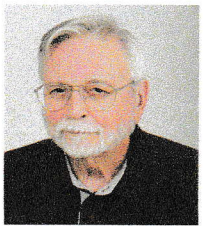
An Approach of an Information-Measure for Attractors from Multi-Reduction-Copy-Machines.

(15-7-2021).



Udo E. Steinemann,
Findeisen-Str,; 5/7,
71665 Vaihingen an der Enz,
Germany.
e-Mail: udo.steinemann@t-online.de .

A. Abstract.

C. E. SHANNON – with his theory of communication – was the first one who developed a concept for capturing information quantitatively. He found a measure for a kind of information relevant when rarity of events is decisive. His approach aimed on character–sequences of equal lengths and composed of elements from a symbol–set. Each symbol possesses its individual meaning and may appear differently often in a selected sequence. In such an environment it's possible to specify a probability of any symbol found in an arbitrary sequence. In the following a similar attempt is made in order to find an information–measure for attractors decoded or encoded by Multi–Reduction–Copy–Machines (MRCMs). But in contrast to SHANNON's theory where the approach is based on statistical considerations, the new measuring–method is guided by considering complexity of structures.

1 Introduction.

C. E. SHANNON by his mathematical theory of communication [1] initially formulated a measure for information strongly coupled to character–sequences. Sequences qualified by encoding of symbols with individual meanings and different frequencies of occurrence. Another approach shall now be attempted in an area of specific planar images decoded by and encoded through Multi–Reduction–Copy–Machines (MRCM's). Both approaches shall next be compared to each other.

1.1 A mathematical Theory of Communication.

Character–sequences of lengths $\langle \mathbf{N} \rangle$ are considered, each one consisting of numbers:

- N_1 of points and
- N_2 of strokes.

Thus for each sequence $\langle \mathbf{N = N_1+N_2} \rangle$ must be fulfilled. The question now comes up, what is the Information one will get, if such a sequence is obtained by transmission? Before such a message has been obtained, the uncertainty about what one will get is the greater, the longer the expected sequence is, and the gain of information will be the greater if the message finally has been received.

Following SHANNON's information–theory, the total number of sequences with length $\langle \mathbf{N} \rangle$ and composed of $\langle \mathbf{N_1} \rangle$ points and $\langle \mathbf{N_2} \rangle$ strokes can be calculated in the following way. Accordingly to the possibilities $\langle \mathbf{R} \rangle$ by which $\langle \mathbf{N_1} \rangle$ points and $\langle \mathbf{N_2} \rangle$ strokes can be distributed over a length of $\langle \mathbf{N} \rangle$ is:

1. $R = N!/(N_1!)\cdot(N_2!)$.

Before a message has been received, the information according to (**1.**) may be called $\langle \mathbf{I_0} \rangle$ while after reception it will change to $\langle \mathbf{I_1} \rangle$. Thus one gets:

- $[(I_0 = 0) \wedge (R_0 = 0)] \wedge [(I_1 \neq 0) \wedge (R_1 = 1)]$.

Now a measure has to be find, which will merge $\langle \mathbf{I} \rangle$ and $\langle \mathbf{R} \rangle$. Because of additivity for $\langle \mathbf{I} \rangle$ – in cases where e.g. two possible realisations $\langle \mathbf{R_1 \wedge R_2} \rangle$ exist – it is demanded:

- $[R = R_1 \cdot R_2] \wedge [I(R_1 \cdot R_2) = I(R_1)+I(R_2)$.

These conditions can be fulfilled by:

2. $I = \text{const}\cdot\ln\{R\}$.

$\langle$**1.**$\rangle$ and $\langle$**2.**$\rangle$ then will lead to:

3. $I = \text{const}\cdot(\ln\{N!\} - \ln\{N_1!\} - \ln\{N_2!\})$.

By using STIRLING's formula (which is allowed for $\mathbf{Q > 100}$):

- $\ln\{Q!\} = Q\cdot\ln\{Q\} - 1$.

$\langle$**3.**$\rangle$ can be further evaluated into:

- $I = \text{const}\cdot[N\cdot(\ln\{N\} - 1) - N_1\cdot(\ln\{N_1\} - 1) - N_2\cdot(\ln\{N_2\} - 1)] = \text{const}\cdot[N\cdot\ln\{N\} - N_1\cdot\ln\{N_1\} - N_2\cdot\ln\{N_2\}]$
- $i = I/N = -\text{const}\cdot[(N_1/N)\cdot\ln\{N_1/N\} + (N_2/N)\cdot\ln\{N_2/N\})]$.

For $\langle N = N_1+N_2+\ldots+ N_x \rangle$ this leads to:

4. $i = -\text{const}\cdot[(N_1/N)\cdot\ln\{N_1/N\} + (N_2/N)\cdot\ln\{N_2/N\} + \ldots + (N_x/N)\cdot\ln\{N_x/N\}]$.

From $\langle$**4.**$\rangle$and $\langle \mathbf{p_J = N_J/N} \rangle$ as the relative probability for a character from the symbol–set one may write:

5. $i = -\text{const}\cdot\sum_{a=(1\rightarrow x)}[(p_a)\cdot\ln\{p_a\}]$.

This can be interpreted as the probability to hit a character from the symbol–set. The probability is identical with the relative frequency of symbols to appear. Herewith it is important to note that information in current sense is not a rating of any kind it is only relevant in connection with the rarity of events.

1.2 Encoding Images by simple Transformations.

Cauliflower is just a 'mutant' of a SIERPINSKI–gasket in a similar sense as a fern is a relative of a KOCH–curve. In this context the question comes up, is there a framework in which a natural structure such as the picture of a cauliflower and an artificial structure such as a SIERPINSKI–gasket are just examples of one unifying approach. This can be answered with a clear yes (please look into [2], [3], [4], [5]).

In order to come closer to the problems, which arise when one tries to answer the above question finally, one should consider first to steps required to build the proper framework.

1.2.1 Multiple Reduction Copy Machine (MRCM).

One may start with the idea of a Multiple Reduction Copy Machine (MRCM). It takes an image as input. It has several independent lens–systems each one reduces the input–image and places it somewhere into the output–image. The assembly of all reduced copies in some pattern as final output of the MRCM.
The dials of the machine are:

- Number of lens–systems,
- Setting of reduction–factors for each lens–system individually,
- Configuration of lens–systems for the assembly of copies.

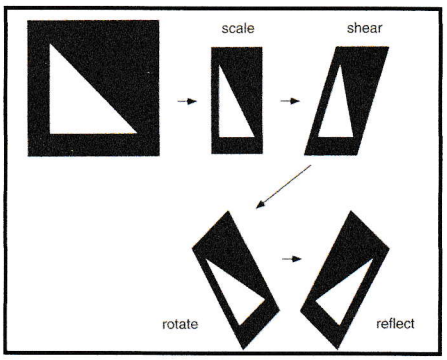
The crucial idea is that the machine runs in a feedback–loop, its own output is fed back as its new input again and again. Moreover, other transformations besides ordinary reductions are allowed. Running the machine once on an arbitrary image one copy reveals all geometric features of the machine in its so called blueprint.

No matter which initial image one takes and runs the MRCM with, one will obtain a sequence of images which always tends towards one and the same final image (the attractor of the machine). Moreover, the machine if started with its attractor, nothing will happen, the attractor keeps invariant under the process. The MRCM is like a bowl with one dish. If a little iron–ball is put in arbitrary initial position within the bowl and then released, it will always come to rest at the bottom of the bowl.

1.2.2 Composing Transformations for a MRCM.

Objects are similar if they have the same shape. Regardless of their size, corresponding angles must be equal and corresponding line–segments must all have the same factor of proportionality. The enlargement–factor between similar objects is called scaling–factor and a transformation between them is called similarity–transformation. In cases where copies of the whole appear at all scaling–stages and are exact and not distorted in any way, one speaks about self–similarity.

Contractions described by a lens–system may be similarity–transformations even if they reduce by different factors in different directions, but they must maintain angles unchanged, while more general contractions may not. In addition to pure similarity–transformations (which are scaling–operations), affine transformations like rotations, shear–operations and reflections are allowed as admissible operations for a MRCM–process.



The lens–systems of the MRCM can be described by affine linear transformations of the plane. Talking about a plane means here, a coordinate–system has to be defined with **x**– and **y**–axis perpendicular to each other.

A linear mapping is a transformation which associates with every point ⟨**P(x,y)**⟩ in the plane a point ⟨**F(P)**⟩ such that:

- $F(P_1+P_2) = F(P_1)+F(P_2)$,
- $F(sP) = sF(P)$ for any real number (s).

A linear transformation ⟨**F**⟩ can be represented with respect to the given coordinate–system by the following equation (with [**P = (x y)**] ∧ [**F(P) = (u v)**] ∧ [**F**← Matrix = **{a b}{c d}**]):

- $$\begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{Bmatrix} x \\ y \end{Bmatrix} \rightarrow \begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{Bmatrix} a\cdot x+b\cdot y \\ c\cdot x+d\cdot y \end{Bmatrix}.$$

This means a linear transformation is determined by ⟨**4**⟩ coefficient ⟨**a b c d**⟩. Affine linear transformations are simply a composition of a linear mapping and a translation ⟨**Q_1(u y)↔Q_2(u+p, y+q)**⟩:

- $$\begin{Bmatrix} u \\ v \end{Bmatrix} = \left[\begin{array}{cc|c} a & b & e \\ c & d & f \end{array}\right] \cdot \begin{Bmatrix} x \\ y \end{Bmatrix} \rightarrow \begin{Bmatrix} u \\ v \end{Bmatrix} = \begin{Bmatrix} a\cdot x+b\cdot y+e \\ c\cdot x+d\cdot y+f \end{Bmatrix}.$$

Already the first application of the MRCM to a given image will reveal its internal affine linear contractions in the blueprint of the MRCM.

1.2.3 Processing of a MRCM formulated by an Iterated Function System (IFS).

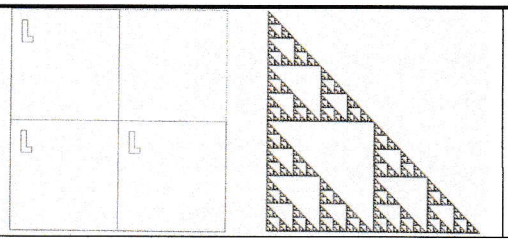
The feedback–mode of MRCM can be understood mathematically by an Iterated Function System ⟨**IFS**⟩, where the affine transformations ⟨$\omega_1\ \omega_2\ \ldots\ \omega_N$⟩ of the lens–systems acting upon the initial image ⟨**A**⟩ produce small copies of it. All these copies are overlaid into the new image [**W(A)**] as output:

1 $W(A) = \omega_1(A) \cup \omega_2(A) \cup \ldots \cup \omega_N(A)$.

The operator ⟨**W**⟩ is called the HUTCHINSON–operator [2] within the IFS–formalism. Running the MRCM in feedback–mode thus corresponds to iterating the operator ⟨**W**⟩. Starting with an initial image ⟨$\mathbf{A_0}$⟩ one gets:

- $A_1 = W(A_0) \rightarrow A_2 = W(A_1) \rightarrow \ldots \rightarrow A_N = W(A_{N-1}) \rightarrow \ldots$.

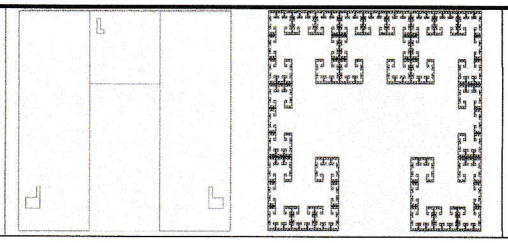
Already the first application ⟨$\mathbf{A_1 = W(A_0)}$⟩ will reveal a MRCM's internal contractions, called the blueprint of the machine. In the following for an initial image a unit–square ⟨**[0.1]×[0,1]**⟩ is used with an inscribed ⟨**L**⟩ in the top left corner to unfold the blueprint:



IFS with **3** similarity–transformations with scaling–factor ($^1/_2$).

	a	b	c	d	e	f
ω_1	($^1/_2$)	0	0	($^1/_2$)	0	0
ω_2	($^1/_2$)	0	0	($^1/_2$)	($^1/_2$)	0
ω_2	($^1/_2$)	0	0	($^1/_2$)	0	($^1/_2$)

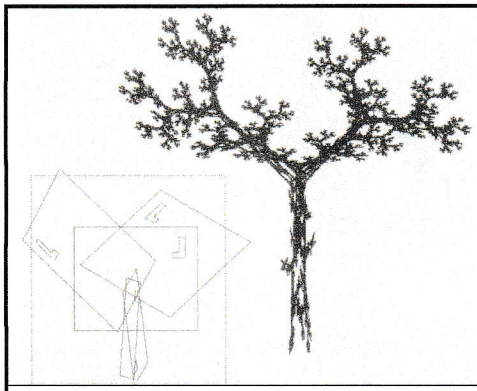
SIERPINSKI–Gasket



IFS with **3** similarity–transformations with scaling–factor ($^1/_2$).

	a	b	c	d	e	f
ω_1	($^1/_3$)	0	0	($^1/_3$)	($^1/_3$)	($^2/_3$)
ω_2	0	+($^1/_3$)	1	0	($^2/_3$)	0
ω_2	0	−($^1/_3$)	1	0	($^1/_3$)	0

CANTOR–Maze



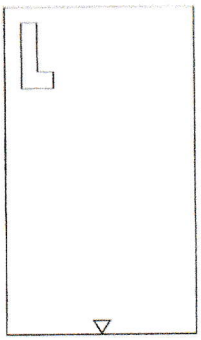
The attractor of IFS with five transformations can even resemble the image of a tree. The attractor is twice as large as the blueprint indicates.

	a	b	c	d	e	f
ω_1	.193	.488	.344	.443	.4431	.2452
ω_2	.382	.414	.252	.361	.2511	.5692
ω_3	.058	.070	.453	.111	.5976	.0969
ω_4	.035	.070	.469	.022	.4884	.5069
ω_5	.637	0	0	.501	.8562	.2513

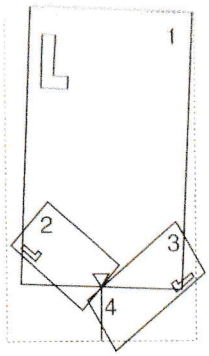
Underlined means negative value.

A Tree

The importance of last example in the recent table (the development of the image of a tree) is, this image looks very natural, but lies in the same mathematical category of constructions as the SIERPINSKI–gasket or the CANTOR–maze. In other words this category not only contains extreme mathematical monsters which seem very distant from nature, but also includes structures which are related to natural formations and which are obtained by only slight modifications of the monsters. Another example to be mentioned within context is BARNSLEY's –fern [3]:



Initial Image



Stage 1

	a	b	c	d	e	f
ω_1	.849	.037	.037	.849	.075	.183
ω_2	.197	.226	.226	.197	.400	.049
ω_3	.150	.283	.260	.237	.575	.084
ω_4	0	0	0	.160	.500	0

Unlined means negative value.

Note transformation (4) contracts a rectangle to a mere line–segment.
The attractor in mathematically precise sense is not self–similar.

BARNSLEY's Fern

Something as complicated and structured seems to have a lot of information content, but the figure above demonstrates, the information from IFS's point of view is extremely small. The apparent complexity of the picture is compressed into a very simple plan. This means many complex structures are relatively simple and compact when they are discussed from an IFS– or MRCM–point–of–view. This amazing conclusion may become appropriate when thinking about another measure of information now in the context of a specific category of planar images.

Without answering the questions:

- How images can be compared or
- What is the distance between two images

one will not be able to precisely verify the conditions under which an IFS will produce a limiting image. Here will help an approach from F. HAUSDORFF. He proposed a method for determining the distance which is now call the HAUSDORFF–distance $\langle \mathbf{h(A\ B)} \rangle$ between images $\langle \mathbf{A} \rangle$ and $\langle \mathbf{B} \rangle$, which has two consequences:

- One can talk about the sequence of images $\langle \mathbf{A_K} \rangle$having the limit $\langle \mathbf{A_\infty} \rangle$, if $\langle \mathbf{A_\infty} \rangle$ is the limit of the sequence $\langle \mathbf{A_0\ A_1 \ldots A_K \ldots A_\infty} \rangle$ with $\langle \mathbf{[K \to \infty] \Rightarrow [h(A_\infty\, A_K) \to 0]} \rangle$.

But even more important:

- HUTCHINSON showed afterwards that an operator $\langle \mathbf{W} \rangle$ which described a collage of images (please look at equation [1.]) contracts with respect to the HAUSDORFF–distance:
  $\langle \mathbf{[W(A) \leftrightarrow W(B)] \le \{[0 < c < 1] \cdot h(A,B)\}} \rangle$ for all compact set $\langle \mathbf{A} \rangle$ and $\langle \mathbf{B} \rangle$ in a plane.

In addition HUTCHINSON was able to inject into his consideration the contraction–mapping–principle finally

formulated by S. BANACH. This principle is nowadays a theorem in metric topology. In essence it says, in a space wherein sequences of contracted images will converge, where this behaviour of convergence crucially depends on the metric in the relevant space. Current discussions are mainly concentrated on Euclidean metric.

1.2.4 Decoding and Encoding are inverse Practises in Image-Processing via MRCMs.

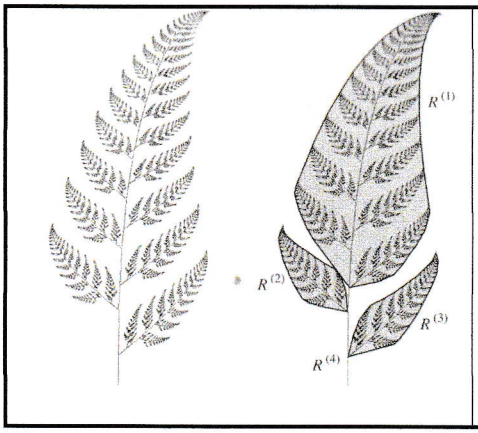
In order to make use of IFSs for image encoding, one first has to solve another crucial problem, namely to construct a suitable MRCM for a given image. Encoding is the inverse of decoding. Of course one cannot expect to be universally able to build an MFCM which produces any given image. However approximations should be possible as close to original as one desires.

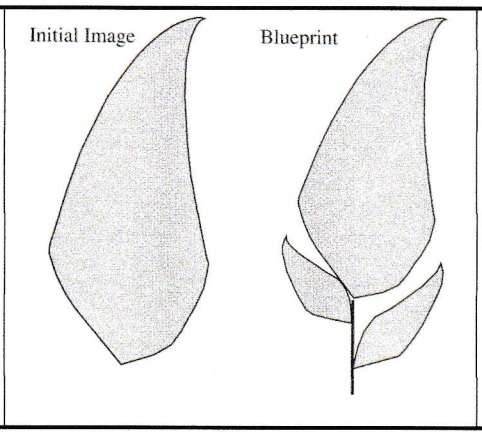
Assuming one has given a black and white picture, digitized at a resolution of $\langle \mathbf{n} \rangle$ by $\langle \mathbf{m} \rangle$ pixels. This image can be exactly reproduces by a MRCM simply by requiring that for every black pixel of the image, there exists a lens which contracts the whole image to that particular pixel. Running the machine just once, starting out with any image will produce the prescribed black and white pixel image. Naturally, this not an efficient way to code an image (for every black pixel one needs one affine transformation to store it), however it demonstrates that in principle it is possible to achieve approximations of any desired accuracy. Thus, the problem is to find ways to construct a better MRCM which does not need as many transformations, but still produces a good approximation. Several difficult questions are to be solved in this context:

- How can the quality of an approximation be assessed?
- How one can quantify differences between images?
- How one can identify suitable transformations?
- How one can minimize the necessary number of affine transformations?
- What is the appropriate class of images suitable for this approach?

Let's assume a given original image has been approximated by a MRCM. One may recall that the blueprint of a MRCM is already determined by the first copy it produces, this copy is a collage of transformed images. Applying the MRCM to the original image, called target–image, one also determines the quality of the approximation. When the copy is identical to the original then the corresponding IFS codes the target image perfectly. When the distance of the copy to the target is small, then one knows from the contraction–mapping–principle, that the attractor of the IFS (which is equal to the target image) is not far from the initial image.

These properties will one enable to find the code for a given target–image, in particular for target–images which contain apparent self–similarities such as the fern. With some practise it is relatively easy to identify portions of the picture which are affine copies of the whole; for instance in BARNSLEY's–fern:





The part $\langle \mathbf{R^{(1)}} \rangle$ is a slightly smaller and rotated copy of the whole fern. This leads to the numerical computation to $\langle \mathbf{1.} \rangle$ affine transformation $\langle \boldsymbol{\omega_1} \rangle$. The same procedure applies to the copies $\langle \mathbf{R^{(2)}} \wedge \mathbf{R^{(3)}} \rangle$in the figure. Even the bottom–part of the stem $\langle \mathbf{R^{(4)}} \rangle$ is a copy of the whole, however its transformation is degenerated by scaling–factor $\langle \mathbf{0} \rangle$ in one direction.
The resulting $\langle \mathbf{4} \rangle$ transformations cover the complete fern–system.

In general one needs a procedure to generate a set of transformations such that the union of the transformed target–images cover the target–image as closely as possible. By the example of the fern–leaf above it has been hinted at, how this works.

Creating an image with a MRCM quite naturally leads to a structure which has repetition to smaller and smaller scales. In cases, where each contraction involved in the corresponding IFS is a similarity with the same reduction–factors, the resulting attractor is called strictly self–similar. Even if different reduction–factors occur, the resulting attractor is said to be self–similar. When the contractions are not similarities, but affine linear transformations, the resulting attractor is called self–affine. But IFSs can also be used to approximate images that are not self–similar or self–affine. In these cases the approximations can be made as accurate as desired. However, the very small features of the corresponding attractor must reveal self–similar structures. The concept of IFSs can be generalized in such a way, that this restriction can be removed (additional references [6] and [7]).

1.2.5 Networking MRCMs for Generation of Images with broken Self-Similarities or Self-Affinities.

In pictures below two ferns are shown which almost look like the familiar BARNSLEY–fern , but they are different from it. Upon close examination of the two ferns, one will observe that the phyllotaxis has changed. The placement of the major leaves on the stem is different from that of the small leaves on the major ones. This means, that the major leaves are no longer scaled down copies of the entire fern. In other words, these ferns are neither self–similar nor self–affine. Nevertheless, one could say that they have some features of self–similarity. But what are these features and how are these particular ferns encoded? The answers to these questions will lead to networked MRCMs or hierarchical IFSs.

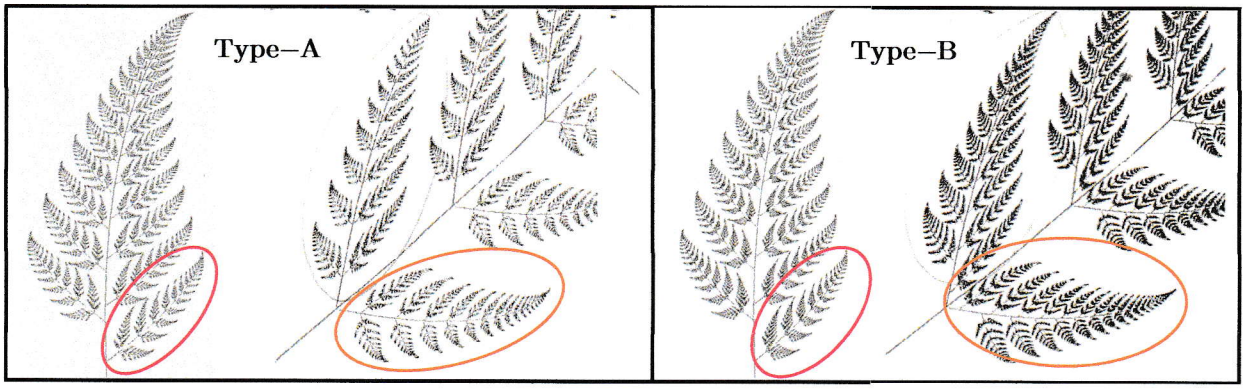
To see some of the hierarchical structures one may look at a blow–up of one of the major leaves from each of the ferns below. The placements of the sub–sub–leaves are different. For the (**Type–A**)– fern:

- Sub–leaves of all stages are always placed opposing each other,

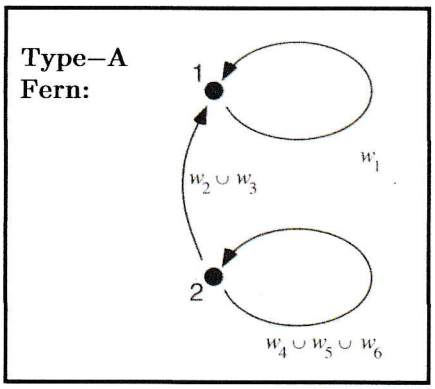
while on the (**Type–B**)–fern:

- This placement alternates from stage to stage. In one stage sub–leaves are placed opposing each other and in the next stage sub–leaves are placed with an offset.

This reveals the different hierarchies in their encodings. One has to realize now that self–similarity–structures are intermixed. Being prepared for further steps, one has to expand the concept of one MRCM to include several MRCMs operating in a network. How two non–self–similar ferns can be obtained by two network–MRCMs, will be demonstrated next. To keep things as simple as possible stems of the ferns are disregarded from further considerations.



The hierarchy of the (**Type–A**)–fern is considered first.
One will identify two basic structures, the entire fern and one of its major leaves. In this case the leaf is a self–affine structure, all sub–leaves are copies of the whole leaf and vice versa. The complete fern is made up of copies of this leaf, but it is not simply a copy of the leaf. This is due to the different placement of the leaf and sub–leaves. This is the crucial difference between BARNSLEY's self–affine fern and this one, where the self–affinity is broken. Due to this breaking of self–affinity the fern cannot be generated with an ordinary MRCM. However one may join two different machines to form a network–MRCM which will accomplish the task.
In this case the network–MRCM will be represented by the graph below:



One machine – named ⟨**2**⟩ in the graph above – is used to produce the main–leaf alone. This machine works as the one for BARNSLEY's–fern (disregarding the stem for simplicity reasons). Thus, it has three transformations $\langle \omega_4 \cup \omega_5 \cup \omega_6 \rangle$:

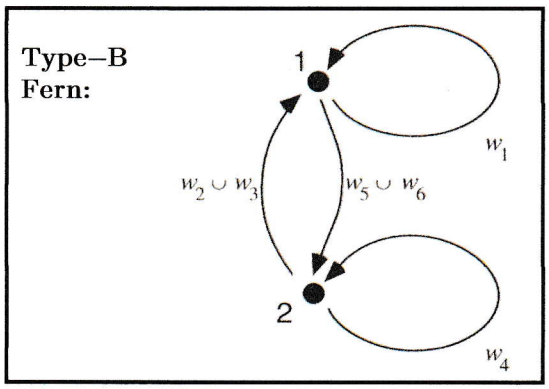
- ω_4: maps the entire leaf to the lower right sub–leaf,
- ω_5: maps in the upper left sub–leaf and
- ω_6: finally maps to all sub–leaves except for the two bottom sub–leaves which are already covered by the two other transformations.

The other machine – called ⟨**1**⟩ in the graph above – produces the whole fern. It has two inputs ($\omega_1 \cup [\omega_2 \cup \omega_3]$) and one output:

- ω_1: is served by its own output.
- $\omega_2 \cup \omega_3$: Is served by the machine (**2**).

Thus, there are three compound transformations in the machine. However each transformation is applied to only one particular input–image. The two transformations $\langle \omega_2 \cup \omega_3 \rangle$ operate on the results produced by MRCM ⟨**2**⟩, These produce the left and right bottom main–leaves at the proper places on the fern. The other transformation $\langle \omega_1 \rangle$ operates on the results from MRCM ⟨**1**⟩. The results of all transformations are merged when they are transferred to the output of MRCM ⟨**1**⟩. Transformation $\langle \omega_1 \rangle$ maps the entire fern to its upper part (i.e. the part without the two bottom–leaves). This was also the case in the plain MRCM for BARNSLEY's–fern. In this way the fern with the prescribed pattern for the leaf–placement from hierarchy ⟨**Type–A**⟩ will be generated.

In order to produce fern as mentioned for hierarchy ⟨**Type–B**⟩, one needs to go just a small step further, interconnecting the two MRCMs both ways:



This fern is characterized by the fact that:

- The entire fern reappears in the main–leaves as sub–leaves,
- while the main–leaves themselves are not copies of the entire fern.

The only change relative to the network for the hierarchy (**Type–A**)–fern is given by the extra–input in the appropriate MRCM ⟨**2**⟩. This input–image ensures (in the limit it is the entire fern) will be transformed to make the two lowest sub–leaves of the main–leaf.

To run these networks one just takes an initial image (like a square for instance) and puts it on the two MRCMs. The machines take these input–images following the connections of the input–lines and produce two outputs, one for the main–leaf and one for the fern. These outputs are now used as new inputs as indicated by the feedback–connections. During iterations one can observe how the leaf–MRCM creates the major lower right hand leaf and the fern–MRCM generates the complete fern.

In conclusion it turns out that the networking–machine has exactly one limit–image (its attractor) and this attractor is independent of the initial images. The networking–machines are encodings of non–self–similar ferns, and their hierarchies decipher the self–similarity features of these ferns. In fact, the hierarchy of the network deciphers the self–similarity of an entire class of attractors.

2. Complexity-Measures of Attractors coded by one or a Network of MRCM's.

The lens–systems of one MRCM are described by set of transformations $\langle \omega_1\ \omega_2\ .....\ \omega_N \rangle$. For a given initial image $\langle \mathbf{A} \rangle$ small affine copies $\langle \omega_1(\mathbf{A})\ \omega_2(\mathbf{A})\ .....\ \omega_N(\mathbf{A}) \rangle$ are produced. Finally the machine overlaps all these copies into one new image, the output $\langle \mathbf{W}(\mathbf{A}) \rangle$ of the machine:

1. $W(A) = \omega_1(A) \cup \omega_2(A) \cup .....\cup \omega_N(A)$.

The output $\langle \mathbf{W}(\mathbf{A}) \rangle$ is obtained by acting the HUTCHINSON–operator $\langle \mathbf{W} \rangle$ of the MRCM on the image $\langle \mathbf{A} \rangle$. Thus running the MRCM in feedback–mode corresponds to an iterative application of the operator $\langle \mathbf{W} \rangle$. This is in essence what is meant by a deterministic iterated function system $\langle \mathbf{IFS} \rangle$. The first application of the IFS to a give initial image $\langle \mathbf{A_0} \rangle$ will result in the image $\langle \mathbf{A_1} \rangle$, called the blueprint of the MRCM:

2. $\mathbf{A_1} = W(\mathbf{A_0}) = \omega_1(\mathbf{A_0}) \cup \omega_2(\mathbf{A_0}) \cup .....\cup \omega_N(\mathbf{A_0})$.

Already the production of its blueprint will reveal the full complexity of an attractor finally coded by the complete IFS.

In addition to the provisional results $\langle \mathbf{1.} \rangle$ and $\langle \mathbf{2.} \rangle$ above (valid for a **1**–MRCM–system so far), further aspects will come to the fore. For instance the message of the two ferns from $\langle \mathbf{1.2.5} \rangle$ is very impressive. Representations for these pictures – structured in such a relatively complicated way – must obviously have a higher level of complexity as one will expect for any picture represented by a **1**–MRCM. Thus by subsequent considerations an approach must be found, where the whole gallery of attractors from above mentioned iterative reduction–processes can be ordered in one and the same complexity–scheme.

There is an extension of the concept of a HUTCHINSON–operator for a network of MRCMs. A hierarchical HUTCHING–operator (corresponding to a network of $\langle \mathbf{M} \rangle$ MRCMs) is given by a $(\mathbf{M}\times\mathbf{M})$–matrix:

3. $$W = \begin{pmatrix} w_{11} & w_{12} & \rightarrow & w_{1J} & \rightarrow & w_{1M} \\ \downarrow & \downarrow & \rightarrow & \downarrow & \rightarrow & \downarrow \\ w_{M1} & w_{M2} & \rightarrow & w_{MJ} & \rightarrow & w_{MM} \end{pmatrix}.$$

$\langle \mathbf{W} \rangle$is now a matrix with a finite number of linear transformations $\langle \mathbf{w}_{K=[1\rightarrow M]\hat{}L=[1\rightarrow M]} \rangle$. For some transformations $\langle \mathbf{w}_{K\hat{}L}(\varnothing) \rangle$ may be allowed, in which case it will be transformed to an empty set only. The operator $\langle \mathbf{W} \rangle$ acts on a **M**–vector $\langle \mathbf{V} \rangle$ of input–images:

- $V = \{v_1 \cup v_2 \cup ..... \cup v_M\}$

and generates a **M**–vector $\langle \mathbf{U} \rangle$ of output–images:

- $U = \{u_1 \cup u_2 \cup ..... \cup u_M\} = {}_{(K=1)}\bigcup^{(K=M)}[w_{J\hat{}K}(v_K)]$.

A network of MRCMs corresponds to a graph of nodes and directed edges (similar to those from $\langle \mathbf{1.2.5} \rangle$). Exactly one node exists for the output of each MRCM in the network and one directed edge for any output–input connection in the network.

- An edge directed from node $\langle \mathbf{J} \rangle$ to node $\langle \mathbf{K} \rangle$ symbolizes – after HUTCHINSON–operator having produced output for input (**J**) – this output is fed into $\langle \mathbf{K} \rangle$.

In both cases ($\langle \mathbf{1} \rangle$ MRCM or a network of MRCMs) it's easy to get a close idea of any attractor's complexity during its blueprint–processing. This means, in any case one has to measure how far image $(\mathbf{A_1})$ has removed from the initial image $(\mathbf{A_0})$ under one action of the appropriate HUTCHINSON–operator.

2.1 Complexity-Measuring for Attractors coded by one MRCM.

Any step $\langle \mathbf{N} \rangle$ in the feedback–processing of IFS starts with an initial image $\langle \mathbf{A_N} \rangle$, which then is acted on by the HUTCHINSON–operator $\mathbf{W}(\mathbf{A_N})$ in order to form the output–image $\langle \mathbf{A_{N+1}} \rangle$ of the step:

- $\mathbf{A_{N+1}} = \mathbf{W}({}_{J=1}\bigcup^{J=N}\omega_J(\mathbf{A_N}))$ $\Leftarrow$ $\omega_J = \left(\begin{array}{cc|c} a_J & b_J & e_J \\ & & \\ c_J & d_J & f_J \end{array}\right)$.

In the following the feedback–process is limited to produce the blueprint only.

Thus:

- $\mathbf{A_1} = W(\mathbf{A_0})$.

$\langle \mathbf{A_0} \rangle$ is reduced to one of its points only:

- $\mathbf{A_0} \leftarrow (x\boldsymbol{i}+y\boldsymbol{j}) \quad \Leftarrow \quad \{\langle [\boldsymbol{i}^2 = \boldsymbol{j}^2 = \boldsymbol{i}\times\boldsymbol{j} = \langle -\boldsymbol{j}\times\boldsymbol{i}\rangle = 1] \wedge [\boldsymbol{i}\cdot\boldsymbol{j} = 0]\rangle \bigwedge \langle x \wedge y \leq 1\rangle\}$.

Thus:

- $\mathbf{A_1} = W(x\boldsymbol{i}+y\boldsymbol{j}) = {}_{J=1}\bigcup^{J=N} \omega_J\,(x\boldsymbol{i}+y\boldsymbol{j})$.

Due to the fact that $\langle \mathbf{A_0} \rangle$ consists of one point only, $\langle \mathbf{A_1} \rangle$ will become the collection of transformed points $\langle [\omega_1 \cdot \mathbf{A_0} = \mathbf{A_{11}}] \cup [\omega_2 \cdot \mathbf{A_0} = \mathbf{A_{12}}] \cup \ldots\ldots \cup [\omega_N \cdot \mathbf{A_0} = \mathbf{A_{1N}}] \rangle$:

1. $$\mathbf{A_1} = {}_{J=1}\bigcup^{J=N} \left\{ \left\{ \begin{matrix} u_J = (x\cdot a_J + e_J) + (y \cdot b_J) \\ v_J = (x\cdot c_J) + (y \cdot d_J + f_J) \end{matrix} \right\} = \left(\omega_J = \begin{pmatrix} (a_J + x^{-1}\cdot e_J) & b_J \\ c_J & (d_J + y^{-1}\cdot f_J) \end{pmatrix} \right) \cdot \left\{ \left\{ \begin{matrix} x \\ y \end{matrix} \right\} = \mathbf{A_0} \right\} \right\} .$$

By running from $\langle \mathbf{A_0} \rangle$ to $(\mathbf{A_1})$ the cardinality $\langle \#(\mathbf{A_1}) \rangle$ of set $\langle \mathbf{A_1} \rangle$ increases and with it the complexity of $\langle \mathbf{A_1} \rangle$ relative to $\langle \mathbf{A_0} \rangle$. Therefore the primarily part of complexity of the blueprint–print can be given by:

2. $$\Gamma = [\#(\mathbf{A_0})] + [\#(\mathbf{A_1})] = 1 + {}_{J=1}\bigcup^{J=N} \left\{ \begin{matrix} (x\cdot a_J + e_J) + (y \cdot b_J) \\ (x\cdot c_J) + (y \cdot d_J + f_J) \end{matrix} \right\} .$$

From $\langle \mathbf{1.} \rangle$ and by setting $\langle x = y = 1 \rangle$ simultaneously, one may derive a futher expression:

- $$\Delta^2 = {}_{J=1}\sum^{J=N} \left(\left\{ \begin{matrix} a_J + e_J + b_J \\ c_J + d_J + f_J \end{matrix} \right\} \cdot \left\{ \begin{matrix} a_J + e_J + b_J \\ c_J + d_J + f_J \end{matrix} \right\} \right)$$

3. $$+\Delta = {}_{J=1}\sum^{J=N} \left[(a_J + e_J + b_J)^2 + (c_J + d_J + f_J)^2 \right]^{1/2}.$$

$\langle \mathbf{3.} \rangle$ offers a further part of blueprint–complexity, which reflects the complete influence of the MRCM's linear–affine transformations (contractions, rotations, reflections and shear–operations). Therefore by pairs $\langle \Gamma \hat{} \Delta \rangle$ one has given a tool at hand, which one enables to order blueprints from appropriate MRCM's according to their complexities.

2.2 Complexity-Measure for Attractors coded from Network-MRCMs.

How will a network of MRCMs operate during the feedback–process? Any directed edge $\langle \mathbf{J} \rightarrow \mathbf{K} \rangle$ between the nodes $\langle \mathbf{J} \in \mathbf{M} \rangle$ and $\langle \mathbf{K} \in \mathbf{M} \rangle$ will change image $\langle \mathbf{A^0}_J \rangle$ by acting transformation $\langle \mathbf{w}_{KJ} \rangle$ on it and the result is feed into $\langle \mathbf{A^1}_K \rangle$:

1. $\mathbf{A^1}_K \leftarrow w_{KJ} \cdot \mathbf{A^0}_J \quad \Rightarrow \quad \mathbf{A^1}_K = {}_{[L=1]}\bigcup^{[L=N(K)]} (w_{KL} \cdot \mathbf{A^0}_L)$.

Limitation of index $\langle \mathbf{L} \rangle$ in $\langle \mathbf{1.} \rangle$ is due to a possibility of:

- $\exists [\{w_{KV}\} \subset \{w_{KL}\}] \rightarrow \{w_{KV} \cdot \mathbf{A^0}_L = \varnothing\}$.

The transformations $\langle \mathbf{w}_{KJ} \rangle$ themselfes are composed out of sets of linear–affine transformations:

2. $w_{KJ} = {}_{[H=1]}\bigcup^{[H=N(KJ)]} \{\omega_{KJH}\}$

Therefore from $\langle \mathbf{1.} \rangle$ and $\langle \mathbf{2.} \rangle$ one will further obtain:

3. $\mathbf{A^1}_K = {}_{[L=1]}\bigcup^{[L=N(K)]} \{ {}_{[H=1]}\bigcup^{[H=N(KL)]} \{\omega_{KLH}\} \cdot \mathbf{A^0}_L \}$.

Normally the unit–square is selected for initial image $\langle \mathbf{A^0}_L \rangle$, but – for subsequent discussion – any unit–square is

reduced to one of its points:

4. $\mathbf{A}^0{}_L = (x_L\boldsymbol{i}+y_L\boldsymbol{j}) \quad \Leftarrow \quad \{\langle[\boldsymbol{i}^2=\boldsymbol{j}^2=\boldsymbol{i}\times\boldsymbol{j}=\langle-\boldsymbol{j}\times\boldsymbol{i}\rangle=1]\wedge[\boldsymbol{i}\bullet\boldsymbol{j}=0]\rangle \bigwedge \langle[0<x_L\leq 1]\wedge[0<y_L\leq 1]\rangle\}.$

Thus starting blueprint–processing from:

5. Point–set $\mathbf{A}^0 = \{{}_{[L=1]}\bigcup^{[L=M]}(x_L\boldsymbol{i}+y_L\boldsymbol{j})\}$
with cardinality
$\Theta = \#\{{}_{[L=1]}\bigcup^{[L=M]}(x_L\boldsymbol{i}+y_L\boldsymbol{j})\}$,

by heeding of ⟨**3.**⟩, one easy realizes that:

- Point–set $\underline{\mathbf{A}}^1{}_K = \{{}_{[L=1]}\bigcup^{[L=N(K)]}({}_{[H=1]}\bigcup^{[H=N(KL)]}\omega_H)\bullet\{x_L\boldsymbol{i}+y_L\boldsymbol{j}\})\}$
has been enlarged against ⟨Θ⟩ to a cardinality
$\Xi = \#\{{}_{[L=1]}\bigcup^{[L=N(K)]}({}_{[H=1]}\bigcup^{[H=N(KL)]}\omega_H)\bullet\{x_L\boldsymbol{i}+y_L\boldsymbol{j}\})\}$ and

6. Point–set $\mathbf{A}^1 = \{{}_{[U=1]}\bigcup^{[U=M]}({}_{[L=1]}\bigcup^{[L=N(U)]}({}_{[H=1]}\bigcup^{[H=N(UL)]}\omega_{UL\,H}\})\bullet\{x_L\boldsymbol{i}+y_L\boldsymbol{j}\})\}$
will get against ⟨Θ⟩ a cardinality of
$\Omega = \#\{{}_{[U=1]}\bigcup^{[U=M]}({}_{[L=1]}\bigcup^{[L=N(U)]}({}_{[H=1]}\bigcup^{[H=N(UL)]}\omega_{UL\,H})\bullet\{x_L\boldsymbol{i}+y_L\boldsymbol{j}\})\}$.

Therefore by ⟨**Ω**⟩ from ⟨**6.**⟩ and ⟨**Θ**⟩ from ⟨**5.**⟩ the preliminary part ⟨**Γ**⟩ of blueprint–complexity leads to:

7. $(\Omega+\Theta)=\Gamma$.

To get of blueprint–complexity on a more advanced level, another characteristics ⟨**Δ**⟩ has additionally to be taken into consideration. To get familiar with ⟨**Δ**⟩ it's useful to go back to ⟨**3.**⟩:

- $\underline{\mathbf{A}}^1{}_K = \{{}_{[L=1]}\bigcup^{[L=N(K)]}({}_{[H=1]}\bigcup^{[H=N(KL)]}\omega_{KLH})\bullet\underline{\mathbf{A}}^0{}_L\} \quad \Rightarrow \quad \mathbf{A}^1{}_K = \{{}_{[L=1]}\bigcup^{[L=N(K)]}({}_{[H=1]}\bigcup^{[H=N(KL)]}\omega_{KLH})\bullet\{x_L\boldsymbol{i}+y_L\boldsymbol{j}\}\}$.

Herein the linear affine transformation ⟨ω_{KLH}⟩ is given by:

- $\omega_{KLH} = \left(\begin{array}{cc|c} a_{KLH} & b_{KLH} & e_{KLH} \\ c_{KLH} & d_{KLH} & f_{KLH} \end{array}\right)$

This will further lead to:

- $\omega_{KLH}\bullet\,\mathbf{A}^0{}_L = \begin{pmatrix} a_{KLH}+x_L^{-1}\bullet e_{KLH} & b_{KLH} \\ c_{KLH} & d_{KLH}+y_L^{-1}\bullet f_{KLH} \end{pmatrix}\cdot\left\{\begin{Bmatrix} x_L \\ y_L \end{Bmatrix}=\mathbf{A}^0{}_L\right\}=\left\{\begin{matrix} (x_La_{KLH}+e_{KLH})+y_Lb_{KLH} \\ x_Lc_{KLH}+(y_Ld_{KLH}+f_{KLH}) \end{matrix}\right\}.$

Thus it can further be obtained from ⟨**3.**⟩:

- $\underline{\mathbf{A}}^1{}_K = \left({}_{[L=1]}\bigcup^{[L=N(K)]}\left({}_{[H=1]}\bigcup^{[H=N(KL)}\left\{\begin{matrix} (x_La_{KLH}+e_{KLH})+y_Lb_{KLH} \\ x_Lc_{KLH}+(y_Ld_{KLH}+f_{KLH}) \end{matrix}\right\}\right)\right),$

this finally will lead to:

8. $\mathbf{A}^1 = \left({}_{[U=1]}\bigcup^{[U=M]}\left({}_{[L=1]}\bigcup^{[L=N(U)]}\left({}_{[H=1]}\bigcup^{[H=N(UL)]}\left\{\begin{matrix} (x_La_{KLH}+e_{KLH})+y_Lb_{KLH} \\ x_Lc_{KLH}+(y_Ld_{KLH}+f_{KLH}) \end{matrix}\right\}\right)\right)\right).$

If condition in ⟨**4.**⟩ becomes generally restricted to ⟨$x_L=1=y_L$⟩, one may derive from ⟨**8.**⟩:

- $\Delta^2 = {}_{[U=1]}\sum^{[U=M]}\left({}_{[L=1]}\sum^{[L=N(U)]}\left({}_{[H=1]}\sum^{[H=N(UL)]}\left\{\begin{matrix} a_{KLH}+e_{KLH}+b_{KLH} \\ c_{KLH}+y_Ld_{KLH}+f_{KLH} \end{matrix}\right\}\cdot\left\{\begin{matrix} a_{KLH}+e_{KLH}+b_{KLH} \\ c_{KLH}+d_{KLH}+f_{KLH} \end{matrix}\right\}\right)\right)$

9. $+\Delta = {}_{[U=1]}\sum^{[U=M]}\left[{}_{[L=1]}\sum^{[L=N(U)]}\left[{}_{[H=1]}\sum^{[H=N(UL)]}\left[(a_{KLH}+e_{KLH}+b_{KLH})^2 + (c_{KLH}+d_{KLH}+f_{KLH})^2\right]^{1/2}\right]\right].$

⟨**9.**⟩ offers a part in blueprint–complexity, which reflects the complete influence of a network's linear–affine transformations (contractions, rotations, reflections and shear–operations) and thus will contribute to the complexity of its blueprint. Therefore by pairs ⟨**Γ^Δ**⟩ one has giveb a tool at hand, which enables to order blueprint–complexities of appropriate network–MRCMs.

2.3 Examples of Complexity-Measures for BARNSLEY-, (Type-A)- and (Type-B)-Fern .

Due to ⟨**2.1^2.**⟩ and according with the appropriate values for ⟨$\omega_1\ \omega_2\ \omega_3\ \omega_4$⟩ to be found in the facts–table of BARNSLEY's–fern in ⟨**1.2.3**⟩, the appropriate ⟨Γ_{Barnsley}⟩ can be derived as follows:

- $$\Omega_{\text{Barnsley}} = \#\left[\begin{Bmatrix} (x\bullet.849+.075)+(y\bullet.037) \\ (x\bullet.037)+(y\bullet.849+.183) \end{Bmatrix} \cup \begin{Bmatrix} (x\bullet.197+.400)+(y\bullet.226) \\ (x\bullet.226)+(y\bullet.197+.049) \end{Bmatrix} \cup \begin{Bmatrix} (x\bullet.150+.575)+(y\bullet.283) \\ (x\bullet.283)+(y\bullet.237+.084) \end{Bmatrix} \cup \begin{Bmatrix} .500 \\ y\bullet.160 \end{Bmatrix} \right].$$

1. $\Gamma_{\text{Barnsley}} = 5 = 4+1$.

Accordingly to ⟨**2.1^3.**⟩ one will further get for ⟨Δ_{Barnsley}⟩:

- $$\Delta^2_{\text{Barnsley}} = \begin{pmatrix} [(.849+.075+.037)^2+(.037+.849+.183)^2] & + \\ (.197+.4+.226)^2+(.226+.197+.049)^2 & + \\ (.15+.575+.283)^2+(.26+.237+.084)^2 & + \\ .5^2+.16^2 & \end{pmatrix} = \begin{pmatrix} (0.924+1.143) & + \\ (0.677+0.223) & + \\ (1.016+0.338) & + \\ (0.25+0.026) & \end{pmatrix} = \begin{pmatrix} 2.067 & + \\ 0.9 & + \\ 1.354 & + \\ 0.276 & \end{pmatrix}$$
  $$= 4.597$$
- $+\Delta_{\text{Barnsley}} = 2.144$.

Thus the complexity–measure for the blueprint of **BARNSLEY**'s–fern will come up with:

2. $\langle\Gamma\hat{}\Delta\rangle_{\text{Barnsley}} = \langle 5\hat{}4.597\rangle$.

According to the graph of (**Type–A**)–fern in ⟨**1.2.5**⟩ and with ⟨**2^3.**⟩, a HUTCHINSON–operator is obtained:

- $$W = \begin{pmatrix} w_{11}=\omega_1 & w_{12}=(\omega_2\cup\omega_3) \\ 0 & w_{22}=(\omega_4\cup\omega_5\cup\omega_6) \end{pmatrix}.$$

Based on⟨**W**⟩ and with reference to ⟨**2.2^3.**⟩ set ⟨$\mathbf{A^0}$⟩ will enable a set ⟨$\mathbf{A^1}$⟩ for the blueprint of (**Type–A**)–fern:

- $$\mathbf{A^1}_{\text{Type–A}} = \left\{ \begin{array}{l} \mathbf{A^1}_1 = \left\{ \left(w_{11} = \left(\omega_1 = \begin{pmatrix} a_1+x_1^{-1}\bullet e_1 & b_1 \\ c_1 & d_1+y_1^{-1}\bullet f_1 \end{pmatrix} \right) \right) \cdot \begin{Bmatrix} x_1 \\ y_1 \end{Bmatrix} = \mathbf{A^0}_1 \right\} \\ \qquad \cup \\ \mathbf{A^1}_2 = \left\{ \left(w_{12} = \left(\omega_2 = \begin{pmatrix} a_2+x_2^{-1}\bullet e_2 & b_2 \\ c_2 & d_2+y_2^{-1}\bullet f_2 \end{pmatrix} \right) \cup \left(\omega_3 = \begin{pmatrix} a_3+x_2^{-1}\bullet e_3 & b_2 \\ c_3 & d_3+y_2^{-1}\bullet f_3 \end{pmatrix} \right) \right) \cdot \begin{Bmatrix} x_2 \\ y_2 \end{Bmatrix} = \mathbf{A^0}_2 \right\} \\ \qquad \cup \\ \mathbf{A^1}_2 = \left\{ \left(w_{22} = \left(\omega_4 = \begin{pmatrix} a_4+x_2^{-1}\bullet e_4 & b_4 \\ c_4 & d_4+y_2^{-1}\bullet f_4 \end{pmatrix} \right) \cup \left(\omega_5 = \begin{pmatrix} a_5+x_2^{-1}\bullet e_5 & b_5 \\ c_5 & d_5+y_2^{-1}\bullet f_5 \end{pmatrix} \right) \cup \left(\omega_6 = \begin{pmatrix} a_6+x_2^{-1}\bullet e_6 & b_6 \\ c_6 & d_6+y_2^{-1}\bullet f_6 \end{pmatrix} \right) \right) \cdot \begin{Bmatrix} x_2 \\ y_2 \end{Bmatrix} = \mathbf{A^0}_2 \right\} \end{array} \right\}.$$

Cardinality ⟨$\#(\mathbf{A^1}_{\text{Type–A–fern}})$⟩ shows the increase in complexity of set ⟨$\mathbf{A^1}_{\text{Type–A–fern}}$⟩ in relation to the initial set ⟨$\mathbf{A^0}_{\text{Type–A–fern}}$⟩, means cardinality increases by ⟨$\Omega_{\text{Type–A–fern}}$⟩ in blueprint–processing. Taking the complexity of

$\langle \mathbf{A^0}_{\text{Type-A-fern}} \rangle$ in form of its cardinality $\langle \Theta_{\text{Type-A-fern}} \rangle$ into account, one will get due to $\langle$**2.2^7.**$\rangle$:

- $\Gamma_{\text{Type-A}} = (\Omega_{\text{Type-A}} + \Theta_{\text{Type-A}}) = 6+3 = 9$.

According to $\langle$**2.2^9.**$\rangle$ one will further obtain from $\langle$**4.**$\rangle$:

- $$\Delta_{\text{Type-A}} = \left(\left(\begin{array}{l} (a_1+e_1+b_1)^2+(c_1+d_1+f_1)^2 \\ + \\ \left(\begin{array}{l} (a_2+e_2+b_2)^2+(c_2+d_2+f_2)^2 \\ + \\ (a_3+e_3+b_3)^2+(c_3+d_3+f_3)^2 \end{array} \right) \end{array} \right) + \left(\begin{array}{l} (a_4+e_4+b_4)^2+(c_4+d_4+f_4)^2 \\ + \\ (a_5+e_5+b_5)^2+(c_5+d_5+f_5)^2 \\ + \\ (a_6+e_6+b_6)^2+(c_6+d_6+f_6)^2 \end{array} \right) \right)^{(1/2)} .$$

The final complexity–measure of the appropriate blueprint is represented by:

3. $\langle \Gamma \hat{} \Delta \rangle_{\text{Type-A}} = \langle 9 \hat{} (_{L=1}\sum^{L=6} [(a_L+e_L+b_L)^2+(c_L+d_L+f_L)^2])^{½} \rangle$.

From the graph of (**Type–B**)–fern in $\langle$**1.2.5**$\rangle$ a HUTCHINSON–operator can be derived in the form:

- $$W = \begin{pmatrix} w_{11} = \omega_1 & w_{12} = (\omega_2 \cup \omega_3) \\ w_{21} = (\omega_5 \cup \omega_6) & w_{22} = \omega_4 \end{pmatrix}$$

Due to the above $\langle \mathbf{W} \rangle$ and with reference to $\langle$**2.2^3.**$\rangle$ the image–set $\langle \mathbf{A^0} \rangle$ will enable an appropriate point–set $\langle \mathbf{A^1} \rangle$ of the blueprint for the (**Type–B**)–fern as shown below:

- $$\mathbf{A^1}_{\text{Type-B}} = \left\{ \begin{array}{l} \mathbf{A^1_1} = \left\{ \left(\begin{array}{l} w_{11} = \left(\omega_1 = \begin{pmatrix} a_1+x_1^{-1} \bullet e_1 & b_1 \\ c_1 & d_1+y_1^{-1} \bullet f_1 \end{pmatrix} \right) \\ \cup \\ w_{12} = \left(\begin{array}{l} \left(\omega_2 = \begin{pmatrix} a_2+x_2^{-1} \bullet e_2 & b_2 \\ c_2 & d_2+y_2^{-1} \bullet f_2 \end{pmatrix} \right) \\ \cup \\ \left(\omega_3 = \begin{pmatrix} a_3+x_2^{-1} \bullet e_3 & b_2 \\ c_3 & d_3+y_2^{-1} \bullet f_3 \end{pmatrix} \right) \end{array} \right) \end{array} \right) \bullet \left\{ \begin{matrix} x_1 \\ y_1 \end{matrix} = \mathbf{A^0_1} \right\} \right\} \\ \cup \\ \mathbf{A^1_2} = \left\{ \left(\begin{array}{l} w_{21} = \left(\begin{array}{l} \left(\omega_5 = \begin{pmatrix} a_5+x_1^{-1} \bullet e_5 & b_5 \\ c_5 & d_5+y_1^{-1} \bullet f_5 \end{pmatrix} \right) \\ \cup \\ \left(\omega_6 = \begin{pmatrix} a_6+x_1^{-1} \bullet e_6 & b_6 \\ c_6 & d_6+y_1^{-1} \bullet f_6 \end{pmatrix} \right) \end{array} \right) \\ \cup \\ w_{22} = \left(\omega_4 = \begin{pmatrix} a_4+x_2^{-1} \bullet e_4 & b_4 \\ c_4 & d_4+y_2^{-1} \bullet f_4 \end{pmatrix} \right) \end{array} \right) \bullet \left\{ \begin{matrix} x_2 \\ y_2 \end{matrix} = \mathbf{A^0_2} \right\} \right\} \end{array} \right\} .$$

Cardinality $\langle \#(\mathbf{A^1}_{\text{Type-B-fern}}) \rangle$ describes the complexity–increase of image–point–set $\langle \mathbf{A^1}_{\text{Type-B-fern}} \rangle$ in relation to the initial image–point–set $\langle \mathbf{A^0}_{\text{Type-B-fern}} \rangle$ during the creation of blueprint, this means an increase $\langle \Omega_{\text{Type-B-fern}} \rangle$ of complexity in blueprint–processing. Taking a complexity of $\langle \mathbf{A^0}_{\text{Type-B-fern}} \rangle$ in form of its cardinality $\langle \Theta_{\text{Type-B-fern}} \rangle$ into consideration, then – due to $\langle$**2.2^7.**$\rangle$ – one will get:

- $\Gamma_{\text{Type-B-fern}} = (\Omega_{\text{Type-B-fern}} + \Theta_{\text{Type-B-fern}}) = 6+4 = 10$.

According to ⟨**2.2^9.**⟩ one will further obtain from ⟨**4.**⟩:

- $$\Delta_{\text{Type-B-fern}} = \left(\left(\begin{array}{l} (a_1+e_1+b_1)^2+(c_1+d_1+f_1)^2 \\ \left(\begin{array}{l} (a_2+e_2+b_2)^2+(c_2+d_2+f_2)^2 \\ + \\ (a_3+e_3+b_3)^2+(c_3+d_3+f_3)^2 \end{array} \right) \end{array} \right) + \left(\left(\begin{array}{l} (a_5+e_5+b_5)^2+(c_5+d_5+f_5)^2 \\ + \\ (a_6+e_6+b_6)^2+(c_6+d_6+f_6)^2 \end{array} \right) + (a_4+e_4+b_4)^2+(c_4+d_4+f_4)^2 \right)^{(1/2)}.$$

The final complexity–measure of the appropriate blueprint is represented by:

4. $\langle \Gamma\hat{}\Delta \rangle_{\text{Type-B}} = \langle 10\hat{}({}_{L=1}\sum^{L=6}[\,(a_L+e_L+b_L)^2+(c_L+d_L+f_L)^2])^{½} \rangle$.

3 Approach for an Information-Measure for MRCM-Attractors.

Considering $\langle(\Gamma\hat{}\Delta)_{\text{Barnsley}}\rangle$, $\langle(\Gamma\hat{}\Delta)_{\text{Type-A}}\rangle$ and $\langle(\Gamma\hat{}\Delta)_{\text{Type-B}}\rangle$ from ⟨**2.3^2.**⟩, ⟨**2.3^3.**⟩ and ⟨**2.3^4.**⟩ respectively, where the position of each element in a pair decides about its priority (its importance for the specification), one can order the complexity–measures in the following way:

- $\langle 5\hat{}4.597 \rangle \prec \langle 9\hat{}{}_{L=1}\sum^{L=5}[\,(a_L+e_L+b_L)^2+(c_L+d_L+f_L)^2]^{1/2} \rangle \prec \langle 10\hat{}{}_{L=1}\sum^{L=6}[\,(a_L+e_L+b_L)^2+(c_L+d_L+f_L)^2]^{1/2} \rangle$.

This offers a possibility to order attractors – coded from single MRCMs or MRCM–networks – via the appropriate blueprints in a sequence of increasing complexity. Moreover, this will also give one a measure at hand (on account of such a complexity–specification) for the information gained by a specific attractor relative to others.

This concept of information–measure may be referred to complexities of planar structures only. It is to be understood in a similar way as information–measure ⟨**1.1^4.**⟩ is used in SHANNON's mathematical theory of communication – here as pure measure for rarity of events –.

4. References.

[1] C. E. Shannon, W. Weaver: The mathematical Theory of Communication, University of Illinois Press, Urbana 1949.

[2] J. Hutchinson: Fractals and self–similarity, Indiana Journal of Mathematics 30, 1981.

[3] M. F. Barnsley, V. Ervin, D. Hardin, J. Lancaster: Solution for an inverse problem for fractals and other sets, Proceedings of the National Academy of Sciences 83, 1986.

[4] M. Berger: Encoding images through transition probabilities. Math. Comp. Modelling 11, 1988.

[5] E. R. Vrscay: Iterated function systems: Theory, applications and the inverse problem, in Bélair, J. and S. Dubuc, (eds.), Fractal Geometry and Analysis, Kluwer Academic Publishers, Dordrecht Holland, 1991.

[6] G. Edgar: Measures, Topology and Fractal Geometry, Springer–Verl. NY, 1990.

[7] R. D. Mauldin, S. C. Wiliams: Hausdorff dimension in graph directed constructions, Trans. Amer. Math. Soc. 309, 1988.