Globular Universe and Autopoietic Automata: A Framework for Artificial Life*

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 present two original computational models — globular universe and autopoietic automata — capturing the basic aspects of an evolution: a construction of self–reproducing automata by self–assembly and a transfer of algorithmically modified genetic information over generations. Within this framework we show implementation of autopoietic automata in a globular universe. Further, we characterize the computational power of lineages of autopoietic automata via interactive Turing machines and show an unbounded complexity growth of a computational power of automata during the evolution. Finally, we define the problem of sustainable evolution and show its undecidability.

1 Introduction

Some 50 years after von Neumann made public his result on self–reproducing automata it appears that this result should not be merely seen as convincing evidence of the existence of complex artificial self–reproducing structures. As McMullin pointed out [3], von Neumann's effort is to be also understood as the first step towards a construction of structures which exhibit an ability of evolutionary complexity growth *"from simpler types to increasingly complicated types",* as von Neumann put it. Unfortunately, von Neumann, in his work, did not tackled the issue of complexity growth of self–reproducing automata in more detail. In conclusion of his work [4] he merely indicated that these are the random mutations which should play the role of the respective evolutionary mechanism and included this problem into the list of problems which should be still investigated in the future (cf. [3] for a detailed discussion of that matter).

Obviously, von Neumann's intuition was right as we nowadays know from the theory of cellular and evolutionary biology. Nevertheless, a problem remains whether random mutations are the only mechanism for the artificial evolution, or whether there exists another mechanism resulting into more efficient evolution. A related question is whether the hypothetical "other" mechanism can become a subject of the evolution and how this can be achieved. Last but not least, is there

* This research was carried out within the institutional research plan AV0Z10300504 and partially supported by grant No. 1ET100300419 within the National Research Program "Information Society".

an unbounded evolution, generating self–reproducing automata with increasingly complicated behavior, from a computational complexity viewpoint? What is the computational power of an unbounded evolutionary process? And can we decide whether a given self–reproducing machine will give rise to an endless evolution under given conditions? Some of these questions were discussed in [3], indicating possible ways to answer them.

In this paper we sketch formal answers to the above mentioned questions. For such a purpose we will present concepts and the results which emerge from the author's recent studies of the related problems. In their preliminary form the respective results were presented partly in a recent workshop [6] and partly as a technical report [7]. Contrary to these works the present paper is a survey paper stressing the main ideas behind the respective models and results rather than their formal description and proofs. This is so because a technical explanation would require more space than it is available in this contribution. The interested reader is invited to read the original sources.

In Part 2 we introduce an original model of globular universe which is the basis for construction of so–called self–reproducing globular automata. In Part 3 we informally define a simple abstract model of self–reproducing automata — so–called autopoietic automata — which obey self–reproducing abilities by their very definition. From their own genetic information these automata are able to compute new, algorithmically modified genetic information and pass it to their offsprings. In Part 4 we show that an arbitrary autopoietic automaton can be realized in a suitable globular universe. The reproduction of the resulting automaton proceeds, in a similar way, as the replication of a DNA strand in the living cells. Further on, we will deal exclusively with the autopoietic automata. In Part 5 we will characterize the computational power of the lineages of such automata by equalling it to the power of interactive Turing machines. We also mention the problem of a sustainable evolution and show its computational undecidability. Part 6 asks for the existence of an autopoietic automaton with an unliminted "self–improvement" property. Such an automaton initiates an evolution generating subsequently all autopoietic automata. This automaton controls its mutations is such a way that the genetic code "syntax" gets preserved; the evolution of the mutational mechanism itself guarantees the coverage of the entire evolutionary space. Section 7 is a closing part.

2 Globular universe

The basic design idea of a globular universe comes from the following gedanken-experiment with a classical cellular automaton. Imagine a standard two–dimensional cellular automaton and cut it by vertical and horizontal cuts into single cells each of which is seen as a single finite–state automaton. Doing so the cells will lose contacts with their neighbors. Let the cells freely fly around in the space, occasionally colliding one with each other much like as under the Brownian motion. Under a collision the cells come again into a passing contact and on that occasion they perform a "computation" (a state transition) similarly as in the case when they were bound in a rigid grid of a cellular automaton. In our case, however, the colliding cells are not allowed only to change their states, they can also change properties of their contact domains. E.g., after the collision the originally neutral contact domains can become "sticky" and thus both cells will get bound together. Or the contact domains will become repulsive — the cells will be forced to move apart from each other. Thus, as a result we get a kind of a programmable matter, or a universe with programmable particles. The previous picture is not quite correct yet as far as the analogy with the Brownian motion is concerned. In our model we do not consider any kinetic aspects of cells moving in the space (like in lattice–gas automata); we are only interested in their final destinations, where they collide with an other object. Thus, to simplify the model further, we will assume that a cell in a proper state (i.e., having the required properties) "will come flying" to a place where we need it and at due time. Such an approach has great advantages over the probabilistic approach where we must care about probabilities by which the required phenomena occur. Our assumption resembles a nondeterministic choice: of all possible alternatives which can in principle occur we assume that the one that suits to our purposes will occur, indeed. Then, if adequately programmed a cell can be incorporated into an object at hand which in this way is being built by self–assembly. The last cosmetic change of the previous ideas is the transformation of square–shaped cells of the original cellular automaton into *globules.* They are all alike, of the same size and on their spherical surface on exactly (computationally) defined locations they have *contact domains* whose *attraction properties* are controlled by a *finite state mechanism* which is the same for all globules. As a result, the *globular universe* is created by an infinite multiset of globules with a fixed set of contact domains defined on their surface. The properties (i.e., the state plus the attraction abilities of contact domains) of globules at interaction times are controlled by so–called *interaction function* (or relation) which says how the states and attraction properties of two interacting globules will change after the interaction (for more details see the original paper [6]). A somewhat similar experimental framework using a finite state mechanism to model contact properties of polyhedral particles moving in a fluid in a simulated physical setting has been described in [2].

It is clear that our model of globular universe is a generalization of both classical cellular automata and contemporary models of self–assembly (cf. [1]) as used in computer science.

3 Autopoietic automata

Our next goal will be a construction of a so–called self–reproducing globular automaton. Its construction will be based on the principles of self–reproduction as "discovered" by von Neumann [4]: the same "program" will be used both for controlling automaton's own computational behavior and as a template for the production of an other program which will later control the offspring of the self–reproducing automaton at hand. In a globular universe we cannot take advantage of the fixed grid structure underlying the classical cellular automaton as it was the case with the von Neumann's design. Namely, we cannot construct a copy of the original machine at a priori computed "free" grid locations in a sufficient distance from the parental machine. Instead, we will make use of the self–assembly properties of the elements of a globular universe. Prior to immerging into a design of a self–reproducing globular automaton we describe a more abstract object, a so–called autopoietic automaton[1] whose formal definition can be found in [7]. This (mathematical) object will serve as a kind of abstract specification of a self–reproducing globular automaton. An implementation of an autopoietic automaton in a suitably designed globular universe we give rise to the required self–reproducing globular automaton.

Autopoietic automata are nondeterministic finite state machines capturing the elementary information processing, reproducing and evolving abilities of living cells. Technically, an autopoietic automaton is a nondeterministic transducer (a Mealy automaton) computing, in addition to the standard translation also the transition relation of its offspring. The design of an autopoietic automaton supports working in two modes. Both modes are controlled by a transition relation. The first of them is a standard *transducer mode* in which external input information is read through an input port. In this phase the results of a computation (if any) are sent to the output port. The second mode is a *reproducing mode* in which no external information is taken into account. Instead, the representation of a transition relation itself is used as a kind of the internal input. For this purpose the representation of automaton's own transition relation is available to an autopoietic automaton on a special, so–called *program tape.* It is a two–way read–only tape. The results of the computational steps in the reproducing mode are written on a special one–way write–only *output tape.* Of course, both tapes mentioned before are finite.

In general, the *transition relation* of an autopoietic automaton is a finite subset of the cartesian product $\Sigma \times Q \times \Sigma \times Q \times D$, where Σ is an ordered set of input and output symbols, Q is the ordered set of states and D is the ordered set of move directions of the head on the program tape. Both sets Σ and Q can be infinite, whereas $D = \{d_1, d_2, d_3, d_4\}$. On the program tape the elements of these sets are unary encoded, i.e., an element $\sigma_i \in \Sigma$, $q_i \in Q$, or $d_i \in D$ is encoded as 0^i (i.e., as the string consisting of i zeros). It follows that a tuple $(\sigma_i, q_j, \sigma_k, q_m, d_n) \in \Sigma \times Q \times \Sigma \times Q \times D$ is represented on the tape as $10^i10^j10^k10^m10^n1$. Tuple $(\sigma_i, q_j, \sigma_k, q_m, d_n)$ is called an *instruction* of the transition relation; a representation of an instruction on the tape is called a *segment.* The semantics of an instruction is as follows: *"reading σ_i in state q_j the automaton outputs σ_k, enters state q_m and shifts its head in direction d_n,"* where the value $n = 1$ means the shift by one cell to the left, $n = 2$ to the right,

[1] The name of autopoietic automata has been chosen both to honor the Chilean biologists Varela and Maturana who coined the term of autopoiesis and also to distinguish these automata by the name from the notoriously known classical notion of self–reproducing automata which are a kind of cellular automata. The autopoietic automata are definitely not meant to model autopoiesis in the sense of Maturana and Varela.

$n = 3$ means no shift and $n = 4$ means "undefined". The segments are written on the program tape one after the other.

Formally, the mode of activities of an autopoietic automaton are derived from the type of state that the automaton is in at that time. For such a purpose the states of an automaton are split into two disjoint sets: a translating and a reproducing set. To distinguish the types of individual states in their tape representation also syntactically we will make use of the last component in the five–tuple representing a segment. When, after entering state q_m, the automaton should work in a translating mode (i.e, when q_m is a translating state), the component in the respective instruction will take value d_4 and in the respective segment value 0^4. Otherwise, to indicate the reproducing states, this component will take values d_1, d_2, or d_3 denoting the move direction for the program tape head. An autopoietic automaton will start its activity in an initial translating state and while remaining in this type of states the automaton continues working in the translation mode. The respective instructions are characterized by value d_4 in their last component signalling the absence of head moves. After the first entering a reproducing state the automaton must stay in the reproducing mode. The reproducing mode terminates by entering the final reproducing state. At that moment the automaton splits into two automata by definition. The first of the two will "inherit" the program tape of the parental automaton as its own program tape (denoted as Program 1 in Fig.1), whereas the second one will use, in place of its program tape, the output tape of the parental automaton (denoted as Program 2). Then both new automata start their activities with empty output tapes. Each new automaton is seen as an offspring of the original automaton. Clearly, one of the offsprings will always be identical to its parent, but the other offspring can be different from its parent, indeed. Note that it has

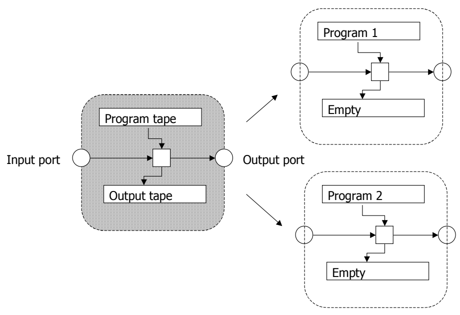

Fig. 1. Autopoietic automaton reproducing by fission

been the admittance of infinite symbol and state sets allowing an autopoietic automaton's offspring to work with a larger set of symbols or states than its parent could. By this we have opened a possibility of an evolution leading from simpler to more complicated automata.

4 Implementing autopoietic automata in a globular universe

Now we design a specific globular universe and show the implementation of autopoietic automata in it. It is obvious that autopoietic automata cannot exist in a universe which is "too simple". For instance, a universe with only single–state globules does not enable any interaction of globules that would result in their state changes. Similarly, no globular complexes can be built in a universe with only neutral globules (i.e. with those unable entering attraction states). In the sequel a universe in which we will implement an autopoietic automaton will not be explicitly described. Rather, the required properties of the globular universe will be implied by the construction of the self–reproducing globular automaton and it will be clear that such a universe does exist, indeed.

Theorem 1. *There is a nondeterministic globular universe in which, for any autopoietic automaton, there exists its implementation in the form of a self–reproducing globular automaton.*

Sketch of the proof: Let $\mathcal{A}$ be an autopoietic automaton and consider its program tape with the segments of the transition relation of $\mathcal{A}$ written on it. In our universe we will represent this tape as a string of globules. The length of this string equals that of the tape. For simplicity we will first assume that globules in our universe have enough states for directly representing the symbols of a finite subset $S \subset \Sigma$ and $R \subset Q$ really used by the automaton. This means, we assume that there is a one–to–one correspondence between the set of states of globules and S or R, respectively. Moreover, assume that some extra states still remain free. These states will be used to hold "auxiliary variables" in our construction. Thus, in our string each globule will be in a state which uniquely corresponds to the symbol encoded on the automaton's program tape. The globules are designed so that they have four equidistant contact domains — poles — around their equator. On each globule one pair of opposite poles is in a "sticky" state forcing globules to form a sequence. Moreover, we can assume that the first and the last globule will also stick together giving rise to a so–called basic ring representing the transition relation of $\mathcal{A}$. This ring will form the basis of the globular automaton $\mathcal{G}$ implementing $\mathcal{A}$.

Now it is time to describe the input "mechanism" by which $\mathcal{G}$ reads its inputs. We will simply assume that $\mathcal{G}$ "reads" its input by its entire "body". I.e., we will assume that there will come a "wave" of the input globules which will attach themselves (because they are programmed so) to the globules' poles located on the same side of the basic ring. We can assume the existence of such a wave thanks to the properties of the nondeterministic universe. Afterwards, $\mathcal{G}$ starts to work in the translation mode as an interpreter of the code which is represented in the basic ring. In order to work in this way there is an additional ring attached to the basic ring where additional globules representing necessary auxiliary "variables" are kept. The globules in this ring serve e.g. as "markers" denoting the current state or the segment with an instruction to be performed, the program head location, by their interaction “circular” signals are sent around the rings, etc. The resulting globular automaton takes a form of a double ring; its globules mirror the actions of a similar classical cellular automaton. The only difference is that $\mathcal{G}$ is “made of” globules instead of cells, but the neighboring spacial relations among the globules and cells are the same. The details of the construction are given in [6]. The output from $\mathcal{G}$ is done in an analogous way to the input, i.e., the original input globules are “transformed” (via a change of their states) into required output globules and are “released” into the environment (via a change of their attraction properties). In this way the actions of $\mathcal{G}$ proceed until $\mathcal{G}$ enters the first reproducing state.

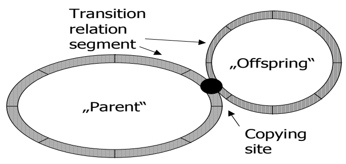

Fig. 2. Self–reproduction of a globular automaton

In the reproducing mode $\mathcal{G}$ slavishly interprets the instructions from its program tape similarly as before. The difference is that now the input is read directly from the globules of the basic ring (for such a purpose the head position must be represented in the auxiliary ring) and the output globules (to which the incoming globules are transformed) are not all released into the environment but those at proper places are subsequently “glued” together to form yet another ring which grows in this manner. In parallel to this ring an auxiliary ring much as the one attached to the basic ring is built. The emerging output and auxiliary rings also form a double ring which touches the original rings at the place where the reading head was initially located. Once $\mathcal{G}$ enters the final reproducing state, the newly generated double ring detaches itself from the original double ring and each complex starts to exist as an independent self–reproducing globular automaton.

The implementation of $\mathcal{A}$ we have just described works in a universe with globules having a sufficient number of states. In a universe with less states we have to work with globules corresponding to the unary coding of states and symbols as required by the definition of autopoietic automata. Thus the simulation process gets more complicated; nevertheless, its main features will remain the same.

$\square$

Even from the previous sketch one can see that in order to realize a self–reproducing globular automaton a nondeterministic universe with a certain minimal number of states is needed. The upper bound on this number could be inferred from a more detailed description of our construction. On the other hand, it is also clear that in universes with too small a number of globular states it might not be principally possible to build a self–reproducing automaton.

5 The computational power of autopoietic automata

In order to get an idea of the computational power of self–reproducing globular automata we will study the power of autopoietic automata. The next theorem will characterize the computational power of the so–called lineages of autopoietic automata with the help of interactive Turing machines. A *lineage of autopoietic automata* is an infinite sequence of autopoietic automata in which each member has exactly one immediate successor which is an immediate offspring of that member. A lineage corresponds to a single path, starting in the root, in a "genealogical" tree consisting of all possible offsprings of a given autopoietic automaton which is located in the root. In this tree, parents are linked to their immediate offsprings. A tree will emerge due to the fact that a single autopoietic automaton can give rise to several offsprings. An *interactive Turing machine* (ITM) is a Turing machine reading a potentially infinite sequence of its inputs via an input port and sending its outputs to the output port [5]. Both in the case of autopoietic automata and ITMs we allow so–called empty inputs that correspond to a situation when no symbol from Σ appears at some port at that time. We say that an ITM *simulates a given autopoietic automaton* (or vice versa) if and only if both devices compute the same translation (mapping) from the input symbol sequence to the output symbol sequence, with empty symbols deleted from both sequences.

Theorem 2. *The computational power of a lineage of (nondeterministic) autopoietic automata equals to that of a nondeterministic interactive Turing machine.*

Sketch of the proof: The proof of the left–to–right implication is relatively simple. The simulation of a given member of a lineage is carried out by the universal ITM which, on its first working tape, has a representation of automaton's program tape and interprets the instructions from this tape. In the translation mode the machine reads its inputs from the input port and sends the outputs to its output port. In the reproducing mode the machine reads its first tape and writes the output to the second working tape. After reaching the automaton's final reproducing state the machine changes the role of its tapes, empties the second tape and the simulation of the next member of a lineage can resume.

The reverse simulation is more complicated. The idea is to see the ITM's computations performed in the space of size i as those of the finite state automaton A_i, for $i = 1, 2, \ldots.$ The automata A_i are realized by corresponding autopoietic automata. What must be designed is the reproducing instructions for autopoietic automata which, as one can see, are the same for all automata. Their task is to "compute" the transition relation for A_{i+1} given the transition relation of A_i. Then, along with the growing space complexity of the Turing machine increasingly bigger autopoietic automata are generated giving the members of the required lineage we are after. For more details see the original report [7].

□

It is obvious that the halting problem for an ITM is undecidable. Thus, it is undecidable whether, given an infinite input sequence and an ITM, the machine will ever halt on that input. A similar question for a given autopoietic automaton and a given infinite input sequence would be the following *problem of a sustainable evolution*: is it decidable whether the automaton will generate an infinite lineage of its offsprings on that input? Referring to the previous theorem the following result holds:

Corollary 1. *The problem of a sustainable evolution is undecidable.*

6 Unbounded evolutionary complexity growth

The previous corollary shows that for a given input sequence we cannot in general decide whether an autopoietic automaton will generate an infinite lineage. It is trivial to see that the situation changes dramatically if we submit to the automata suitable inputs that will cause their entering final reproducing states. To see this it is enough to consider an automaton which replicates on some input and to submit the same input to that offspring which equals its parent. But now we present a much less obvious result — we show an unbounded complexity growth of automata during an evolution by constructing an automaton which in a suitable nondeterministic universe generates all possible autopoietic automata.

Theorem 3. *There exists a nondeterministic autopoietic automaton which, in a suitable nondeterministic universe, generates all existing autopoietic automata.*

Sketch of the proof: We design an automaton whose code contains only the reproducing instructions. These instructions read the segments from the automaton's program tape, modify them and rewrite them onto the output tape. The modifications do not concern the syntax of the segments, i.e., the separators (symbols 1) between the sectors as well as the number of sectors remain unchanged. Then the modifications are threefold:

– A change within a sector: when copying a sector the automaton adds or omits one zero; it follows that the successor automaton will work with other symbols or states than its parent;
– Adding one segment whose contents is nondeterministically generated; this will potentially give rise to a "more complex" automaton;
– Omitting a segment (the automata get "simplified").

It is clear that the initial automaton will subsequently generate autopoietic automata with all possible transition relations. Those of these automata which could in principle reach the final reproducing state on some input will indeed get such an input and hence will self–reproduce. The other automata will not reproduce. Again, for the details see the original report [7].

□

It is important to realize that the construction just described enables an evolution of the mechanism which is responsible for the algorithmic modification of the transition relation. Thus, what we got is a mechanism for the evolution's evolution. Putting it differently, in our latest autopoietic automaton the evolution is not guided by a fixed set of rules; rather, these rules themselves are subjects of an evolution. This is a property not possessed by the automata constructed in the proof of Theorem 2. Last but not least, note that for covering the whole evolutionary space of autopoietic automata it was of fundamental importance that both components of an autopoietic automaton's control — the translating and the reproducing one — were a subject of an evolution.

7 Conclusion

In the paper we surveyed several fundamental results concerning the self–reproducing automata. All these results were based on a new formal framework enabling computational modelling and mathematical analyzing of the respective phenomena. The results show the viability of the approach, bring new insights into the nature and power of evolutionary processes and thus are of interest both from the artificial life as well as from the computational complexity theory point of view.

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. Dorin, A.: Physically Based, Self-Organizing Cellular Automata, in Multi–Agent Systems — Theories, Languages and Applications, Zhang C., Lukose D. (eds), Lecture Notes in Artificial Intelligence 1544, Springer-Verlag, 1998, pp. 74-87.
3. McMullin, B.: John von Neumann and the Evolutionary Growth of Complexity: Looking Backwards, Looking Forwards... Artificial Life, Vol 6. Issue 4, Fall 2000, pp. 347-361
4. von Neumann, J.: Theory of Selfreproducing Automata. A. Burks (Ed.), University of Illinois Press, Urbana and London, 1966
5. van Leeuwen, J., Wiedermann, J.: The Turing machine paradigm in contemporary computing, in: B. Enquist and W. Schmidt (Eds), *Mathematics Unlimited – 2001 and Beyond*, Springer-Verlag, Berlin, 2001, pp. 1139-1155.
6. Wiedermann, J.: Self–reproducing self-assembling evolutionary automata. Proc. of the Workshop on Tilings and Cellular Automata, WTCA'04, CDMTCS Technical Report, University of Auckland, Dec. 2004
7. Wiedermann, J.: Autopoietic automata. Research Report V-929, Institute of Computer Science AS CR, Prague, February 2005, cf. http:=//www.cs.cas.cz