Scientific Research Publishing

Open Journal of Philosophy, 2017, 7, 48-63
http://www.scirp.org/journal/ojpp
ISSN Online: 2163-9442
ISSN Print: 2163-9434

Paradoxes, Self-Referentiality, and Hybrid Systems: A Constructive Approach

Jürgen Klüver

CoBASC Research Group, University of Duisburg Essen, Essen, Germany
Email: juergen.kluever@uni-due.de

How to cite this paper: Klüver, J. (2017). Paradoxes, Self-Referentiality, and Hybrid Systems: A Constructive Approach. *Open Journal of Philosophy, 7,* 48-63.
https://doi.org/10.4236/ojpp.2017.71004

Received: October 25, 2016
Accepted: January 21, 2017
Published: January 24, 2017

Abstract

Since the discovery of the paradoxes of self-referentiality or self-reference respectively logicians and mathematicians tried to avoid self-reference when constructing formal systems. Yet "real" complex systems like the mind are characterized by self-reference and can accordingly only be modeled by formal systems that are also basically self-referential. In this article I show that and how self-referential computer programs, understood as algorithmic formal systems, are not only possible but also since some time quite common in special branches of computer science. Examples for this argument are neural networks and so-called hybrid systems, i.e. combination of different sub systems. The hybrid system SOCAIN, a combination of a cellular automaton, a neural network and a genetic algorithm is an example for the fruitfulness of using self-reference in a systematic way. In particular, such systems consist of mutually dependent sub systems, i.e. form no static hierarchy.

Keywords

Logic, Paradoxes, Self-Referentiality, Hybrid Computer Programs

1. Introduction: Paradoxes and Self-Referentiality

When more than a century ago Russell discovered his famous paradox of the set that consists of all sets, which do not contain themselves as elements; he draw attention to the problem of self-referentiality: A formal system must not be self-referential in order to avoid the according paradoxes. Consequently formal systems like the *Principia Mathematica* of Russell and Whitehead (the theory of types) and the set theoretical axioms system of Zermelo and Fraenkel (ZFC) were constructed by banning self-referentiality. For example, in ZFC a set is of a higher order than its elements and hence a set cannot include itself as an ele-

ment—or other sets of the same order.[1] The same avoiding of self-referentiality is the base for the hierarchy of meta languages, developed in particular by Tarski (1956).

The (in) famous paradox of Curry is an illustrative example for paradoxes that might occur by not paying sufficient attention to the problems of self-reference: "A sentence A states that A implies B, and B is an obviously wrong statement like 1 = 2." If A is true then "A implies 1 = 2 is true" and because of the logical rule of implication it follows that 1 = 2 is also true. Hence A must be wrong. If A is wrong then "A implies 1 = 2" is true because of the logical principle ex falso quodlibet and hence A is true. From a set theoretical sense though this paradox is based on the contradictory definition that A contains itself as a proper subset—A = {A, B}, i.e. A contains elements that A does not contain. In this example obviously the usage of self-reference does not only lead to a paradox but is also the base for a contradictory definition for the paradox.

To be sure, there are many paradoxes, whose source is not the explicit usage of self-reference like, e.g. the classical paradoxes of infinity by Burali-Forti and Cantor.[2] In this article I just deal with the problem of self-reference for mainly two reasons:

On the one hand the dealing with the problem of self-referentiality had not only the mentioned consequences for the construction of formal systems without self-reference but lead also to the discovery of principal limits of certain formal systems. In his famous article about the incompleteness of formal systems like the Principia Mathematica Gödel (1931) explicitly explained that he got the idea for his fundamental proof by getting acquainted with Russell's paradox and the canonical formulation of paradoxes of self-reference, namely the liar paradox. Accordingly Gödel combined in his proof a logical negation of a self-referential statement—"this sentence is not derivable in this system"—with Cantor's diagonal method and obtained his fundamental results. The same proof principle used Turing (1937) in his equally fundamental proof of the insolvability of the halting problem.

Therefore, the discovery of the paradoxes of self-referentiality seemed to prove that they showed a principle limit of formal reasoning. This was (and sometimes still is) the reason why these paradoxes and the derived theorems of Gödel and Turing were mentioned as important aspects of the role of formal logic in philosophy.

Gödel and Turing apparently saw that the combination of self-referentiality and logical negation might lead on the one hand to the paradoxes and on the other hand might be used as proofs of principal limits of formal systems—num-

[1]ZFC means the set of axioms formulated by Zermelo and Fraenkel plus the continuum hypothesis C, i.e. that the cardinal number of the real numbers is equal to ℵ 1 (Aleph 1), the next larger cardinal to that of the integers and rational numbers. C is independent of the other axioms and consistent with them, the same case as that of the parallel axiom with respect to the axioms of Euclidean geometry.

[2]The paradox of Cantor, by the way, has an interesting similarity to the set theoretical reconstruction of Curry's paradox: The set of all sets (Cantor) must contain not only itself by definition as a proper subset but also its power set as an element. Hence its size—its cardinal number—must be larger than that of its power set. Thus the concept of the set of all sets leads to a contradiction.

ber theory or computer programs. Conversely one might assume that self-reference alone, i.e. without logical negation, must not be something that has to be avoided by all means. Yet in any case the impacts of the "limitation theorems" of Gödel, Turing, and the less famous theorem of Church were such that frequently these theorems were thought to define a definite limit of formal reasoning and hence the construction of formal systems at all. For example, the mathematician and physicist Penrose (1989) used these theorems to prove that a real Artificial Intelligence (AI) will never be possible because of the fundamental limits of formal systems. If in particular formal systems cannot deal with self-reference then no formal system can be equivalent to the mind (or the brain respectively).

On the other hand there can be no doubt that there exist complex systems "in reality" that can be characterized as fundamentally self-referential. The most famous example is certainly the mind (or brain respectively) that is able to think about itself, to change its thoughts after evaluating them, and even to construct a picture—a mental image—of itself. Hence scholars like Penrose are certainly right if they argue that formal systems without such capabilities of self-reference can never be a sufficient, i.e. complete model of the mind and hence not an Artificial Intelligence that earns this name. Other examples are social systems that can evaluate themselves and change their performance according to their successes in competition to other social systems. One might even postulate that the analysis of such complex systems, particularly known in the social and cognitive sciences, by using formal models is possible only if such systems can be modeled by self-referential formal systems. Yet the paradoxes of self-referentiality and the limiting results of Gödel and Turing seem to make such an enterprise rather hopeless.

There exist numerous attempts to deal with the paradoxes in different ways, which I cannot even enumerate in this article. Because my research interests have been for a long time the modeling of complex systems with according computer programs the seeming limits of the paradoxes and the limitation theorems, I was and still am fascinated by the possibilities such computer programs offer for transcending these limits. The paradoxes, hence, served as an orientation how the construction of formal systems on the one hand has to avoid the paradoxes but on the other can make use of the possibilities of self-reference. That is why I focus in this article on these possibilities.

For fortunately things are not so bad as the limitation theorems and the paradoxes suggest. The possibilities of using self-reference in different ways have since some time be explored in mathematics and in particular in computer science. In the next sections I shall describe some different forms of self-reference that can be applied without contradictions and have shown to be very fruitful.

2. Recursivity—A Weak Form of Self-Reference

Independently of the paradoxes of self-reference the concept of recursivity has been used for a long time in mathematics. That is why I remind here of the fact that this form of self-referentiality is a common form of mathematical thinking;

it is a historical early example that the usage of self-reference in formal thinking *per se* has nothing to do with paradoxes.[3]

In a literal sense recursivity or recursiveness respectively means a process of "going backwards" (in Latin *recurrere*): A recursive dynamical system makes its progress by taking into regard all states or several of them it has reached so far and by computing the next state from the former states. The perhaps most famous example of a recursive sequence is the Fibonacci sequence, namely

$$x_n = x_{n-1} + x_{n-2} \tag{1}$$

i.e. the value x_n is computed by adding the last value and the before last one. The Italian mathematician Leonardo di Pisa, called Fibonacci, developed this sequence already in the 13th century, when he had the task to compute the reproduction of rabbits, starting with one pair. In other words, for generating the next value the sequence has to go back to the last two values already generated—it has to refer to itself with respect to its former states.

The usage of recursive equations in particular for the mathematical description of growth processes is quite common in mathematics; another example is the also well-known general growth equation by Verhulst

$$x_{n+1} = ax_n \tag{2}$$

which means simply that the new unit x_{n+1} of this recursive sequence is just the product of a growth factor a and the preceding unit x_n. Again the sequence has to go backwards.[4]

From a philosophical point of view another example is certainly much more interesting, namely the recursive dynamics of Boolean networks (BN) or logical networks respectively. Boolean networks (BN), named after the founder of mathematical logic George Boole, are directed but not weighted graphs; the nodes of the graph or the units are connected by logical functions or Boolean functions respectively. In the case of functions with one or two variables the functions are nothing else than the well-known logical operators like AND, OR, or IF-THEN (the logical implication). Usually BN are binary networks, i.e. the units either have the value 1 or 0, i.e. the truth-values of propositional logic. Figure 1 shows a simple BN with three units a, b, and c, and the two connecting functions AND and OR with AND (a, b) = c and OR (b, c) = a.

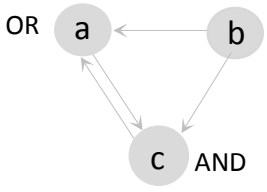

Figure 1. A simple BN with three units and two connecting functions.

[3]It has been frequently noted that the paradoxes of self-reference chiefly were constructed by a combination of self-reference and logical negation (cf. the proofs of Gödel and Turing). Recursivity, hence, is an example how to use self-reference without negation and thus without paradoxes.

[4]In reality of course growth processes cannot go on forever as the general equation implies. Hence so-called dampening factors must be added to equation (2), but this is not of importance in this context.

So far BN are nothing else than a graph theoretical representation of propositional logic, a seemingly uninteresting field of mathematical logic. Things though become more interesting if one understands BN as dynamical systems, namely by generating a particular dynamics. This is done by defining an initial state, consisting of certain values—1 or 0—for the three units. If for example a = 1, b = 0, and c = 1 we obtain the following succession of states:

a	b	c
1	0	1
1	0	0
0	0	0
0	0	0

The state (0, 0, 0) is a so-called point attractor, i.e. a state the system does not change any more. This particular trajectory is generated by using the classical definitions of AND and OR: AND(a, b) = 1 if and only if both a = 1 and b = 1 and AND (a, b) = 0 in all other three cases; OR (a, b) = 0 if and only if both a = 0 and b = 0. A specific BN, hence, generates its recursive dynamics by taking into regard the states just reached and determining from these states the next one.

One can see BN as a dynamical extension of propositional logic, which, as I remarked, seems to be a rather uninteresting field. When rather early it was discovered that the propositional calculus is complete and decidable, in a certain sense Kant seemed right when he postulated in the *Kritik der reinen Vernunft* that logic only delivers analytic statements a priori and therefore is much less interesting than mathematics with its synthetic a priori insights. Yet the dynamical extension of propositional calculus in form of BN demonstrates quite another picture; that is why I showed this example of recursiveness in more detail. Despite their logical simplicity BN are equivalent to potential universal Turing machines, which means that one can generate any complex dynamics with suited BN (cf. e.g. Kauffman, 1993). In addition, BN are still not completely understood in the sense that one knows general laws or regularities that allow to explain the different complex behaviors of BN. So far "only" statistical regularities in form of so-called ordering parameters are known, i.e. parameters that describe the most important mathematical characteristics of BN (cf. e.g. Klüver & Schmidt, 2007).

Other rather sophisticated examples of recursive sequences can be found, for example, in Hofstadter (1979). Yet despite the fruitfulness of the concept of recursiveness in mathematics it would seem a bit tricky to claim that by using recursiveness the problems of self-reference can be solved once and for all. Penrose for example of course knew about recursive sequences and according recursive equations, but he would also certainly argue that this concept is far too simple for the "real" problems of self-reference and in particular those with respect to the limiting theorems of Gödel and Turing. After all, the "going back" of

recursive systems is much less a reference of the system to itself, i.e. to former states, than statements *within a system about the system* as in the proofs of Gödel and Turing.

Therefore, the demonstration of the possibilities of using recursiveness is not enough to show that it is possible to construct self-referential formal systems without the dangers of paradoxes and the principal limits of formal reasoning. To be sure, the results of Gödel and Turing remain valid, but they are perhaps not as important for the construction of formal systems as many scholars believed.

3. Algorithms and a Definition of Self-Referential Computer Programs

In a general sense algorithms as the (mathematical and logical) basis for computer programs always constitute recursive sequences in form of the runs of the programs. An algorithm is of course just a more or less sophisticated computation method. Accordingly algorithms that generate sequences of values must do this by operating in a recursive way—go from state x_n to state x_{n+1} by applying always the same computation rule. An algorithmic pseudo code for the small BN in Figure 1 is for example:

IF a, b, and c are in states x, y, and z, and IF x, y, z ∈ (0, 1), THEN c = AND (a, b), a = OR(b, c), and b = y (i.e. the value of b does not change in all runs). The IF-THEN structure means in this example the next step in the computation of a, b, and c.

One might say that without the permanent usage of recursiveness no computer simulations of dynamical systems would be possible at all. Yet in spite of this general characteristic of algorithmic computer programs no mathematician or computer scientist would call a program like the shown BN code a self-referential program in a strict sense of the word.

A significantly more sophisticated formal system than the "only" recursive Boolean networks are artificial neural networks (NN) that belong to the most important tools in the research of Artificial Intelligence (AI). NN are basically directed and weighted graphs, i.e. their units—the so-called neurons—are connected by directed edges with certain numerical values, the so-called weights. Figure 2 shows one of the simplest types of NN, namely a two-layered feed forward network:

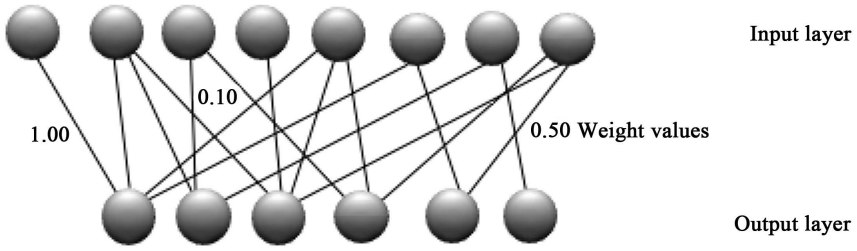

Figure 2. A two layered feed forward net; the top line is the input layer, the bottom one the output layer; the numbers show the weight values.

"Feed forward" means that the "activation flow" in the network has only the direction from the input layer to the output layer. A user gives certain values as input to the input layer and the NN computes according to so-called activation functions the values of the output layer. A standard activation function is the "linear one":

$$a_j = \sum a_i w_{ij} \tag{3}$$

which simply means that the "activation value" a_j of the "receiving" neuron j is computed by the values of the "sending" neurons i, multiplied with the values of the connections to the receiving neurons and added up.

The interesting aspect of these artificial systems is the fact that they are able to "learn", i.e. to change their behavior according to certain rules and "learning goals". To make this possible a complete NN-system consists of (a) the basic directed and weighted graph—the (initial) topology of the system, (b) a "learning rule", and (c) a computational method to measure the learning success of the system.

For example, a frequently used type of learning is the *supervised learning type*. This means that the network gets a specific learning goal in form of a target vector, external to the network proper. The learning process is now steered by the goal in order to minimize the difference between the output layer—the output vector—of the network and the target vector. The learning goal has been reached if this difference is equal to zero or below a certain predefined threshold.

The learning process in the case of supervised learning and feed forward networks consists of the following steps:

a) After the insertion of an input vector compute the output vector by, for example, using function (3).

b) Compute the distance between the output vector and the target vector, e.g. via the Euclidean distance.

c) Change the weight values of the connections by, e.g. using the so-called Delta rule (see below).

d) By applying the initial input vector compute the new output vector as in a).

e) Repeat step b) with the new output vector.

f) Repeat step c).

g) Repeat a)-e) until a certain limit has been reached.

The Delta rule is one of the simplest learning rules, namely

$$\Delta w_{ij} = \eta\left(t_j - a_j\right)o_i = \eta\delta_j o_i \tag{4}$$

Δw_{ij} is the changing value for the weight w_{ij} from the sending neuron i to the receiving neuron j; η is a so-called learning rate, which determines the fastness of the process, t_j is the value of the target component j, a_j is the activation value of the output neuron j, and δ_j is accordingly the size of the error measured as the difference between the output neurons and the according components of the target vector (hence Delta rule). In other words, the variation rate of the weight values is proportional to the size of the error, i.e. the distance between output and target vector.

Other learning types, learning rules, activation functions, and network topologies are known and frequently used, but for the purpose of this article this rather simple example is sufficient.

It is easy to see that the whole process a)-g) also generates a recursive sequence of different states. Yet in contrast to the recursive sequences shown in the preceding section the learning process has additionally at its core a fundamental self-referential aspect: Step b) means an evaluation of the system *by itself*, namely the evaluation of its previous performance in proportion to the learning goal. Step c) is in consequence the correction of specific components of the system, i.e. the weight values as main part of the system's topology, in order to better the system's performance. This correction process is again performed by the system itself, i.e. without any external interference.

Here it is important to remember that the whole network consist of the components (a), (b), and (c) described above. Hence the measuring of the learning success and in consequence the changing of the topology are internal processes, performed by the network. The network, so to speak, searches a way through a "learning space" by permanently applying operations of self-reference.

Because of these characteristics of neural networks it is no wonder that initially these systems were looked at in mainstream computer science as rather strange and esoteric. Yet since several years things have changed significantly: The construction of computer programs that are able to evaluate and change themselves according to specific criteria has become a special branch of computer science, the so-called field of *Metaprogramming* or *Generative Programming* respectively (cf. e.g. Czarnecki & Eisenecker, 2000). One might call these developments in computer science as the quest for the goal making computer programs as independently as possible, i.e. independently from external interferences, by equipping them with the capabilities of special forms of self-reference.[5]

To put it into a nutshell, self-referential computer programs like the described neural networks are formal algorithmic systems that are able a) to observe themselves, i.e. evaluate their performing success, and b) to change themselves according to the desired goals by applying certain types of "meta rules", namely rules that change specific structure components of the system.[6] Experiences with the numerous constructions and practical applications of such systems have since long demonstrated that self-referentiality is not a problem that must by all means be avoided but a very important chance for the development of computer programs that can be applied to real complex problems.

4. An Example: The Hybrid System SOCAIN

Hybrid systems are formal systems consisting of the combination of two or more

[5]When I wrote this article in Autumn 2016 nearly each month one of the big IT firms like Google and Apple announced new breakthroughs in the field of Artificial Intelligence, in particular by using rather complicated neural networks. These announcements are of course frequently just done for marketing purposes, but they show that self-referential programs, so to speak, are now a field of high economic interests and according researches.

[6]A detailed description and analysis of meta rules can be found in (Klüver, 2000).

different sub systems (cf. e.g. Goonatilake & Khebbal, 1995). "Different" means in particular that the sub systems are constructed by applying different techniques. For example, if a BN should be improved by changing its topology for some purpose, i.e. the connections between the neurons, then one could combine the BN with a certain optimization algorithm, e.g. an evolutionary algorithm (EA) that varies the topology of the BN. The whole system BN/EA is then called a hybrid one.

Principally hybrid systems can be constructed in a *horizontal* and in a *vertical* fashion: In a horizontally coupled system the different sub systems operate in a kind of division of labor, i.e. one sub system performs a certain task, hands over its results to a second sub system that uses these results as a starting point for its own tasks and so on. In vertically coupled systems one sub system operates *on* another sub system in order to evaluate the performance of the first and to change it, if necessary. One could call in such cases the second sub system a *meta system* with respect to the first one. The following example is a hybrid system that uses both forms of coupling.

This hybrid self-referential system was constructed by our research group (Stoica, Klüver, & Schmidt, 2004); it demonstrates different possibilities of introducing self-referentiality into formal systems. The task of the program is the modeling of social systems that are capable to construct *models of themselves* and by improving their own models improve themselves. Being a hybrid system it consists of several, in this case three different coupled sub programs that operate "horizontally" with and "vertically" on each other. Because the horizontally coupled sub systems are a cellular automaton (CA) and an "interactive neural network" (IN) the program was called SOCAIN-SOciety consisting of CA and IN.

A cellular automaton is a geometrically simplified version of Boolean networks; the units of the CA are called cells (hence the name). An interactive neural network is a "recurrent" network, i.e. each neuron of the net is principally connected with each other neuron. Hence its topology is much more complicated than the simple feed forward network shown in **Figure 2**.

The CA models the "real" society in the way that the members of the society are represented by the individual cells of the CA; each member belongs to a specific social class. The IN represents the model of the CA in an aggregated way: Each artificial neuron of the IN represents one social class; the number of class members are represented by the so-called activation values of the neurons (cf. the preceding section). Technical details of the whole model, which are not important for this article, can be looked up in (Stoica, Klüver, & Schmidt, 2004).

According to certain environmental conditions this system has the task to improve its system's value. The theoretical assumption is here that the system's achievements depend on the distribution of the social members to the different classes. This improvement is performed by a third sub program, namely a so-called genetic algorithm (GA). A GA belongs to the class of evolutionary algorithms (EA); it performs its optimization tasks by simulating biological evolu-

tion: The respective problem is coded as a vector; then a population of such vectors is generated at random, and subsequently the vectors are varied by the "genetic operators" of mutation and recombination (crossover) until a certain optimization goal has been reached.

The whole system operates the following way: The CA performs several runs; the results are evaluated with respect to the environmental conditions. In the next step the GA betters the CA. Afterwards the CA transforms itself into the IN, i.e. the aggregated image of the "original" society, which is again optimized by the GA. The optimized IN is then back transformed into the CA that performs several runs again. The GA optimizes now the CA that is again transformed into the IN and so on until a sufficient final optimization result is reached. The whole self-referential system hence has the structure shown in Figure 3.

The goal of this program was to demonstrate the possibility of formally modeling self-referential operations like self-observing, self-modeling, and self-improving. By the way, the hybrid system SOCAIN performed much more efficient, i.e. it reached its optimizations goals much faster than a simpler hybrid system that consists "just" of a CA and an improving GA. Constructing models of themselves hence can be a significant advantage for complex systems that have to adapt in difficult environments. In particular, the emergence of consciousness during the evolution of the human species is closely connected with the emergence of the self-modeling ability, i.e. the constructing of "mental images" of one-self.

SOCAIN obviously contains several components of self-reference: The CA sub system and the IN subsystem are horizontally coupled; each of the sub systems is evaluated by the same criteria and they permanently refer to each other by transforming their respective states at a certain time into the other. One might call these transformations mutual mappings of the CA into the IN and *vice versa*. Therefore each sub system is a self-observing one, because only the

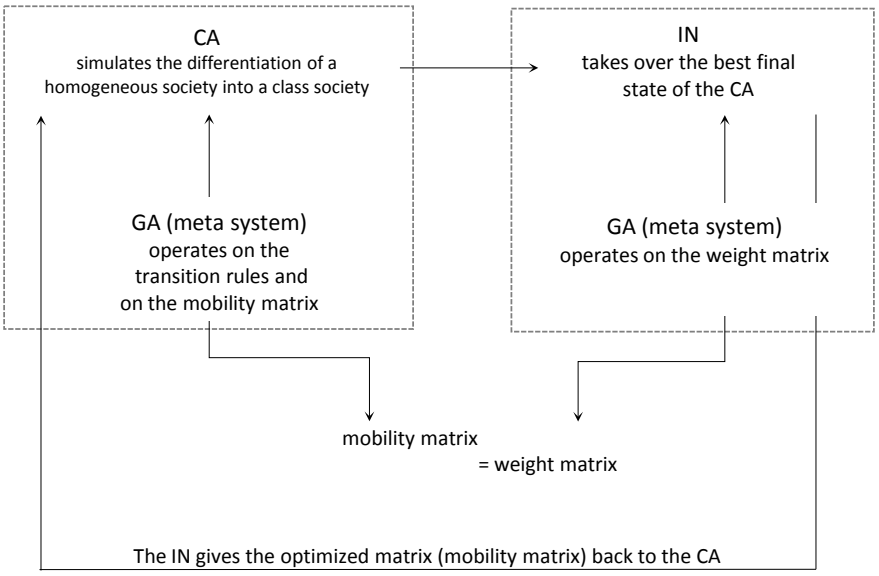

Figure 3. The self-referential system SOCAIN.

best achieved states are each time given to the other sub system. The genetic algorithm additionally optimizes the two sub systems, i.e. the original society and its aggregated image, which means a vertical coupling. Because the evaluation criteria and the according optimization processes are the same for both sub systems the construction generates the same form of self-observing and self-improvement, as that shown in the last section with the example of neural networks, for the whole system.

In nuce, the self-referential processes of SOCAIN consist of a) the construction of a self-image (the IN) of the original social system (the CA), b) the self-evaluating and self-improvements of the vertically coupled sub systems CA/GA and IN/GA, and finally c) the self-observation of *the whole system* SOCAIN until the final result that the optimization process can be stopped. In addition, there is a permanent feedback loop between the basis system, the CA, and the image system: The image system improves the CA by handling its performed states to the basic system; the CA gives the results of its own improvements by the GA back to the IN image system, that subsequently again transfers its results to the CA, and so on until a satisfying end result has been achieved.

The intention for the construction of SOCAIN was chiefly, as I mentioned, the development of a formal system that is able to construct an image of itself; in Kantian terms SOCAIN should serve as the demonstration of the condition of the possibility for such formal systems. The insight that such systems can perform significantly better than comparable systems without this capability is an interesting additional gain of the construction, which should be analyzed further. In any case, even such an advanced form of self-referentiality like the construction of self-images within a whole system obviously is possible too.

5. Self-Referentiality, Meta Rules and Outgödeling

The basic architecture of such self-referential programs is certainly the implementation of specific meta rules in addition to the basic rules of the program. The neural network, e.g., described in section 3, consists of activation functions on the one hand that determine the network's dynamics in form of recursive sequences of system's states. This dynamics depends on the specific functions on the one hand and the (initial) topology on the other, namely the conditions, which neurons are connected with which others and in particular the directions and weights of the connections. Additionally there exists a learning rule, i.e. a rule that operates on the original network and changes several components, in this example the weights of the connections. Such a learning rule might be called a meta rule, because it operates *on* the original system.

In the certainly much more complicated example of SOCAIN the meta rules are defined by the rules of the genetic algorithm, which is itself a formal system and hence operates as a meta system on the two other systems. Here it is important to remind that such meta rules, or in the case of vertically coupled hybrid systems meta systems, are necessary to take appropriate measures from the self-evaluation of the system, namely to change some components of the system in

order of improving the system's behavior. One might call the function of the meta rules the practical consequence of the self-evaluation; the evaluation always needs an additional computational method and could in principle be without consequences. This would be the situation of a human learner who knows that he must change his learning behavior but is too lazy to do it.[7]

To put it into a nutshell, the self-referentiality of such formal systems consists of a) special algorithms for self-observing and in particular self-evaluation, and b) certain meta rules that change the system's structure and accordingly its behavior, depending on the goals the system has to fulfill. Only by inserting meta rules into the original system a self-referential variation of the system is possible.

The usage of the terms *meta* rules and *meta* systems is of course not by chance but orientated to the classical concepts of Hilbert's *Metamathematics* and the hierarchy of meta languages, i.e. concepts that were introduced for avoiding self-reference. Hence one might argue that the self-referentiality of programs like neural networks or SOCAIN could be achieved only by constructing hierarchies of rules and other sub programs that operate always on the next lowest level but avoid self-reference the same way as for example the distinction of different levels of languages does. Consequently according to this critical argument these programs are not more self-referential than axiomatic set theory or a set of meta languages.

To be sure, realizing computer programs like those described is possible only by introducing some different levels of operations, e.g. an original initial system, rules of self-evaluation, and rules to change it. As a *whole system* such a program is certainly self-referential. Yet in addition and in contrast to static hierarchies in such computer programs the different levels are not independent from another, because it is easily possible to construct permanent feedback loops between the different levels. For example, in a vertically coupled hybrid system like the coupling of a CA with a GA in SOCAIN the optimizing operations of the GA—the meta system—might be influenced by the performance of the (optimized) CA too: The better the performance of the CA is in proportion to specific environmental conditions the smaller could be the variation rate by which the GA varies the "object system" of the CA. It is a technique quite frequently used in cases where developing or learning systems should not transcend a certain limit and hence should approach this limit only cautiously. Of course, the opposite is also possible: If a learning or optimization progress is too slow then dependent on the progress of the object system the operations of the meta system can be quickened. Our research group constructed such mutually dependently coupled hybrid systems too.

Therefore, such systems do not simply form a hierarchy of logically different levels but *a dynamical unity* of mutually dependent components with different functions. In the example of SOCAIN I already mentioned the fact that the ho-

[7]The German author Erich Kästneronce characterized such an attitude by remarking "EsgibtnichtsGutes, außer man tut es" (there is no good except one does it). Indeed, what use would be a computer program without practical consequences from self-observing and self-evaluation?

rizontally coupled IN and CA systems permanently influence each other. The same is the case if one makes the GA meta system dependent on the progress of the two object systems. This is much more than the classical self-reference avoiding hierarchies and it justifies the statement that systems like SOCAIN are indeed self-referential. A similar interdependency of different levels, by the way, can and was introduced in neural networks by e.g. varying the learning rate η in the Delta learning rule (4) in dependency of the learning progress of the network. Figure 4 shows the basic architecture of such interdependent sub systems.

The introduction of different levels in such systems, therefore, is not the same avoidance of self-referentiality as in the classical attempts to avoid the paradoxes of self-reference, but just the opposite: It is a methodical way to use the chances of self-referentiality for modeling real complex systems and to construct rather autonomous computer programs (cf. footnote 4).

Finally, what about the limiting theorems of Gödel, Church, Turing? Their fundamental results, of course, remain valid, as I remarked before. Yet their practical importance for the construction of formal systems, in particular computer programs, must be seen in another light. Programs like those described can in principle "outgödel" themselves, to use an apt expression of the physicist Tipler (1994): By self-observing and self-evaluation they can get insights with respect to their own limits and an according necessity for improving their performance, the necessity for enlarging themselves, to remove discovered inconsistencies, and much more. Our research group, for example, developed a tool for cellular automata that informs a user about inconsistent rules and how to deal with them, i.e. the tool gives orientations how to avoid the inconsistencies by inserting other rules. If the user whishes the system improves itself by

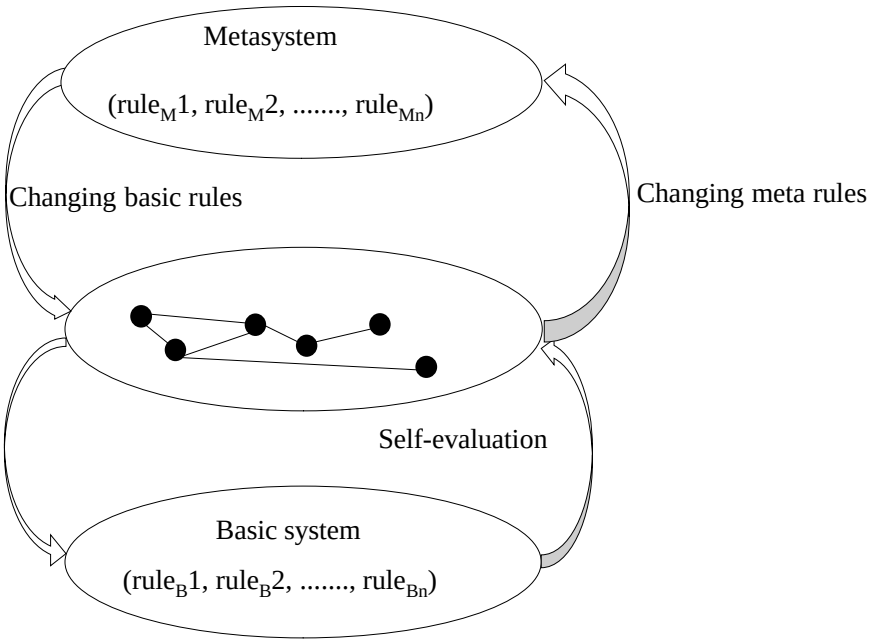

Figure 4. The basic system is varied by the meta system, which permanently varies its operation rules in dependency of the basic system's progress.

automatically inserting corrected rules. Therefore, by such a self-improving the system can correct its own constructor.

At present we develop a hybrid system that is self-developing, orientated to the classical developmental model of Piaget, namely the permanent interplay between accommodation and assimilation. The system consists of a three-layered feed forward net (cf. above section 3) and a self-organized learning network, which we developed in our research group (cf. Klüver & Klüver, 2013), a so-called Self Enforcing Network (SEN). The two networks permanently influence each other: In response to the assimilation results of the SEN the feed forward network varies its own structure, i.e. its basis for accommodation processes. In particular it is able not only to change its weight values but also to *enlarge* itself by inserting additional neurons to the different layers. The results of the varied feed forward net is given to the SEN, which accordingly also enlarges itself and such can perform its assimilation tasks in a more and more differentiated way.

By such a permanent self-reference in form of structure variation and self-enlargement the whole system indeed is able to develop itself without external influences. Such a system is therefore able to outgödel its initial limits not only by perceiving them but in particular by changing its initial structure according to the developmental tasks. To be sure, this system also consists of different rules and meta rules for varying and enlarging its own structure. Self-referentiality, hence, is obviously at the core of autonomous developmental processes and can be modeled in formal systems without principal logical problems.[8]

Of course, according to Turing no self-referential computer program will ever be able to make statements about *all* imaginable programs. But this is certainly not so important for practical purposes as one might assume. Self-referential programs can deal with each *specific* problem, with which they are confronted, by transcending their initial limits—if the problem can be solved at all by the type of programs they belong to. This is the same way human minds deal with specific problems. The principal limits demonstrated by the limiting theorems, therefore, are limits for certain and universal kinds of formal reasoning. But fortunately other ways are possible and obviously fruitful.

6. Conclusion

The methodical core of my arguments in this article is obviously the construction and analyzing of computer programs that operate in a self-referential way without any paradoxes. Because the questions I dealt with are basically philosophical and logical fundamental questions one might call this methodical way "Experimental Philosophy", a concept used in the early Enlightenment to distinguish the new experimental sciences from speculative philosophy and in par-

[8]In computer science such a self-enlargement is frequently called "bootstrapping". This rather strange name comes from the famous story of the Lying Baron Münchhausen, who claimed that he had drawn himself together with his horse out of a swamp. Although the Baron used his hair and not his bootstraps the self-developmental operations of a system like our hybrid network system certainly remind of this story.

ticular theology.[9] With the emergence of computer technology and according logical frameworks for the construction of self-referential programs it is now possible to apply this methodical way even to basic problems of philosophy and thus give the term "Experimental Philosophy" an additional and literal meaning. It is probably not exaggerated to call this a new paradigm of scientific thinking.

In any case, it has been a long way from the discoveries of the paradoxes of self-referentiality to the constructions of self-referential formal systems like the described computer programs. So what, after all, can still be the importance of these paradoxes?

In my opinion their historical role and present importance is best described with the famous ladder metaphor of Wittgenstein:

"My statements are explained this way that who understands me will recognize them in the end as nonsensical when he by them has climbed up over them… (he must so to speak throw away the ladder after he has climbed up on it)" (*TractatusLogico-Philosophicus* 6.54, my translation from the German original).

In this sense we can understand the paradoxes as important steps towards some great achievements in mathematical logic and the foundations of mathematics. Yet now it is time to walk new paths without them.

References

Czarnecki, K., & Eisenecker, U. W. (2000). *Generative Programming.* New York: Addison Wesley.

Gödel, K. (1931). Über formal unentscheidbare Sätze der PrincipiaMathematica und verwandter Systeme I. *Monatsheftefür Mathematik und Physik, 38,* 173-198. https://doi.org/10.1007/BF01700692

Goonatilake, S., & Khebbal, S. (1995). *Intelligent Hybrid Systems.* London: John Wiley Sons.

Hofstadter, D. R. (1979). *Gödel, Escher, Bach: An Eternal Golden Braid.* New York: Basics Books.

Kauffman, S. (1993). *The Origins of Order.* Oxford: Oxford University Press.

Klüver, C., & Klüver, J. (2013). Self-Organized Learning by Self-Enforcing Networks. In I. Rojas, G. Joya, & J. Cabestany (Eds.), *Proceedings of the 12th International Work-Conference on Artificial Neural Networks (IWANN 2012), Part I. Lecture Notes in Computer Science,* Vol. 7902, Springer, Berlin, Heidelberg, 518-529. https://doi.org/10.1007/978-3-642-38679-4_52

Klüver, J., & Schmidt, J. (2007). Recent Results on Ordering Parameters in Boolean Networks. *Complex Systems,* 1729-1745.

Klüver, J. (2000). *The Dynamics and Evolution of Social Systems. New Foundations of a Mathematical Sociology.* Dordrecht, NL: Kluwer Academic Publishers. https://doi.org/10.1007/978-94-015-9570-4

Penrose, R. (1989). *The Emperor's New Mind. Concerning Computers, Minds, and the Laws of Physics.* New York: Oxford University Press.

Stoica, C., Klüver, J., & Schmidt, J. (2004). In the Looking Glass: The Self-Modeling of Social Systems. In Y. Bar-Yam, & A. A. Minai (Eds.), *Unifying Themes in Complex*

[9]Cf. Henry Power, Experimental Philosophy, London 1663.

Systems II. Proceedings of the Second Conference on Complex Systems. Boulder: Westview Press.

Tarski, A. (1956). *Logic, Semantics, Metamathematics.* Oxford: Oxford University Press.

Tipler, F. J. (1994). *The Physics of Immortality*. New York: Double Day.

Turing, A. M. (1937). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society Series 2, 42,* 230-265. https://doi.org/10.1112/plms/s2-42.1.230