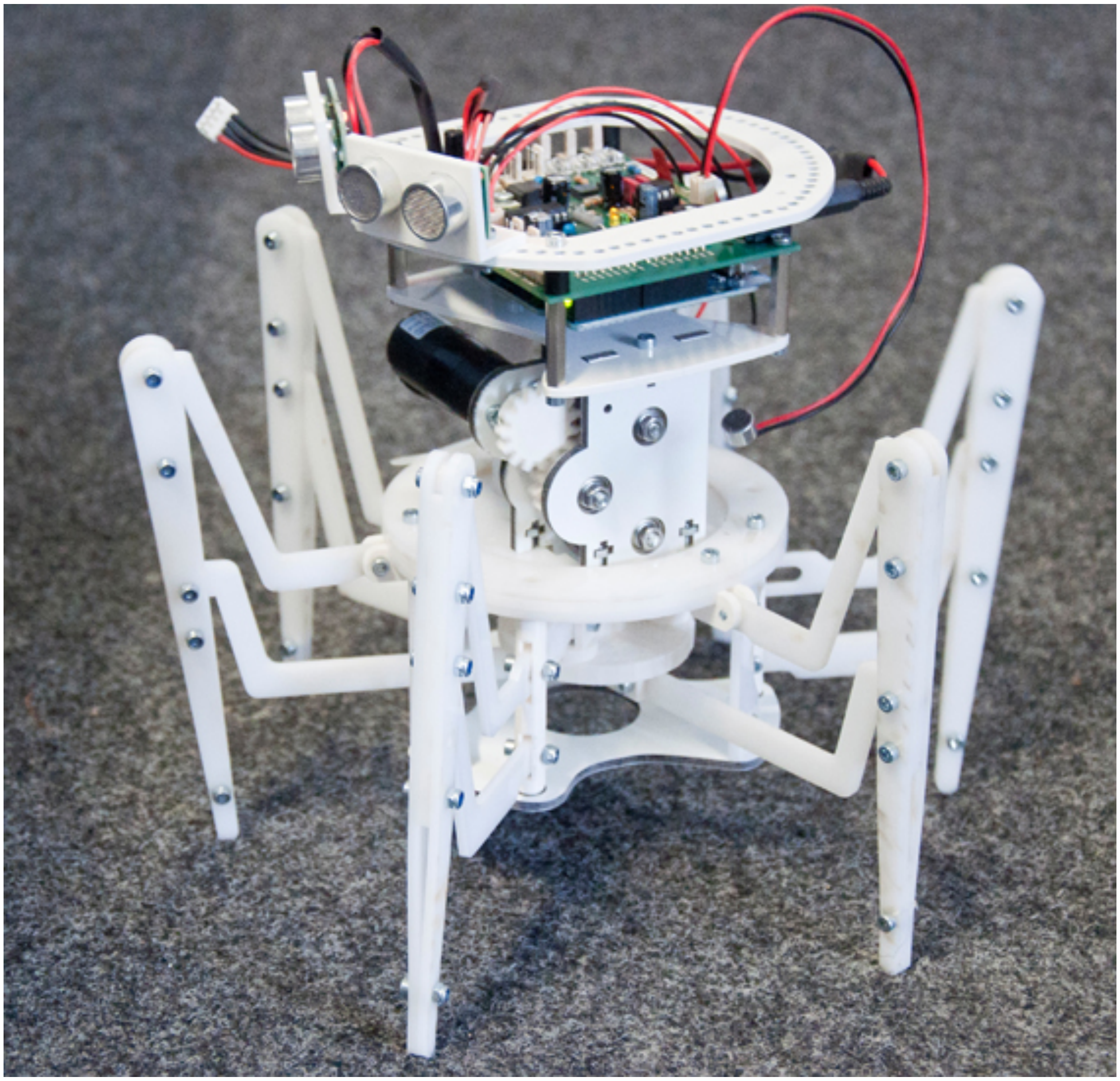
Biological machines and the mechanization of life

© GEORG TROGEMANN
ACADEMY OF MEDIA ARTS COLOGNE
2013



The following text is partly identical to the German version: Synthese von Maschine und Biologie – Organische Maschinen und die Mechanisierung des Lebens. Published in: SYNTHESIS. Zur Konjunktur eines philosophischen Begriffs in Wissenschaft und Technik, Gabriele Gramelsberger, Peter Bexte, Werner Kogge (Herausgeber), Transcript 2013.

Biological machines and the mechanization of life

Georg Trogemann

Abstract
During its long history from antique hand-operated instruments to modern information processing automata the notion of the *machine* has several times received a shift in meaning. Today the concept of the machine has completely lost its attachment to any concrete material and is instead characterized by its functional behavior. *Symbolic machines*, i.e. the mathematical idea to mechanically operate with symbols, became a fundamental skill in many different scientific disciplines. In this paper we take a look on synthetic biology from the computational point of view and especially address the question whether it will once more challenge the notion of the machine. One obvious consequence of future biotechnologies is that we cannot any longer draw a strict line between technique and life. In the past machines did not assemble, maintain and reproduce themselves, they had to be fabricated by man and required human monitoring and directing. Through the technical use of biological processes this hallmark of the living becomes untenable. Self-strategies and especially self-referential functional descriptions like self-assembly, self-reproduction, and self-modification are at the center of the convergence of the natural and the artificial. Conversely the adoption of life-like qualities by technical artifacts will also challenge our image of life and organisms and our understanding of what *aliveness* could mean.

Keywords: machine, algorithm, organism, self-reproduction, self-assembly

1. The notion of the machine

According to Lewis Mumford [1] machines have been an essential part of our technical heritage for the last three thousand years. "Machine" first referred to simple machines of classical antiquity, levers, wedges, wheels, winches, pulleys, screws and inclined planes, and combinations of them. But Mumford points out that although the Chinese, the Arabs, the Greeks, the Egyptians, and the Romans took the first steps towards the machine they did not develop *the machine*. By *the machine* Mumford refers to the entire and mutually dependent technological complex, which includes physical knowledge and technical and cultural skills as well as the entire spectrum of technical artifacts like utensils, apparatus, utilities, tools, and machines. He complains that the role of utility and apparatus has been neglected in most discussions of the machine and the development of our modern environment. Within this broader image of *the machine*, Mumford also cites the definition of what we today call the classical mechanic machine and which was introduced in 1876 by Reuleaux in his book *The Kinematics of Machinery: Outlines of a theory of Machines* (see [1]): "A machine is a combination of resistant bodies so arranged that by their means the mechanical forces of nature can be compelled to do work accompanied by certain determinant motions". In this definition the work conducted by the machine can still depend on human power. Machines that use external sources of power Mumford calls automatic machines. For him the basic distinction between a machine and a tool lies not in the question where the driving force comes from but in the degree of independence in the operation from the skills and directive of humans. Mumford's classification of basic types of artifacts has been taken up later and slightly modified by Carl Mitchum [2]. He especially introduces the difference between machines and automata. Mitchum subdivides machines from the viewpoint of energy into four classes: (1) machines that depend on human or animal power, (2) machines that employ direct mechanical energy from nature, (3) machines that create their own mechanical energy from heat, and (4) machines that use some sort of abstract energy. This differentiation reflects the chronological development of machines and the development of our abstract concept of energy. While only the first type of machines requires human energy input the others use external sources of power. The last two categories have been further subdivided into those machines that generate or transform energy and those that transmit power and perform work. But all machines still require human direction while the concept of automata (or cybernetic machines) requires neither human energy input nor immediate human direction. Automated devices feed parts of their output back into the device itself and use it as a form of control.

The notion of the machine was always directly connected to the technological developments of the related time. It successively had to be extended to keep pace with actual technological skills and practices and had always to unite the old and the new types of machines. Although we still produce and use mechanical devices, the classic mechanical machine concept is outdated and had long ago being extended to include new technical facts. Reuleaux's concept was formulated before the advent of electrical, chemical and nuclear energies. Our current machine concept is compared to former notions more abstract in at least three respects. Firstly, it is abstracted with respect to the driving power of the machines. The concept of energy allows the consideration of the different forms of power producing and power consuming machines under the same abstract notion. Secondly, we realized that machines are not bound to any physical material but can better be characterized by its functional behavior. This equalization of varying material solutions of a machine is only possible through the mediating role of formal sign systems, which are used for the precise notation of its spatial, temporal, and logical behavior. Today only those proficiencies are built into industrially produced machines, which before have been formalized in different multi-layered symbolic descriptions. Formal sign systems are the central tool within the current concept of *the machine* in the sense of Mumford. Thirdly, the notion of the machine is abstracted with respect to the concept of information processing. The work machines perform can be physical work as well as intellectual work. The following definition from the cyberneticist Georg Klaus [3] demonstrates the transformation that the notion of the machine went through because of the advent of information processing machines:

> "Machine: any instrument, any device, any system that processes a certain input (respectively certain types of inputs) to a certain output (respectively certain types of output). Under this viewpoint any machine is identical with the material model of a certain transformation. This generalization of the classical notion of the machine is necessary if machines for processing materials, machines for generating certain forms of energy from other forms of energy and for transferring energy, as well as machines for generating, transferring, and storing of information want to be brought together under one generic term."[1]

This definition of the machine as an input-output-device is almost indistinguishable from the most general definition of a *system* provided by system theory. The only difference seems to be that the machine still needs some sort of material realization, on the other hand the actual material is not important as long as it fulfills its designated function.

Although already by the 16th century, Western technology had reached a level where tools and machines began to show lifelike characteristics, there was always a clear divide between the natural and the technical. While living systems were self-replicating and self-maintaining, machines did not assemble and reproduce themselves, they had to be fabricated by man and required human monitoring and directing. If we want to consider the interrelations between machines and biology we have to analyze organisms, which are the primary units of life. The two notions *organism* and *life* are at the center of theoretical foundations of biology. Essential processes of life, like reproduction, metabolism, behavior, self-maintenance and regulation are qualities of organisms. Interestingly

[1] Translation form German by the author.

Laubichler [4] points to the fact that a newly framed and formalized concept of organism will play a central role in the biology of the 21st century. The concept of function in biology is racked up with technical functions, e.g. as of parts of machines or of artifacts, which are linked to intentionality and teleology. For Laubichler the intended formal notion of organism will be the starting point for mathematical models and theoretical reflections especially in evolutionary-, developmental-, immuno- and neurobiology. He also refers to the problem that the newly propagated notion of organism has the problem to distinguish itself from the notion of a *system* and also puts the question, whether there is a difference at all, and if so, does it matter? So biology, as well as technology, seems to be approaching a more and more abstract and formal notion of the machine resembling that of the *system*. This points to another meta-theory, since system theory has strong connections to Wiener's cybernetics. Norbert Wiener's approach explicitly tried to explain communication and control in the living and the machine on the basis of the same abstract mechanisms. Cybernetics from the beginning also dealt with similarities between living systems and machines. With the concentration on questions of autonomy, cognition, self-organization, and especially the role of the observer in modeling a system in the 1970s, the term *second-order cybernetics* was established. Meanwhile the core ideas of cybernetics have been assimilated by other disciplines. Along these lines *artificial life* can be seen as an updated version of questions asked in 2nd order cybernetics. Christopher Langton, one of the founders of the field, defined artificial life as follows [5]:

> "Artificial Life is the study of man-made systems that exhibit behaviors characteristic of natural living systems. It complements the traditional biological sciences concerned with the *analysis* of living organisms by attempting to synthesize life-like behaviors within computers and other artificial media. By extending the empirical foundation upon which biology is based *beyond* the carbon-chain life that has evolved on Earth, Artificial Life can contribute to the theoretical biology by locating *life-as-we-know-it* within the larger picture of *life-as-it-could-be*".

This approach views life as a property of the organization of matter, rather than a property of the matter itself. The way we describe organizational principles today are formal sign systems.

Recapitulating the first section there are three points to emphasize: 1. The key characteristic of machines compared to other technical artifacts, like for example tools, is their independence from human directive. All machines seek for non-reflective repeatability. A formerly achieved insight into a process or chain of thoughts is functionalized and thereby accessible to technical realization. Within this process formal models and symbolic descriptions are very handy ways of functionalizing insight. 2. The material basis of the machine is secondary compared to its logical from. The advancing abstraction of control mechanisms for machines has let to the insight that the essential qualities of any control structure can be captured within systems of abstract rules. Thus the modern equivalent of the machine are formalized procedures, i.e. *algorithms*. 3. In biology there is also a clear tendency to describe organisms and live on the basis of formal models. The concept of function in biology is amalgamated with technical functions like that of machines. In the field of *artificial life* this leads to the assumption that life can, like it was shown for machines, be separated from its material basis and aliveness consequently is then a property of the logical form.

The following sections of the paper will concentrate on some of the key principles for biological machines and the formal models that allow for the algorithmic description of life-like qualities.

2. Symbolic machines and universal computers

During the first half of the 20th century the groundwork for the abstraction of the logical form of the machine was laid. Arithmetic and mathematics in general were finally recognized as a mechanical process of dealing with sign systems. Computability theory at this time asked questions like: what can be effectively automated? Or the other way around: can we formulate problems, which are unsolvable by any machine in any conceivable programming language? To give answers to these questions formal descriptions have been developed of what we precisely mean by a *problem* or by an *effective procedure* that solves the problem. These formal operational models of effective procedures are called *algorithms*. Our current notion of algorithms mainly goes back to mathematical works of several people during the 1930s, e.g. Alan Turing, Alonzo Church, Kurt Gödel, and Emil Post. Meanwhile a vast number of formal models are available that all have turned out to be equivalent in their fundamental computing capabilities. Despite considerable differences in formalism, they all share some common characteristics: (1) effective procedures are composed of a series of elementary steps, which are executed one by one, (2) only a certain number of different types of elementary operations is needed, (3) each operation as well as the selection of the next are absolutely deterministic. From the theoretical point of view nothing else is necessary to construct the whole computational universe. For efficiency reasons a series of additional concepts have been developed to support the description and realization of reliable and reusable complex computational systems. Most important is the encapsulation of series of elementary operations and its operands to new elementary units and the nesting of these units in hierarchies. From this abstract viewpoint, algorithms are a very general concept, applicable to any operation cycle in craft, industry, or organizations, except that in the strict sense of an algorithm we demand a formal description in order to avoid ambiguities.

A model that is easy to understand and structurally close to real-world computers is the *random access machine (RAM)*, which is an abstract model from the class of so called *register machines*. The original definition of *random access machines* goes back to Sheperdson and Sturgis in 1963 [6]. The following set of operations is just one possible version, in literature many alternative sets of elementary operations have been described. The basic element is an infinite number of *cells*, where each cell is a memory unit that consists of a *location* (specified by an index) and *content* (a natural number). The following set of elementary operations might be defined over the cells:

Operations on cells

```
1)    A_i = 0;              // set value of cell A_i to zero
2)    A_i = A_i + 1;        // increase value of cell A_i
3)    A_i = A_i - 1;        // decrease value of cell A_i
4)    while (A_i != 0) {    // loop condition
      ...
      ... do some operations of type 1-4
      ...
      }
```

The hardware of a RAM consists of a sufficient amount of cells and the implementation of the elementary operations. A RAM-program consists of a specific linear sequence of elementary operations. The *while*-operator is the central controlling construct that allows for a nonlinear execution of the program. Depending on the intermediate content of the cells while the calculation is performed, operations of the program are executed or skipped. A *random access stored program machine (RASP)* is a RAM that holds the data as well as the program in the cells. In this general case the machine is equivalent to the Universal Turing machine and very close to the description of the von Neumann architecture, our practical model of computers. To solve a certain problem by writing a RAM-program requires the deliberate decision about any single step the machine will eventually perform. Despite the great advancements in programming languages, e.g. object-oriented approaches or genetic programming, high-level descriptions are still (e.g. by the compiler) broken down to a sequence of elementary operations. On the machine level nothing will happen that is not exactly and repeatably determined in the program code. Malfunction on the programming level always means errors in reasoning or the human translation of reasoning into the algorithm, not in hardware.

The most important feature of computational models is the separation of the machine in *hardware* and *software*. The data as well as the operations are represented as symbols. The hardware of the machine is endowed with specific properties that enable it to read and perform the operations and thereby manipulate the stored data. The programmer is responsible for selecting the appropriate elementary operations to perform a certain task. This means, that there are three players in the game of computation: human thinking, formal systems, and material (hardware). In our classical model the formalisms (mathematics, text, programs) and human thinking (mathematical reasoning, problem analysis, interpretation and handling of input/output relations) are the flexible constituents while the hardware is the fixed and invariant part of computation. The strength of the universal machine, having a rigid hardware and the necessary flexibility for solving different tasks on the same machine to the software, was already described by Alan Turing, the inventor of this concept [7]:

> “The importance of the universal machine is clear. We do not need to have an infinity of different machines doing different jobs. A single one will suffice. The engineering problem of producing various machines for various jobs is replaced by the office work of ‘programming’ the universal machine to do these jobs.”

A *Universal Machine* is a machine, which is able to read the description of any other machine and to execute this description. So the *Universal Machine* can simulate any other machine. All modern computers are *Universal Machines*. Their operating system allows them to load or edit programs and execute them. Since the program as well as the data are both coded as symbols, the *Universal Machine* is not only allowed to read the program and accordingly to the operations of the program manipulate the data but also can in principle rewrite the program itself.

Concepts like the Turing Machine or the RAM have shown that the material requirements to gain universality are amazingly low. The hardware needs only to be capable of a few basic instructions. How and on what material basis the hardware is realized is completely irrelevant in terms of the functionality of the machine. The series of instructions and thereby the function of the machine is controlled by the software. What is important for our considerations is the fact that any universal model of computation needs in its center a self-referential control element. In the case of the RAM it is the *while*-operator that allows for an indirect form of self-reference. Depending on the content of cell A_i within the while-condition, the following block of operations between the curly brackets is executed or not. But the content of cell A_i may well be a result of primer operations of the program. Thereby the whole calculation process is controlling itself, depending on certain intermediate results. Programming a machine does not mean to decide the exact series of operations before the program is run, it rather means that for any state the process will reach during execution, the structure of the program ensures that the next step is unambiguously determined. The actual series of operations depends on the current inner states of the machine. This means that self-reference has been important for the construction of universal computing models from the very beginning. Similar forms of self-reference are also at the heart of recursion theory and of feedback strategies in cybernetics.

3. Biology and computation

The rigidity of pre-produced and fixed hardware, which was assumed by the RAM model or the Turing Machine is given up in the case of biological machines. The execution of a calculation is becoming a material process of self-assembling. After half a century of experience with the stepwise deterministic models of computation, different practical limitations of the approach have been recognized. Despite their theoretical universality, digital electronic circuits are inflexible by design. Once implemented they can not evolve or adapt to changing conditions and tasks, which often leads to time, resource, and energy inefficient solutions. Computer architectures based on the classical model of sequential processing are also not best suited for handling problems with nonlinear complexity in time and memory. Synthetic biology, with its fluid scalable hardware, intends to offer new opportunities to overcome the limitations of the classic models of computation (see for example [8]). Also the traditional algorithmic methodology was already strongly influenced by principles that have been learned by watching nature, evolutionary computing, artificial neural networks, swarming behavior, and L-Systems are just a few examples. Thus biology and technology not only met in certain practices, they always shared the need for common perspectives and formal operational models. Although there have always been productive connections between computer science and the science of biology, the targeted dynamization of hardware in the field of biomolecular computing will probably change our practical understanding of computation and finally our theoretical notion of algorithms. The theoretical Turing-border of computation might also hold for biological machines, this question is still highly controversial. Although research in biological computation is still in a state of laboratory experimentation, the relevance of biological processes and the qualities of *biological materials* for computing has always been recognized and formulated from the very beginning of automated computing. In this paper the argument is that not the computing capabilities of biological materials are at the center of biological computing but the theoretical models of self-reference.

To get a general idea of some principles of biological machines we look at some concrete examples, without going into detail: (1) bacterial computer that solve the Hamiltonian Path Problem, (2) slime moulds that find the shortest path in a maze, (3) biologically inspired robotics, and (4) the multitudinous field of cyborg technologies, which also needs to be discussed under the aspect of the biological machine.

(1) The mathematical Hamiltonian Path Problem (HPP) asks whether there is a sequence of vertices in a directed graph from a starting nod to an ending node, where each node is only allowed being visited exactly once. Scientists have succeeded to program bacteria with a genetic circuit that enables them to evaluate all possible paths in a directed graph in order to find a Hamiltonian path [9]. The design of the bacteria needs several steps of abstraction of a DNA sequence into the edges and nodes of a Hamiltonian path. In this special mapping, DNA segments are treated as edges of a directed graph. Nodes, except the terminal one, are treated as genes split into two halves. The first half of the gene is found on any DNA edge that terminates at the node, the second half of the gene is found on any DNA edge that originates at the node. Each node of the graph is represented by a gene that encodes an observable phenotype, for example, fluorescence. Bacterial colonies that contain an HPP solution will express a unique combination of phenotypes that can be detected directly or found by selection (for details see [9]). The number of permutation of edges in a graph grows very fast with the number of edges. For a graph with *n* edges there is a total of $n! \bullet 2^n$ possible configuration of edges. The growing computational complexity is encountered by an exponentially growing number of *processing elements*. The homogeneous population of bacteria constitutes the computer. A growing bacterial colony will produce a vast number of different edge configurations. But we never can be sure that a HPP solution will be part of the growing colony. The authors [9] show that for a 99.9% security of finding an HPP solution in a graph with 14 edges, at least one billion independent, identically distributed bacteria are needed, which can in principle grow overnight in a single culture. Thus, the creation of mutating bacteria is used to solve a mathematical

problem.

(2) A group of plasmodial slime moulds has been extensively interpreted in terms of computation. It has been shown that this primitive organism has the ability to solve the shortest-path problem between two points in a maze. If food is placed at different points of a maze the slime mould will adapt its tubular channels between the points while foraging the food sources. Recently is has been shown for the Tokyo rail system that the slime mold *Physarum polycephalum* forms networks with comparable efficiency, fault tolerance, and cost to those of real-world infrastructure networks [10]. The adaptive process of the growing slime mould has also been mathematically described by a feedback mechanism.

(3) Machines are typically bad in adapting to partial failure or unexpected damage. Normally the performance of the machine collapses and the system sustains complete failure. Animals on the other hand develop qualitatively new compensatory behaviors when confronted with the same situation. Therefore robotics tries to develop machines that also can recover from disorders. One crucial element in this form of adaption is to facilitate continuous self-modeling. In [20] a four-legged machine uses actuation-sensation relationships to indirectly infer its own structure. Subsequently it can use this self-model to generate forward locomotion. When parts of the machine are removed, the self-model is automatically aligned and alternative gaits are generated.

(4) Since the construction of very small flying machines that perform really well in natural environments is an extremely difficult task, scientists came up with the idea to use the amazing flight skills of insects and try to remote control their movement in space. Results from neurophysiology and dynamics of insect flight together with the ongoing miniaturization of electronic circuits and the availability of low power radio systems has led to development of a new class of implantable interfaces capable of controlling insects in free flight for extended periods. Recently several groups have started to explore the advantages of interfaces implanted in insects during pupation [21]: „Beyond the issue of control, insects which undergo complete metamorphosis may present a unique system with which to study synthetic-organic interfaces. [...] Given the extensive re-working of the insect physiology during pupation, it is tempting to hypothesize that interfaces inserted during this period could somehow co-opt the developmental processes for an engineering advantage".

NP-hard problems like the described Hamiltonian Path Problem are a convenient application area for biological computing because they demonstrate how computational processes can be speed up by the use of biological growth processes which are naturally fast. From the viewpoint of programming a classic computer like the above-mentioned RAM or a Turing Machine, the task could be summarized as follows: "How to design a set of elementary operations so that a certain type of end-states will be reached." In molecular programming, the same task is reformulated and described quite differently (e.g. [11]): "How to design a set of initial molecules so that a certain type of molecular complexes will be formed." This means a tremendous change in constructing and handling computational machines. The programmer of biological machines no longer concentrates on finding the correct sequence of elementary operations, but rather on the proper preparation of a self-controlled biological process. The involved biological mechanisms are non-terminating, massively parallel, stochastic, self-referential, adaptive, and self-modifying. In the past there have been many attempts to extend the classical von Neumann Architecture to achieve some of these qualities. But in terms of efficiency the results are still very poor. On the other hand, massive parallelism, adaption and self-modification are inherent qualities of biological systems. But these qualities are in conflict with the classic algorithmic approach of stepwise determinism, where the place and duration of any single operation, and thereby the whole machine activity, is under total control.

4. The formalization of life

The German engineering scientist, Franz Reuleaux developed a symbolic notation for the classification of machines already in the 19th century [12]:

> "In contrast to the physical sciences of mechanics and electromagnetism, where natural laws were codified with mathematical equations, in chemistry and biology attempts were made to classify the objects of these sciences with tables and abstract notation."

One of Franz Reuleaux's unique contributions to kinematics was the creation of a symbolic language with which to classify a machine. The syntax for kinematic devices, which he proposed as a tool to address the problem of synthesis is a language for machine invention. In his quest for an alphabet of machine devices, Reuleaux built the world's largest collection of machine components, a dictionary of sorts of over 800 models. Using his symbolic system, along with his models, Reuleaux sought to deconstruct every machine that had been or would be invented in the future, a "Genome project for the Machine Age." But a general system to classify machines is not sufficient if we want the machines to realize themselves. As explained at the beginning Reuleaux machine is a combination of resistant bodies so arranged that by their means the mechanical forces of nature can be compelled to do work accompanied by certain determinant motions. Defining reproduction on a mechanical basis means that we have to describe the structure of the machine as a fabric of modifiable parts and that this structure is able by itself to repeatedly reconstruct the relations of the parts to one another. One possible strategy is to introduce as a sort of mediator a symbolic description of the machine and a mechanical system that is able to read and execute this plan in a self-controlled manner.

In mathematical terms the simplest form of self-reference is a variable y that is mapped by a function f onto itself.

$y = f(y)$ or in the iterated discrete version $y_{t+1} = f(y_t)$

Thereby the variable y and the function f can stand for different types of circularities. The variable can for example model a voltage in a closed electric circuit or an image in a video feedback, where a camera points to it own image shown on a monitor. The output (data) of some transformation is fed back into system as next input. These types of self-reference are well examined in chaos theory and self-organization. The transformation f is in this case fixed, only the data is subject of transformation.

Self-assembling systems mark a different field of self-referential problems. Here the structure (hardware) of the system itself is subject of alteration. The aim is to establish a process that starts with some simple elements and ends with a functioning artifact that implements certain qualities. The basic physical characteristics of such self-assembling artifacts are described by Pelesko. According to him the four key components of self-assembly in nature are [13]:

Structured particles – These are the basic assembling elements. Their structure restricts the complexity of the resulting system. Often the internal structure of the particles can be modified be external stimuli and thereby offers a mean of controlling the process.
Binding forces – These are the typically reversible forces that hold the assembling elements together. Changing the binding forces offers a second variable to control the process of self-assembling.
Environment – That is the total conditions where structured particles live in. Altering the environment is a third means of controlling the dynamics of the system.
Driving forces – The driving force is usually thought of as noise. Its presence makes sure that the particles can interact stochastically. This stochastic interaction is the central principle that compels the system to move through different configurations on its way to a final state.

This general characterization of self-assembling processes holds for microstructures like biological cells and bacteria as well as for macrostructures like magnetic units arranged on a table. We will typically find these components in many biological based computing systems, e.g. the above-described solution for the Hamilton -Path -Problem. In the earlier sections of the article it was stated that formal principles and self-reference are at the center of organisms and living creatures. Therefore it must be possible to substitute Pelesko's physically oriented self-assembly by a more general formal description. The problem of self-assembly is strongly connected to problems we know from bootstrapping complex systems. The basic figure of bootstrapping is, that while the system is booting more and more complex functions are becoming available. The process starts with just the basic capability that is necessary to build the next a little bit more sophisticated function. This function then is used to build other functions and so on.

Self-replication or self-reproduction is another form of self-reference. In his famous book *Gödel, Escher, Bach – An Eternal Golden Braid*, Douglas R. Hofstadter [16] emphasized the parallels between mechanisms, which allow systems to reproduce themselves and mechanisms, which create self-reference. Hofstadter extensively examines *quines*, named after the American logician Willard van Orman Quine. Quines are self-replicating formal sign systems, especially computer programs, which print its own listings. The existence of quines in any Turing complete programming language is ensured by the fixed-point theorem, which is itself an instance of Cantor's famous diagonal argument. When a quine program is run on a computer, it must precisely print its own instructions without accessing the source file. In most programming languages it is also not possible to manipulate its own textual representation. Therefore to write a quine we have to use the two usual elements of programming, which we call *data* and *instructions*. The idea behind the realization of quines can best summarized by the syntactic construct "*write 'write'* ". In terms of programming language the word *write* plays the role of instructions and the word '*write'* (in quotation marks) plays the role of data. The data of a quine represents the code and therefore can be directly derived from the code, mainly by putting quotation marks around it. During the execution of the program the data is used twice. First the code uses the data to print the code. Then the code uses the data to print the data. This second string can be obtained from the first by a simple algorithmic transformation.

The following program is an executable quine written in the programming language *processing*. When it is run it exactly prints the lines it consists of.

```
void setup() { char a=34;
String h, g;
h ="void setup() { char a=34; String h, g; h =";
g ="print(h+a+h+a+';'+'g'+' '+'='+a+g+a+';'+g);}";
print(h+a+h+a+';'+'g'+' '+'='+a+g+a+';'+g);}
```

But computing processes that involve growth and reproduction of elements (bacteria in the example above) usually cannot entirely satisfactory being explained on the basis of self-assemblage. From the standpoint of organizational complexity self-assembly seems to be a weaker challenge then self-reproduction. Models of self-replication or of self-assembly as described above do not necessarily need an explicit construction plan, as the material oriented description of Pelesko shows. It can be a result of material dynamics embedded in a suitable environment. In living beings, on the other hand, the DNA is carrying the genetic information of the organism. Today we interpret the DNA as part of a construction plan. Using this sort of description for machines means that the machine has to contain within itself an explicit description of its own construction. Turing's notion of the universal computer gave John von Neumann the idea that there might be an equivalent to universal computation on the construction side. He asked for the logic principles of a universal constructor, i.e. a machine that can build any other machine. Von Neumanns theory of self-reproducing automata can be seen as the precursor of a formal theory of growth and reproduction. By a constructor *A* we mean a machine, which, when furnished with a suitable description *I(N)*, will construct a copy on *N*. But we have to do more to achieve a universal constructor. *A* is not self-reproducing, since *A* attended with a description A(*I(A))* will surely produce a copy of *A*, but not a copy of the description *I(A).* So the reproduction cycle is not self-contained and will already stop after the first construction step. The newly constructed machine will not itself start to produce descendants. To correct this defect von Neumann suggested the following complex of machines *A*, *B*, and *C* together with their descriptions *I(A)*, *I(B)*, and *I(C)* [see for example 14]:

Construction machine A
Function: feed *A* with *I(X)* and you will get *X*
Copy machine B
Function: feed *B* with a description $I_1(X)$ and you get a copy $I_2(X)$ of the description
Control machine C
Function: activates *A(I(X))*, activates *B(I(X))*, puts *I(X)* into newly build *X* and releases it to the world
The *Self-reproducing machine E* finally consists of the combination of these three machines
E = D + I(D) where *D = A + B + C* and *I(D) = I(A + B + C*)

Von Neumann's automaton was the first completely mechanical description of a self-reproducing artifact without any regress to biology or even vitalism. His concept from 1951 is still important since its analogy to biology is obvious, including the important turn that biochemistry from this standpoint is only a special instance of a general model. Not the material itself or the material building blocks are crucial but the general principle of organization, which comprises a complete description of itself. We then can further ask where the description comes from. Instead of assuming a preexisting plan of the machine we can also develop machines that in a first step construct a self-description and only then start to reproduce according to that construction plan. For kinematic self-replicating machines Freitas/Merkle provides a comprehensive overview [17].

The Hamilton-Path-Problem-solving biological computer described above includes the growth and duplication of bacteria and thus can be analyzed under the aspect of self-reproduction. By moving from computing that just uses DNA as material to living bacterial computers researchers expect several improvements [9]: „Programming bacteria to compute solutions to difficult problems could offer the same advantage of parallel processing that DNA computing brings, with the following additional desirable features: (1) bacterial systems are autonomous, eliminating the need for human intervention, (2) bacterial computers can adapt to changing conditions, evolving to meet the challenges of a problem, and (3) the exponential growth of bacteria continuously increases the number of processors working on a problem." In the examples of the bacterial computer and the maze-solving slime mould the self-maintenance and growth of the system and its computational qualities are not separable as in the case of the extended von Neumann self-reproduction. The exponential growth of the bacteria respectively the growing slime mould are the computation. Computation and self-replication are not separated but a merged procedure. The growth is part of the qualities of the biosystem while its computational qualities are externally imprinted by means of preparation of the system.

At this point our notion of computation and of machines enters the realm of *autopoiesis*. According to Maturana and Varela "an autopoietic machine is a machine organized (defined as a unity) as a network of processes of production (transformation and destruction) of components which: (i) through their interactions and transformations continuously regenerate and realize the network of processes (relations) that produced them; and (ii) constitute it (the machine) as a concrete unity in space in which they (the components) exist by specifying the topological domain of its realization as such a network" [15]. The aspect of autopoiesis, e.g. autonomous and self-maintaining components that recursively reproduce themselves, is like self-assembly not sufficient to explain the total capabilities of biological machines. Classical autopoietic systems have no apparent inputs or outputs. They don't produce other things than themselves. While traditional machines like assembly lines or computers are allopoietic, which means they typically produce something different from themselves, for examples cars and furniture in the case of the assembly line, or information in the case of computers. To understand biological computation on the basis of self-reproduction the machine therefore needs autopoietic as well as allopoietic qualities. John von Neumann's self-reproducing machines already comprise both aspects of production. His concept of self-reproduction can easily be extended in order to gain machines that not only produce copies of themselves but also additional machines. To achieve that, we just have to feed the above-described self-reproducing machine with a description of itself together with a description of another machine. This extended machine will first reproduce itself and than construct the second machine implement a completely different function.

With respect to qualities of biological organisms like self-assembly, self-reproduction, self-maintenance, self-repair, self-improvement, and other self-strategies, we still lack a deep understanding of the formal principles for these different constellations of self-reference. We need a much broader basis of formal models and practical examples that explore the variety of configurations between symbolic descriptions and manageable material implementations. Although the central theoretical standpoint states that the discussed life-like qualities are a result of logical organization principles and not of the material, for real machines we have to bring the material qualities and the logical construction into line.

5. Convergence of mechanic and biologic qualities

In the previous sections we have concentrated on the shift of our image of the machine through the implementation of biological principles. But our understanding of organisms will conversely also shift through the application of mechanic principles in the realm of the living. Similar to the research in artificial intelligence, which has changed our concept of human intelligence, the development of biological machines will change our image of the living. We have learned that the computer is able to reproduce certain logical operations and thereby is able to model particular mental work. But at the same time we understood that human thinking is able to transcend this form of thinking. A formal model for logical reasoning is not identical to human thinking.

Synthetic biology and biological computing have in common that they work with living material. Engineers have started to build systems on the basis of existing organic elements or the use of mechanic, electronic or chemical components and organize them in such a way that they show organic qualities. Thereby the methodologies for the realization of biological machines not only run into deep ethical problems but also into fundamental problems concerning the range of the involved formal models. The strength of the formal approach is that it can capture human logics and experience on the basis of successive rules, but they are always external models. A cake recipe only includes instructions for handling the ingredients of the cake and the tools for making it. But if we exactly follow the given directions the cake will succeed. Although the recipe does not contain information about the flavor it nevertheless transports the taste of the cake. The taste of the cake is an interaction of the employed material and the gustatory nerves, not of the algorithm. Nevertheless, if we change the instructions the taste will be different. Algorithms are a powerful approach to transform human experience into mechanic actions in space and time. But algorithms are still reductions. Of course we always can augment algorithmic descriptions by additional formal models, e.g. a functional description of the interaction between taste buds and cake ingredients. We have learned that we need different models of abstraction to describe different phenomena. The fundamental presumption of the artificial life approach is that in the end any phenomenon can be grasped by adequate logical structures without any regress to subjective experience or other non-formal qualities. But the formalizations we considered so far are objectifications that describe the systems from the outside. This perspective is sufficient for classical machines, but for biological machines this viewpoint needs to be complemented by inside perspectives and descriptions. Here we meet the established question in philosophy, the "What's it like to be?" argument. The philosopher Thomas Nagel argues that an organism has conscious mental states, "if and only if there is something that it is like to be that organism –something it is like *for* the organism" [22]. He also argues that the subjective aspect of experience will never be sufficiently explainable by objective methods. What philosophy calls qualia would therefore never be describable by reductive formal models. But for biological machines we nevertheless have to ask, whether we create subjective worlds for these machines, when we realize them on the basis of organisms and living materials.

From biological research and especially physiological experiments we have learned long-since that organisms live in different milieus (*Umwelten*), even though they share the same environment (see for example [18], [19]). The relationship of an organism with its environment is determined by a functional circle, which inseparably connects its inner world with the outside. Organisms create and reshape their Umwelt by interacting with the environment through their *receptors* and *effectors*, which are central components of a closed feedback circle. Since the *receptors* and *effectors* of various species and their typical environment are normally different, the Umwelten of organisms will consequently also differ drastically. Jakob von Uexküll showed that different species live in different worlds. He has argued that space and time are subjective products of organisms that depend on their physiological characteristics. The space of a worm is different to the space of a bird. While for a slug all movements in its Umwelt is faster than in our human perception the Umwelt of fish that live of fast moving prey, the environment is moving in slow motion compared to our human perception. The milieus of living beings are inseparably connected to their inner worlds. What do the milieus of biological machines look like?

We are already able to realize new experiences of self-perception with fairly primitive means. Installing a first-person-view-flight-kit on a model aircraft enables a subjective experience of an aviation-perspective. Right in the moment when you feel up there in the aircraft you might suddenly spot yourself standing on a field deep down and perceive yourself as a spatially distributed individual. Those simple examples make clear that we have to reopen the relations between the Umwelt and the inner world for biological machines. Living beings normally have a well-defined spatial border. But where is the border of a biological machine? What belongs to the machine and what to its environment? Any modular constructed machine can easily be extended by additional parts, even with parts that have their own local controlling elements and just communicate with already existing controlling device to achieve a certain overall function. The question of an exact border and of the inside-outside-relation is in the world of classical machines not least a question of perspective. What do such possibilities of reconfiguration mean for organism-like machines? In what circumstances makes it sense to speak of the milieu of a machine, how is it constructed, what does it look like, and how is it connected to its construction plan? Since biological machines will be implemented on the basis of formal descriptions, we can expect that some of our well-known mechanical features will coalesce with typical biological qualities. To give another simple example, we can ask whether biological machines will have an on-off button. Or do they have to die like organisms? If life is a quality of the logical organization and not the applied material there is no reason not to abruptly stop a biological machine and proceed at exactly the same *state* somewhere later in time. One of the most important qualities of classical machines is their modular and hierarchical organization. Complex machines are designed and manufactured from standard elements and put together in hierarchical organized levels. The main difference to complex biological systems, which are also organized in modules and hierarchies, is their openness against exchangeability of its parts. What does it mean from the subjective perspective of a biological machine, if we are able to exchange receptors and effectors and thereby enforce new milieus?

6. Conclusion

Our machines begin to resemble living creatures. They are self-assembling, self-maintaining, inherently non-terminating, massively parallel, stochastic, adaptive, and self-modifying. For present production or computing machines, the task of programming can be described as finding a correct sequence of elementary operations so that a certain function is performed or in the case of computing a certain type of end-states will be reached. In contrast, the programmer of a biological machine concentrates on the proper preparation of a self-controlled biological process. Solving calculation problems with *living materials* is not a result of stepwise controlling a machine in the traditional sense but of reconfiguring the initial conditions of a biological system and its environment. From the material perspective a computer can be seen as an object that by reconfiguration can implement different functions and thus produce different answers to different questions. What we are looking for are universally reconfigurable systems that for example allow for solving any computational problem or produce any achievable device.

Architects, engineers and designers who are interested in self-controlled production methodologies for everyday objects or buildings come across the same issues as biologists and computer scientists. It is obvious that different fields like biocomputing, biotechnology or the industrial production of "smart" materials face the same type of self-referential logical problems. Therefore insights into human designed self-referential processes can be generally applied for the construction of macroscopic as well as microscopic objects.

Today our machines have *sensors* and *actuators*. Biological machines consequently will from a certain point have *receptors* and *effectors*. This reopens Uexküll's question about the inner world and Umwelt of animals for machines. We will have to face the problem of what its like to be a machine. Research in *artificial life* and on biological machines start from the assumption that life can be separated from its material basis and aliveness will finally turn out to be a property of the logical form. Many qualities of traditional machines like modularity, extendibility, hierarchical composition and others are also elementary qualities of these logical forms. The essential point of biological machines is not that we just reconstruct and modify some well-known lifeforms on an artificial basis, but that our technical society enters a next level. It is not familiar life that is re-created, like it was not intelligence that has been re-created by AI, but new forms of abstracted and functionalized qualities of the living. It seems to be obvious that in the future we will face many machines that show a mixture of mechanic and organic qualities.

References

[1] L. Mumford, Technics and Civilization, George Routledge & Sons, London, 1947.
[2] C. Mitchum, Thinking trough Technology – The Path between Engineering and Philosophy, The University of Chicago Press, Chicago, 1994.
[3] G. Klaus (ed.), Wörterbuch der Kybernetik, Bd. 1, Frankfurt am Main, 1971, p 380.
[4] M. D. Laublichler, Systemtheoretische Organismuskonzeptionen, in: U. Krohs, G. Toepfer (eds.), Philosophie der Biologie, Suhrkamp Verlag, Frankfurt am Main (2005), p. 123.
[5] C. G. Langton (ed.), Artificial Life, Addison-Wesley Publishing Company, 1989, p. 1.
[6] J.C. Sheperdson, H.E. Sturgis, Computability of Recursive Functions, Journal of the ACM, Vol. 10 (1963) 217 – 255.
[7] Turing, A.M.: Intelligent Machinery, see: The Turing Digital Archive. http://www.turingarchive.org/browse.php/C/11, Image 8. (March 2012)
[8] C. Tan, H. Song, J. Niemi, L. You, A synthetic biology challenge: making cells compute, Mol. BioSyst., 3 (2007) 343 – 353.
[9] J. Baumgardner, et. al., Solving a Hamiltonian Path Problem with a bacterial computer, Journal of Biological Engineering, 3:11 (2009). http://www.jbioleng.org/content/3/1/11 (March 2012).
[10] A. Tero et al., Rules for biologically inspired adaptive network design, Science 327, 439 (2010).
[11] G. Rozenberg, The Nature of Computation and Computation in Nature, Acceptance speech on

receiving an Honorary Degree from the University of Bologna, www.liacs.nl/~rozenber/bolognae.pdf (March 2012)
[12] F. Reuleaux: Contributions to 19th century kinematics and theory of machines, Appl. Mech. Rev., Volume 56, Issue 2 (2003) 261 – 285. http://dx.doi.org/10.1115/1.1523427 (March 2012)
[13] J. A. Pelesko, Self Assembly – The Science of Things That Put Themselves Together, Chapman & Hall/CRC Press, 2007.
[14] M.A. Arbib, From Universal Turing Machines to Self-Reproduction, in: R. Herken (Ed.), The Universal Turing Machine – A Half-Century Survey, Springer-Verlag, Wien NewYork (1994) 161 – 172.
[15] H.R. Maturana, F.J. Varela, Autopoiesis and Cognition: The Realization of the Living, Boston Studies in the Philosophy of Science, Vol 42, D. Reidel Publishing Company, 1st edition, 1980.
[16] D. R. Hofstadter, Gödel, Escher, Bach – An Eternal Golden Braid, Vintage Books Edition, 1980.
[17] R. A. Freitas, R. C. Merkle, Kinematic Self-Replicating Machines, Landes Bioscience, 2004.
[18] J. von Uexküll, G. Kriszat, Streifzüge durch die Umwelten von Tieren und Menschen. Rowohlt Verlag, 1956.
[19] J. von Uexküll, Umwelt und Innenwelt der Tiere, Springer Verlag, 1909.
[20] J. Bongard, V. Zykov, H. Lipson, Resilient Machines Through Continuous Self-Modeling, Science, Vol 314 (2006), 1118 – 1121.
[21] H. Sato, M. Maharbiz, Recent developments in the remote radio control of insect flight, Frontiers in Neuroscience, Vol. 4 (2010).
[22] T. Nagel, What is it like to be a bat?, in: P. Alan (Ed.), A Historical Introduction to the Philosophy of Mind, Broadview Press (1997), 391ff.