— vordenker-archive —

Rudolf Kaehr

(1942-2016)

Title

Sketch of a Typology of Abstract Memristic-Machines

Archive-Number / Categories

2_52 / K11

Publication Date

Keywords

Memristics, Memristor, Interchangeability, Chiasm, Morphogrammtics, Polycontexturality

Disciplines

Cybernetics, Computer Sciences, Systems Architecture and Theory and Algorithms

Abstract

A typology of memristic machines is sketched. This sketch gives an overview and orientation to the paper "Towards Abstract Memristic Machines". It also intents to propose a concise systematization of the newly introduced terms and strategies to memristics and morphogrammatics. This sketch is introducing four types of sign-use for four types of machines of fundamentally different paradigms: 1. semiotic, 2. monomorphic, 3. polymorphic and 4. bisimilar abstract machines. Further definitions of abstract machines have to be based on those graphematic notational systems. A realization of such constructions of abstract machines, in contrast to existing abstract machines of the theory of automata, might be an interesting exercise for the reader.

Citation Information / How to cite

Rudolf Kaehr: "Sketch of a Typology of Absract Memristic-Machines", www.vordenker.de (Sommer Edition, 2017) J. Paul (Ed.), URL: http://www.vordenker.de/rk/rk_Sketch-of-a-Typology-of-Abstract-Memristic-Machines_2010.pdf

Categories of the RK-Archive

Sketch of a Typology of Abstract Memristic Machines

Some orientational attempts to "Towards Abstract Memristic Machines"

Rudolf Kaehr Dr.[@]
ThinkArt Lab Glasgow

Abstract

A typology of memristic machines is sketched. This sketch gives an overview and orientation to the paper *"Towards Abstract Memristic Machines"*. It also intents to propose a concise systematization of the newly introduced terms and strategies to memristics and morphogrammatics. This sketch is introducing four types of sign-use for four types of machines of fundamentally different paradigms: 1. semiotic, 2. monomorphic, 3. polymorphic and 4. bisimilar abstract machines. Further definitions of abstract machines have to be based on those graphematic notational systems. A realization of such constructions of abstract machines, in contrast to existing abstract machines of the theory of automata, might be an interesting exercise for the reader.

1. Preliminaries

History between holism and elementarism

Machine, automata, algorithms, procedures, interactions, etc. are all depending on the experiences and concept of iterability (Wiederholung des Alten/Neuen). A typology of abstract machines, therefore, depends on the different kinds, notions, conceptualizations and formalizations of the notion of iterability.

Philosophy developed interesting concepts of iteration, iterability and repeatability.
This might be observed during the western history of philosophy from Plato, Aristotle, Augustinus, Hegel, Kierkegaard, Husserl, Heidegger, Whitehead and Deleuze, Deridda and Samuel Weber. To mention some important achievements.

But a decisive scientific explanation of iteration was developed by mathematical logic, mainly based on the philosophy of the Vienna Circle. Especially Rudolf Carnap and John von Neumann had given profound mathematico-logical analysis of iterability. Important work was done by Skolem about recursivity before, and the attempts to develop algorithm theory ocurred.
From Skolem, Turing, Church, Post, etc. to Markov and beyond. The common ground of such attempts, i.e. sign theory, had been clearly stated by A. A. Markov with his principles of identification, separation and potential realizability.

Peter Wegner, Dina Goldin, Robin Milner, and others, made a radical progress to liberate such closed systems towards a theory of interactive algorithm and machines.
http://books.google.co.uk/books?id=0z5AyN-SwdQC

Nevertheless, what is common to all such concepts, and its more advanced explanations, is the *principle of identity.* The identity of the signs, the identity of the designer and the identity of the process of using signs.

On the opposite of such scientific rationality always was a desire for wholeness (Gestalt), non-identifiability, ambiguity, self-referentiality, flux and dynamics of fundamental axioms of reasoning and reality in Western thinking.
Only late, thinkers from the West discovered Chinese and Japanese approaches to non-axiomatic thinking.

An approach which is connecting and mediating both paradigms and trends of thinking, West and East, axiomatics and holistics, into a new rationality was proposed by the philosopher and cybernetician Gotthard Gunther (1900 -1984).

Polycontexturality and morphogrammatics is designed as a theory which is taking both aspects, holism and computalism, into account in a new way of thinking and computing in the mode of using and deconstructing signs.

Results which have been achieved might be summarized by a typology of sign-use, algorithms and machines.

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

Slogans: “the same is different”, “iteration alters”, “iteration of the new”, “repetition of the identical"

First- and second-order approaches to morphogrammatics

Keno- and morphogrammatics based on kenograms (kenoms) are modeled along the experiences of word arithmetic, word algebra or semiotic string theory. Hence, the objects of morphogrammatic operations in such an approach are *sequences* of kenoms, i.e. kenograms and morphograms. This approach might be called *first-order* thematization. It corresponds to the historical development of kenogrammatics and morphogrammatics introduced by Gotthard Gunther (1962/68) and further developed by Kaehr, Mahler, Niegel, Kronthaler, Toth. Therefore, the main criteria for equality of morphograms demands for the *same length* of keno-sequences. This is still an eminent restriction for a process of deliberating the general use of signs and its theory.

A possible *second-order* approach is based, not on kenogram sequences and their length, but on the *operators* of kenogrammatics and on monomorphies of morphograms instead of kenograms. It gets some inspiration from the category-theoretical turn, which separates decisively set theory from category theory. Set theory is studying the internal structure of sets (theories) and category theory is

studying the *interaction* between objects. Interactions are operations. Hence, abstractions have not to be based on sets and equivalence relations over sets but are possible now on operators. Abstractions over operators are *second-order* activities and their results are fundamentally different from first-order abstractions, relations and operations

This fundamentally different approach, applied on keno- and morphogrammatics, opens up results, which are strictly paradox from the viewpoint of first-order thematizations. A first striking result of such an application is the intriguing insight and construction of the possibility of the *sameness* of morphograms of different kenomic complication, i.e. different length.

Morphograms as such are in fact unconceivable. What might achieved is to observe and register the results of interactions with and between morphograms. Hence, different morphograms might give similar responses to interactions. This leads to a new concept of equivalence, similarity and bisimilarity:

Two morphograms are morphogrammatically equivalent if their parts (monomorphies) are indistinguishable.

This forms an operational and interactional or even interventional equivalence for all sorts of algorithms and machines.

Metaphors

Some metaphors might elucidate the journeys through the four types of morphogrammatic journeys.

„Einen Weg bahnen, z.B. durch ein verschneites Feld, heißt heute noch in der alemannisch-schwäbischen Mundart wëgen*. Dieses, transitiv gebrauchte Zeitwort besagt: einen Weg bilden, bildend ihn bereithalten. Be-wëgen (Be-wëgung) heißt, so gedacht, nicht mehr: etwas auf einen schon vorhandenen Weg hin- und herschaffen, sondern: Weg zu (...) allererst bringen und so der Weg sein.“* Martin Heidegger, Unterwegs zur Sprache, S. 261, 1959

Jedoch, was heißt Weg, einen Weg wählen, was heißt Unterwegssein?
Der Weg: weg von/Weg hin (w/W).
Das Wëgen ermöglicht Weg, Ziel und Unterwegssein.

Der Weg führt zum Ziel: „Am Ende ist alles gut“
Der Weg will gewählt sein: Pfade durch das Labyrinth
Der Weg ist das Ziel: „On the Road again“
Das Wëgen selbst wëgt den Weg. Der Weg wëgt sich und dich mit ein in die Be-Wëgung des Wegs.

Some metaphors resit translation into a latin-based language.

The four ways of ways

1. There is a path that leads to the aim.
The problem is solvable. What is in the focus is the aim, the solution, the goal, the end.
The path is pre-given but has to be found. Hence, a lot is presumed and pre-given: the start, the path, the field and the aim; and the gurantee that there is an end.

Problem solving
A computation is a unique *transition* from a start to an end according some stable rules.
This corresponds with the *semiotic* concept of computation and abstract machines.

2. The path has to be chosen
Decisions have to be made on this type of journey to find a path through the labyrinth leading to the goal.
Pre-given is here the start and the aim. The field of paths, the singular labyrinth, is pre-given too. But the path of the journey has to be chosen, found, elaborated, to find the aim. Questions of optimization of the length and character of the path, like efficiency and elegance, have to be decided.

Optimization of problem solving in the framework of a stable contexture of a hidden labyrinth
A computation is a simple *transition* from a start to an aim according a set of stable rules applicable in a stable space of solutions.
This corresponds with the *monomorphic* concept of computation and abstract machines.

3. The journey is the aim
The start and aim might be given, but only as a hidden motive to be on the road. What is still pre-given for this type of journeys is the traveler herself and the field of possible path. It's about the fun to be on the way, orthogonally and transversal, between different labyrinths. Hence the journey might be on a path that is not moving away from the beginning nor towards an ending.

Elaborating possibilities of paths between discontextural labyrinths
A computation is a complex *transition* from an arbitrary start at a place in a contexture to an other arbitrary aim of another contexture according to a changing set of interacting stable rules. Hedonistic panalogy.
This corresponds with the *polymorphic* concept of computation and abstract machines.

4. The journey is 'journing'
The journey itself is defining the path and the traveler, and therefore the field of possible paths. There is no start and no aim necessary, because the field of paths is pre-given neither.

Co-creation of computation, user, labyrinth and paths
A co-creative computation is a transition of all the constituents of computation into itself.
This corresponds with the *bisimilar* concept of co-creative computation and abstract machines.

2. Typology of abstract machines

Also the field of memristics and abstract memristic machines is not yet well defined

and still in progress, a kind of a typology as a guide and orientation might be proposed. This scheme of orientation might quickly be transformed in the process of its application. Nevertheless, the following distinctions shouldn't be overlooked.

The leading distinctions might be "*construction/behaviour"* and *"hidden/visible"* formalized in coalgebras and algebras. Algebras are initial, coalgebras final. Again, this distinction is not in the sense of morphogrammatics and has to be deconstructed. Morphogrammatic complexions are neither terminal nor initial, they are interactionally both at once. Encounters are not experiments or (uni-directional) interactions but interactional, reflectional and interventional.

Nevertheless, those distinctions and their formalizations shall be used as along as they are of help.

A first *cut of* from memristic machines, as speculated here, has to be made between the *abstract machines* of the theory of automata and *quantum machines*, dealing with qbits inside the framework of general Turing machines. A comparison between those three, and others, might happen at another place. The term "abstract" for "abstract memristic machines" refers to the *conceptual* character of the exploration, in contrast to the nano-technological and engineering attempts of the endeavour.
Another series of "cut offs" to follow are with Chemical Abstract Machines (Berry, Boudol), Molecular Abstract Machines (Adleman), DNA Abstract Machines (Paun, Rozenberg, Salomaa) and many others.

2.1. Types of memristic machines

2.1.1. Types of memristic realizations

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

multiple or single operators, poly – or monomorphy, complex or simple monomorph, atomic signs, self – referential recursion

Different types of abstract machines are depending on the way they are using their basic "material", i.e. as signs or marks, or as kenograms or as monomorphies or as polymorphies to generate their type of abstract machine. There might be other possibilities, not thematized in this paper. Especially the definition of equality, equivalence, sameness and bisimularity plays a crucial role to determine the type of sign-economies, i.e. abstract machines.

Because of the property of *self-referentiality* of the non-semiotic systems, a close connection to different types of memristive systems and machines is opend up. Such a self-referentiality of the behaviour of memristive systems differs from the concept of recursivity. Recursivity is a property of algorithmic systems based on stable alphabets, hence, ideal for the ddefinition of classical abstract machines, characterizing their transition function recursively. Nevertheless, methods of recursive function theory may be applied to the study of memristive systems.

The most complex sign-use is labeled as "*bisimularity*" where the topic of sign-usage is defined by an interplay between a *multitude* of operators on complex *polymorphic* 'objects' (monomorphies) and their sameness is understood as a bisimilarity between the operators.

The following types are understood as a kind of reductions from the bisimular type. Positively, the other types are focusing on other aspects of machinal realizations than the type of complete bisimilarity.

Polymorphic systems are containing all properties of "bisimilarity" except that the multitude of operators is reduced to the application of a single operator to define complex polymorphic, self-referential sameness.

Monomorphic systems are containing all attribute of the "polymorph" type, except that the poly-morph objects are reduced to the involvement of monomorph object to define single operator, simple, self-referential monomorphic sameness.

Semitotic equality is then a reduction from the monomorph type, eliminating monomorphy for atomicity, and eliminating any retro-grade recursivity to define single operator, simple atomic equivalence.

Corrado Böhm about ambiguity

"Theory is knowledge of the truth; a formalism is based on a set of signs or symbols (concrete tokens) whose assemblages obey very precise formation rules and whose features mimic those of more abstract concepts; constructivity, today, means computability or representability inside a computer.

"About eighty years ago M. Schönfinkel defined a family of entities (combinators) whose main property was that, combined together, they could act as operators as well as operands. This dichotomy was solved in an impartial way, depending only on their mutual position.

"Our proposal is to reintroduce ambiguity up to a certain extent, following a principle which has always been successful each time it has been introduced in computer science (and perhaps not exclusively): the "lazyness" principle (see lazy evaluation lazy compilers, etc.). To apply lazyness means to delay the decisions until the last moment. We propose for instance to delay the decision of what is the meaning of a symbol to the moment where that symbol is written down together with other symbols."

Corrado Böhm, INFORMATION PROCESSING BY AMBIGUOUS FORMAL ENTITIES
http://www.neuralcoding.org/workshops/2003/aullapdf/56.pdf

2.1.2. Equality for semiotic machines

In contrast to the complex multi-operator type of bisimilarity, the semiotic approach is realized in the mode of a *single-operator* strategy, which is a reduction from the multiple operator approach of bisimilarity, and with a reduction of the monomorphies to *atomic* signs or marks and a *simple* zero-time structure with repetition of the signs in linear time.

The single operator is "*concatenation*" or dual "*substitution*".

The operational set of the operator is an alphabet of *abstract signs*.

Abstract semiotic systems as they are involved in algorithm theory are perfectly characterized by principles of identification, separability and potential iterability. A hidden axiom is: Semiotic iterations don't alter.

All this is not working without producing some paradoxes. A sign cannot be separated if it is not identified, and a sign can not be identified if it is not separated. Hence, both are mutually depending on each other and producing a definitional circularity. The same holds for the principle of potential realizability. Its Iterability is not possible without identification, and identification is not possible without iteration. This antinomical situation repeats itself with the definition of an abstract sign. Elementary and abstract signs are defining each other mutually. It is a decision to stop such circularity and to establish a hierarchy of notions. On the base of a decisional interruption of circularity, semiotics and therefore the theory of algorithm, can start as the fundamental discipline of writing, calculating and computation, i.e. of abstract machines.

Following A. A. Markov, the principles are:
Principle of identification
Principle of separation
Principle of potential realizability.

„5. Elementary signs are signs that we shall consider as **not having parts**. The content of this concept depends upon the conventions that are assumed. (...)
6. In simultaneous consideration of any two elementary signs, we determine whether they are the same or different. These concepts are also conditional.
7. The possibility of determining when two elementary signs are the same permits us, applying an **abstraction of identification**, to speak of two identical elementary signs or of one and the same elementary sign. On this basis, we introduce the concept of an **abstract elementary sign**, that is, of an elementary sign, considered up to identity. Concrete elementary signs will be considered as representatives of the corresponding abstract elementary sign. Two concrete elementary signs represent one and the same abstract elementary sign if and only if they are identical.
8. Lists of elementary signs are called **alphabets**. We shall call two alphabets equal if every elementary sign appearing in the first alphabet is identical with a certain elementary sign appearing in the second alphabet, and conversely. Alphabets considered up to equality will be called abstract alphabets."
11. Another abstraction, (...), is **abstraction of potential realizability.**
This consists in departing from real limits of our constructive possibilities and beginning to discuss arbitrarily long abstract words as if they were constructible. Their realizability is potential: their representatives could be practically realized if we had at our disposal sufficient time, space, and materials." A. A. Markov
(cf., http:// www.thinkartlab.com/pkl/media/SKIZZE-0.9.5-medium.pdf)

Stability of the notational system
During the process of computation, the semiotic base set, i.e. the alphabet and the semiotic rules, are not changing at all.
This fundamental stability principle for rules and "base sets" doesn't rule anymore for memristic machines.

Inalterable Base Set

"While the base set can change from one initial *state to another, it does not change* during *the computation. All states of a given run have the same base set. Is this plausible?*
There are, for example, graph algorithms which require new vertices to be added to the current graph. But where do the new vertices come from? We can formalize a piece of the outside world *and stipulate that the initial state contains an infinite naked set, the* reserve*. The new vertices come from the reserve, and thus* the base set does not change during the evolution.*"* (Y. Gurevich)

Is this plausible? For semiotics, yes; for morphogrammatics, no!

2.1.3. Sameness for monomorphic machines

A further reduction from the complex bisimilarity type of machines happens with the *single-operator* strategy where complex monomorphies are reduced to *simple* unary monomorphies and the complex self-referential time- and history-dependence is realized as a simple *self-referential* time succession.

The single elementary operators are “monomorphic evolvements” or dual “monomorphic concatenation”, the
operators of melting and de-melting, and operator of chaining, and de-chaining. All operators have to take into accont the “holistic” charater of morphogrammatics, thus arbitrary inventions of operators have to be critically tested against the holistic criteria.

For monomorphic sameness, the semiotic principles of separation and identification becomes second-order principles:
The principle of identification and separation of signs becomes the principle of the identification and separation of the *operations* of identification and separation.

Principle of monomorphic identification
Morphograms consists of monomorphies, and not of signs.
Monomorphies are having parts. Hence, they are neither concrete nor abstract signs, i.e. the type/token difference is not leading anymore the use of signs (kenograms).
Monomorphies might overlap or be separated. As identified patterns they depend on the interpretation of the operation of identification.
In fact, morphograms don’t exist, they become observable by the abstraction of enaction.
Hence, for morphograms, the principle of identification becomes a *principle of enaction.*

Principle of monomorphic separation
Monomorphies might overlap or be separated.
Overlapping or separation is not a property of monomorphies as objects of knowledge but a property of operators, and therefore depending on the position of the designer of morphograms in a positional system for generating morphograms.
There is no separation without enacting differences that are generating monomorphies of a morphogram.

Principle of monomorphic self-referential potential realizability.
Morphograms might be considered by the principle of potential realizability as arbitrarily long. But each step of such a potentiality is retrograde involved with the previous existing, i.e. just constructed, morphogram.

Therefore, the principle of potential realizability becomes a principle of concrete realizability based on the iterability of self-referential operations. As holistic, morphograms are finite patterns. The concept of potential infinity, which is suposed by the principle of potential realizability, is not leading anymore.

Alterable Base Set
Because ***"iterations alter"*** for monomorphic machines, the *"base set"*, i.e. the constituents of the monomorphic system, are alterable in the frame of simple monomorphies. As pointed out elsewhere, morphogrammatics is not based on alphabets, thus there are in fact also no rules based on alphabets.

Technical realizations, exploring, enabling or enacting monomorphic "fields" of memristive computation of simple monomorphic structure might be realized by multi-sorted *multi-layered crossbar* systems. A mechanism of "*alterability*" occurs for memristive systems as a chiastic interaction between the memory- and the computational functionalities of memristors.

2.1.4. Polysemy for polymorphic machines

Such a lively constellation, like bisimilarity, gets reduced by a *single-operator* strategy, where *complex* monomorphies are involved and time- and history-dependence is still realized in a complex s*elf-referential* time-dynamics.
Additional to the properties of the monomorphy-type, polymorph machines as a complex field of monomorphies, are creating a complex self-referential structuration of time-events.

Such complexions of different monomorphies are enabling different kinds of paths, i.e. journeys, for chiastic transformations and metamorphosis with aspects of reflectionality, interactionality and interventionality between arrays of monomorphies.

Because ***"iterations alter"*** in a complex way for polymorphic machines, the *"base set"*, i.e. the constituents of the polymorphic system, are alterable during a run in the frame of complexions of monomorphies, i.e. polymorphies too.

Corrado Böhm's lazy solution of ambiguity
"Theory is knowledge of the truth; a formalism is based on a set of signs or symbols (concrete tokens) whose assemblages obey very precise formation rules and whose features mimic those of more abstract concepts; constructivity, today, means computability or representability inside a computer.

"About eighty years ago M. Schönfinkel defined a family of entities (combinators) whose main property was that, combined together, they could act as operators as well as operands. This dichotomy was solved in an impartial way, depending only on their mutual position.

"Our proposal is to reintroduce ambiguity up to a certain extent, following a principle which has always been successful each time it has been introduced in computer science (and perhaps not exclusively): the "lazyness" principle (see lazy evaluation lazy compilers, etc.). To apply lazyness means to delay the decisions until the last moment. We propose for instance to delay the decision of what is the meaning of a symbol to the moment where that symbol is written down together with other symbols."

Corrado Böhm, INFORMATION PROCESSING BY AMBIGUOUS FORMAL ENTITIES
http://www.neuralcoding.org/workshops/2003/aullapdf/56.pdf

The action of an *application* as an operator never becomes an operand in CL! That's a funny situation! We can read everywhere that combinators are self-applicative, and that the distinction of operator and operand doesn't hold anymore; at least not in the strict sense as in other mathematical theories.

Böhm insists on this *simultaneity* of operators and operands *"they could act as operators as well as operands".* Which is emhasizing a "type-free", i.e. unified unique world (of formal distinctions). This is certainly correct, and he emphasizes that the change of their role depends on the *position* they take in a calculus. Furthermore, he weakens the strict simultaneity of the combinators as playing the role of operator and operand with the hint to "lazyness": *"To apply lazyness means to delay the decisions until the last moment."* This is a clever strategy for programming, say in functional languages, but not really helpful for conceptual analysis of the simultaneity of the two parts of the fundamental dichotomy of operator and operand. Where multitude is obvious, its acceptance might be procrastenated to the end of the world. Hence, the unity of the worlds is saved; at least for the believers in Church.

Technical realizations of a chiastic ambiguty of operator and operand, i.e. of logical element and memory-function, might be achieved by distributed discontextural *poly-layered crossbar* constructions of memristive computational systems.

2.1.5. Interactionality for bisimilar machines

Two machines are bisimilar if their visible behavior is similar, i.e. if their visible behavior cannot be distinguished by observations. Their hidden structure might be disimilar. The hidden structure of bisimilar machines is accessible to the designer of the machine but not to the observer, which might be the designer too, who is observing the bahavior of the machine. Therefore, at least two positions, the designer and the observer, have to be involved to realize morphograms and morphogram-based memristic machines. Between designer and observer a complex interplay takes place.

This situation holds for the *multi-operator* strategy where *complex* monomorphies are involved and the time- and history-dependence is realized in a complex *self-referential* time-dynamics. The abstraction of bisimulation of two machines happens on the level of the *operators* of the different machines and not on the level of their objects as in the following cases.

Bisimilarity is incorporating all features of polymorphy but interacting simultaneously with a multitude of operators.

Identification, separation, i.e. enaction and potential realizability are defined on the interplay of different morphogrammatic operations, like evolvement and cooperation, melting and de-melting, chaining and de-chaining.

Not only the "base sets" for morphograms, i.e. mono- and polymorphies, are alterable but the "*base rule set*" is alterable too for bisimilar machines. Because interactionality alters in a complex way for polymorphic machines, the operational

constituents of polymorphic systems, polymorphies and operators, are alterable in the frame of the interplay of interacting operators.
Such operational alterability is not possible for semiotic systems, simply because semiotics is operationally based on one and only one operator per operation.

Technical realizations might be achieved by a *complex interplay* between discontextural *poly-layered crossbar* constructions of memristive memory/computational systems.

2.2. Polymorphic operational bisimularity

Multiple operator strategy with complex monomorphies and complex history-dependence realizing complex anti-dromic self-reference between discontextural constellations.

2.2.1. Characterization of morphogrammatic bisimilarity

table: multiple poly complex self-ref

Polymorphy:

$$M = \left(R \middle| \overleftarrow{r}\right) ::$$

$$M = \left[M \middle| r_1 r_2 r_2 r_1\right]$$

$$M = \left[M \middle| r_1 r_2 r_2 r_1\right] : \begin{pmatrix} M^1 \middle| r_1 \\ M^2 \middle| r_{2.2} \\ M^1 \middle| r_1 \end{pmatrix} = \begin{pmatrix} M^1 \middle| r_1 \\ \amalg \\ M^2 \middle| r_{2.2} \\ \amalg \\ M^1 \middle| r_1 \end{pmatrix} = M^{(1,2,1)}$$

Complex self-referential evolvements (time-structure)

$$\text{evol}\left(\left[M \middle| r_1 r_2 r_2\right]\right) \Longrightarrow$$

$$\left(\begin{array}{l}
\text{evol}_{r_1}\left(\left[M \mid r_1 r_2 r_2\right]\right) = \left[M \mid r_1 r_2 r_2 r_1\right] \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \amalg_{1.2} \\
\text{evol}_{r_2}\left(\left[M \mid r_1 r_2 r_2\right]\right) = \left[M \mid r_1 r_2 r_2 r_2\right] \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \amalg_{1.2.3} \\
\text{evol}_{r_3}\left(\left[M \mid r_1 r_2 r_2\right]\right) = \left[M \mid r_1 r_2 r_2 r_3\right] \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \amalg \\
\text{evol}_{r_{1.1}}\left(\left[M \mid r_1 r_2 r_2\right]\right) = \left[M \mid r_1 r_2 r_2 r_1 r_1\right] \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \amalg_{1.2} \\
\text{evol}_{r_{2.2}}\left(\left[M \mid r_1 r_2 r_2\right]\right) = \left[M \mid r_1 r_2 r_2 r_2 r_2\right] \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \amalg_{1.2.3} \\
\text{evol}_{r_{3.3}}\left(\left[M \mid r_1 r_2 r_2\right]\right) = \left[M \mid r_1 r_2 r_2 r_3 r_3\right]
\end{array}\right)$$

Existence

$\exists\, A, B\colon A \neq_{MG} B \Longrightarrow A =_{BIS} B$

$\neg \forall\, A, B\colon A \neq_{MG} B \Longrightarrow A =_{BIS} B.$

Multiple operators and bisimulation

Bisimilarity of A and B

$< op_1, op_2\, op_3, op_4 >:$

$$\left(\left[M\middle|r_1 r_2 r_2 r_1\right], \left[M\middle|r_1 r_2 r_1\right]\right) \longrightarrow \left(\left[M\middle|r_1 r_2 r_2 r_1\right], \left[M\middle|r_1 r_2 r_1\right]\right):$$

$$\begin{pmatrix} \left[M\middle|r_1 r_2 r_2 r_1\right] & \xrightarrow{EVk} & \left[M\middle|r_1 r_2\right]\left[M\middle|r_1 r_2\right] \\ \Vdots & Vs \times VK & \Vdots \\ \left[M\middle|r_1 r_2 r_1\right] & \xrightarrow{EVs} & \left[M\middle|r_1 r_2\right]\left[M\middle|r_1 r_2\right] \end{pmatrix}$$

$$\Longrightarrow$$

$$\left[M\middle|r_1 r_2 r_2 r_1 r_1\right] =_{BIS} \left[M\middle|r_1 r_2 r_2\right].$$

$< op_1, op_2\, op_3, op_4 > = < Vs, Vk, EVk, EVs >$

$$A = \left[M\middle|r_1 r_2 r_2 r_1\right]$$

$$B = \left[M\middle|r_1 r_2 r_1\right]$$

$$C = \left[M\middle|r_1 r_2\right]$$

2.2.2. Bisimilarity and interchangeability

A poly-categorical thematization of monomorphy-based morphogrammatic equivalence shall be introduced.
How might this morphogrammatic interaction be caught by poly-categories and interchangeability?

Both type of interactions are structurally *discontextural* (disjunct) but mediated, i.e. $\mathcal{U}_1 \cap_{1.2} \mathcal{U}_2 = \emptyset$,
with Vk, EVs$\in \mathcal{U}_1$ and Vs, Vk $\in \mathcal{U}_2$. Both, $\mathcal{U}_1$ and $\mathcal{U}_2$ are universes (contextures) of a polycontextural theory.
Both interactions, *concatenation* (Vk) and *melting* (Vs) as well as de-concatenation (EVk) and de-melting (EVs) are holding simultaneously, therefore they have to be mediated ($\amalg_{1.2}$) to stay in the game, i.e. $\mathcal{U}^{(2)} = \mathcal{U}_1 \amalg_{1.2} \mathcal{U}_2$.
Hence, a formalisation as an *interchangeability* of composition and mediation in respect of the operations (Vk, Vs, EVk, EVs) applying to A and B seems natural.

EVk(A) ~= EVs (B) ==> Vs(EVk (A)) ~= Vk(EVs (B))

Example:

$$\left(\left[\mathrm{MG}\right], <\mathrm{op}>\right)^{(2)}:$$

$$\mathcal{U}_1 \bigcap_{1.2} \mathcal{U}_2 = \emptyset$$

$$\mathcal{U}^{(2)} = \mathcal{U}_1 \amalg_{1.2} \mathcal{U}_2:$$

$$\mathcal{U}_i = \left\{\left[\mathrm{MG}\right]_i, \left[\mathrm{op}\right]_i\right\},\ i = 1, 2$$

$$\begin{bmatrix} \left[\mathrm{EVs}\right]_1 & \left[\mathrm{EVk}\right]_2 \\ \left[\mathrm{Vk}\right]_1 & \left[\mathrm{Vs}\right]_2 \end{bmatrix}:$$

$$\begin{pmatrix} \mathrm{Vk}_1 \\ \amalg_{1.2} \\ \mathrm{Vs}_2 \end{pmatrix} \circ_{1.2} \begin{pmatrix} \mathrm{EVs}_1 \\ \amalg_{1.2} \\ \mathrm{EVk}_2 \end{pmatrix} = \begin{pmatrix} \mathrm{Vk}_1 & \circ_1 \mathrm{EVs}_1 \\ & \amalg_{1.2} \\ \mathrm{Vs}_2 & \circ_2 \mathrm{EVk}_2 \end{pmatrix}$$

Because of the super-additivity of polycontextural mediations, the third system (contexture) has to be considered. It represents the contexture where the *result* of the interplay of the two contextures is explicitely stated, i.e. that, on the base of the interchangeability of the two contextures, the morphograms, A and B, are behaviorally equivalent. In other words, the interchangeability game of the two involved contextures gets a reflection, i.e. an own location where this interplay is thematized with the question of sameness of differentnes of the involved kenomically different morphograms. In this sense, the third system (contexture) is reflecting and mediating the two mediated contextures.

Example: $\left(\left[\mathrm{MG}\right], <\mathrm{op}>\right)^{(3)}$:

$$\mathcal{U}_1 \bigcap_{1.2} \mathcal{U}_2 \bigcap_{2.3} \mathcal{U}_3 = \emptyset$$

$$\mathcal{U}^{(3)} = \left(\mathcal{U}_1 \amalg_{1.2} \mathcal{U}_2\right) \amalg_{1.2.3} \mathcal{U}_3 :$$

$$\mathcal{U}_i = \left\{\left[\mathrm{MG}\right]_i, \left[\mathrm{op}\right]_i\right\}, \; i = 1, 2, 3$$

$$\begin{bmatrix} \left[\mathrm{EVs}\right]_1 & \left[\mathrm{EVk}\right]_2 & \left[\mathrm{CEVks}\right]_3 \\ \left[\mathrm{Vk}\right]_1 & \left[\mathrm{Vs}\right]_2 & \left[\mathrm{CVks}\right]_3 \end{bmatrix} :$$

$$\begin{pmatrix} \begin{pmatrix} \left[\mathrm{Vk}\right]_1 \circ_{1.0.0} \left[\mathrm{EVs}\right]_1 \end{pmatrix} \\ \amalg_{1.2.0} \\ \begin{pmatrix} \left[\mathrm{Vs}\right]_2 \circ_{0.2.0} \left[\mathrm{EVs}\right]_2 \end{pmatrix} \\ \amalg_{1.2.3} \\ \begin{pmatrix} \left[\mathrm{CVks}\right]_3 \circ_{0.0.3} \left[\mathrm{CEVks}\right]_3 \end{pmatrix} \end{pmatrix} = \begin{pmatrix} \begin{pmatrix} \left[\mathrm{Vk}\right]_1 \\ \amalg_{1.2.0} \\ \left[\mathrm{Vs}\right]_2 \end{pmatrix} \\ \amalg_{1.2.3} \\ \left[\mathrm{CVks}\right]_3 \end{pmatrix} \circ_{1.2.3} \begin{pmatrix} \begin{pmatrix} \left[\mathrm{EVs}\right]_1 \\ \amalg_{1.2.0} \\ \left[\mathrm{EVk}\right]_2 \end{pmatrix} \\ \amalg_{1.2.3} \\ \left[\mathrm{CEVks}\right]_3 \end{pmatrix}$$

2.2.3. Bisimilarity and metamorphosis

A nice metamophosis happens when the complexity of bisimilar morphograms is set into a metamorphic transformation, which is changing the roles of the bisimilar morphograms depending on the position they are involved into the interplay of metamorphosis.

2.3. Polymorphic sameness

Single-operator strategy with complex monomorphies and complex history-dependence realizing complex anti-dromic self-reference within singular constellations

2.3.1. Characterisation of polymorphy

table:

single	poly	complex	self – ref

$M = \left(R \middle| \overleftarrow{r}\right)$:

single operator: *evol*

polymorphy : $\left[M \middle| r_1 r_2 r_2\right]$

single operator and complex self – reference

$$\text{evol}\left(\left[M \middle| r_1 r_2 r_2\right]\right) = \begin{pmatrix} \left[M \middle| r_1 r_2 r_2 \mathbf{r}_1 \mathbf{r}_1\right] \\ \left[M \middle| r_1 r_2 r_2 \mathbf{r}_2 \mathbf{r}_2\right] \\ \left[M \middle| r_1 r_2 r_2 \mathbf{r}_3 \mathbf{r}_3\right] \\ \text{etc.} \end{pmatrix}$$

2.3.2. Polymorphic interplay

Chiasm

$$\chi\left(M^{1.2.1}, M^{1.2.3}, r_{1.2.2.[1.1]}, r_{1.2.2.[3.3]}\right) =$$

$$\begin{pmatrix} M^{1.2.1} \circ r_{1.2.2.[1.1]} \\ \Diamond \\ M^{1.2.3} \circ r_{1.2.2.[3.3]} \end{pmatrix} = \begin{pmatrix} M^{1.2.1} \\ \Diamond \\ M^{1.2.3} \end{pmatrix} \circ \begin{pmatrix} r_{1.2.2.[1.1]} \\ \Diamond \\ r_{1.2.2.[3.3]} \end{pmatrix}$$

2.4. Kenomic equivalence (monomorphy)

Single-operator strategy with single kenom, i.e. unary monomorphy, and simple history-dependence realizing anti-dromic self-reference within singular constellations.

2.4.1. Characterisation of monomorphy

table:

single	mono	simple	self – ref

$M = \left(R \middle| \overleftarrow{r}\right)$:

Memristor as *a* 2 – layered resistor with resistance *R* and $\overleftarrow{r}$ as retrograde memristance.

Monomorphy

$\left[M \middle| r_1 r_1\right]$

Single operator, simple self – reference

$$\text{evol}\left(\left[M\middle|r_1\,r_1\right]\right)\longrightarrow\begin{pmatrix}\left[M\middle|r_1\,r_1\,\mathbf{r_1}\right]\\ \amalg\\ \left[M\middle|r_1\,r_1\,\mathbf{r_2}\right]\end{pmatrix}$$

$$\text{evol}:\ \left(\left[M\middle|r_1\,r_1\right]\longrightarrow\left[M\middle|r_1\,r_1\right]\right)\Longrightarrow\left[M\middle|r_1\,r_1\,\mathbf{r_1}\right]$$

$$\text{evol}:\ \left(\left[M\middle|r_1\,r_1\right]\longrightarrow\left[M\middle|r_1\,r_1\right]\right)\Longrightarrow\left[M\middle|r_1\,r_1\,\mathbf{r_2}\right]$$

Correspondence of kenomic differences between morphograms, leads to the idea of equal length of morhograms.
The kenomic equivalence rules a kind of iterability which is involved into retro-grade finiteness.

2.4.2. Combinatorics of monomorphy

How to construct monomorphies mathematically?
From a mathematical point of view,monomorphies are partitions of mappings. This is well elaborated by [Schadach 1967].The procedure to build monomorphies out of morphograms, as it is mathematically defined by Dieter Schadach's approach, shall be called *monomorphic decomposition*, short "Dec". Hence, Dec(MG), is the operation to produce monomorphies from morphograms MG.

" Let A and B be non – empty finite sets,
$A=\left\{a_1,\,a_2,\,\ldots,\,a_n\right\}$, $B=\left\{b_1,\,b_2,\,\ldots,\,b_m\right\}$. Let denote B^A the set of all mappings from A to B,
$B^A=\left\{\mu\,\middle|\,\mu:A\longrightarrow B\right\}$, $\text{card}\,B^A=\left(\text{card}\,B\right)^{\text{card}\,A}=m^n$.
The following theorem shows that every family of subsets B^A defines a certain *partition* of B^A.

Theorem 1.
Let $\left\{R_i\,\middle|\,i\in I\right\}$ be *a* family of subsets of B^A where I is *a* finite index set; $R_i\subseteq B^A$ for each $i\in I$.

The family $\{R_i\,|\,i\in I\}$ defines *a* partition of B^A
such that the elements of the partition (the equivalence classes of mappings) are

$$\left[\mu\right]_{I_x}=\bigcap_{i_x\in I_x}R_{i_x}-\bigcup_{i_y\in I-I_y}R_{i_y}$$

where I_x runs through all subsets of I.

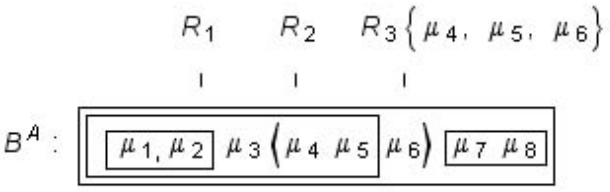
Example.

Let be $B^A = \{\mu_1, \mu_2 \ldots, \mu_8\}$ and the family of subsets $\{R_i \mid i \in I = \{1, 2, 3\}\}$ where

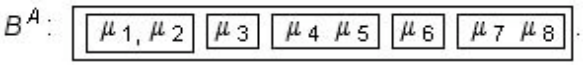
$R_1 = \{\mu_1, \mu_2\}$,

$R_2 = \{\mu_1, \mu_2, \mu_3, \mu_4, \mu_5\}$,

$R_3 = \{\mu_4, \mu_5, \mu_6\}$.



I_x	$[\mu]_{I_x}$
$\emptyset$	$B^A - (R_1 \sim R_2 \sim R_3) = \{\mu_7, \mu_8\}$
$\{1\}$	$R_1 - (R_2 \sim R_3) = \emptyset$
$\{2\}$	$R_2 - (R_1 \sim R_3) = \{\mu_3\}$
$\{3\}$	$R_3 - (R_1 \sim R_2) = \{\mu_6\}$
$\{1, 2\}$	$(R_1 \sim R_2) - R_3) = \{\mu_1, \mu_2\}$
$\{1, 3\}$	$(R_1 \sim R_3) - R_2 = \{\mu_7, \mu_8\}$
$\{2, 3\}$	$(R_2 \sim R_3) - R_1 = \{\mu_4, \mu_5\}$
I	$R_1 \sim R_2 \sim R_3 = \emptyset$



(Dieter J. Schadach, BCL Report No. 4.1, August 1, 1967)
http://www.ballonoffconsulting.com/pdf/1987AppendixII.pdf

2.5. Semiotic equality

Single-operator strategy with single atomic sign, zero history-dependence without self-reference within singular situations.

2.5.1. Characterization of semiotic identity

table: single simple atomic

single operator, simple, atomic

$$M = \left(R \middle| \overleftarrow{r}\right) \Longrightarrow M = \left(R\right)$$

$$\text{op}: \left(R\right) \longrightarrow \left(R\right)$$

$$\text{succ}\left(R\right) = R + R$$

Equivalence of two sign-sequences implies equal *length* and *identity*, i.e. graphemic equivalence, of the elements of the compared sequences.
This opens up the well established realm of *recursivity* for abstract automata, computable languages, Turing machines, Chomsky hierarchies, decidability/nondecidability, and funny problems like the Halting problem.

Reduction of the alphabet of sign sequences to stroke and nil, leads to an *arithmetization* of sign-sequences, and their operators are reduced to concatenation/substitution.