Self–reproduction by self–assembly and fission*

Jiří Wiedermann

Institute of Computer Science, Academy of Sciences of the Czech republic,
Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic
jiri.wiedermann@cs.cas.cz

Abstract. We introduce so–called biomata which represent a novel approach to the construction of self–reproducing automata within the automata theory. The design of our automata has been motivated by the ideas of cellular biology on the origin of life. Unlike the von Neumann's model our model replicates by fission and need not give much attention to the exact guiding of its own assemblage; rather, this process relies on self–assembly abilities of the respective parts produced by the biomaton from input objects not possessing such quality. The model represents an interesting fusion of computational and self–organizational processes. We believe that by capturing the basic aspects of the assumed origin of real life our modelling leads to a conceptually simpler and hence more plausible scenario of natural self–reproduction than the previous attempts did.

1 Introduction

In late 1940s, when John von Neumann started his quest for a logical, rather than material, basis of biological self–reproduction, he first proposed a mechanistic model. It consisted of a "robot" operating in a sea of its own spare parts. The robot had some elementary functions for moving around, identifying and collecting the required parts and assembling them together and possessed a tape with instructions for building a copy of itself by making use of these elementary functions. After constructing a replica of itself, the robot finally copied its instruction tape and inserted it into the replicated robot which could then start the same activities. By this design, it is generally agreed that von Neumann discovered the basic principles for the process of self–reproduction. Namely, there had to be a program, instruction sequence to be used in two different ways: (1) to be interpreted as instructions for constructing an offspring, and (2) to be copied passively, without being interpreted. Quite understandingly von Neumann was not able to construct a working model of his mechanistic self–reproducing automaton which would represent a convincing proof of soundness of his design idea. However, in 1953, following Stanislaw Ulam's vision of cellular automata, he invented a cellular automaton implementation of his mechanistic model. His

* This research was partially supported by grant No. 1ET100300419 within the National Research Program "Information Society".

cellular "robot" made use of a cellular automaton with 29 states per cell and consisted of approximatively 200 000 cells [7]. By this the topic of self–reproduction entered the field of the automata theory and it seems that until now nobody has questioned the uniqueness of von Neumann's scenario of self–reproduction in this field. There seems to be no formal computational model of self–reproduction based on a different scenario than the one mentioned above.

In this paper we suggest a different scenario of self–reproduction. Our model reflects the ideas of theoretical biology on the origin of life. According to these ideas life emerges in the form of protocells from the union of two fundamentally different kinds of replicating systems: an information genom and a membrane in which it resides (cf. [6]). Thus, replication lies in the heart of both systems. Roughly speaking, a genom controls its own replication and production and properties of parts for building the membrane which itself is a spontaneously replicating entity. The membrane shields the genom from the environment. This is enough to give rise to a self-replicating system. Addition of randomness into the genom replicating process leads to darwinian evolution of the whole system, and to so–called minimal life systems, but this question is outside the scope of the present paper (cf. [8] for an attempt to model minimal life).

In our setting, the basic functional aspects of protocells are modelled by so–called biomata. A biomaton consists of a so–called Turing computational field and its encapsulating membrane. By the activity of the Turing field certain input objects that permeate the membrane from outside are transformed into new objects with self–assembly properties; the input objects alone do not possess such properties. The membrane "grows" by incorporating new objects into the already existing membrane. All computations within the Turing field are controlled by a special object representing a finite state program playing the role of a genom. The genom itself can become a subject of the processing in the Turing field and indeed, inside the membrane its copy can be produced from suitable input objects. Providing that at that time the encompassing membrane doubles its volume, by definition it will split by fission into two parts each containing a copy of the original genom: the biomaton will reproduce itself. Even from the above rough sketch of biomata it is obvious that they combine computational processes with self–organizational ones. In our model the fission represents a non–computational process whose action and result are merely postulated. In real life a membrane fission is a consequence of the physical laws acting upon the membrane. Neither the self–assembly processes nor the membrane fission are under the computational control of the genom. In a sense, the respective non–computational mechanisms evoke the idea of an oracle that similarly as in the case of Turing machines is used for achieving the effects that principally cannot be obtained in a pure computational way by a device itself.

We believe that the main contribution of our paper lies in pointing to a new research direction within the automata theory that aims towards the design and exploitation of a formal model of self–replicating automata which are not based on classical computational mechanisms as the von Neumann's models were. While for biologists this model represents an abstraction of a protocell,

within automata theory it formalizes an alternative scenario of self–replication that is closer to reality than the previous models. The mathematical theory of self–assembly is an emerging research field (cf. [1]) and our framework opens new avenues for its further development.

The paper consists of 4 sections. The first section contains the introduction. The biomaton itself is described in Section 2 in two subsections. The first subsection introduces the so–called multitransducer that represents the computational analog of a genom and in fact defines the Turing computational field. The second subsection explains how the multitransducer must be designed in order to produce its encapsulating membrane, how its genom gets replicated, and finally how the fission of the membrane is achieved. A multitransducer operating in this way gives rise to a biomaton. Section 3 discusses the previous achievements from the viewpoint of the automata theory with regard to cellular automata, P–systems, evolution theory and artificial life. The closing fourth section contains the summary of the achievements.

The paper describes the research in progress and so far neither the model nor the formalism and the terminology are definitive; similarly, the results and their interpretation only start to emerge. Some preliminary ideas related to the concept of a multitransducer in the context of minimal life have been presented in a workshop on membrane computing [8].

2 The Biomaton

Interactive finite multitransducer First, we will concentrate on the information processing aspects of our model. For that purpose we will make use of modified finite automata. We will use them in the mode of transducers (or as Mealy automata) — i.e., as automata processing multisets of the finite input strings of symbols and producing similar strings of output symbols. Moreover, we will consider a so–called *multitransducer* which is a multiset of transducers of finitely many types that all work asynchronously. Even though the description of a multitransducer, that is, of all types of automata together, is finite, the cardinality of their multiset can be arbitrarily large. This cardinality varies with time and depends on the number of strings that are available for processing at each time — see the description of the multitransducer's activity in the sequel. Thus, from the computational viewpoint a multitransducer is a highly parallel information processing device.

Now we will give a formal definition of a multitransducer for the simplest case when each automaton reads its inputs via a single input port. Then we will describe the way the machine works.

Definition 1. *An interactive asynchronous finite multitransducer with single–input ports is the six–tuple* $T = (I, O, S, B, F, \delta)$*, where*

- I *and* O *are finite alphabets of symbols,* I *is the input and* O *is the output alphabet;*
- S *is a finite alphabet of states;*

- $B \subseteq S$ *is the subset of initially active states;*
- $F \subseteq S$ *is the subset of final states;*
- δ *is the transition function of form* $I \times S \to S \times O \times S \times \{0,1\} \times \mathbf{N}$ *which for each* $v \in I$ *read (and "consumed") at the input port of some automaton and each state* $s \in S$ *assigns a new state* $r \in S$*, sends* $w \in O$ *at the input port, and sets the activation value of state* $q \in S$ *to either 0 or 1; here 0 denotes non–initial (passive) and 1 initial (active) state; this is formally written as* $\delta(v,s) = (r,w,q,0,t)$ *or* $\delta(v,s) = (r,w,q,1,t)$*, respectively;* t *is the speed parameter saying that it takes* $t \in \mathbf{N}$ *time units to realize the transition at hand.*

In the multitransducer each type of the Mealy automaton is described by its own transition function of form as given in Definition 1. We assume that the multitransducer finds itself in an environment consisting of a multiset of strings. Also this multiset is not given beforehand, it can change over time, that is, the multiplicity of the same strings in it can vary. A multitransducer operates by systematically and repeatedly transforming strings into other strings. The input strings are read sequentially, symbol by symbol by automata via their input ports and the output strings are produced in a similar way at their output ports. The automata work in an asynchronous manner. We assume that each automaton has its own clock. For simplicity we also suppose that in all automata the duration of one unit of time is the same, however, the clocks are not synchronized. Since there is no notion of global time it is not possible to define a "configuration of the system" at a given time. By allowing more than one input port each automaton can be designed so as to be able to process several strings in parallel, similarly as classical multihead automata. In order that the operation of a multitransducer can work smoothly we assume that each automaton reads the strings in a selective way, that is, it has a specific ability to find this string in the environment for the processing that is programmed for, as long as such a string exists in the environment. This property can also be seen as a property of the environment — it is as though the environment attempted to process each string by each multitransducer's automaton, and as long as such a pair (string, automaton–able–to–process–this–string) exists, then processing takes place. Henceforth, the environment has a potential for realization of highly parallel computations. Should there be two automata able to process a given string, one of them is selected randomly. After being processed, a string "disappears", being transformed into a corresponding output string. The environment in which a multitransducer operates in the way described above is called *Turing computational field*. The apparently strange behavior of the Turing computational field is motivated by the idea of modelling the chemical reactions by such a field. Namely, certain chemical reactions take place if there are corresponding reactants (i.e., inputs) available and only if there is a corresponding catalyzer (i.e., a corresponding automaton with an activated initial state) ready. Note that in a similar way also the computations within the membrane systems are defined (cf. [4],[5]).

A multitransducer differs from the set of standard Mealy automata in two aspects. First, the set of initial states of all automata is not fixed, i.e., it is not given once for all at the beginning of the computation. Rather, depending on the course of computation this set can change with time: some states can loose their property of being initial states, others can obtain this property. The instructions for activation/deactivation of initial states are included in the transition function of the multitransducer. The states that are at the moment initial states will be also called *active states*. The initial activation of states is given by set B as a part of the multitransducer definition. Depending on the inputs read in subsequent steps and on their order, (recall that automata work asynchronously) the activity of states can change. The dynamic activation of its states enables the multitransducer to switch "off" or "on" certain automata and to control the interactive processing in this way. The automata whose initial state has been deactivated cease to be a part of the Turing computational field and remain so unless their initial state gets reactivated by an other automaton. The intended use of the initial state activating mechanism is to model the gene switching in real cells. The second point of the departure of a multitransducer from the definition of classical automata is the possibility of controlling the processing speed of individual transitions. In order to be able to change the speed of transitions we assume that with each transition, a so–called *speed parameter* (a natural number), is associated that defines the speed taken by the realization of that transition. This possibility can be used in tuning the synchronization among various automata[1].

With respect to the process of self–reproduction it seems natural to claim that a multitransducer cannot transform non–empty strings into empty strings and vice versa, that is, a multitransducer can neither generate something from nothing nor nothing from anything. Note that, syntactically, in the transition function representation, there is no visible "boundary" between the automata of which the multitransducer consists — from the description of its activity it is clear that once the processing of a string gets started by a transition containing an active state on its left–hand side the processing will be prolonged by any transition that applies to the new state and the symbol read at that very moment. In this way, the processing goes on via a chain of admissible transitions until the string gets "consumed" and a final state is reached. If the initial state of the automaton at hand is still active, than a new processing can be launched. In what follows instead the term "string" we will often use the term "object" to denote either a symbol or a string of symbols. Depending on the context we will consider an object either as data it represents or as a physical object possibly having certain self–assembly properties which will be used to our advantage.

Encapsulating the multitransducer Since the final goal of our efforts is modelling of self–reproduction we will have to "engage" a multitransducer in its own replication. The resulting device will be called a biomaton. To this end, following the ideas from cellular biology, we will let the multitransducer replicate its

[1] The speed parameter can be avoided at the expense of allowing epsilon transitions in the formal definition of a multitransducer.

"genom" and build its own "body" — a *membrane* endowed by a self-replicating property. The purpose of the membrane will be

- to protect the multitransducer's control mechanisms from the unwanted influence of the environment;
- to restrict the range of multitransducer's computational influence (i.e., the Turing computational field) to a certain finite domain;
- to let selectively pass some input objects into the membrane;
- to enable the biomaton's development (especially its growth and multiplication).

The membrane is constructed of so–called *tiles* which are special objects produced by specific automata in the Turing computational field encapsulated by the membrane. The membrane allows translocation of certain input objects from the outside environment; tiles are produced from such input objects. Tiles have a special shape and special properties. Their shape is such that they are able to form a three–dimensional spherical structure — a membrane which prevents the encapsulated objects to escape and allows certain input objects to enter. The tiles possess self–assembly property meaning that any random cluster of tiles that are sufficiently close to each other will spontaneously self–assemble into a membrane and, moreover, if there should be further tiles in the vicinity of such a membrane they will get spontaneously incorporated into it. In this way a membrane can grow. When roughly doubling its volume, a membrane tears in two approximatively equal parts that both spontaneously again organize into membranes. The pace of membrane growth depends on the supply of tiles which are generated by automata from elements that are not endowed by self–assembly property.

The activities of a multiset of automata within the Turing computational field are controlled by a "program" that takes the form of a rewritten tape which finds itself inside the membrane. This tape contains the description of the multitransducer's transition function in a linear form. This description consists of a series of segments each of which corresponds to one transition of form $I \times S \to S \times O \times S \times \{0,1\} \times \mathbf{N}$. Of course, such a program resembles a genom residing inside a biological cell. Similarly as a genom, it consists of a series of instructions for production of various objects that can be further used for membrane construction or for constructing the genom's copy. All this happens via activation or deactivation of the respective automata. From the viewpoint of a multitransducer, a program is an object as any other objects and therefore the program tape can also become a subject of an automaton's processing within that multitransducer. An important automaton in that respect is a copying automaton whose task is to produce a copy of the program tape. Such an automaton has two inputs — one by which it reads the current program tape and the other by which it accepts "stuff" (objects) from which a tape's copy is to be constructed. Of course, the copying process does not destroy the original tape. Note that it is (also) here where our scenario of self–reproduction deviates from the classical von Neumann's ideas. Namely, in our case in the multitransducer's description there is no need to give a "recipe" how to build the "body" — a

membrane, when and how to split it, how to see that there is a single copy of the program tape in each newly emerging membrane, etc. While this is technically possible (as shown by von Neumann), in our case the self–assembly processes, their proper triggering and timing by a multitransducer, and a postulation of a non–computational (albeit in reality natural) operation of membrane splitting take care about this kind of self–reproduction activities that von Neumann had to program laboriously.

The idea of an embodied transducer emerging above is captured in the following "descriptive" definition of a biomaton:

Definition 2. *A* biomaton *is a self–reproducing multitransducer which works in the following way:*

- *the activity of the multitransducer's Turing computational field is controlled by the multitransducer's transition function which is represented as a special object — called tape;*
- *this tape resides inside a membrane with a certain initial volume which has been constructed by self–organization from tiles that are produced by the Turing computational field from specific input objects permeating freely the membrane from outside; the respective input objects do not possess self–assembly properties; the membrane steadily increases its volume by incorporating new tiles;*
- *along with the growth of the membrane the process of tape copying is in progress; the copy of the tape is also built by the Turing computational field from input objects permeating the membrane;*
- *the growth and copying processes are synchronized so that at the time when the copying process ends the initial volume of the membrane doubles; at that time the membrane splits into two membranes, each retaining one copy of the tape.*

Note that after the fission each of the pair of newly emerging biomata has the original size of their parent biomaton. Thus, under a sufficient supply of input objects the same process can be repeated *ad infinitum*.

Even from the above informal description one can see that for its self–replication a biomaton needs a hierarchy of objects with various properties. We start with simple input objects possessing no self–assembly abilities. However, these objects must be such that a multitransducer can generate out of them other objects already possessing self–assembly properties (tiles). These self–assembly objects further self–organize into complex structures (membranes) that by definition are endowed with still other emergent properties not possessed by their parts (e.g., membrane splitting).

In order to show that the definition of a biomaton is sound one has to prove that an entity satisfying it does exist. In a sense this is a problem similar to that von Neumann faced after describing the idea of his mechanistic self–replicating robot without actually constructing it. In our case, a proof of the last claim concerning the existence of a biomaton would require a formal design of a concrete multitransducer with properties according to Definition 2. While we believe that

in principle this is possible, for the time being we feel that we do not have a sufficiently developed formalism for capturing all the necessary spacial, temporal and functional aspects of self–organizational processes needed for our purposes. Initial attempts in this direction can be seen in the emerging mathematical theory of self–assembly (cf. [1]). In [9] a so–called globular universe (a kind of cellular automaton) has been described in which the existence of self–replicating structures resembling the tape from Definition 2 is constructively shown. Moreover, these structures are shown to posses an evolutionary potential, i.e., they can evolve so as to realize any given finite control mechanism. Nevertheless, for the time being we have to refer to an "indirect" evidence pointing to the existence of biomata. Namely, self–assembly and splitting of a sufficiently large membrane which are basic assumptions postulated in our model are justified by the existence of similar phenomena in reality, at the level of real bacteria. There, the physical laws work "as needed" for a bacterium to operate correctly. Both the self–assembly property and the physical laws acting, e.g. in the case of a membrane splitting or input objects permeating the membrane, are "present" all the time without a need to be invoked; what is done in a bacterium is harnessing these essentially non-computational phenomena for the purpose of life. All these non-computational phenomena are captured by our model at the level of assumptions. This evidence from real life is supported by efforts in cellular biology for synthesizing life from scratch (cf. [3]).

3 Discussion

Let us compare "our" scenario of self–reproduction with that of von Neumann (as briefly sketched at the beginning of this paper). Obviously, the basic principles are the same: in both cases there is a program that is both actively interpreted and passively copied. But there is a difference in both approaches concerning the replication: while von Neumann builds a copy separately, outside the original body, right from the scratch, taking care over all details of the body building, in our approach a copy emerges by splitting the original body without taking care over the details of such a process. In our setting there is never a phase of a "half finished" automaton that is not yet functional. To our mind, our approach reflects the self–reproduction on the level of the simplest cells while von Neumann's approach (via cellular automata) corresponds more to multicellular organisms. In order that a cellular automaton should reproduce itself it needs a supply of finished fully functional cells that need to be only activated. In our case, the transformation from a "non–living" to an "animated" entity is more gradual: we start with input objects having no self–assembly properties, produce out of them objects with such properties, and finally let them self–organize. It seems that such a process needs less sophisticated central computational control and leads to a better parallelism exploitation. That is perhaps why it has been favored by evolution.

The idea of biomata brings new impetuses into the automata theory since it introduces a new computational model mixing data processing with object

construction while utilizing non–standard computational resources. This leads to new classes of computational problems which wait to be formulated, formalized and solved. A characterization of the processing power of multitransducers (or biomata) is open. Undoubtedly, any progress along these lines must be matched by an analogous progress in the theory of self–assembly.

In the context of computational models it is of interest to discuss the relation of biomata to the membrane systems (cf. [5]). Although originating from the same source of ideas (viz. cell biology), we see the main differences between the two systems both in their different purposes for which they were designed and in their different architecture. These points are summarized bellow:

- in standard membrane systems the membrane is a part of the model that is not produced by a model; in the case of biomata, the membrane is a product of input processing;
- for their activity the membrane systems make use of a hierarchy of distributed computations; the biomata make use essentially of three different cooperating resources:
  - distributed computational power controlled centrally via biomaton's tape and state switching;
  - distributed self–assembly processes governed by local assembly rules;
  - non–computational phenomena modelling the effect of physical laws;
- the computations of membrane systems are governed by rules, whereas the biomata are controlled by finite state machines (of course, the computational power of both mechanisms is the same);
- the "program" of biomata is both actively interpreted for controlling the biomaton's activities and passively copied for the self–reproduction purposes; without modifying their functionality, this cannot be mirrored by the membrane systems;
- the primary aim in the design of membrane automata has been their computational universality, as indicated e.g. in [4]; in the case of biomata, the aim has been to achieve their self–reproduction ability;
- an evolutionary aspect can easily be introduced into a framework of biomata; in fact for such a purpose it is enough to admit errors in the copying process of genetic information. By the very construction of biomata, the "genotype" of the system is closely related to its "phenotype" and thus the system as a whole can become a subject of darwinian evolution. The membrane systems cannot be straightforwardly adapted for such a modelling.

We believe that our model is of interest also in the context of cellular and evolutionary biology, exactly for reasons mentioned in the last item of the previous paragraph: it enables a further insight into the mechanisms of adaptive evolution and perhaps will also enable computational experiments along these lines.

To some extent, perhaps the biomata can also contribute to the eternal question on the relationship between living and non–living matter (cf. [2]). Namely, in biomata we start with the input objects not endowed by self–assembly property, in the next step we construct ("compute"?) objects already possessing such

a property, and we end up with a "living matter", to some extent. All this happens in an abstract medium, within a mathematical model; a possible candidate for such a model has been proposed in [9]. In this context, biomata are a typical instance of artificial life. To what extent our models correspond to reality remains to be seen.

The last remark concerns the relation between computing and constructing. For our approach it has been of a prime importance that the objects can be seen both as data for information processing (e.g. the multitransducer's tape, the "genom", has been seen as a set of instructions to be interpreted by the Turing's computational field), and real physical objects obeying physical (or chemical) laws (e.g. the tape can be copied, from the input objects self–assembly objects can be produced, the tiles organize themselves spontaneously into a membrane, an oversized membrane ruptures and splits). Perhaps we are witnessing the dawn of a new field of computing, a "constructive computing", with self–assembly being its harbinger.

4 Conclusion

We devised a biomaton — a novel model of a self–reproducing machine which is driven by a finite state program. From the surrounding input objects a biomaton constructs its "spare" parts endowed by self–assembly properties. Consequently, these parts organize themselves spontaneously into the biomaton's "body" that takes the form of a membrane. Eventually, the biomaton produces a copy of its program and splits its membrane into two equal parts, each containing one copy of the original control program. When compared with the standard von Neumann's model of self–reproduction our scenario leads to a new model of self–reproduction that captures this process at the level of a single cell rather than at the level of multicellular organisms as von Neumann's cellular automaton model in fact does. In the automata theory, in the related context of cellular and evolutionary biology, and in artificial life our model seems to have a great potential for its further development and investigations. The new model presents a case of constructive computing, in which the physical properties of data representations are equally important as the computational properties of the data themselves.

References

1. Adleman, L.: Toward a mathematical theory of self-assembly. Tech. Rep. 00-722, Dept. of Computer Science, University of Southern California, 2000.
2. Brooks, R.: The relationship between matter and life. Nature, Vol. 409, 18 January 2001, pp. 409–411
3. Hanczyc, M. M., Fukijawa, S. M., Szostak, J. W.: Experimental Models of Primitive Cellular Compartments: Encapsulation, Growth, and Division. Science. 302 (2003), pp. 618-622.
4. Paun, G., Rozenberg, G.: A Guide to Membrane Computing. Theoretical Computer Science 287 (2002), pp. 73-100.

5. Paun, G.: Membrane Computing. An Introduction, Springer, 2002
6. Szostak, J.W., Bartel, D.P., Luisi, P.L.: Synthesizing Life. Nature 409 (2001) 389-390.
7. von Neumann, J.: Theory of Selfreproducing Automata. A. Burks (Ed.), University of Illinois Press, Urbana and London, 1966
8. Wiedermann, J.: Coupling computational and non–computational processes: minimal artificial life. Pre–proceedings of the Fifth Workshop on Membrane Computing (WMC5), G. Mauri, Gh. Paun, C. Zandroni (Eds.), Dept. of Comp. Sci., University of Milan — Bicocca, Italy, June 16–16, 2004, 444 p.
9. Wiedermann, J.: Self–reproducing self–assembling evolutionary automata. Manuscript, September 2004