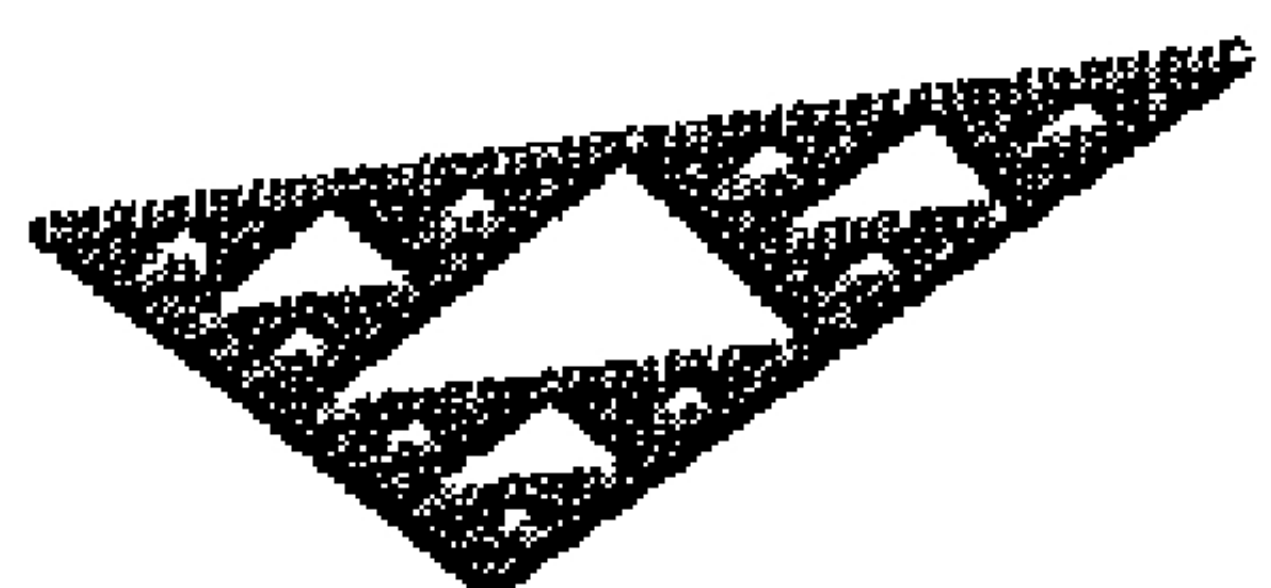
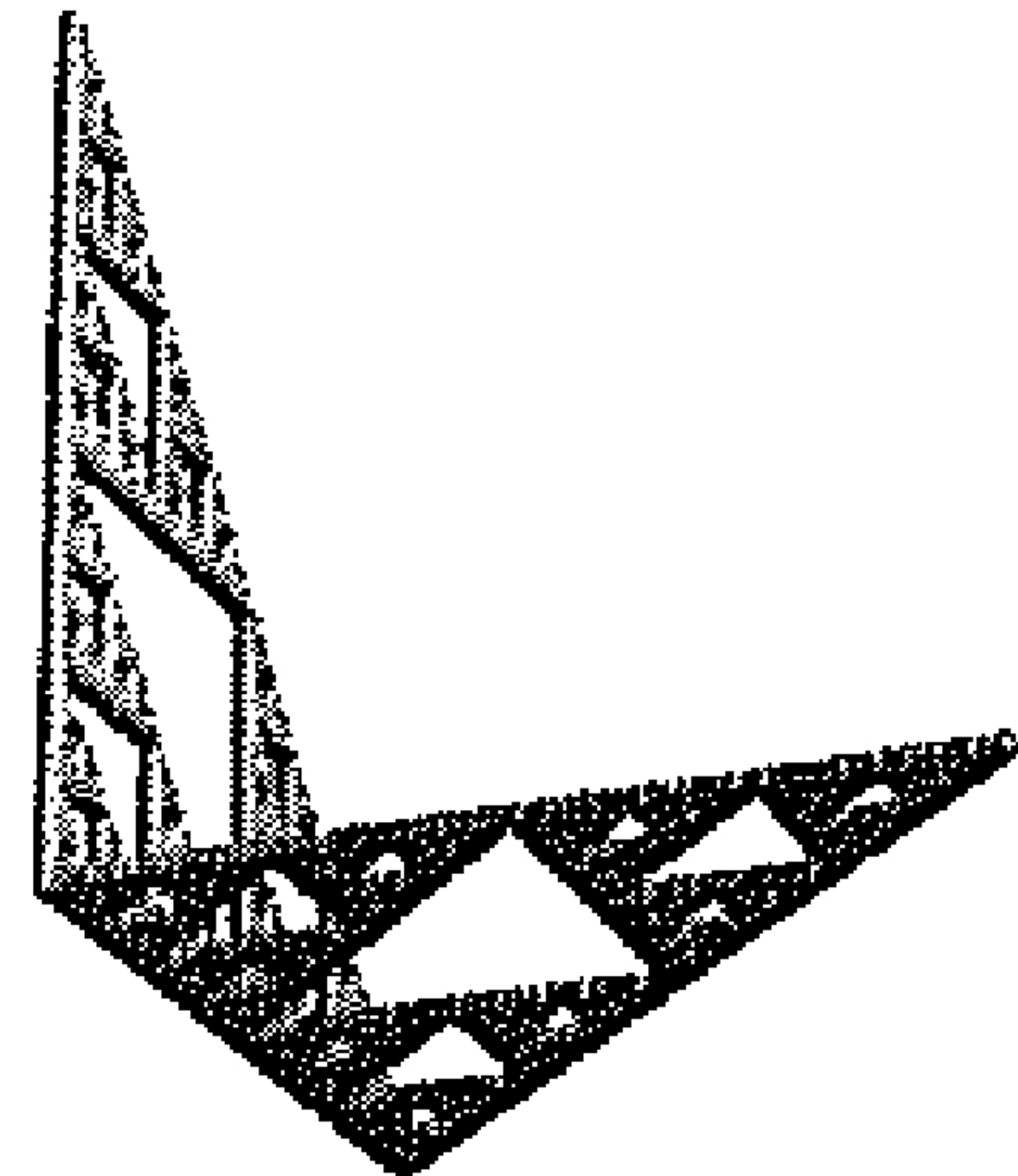
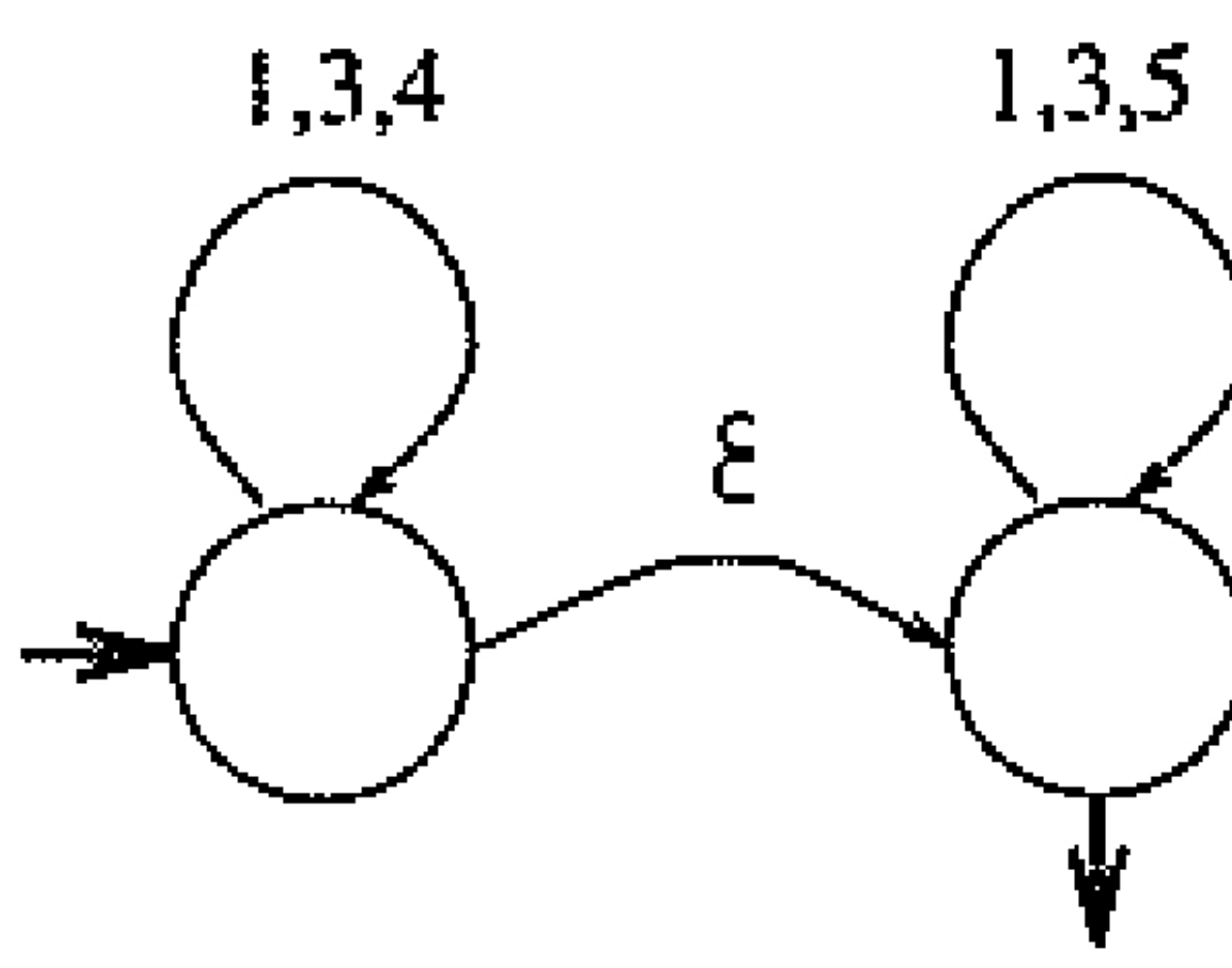
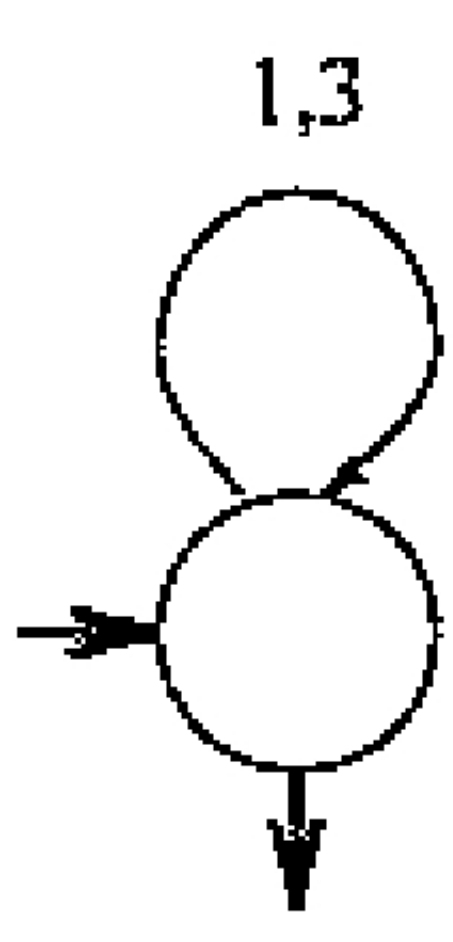
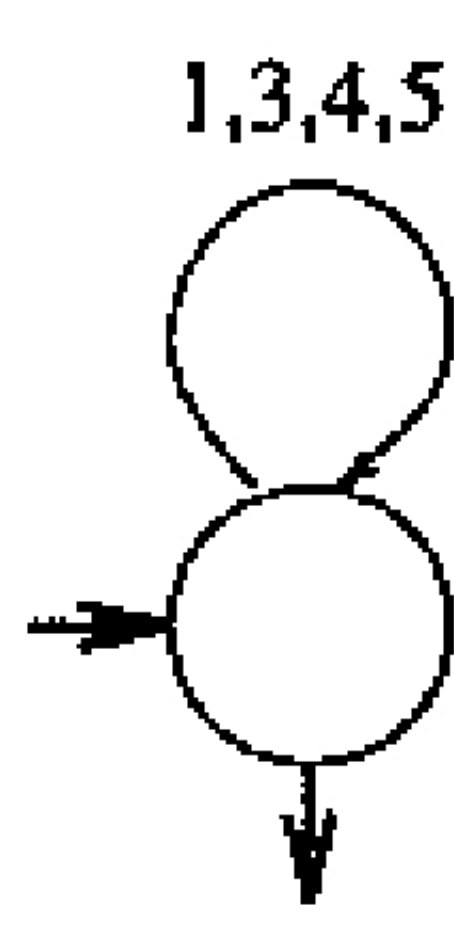
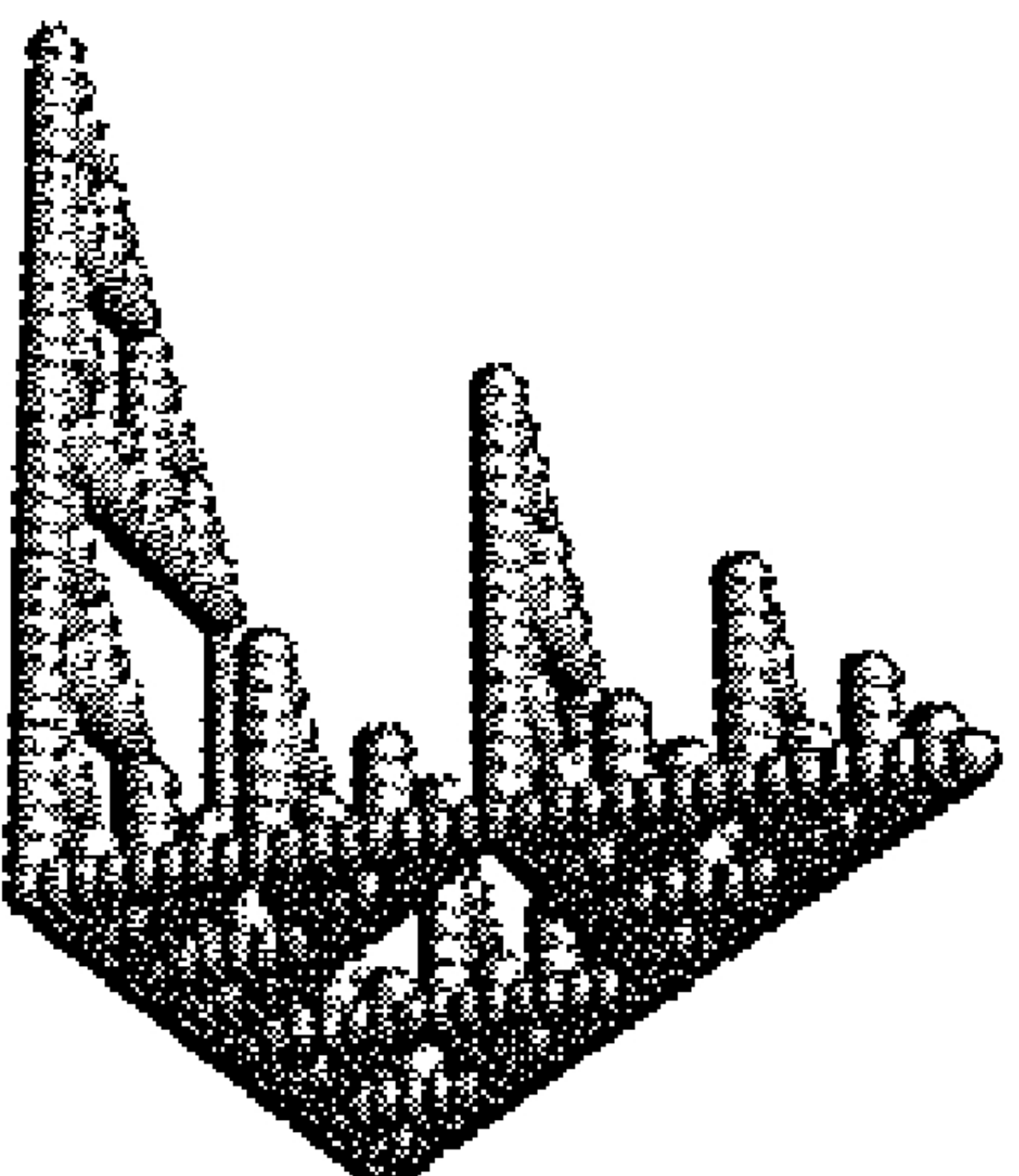

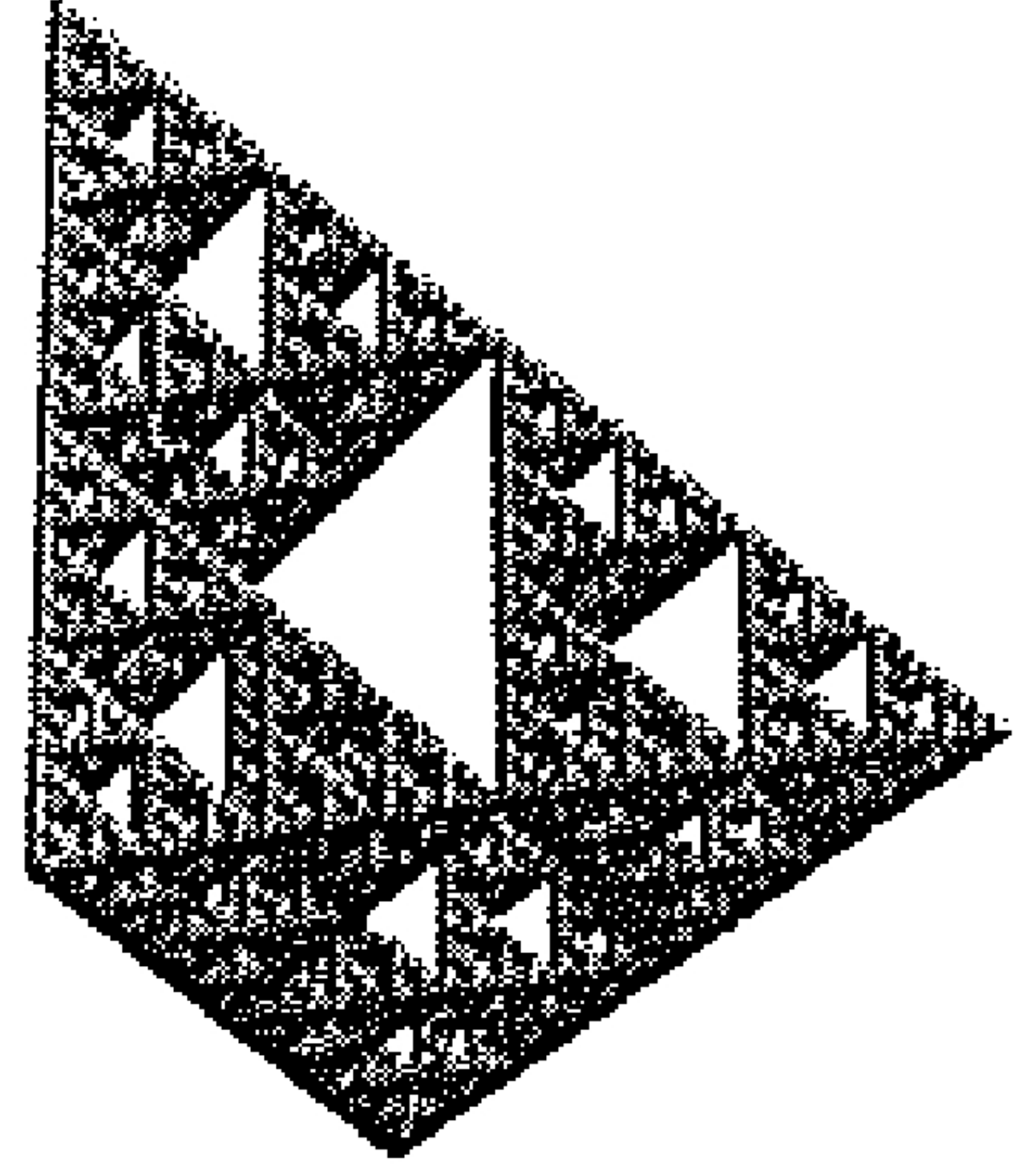
\mathcal{M}	\mathcal{M}'	$\mathcal{M} \cup \mathcal{M}'$
		
		
$\mathcal{M} \cdot \mathcal{M}'$	$\mathcal{M} \cap \mathcal{M}'$	$\mathcal{M} \sqcup \mathcal{M}'$
		
		

Figure 1: Examples of operations over attractors.

Proposition 3.4

$$\begin{aligned}\mathcal{A}(\mathcal{M}'') &\supseteq \mathcal{A}(\mathcal{M}) \cup \mathcal{A}(\mathcal{M}') \\ \mathcal{A}(\mathcal{M}'') &\supseteq h(L(\mathcal{M}')) \circ \mathcal{A}(\mathcal{M}) \cup h(L(\mathcal{M})) \circ \mathcal{A}(\mathcal{M}')\end{aligned}$$

3.5 Examples

In order to give examples we use affine transformations of $\mathcal{X} = \mathbb{R}^3$. The following notations are used :

- $T(a, b, c)$ denotes the translation by the vector (a, b, c) .
- $Rx(a)$ (resp. $Ry(a), Rz(a)$) denotes the rotation of angle a around the Ox (resp. Oy, Oz) axis.
- $H(a)$ denotes the scaling with respect to the origin of the coordinate system.

The visualization is made by the deterministic algorithm [TT93]. The primitive we use in the algorithm is the sphere. To illustrate each operation we use the octree transformations :

$$\begin{aligned}S_1 &= H(0.5) \\ S_2 &= T(0.5, 0, 0) \circ H(0.5) \\ S_3 &= T(0, 0.5, 0) \circ H(0.5) \\ S_4 &= T(0, 0, 0.5) \circ H(0.5) \\ S_5 &= T(0.5, 0.5, 0) \circ H(0.5) \\ S_6 &= T(0.5, 0, 0.5) \circ H(0.5) \\ S_7 &= T(0, 0.5, 0.5) \circ H(0.5) \\ S_8 &= T(0.5, 0.5, 0.5) \circ H(0.5).\end{aligned}$$

Let $L(\mathcal{M}) = \{1, 3, 4\}^*$ and $L(\mathcal{M}') = \{1, 3, 5\}^*$. Figure 1 shows \mathcal{M} , \mathcal{M}' , $\mathcal{M} \cup \mathcal{M}'$, $\mathcal{M} \cdot \mathcal{M}'$, $\mathcal{M} \cap \mathcal{M}'$, $\mathcal{M} \sqcup \mathcal{M}'$ and the corresponding attractors.

3.6 Remark on attractors definitions

Our definition of an attractor and PRUSINKIEWICZ's one [PH91] are slightly different. Indeed, in our definition, an attractor is a fixed point in the set of compact sets. Thus it does not depend on an initial point. It is a way to avoid the sufficient condition (prefix extensibility of the language) given in [PH91]. Proofs can be found in [TT94]. However the shuffle and the concatenation of two prefix extensible languages are prefix extensible and thus PRUSINKIEWICZ definition allows these operations.

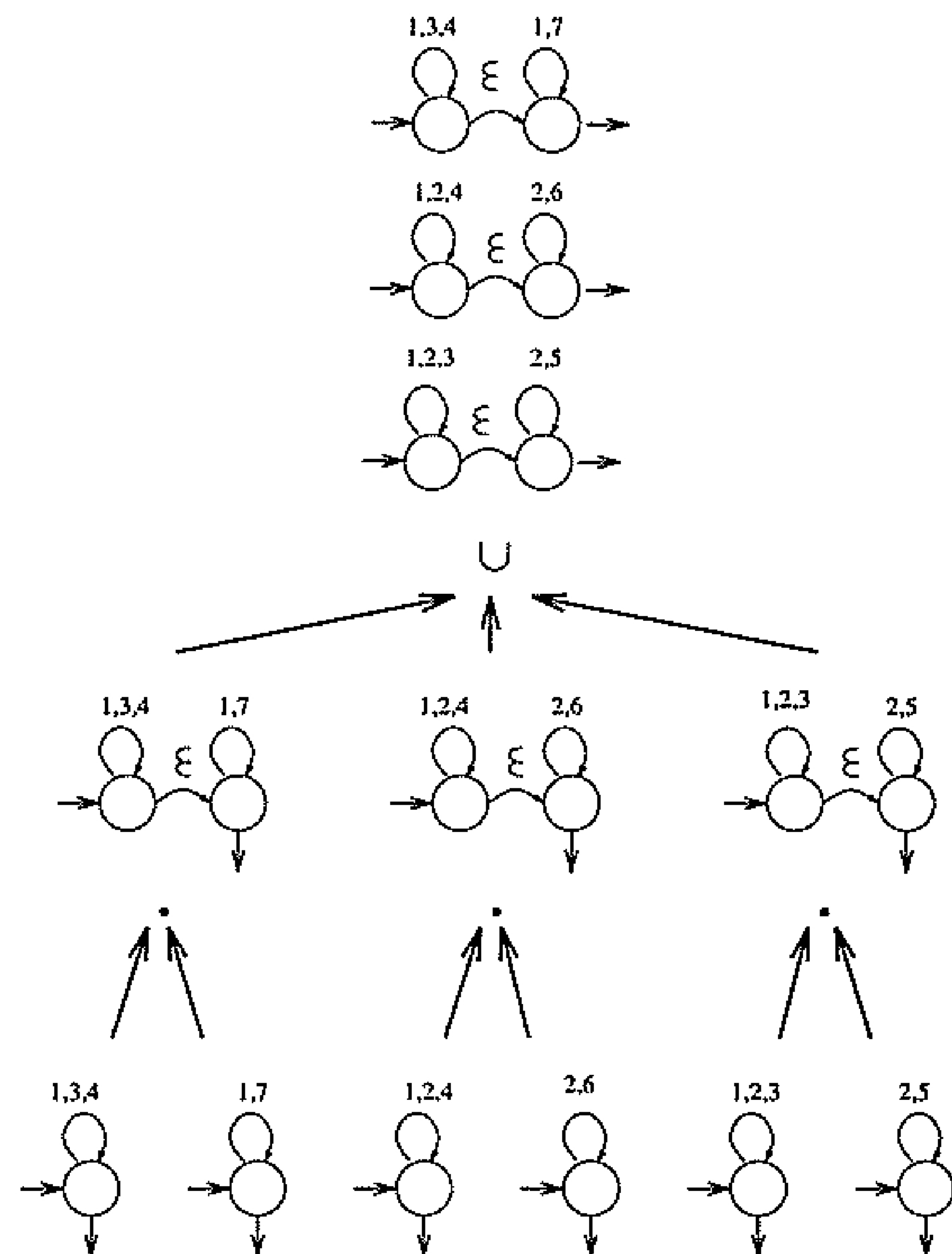


Figure 3: Construction tree of a language

4 Constructive Fractal Geometry

We have defined operations over languages that allow to compose two attractors. These operations can be used, as in Constructive Solid Geometry, to build a construction tree. We will give an example of “Constructive Fractal Geometry tree”, and then investigate the relation between CSG and our approach of Constructive Fractal Geometry.

4.1 Construction trees

We can now construct “CFG trees” in the same way as CSG trees. Indeed, given simple languages we can construct a complex language using operations. Thus, given simple fractals we can construct a complex fractal using these operations.

Example : We still use the octree transformations. Figure 3 shows the construction tree of the language and figure 2 shows the corresponding “CFG tree”.

4.2 Operations

We have seen that the union operation is the same in our approach as in CSG. On the contrary, the intersection operation leads to a particular problem. In-

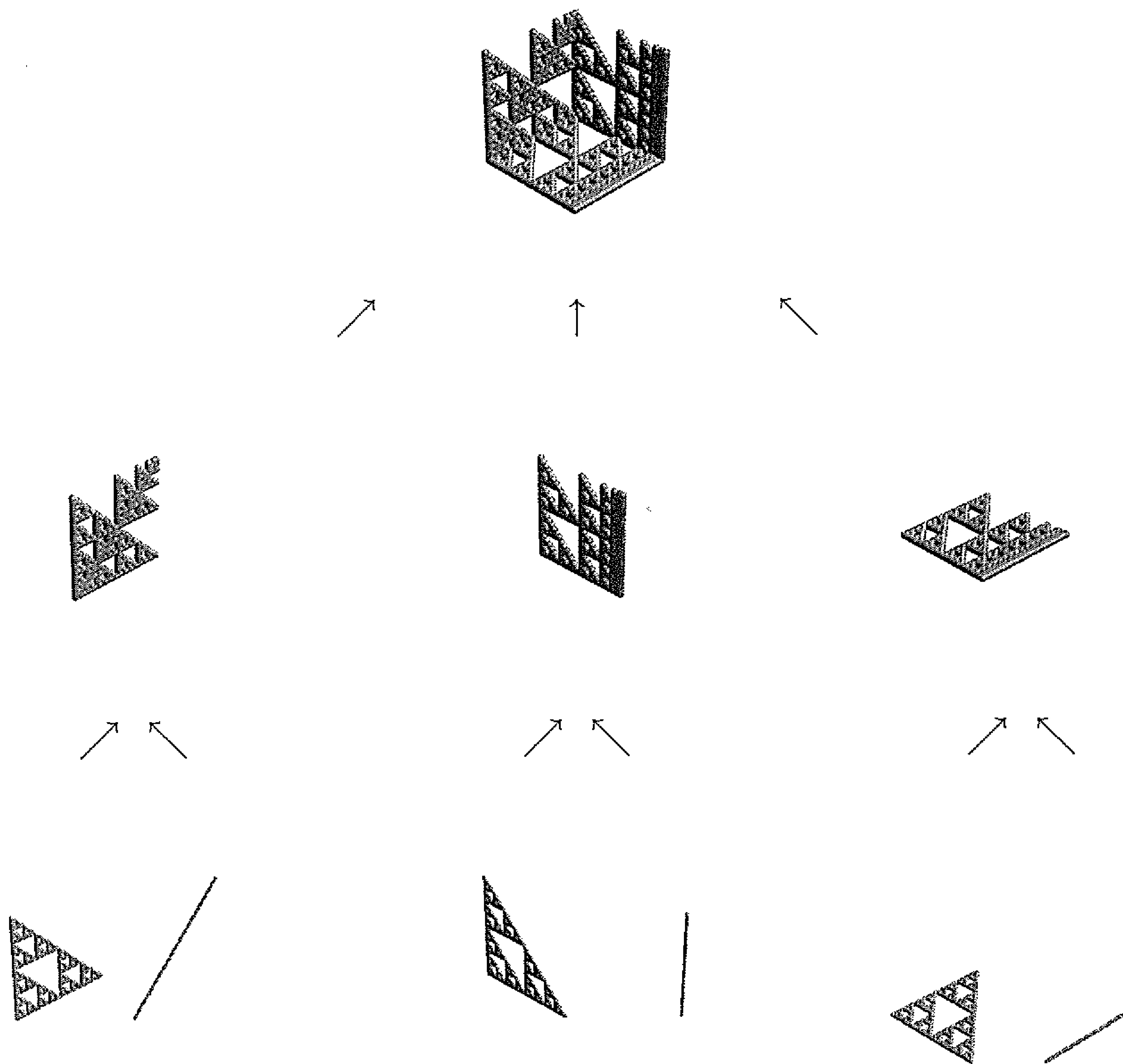


Figure 2: CFG tree

deed, two different languages can produce the same attractor and the result of the intersection may depend on the choice of the languages.

For instance, let $T_1 = H(0.5)$, $T_2 = H(0.3)$. If $L(\mathcal{M}) = \{1\}^*$ and $L(\mathcal{M}') = \{2\}^*$ then $\mathcal{A}(\mathcal{M}) = \mathcal{A}(\mathcal{M}') = \{O\}$: the origin of the coordinate system. Thus $L(\mathcal{M}) \cap L(\mathcal{M}') = \emptyset$ when $\mathcal{A}(\mathcal{M}) = \mathcal{A}(\mathcal{M}')$.

The concatenation and shuffle operations are typical operations over languages and thus they have no equivalent in CSG. The obtained results are often interesting from a graphical point of view.

Example : In addition to S_1, \dots, S_8 given in section 3.5, we use the following transformations :

$$\begin{aligned} S_9 &= T(0, 1, 0) \\ S_{10} &= H(1/3) \\ S_{11} &= H(1/3) \circ T(1, 0, 0) \circ Ry(\pi/3) \\ S_{12} &= S_b \circ T(1, 0, 0) \circ Ry(-2\pi/3) \\ S_{13} &= H(1/3) \circ T(2, 0, 0) \\ S_{14} &= T(0, 0, 1) \circ Ry(\pi/4) \circ H(1/2) \\ S_{15} &= T(0, 0, 1) \circ Ry(-\pi/4) \circ H(1/2) \end{aligned}$$

Figure 4 gives two examples of concatenation operation :

$$L_1 = \{1, 4, 5\}^*, L'_1 = \{1, 2, 4\}^*.9$$

$$L_2 = \{1, 4, 7\}^*, L'_2 = \{10, 11, 12, 13\}^*$$

and an example of shuffle :

$$L_3 = \{1, 4\}^*.\{14, 15\}^*, L'_3 = \{1, 4\}^*.\{14, 15\}^*.9$$

This example shows that one can make extrusion of an attractor using the shuffle operation. The transformations of the tree are those used in [PH91].

4.3 Primitives

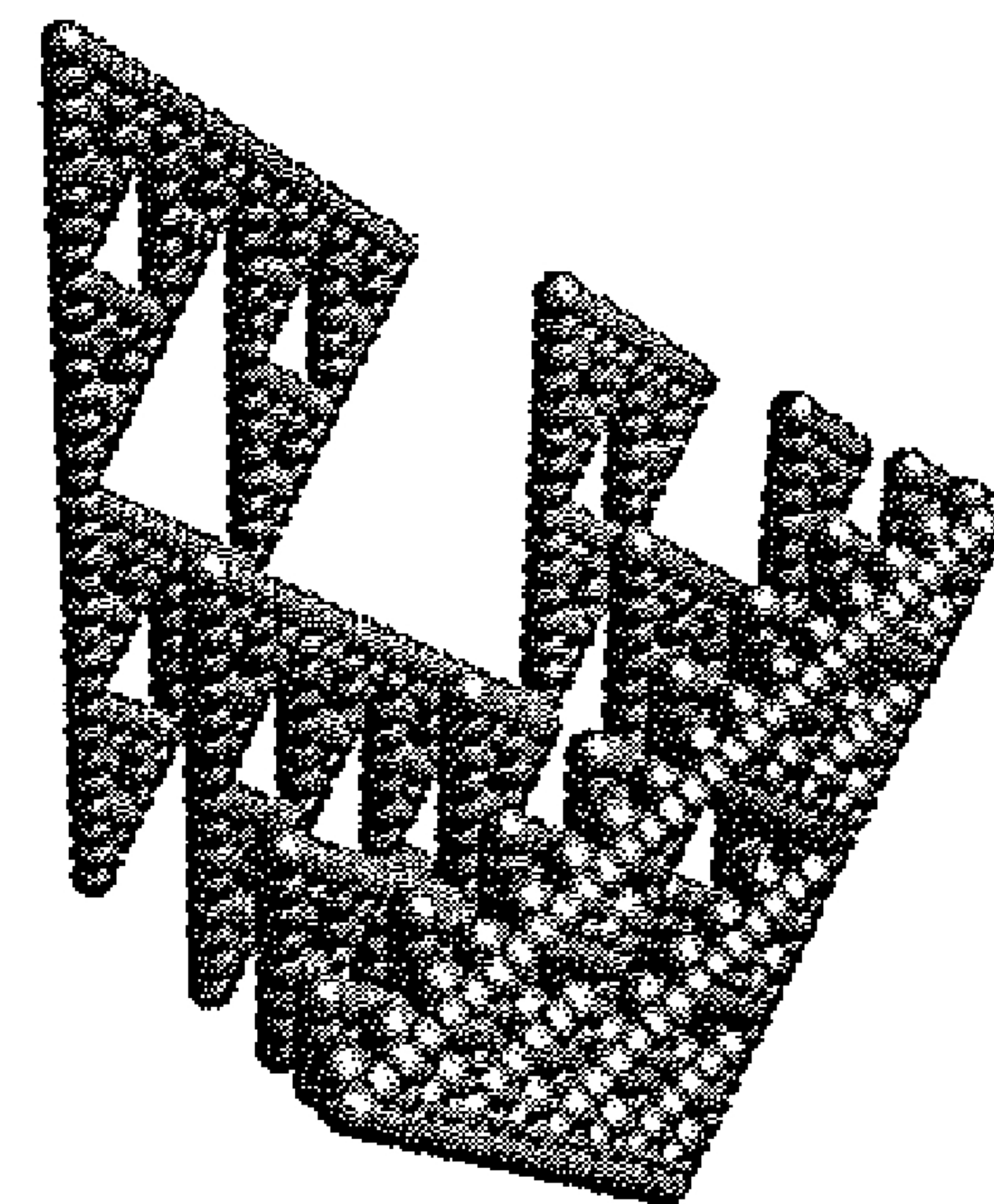
As in CSG we should define what the primitives of the Constructive Fractal Geometry are. The set of these primitives should be an independant set of languages such that any other language could be constructed from this set by applying the above operations.

Proposition 4.1 *Let $\Sigma = \{1, 2, \dots, N\}^*$ be an alphabet. Then the set of languages :*

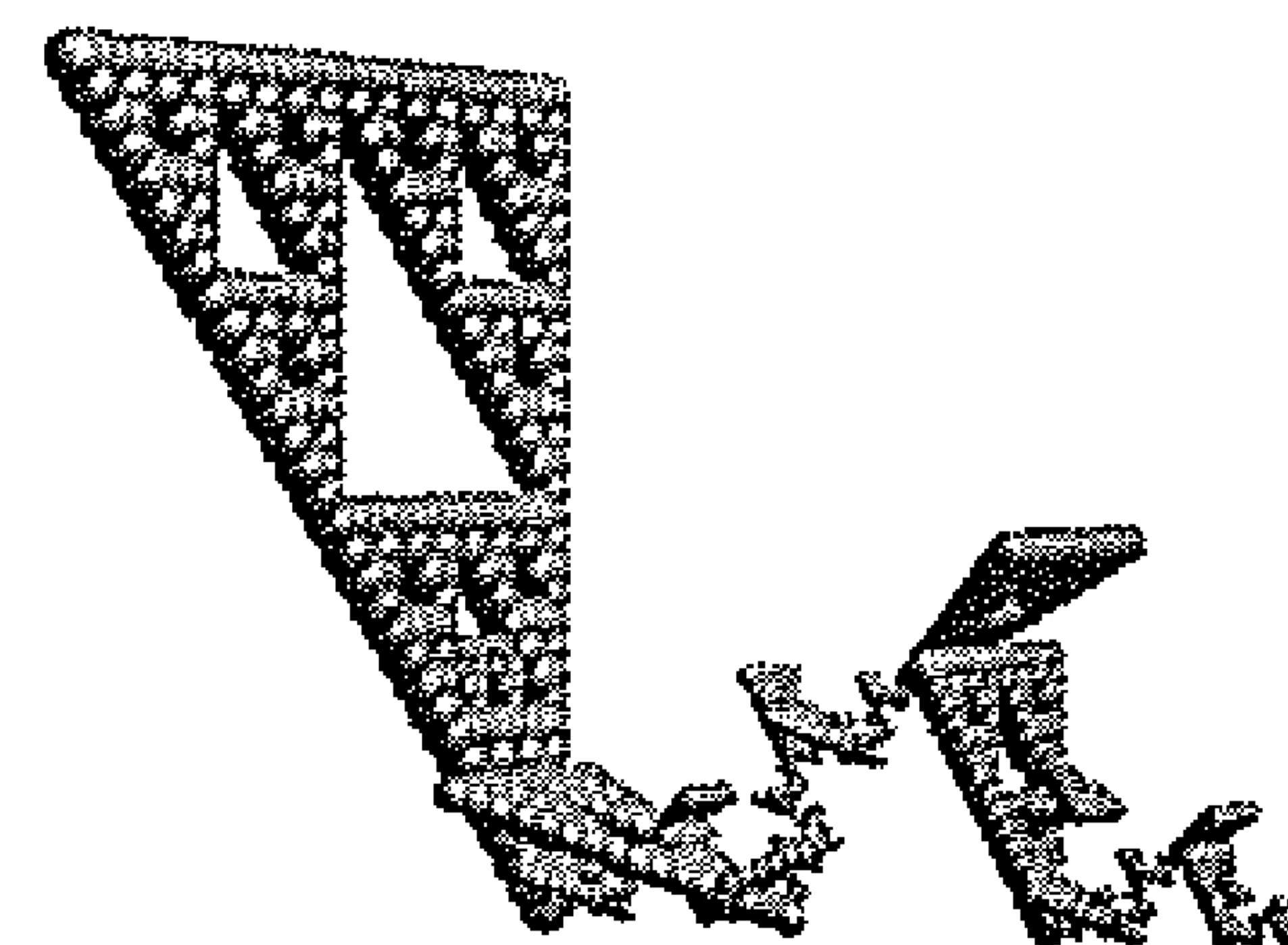
$$\{\emptyset, \{\epsilon\}, \{1\}, \{2\}, \dots, \{N\}, \{1\}^*, \{2\}^*, \dots, \{N\}^*\}$$

is a set of primitives of the Constructive Fractal Geometry.

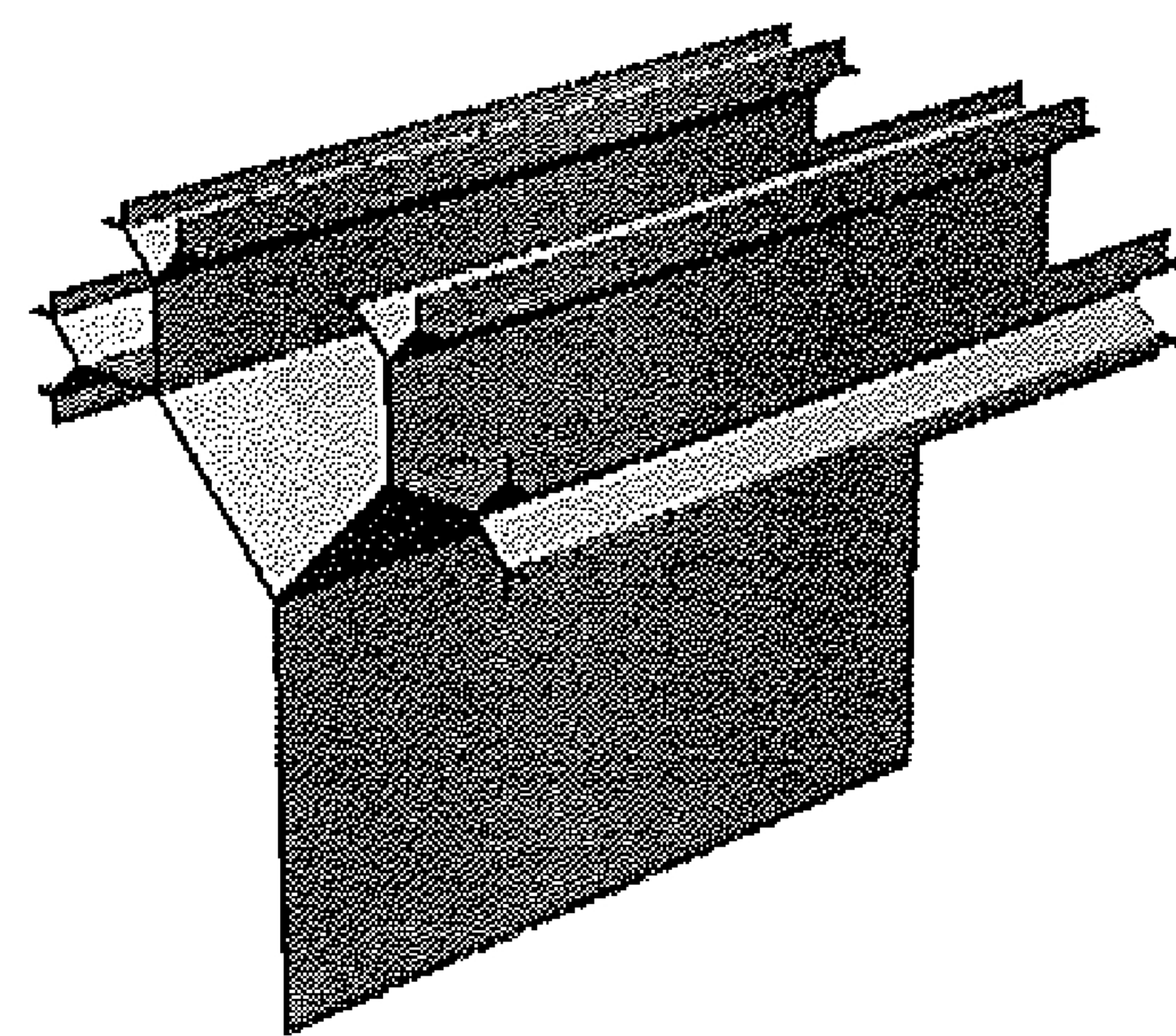
Proof : See [TT94]



$L_1.L'_1$



$L_2.L'_2$



$L_3 \sqcup L'_3$

Figure 4: example of concatenations and shuffle

5 Conclusion

We have tried to define a constructive approach for fractals. This approach allows construction of complex fractal shapes from simple ones and provides not only classical operations of CSG (union, intersection) but also languages operations (concatenation, shuffle).

There are still open questions. In particular, the intersection of two languages does not produce the intersection of the two attractors associated with these languages. Indeed, the result we obtain depends on the languages chosen to represent the two attractors. This problem may be solved by restricting the set of basic transformations we use. More generally, we should try to find a class of languages such that there is a unique language associated with a given attractor.

References

- [Bar88] M.F. Barnsley. *Fractal Everywhere*. Academic press, INC, 1988.
- [BM89] J. Berstel and M. Morcrette. Compact representation of patterns by finite automata. In *Proceedings of Pixim*, pages 387–395, 1989.
- [BNA89] J. Berstel and A. Nait Abdallah. Tétrarbres engendrés par des automates finis. *Langages et algorithmes du graphique, Bigré n° 61-62*, Avril 1989.
- [CD92] K. Culik II and S. Dube. L-systems and mutually recursive function systems. To appear, 1992.
- [CD93a] K. Culik II and S. Dube. Balancing order and chaos in image generation. *Computer & Graphics*, 17(4):465–486, 1993.
- [CD93b] K. Culik II and S. Dube. Rationnal and affine expressions for image synthesis. *Discrete Appl. Math.*, 41:85–120, 1993.
- [Edg90] G.A. Edgar. *Measure, Topology, and Fractal Geometry*. Springer-Verlag, 1990.
- [Gen92] C. Gentil. *Les Fractales en Synthèse d'Images : le Modèle IFS*. PhD thesis, Université Claude Bernard LYON 1, France, 1992.
- [GM86] M. Gondran and M. Minoux. *Graphes et algorithmes*. Editions Eyrolles, 1986.
- [GT91] C. Gentil and E. Tosan. Descriptions of fractal with IFS. *Compugraphics*, 1991.
- [Har78] M.A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley Publishing Company, 1978.
- [Har92] J. Hart. The object instancing paradigm for linear fractal modeling. In *Proceedings of Graphics Interface'92*, May 1992.
- [MW88] R.D. Mauldin and L.C. Williams. Hausdorff dimension in graph directed constructions. *Trans. Amer. Math. Soc.*, 309:811–829, 1988.
- [PH91] P. Prusinkiewicz and M.S. Hammel. automata, languages and iterated function systems. In *lecture notes for the SIG-GRAPH'91 course : "fractal modeling in 3D computer graphics and imagery"*, 1991.
- [PH92] P. Prusinkiewicz and M.S. Hammel. Escape-time visualization method for language-restricted iterated function systems. In *Proceedings of Graphics Interface'92*, May 1992.
- [PJS92] H.O. Peilgen, H. Jurgens, and D. Saupe. Encoding images by simple transformations. In *Fractals For The Classroom*, New York, 1992.
- [Req80] A.A.G. Requicha. representations for rigid solids : theory, methods and systems. *ACM computing surveys*, 12(4), December 1980.
- [Rot82] Scott D. Roth. Ray casting for modelling solids. *computer graphics and image processing*, 18:109–144, 1982.
- [TT93] J. Thollot and E. Tosan. Construction of fractals using formal languages and matrices of attractors. In *Proceedings of Compugraphics'93*, December 1993.
- [TT94] J. Thollot and E. Tosan. Automata and constructive fractal geometry. Technical report, LISPI/LIGIA, Université Claude Bernard, Lyon, France, 1994.