

# Organization, Not Behavior

## (An Essay about Natural and Artificial Creatures)

by  
George Kampis

Department of History and Philosophy of Science  
ELTE University, Budapest  
H-1518 Budapest, P.O.Box 32.  
email: gk@hps.elte.hu

### ***Abstract***

*This paper can be summarized by the following three statements: (1) in the modeling of life and cognition, principles are more important than performance; (2) there is a structural similarity between the foundational problems that arise in ALife and AI/cognitive science (therefore, the problems can be treated simultaneously); (3) a non-computational yet rationally definable mechanism can help exploit the difference between what I will call organizational or "principle-oriented" and purely behavioral or "performance-oriented" approaches in both fields.*

## **I. Introduction**

One of the common themes of Artificial Life and Artificial Intelligence (read "strong" Artificial Intelligence) is the concern with dynamic, or "self-organizing" phenomena, as exemplified by learning, evolution, discovery, theory building, and the like. In all these cases, the focus is on a procedural aspect of the phenomenon under study. I will address here the broadly general question, what kind of process models are suitable for such procedural studies.

We begin with some general philosophical remarks, and proceed towards an increasing degree of concreteness, to conclude with a review of some more specific models that reflect the new ideas the present writing is concerned with.

## **II. The Turing Test: What Are We Looking For?**

In "strong" Artificial Life, we want computers to be as alive as plants and animals. In strong Artificial Intelligence, we want computers to be as smart as humans. In both these projects, it may not be easy to succeed, but to decide if we succeeded is not simple either. ALife depends on AI in a variety of ways, so I start discussing AI's basic idea.

Proposed by A.M. Turing in his famous 1950 paper, the test named for him suggests to judge intelligence on the basis of samples of behavior. This suggestion (motivated by a pre-cognitivist attitude dominated by American behaviorism) considers a dialogue via computer keyboard between a human and an unknown entity (which can be a human or a machine). One of the best known definitions of intelligence asserts that a machine should be considered intelligent if it survives the Turing test without being unveiled as a machine.

Let us briefly rush through the logic of this suggestion and its consequences. Some limited version of the Turing Test is easy to pass in actuality. Or at least it is easy to pass the test in some sense - for instance, in the sense that some very simple machine contestants of the annual Loebner Prize competitions (Kampis 1996) were consistently judged as animate by certain human observers. Decisions of this kind are always based on very limited experience, and the human experimental subjects are exposed to the tests under very special circumstances. In particular, in the Loebner competition the dialogue must always be confined to a single, usually very sharply defined topic. This gives machines an unfair advantage - a well-known and often acknowledged fact. At this point the defenders of the Test, such as Harnad, usually propose multiple-pass or temporally unlimited Turing Tests to overcome the problem. The idea is that the restricted performance of the machine will be revealed, if not immediately, then in the limit - unless the machine is indeed equipped with intelligence comparable to that of a human. A standard objection to this suggestion is that maybe a large enough vocabulary or an immense catalog of pre-fabricated answers could fool an observer for life, yet requiring no intelligence beyond that of a simple search-and-compose program.

But here is this fact: most human-to-human conversations follow as simple patterns as machine conversations do, virtually never transcending the almost ritual forms ensured by the given situation. This was revealed by studies of Erving Goffman and others. Such everyday dialogues pertain to what Gordon Pask suggested to call the domain of "strict conversations". In a strict conversation, there is an (implicit or explicit) prior agreement about the applicable messages, expressions and meanings.

Even more interestingly, as Pask's Conversation Theory further suggests, not only machine but also human performance can break down when unexpected sentences are heard or read that transcend the closed conversation domain of a strict conversation. By suddenly changing the subject from food prices to politics or Bordeaux vine we may be able to tell a communication expert from a hundred-line PC program as dialog partners, but the same method may already fail with ordinary people, like you and me, who are just less prepared to handle baffling situations and are better trained in something else. The simplicity or "communicative emptiness" of typical human talk explains why ELIZA and other, hard-wired or menu-based dialogue programs (not to speak of RACTER and other sophisticated tools) can continue to mislead people, as J. Weizenbaum, the constructor of ELIZA, has rightly predicted as early as back in 1967.

Nevertheless, that both ELIZA and I may use the same silly sentences when in trouble does not imply that we use them for the same reason. What do we really want - behavioral similarity or something else? Children can mimic face expressions and bodily gestures long before they could make sense of them, or at least before they could gather information via learning about anything beyond the bare motor scheme. To do something, or to understand something, is very different. It may have been Wittgenstein who first elaborated this idea in detail. He claimed that habits, rather than reason and rules, govern our behavior. We act but do not know why.

Searle re-used a certain form of this idea in his famous Chinese Room example, in order to distinguish between meaningful and meaningless acts, and to show the limits of the behavioral

studies exemplified by the Turing Test. Searle suggests that it is not behavior but understanding that tells humans from programs. "Understanding" is, however, a very problematic word, indeed a murky word, as nobody exactly knows what it means. Besides, several authors have pointed out flaws in Searle's argument, among them, that he implicitly assumes a Mind's I, an internal homunculus who understands what the rest of the mind does. Then he needs yet another homunculus, an so on, *ad infinitum* (for otherwise, who could tell what the first homunculus thinks?).

Many theorists agree that the homunculus must be avoided somehow. But before we have a theory of understanding, there can be no test for that, and anyway, if we can perform no tests other than behavioral, and if understanding is not behavioral, then how could we ever develop a theory of understanding in the first place?

### III. Behavioral and Organizational Modelling

My suggestion is to use other criteria instead of "understanding" in order to assess competence. In what follows, I will outline and show some applications of an alternative, based on the notion of "organization", a biological concept. In order to present this suggestion, we have to first go back in time. There is an old distinction between two different forms of representation that we will find of relevance. The idea comes from the philosophy of science, and to a lesser part, from cybernetics.

We begin with the notion of instrumentalism. An instrumentalist conception of science maintains that the purpose (the *only* purpose) of a scientific theory is to build a functional system on it. (Here we see a distant origin of AI's and AL's "design stance" *à la* Dennett). In other words, the instrumentalist (or instrumentalist-utilitarian) holds that a theory is practicable if it leads to the construction of usable tools.

The critique of this conception emphasized that such a harnessing of Nature requires but very shallow knowledge. Popper and Polanyi pointed out that theory, experiment, and the construction of equipment allows for significant degrees of freedom with respect to the amount and form of knowledge applied<sup>1</sup>. The idea can be illustrated on everyday situations as well. Let us take car driving. At one extreme, imagine a driver whose only knowledge is about the effect of the pedals and the steering wheel on the course of the ride. This kind of knowledge results in some "naïve physics" in the sense of Hayes. Such an ignorant driver knows nothing about the joints and the gears, about the chemistry of internal combustion or the electronic circuitry of fuel injection. Yet he can fulfil his task. There is another extreme driver we can imagine, one who focuses, while driving, on how the internal interplay of the car's parts, the "whole" car leads to the desired motion. (For some readers this could be something in the spirit of the "*Zen of motorcycle maintenance*" or Goethe's *Naturphilosophie* of plant growth, but see below.)

The first approach focuses on what little is absolutely necessary in order to achieve the goal in instrumentalist terms, whereas the second goes for a detailed study of the factors that make the goal achievable in the first place.

---

<sup>1</sup> These were the good old days of cybernetics! W.R. Ashby and D. MacKay put Popper's statement about the existence of different kinds of knowledge into a more tractable form. Their concern was with control systems. As a starting point, they have proven, using different frameworks, that every controller must include a model of the system controlled. As a further result, for our present purposes more interestingly, they obtained the insight that equally successful models can still differ in the amount of information represented, a difference analyzable in scientific terms.

The first approach is behavioral, whereas the other is what I will now call organizational. The first is minimal, the second is pretentious. The first is a mirror, only concerned with the picture to reflect; the other puts flesh on the bones, constructing a whole world behind the mirror (to paraphrase a well-known title by K. Lorenz). In one word: organization means depth of representation.

## **IV. Organization, Other Minds and Other Life Forms**

Armed with this new concept, let us go back to the Turing Test and related matters. Let us talk about "other minds" first.

The problem, whether there can be ultimately any guarantee for the existence of minds other than the speaker's own, is known since Plato. Descartes' answer (which anticipates S. Lem's *Kyberiad*, Putnam's *Brains in a Vat* as well as today's VR experiments) is that all we experience could be a dream, and yet we would never notice. (So, in particular, other human beings could be parts of my dream, therefore having no minds.) Of course, unless someone is willing to hold the philosophical position of solipsism, Descartes' warning does not have to be taken all too seriously. It is serious enough, however, to indicate that we need additional assumptions if we want to deal with other minds.

The simplest candidate at hand is the idea of phylogenetic relationship. Biologically, so goes the argument, we are descendants of the same ancestors. It follows that every human being has the same in-depth internal constitution that determines how our mind functions. One can then infer that if I have a mind, so does, by analogy, everybody else. A nice property of such a "new" criteria is that it can be generalized to other species, to chimpanzees, other primates, even to dolphins - but, alas, this stops after a point. We cannot generalize to Martians and to robots, since we share no common history with them.

What do we have here? It is clear that the phylogenetic similarity argument uses the concept of organization as defined before. But there is a problem - organization, if defined by constitution, is too narrow a concept. In other words, the applied implicit concept of organization is too strongly bound to concrete material realization: it seems to be on the right track but it is just not abstract enough.

## **V. Internal Versus External Programming as an Example for Contrasting Organization with Behavior**

What we would need at this point is an operative definition of the concept of organization, to allow for organizations to be "relocatable" (as an old computer jargon says - a relocatable code is an object format not specific to machine architecture). Whereas I doubt that such a definition could be easily found in general, in this section I will try to offer a particular definition that may benefit our present purposes. Here I will utilize an earlier suggestion published in Kampis (1991).

We will consider two mechanisms, external and internal programming. I will argue that the particular organizations that are of interest for the life scientist and the cognitive researcher apply internal programming, whereas equivalent behaviors without the corresponding deep structures can be generated by many different organizations that utilize external programming. To define terms:

"external programming" simply means programming in the current, present-day style, that is, by using computers "for what they are". In other words, the external programming of a computer amounts to writing a program. An of course a program is a program is a program: it is simply an *a priori* defined symbolic behavior scheme; external programming means forcing symbols on the machine. Now I will develop the argument about internal programming in a number of steps.

### *A Byway In Computing*

It is well known that strong statements hold about the behavioral effectiveness of the "external" programming method. In particular, it is well known that every algorithm corresponds to some external program. In a mathematical form, this is expressed by the Church Thesis, a very robust conjecture. The thesis states, informally, that every mechanistically definable procedure can be translated into some algorithm, which is programmable on a computer. This is not a purely mathematical conjecture, as it can never be proven. The reason is that one of the concepts, that of effectively computable procedure (or, as above, of mechanistically definable procedure) is not mathematical at all. Indeed it is the very Church Thesis that anchors down the intuitive notion of effective computability to a particular formulation, that of a universal machine with a certain set of well-defined properties.

The robustness of the Church Thesis comes from the fact that in the course of time many general models of computation have been tested (besides Turing Machines, there are Markov normal algorithms, Post systems, Wang machines, and others) and all have been found equivalent. It seems we have arrived at a "final word" in computability, and there can be nothing beyond - no "better" machines than the ones we already have.

Perhaps even more interesting than the original Church Thesis is an extension, sometimes called the Church-Turing Hypothesis, or briefly CTH. It was D. Hofstadter's book "Godel, Escher, Bach" that made this concept popular. The idea is that computation should be conceived as something that goes beyond pure mathematics and corresponds to a general theory (or a general framework) of real-world processes. In this spirit, the CTH claims that every physical process is computable. The thesis is of direct interest for the cognitive scientist and Alife modeler.

Hofstadter discusses several versions of CTH, of which, because of its directness and adequacy to our subject, the informal "hackers version" requires most attention here. It goes like this: "If you can define it, we can simulate it". The point is that if, by whatever conventional symbols (be they related to dancing, singing or otherwise), we succeed to communicate a detailed desire for a given behavior, then, assuming CTH is true, a good programmer (i.e. a proverbial computer wiz) can always turn this into into computer code.

It is not difficult now to recognize CTH as a behaviorist statement, and with this remark we are back to our central theme. From the point of view of applicability to real processes, external programming is simply mimicking, or copying of the behavior trajectory of a given process.

### *The Internally Programmed Machine*

The same programming (and computing) performance as above can also be achieved by means of a flexible, self-modifying system that internally programs and reprograms itself. From the behavioral

point of view (and let us emphasize this point repeatedly) such a system can be no stronger than an externally programmed system - in fact, nothing can be stronger than a universal simulator.

At first look the concept of internal programming is somewhat controversial. Complete internal programming is not possible within the framework of computations - instead, the notion of computation must be extended to incorporate it. For a system to program itself it would be necessary to have no program at all in the beginning - but then how could it do anything? Once there is an (externally given) program at the start, all that happens is execution. The difficulty lies in the fact that a truly self-modifying, self-programming algorithm would be at least as clever as baron Munchhausen, who pulled himself out of a swamp by his own hair (in another version, by his bootstraps). Algorithms, just like elevators, need an Archimedean point to work at all, and this point cannot be part of what will be altered, lifted, or programmed.

Yet as I have suggested elsewhere (in particular, in Kampis 1991) this problem can be solved. The organization of various biological systems, like evolving communities or the cell, show examples for processes which produce new defining primitives that transcend the concerns of ordinary computation theory and ordinary behavior simulation. The processes in question can be shown to bring forth new "Archimedean points" and new algorithms literally out of nothing.

So - even if we cannot tell what does it mean to be an organization in general, we may be able to identify classes of organizations as characteristic of biological systems and possibly minds.

## **VI. Applications of Organizational Thinking**

Turning to ALife, here I would like to briefly review a few concrete concepts and models in which organizational representation, in the sense discussed above, is clearly superior to behavioral ones.

### *Shifting reading frames*

Abstracted from a well-studied biological phenomenon, the notion of shifting reading frame is a crucial concept that allows for a context-dependent definition of information. Instead of being bound to a structural representation, here information content is related to a dynamic mechanism that can even redefine or alter it. This is organizational change insofar as not behavior that depends on the information content but the changing internal mode of processing is the subject of representation.

Shifting reading frames can be found in the genetic translation machinery of the cell. The same mRNA strains can translate to proteins in many different ways while using the same genetic code. The phenomenon occurs as a function of how the readout mask of mRNA is placed on the DNA. This determines what counts as a triplet, that is, alters where the code begins.

More generally speaking, the phenomenon of shifting readout occurs whenever one definition frame is changed into another. At this point I usually apply the metaphor of an imaginary Turing machine with a knot tied on its tape. The readout method can switch from a one/nought mode to the knot/knotless mode, or maybe to modes even beyond that. There is no limit.

In a behaviorally equivalent way, a system with a shifting reading frame could be conceived as a system with multiple reading frames, or a system with just a bigger alphabet, but that does not

reflect the organization of the system. The organization is that new frames can be allocated by processes controllable by the own dynamics of the system.

### *Distributed code systems*

Another model that allows for an interactive definition of information content is that of a distributed code system. A code system is something that determines the relation of symbols to other symbols. Classical Turing machines utilize fixed, once-and-forever defined coding schemes to represent their programs and data for the controller unit, which behaves like the CPU of an electronic computer.

By contrast, a distributed code system can be conceived as a network of communicating Turing machines, where each machine uses its output to define (or redefine) the coding scheme of others. Here the biological counterpart is the closed circular pathway of information transfer in the cell. Genes code for the proteins, and proteins together with other gene products "code" for the genes - in the literal yet special sense that these products are factors that determine the relation of genetic symbols to ribosomal expressions. This mutualism allows for an evolutionary "wobbling" of the codes, which parallels the mutation of the genes.

Again, behaviorally (but not from the organizational point of view), a distributed code system is equivalent to some traditional computational system that operates with pre-set codes. But the organizational difference is essential. Distributed code systems can be shown to be structurally nonprogrammable (in the sense used by M. Conrad). This means that the system cannot be programmed from the input (as is the case with ordinary computers) but by means of inserting various components.

Distributed code systems can be directly simulated on a computer and can also be studied analytically. They are perfectly computational in this sense. On the other hand, any individual machine in the network realizes processes that no Turing Machine can compute at all, since the coding comes dynamically, from other machines, in runtime.

### *Component-systems*

For the present purpose, a component-system will be defined by what O.E. Roessler calls the "privileged zero" property. Different from electronic systems, where the number of required variables is independent from how many voltages differ from zero, component-systems have the property that the variables with a zero value are not necessary. A trivial example is a chemical system where molecules of which there is a zero amount are clearly just not there, and if all concentrations equal to zero, the whole system is physically empty. (If the computer is switched off, it still rests on your desk.)

With a slight change of the emphasis, a component-systems is definable as an arbitrary system that produces and destroys its own components.

There can be many behavioral models of component-systems. One model could be based on a list of every possible component and every interaction. This would result in an enormous system, which is difficult to represent and to solve, and one which does not show the economy of the component production process in reality. Real component-systems only produce those variables

which are indeed necessary. Here, an organizational model directly concentrates on the privileged zero property, using a temporally varying number of variables and varying component properties. Since new components usually realize previously nonexistent interactions with the old components, an organizational model should also allow for the dynamic redefinition of what counts as "component property". This is the origin of the idea of internal programming, an idea not tractable within a purely behavioral or setting interested in transformations only.

### *The SPL system*

The SPL system is a technical tool, a large-scale model system to incorporate a few known biological models and to test some new principles. The system was developed by the author together with Mr. Vargyas. The SPL system can realize several distinct modes of functioning with the aid of a simple molecular-computer-like mechanism, embedded in a virtual environment. Among other things, it can simulate Turing Machines, Liberman-type molecular computers, and Tierra-like evolution engines. Moreover, it can allow for the modelling of context-sensitive properties, essential to the realization of component-systems, and systems with shifting reading frames in the sense discussed.

In the SPL system a pattern-based string-processing language serves as a basis for representing complex interactions. One particularly interesting type of interaction permits the components (represented as one program string each) to directly alter other components' program, according to some deterministic or nondeterministic rules, which are defined dynamically by the components themselves, in runtime.

In my view, this is as close as we can get to doing internal programming on a computer. In particular, the system can use random seeds for the generation of new interactions, or, what is equivalent, it can utilize a suitable coupling of independent deterministic programs. This makes a non-algorithmic component definition possible. This process is non-algorithmic in the sense that it is random according to the von Mises axioms and the Kolmogorovian notion of information complexity; and we know that for randomness there can be no algorithms.

It is important to understand, however, that SPL is not simply a stochastic system - for instance, it does not have to be indeterministic or probabilistic at all. In a stochastic system randomness occurs in the form of noise over well-defined events, whereas in SPL it can be used for extending the original event system, which is a completely different issue.

## **VII. A Case Study: Machine Evolution**

Finally, after the abstract examples presented above I would like to discuss a well known case from the ALife paraphernalia. The selected example is the evolution of computer programs, and I will show in a very detailed way, why they presently fail. Furthermore, I will discuss what good the introduction of organization could do to them. In other words, this will be a case where the system cannot be complete even in the behavioral sense, due to a lack of proper organization.

I will talk about the kind of systems that include Tom Ray's Tierra and its relatives or predecessors, among them, Vyssotsky's "Darwin" system, Core War, Venus, Psoup, or, from the commercial palette, El-Fish, SimLife or Evolve. Let us call them, as I already did, evolution engines.

These systems offer variations of the same theme: one gets a finite field of computer memory, inhabited by programs that reproduce and mutate. Besides, the programs can also do other things, such as performing arithmetical operations, jumping to other locations of memory, and so on; the usual assembly language instructions. As a result, we get a population of competing programs that stand under continual selection which favors the faster or trickier ways of reproduction. To test such a system is very instructive: the observer sees a plethora of "organisms" rise and fall, and interact in ways that would be truly difficult to imagine in an armchair.

A critical study reveals, however, a big problem. The evolution process in all these systems is always degradative, a perhaps surprising fact. Long or more complex programs arise only as mutants and go extinct after a short period of time, leaving room for the smaller, less interesting but more aggressive candidates. A not uncommon outcome is that parasites arise (whose reproduction is the most effective of all, since they can be simpler than the full-scale replicator) and they drown their hosts as well as themselves into the mud. The result is that the system dies out.

An immediate reason for this kind of behavior is competition itself: competition alone favors raw speed but not complexity or adaptation, as well known from "wet" biology. The smaller the faster - and that's it. However, in a further analysis, it is easy to find that the ultimate reason for competitive breakdown is lack of sufficient richness. The models are too simple. What I believe is missing here is richness of a particular type, identical to what we have called "organization".

Evolution engines use only one kind of raw environmental constraint, the size of the computer memory, to control evolution. In an ecosystem, on the other hand, always new constraints emerge as new organisms (new species) appear, and the field of competitors becomes structured. Mice do not compete with elephants. In reality, the dynamic structuring and restructuring of the system of constraints occurs due to several factors. One of them is the appearance of trophic chains, where the consumers become competitively subordinated to their specific prey and not to others. Another source of the structure is the dynamic definition of new evolutionary forces. These forces do not necessarily have to do with feeding relationships but can be more complicated, such as giving shelter (as in a tree) or providing nest material for other species. Also, in a real ecosystem (but not in the evolution engines) there is a basic distinction between producers and consumers. The first can increase the amount of available resources, and the second can decrease that. In Tierra and similar systems, however, there are only consumers, and the only resource to consume is physical room, which is not produced or reproduced by any program or any "digital organism".

What these models correspond to is not evolution but the solution of a kinetic optimization problem with predefined evolution forces, just as in population genetics, nothing more. So, when R. Dawkins or J. Maynard Smith speak in favor of these evolutionary computer models, they also speak of evolution as a behavioral trajectory, and not as an organized process.

We can illuminate the point by an analogy, which comes from mathematical chemistry. Mathematical chemistry is structured into different types of problems. Reaction kinetics deals with the velocities of chemical reactions. Once we have a reaction kinetic setting, the problem is to find the fastest reactions, which one can expect to be the most important ones in a system. However, that is exactly one half of the story. Chemical equations cannot be reduced to reaction kinetics much as evolution cannot be reduced to population genetics. Kinetics is just the quantitative side of things chemical. Also there is stoichiometry, which describes the interaction between different chemical qualities. That is the place where the "what" part of the reactions is defined (as opposed to the "how" left to the kinetic part).

Stoichiometry (or, more generally, reaction topology) is a purely organizational concept, of course, as one can realize the same kinetics in various topologies, but not the other way around. In other words, chemistry teaches us that when the organization that defines the basic interactions is given, then, and only then, does the problem of reaction systems reduce to a behavioral problem. To conclude the analogy, in evolution the real question is not what happens in a given system of well-defined selection forces, but what happens to that system of forces. This has to do with the ecosystem level, which offers the "stoichiometry" of evolution and tells what "evolutionary reactions" can take place at all.

## **VIII. Co-Evolution: From Physical Space to "Fruits" and More**

A final question is, then, how to add organization to the evolution models. I would like to briefly sketch a strategy, as a final outcome of this paper. The suggestion will be very simple and certainly far from complete. We will not consider realistic trophic relations, nor other interactions that are even more complicated. Only the basic problem of dynamic constraint definition will be dealt with.

My idea is to build a population of self-reproducing programs that cannot reproduce alone but must cooperate by means of exchanging specific pieces of information. Informally, this could be interpreted as a mutual or cyclic feeding relationship where the exchanged information carriers (to be thought of as "fruits", perhaps) must be found and consumed. They could serve as a prerequisite condition for replication. In such a system, competition could occur not directly for physical room but (if the room is large enough) for the fruits that enter as limiting factors. As part of the same system as the "organisms", fruits could themselves be variable. Their production could be subjected to the same mutation and selection process that otherwise applies to producers and consumers. Co-evolution could get a start. This way, the emergence of niche segregation is expected, even starting with an initially homogeneous population. It will probably pay out for the organisms to avoid competition by abandoning the resources everybody uses (and are therefore depleted), and to use initially scarce resources, whose amount may be increased when a mutation in a consumer develops the ability to provide reinforcement to the producers of these resources. "Fruit trading loops" could be opened, their chance depending only on mutation rates and on the initial availability of primary producers. Not only two-member but multiple-member loops are possible that may spontaneously converge to a supercyclical organization having both divergent and convergent branches. A primitive ecosystem would arise.

This may still be a poor metaphor for evolution in almost every respect, yet this would be a metaphor which already allows for a direct experimentation with organizational aspects, and not just with the behavior of competitors in a predefined system. I expect that the suggested strategy can be realized using SPL's ability for self-programming. The project is currently under way and is hoped to produce first results soon.

## **IX. Conclusions**

The enterprise of current ALife and AI is too much biased towards repetition, behavioral mirroring and mimesis, and too little concerned with models based on organizational ideas. Take the example of self-reproduction.

Even a superficial look at the current cellular automata models of self-reproduction suffices to reveal this, but we could take many other examples. Self-reproduction is a complex phenomenon in reality, not reducible to the doubling of a pattern, such as the ones on the computer screen - what would be so difficult about this? Still, recent models apparently aim at mere pattern repetition. Probably the only exception, and a very remarkable one, is J. von Neumann's original self-reproducing automaton, dated from the fifties. It was based on two very strong realization theorems, the construction fixed point theorem, and the building block fixed point theorem.

The first theorem shows that a universal constructor can also construct a perfect copy of itself, which is a highly nontrivial claim. The second, even more surprising statement asserts that there exist a set of building blocks, from which such a universal constructor can be built, such that it can produce every other automaton built from these very same building blocks. Only a few people have recognized the significance of these twin theorems, which go way beyond the possibility of simple replication. A notable exception is M.A. Arbib who in his 1969 automata theory discusses the issue at length, and of course there are L. Lofgren, R. Rosen and others.

To be sure, even the von Neumann model suffers from obvious drawbacks, as it uses (or perhaps misuses) certain abilities of machines for which Nature has no counterpart. For instance, nobody has ever seen a universal constructor outside a computer. In this respect, the whole issue rests not on similarity, but on difference from reality. Even in the case of the von Neumann system, a further study is required, as I have repeatedly suggested in earlier writings. But the spirit of the work is something from which I believe we have got much to learn. It suggests that one should go back right to the beginning of things, and re-build models with an organizational philosophy.

## **Acknowledgment**

The partial support of the research grant OTKA #25880, Hungary, is gratefully acknowledged.

## **An Annotated Bibliography**

Kampis, G. - Csányi, V. 1987: Replication in Abstract and Natural Systems, *BioSystems* 20, 143-152. A first exposition of the repetition versus realization problem in the framework of replication.

Kampis, G. 1989: Two Approaches for Defining 'Systems', *Int.J.General Systems* 15, 75-80. A discussion of constructivist and nominalist approaches to modelling. The paper connects with several lines to the behavior/organization problem.

Kampis, G. 1991: Process, Information Theory, and the Creation of Systems, in: *Nature and the Evolution of Information Processing* (ed. K. Haefner), Springer, Berlin, pp. 83-103. Introduces and discusses the notion of shifting reading frames and its relation to the construction of meaning.

Kampis, G. 1991: *Self-Modifying Systems: A New Framework for Dynamics, Information, and Complexity*, Pergamon, Oxford-New York, pp 543+xix. Brings a chapter on the Church-Turing Hypothesis, and another one on self-reproducing automata. A detailed discussion of self-modifying systems.

Kampis, G. 1993: *Coevolution in the Computer: The Necessity and Use of Distributed Code Systems*, ECAL '93, Proceedings. The introduction of the notion of distributed code systems, with a first discussion of the "fruits" model of coevolution.

Kampis, G. 1993: *Computing Beyond the Machine Metaphor*, in: *Computing with Biological Metaphors* (ed. R. Paton), Chapman and Hall, London, to be published. An exposition of SPL as well as a general discussion of computability in the light of process philosophy.

Kampis, G. 1996: The Loebner Prize Pages, URL:<http://hps.elte.hu/~gk/Loebner/TT.html>

## References

Conrad, M. 1983: *Adaptability. The Significance of Variability from Molecule to Ecosystem*, Plenum, New York.

Dennett, D.C. 1971: Intentional systems. *Journal of Philosophy* **68**, 87-106.

Goffman, E. 1974: *Frame Analysis: Essays on the Organization of Experience*, Harper, New York.

Harnad, S. 1991: Other bodies, other minds: A machine incarnation of an old philosophical problem. *Minds and Machines* **1**: 43-54.

Hayes, P. J. 1985: The naive physics manifesto, in: (Hobbs, J. R. and Moore, R. C., editors) *Formal Theories of the Commonsense World*, chapter 1, pages 1-36, Ablex, Norwood, New Jersey.

Hofstadter, D. 1979: *Gödel, Escher, Bach*, Basic Books, New York.

Neumann, J. von 1966: *The Theory of Self-Reproducing Automata* (Burks, A.W., editor), Univ. of Illinois Press, Chicago.

Pask, G. (1975). *Conversation, Cognition, and Learning*, Elsevier, New York.

Polányi, M. 1968: Life's Irreducible Structure, *Science* **160**, 1308-1312.

Popper, K.R. 1959: *Logic of Scientific Discovery*, Hutchinson, London.

Rössler, O.E. 1984: Deductive Prebiology, in: *Molecular Evolution and Protobiology* (Matsuno, K., Dose, K., Harada, K., and Rohlfing, D.L., editors), Plenum, New York, pp. 375-385.

Searle, J.R. 1980: Brains, Minds and Computers. *The Behavioral and Brain Sciences* **3**, 417-457

Turing, A.M. 1950: Computing Machinery and Intelligence, *Mind* **54**, 236-245.

Weizenbaum, J. 1965: ELIZA -- A computer program for the study of natural language communication between man and machine. *Communications Association Computing Machinery* **9**, 36-45.