— vordenker-archive —

Rudolf Kaehr

(1942-2016)

Title

Morphogrammatics and Computational Reflections

Archive-Number / Categories

2_49 / K09

Publication Date

2010

Keywords

Morphogrammatics, Reflection, Recursion, Meta-Programming

Disciplines

Cybernetics, Artificial Intelligence and Robotics, , Epistemology, Logic and Foundations, Theory of Science,

Abstract

Turning back from the studies of morphogrammatics to some open questions of reflectional programming, the recountered problematics might be put into a different light and new methods of handling formal aspects of reflection and reflectionality shall be introduced. Albeit the use of light-metaphors, morphogrammatic reflection is not sketched along the paradigm of optical metaphors. Morphograms are presenting neither propositions nor perceptions able for mirroring (representation). Exercises in defining morphogrammatic retro-grade recursion and reflection schemata are continued from the paper "*Sketches to Morphogrammatic Programming*".

Citation Information / How to cite

Rudolf Kaehr: "Morphogrammatics and Computational Reflections", www.vordenker.de (Sommer Edition, 2017) J. Paul (Ed.), URL: http://www.vordenker.de/rk/rk_Morphogrammatics-of-Reflections_2010.pdf

Winter November, 2010

Morphogrammatics of Reflection

Rudolf Kaehr

Morphogrammatics and Computational Reflection

Applying insights from the retro-grade recursivity concept of morphogrammatics to questions of reflectionality and interactionality of programming

Rudolf Kaehr Dr. phil[@]

ThinkArt Lab Glasgow ISSN 2041-4358

Abstract

Turning back from the studies of morphogrammatics to some open questions of reflectional programming, the recountered problematics might be put into a different light and new methods of handling formal aspects of reflection and reflectionality shall be introduced. Albeit the use of light-metaphors, morphogrammatic reflection is not sketched along the paradigm of optical metaphors. Morphograms are presenting neither propositions nor perceptions able for mirroring (representation).
Exercises in defining morphogrammatic retro-grade recursion and reflection schemata are continued from the paper *"Sketches to Morphogrammatic Programming"*.
As for previous papers, this is *work in progress* and not a chapter of a text-book.

1. Standards of Reflectional Programming

1.1. Marginality of general concepts and devices

It isn't in any sense new, and there is no need to be new, but it might be expanded to even an inflationary use, that terms like reflection, reflexivity, recursion, re-entry, self-referentiality, Self, and Identity are labels in nearly all contemporary fields of writing in science, culture, ideology, art, comedy and everywhere else.

This sign of reflexivity is not necessarily connected with a flexible awareness of others. Might they be other productions in the field of the Self-business or happening in disjunct codes, media, habits, cultures and languages.

It is therefore very astounding to read a bulk of papers and books in Anglo-British language about and of the mentioned topics, sujets, challenges, debates without finding any reflection on the fact that the whole debate is encapsulated in a specific and local idiom.

I'm not speaking about African dialects or Siberian historic languages, not even about well known European languages like Eastern European languages, no I speak about the languages of the Post War Europe.
To read a couple of books about Self, reflexivity, recursion and reflection without encountering a single German or French citation is disturbing if not catastrophic. I'm not speaking about the 3 mentioned thinker, French or German, for whom there are some rudimentary translations available at Amazon.

Such ignorance at a time of maximal accessibility is paramount. And its aftermath catastrophic, when our Chinese friends who studied at such great institutions like Oxbridge or Goldsmiths are overrunning us with our local theories, now transformed into global truth. This movement is further advanced than we like to accept if we get forced to learn from Singapore what Jacques Derrida really has written to us. And then

there are immediately our PC maniacs in duty.

But there is no Anglo-British academic text about reflexivity which would take the courage to reflect its own marginal insularity. Hence, the concepts and strategies of system and environment, presupposed for reflection, reflexion and reflexivity, are not applied to the conditions of production of those eminent pretentious textual elaborations. Such texts are not reflecting their inter-textuality.

A possible language barrier is no excuse at all for the inflexibility of reflection; it is a conscious strategy. And this becomes even more crucial if we forget the whole language debate and its different cultures by reflecting on different ways of writing.
Texts about the mentioned topics of flexible reflexivity are written in stable homogeneity realizing the narrative forms of essays, monologues or conceptual novels.
Whatever those texts might be from a media-theoretical perspective, there is no disruption between formal and notional languages and writing. What is written is easily be spoken and lectured too.

The whole tradition of formal-mathematical studies about identity, reflexivity, reflection, self-reference, iteration, recursion and much more is segregated as non-profound calculations missing the deepness of philosophical, sociological and psychological contemplation. Crucial techniques, methods and results don't get any mentioning. Such redlined endeavours might be seen as good enough for reflectional computers but of no serious relevance for human cultural studies.

1.2. What are others planting?

"The principles of extending reflexive theories, formulated so far (Gödel, Turing, Feferman) have been limited to incremental, linear advance along the progression of (transfinite) ordinals (Giunchiglia & Smaill 89). Such advance is, however, non-reflexive: the usual extension operator does not take into account its own role in the process of extension: it only repeatedly reproduces the basis for its application. A reflexive extension operator would not do away with the incompleteness of a reflexive theory, but it could extend it in longer leaps along the progression of ordinals. Such reflexive progressions of reflexive theories could be a better model of the kind of reflection which is peculiar to consciousness and which is usually considered to surpass the reflexivity of reflexive formal theories." (Damjan Bojadziev)
http://nl.ijs.si/~damjan/phen.html

Reflective computational systems allow computations to observe and modify properties of their own behavior, especially properties that are typically observed only from some external, meta-level viewpoint.

For example, by representing its interpreter, a program could monitor its own execution to detect loops and then modify (itself or) its interpreter to avoid them.

Reflection in the Integral Object-Oriented System
"Reflection is the capability of a computational system to *"reason about and act upon itself"* (Maes 1987) and adjust itself to changing conditions. The computational domain of a reflective system is the structure and the computations of the system itself. Two kinds of reflection can be observed: *structural* and *computational* reflection (Ferber 1989).
• *Structural Reflection*: is the most obvious and still the most developed form of reflection. It concerns the infinitary status of some data structures defined by reflexive domains (Ferber 1988). The Java Reflection API (Sun 1997) is an example of a restricted kind of Structural Reflection (better called).
• *Computational* or Behavioral Reflection: Is the ability for a process to describe, analyse and modify itself while running."

2. Memristic recursivity

Reflection may one day be as common as recursion - Brian Smith, Reflection and semantics in Lisp
http://nl.ijs.si/~damjan/cr.html

2.1. Towards reflection

Recursion, everywhere
Recursion is re-currence without retro-gradness.
Hence, recursion is iteration of the application onto different results, and never onto the application itself.

Reflection was conceived by computer scientists in the mode of recursion but with the desire to surpass it. That's way self-application was restricted by a last security, offering foundation of restricted self-referentiality. Meta-programming was conceived as a program for implementation of hardware which is by definition not self-referentially constructed.

A chain of terms: Reflexivity - reflection - reflexion - iteration - recursion. (Hibbert 2010)

"The second mode of recursion is perhaps less well described in the literature because it is radically different from the "classical" conceptualization of reflexivity as an active cognitive process. In contradistinction to this popular conceptualization, there are a number of authors that talk of reflexivity as an *unconscious process* by which the process of reflection is itself modified (Beck, 1994; Hoogenboom and Ossewaarde, 2005; Adams, 2003, 2006)."

Functions and relations
"***Recurrence relation***: is an adequation that recursively defines a sequence: each term of the sequence is defined as a function of preceding terms."

Example: Fibonacci numbers
$F_n = F_{n-1} + F_{n-2}$
$F_0 = 0$
$F_1 = 1$

Iteration
$f^0 = id_X$
$f^n = f \circ f^{n-1}$

Systems and programs

*"Computation actually results when a processor (interpreter or CPU) is executing (**part of**) this program'. A reflective system is a system which incorporates structures representing (**aspects of**) itself. We call the sum of these structures the* self-representation *of the system. This self-representation makes it possible for the system to answer questions about itself and support actions on itself." (*Pattie Maes)

Partial self-referentiality in formal systems is save. Total self-referentiaity in formal systems is destructive and producing antinomies. Non-founded sets are not easy accessible to programming and realization.

Dorothy L. Grover, 'Propositional Quantification and Quotation Contexts': "*Therefore our results show that - although unrestricted self-reference leads to inconsistency - partial self-reference need not.*"

2.2. Reflection vs. recursion

2.2.1. Levels of abstractions

Recursitivity is a productive abstraction from the iterativity of functions which are based on identification. Following some notions and constructions of diamond category theory a better understanding of the difference of classical, i.e. identity-based and trans-classical approaches, might be achieved.

identification of identity
iteration
recursion
reflection

Questions
How do I distinct and *identify* an object out of a field of others?
How do I know that I am *repeating* the same object in the process of iteration?
How do I know that my recurrence is not missing its re-entrance of *recursion*?
How is it possible that a system is *reflecting* on itself as a system?

Identity
Identification is separating an entity from its context. No separation without identification, and vice versa, no identification without separation. Therefore, identification is a double act of identifying and separating.

Thus, instead of A = A, we get A |a = A | a.

Automorphism:
$id_{\Delta}A = A$: A --> A o A --> A.

Chiasm:
$id_{\Delta}A = A$: A --> A o A --> A

$$\chi.id_{\Delta}A = \begin{pmatrix} A \longrightarrow A \\ \updownarrow \quad X \quad \updownarrow \\ A \longleftarrow A \end{pmatrix}$$

Diamond:
$\delta(id_{\Delta}A)$: A --> A o A --> A | a <- a

```
        a <- a
        |    |
A --> A o A --> A
```

Iteration (accretion and disremption)
Iteration is repeating a identified, i.e. separated entity as itself: A ==> AA. The crucial question is: What is guaranteeing the sameness of the repeated entity (sign)? Therefore, iteration is involved with the decision to iterate A as A and not as A'. Like identification has to chose its context, iteration has to chose, i.e. *select* its entity A from its context as it is or to *elect* another contexture on the base of the as-characterization of entities and signs.

iter(A): A ==> AA:
$A \in$ Alph, $A' \in$ Alph', Alph = Alph', A = A': iter(A) = AA' = AA.

In a highly elaborated textual constellation it might be a difficult task to find a path through the labyrinth back to the conditions for the specific local iteration. There is not even a guarantee implemented that the object still exists.

Hence, *classical iterability* of iteration, i.e. iteration without alteration, implies the whole machinery of logocentrism (identity, space, time, self).

Thus, why not implementing the *"a priori"* of the operation into the operation itself?
Hence, a morphic iteration is relying on itself only. Repeating *A* means repeating by retrograde recurrence related to the object to be repeated. There is no need to take recourse to a pre-given alphabet.

$$A \in \text{Morph} : \text{disremption}(A)$$

$$\text{disr}(A) = \begin{pmatrix} \text{iteration}(A) \\ \text{accretion}(A) \end{pmatrix} = \begin{pmatrix} AA \\ AB \end{pmatrix}$$

$$\text{diam}(\text{disr}(A)):$$

$$AA : A \circ A \Longrightarrow A \circ A \mid a \longleftarrow a$$

$$AB : A \circ B \Longrightarrow A \circ B \mid a \longleftarrow b.$$

Recall, classical iterability/repeatability (John W P Phillips)
"The relation to self of a letter or mark in its *repeatability* gives rise to the possibility of both the previous kinds of mimesis and thus of any abstraction whatever. This is also the possibility of--or better, the definition of--the meta- (in metalanguage, metafiction, meta-real).

The repeatability of the mark is absolutely egalitarian, because it functions equally wherever languages of any kind (including idioms of design and architecture) are spoken, written or otherwise constructed and--as

becomes clearer every day--wherever a determinate condition meets a process of random indetermination.
The following diagrams demonstrate the kinds of condition that are rendered accessible to experience by the repeatability of the mark. The operation of the repeatable mark varies according to differences between the historicities that determine how cultures react to it."

"The division of experience into space and time, for instance, is preceded by the possibility of the exact doubling of the mark (the letter “c” repeated spatially in the diagram), a situation that applies whether we are discussing alphabetical letters or Chinese characters, each of which function according to the same law (the law demanding the possibility of their repetition in principle to infinity). The same law implies (if one imagines the mark turned and facing the direction of time’s arrow) repeatability into the future, which allows an illimitable play of combinations, substitutions and re-combinations and introduces the irreducible sphere of randomness into experience as a basic condition of its possibility.”
http://courses.nus.edu.sg/course/elljwp/marks09.htm

In radical contrast: Iteration alters (Derrida)
"Play is the disruption of presence. The presence of an element is always a signifying and substitutive reference inscribed in a system of differences and the movement of a chain. Play is always play of absence and presence, but if it is to be thought radically, play must be conceived of before the alternative of presence and absence (SSP 292)." (Derrida)

Recursion
Recursion is inheriting the characterizations of identification and iteration but gets into more sophisticated conditions which have to guarantee the *“re-entrance of the re-entry”* function. The loop of iteration, i.e. its self-application (auto-morphism) has a minimal labyrinthine complexity. Recursion are easily involved with highly complex loop-structures where the re-entrance has specially be pre-established to find a path through its labyrinthine recursions. Such pre-conditions are helpful to guarantee re-entrance but are the (hidden)source of a profound denial of creativity.

Reflection
Reflection is contemplating on the conditions of recursiveness as such. As a second-order term it is thematizing the minimal conditions of recursion as a property of a specific system as such. The machinery of reflection is more complex than the intra-theoretical constructs of identity, iterability and recursion, because it offers methods to deal with the whole instead of with the parts (metatheory, meta-mathematics, meta-logic, etc.).

From a trans-classic point of view it is not sure and granted that this pretension is a mere illusion because the notions and instruments applied remain strictly intra-theoretical and classical. Nevertheless, a theory of reflection has to take the Gödelian limitation theorems as a non-refutable result of classical formal thinking on itself.

2.2.2. Common approaches

It is common to say that recursion is a specific case of iteration where the iterated function enters its own domain. Therefore, it is no surprise that recursivity is ‘easily’ implemented in identification-based programming languages, like Lisp.

Iteration of a function is more or less a semiotic pre-condition of any formalization and programming language construction.
A.A. Markov: Abstraction of potential realizability.

Thus we have to ask what does it mean that reflection as a new abstraction may become as common as recursion?

"Reflection may one day be as common as recursion" - Brian Smith, Reflection and semantics in Lisp

Reflection as a reference of parts of a programming system itself (Maes) sounds still very much in the spirit of recursion theory.

Smith introduced the notion of "*reflection*" into computer science in a much more substantial intention:

"The thesis embodied in 3-Lisp was that reflection, a much more substantial form of self-reference than the mere referential cyclicity of such antinomies as "I am lying", more like what philosophy would call self-knowledge than self-reference, was relatively simple if based on a semantically rationalized base." Brian Cantwell Smith, On the Origin of Objects, p. 37n. 16

At least since Niklas Luhmann's sociological systems theory, the notion of recursion and recursivity as well as reflection (reflexion) is well known in non-technical applications in the humanities. Mostly, those terms are not used as notions and formal constructs but as metaphors or other rhetorical figures.

"The reflexive monitoring *of action in situations of co-presence is the main anchoring feature of social integration..."* (Giddens 1984, 191).
For Giddens, reflexivity begins with the availability of individuals and institutions to reflect upon their own circumstances.
"The point is that reflection *on social processes (theories, and observations about them) continually enter into, become disentangled with and re-enter the universe of events they describe. No such phenomenon exists in the world of inanimate nature, which is indifferent to whatever human beings might claim to know about it....It is impossible to have a modern sovereign state that does not incorporate a discursively articulated theory of the modern sovereign state. The marked tendency towards an expansion of political* 'self-monitoring' *on the part of the state is characteristic of modernity in the West in general, creating the social and intellectual climate from which specialized, 'professional' discourses of social sciences have developed but also both express and foster."* (Giddens 1984, xxxiii),
from: http://junana.com/CDP/corpus/GLOSSARY22.html

Therefore it is interesting to learn what happens with the use of the notion recursion in the theory of managment sciences

"First, the principal dimensions of reflexivity - reflection and recursion - are identified and delineated. Second, recursion is shown to have two modes, active and passive. Third, reflection is shown to have both closed, self-guided and open, relation modes. Fourth, through integrating the detailed characterizations of each of the dimensions, different types of relfexivity are identified and defined." (Hibbert) http://strathprints.strath.ac.uk/13706/

Recursion is a scheme which allows iterative application on the results of previous calculations by the same scheme. The scheme itself doesn't change during such iterative applications, the data only are changing.

Reflection sounds similar to recursion but gets applied not on data but on full programs containing recursions calculating iterative data. Hence, reflection is a second-order term, while recursion still is a first-order term designing a scheme of applications within the use of iterative repetitions.

Reflexion vs. self-reference

„Thus, the appearance of second order cybernetics is the appearance of a new dimension - *reflexion*.
However, this dimension was developed differently in the Soviet Union and the West. In the Soviet Union, the idea of reflexion was combined with the idea of *structure*; as a result, reflexive analysis appeared. In the West, the idea of reflexion was combined with the idea of *computation*; as a result, calculations with self-reference appeared." (Lefebvre 1986, 128)

The paper hints to a concept of reflexivity which might contain recursion and reflection as its dimensions.

2.2.3. Philosophical theory of Self-awareness

Self-awareness

„Damit zeichnet sich eine Antwort ab auf die Frage,..., inwiefern jemand sich in seinen praktischen Ja/Nein-Stellungnahmen - in seinem 'ich kann -' - zu sich verhält. Die Antwort lautet: nicht indem das Subjekt sich selbst zum Objekt wird, sondern indem es sich zu seiner Existenz verhält."(Tugendhat 1979, 38)

„Daß ich mich voluntativ-affektiv zu meiner Existenz *verhalten kann, gründet darin, daß die Proposition, zu der ich mich dabei verhalte, nicht das Faktum ist, daß ich existiere, sondern die bevor stehende Existenz und das heißt die (praktische) Notwendigkeit, daß ich zu sein habe, und in eins die (praktische) Möglichkeit, zu sein oder nicht zu sein bzw. so und so zu sein oder nicht zu sein."* (Tugendhat 1979,189), engl.:Tugendhat, p. 168

Antinomies
Which level of thematization are we forced to establish if self-reference of a subject is not to be thought as a relation to itself but as a *"voluntative-affective"* behavior to its own existence? This is not easy to decide and Tugendhat is not elaborating explicit answers to handle it. But it seems to be clear that the intended level of relationality is not in any way determined by the distinction of subject and object. It is not a relation of a subject to itself as an object.
http://www.thinkartlab.com/pkl/media/DISSEM-final.pdf

Morphogrammatics
This is hint enough to connect this approach beyond subject and object with the very intention of morphogrammatics. According to Gotthard Gunther, morphograms are connected neither with subjectivity nor with objectivity, i.e. the distinction of operator and operand fails too. Technically speaking, morphograms are negation-invariant patterns of kenograms.

It might be therefore reasonable to associate the morphogrammatic abstraction which leads to morphogrammatics with the 'abstraction' involved in the praxeological abstraction/distancing from subject and object. Hence, the intention of reflexion and reflectional programming might be closer understood in the framework of an *'existential praxeology'* and *morphogrammatics*.

Rudolf Kaehr, Vom Selbst in der Selbstorganisation.
Reflexionen zu den Problemen der Konzeptionalisierung und Formalisierung selbstbezüglicher Strukturbildungen (1992)
http://www.thinkartlab.com/pkl/media/SelbstB2.frame.pdf

2.3. Citations from Gunther's cybernetic theory of reflection (1962)

"We all know from our own psychological introspection that our consciousness has a capacity for a theoretically unlimited self-iteration of its concepts. Fichte has drawn our attention to its (negative) logical significance. We have, he says, a concept of something and may iterate it into a: concept of a concept of a concept.....of something and so on ad nauseam. He and later Hegel point out that after the second step no increase in logical structure can be expected. The endless iteration of our reflection is, to use a term of Hegel, "eine schlechte Unendlichkeit" (a bad infinity). It is important to point out that there are indeed two utterly different ways in which a formal increase of reflection may be obtained: first, by (empty) iteration of a morphogrammatically saturated system and second, by a growth of morphogrammatic structure. It is a serious argument against the reflective power of the infinite hierarchy of two-valued meta-systems that this hierarchy represents an iteration of the first kind.

"A system which is described with the exclusive use of categories derived from a logic with the above morphogrammatic restriction has a most significant property: *it has no environment of its own*! Environment would mean a third value! It also means structural asymmetry.

"Quite a different thing is a system which reflects its environment by organizing itself and producing additional structure.

"In Hegel´s logic the phenomenon of reflection is subdivided into three parts: He defines them as:
a) retroverted reflection (Reflexion-in-sich)
b) transverted reflection (Reflexion-in-Anderes)
c) retroverted reflection of retroversion and transversion (Reflexion-in-sich der Reflexion-in-sich und-Anderes)
Section (a) represents the physical system of the external world described by its specific reflective properties. But (b) and (c) signify the additional capacities of reflection which sub-systems of the Universe must possess if they are to be called subjects.

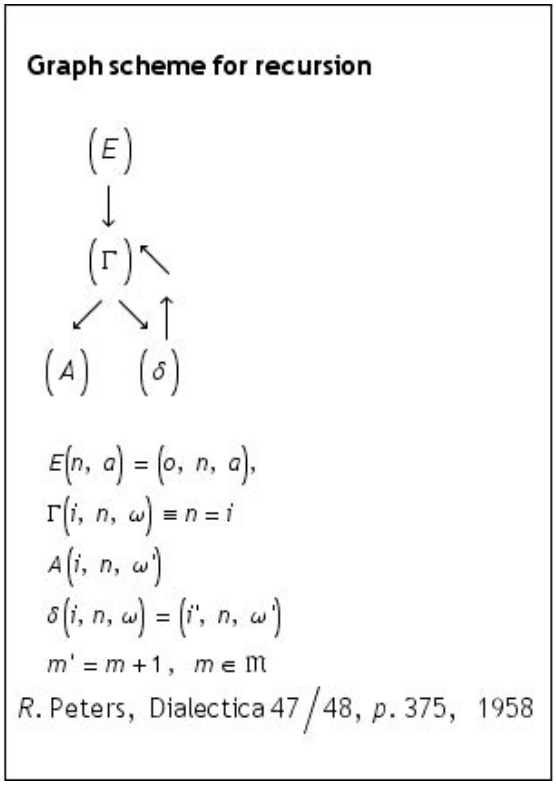
"We talk about self-organizing systems and their environments; but Hegel´s distinction between (a), (b) and (c) shows that this is not enough. A self-reflective system which shows genuine traits of subjective behavior must be capable of distinguishing between two types of environment and be able to react accordingly. First

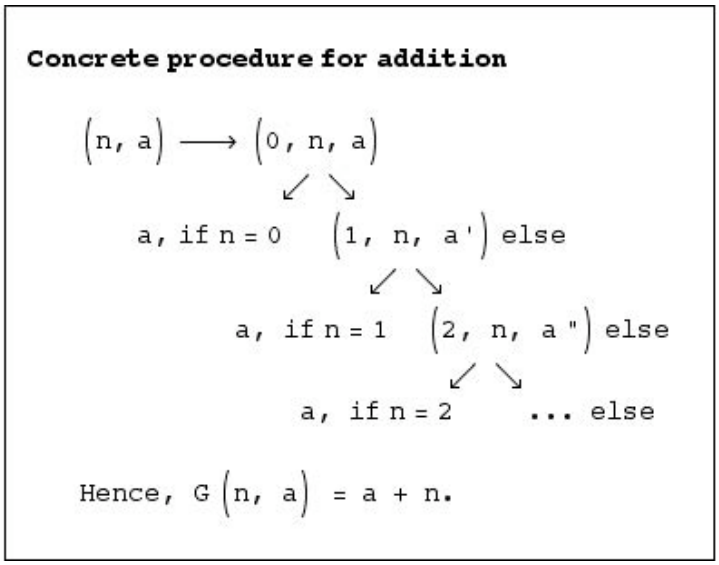
it must reflect an "outside" environment which lies beyond its own adiabatic shell and second it must be capable of treating (b) as an environment to (c)."
Gotthard Gunther, Cybernetic Ontology and Transjunctional Operations,
Self-Organizing Systems, M. C. Yovits, G. T. Jacobi G. D. Goldstein
(eds.), Washington D. C. (Spartan Books) 1962, 313-392
http://www.vordenker.de/ggphilosophy/gg_cyb_ontology.pdf

3. Diamond theory of recursion and reflection

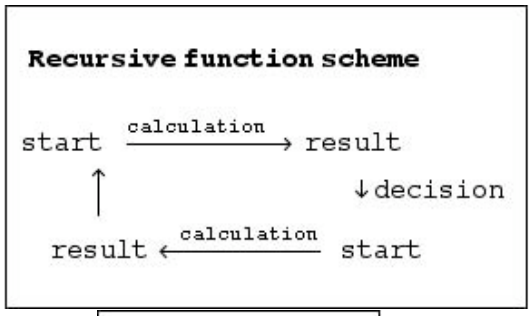
3.1. Recursion, revised

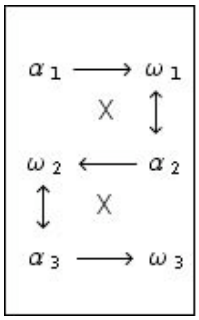
3.1.1. Recursion, iteration and chiasm

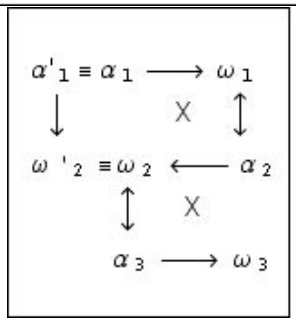




Graph schema of iteration **Open chiasm** **Open chiasm with results**



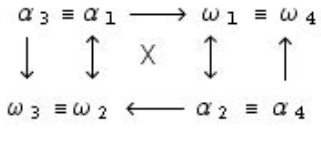




An *open chiasm* is not yet closing the conditions of iterability, nevertheless it offers a scheme which guarantees the re-entry to catch its entrance. Obviously, the *numbers* (signs) *m* are taken from the *reservoir* M, i.e., $m \in \mathbb{M}$. This kind of iterativity, basic for recursivity, is ruled by the *coincidence* (similarity) relation between beginnings(α) and ends (ω), i.e. α_i - α_{i+1} and ω_i - ω_{i+1}. The interchange between beginnings and ends is ruled too by the *exchange* relation between different levels, i.e., α_i <-> ω_{i+1}. The chiasm would get some more strictness if the exchange relations between the inverse beginnings and ends, i.e. α_{i+1} <-> ω_i would be involved into the characterization of recursivity in general. The step-wise results are given at α'_1 --> ω'_2, etc.
With those relations implemented, no ghost or genius is needed anymore to guarantee the claim of recursivity.

3.1.2. Diamond of recursion

Closed chiasm as a diamond



Diamond of recursion as an inscription of the closed diamond. In other words, open recursion is embedded into its own environment by the saltitional morphism, while the recursion result is given by the acceptional morphism of the composition of the recursion morphisms.

Recursion
start: $A_1 = \varnothing$
procedure: $A_n = A_{n-1} \cup \{a_i\}$

Diamond of recursion

Diamond of recursion

$$\mathcal{A}_1 = \varnothing \mid \square$$

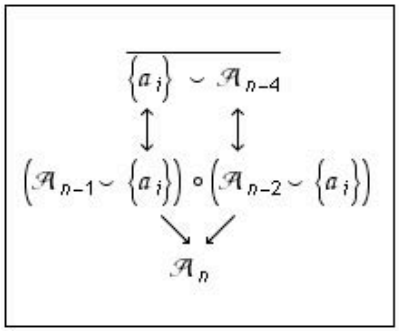
$$\mathcal{A}_n \mid \overline{\mathcal{A}_{n-1}} = \mathcal{A}_{n-1} \frown \{a_i\} \mid \overline{\{a_i\} \frown \mathcal{A}_{n-2}}$$

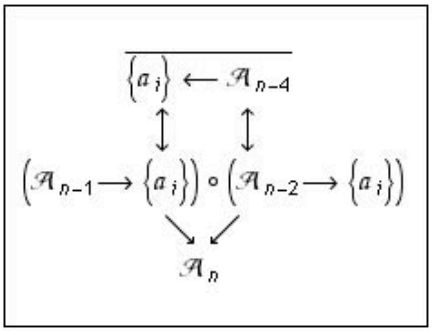
Categorical characterization:

$$\mathcal{A}_n = \mathcal{A}_{n-1} \frown \{a_i\} :$$

$$(\mathcal{A}_{n-1} \frown \{a_i\}) \circ (\mathcal{A}_{n-2} \frown \{a_i\})$$

$$\text{diam}\left((\mathcal{A}_{n-1} \frown \{a_i\}) \circ (\mathcal{A}_{n-2} \frown \{a_i\})\right) = \mathcal{A}_{n-3} \frown \{a_i\} \mid \overline{\{a_i\} \frown \mathcal{A}_{n-4}}$$





Diamond recursion schema

$$\mathcal{A}_{n\,3} \equiv \mathcal{A}_{n-1\,1} \longrightarrow \{a_i\}_1 \equiv \{a_i\}_4$$

$$\downarrow \quad \updownarrow \quad X \quad \updownarrow \quad \uparrow$$

$$\{a_i\}_3 \equiv \{a_i\}_2 \longleftarrow \mathcal{A}_{n-2\,2} \equiv \mathcal{A}_{n-3\,4}$$

Recursion, again, is defined by atomic linear concatenation, albeit with some recurrence to the former results, but without any retro-grade and context ‘sensitive’ constructs.

From a strictly formal point of view it is provable that recursion and iterative induction are equivalent.

The concept and strategy of recursion entered formal mathematics as a device to avoid antinomic situation because of uncontrollable consequences of constructions.
Recursion offers a general and save strategy to develop step by step most kind of mathematical constructions in a save way.

"The third system which was developed to escape the reflexive paradox was Skolem's recursive arithmetic. Skolem observed that he could avoid the paradox without recourse to the restrictions of type theory [Russell] and without the rejection of any rule of classical logic [Brower] if he did not take the existence as one of the primitive notions of logic.

"The sacrifice of existence as a primitive notion deprived Skolem of the classical method of function definition and in its place he introduced definition by recursion.

"A function f(n) is said to be defined by recursion id, instead of being defined explicitly, only the value of f(0) is given, and f(n+1) is expressed as a function of f(n). In other words a recursive definitiondoes not define f(n) itself, but provides a process wherby the falues of f(0), f(1), f(3) and so on, are determined one after the other."
R.L. Goodstein, Recursive Number Theory, 1964, p. VIII.

Funny enough, today we are not even convinced by such prudence.
WiKi states correctly for the recursive definition of natural numbers:

"1 is in N
if n is in N, then n+1 is in N.

"(The doubt with this definition is that we assume: 1. we understand the "+" operation and 2. n + 1 is not in current N . These two assumptions mean that before we understand the natural numbers, we already know the "+" operation on them.)"

In other words, that recursion is working, we have to presume the correctness and stability of the re-entry into the re-current function.

Even more fundamental critics which are not assuming anymore the *uniqueness* of natural numbers has been developed by the Ultra-Intuitionists (A. Esenin Volpin, Geiser, Isles, Parikh).
http://www.thinkartlab.com/pkl/lola/Games-short.pdf
Rohit J. Parikh, Existence and Feasibility in Arithmetic, J.S.L. 36, 494-508, (1971)

If we don't believe in the uniqueness of the natural number series we have to guarantee even more precisely that the re-entrance of recursion takes place within the desired natural number series and not at another one.

3.1.3. Morphic recursion

How is the new morphogram be generated if the successor operation is not able to make recurs to an external resource? At least there is still an internal resource given, i.e. the encountered morphogram. Hence, the encountered morphogram offers all possible continuations of itself. This happens as an iteration of its parts (monomorphies or kenograms) and the generosity of adding something new to the encountered morphogram by accretion. As much as the range of iteration is ruled by the structure of the pre-given morphogram, the range of the new is restricted by the morphogram too. The new is new only in respect to the morphogram and not in respect to an imaginary repertoire of unlimited resources.

This is in a very hard contrast to continuations in semiotics. Semiotic sign sequences might be prolonged by any sign of the pre-given external repertoire, independently of the structure of the sign sequence, and independently of time, space and matter too.

Hence, a pro-gression of a morphogram is involved into a simultaneous retro-gression of the continuation operation. But this alone is not yet guaranteeing a successful operation. There is still the possibility open that pro- and retro-gression don't meet at the meeting point. To match, conditions of matching have to play their part. Re-entering the entry excludes the possibility to miss the re-entrance.
Hence, an interplay of forwards and backwards, pro- and retro-, has to manage the game. Until now, this management has been guaranteed by the designer and administrator of the system, and not by the concrete operations in concrete situations.

The desired prolongation as the start has to select the element of prolongation in the given morphogram by a retrograde action as the end. This end, as a chosen element, i.e. as the begin of the progression action, has to be put as the end of the prolongation activity at the new morphogram.

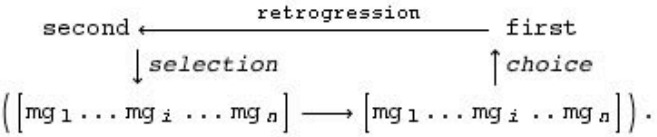
The mechanism is defined as a chiasm between progression, retro-gression, begin (choice) and end (selection) over two loci (places of morphograms).

Instead of just writing a self-cycle of a mapping into itself to produce a successor, the self-mapping is made operationally explicit and visible with a distribution of the morphogram as the given, the addressed and as the producing (prolongating) morphogram and finally its result, the changed morphogram.
As usual, the strategy is to change self-circles into tetralectic chiasms.

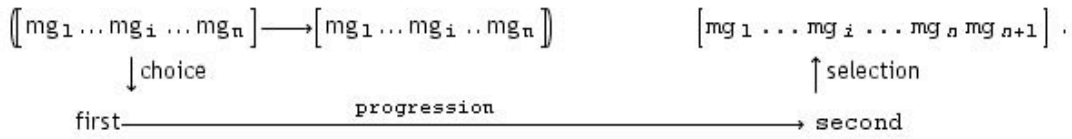
The next step goes beyond chiasms and is reflecting the matching conditions of the chiastic 'cycle' as a closure of the tetradics.

Steps of explanation

1. $\left[\mathrm{MG}\right] \longrightarrow \left[\mathrm{MG}\right]$:



2. $\mathrm{evol}_i\left(\left[\mathrm{MG}_n\right] \longrightarrow \left[\mathrm{MG}_n\right]\right) = \left(\left[\mathrm{MG}_n\right] \longrightarrow \left[\mathrm{MG}_n\right]\right) \longrightarrow \left[\mathrm{MG}_{n+i}\right]$:

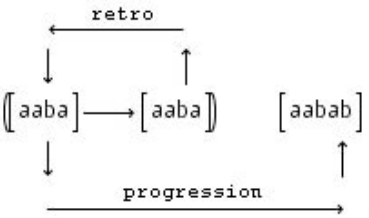


3. $\left[\mathrm{MG}_{n+i}\right]$:

$\mathrm{evol}_i\left(\left[mg_1 \ldots mg_i \ldots mg_n\right] \longrightarrow \left[mg_1 \ldots mg_i \ldots mg_n\right]\right) \Longrightarrow \left[mg_1 \ldots mg_i \ldots mg_n\, mg_{n+1}\right]$.

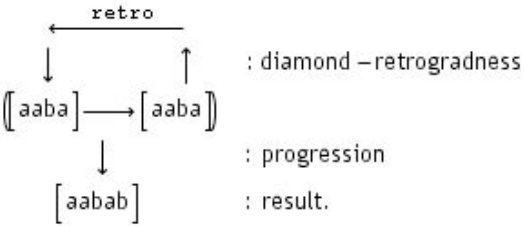
Example

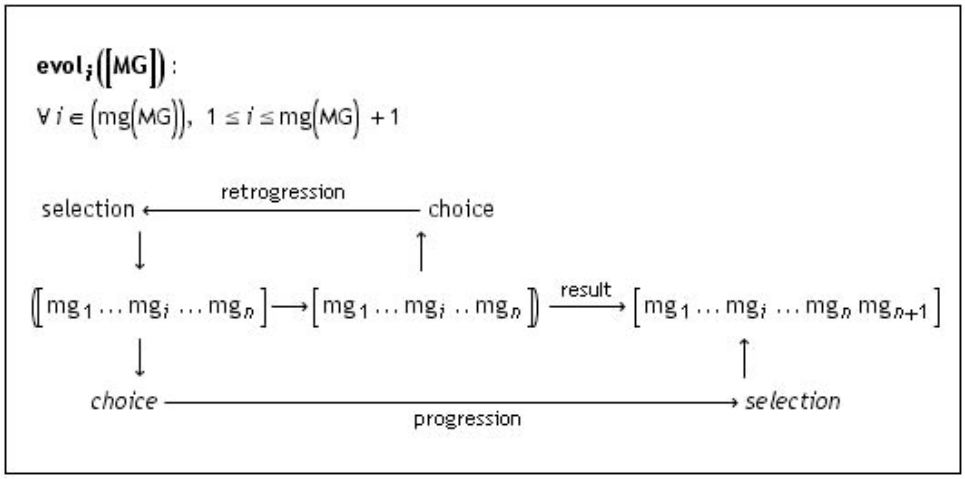
$\mathrm{evol}_b\left(\left[\mathrm{aaba}\right] \longrightarrow \left[\mathrm{aaba}\right]\right)$:

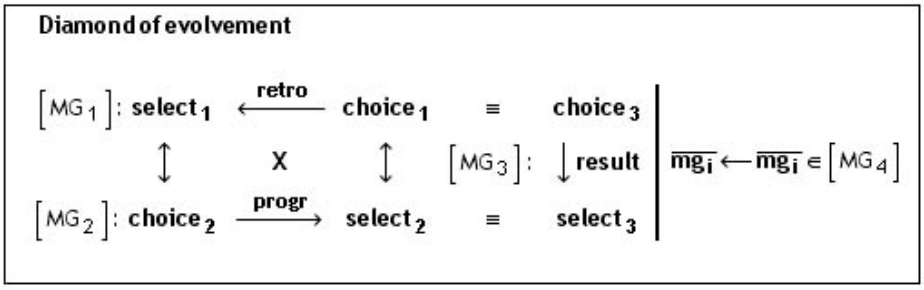


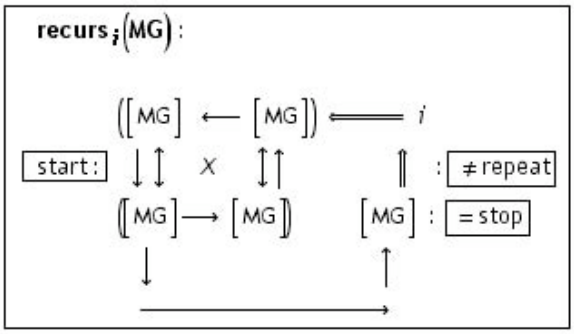
$\mathrm{evol}_b\left(\left[\mathrm{aaba}\right]\right) \Longrightarrow \left[\mathrm{aabab}\right]$.

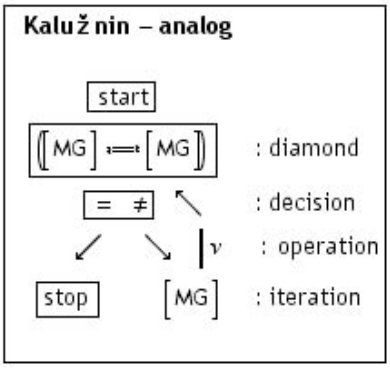
Shorter version :











3.1.4. Implementation of recursion

In FORTH, recursion is simply defined as the *"ability to call a word within itself"*.
Single-level recursion is simply implemented with the word MYSELF (or RECURSE), which is defined as:

```
: MYSELF LATEST PFA CFA, ; IMMEDIATE
```

```
: CHIASM (1.2) ( context1 : MYSELF LATEST PFA CFA, ; IMMEDIATE; ; elect2
                 context2 : YOURSELF FIRST CFA PFA, ; IMMEDIATE; ; elect1 )
```

3.2. Retrograde recursion

3.2.1. Steps towards recursion schemes

From recursion to chiasm and diamonds

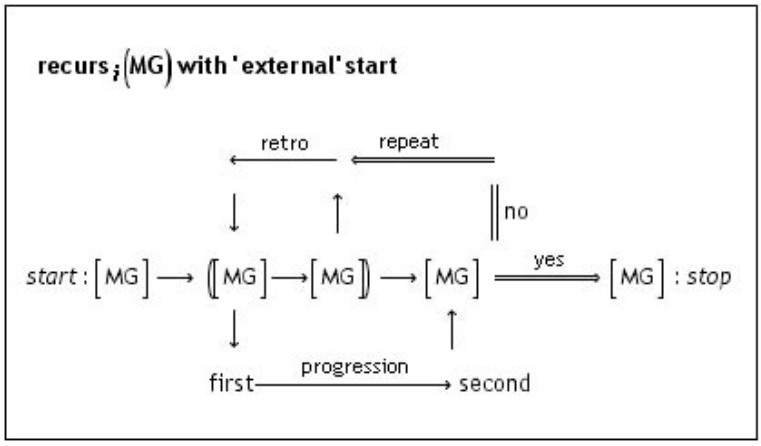
From recursion as an open chiasm to reflection as a closed chiasm, i.e. a diamond.

The term *start* in the morphogrammatic schemes for recursion is not referring to an alphabet as a sign repertoire but just to the start of the self-applicative operation, i.e. the chiastic self-generation of the continuations and their implied signs. Iterative continuations are trivially determined by the kenomes of the preceding morphogram. A more interesting question arises for accretive prolongations. Here, a new sign, marking a difference is introduced. Where is it from?

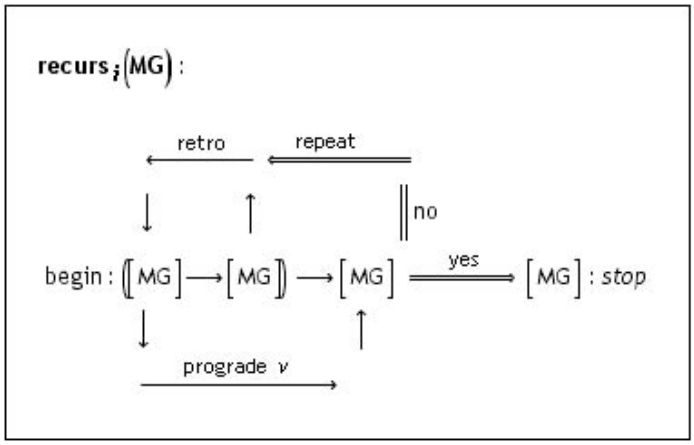
As developed many times at different places, morphogrrams are not dealing with signs or abstractions from signs but with differences. The main difference for morphograms is the difference between the process of the inscription (production) of morphograms and the inscription of the possibilities of their situational inscription. Morphogrammatic operations are strictly defined by their environment, i.e. the conditions of their possibility, therefore, morphograms are not atomistically build but are realizing the interplay of categorical and saltatorical actions of diamonds.

From a strictly formal point of view, morphogrammatic prolongations are compositions of 'morphisms'. Diamond category theory shows that each composition is involved with its complementary saltisitions. Saltisitions are the place where the *'in-sourcing'* of the matching conditions for the composition of morphisms take place. Hence, recursive compositions are always accompanied by their complementary saltisitions as the environment of their existence. This complementary difference allows to realize the procedure of disremption. Disremptions are introduced as being simultaneously iterative and accretive, thus, repeating the 'old' and evoking the 'new' at once. All that is not involved with ontological questions of *creatio ex nihilo* ("creation out of nothing") or the military strategy of "*doubling*" (dédoublement), [German, das Kalben] but only, as the name suggests, when forming *two new units on the framework of one old one"*. (WiKi)

In contrast to the successor operation in word algebras, the operation of disremption, with its two aspects of iteration and accretion, is always defined by the simultaneity of a retro-grade and a prograde action.
http://www.thinkartlab.com/pkl/media/Diamond_Disremption/Diamond_Disremption.pdf

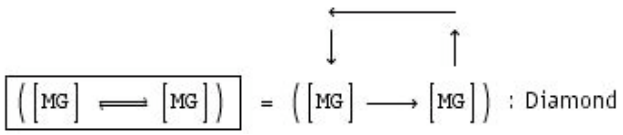


Because there is no eternal start element of a morphogrammatic recursion, the start begins everywhere with an encountered morphogram. Therefore, morphogrammatic starts are just beginnings based on what enters the game. There are no systematics starts with elements of a pre-given alphabet. Hence, the "start" analogy of the scheme "start: [MG] -->" of the diagram shall be omitted.



The scheme shall be adjuted to standard representations as it can be found in the literature as Kaluž nin Graph Schemata with decision logic.
The crucial element of a morphogrammatic recursion scheme is obviously its chiastic concept of retro-grade iteration defined by the self-application unit:

”d*iamond*".



The iteration *operation* (prolongation) ν : [diamond] $\xrightarrow{\nu}$ [MG] and the *decision* unit: $\boxed{= \neq}$ with its two modes: "=" for “stop”, and “!=” for repetition “repeat”.

All together defines the morphogrammatic recursion scheme as a composition of the units “diamond”, “decision”, “operation, calculation” and “repetition, iteration”.
Except of the diamond unit and the lack of an external repertoire for the application of the repetition, the difference to the classical recursion scheme might not look too impressive.

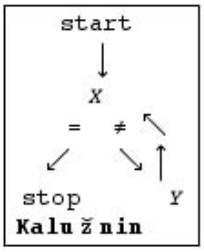
A closer look shows us two and not only one iteration circles, the *internal* diamond circle, and the *external* calculation circle, depending on the diamond production and on the decision unit. Hence, a morphogrammatic recursion scheme is a second-order construct, i.e. a recursion of a recursion, one with a retrograde, the other with a prograde orientation. Both operations are running and holding simultaneously and are therefore mediated, and ‘reflecting’ each other.

It might be said, and later more elaborated, that morphogrammatic recursion schemes are in fact schemata of reflection, based on a two layered simultaneous recursion. It is well known that the cybernetician Gotthard Gunther developed a theory of reflexion based on his theory of polycontexturality where the change of contextures is identified as an action of the reflexivity or reflexionality of reflexion.
Similar constructions of two layered computation had been introduced in the context of reflectional programming with the distinction of object- and method-computation.

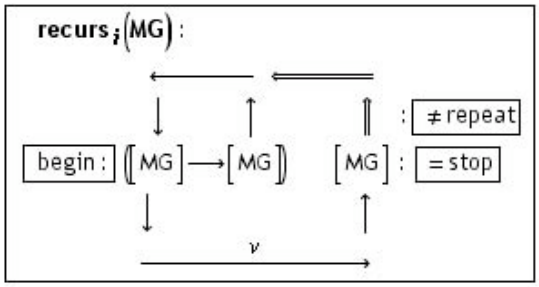
"Reflective architextures provide fundamentally new paradigm for thinking about computational systems. In a reflective architexture, a computational system is viewed as incorporating an object part and a reflective part." (Pattie Maes, p. 23)

Hence, morphogrammatic recursivity, understood as reflexion, is able to change the frame of recursion while applied.

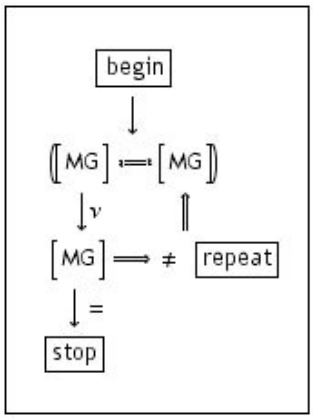
This is in strict difference to the concept of the classical recursion schema as it is definitively put by Kaluznin: There is a *start*, a *decision*, a *repetition* and a *stop* over an *identitive* repertoire of signs. Non-terminating algorithms are not excluded, they simply don’t stop.
There is no possibility offered by the architectures (tectonics) of the schema to change the schema while running a calculation.

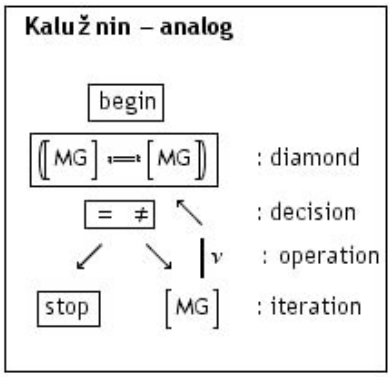


Morphogrammatic reflection scheme

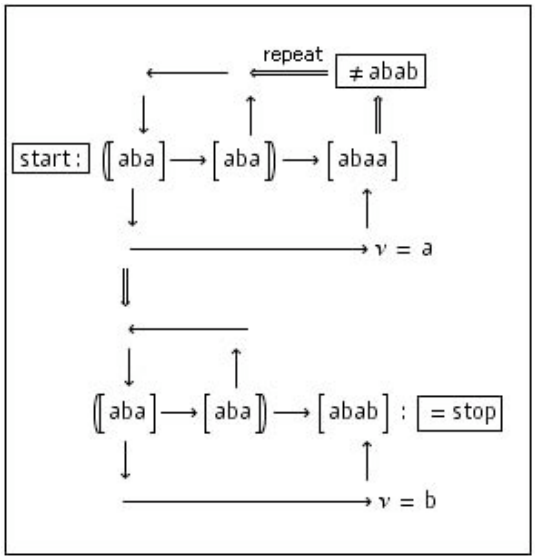


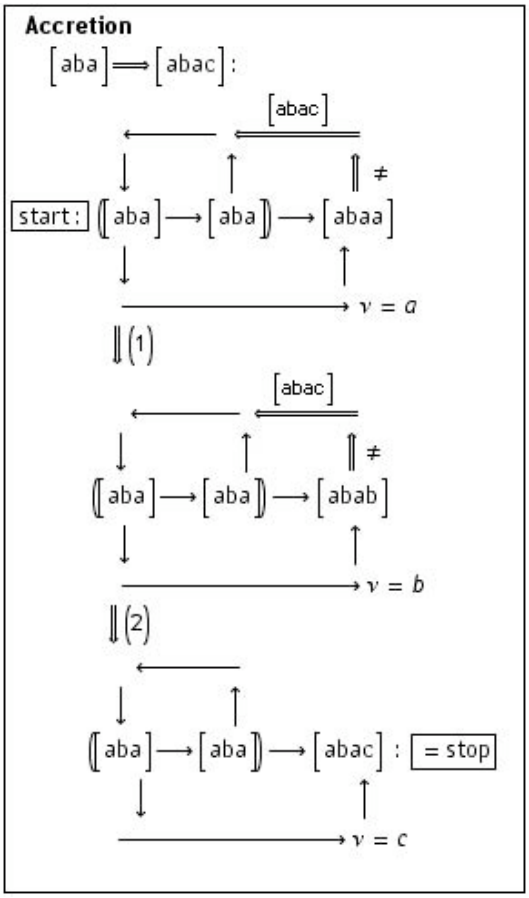
Analogs to classical recursion schemes

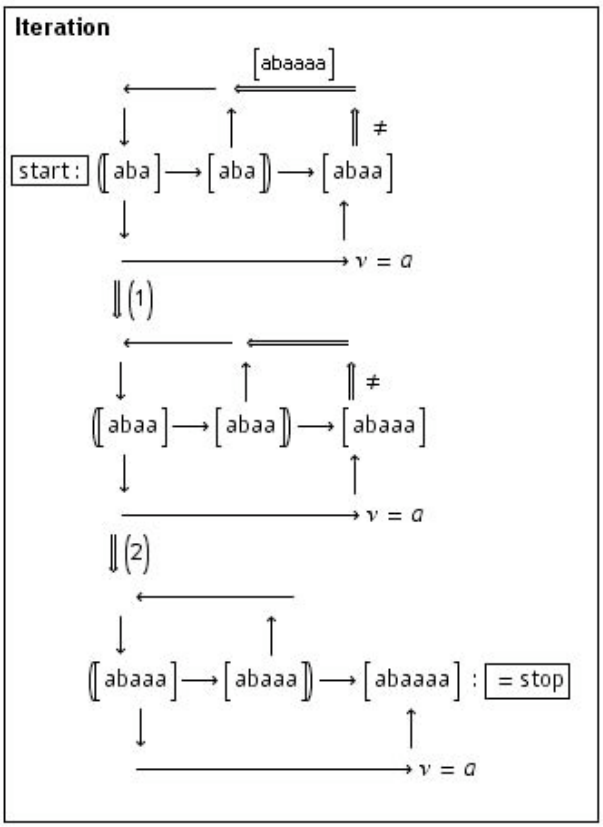


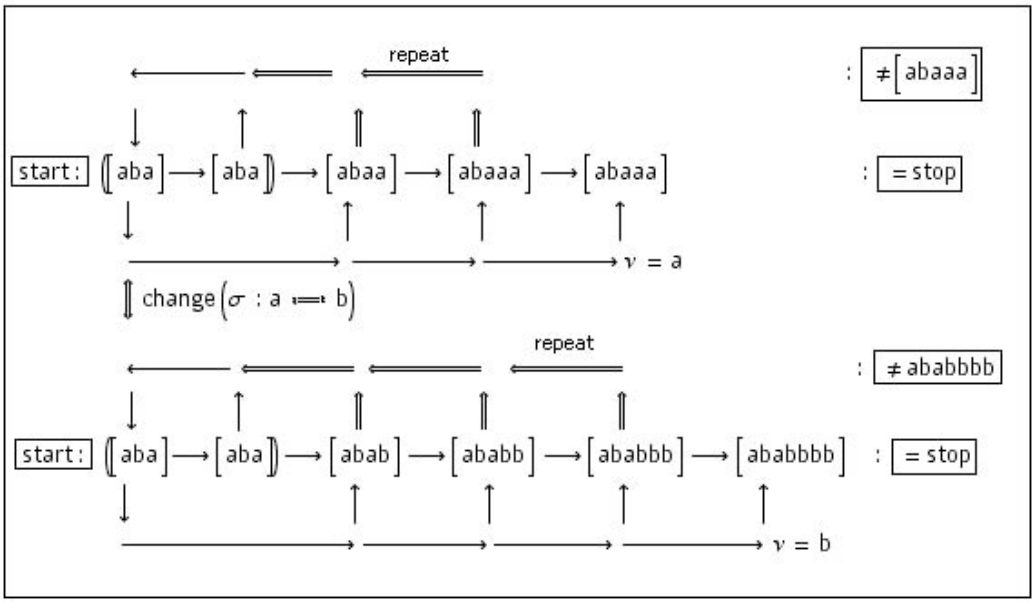


3.2.2. Applications of morphogrammatic recursion schemes

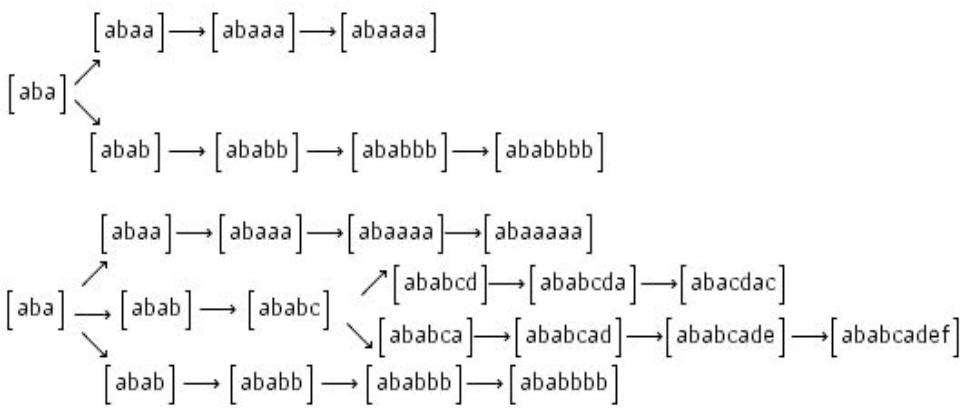


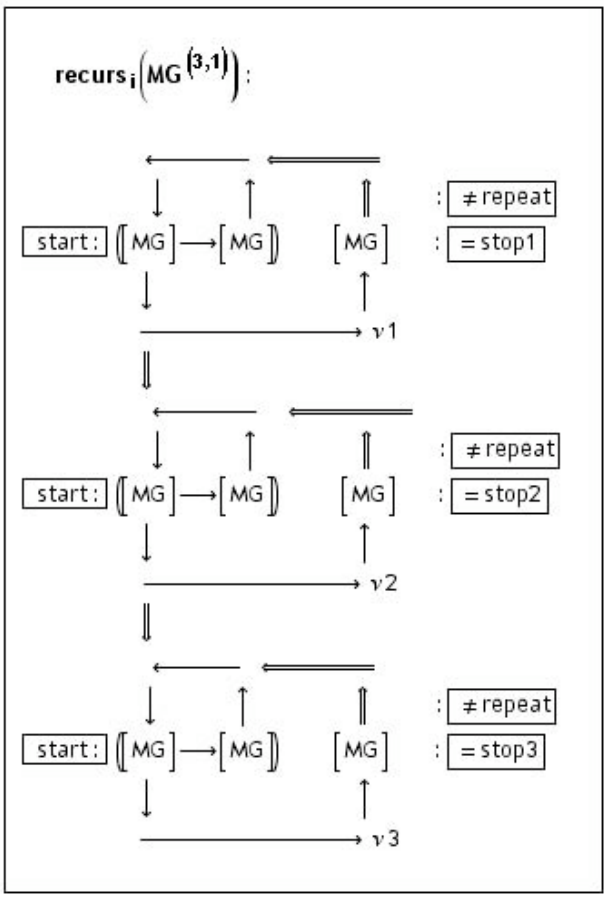

Accretion
[aba] ⟹ [abac] :
[abac]
≠
start : [aba] ⟶ [aba] ⟶ [abaa]
v = a
(1)
[abac]
≠
[aba] ⟶ [aba] ⟶ [abab]
v = b
(2)
[aba] ⟶ [aba] ⟶ [abac] : = stop
v = c


Iteration
[abaaaa]
≠
start : [aba] ⟶ [aba] ⟶ [abaa]
v = a
(1)
≠
[abaa] ⟶ [abaa] ⟶ [abaaa]
v = a
(2)
[abaaa] ⟶ [abaaa] ⟶ [abaaaa] : = stop
v = a



Possible continuations





3.2.3. Different types of repeatability and similarity

Until now the recursion schemes understood as recursion or as reflexin had been cnsidered under a simple type of repetability and as a consequence of comparability: the case of concatenative repeatability and its kenomic equivalence.
The logic unit $\boxed{= \neq}$ for decisions, i.e. the comparision between the produced and the intended morphogram, had been defined by the equivalence or non-equivalence of morphograms. Also, the production mode of the diamond unit was running in the kenomic mode of [single, simple, self-reference]. Hence, the iteration operation iterated this type of repeatability.

This restriction to a simple type of morphogrammatic repeatability is reasonable for the introduction of some features of recursivity and reflexivity into morphogrammatics, like the crucial feature implemented by diamonds to guarantee the re-entry to re-enter the function at the re-entrance and its independence from an external sign repertoire.

In earlier papers the presentation of this topic was conceived slightly more comprehensive. Hence, some hints are added.

The many faces of reapetability

bisimular resemblance - co-creative machines - metamorphosis
monomorphic similarity - self-organizing machines - alterability
kenomic equivalence - non-trivial machines - iterability
semiotic equality - trivial machines - iteration

□	multiple	single	poly	mono	complex	simple	atomic	self – ref
Bisimilarity	x	□	x	□	x	□	□	x
Polymorphy	□	x	x	□	x	□	□	x
Monomorphy	□	x	□	x	□	x	□	x
Identity	□	x	□	□	□	x	x	□

$$\left[\left[MG\right] \Longrightarrow \left[MG\right]\right] \longrightarrow \left(\text{mono} \amalg \text{poly} \amalg \text{bism}\right) :$$

$$\text{Monomorph} : \text{mono} \in \left[MG\right] : \text{op}\left[MG\right] \longrightarrow \left[MG, \text{mono}\right]$$

$$\text{Polymorph} : \text{poly} \in \left[MG\right] : \text{op}\left[MG\right] \longrightarrow \left[MG, \text{poly}\right]$$

$$\text{Bisimilar} : \text{bism} \in \left[\left[MG_1\right], \left[MG_2\right]\right] : \left(\text{op}_1 \ldots \text{op}_n\right) \left(\left[\left[MG_1\right], \left[MG_2\right]\right] \longrightarrow \left[\left[MG_1\right] =_{\text{bis}} \left[MG_2\right]\right]\right)$$

http://memristors.memristics.com/Machines/Orientation/orientation.html

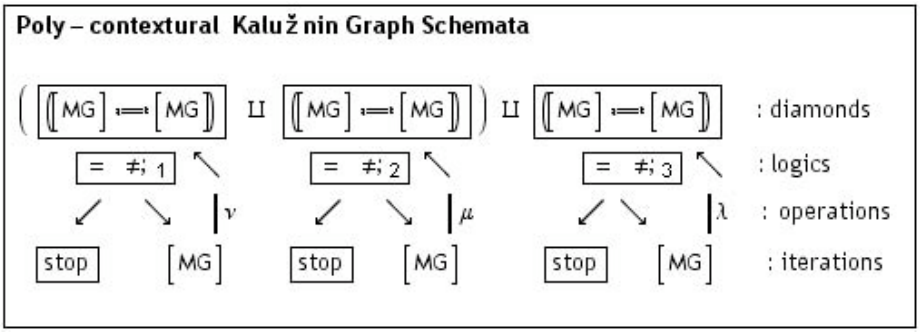
3.2.4. Polycontextural decision logics for recursion schemes

The decision logic for recursion seems to be trivial, its function is to rule the decision of “yes” and “no” of the equality between the produced and the intended result.

Things are getting more intriguing if we allow heterarchically distributed recursion schemes. Such a radicalized parallelism is demanding for a distributed decision logic. This demand is easily fulfilled by polycontextural logics and their multi-negational systems.

Again, the polycontextural approach to recursion has not to be confused with a general *many-successor* word-arithmetics which is based on a common alphabet and a corresponding single logic.

Hence, the monocontextural decision logic $\boxed{= \neq}$ has to be distributed, at first indexed: $\boxed{(= \neq) i}_{i \in s(m)}$ by the place-values of a polycontextural logic for its *distribution*. But such distributed systems are not running in separation but are interacting together, therefore they need to be *mediated* too.



The same morphogram is 'source' for different simultaneous morphic successor systems, all generating their own discontextural recursive behavior implemented by different recursion schemes.

Recursion scheme $\text{rec}^{(3)}_{(\nu\mu\lambda)}$

$$\text{rec}^{(3)}_{(\nu\mu\lambda)} : \left(\text{rec}_\nu\left(\left[MG\right], \log_1\right) \amalg \text{rec}_\mu\left(\left[MG\right], \log_2\right)\right) \amalg \text{rec}_\lambda\left(\left[MG\right], \log_3\right)$$

Decision logics

$$\log_1 = \left\{t_1, f_1\right\}$$

$$\log_2 = \left\{t_2, f_2\right\}$$

$$\log_3 = \left\{t_3, f3\right\}$$

Matching conditions

$$\text{cod}\left(\log_1\right) = \text{dom}\left(\log_2\right), \ \text{dom}\left(\log_1\right) = \text{dom}\left(\log_3\right), \ \text{cod}\left(\log_2\right) = \text{cod}\left(\log_3\right).$$

Interchangeability

$$\begin{bmatrix} \begin{pmatrix} \log_{1.0.0} \\ \amalg_{1.2} \\ \log_{0.2.0} \end{pmatrix} \\ \amalg_{1.2.3} \\ \log_{0.0.3} \end{bmatrix} : \begin{bmatrix} \begin{pmatrix} t_{1.0.0} \\ \amalg_{1.2} \\ t_{0.2.0} \end{pmatrix} \\ \amalg_{1.2.3} \\ t_{0.0.3} \end{bmatrix} \circ \begin{bmatrix} \begin{pmatrix} f_{1.0.0} \\ \amalg_{1.2} \\ f_{0.2.0} \end{pmatrix} \\ \amalg_{1.2.3} \\ f_{0.0.3} \end{bmatrix} = \begin{bmatrix} \begin{pmatrix} t_{1.0.0} \circ f_{1.0.0} \\ \amalg_{1.2} \\ t_{0.2.0} \circ f_{0.2.0} \end{pmatrix} \\ \amalg_{1.2.3} \\ t_{0.0.3} \circ f_{0.0.3} \end{bmatrix}$$

Inter- and transcontextural substitution for contextural recursion schemata
Substitution is an elementary operation for algorithm theory, like its dual operation concatenation.

"Most familiar algorithms can be broken down into a few simple steps. Starting from this observation, and following Markov [1954], we select a particular operation, substitution *of one word for another, as the basic unit from which algorithms are to be constructed."* (Mendelson, Introduction to mathematical logic, 1964, p. 208)

Said this, a distribution of the substitution scheme over discontextural places follows naturally.

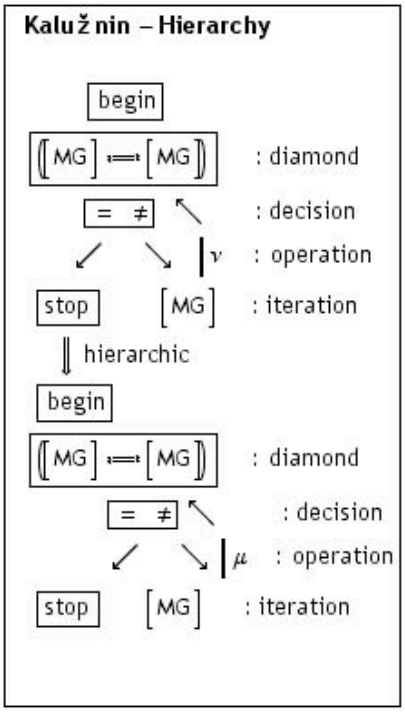
As a consequence, a new distinction has to be introduced. Again, it is the distinction between the intra-contextural substitution based on *selection* and the transcontextural substitution based on *election*.
Selection rules the yes/no-logic of the decision procedure for intra-contextural recursive calculations. Election rules the choice of contextures where the calculation gets its continuations.

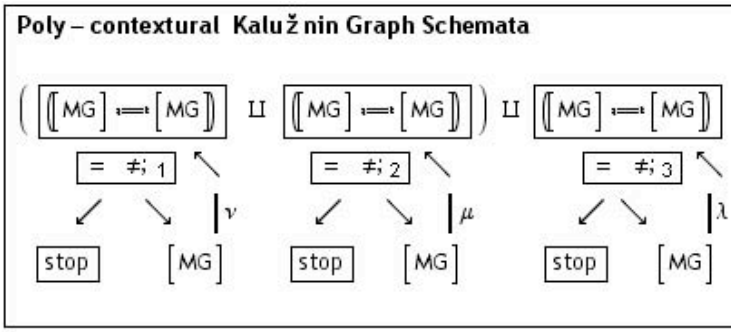
This possibility of changing contextures is not only necessary for the distinction of recursion and reflection but opens up the possibility of changing types of repeatability and comparability of morphogrammatic productions.

It is only a question of tedious applications of this distinction to show a new field of meta-mathematical considerations in which the simple direct decidability and non-decidability distinction gets transformed into an interplay over different systematic loci ruled by the so called super-operators.

3.2.5. Compositions and yuxtapositions of Recursion Schemata

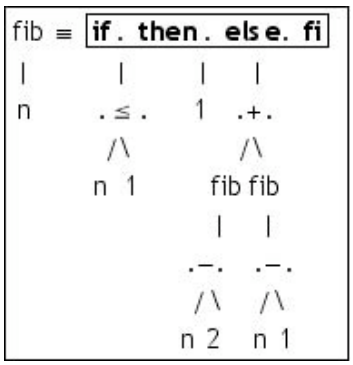
Iterative compositions of recursion schemes are well known. The difference of hierarchic and heterarchic combinations of recursion schemes has to be distinguished from intra-contextural applications of recursion schemata.
The whole approach of a dissemination of formal systems over the kenomic matrix has to be applied for recursion schemata again.



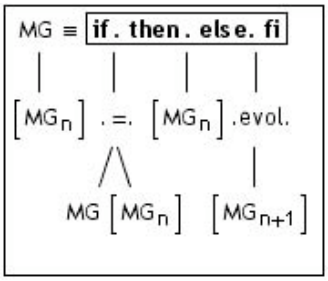


3.2.6. Kantarovic-tree analysis of recursion

The controll structure of the recursion for the Fibonacci numbers is : **"if.then.else.fi."**



If the encountered morphogram $[MG_n]$ is the searched morphogram $[MG_n]$, **then** accept the morphogram $[MG_n]$, **else** generate the searched morphogram with evol($[MG_n]$)=$[MG_{n+1}]$. **fi**.

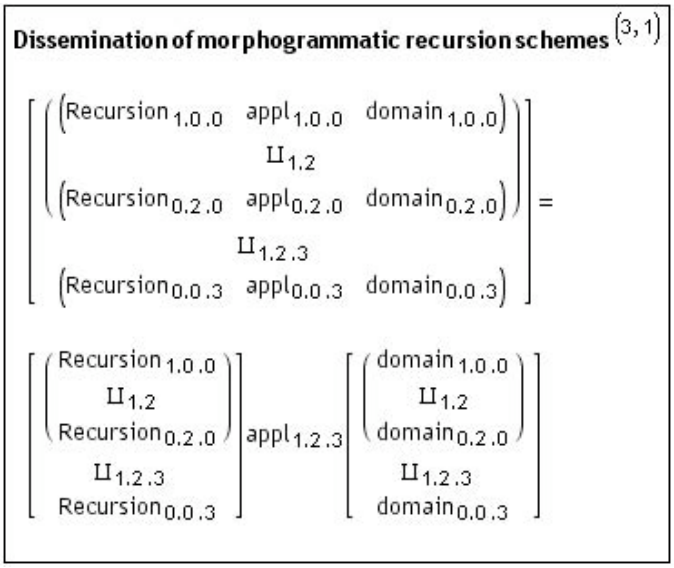


But this modeling is not yet telling anything about the chiastic structure of iterability in recursion.
The point is in the generation of the (new) morphogram: "**else** generate the searched morphogram with evol($[MG_n]$)=$[MG_{n+1}]$. **fi**." Because there is no need for an external alphabet, the new morphogram of the following step has to be generated (produced) by itself and cannot just be consumed from the pre-given resources.

3.2.7. Intergangeability of distributed recursion schemes

Morphogrammatic recursion schemes offer the possibility of distribution ruled by the different definitions of their iteration rules.

Recursion schemes and domains of recursion



3.3. Recursivity of morphogrammatic operations

3.3.1. Retro-Recursivity of coalition

An intuitive development of coalition building (addition) follows the following steps. The chosen modus of development is "*concatenative prolongation*". This holds for the next considerations too.

Example1.
[aba] + [ab] = {[abaab], [ababa], [abaca], [abaac], [ababc], [abacb], [abacd]}

$\text{coal}_{\text{evol}}([a\,b\,a], [a\,b]) =$

$[a\,b\,a\,a\,b]$, $[a\,b\,a\,a\,c]$,

$[a\,b\,a\,b\,a]$, $[a\,b\,a\,b\,c]$,

$[a\,b\,a\,c\,a]$, $[a\,b\,a\,c\,b]$, $[a\,b\,a\,c\,d]$

This approach is depending on the awareness, i.e. mental perception, of the *pattern* structure of the added morphogram [ab]. There is no specific algorithm offered yet to produce the coalition of morphograms formally.

A significant step to a formalization is introduced with recursion and the '*recursive*' coalition building on the base of monomorphies, and the *proviso* (context rules) which saves the pattern structure of the added morphogram during the calculation. All that is conceived principally in analogy to classical word arithmetics, i.e. string theory, albeit with the necessary transformation of the notions and methods to deal with morphogrammatics and their retro-recursive character.

Example2.

$\text{add}([\text{aba}], [\text{ab}]), \text{dec}([\text{ab}]) = ([\text{a}], [\text{b}]), \text{prov}(\nu)$

$[\text{aba}]$

$\downarrow$

$\longrightarrow [\text{aba}][\text{a}] \longrightarrow [abaa][a],\ [\text{abaa}][\text{b}],\ [\text{abaa}][\text{c}]$

$\downarrow \qquad : ([\text{abaa}][\text{a}]) \notin \text{prov}$

$\longrightarrow [\text{aba}][\text{b}] \longrightarrow [\text{abab}][\text{a}],\ [abab][b],\ [\text{abab}][\text{c}].$

$\downarrow \qquad : ([\text{abab}][\text{b}]) \notin \text{prov}$

$\longrightarrow [\text{aba}][\text{c}] \longrightarrow [\text{abac}][\text{a}], [\text{abac}][\text{b}], [abac][c], [\text{abac}][\text{d}].$

$: ([\text{abac}][\text{c}]) \notin \text{prov}$

Hence,

$\text{add}([\text{aba}], [\text{ab}]) = \{[\text{abaab}], [\text{abaac}], [\text{ababa}], [\text{ababc}], [\text{abaca}], [\text{abacb}], [\text{abacd}].\}$

Thus, the decomposition of the added morphogram into monomorphies has to take into account its morphic structure (Gestalt), i.e. to remember/restore, as the proviso of abstract retrograde concatenation and addition. Furthermore the monomorphic coalitions are determined by the *retro-gradness* of the concatenation operation involved in the operation of addition, i.e. coalition building.

Epistemological parenthesis

All that proofs that the "history- and time-dependence" introduced with the basic morphogrammatic operation of prolongation (succession, concatenation, continuation) is passed to the 'higher' levels of formalization and implementation of morphogrammatic operators.
Therefore, the whole of morphogrammatics, from its very first introductional characterizations and definitions, is structured by the trans-classic interplay of the retro/progression-diamond.

Again,*"Unlike a conventional ohmic component, the instantaneous resistance of the device depends on the* entire past history *of the voltage applied across it or current passing through it. In more simple terms, it is*

a resistor with memory and hence its name." (P.S. Georgiou)
Quantitative Measure of Hysteresis for Bernoulli Memristors
P.S. Georgiou, S. Yaliraki, M. Barahona and E.M. Drakaki
http://arxiv4.library.cornell.edu/abs/1011.0060

Memristic systems are as nano-physical systems structurally determined by *"time- and history-dependence"* of processuality. It therefore suggests itself that morphogrammatics is catching an important structural aspect of memristive systems which is definitively different to a quantitave and physical understanding of time and history, i.e. retro/pro-grade recurrence.

"However, the analysis of these models is mostly based on numerical temporal integration due to the absence of a general mathematical framework which is able to provide analytical solutions to the differential equations describing the dynamics of the system under consideration.

"Framework is based on the compliance of a general class of ideal memristor dynamics with Jacob Bernoulli's differential equation. The advantage of our formalism relies on the fact that systems governed by Bernoulli's dynamics can always be linearised. This provides a powerful and systematic methodology which can aid in the analysis, characterisation and design of such devices." (ibd, p.1/2)

The morphogrammatic approach, which is fundamentally *graphematic*, i.e. focused on its own scripturality instead of physical phenomena "outside" in the "real world", stands in no conflict with the more classical approach to describe quantitatively *"the input and output of the device"* by differential equations of Bernoulli dynamics. In respect to this classical quantitative approach, the morphogrammatic approach has to be considered as eminently "qualitative".

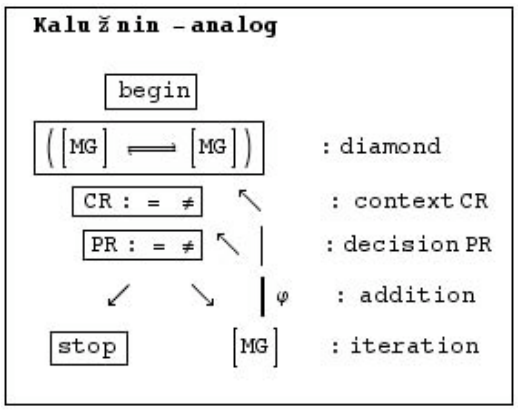
"Having analytical expressions describing the input and output *of the device can save up a lot of computational power and time and provide an important aid in the theoretical investigation and understanding of the device. We believe our framework can be used as the foundation of a general set of tools and methods, irrespective of what is the controlling* quantity*, for the design and analysis of individual components or memristors as part of larger networks of analog components."* (ibd., p. 10)
END of Parenthesis.

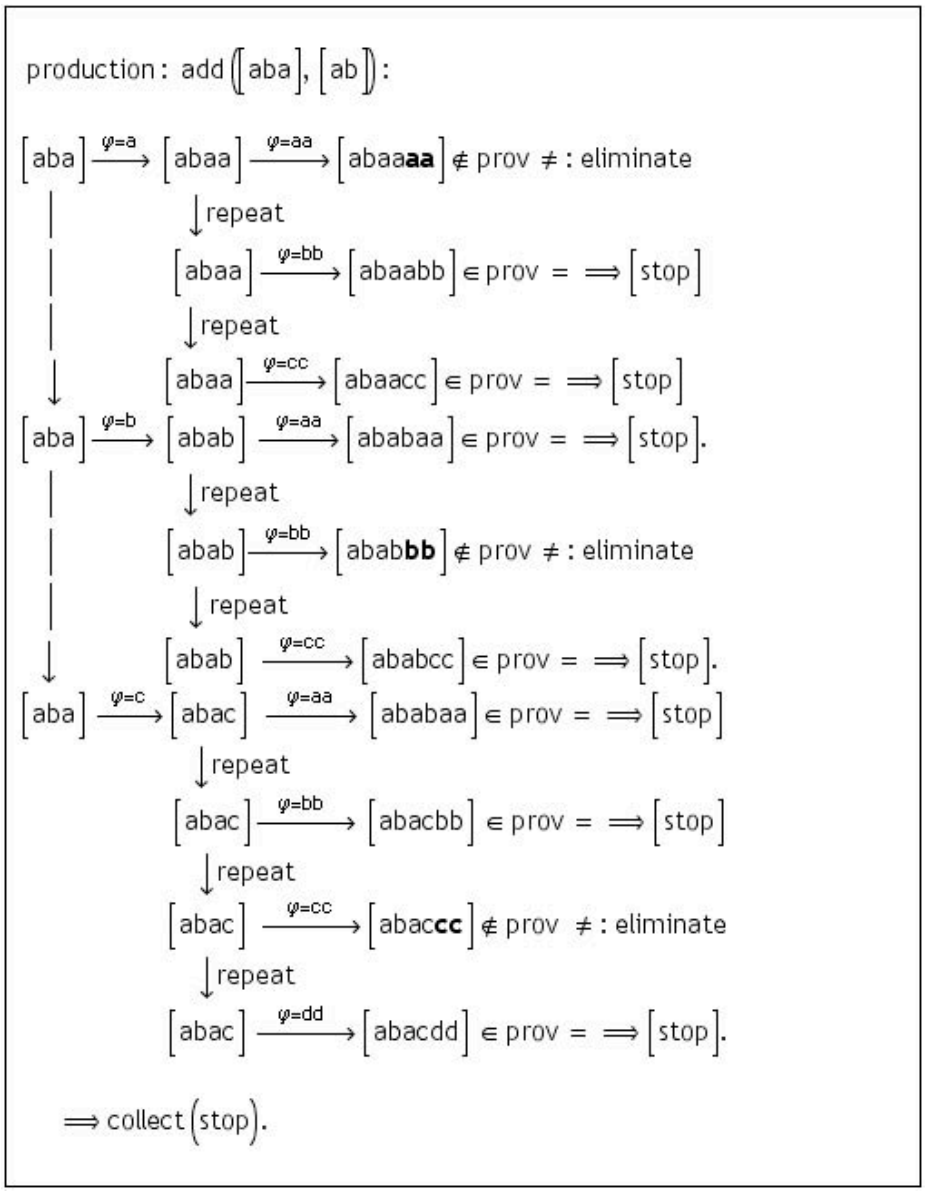
Therefore, the production "[aba][a][a]" is not accepted as a step in the addition of [aba] and [ab] because it is not fulfilling the proviso for restoring the *pattern* involved in the addition, i.e. [aa] has the ε-structure instead of the v-structure, and therefore violates the inherited pattern conditions.

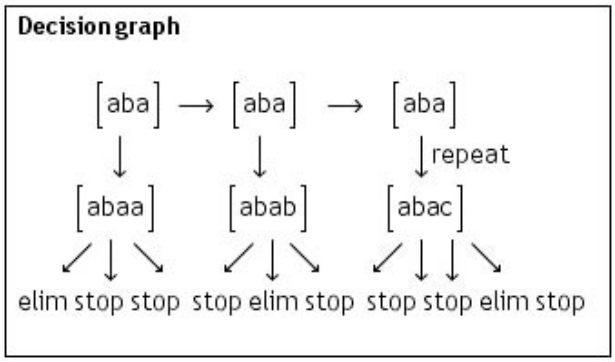
Strategically, a recursive rule, program, is producing all coalitions based on the set of monomorphies. Second, an algorithm has to eliminate all occurrences of the recursion which are not fulfilling the pattern proviso.

This can be avoided, i.e. formalized, by an implementation which is considering the retrograde concatenation and simultaneously the pattern condition of the addition.

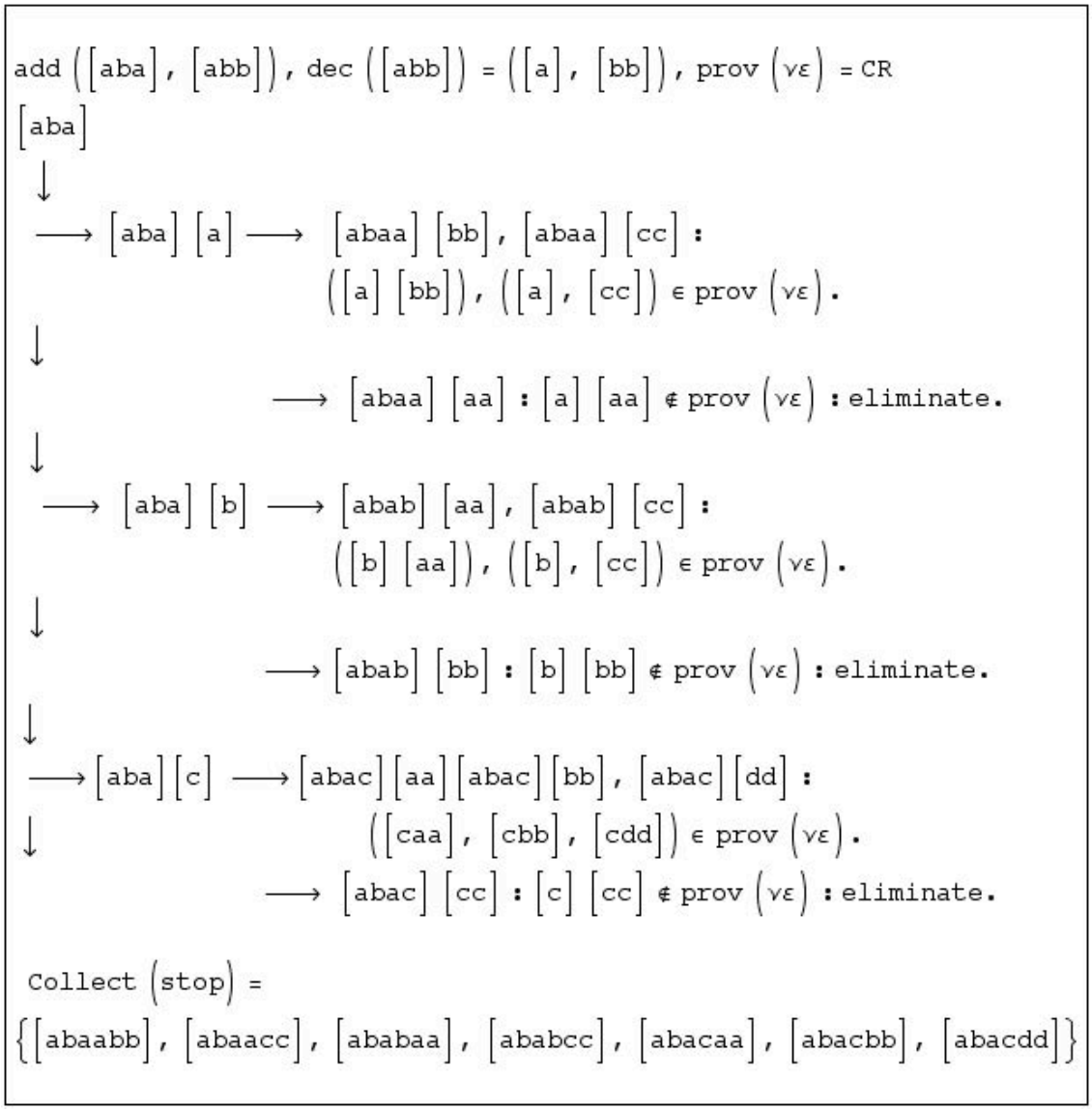
Hence, the retrograde recursion gets a meta-rule which controls the addition-procedure in respect of the structure of the added morphogram. This structure is called epsilon/nu-structure (ϵ/v-structure) for ε=equal, v= not-equal.


Kalužnin – analog
begin
([MG] ⟹ [MG]) : diamond
CR : = ≠ : context CR
PR : = ≠ : decision PR
φ : addition
stop
[MG] : iteration


production : add([aba], [ab]) :
[aba] —φ=a→ [abaa] —φ=aa→ [abaaaa] ∉ prov ≠ : eliminate
repeat
[abaa] —φ=bb→ [abaabb] ∈ prov = ⟹ [stop]
repeat
[abaa] —φ=cc→ [abaacc] ∈ prov = ⟹ [stop]
[aba] —φ=b→ [abab] —φ=aa→ [ababaa] ∈ prov = ⟹ [stop].
repeat
[abab] —φ=bb→ [ababbb] ∉ prov ≠ : eliminate
repeat
[abab] —φ=cc→ [ababcc] ∈ prov = ⟹ [stop].
[aba] —φ=c→ [abac] —φ=aa→ [ababaa] ∈ prov = ⟹ [stop]
repeat
[abac] —φ=bb→ [abacbb] ∈ prov = ⟹ [stop]
repeat
[abac] —φ=cc→ [abaccc] ∉ prov ≠ : eliminate
repeat
[abac] —φ=dd→ [abacdd] ∈ prov = ⟹ [stop].
⟹ collect([stop]).



Example 3.



Recursion formula for morphogrammatic addition

$$\text{add}\left(\left[\text{MG}\right],\ \emptyset\right) = \left[\text{MG}\right]$$
$$\text{add}\left(\emptyset,\left[\text{MG}\right],\right) = \left[\text{MG}\right]$$

$$\text{add}\left(\text{prov}\left(\left[\text{MG}_1\right],\left[\text{MG}_2\right]\right)\right) =$$
$$\text{add}\left(\text{prov}\left(\left[\text{MG}_1\right],\text{dec}\left(\left[\text{MG}_2\right]\right)\right)\right) = \text{add}\left(\text{prov}\left(\left[\text{MG}_1\right],\left(\text{mg}_1,\ \ldots,\ \text{mg}_n\right)\right)\right) =$$
$$\text{add}_n\ \left(\text{prov}_n\left(\ldots\left(\text{prov}_2\left(\text{add}_2\left(\text{prov}_2\left(\text{add}_1\left(\text{prov}_1\left(\left[\text{MG}\right],\left[\text{mg}_1\right]\right)\right),\left[\text{mg}_2\right]\right),\ \ldots,\left[\text{mg}_n\right]\right)\right)\right)\right)\right)$$

$$\text{With add}\left(\text{prov}\right) \equiv \text{addpr}:$$
$$\text{addpr}_n\left(\ldots\left(\text{addpr}_2\left(\text{addpr}_1\left(\left[\text{MG}\right],\left[\text{mg}_1\right]\right),\left[\text{mg}_2\right]\right),\ \ldots,\left[\text{mg}_n\right]\right)\right)$$

$$\text{add} \in \text{CR}:$$
$$\text{add}\left(\left[\text{MG}\right],\ \emptyset\right) = \left[\text{MG}\right]$$
$$\text{add}\left(\emptyset,\left[\text{MG}\right],\right) = \left[\text{MG}\right]$$

$$\text{add}\left(\left[\text{MG}_1\right],\text{dec}\left(\left[\text{MG}_2\right]\right)\right) = \text{add}\left(\left[\text{MG}_1\right],\left(\left[\text{mg}_1\right],\ \ldots,\left[\text{mg}_n\right]\right)\right).$$

All those simple results concerning the basics of morphogrammatics are new, and haven't found yet a proper treatment in earlier approaches, except in the 'hidden' algorithm of the ML implementations for simple concatenation-style addition and multiplication in the book "*Morphogrammatik*".

Such retrograde recursion or retro-recursion might also be called meta-recursion, reflection and meta-reflection, or reflexivity. The topics of meta-recursion are not the objects of recursiveness but the rules of recursion, i.e. the meta-rules, characterizing the rules of recursiveness.

Recursive rules for addition

Exmple1
add([MG], succ([MG])) =? succ(add([MG], [MG]))

add([ab], succ([a])) =? succ(add([ab], [a]))

add([ab], succ([a]) =
add([ab],([aa], [ab])) =
[abaa], [abbb], [abcc],
[abab], [abba], [abac],[abbc], [abca],[abcb], [abcd].

succ(add([ab], [a]))=
succ([aba], [abb], [abc])=
[abaa], [abab], [abac],
[abba], [abbb], [abbc],
[abca], [abcb], [abcc], [abcd].

Hence, add([ab], succ([a]) = succ(add([ab], [a])).

Individul situations
But there are trivially discrepancies possible.
If i!=k, j!=h, then the individual results may differ.

Case: i=j=a, h=k=b∈range([MG]):

$add_i([ab], succ_j([a]) = succ_h(add_k([ab], [a]))$, i,j,h,k∈range([MG])

$add_a([ab], succ_a([a]) = succ_b(add_b([ab], [a]))$,

$add_a([ab], [aa]) = succ_b(([abb]))$,

([abaa]) *!=* ([abbb]), for i=j=a, h=k=b∈range([MG])

Example2
add([MG], succ([MG]) =? succ(add([MG], [MG])

add([aba], succ([ab]) = succ(add([aba], [ab]))

add([aba], succ([ab]) = add([aba], ([aba], [abb], [abc])=

[abaaba], [ababab], [abacab],
[abbaba], [ababab],
[abaabc], [ababca], [abacba], [abacab].

succ(add([aba], [ab]))=
succ([abaab],[ababa], [abaac], [abaca], [ababc], [abacb], [abacd]) =

succ([abaab]) = [abaaba], [abaabc]
succ([ababa]) = [ababab], [ababac]
succ([abaac]) = [abaaca], [abaacb], [abaacd]
succ([abaca]) = [abacab], [abacac], [abacad]
succ([ababc]) = [ababca], [ababcb], [ababcd]
succ([abacb]) = [abacba], [abacbc], [abacbd]
succ([abacd]) = [abacda], [abacdb], [abacde].

3.3.2. Recursive definition of the reflector

A reflector is an unary operation on morphograms, it *reverses* the order of the monomorphies of the morphogram. This concept is presuming a *lexical order* of the monomorphies of a morphogram. Introduced for technical reasons only, there is no conceptual conflict involved. From a systematic point of view it has to be considered that morphograms are not necessarily ordered in a linear fashion.

A reversal of a morphogram refl([MG]) is a reversal of its monomorphies.

$[MG] = ([mg_1], [mg_2], \ldots, [mg_n])$:

$refl([MG]) = ([mg_n], [mg_{n-1}], \ldots, [mg_1]$

$refl_{trans}$:[MG_{junct}] --> [MG_{trans}], with refl(refl([MG])) != [MG].

$refl_{trans}$:([MG_{junct}] --> [MG_{junct}]) --> [MG_{trans}]

refl([MG]) = [MG]
refl(succ([MG])) = add(rev([Mg], [MG]))

refl([a]) = [a]
refl([ab]) = add(rev([a], [b])) = add([b], [a]) = [ba] = [ab].

add([b], [a]) = [ba], [bc] ==> [ab], [ab]==> [ab]

refl([abb]) = add(rev([a], [bb])) = add([bb], [a]) = [bba] = [aab].

refl([abb]) = add(rev([a], [bb])) = add([bb], [a]) = [bba], [bbc] = [aab].

refl(abbac]) = add(rev([a], [bb], [a], [c]))=
add(rev(rev([a], [bb]), [a]), [c])) =
add([c], (rev([bb], [a]), [a]))=
add([c], [a], [bb], [a]) = [cabba] = [abccb].

3.3.3. A simple application: Morphic stack

"*A stack is simply a linear data structure that stores a sequence of objects.*"

PUSH :
PUSH (abc) = PUSH(c) -> PUSH(b) -> PUSH(a).
POP:

Hence, it supports atomistic linear non-retrograde recursivity.

Morphic PUSH, MorphPUSH, is applied on monomorphies of morphograms, hence the stack has to keep its structure memorized, i.e. to fulfil the context conditions. It might be reasonable for presentation and calculations too, to divide the stack into different parts (loci) to place the involved monomorphies.

MorphPUSH ([abbcdda]) =
mPUSH([a])-->mPUSH([dd])--> mPUSH([c]) --> mPUSH([bb])--> mPUSH([a]).

[MG] = {[mg_1], [mg_2], [mg_3], [mg_4], [mg_1]:

[MG] = [aabbcdda] =

a	***b***	***c***	***d***	***a***
a	***b***	***o***	***d***	***o***
mg_1	mg_2	mg_3	mg_4	mg_1
loc_1	loc_2	loc_3	loc_4	loc_5
$stack_{1,1}$	$stack_{2,2}$	$stack_{3,3}$	$stack_{4,4}$	$stack_{5,1}$

[MG] =

mg_1	mg_2	mg_3	mg_4	mg_1

MorphPUSH([MG]) =
mPUSH([mg_1] --> mPUSH([mg_4]) --> mPUSH([mg_3]) --> mPUSH([mg_2])-->mPUSH([mg_1])

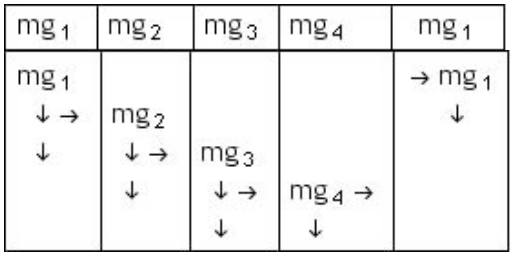
Example

MG1 MG2 MG3 add mult
empty--> PUSH MG1 --> PUSH MG2 --> PUSH MG3 --> add --> mult; postfix notation

MG1 = [ab], MG2 = [a], MG3 = [a]
empty--> PUSH [ab] --> PUSH [a] --> PUSH [a] --> add --> mult
PUSH [ab] --> PUSH [a] --> add : [aba], [abb], [abc]
--> PUSH [a]--> mult : [aba][a], [aba][a], [abb][a], [abc][a]

Mediation and monomorphies

Each locus gives space for a one-element recursive arithmetic. The rules between the loci are determining the mediation of recursive functions over monomorphies. Hence, the context rules for morphogrammatic recursion for morphograms are in fact the mediation rules between contextural recursion systems.



Every continuation in monomorphies has a representation in a continuation of a morphogram.
[abbcdda] --> [abbcccdda]:

Multi – successor systems for morphograms

$[MG] = [[mg_1], [mg_2], [mg_3], [mg_4], [mg_1]]$:

$succ_{1.4}([MG]) =$

$[succ_1([mg_1]), [mg_2], [mg_3], succ_4([mg_4]), [mg_1]]$

$[MG] = [abbcdda]$

$succ_{1.4}([MG]) = [succ_1([a]), [bb], [c], succ_4([dd]), [a]] =$

$[[aa], [bb], [c], [ddaa], [a]]$
$[[ab], [bb], [c], [ddaa], [a]]$
$[[aa], [bb], [c], [ddbb], [a]]$
$[[ab], [bb], [c], [ddbb], [a]]$
$[[aa], [bb], [c], [ddcc], [a]]$
$[[ab], [bb], [c], [ddcc], [a]]$
$[[aa], [bb], [c], [ddee], [a]]$
$[[ab], [bb], [c], [ddee], [a]]$.

$succ_{1.4}([MG]) = ?\ succ_{4.1}([MG])$

$succ_4(succ_1([MG])) = ?\ succ_{1.4}([MG])$

$succ_1([MG])$:

$[[aa], [bb], [c], [dd], [a]] =$
$[[ab], [bb], [c], [dd], [a]]$

$succ_4(succ_1([MG])) =$

$[[aa], [bb], [c], [ddaa], [a]], [[ab], [bb], [c], [ddaa], [a]]$
$[[aa], [bb], [c], [ddbb], [a]], [[ab], [bb], [c], [ddbb], [a]]$
$[[aa], [bb], [c], [ddcc], [a]], [[ab], [bb], [c], [ddcc], [a]]$
$[[aa], [bb], [c], [ddee], [a]], [[ab], [bb], [c], [ddee], [a]]$

Simple multi – successor systems

$$\text{succ}_{1.1.1.4.4.4}([MG]) = [[aaaa], [bb], [c], [ddddd], [a]]$$

$$\text{succ}_{1.1.1}([MG]) : [[aaaa], [bb], [c], [dd], [a]]$$

$$\text{succ}_{4.4.4}([MG]) : [[a], [bb], [c], [ddddd], [a]]$$

Remarks

This construction might be used as a distribution mechanism for arithmetical systems. Despite the parallelism of the distributed arithmetical systems there is still the *mediative* character of the whole pattern (morphogram) to be respected. Considering this observation the difference to classical multi-successor systems shall be obvious.

Second, additional to the separated but mediated parallelism, some kinds of interactional activities between distributed number systems are accessible.

3.3.4. Retro-recursivity of cooperation (multiplication)

Cooperation was modeled before as morphogrammatic multiplication.

The minimal mechanism of multiplication is depending on *retrogradnes* of the successor operation, *pattern* sensitivity for addition and specific *context rules* for multiplication.

Recursion schema for numeric multiplication

$$(\text{succ}, \text{add}, \text{mult}) \in CR :$$
$$\text{mult}([MG], \emptyset) = \emptyset$$
$$\text{mult}(\emptyset, [MG]) = \emptyset$$
$$\text{mult}([MG_1], \text{succ}([MG_2])) = \text{add}(\text{mult}([MG_1], [MG_2]), [MG_1])$$

Example 1

$$\text{mult}([a], \text{succ}([a])) = \text{add}(\text{mult}([a], [a]), [a])$$

$$\text{mult}([a], ([aa], [ab])) = \text{add}([a], [a])$$

$$[aa], [ab] = \text{add}([a], [a]) = [aa], [ab] : (CR = \text{retro})$$

Example2

$\mathrm{mult}\left(\left[\mathrm{ab}\right], \mathrm{succ}\left(\left[\mathrm{a}\right]\right)\right) = \mathrm{add}\left(\mathrm{mult}\left(\left[\mathrm{ab}\right], \left[a\right]\right), \left[\mathrm{ab}\right]\right)$

$\mathrm{mult}\left(\left[\mathrm{ab}\right], \mathrm{succ}\left(\left[\mathrm{a}\right]\right)\right) = \mathrm{add}\left(\left[\left[\mathrm{ab}\right], \left[\mathrm{ab}\right]\right)\right) =$
$\mathrm{mult}\left(\left[\mathrm{ab}\right], \left(\left[\mathrm{aa}\right], \left[\mathrm{ab}\right]\right)\right) = \mathrm{add}\left(\left[\left[\mathrm{ab}\right], \left[\mathrm{ab}\right]\right)\right)$

Addition

$\mathrm{add}\left(\left[\left[\mathrm{ab}\right], \left[\mathrm{ab}\right]\right)\right) =$
$\left[\mathrm{abab}\right], \left[\mathrm{abac}\right], \left[\mathrm{abba}\right], \left[\mathrm{abbc}\right], \left[\mathrm{abca}\right], \left[\mathrm{abcb}\right], \left[\mathrm{abcd}\right].$

Recursion : add

$\mathrm{add}\left(\left[\left[\mathrm{ab}\right], \left[\mathrm{ab}\right]\right)\right) = \mathrm{add}\left(\left[\left[\mathrm{ab}\right], \left[\mathrm{a}\right], \left[\mathrm{b}\right]\right)\right) =$

$\left[\mathrm{ab}\right]$
$\downarrow$
$\longrightarrow \left[\mathrm{aba}\right] \longrightarrow \left[\mathrm{aba}\right]\left[\mathrm{b}\right] \longrightarrow \left[\mathrm{aba}\right]\left[\mathrm{c}\right].$
$\downarrow \qquad\qquad \longrightarrow \left[\mathrm{aba}\right]\left[\mathrm{a}\right] \notin \mathrm{CR}$
$\longrightarrow \left[\mathrm{abb}\right] \longrightarrow \left[\mathrm{abb}\right]\left[\mathrm{a}\right] \longrightarrow \left[\mathrm{abb}\right]\left[\mathrm{c}\right]$
$\downarrow \qquad\qquad \longrightarrow \left[\mathrm{abb}\right]\left[\mathrm{b}\right] \notin \mathrm{CR}$
$\longrightarrow \left[\mathrm{abc}\right] \longrightarrow \left[\mathrm{abc}\right]\left[\mathrm{a}\right] \longrightarrow \left[\mathrm{abc}\right]\left[\mathrm{b}\right] \longrightarrow \left[\mathrm{abc}\right]\left[\mathrm{d}\right]$
$\longrightarrow \left[\mathrm{abc}\right]\left[\mathrm{c}\right] \notin \mathrm{CR}.$

Collect *add* : $\left[\mathrm{abab}\right], \left[\mathrm{abac}\right], \left[\mathrm{abba}\right], \left[\mathrm{abbc}\right], \left[\mathrm{abca}\right], \left[\mathrm{abcb}\right], \left[\mathrm{abcd}\right].$

Multiplication

Recursion : mult

$\mathrm{mult}\left(\left[\mathrm{ab}\right], \left(\left[\mathrm{aa}\right], \left[\mathrm{ab}\right]\right)\right) = \mathrm{mult}\left(\left[\mathrm{ab}\right], \left[\mathrm{aa}\right]\right), \mathrm{mult}\left(\left[\mathrm{ab}\right], \left[\mathrm{ab}\right]\right)) =$
$\mathrm{mult}\left(\left[\mathrm{ab}\right], \left(\left[\mathrm{aa}\right], \left[\mathrm{ab}\right]\right)\right) = \mathrm{mult}\left(\left[\mathrm{ab}\right], \left[\mathrm{a}\right], \left[\mathrm{a}\right]\right), \mathrm{mult}\left(\left[\mathrm{ab}\right], \left[\mathrm{a}\right], \left[\mathrm{b}\right]\right)) =$

$\left[\mathrm{ab}\right]$
$\downarrow$
$\longrightarrow \left[\mathrm{a}\right] \longrightarrow \left[\mathrm{ab}\right]\left[\mathrm{ab}\right] \longrightarrow \left[\mathrm{ab}\right]\left[\mathrm{ac}\right]$
$\downarrow$
$\longrightarrow \left[\mathrm{b}\right] \longrightarrow \left[\mathrm{ab}\right]\left[\mathrm{ba}\right] \longrightarrow \left[\mathrm{ab}\right]\left[\mathrm{bc}\right]$
$\downarrow$
$\longrightarrow \left[\mathrm{c}\right] \longrightarrow \left[\mathrm{ab}\right]\left[\mathrm{ca}\right] \longrightarrow \left[\mathrm{ab}\right]\left[\mathrm{cb}\right] \longrightarrow \left[\mathrm{ab}\right]\left[\mathrm{cd}\right].$

mult([ab], [aa]) = [abab], [ab ac];
mult([ab], [ab]) = [abba], [abbc], [abca], [abcb], [abcd].

collect *mult*:
[abab], [abac], [abcb]; [abba], [abca], [abbc], [abcd].

collect *mult* equal collect *add*, hence:
mult([ab], succ([a])) = add(mult([ab], [a]), [ab]).

3.3.5. Retro-gradness, recalling own histories

Auto-rekursiv: Synthetische retrograde Ausgliederung, „Wirklichkeitsnähe".
Die Progression erfolgt somit über einen retro-graden Umweg durch das Morphogramm hindurch.
Diese definiert den Grad der *simultanen* Parallelität ihrer Nachfolger." (Kaehr, 1982)

"Wiederholungen im Medium des Semiotischen sind immer Iterationen eines Repertoires von vorgegeben Zeichen. Das heißt auch, daß jede Iteration einen und nur einen jeweiligen Anfang hat, die Anzahl der Nachfolger ist dabei vorerst beliebig.
Rein formal ist allerdings ein Mehr-Nachfolger-System, etwa eine Wortarithmetik, immer auf eine übliche uni-lineare Arithmetik ohne formalen Verlust zurückführbar.
Ebenso ist zu beachten, daß eine Iteration unabhängig von ihrem Vorgänger vollzogen wird, sie ist atomar, entsprechend ihrem Zeichenrepertoire. Die Iteration hat keine Geschichte, sie ist nicht durch ihr
Vorher bestimmt.

"Rekursionen basieren auf Iterationen und Variablen eines Rekursionsschemas. Das Rekursionsschema definiert den Ablauf und den Typ der Rekursion. Die Objekte der Rekursion sind selber nicht rekursiv. Kenogramme bzw. Morphogramme sind in ihrer Wiederholungsstruktur Selbstabbildungen und keine Iterationen eines ursprünglichen Zeichenrepertoires. Kenogramme werden somit im Vollzug ihrer Selbstabbildung erzeugt und eröffnen damit nicht nur die Möglichkeit der PKL, sondern auch die der transklassischen Arithmetik." (Kaehr, Disseminatorik 1993)

"Die kenogrammatische Operation der Nachfolge dagegen wird nicht durch ein vorgegebenes Alphabet definiert, sondern geht aus von dem schon generierten Kenogramm. Jede Operation auf Kenogrammen ist "historisch" vermittelt. D.h. die Aufbaugeschichte der Kenogramm-Komplexionen räumt den Spielraum für weitere Operationen ein. Diese können nicht abstrakt-konkenativ auf ein vorausgesetztes Zeichenrepertoire zurückgreifend definiert werden, sondern gelten einzig retro-grad rekursiv bezogen auf die Vorgeschichte des Operanden.

Diese Bestimmung des Begriffs der Wiederholung als retro-grad rekursiv involviert vier neue Aspekte, die der Rekursion als rekurrierender Wiederholung, fremd sind: einen Begriff der *Selbstbezüglichkeit*, der *Transparenz*, des *Gedächtnisses* bzw. der Geschichte und einen Begriff der *Evolution* im Gegensatz zur abstrakten Konkatenation und Iteration." (Kaehr, SKIZZE, 2003)
http://www.vordenker.de/ggphilosophy/kaehr_skizze_36-120.pdf

3.4. Consequences

As a consequence, such avantegard approaches based on classical recursivity like Spencer-Brown's re-entry, Luhmann's self-referentiality, Heinz von Foerster main notions of second-order cybernetics, like recursive eigen-values, etc. are becoming obsolete. The whole machinery of information-processing with its hegemonial approach had been riducled before by the results of the just mentioned trends.
What still remains of interest for the design of a new paradigm of *artificiality* are the adventurous endeavours of Gordon Pask about chiastic figures and the philosophical speculations of Gotthard Gunther about proemiality, polycontexturality and kenogrammatics. Both are not yet in the focus of academic research. The reasons are obvious.

3.4.1. Self-referentiality, paradoxes and reflectional programming

Distributed recursion enables local reflexivity and its paradoxes and trans-contextural 'journey' of circular structure escaping reflectional paradoxes.

Each recursion scheme has its own type of recursivity and decision logic.
Each recursion scheme is connected with its neighbor systems by the operation of system change.
Other criteria of mediation are omitted at this point.

What does it mean that a recursive function is changing in the process of its application?

A recurrence relation is giving directly an answer:
$b_n = nb_{n-1}$: procedure
$b_0 = 1$: start
A recurrence relation is not changing in the process of its application. What is changing are the values of the function involved in the process of calculation.

This is classically demonstrated with the recursive formula for Fibunacci numbers (for integers). Again, what is changed is not the recursion scheme for *fib* nor the function *fib* but the numbers *n* as the values of the recursion. Hence, there is no such thing as a self-application of the recursion schema or the function on itself. Mathematicians lazy parlance on the base of proper formalisms has produced a bulk of academic dissertations on the side of humanities innocent for math.

function *fib* is:
input: integer n such that n >= 0
1. if n is 0, **return** 0
2. if n is 1, **return** 1
3. otherwise, **return** [*fib*(n-1) + *fib*(n-2)]

end *fib*

In other words, if there would be a self-application of the function, i.e. the recursion, onto itself, the pattern *"function, input, return, end"* would have to be changed too, and not just the values of the function "*fib*".
http://www.thinkartlab.com/pkl/lola/FIBONACCI.pdf
How are such metamorphic changes implemented in recursive morphogrammatics?

3.4.2. Is there an ontology for morphograms?

Ontology has some renaissance since the advent of the semantic web. Such ontologies are designed in a strictly classical fashion, and the principle of identity and identification of objects is paramount.

Because of the diamond structure of morphograms, morphograms are better characterized as non-identifiable self-referential patterns/processes.

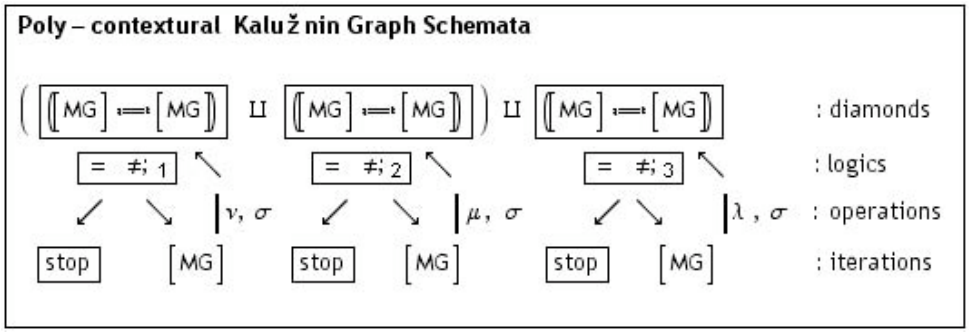
If we would like to start an ontology based on morphograms instead of signs and entities and their substance, attributes and meanings, we would have to start with 'objects' which are neither entities nor events, therefore don't offer any notions and methods to deal with it.

Morphograms are not identities or automorphisms but diamond-structured autopoietic 'phenomena'.
What kind of ontology could be designed on the base of such 'autopoietic' objects?

Java applets for memristors
http://www.falstad.com/circuit/e-mr-sine3.html
http://www.falstad.com/circuit/e-mr-sine2.html
http://www.falstad.com/circuit/e-mr-sine.html
http://www.falstad.com/circuit/e-mr-square.html
http://www.falstad.com/circuit/e-mr-triangle.html

3.5. Transjunctional recursion schemata

Up to now recursion schemata for morphogrammatic operators had been developed more or less independent of contextural distinctions. This is natural because morphograms had been studied as kenomic patterns or patterns of kenograms, and therefore distinctions such as intra-/trans-contextural of polycontextural complexions didn't yet apply. A step towards polycontextural recursion theory is risked with the introduction of the two modes of distribution of recursion schemata: the *hierarchical* and the *heterarchical*.
Hence, the common wording of a distribution over the kenomic matrix might be applied again.

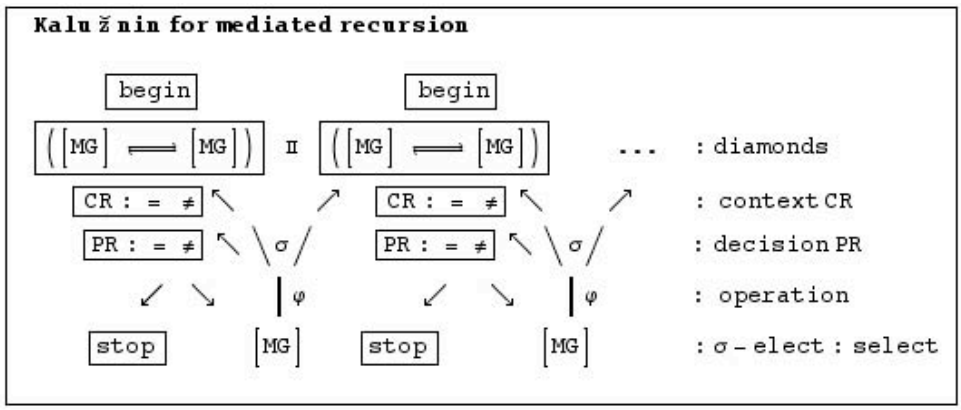


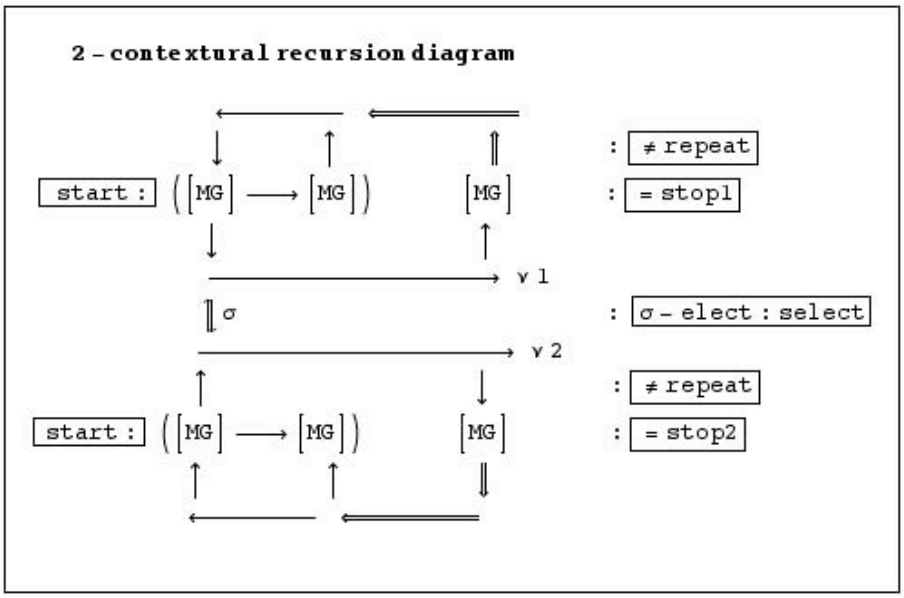
Therefore, the slogan "*iteration alters*" has to come into the game again. As it is well known, a formalization of iterability as an *intra*-contextural process and as a *trans*-contextural event is approached with the distinction of *elect* for contextures and *select* for the iteration of the operations of contextures.

In fact, each step of iteration in a recursion schema has to decide if it is operated by an *election* σ or by a *selection* σ' or by both at once. For a mono-contextural setting, this decision is omited because there is no choice offered. This in mind, a new reading of the papers "*poly-Lambda*" and "*Ruby to Rudy*" could be inspiring.
http://www.thinkartlab.com/pkl/lola/poly-Lambda_Calculus.pdf
www.thinkartlab.com/pkl/lola/From%20Ruby%20to%20Rudy.pdf

Polycontextural graph schemata for morphogrammatic recursion, thus, has to be reformulated in respect of the additional super-operators: *elect* and *select*. This was demonstrated for polycontextural systems with polycontextural logics but it holds for morphogrammatics too.

Recursion = (obj, logic, operation, iteration)
Recursion $^{(m, n)}$ = (obj, Diss$^{(m, n)}$(Recursion), elect, select).





http : // www.thinkartlab.com / pkl / lola / Transjunctional%20 Semiotics / Transjunctional%20 Semiotics.html

4. Reflection and reflectional programming

4.1. Concepts of morphogrammatic reflection

How could reflectional programming formally be conceived without recourse to optical metaphors and identity-preserving strategies of an observer and the observed?
Kaehr, ConTeXtures. Programming Dynamic Complexity, 2005 http://works.bepress.com/thinkartlab/20/
Amalgamation, meta-programming and reflection
There are two approaches which could radically simplify the task of sketching theories and programming languages for reflection (in a broader sense).

Reflection is not a special method like iteration and recursion, defined within a language but an interaction between languages but reduced to mutual mono-contexturality. A simple case is the interaction between an *object*- and a *meta*-language of the object-language in a single logical contexture.

"*The amalgamation of L and M is a conservative extension in the sense that no new theorems are provable in the amalgamation that were not already provable in either L or M.*" (Motta in: Maes, p. 224)
Again, *"We need to be able to describe processing strategies in a language at least as rich as that in which*

we describe the external domains, and, for good engineering, it should be the same language." L. Aiello in : Maes, p.245

Reflection is a specal form of meta-programming: *"Reflection is the ability of a program to manipulate as data something representing the state of the program during its own execution.*
There are two aspects of such manipulation: introspection *and* intercession.
Introspection *is the ability of a program to observe and therefore reason about its own state.*
Intercession [intervention, rk] *is the ability of a program to modify its own execution state or alter its on interpretation or meaning."* K. Czarnecki, p. 400

Stuff to read:
P. Maes, D. Nardi, Meta-Level Architectures and Reflection, 1988
K. Czarnecki, U.W. Eisenecker, Generative Programming, 2000, Chapter 10

Polycontexturality
There are different approaches to identify. Here it is enough to consider just two trans-classical possibilities. The amalmagation of languages or their meta-levels might be identified as *contextures* of a polycontextural logic and thus understood as distributed perspectives, points of view, general methods, and their mediation without amalgamating them into the uniformity of a "type-free" language. Hence, instead of a hierarchic order of object- and meta-language, an interactional heterarchy is supposed.
This topic got some treatment in previous papers about polycontexturality, meta-programming, reflection and introspection. (Kaehr, ConTeXtures)

Morphogrammatics
The opposite trans-classic approach is abstracting from any referentiality to domains of the languages and is considering their morphic constitution and their constitutional transformations only.
Morphogrammatics is not referring to any semantics or other models of reference. Hence, the morphogrammatic approach is neither opting for a meta- nor for a reflectional conceptualization.

What we can study is the self-transformation of morphogrammatics as responses to interactional perturbations. After that, a new attempt to connect this pre-semiotic endeavour might be re-established with contextural and semantic thinking.

A morphogrammatic system as a whole might emanate its complication or get into evolution by changing its complexity to stabilize or harmonize its organization after being involved into perturbations.

Inside a stable morphogrammatic system different forms of self-transformation can be studied.

Reflectors
One very specific study contains morphogrammatic *reflectors* which are transforming morphograms into each other by the operation of morphic reflection.
The advantage of this reflectorial approach is given by the fact that it is strongly related to polycontextural logic and its operators.

4.2. Morphogrammatics of reflection

4.2.1. Steps towards an reflexional interpretaion of morphogrammatics

Morphogrammatics of reflectional system-architectures might thematize the distribution of logical functors over different levels of the hierarchical system. In other words, logics in reflectional systems might distribute conjunctions, disjunctions and negations over different places of the hierarchical system of "object-" and "meta-"levels.

Morphogrammatics is abstracting from such logical properties introduced by distribution and is cutting out the very structure of such logical functors. Hence, a morphogram of logical operators is free of any syntactic and semantic properties of the actionality of the logical operators. This becomes more clear with the negational invariance of morphograms. (Kaehr, Mahler, Morphogrammatik, 1993)

In the inverse turn it seems to be reasonable and operatively of interest to see morphogrammatics as a "unifying" grammar of reflectional languages. Because morphograms are negation-invariant, antinomies based on negation, are becoming inexistent, there are also no other constructions known to build morphogrammatic antinomies.

Using operators like reflectors in morphogrammatics suggests that the dichotomy of operator and operand is reßstablished. Because the whole corpus of morphogrammatics is charcterised by retro-grade recursiveness were not even for the basic 'operations', like concatenation and addition, the strict dichotomy of operator and operands holds. An operator which is structurally depending on its operands is not anymore an operator in the sense of operator theory, i.e. logic and mathematic.

It is not just for reasons of convinience to use the common operator terminology. This choice of the usage of old terms in new contexts belongs to the deconstructive strategy of *paleonymics*.

A strictly *graphematic* theory of computationality, reflextionality, interventionality and anticipativity (ConTeXtures) would have to develop its strategies from the interaction between morphogrammatics and polycontextural logics, including semiotics and arithmetics.

4.2.2. Morphogrammatic reflectors of reflectional patterns

A morphogram of a logical conjunction and its multi-negations might be reflected by a simple reflector:
$refl([\wedge]) =_{MG} [\vee]$:

$$MG_{[\wedge]} = \begin{bmatrix} a & b \\ b & b \end{bmatrix}, \; MG_{[\vee]} = \begin{bmatrix} a & a \\ a & b \end{bmatrix}$$

$$refl: \begin{bmatrix} a & b \\ b & b \end{bmatrix} \dashrightarrow \begin{bmatrix} a & a \\ a & b \end{bmatrix}.$$

Composition of 3 morphograms, reflected by $refl^1$:

$MG^{(3,1)} = [MG_1 \amalg MG_2 \amalg MG_3] = [MG_1, MG_2, MG_3]$

dom([MG]) = head ([MG]),
cod([MG]) = tail ([MG]).

Matching conditions:
$cod([MG_1]) =_{MG} dom([MG_2])$
$cod([MG_3]) =_{MG} cod([MG_2])$
$dom([MG_1]) =_{MG} dom([MG3])$.

Example
$[MG_1] = [abbb]$
$[MG_2] = [bccc]$
$[MG_3] = [aaac]$

$$MG^{(3,1)} = \begin{bmatrix} a & b & a \\ b & b & c \\ a & c & c \end{bmatrix}$$

$refl^1(MG^{(3,1)}) = refl^1 [MG_1, MG_2, MG_3] = [refl^1(MG_1), MG_2, MG_3]$

$$refl^1 \begin{bmatrix} a & b & a \\ b & b & c \\ a & c & c \end{bmatrix} =_{MG} \begin{bmatrix} a & a & a \\ a & b & c \\ a & c & c \end{bmatrix}$$

$refl^1([\wedge \wedge \vee] = [\vee \wedge \vee]$.

Combined reflections of a composed morphogram:

$refl^{1.2}(MG^{(3,1)}) = [refl^1(MG_1), refl^2(MG_2), MG_3]$

$$\text{refl}^{1.2}\begin{bmatrix} a & b & a \\ b & b & c \\ a & c & c \end{bmatrix} =_{MG} \begin{bmatrix} c & c & a \\ c & b & b \\ a & b & a \end{bmatrix} =_{MG} \begin{bmatrix} a & a & a \\ a & b & b \\ a & b & c \end{bmatrix}$$

$\text{refl}^{1.2}([\wedge\ \wedge\ \vee] = [\vee\ \wedge\ \vee]$.

$\text{refl}^{1.3}(MG^{(3,1)}) = [\text{refl}^{1}(MG_{1}), MG_{2}, \text{refl}^{3}(MG_{3})]$

$$\text{refl}^{1.3}\begin{bmatrix} a & b & a \\ b & b & c \\ a & c & c \end{bmatrix} =_{MG} \begin{bmatrix} c & c & a \\ c & b & c \\ a & c & a \end{bmatrix} =_{MG} \begin{bmatrix} a & a & a \\ a & b & a \\ a & a & c \end{bmatrix}$$

$\text{refl}^{1.3}([\wedge\ \wedge\ \vee] = [\vee\ \oplus \vee]$.

This kind of reflector activity is changing the properties of the second morphogram by augmenting its complexity. Hence, we have a real and extremly simple concretization of the saying that a reflection is changing its process while running. Reflection therefore is not restricted by the modi of repetition, dualization and inversion which are not changing the structure of their pattern. Morphogrammatics is including all classical aspects of reflection too.

1. repetition: refl_{id}: [MG] --> [MG], with refl([MG]) = [MG],
2. dualization: refl_{dual}: $[MG_{junct}]$ -->$[MG_{junct}]$, with refl(refl([MG])) = [MG],
3. inversion: refl_{inv}: [MG -->[MG], refl([MG] = $\overline{[MG]}$,
4. transformation: refl_{trans}:$[MG_{junct}]$ --> [MG $_{trans}$], with refl(refl([MG])) != [MG].

The first examples are changing the structure of their pattern without leaving the classificational category of the morphograms of junctions (con- and disjunction). The fourth example is trans-forming the “junctional” pattern into a transjunctional pattern. That is, the transformation is producing a morphogram for transjunction on a logical level.
Kaehr, Mahler, Morphogrammatik, http://works.bepress.com/thinkartlab/15/

Reflexivity as a self-modifying process
The amalmagation approach to meta-programming, reflection and introspection might be classified as the type of reflexivity of an *active cognitive process*, while the morphogrammatic approach might be the type of *passive reflexivity of an unconscious process*. Interpretations in this direction had been published in the 70s (Kaehr, Materialien, 1973-75, in: G. Gunther, Idee und Grundriss, 1978)

*"Die Modellierung der unbewussten Prozesse (*Traumarbeit*) wird durch die Morphogrammatik geleistet. Morphogramme fungieren dabei als Hieroglyphen. Die überdeterminierten signifikativen Prozesse werden durch die polykontexturale Logik modelliert.*
Ein einfacher Operator der Kreativität (Verschiebung, Verkehrung, Verformung) ist der Reflektor. *Je nach dem Grad der Intensität und dem Typ der Umformung, die durch den Reflektor in der Logik erzeugt wird, lässt sich die Art seiner Einwirkung bestimmen.”* (ibd., p. 113)

Hence, there is an interaction between the logical system with its object- and meta-levels and its general morphic sub-structure. Because this level of symbolization is “*unconscious*” it might be connected to the “*passive mode of reflection*" (P. Hibbert).

Again, “The second mode of recursion is perhaps less well described in the literature because it is radically different from the “classic” conceptualization of reflexivity as an active *cognitive* process.
In contradistinction to this popular conceptualization, there are a number of authors that talk of *reflexivity as an unconscious process by which the process of reflection is itself modified.*” (P.Hibbert et al, p. 49)

Certainly, this is a very vague analogy, specially because morphogrammatics is introduced as being beyond “logical” dichotomies, binaries and similar distinctions. Hence Hibbert's classification of reflexivity with the dichotomy of active and passive modes is not applicable for morphogrammatics.

There might be a connection to the concept of a unconscious aspect of reflexivity to morphogrammatics (Kaehr, 1976) but there is no concept of an unconscious aspect for recursion in morphogrammatics.