

Coupling computational and non-computational processes: minimal artificial life*

(Extended abstract)

Jiří WIEDERMANN

Institute of Computer Science

Academy of Sciences of the Czech Republic

Pod Vodárenskou věží 2, 182 07 Prague 8, Czech Republic

E-mail jiri.wiedermann@cs.cas.cz

Abstract

We propose a formal abstract hybrid system, so-called bacteroid that combines computational and non-computational mechanisms in its activity. We show that in the environment underlying certain physical laws and consisting of artificial molecules endowed with a certain self-assembly properties there exist bacteroids hallmarking minimal life: they are autonomous, replicate and are subject to darwinian evolution. More complex models equipped with sensor and motor organs exhibit preferential behavior and chemotaxis. The design of bacteroids is inspired by ideas of contemporary molecular biology on no longer existing (or perhaps so-far undiscovered) forms of protolife and from the computational viewpoint it presents a variation on the theme of membrane computing applied to cognitive multiset processing within artificial life.

1 Introduction

Is there a computational definition of life? Can we describe all aspects of living systems purely in computational terms? What is the difference between animated and unanimated matter? These and similar questions are traditionally also dealt with in the disciplines of artificial intelligence (AI) and artificial life (AL) (cf. [3], [4]). Despite numerous efforts no sufficient insight into the mechanisms of life has been achieved. In these efforts computer science is playing an increasingly important role. The history of computer science abounds with computational models that, being inspired by abilities of real biological systems, can be seen as highly oversimplified mathematical models of living systems. From this viewpoint the respective models usually capture but some aspects of AL. Such an aspect could be, e.g., the information processing ability (in the sense of Turing-computability) and the degree of such ability. Recently, a lot of attraction was gained by membrane computing (cf. [11]) which is clearly inspired by cellular biology and rewriting systems. Other model, self-assembly systems (cf. [1], [18]), motivated by self-assembly abilities of

*This research was partially supported by GA ČR grant No. 201/02/1456.

biomolecules present the recent hit. Typical for all these models is their declination from their original biological inspiration when it comes to evaluation of their meaning within computer science. Membrane systems, especially the earlier approaches, can serve as typical examples: in [10] it is stressed that their goal lies not in cellular life modelling, but rather in studying the computational power of models inspired by cellular biology. It is true that in these days the area is much concerned with applications in biology trying to cover as much as possible from the real cell [11]. On the other hand, although the universal computational potential of self-assembly systems was recognized a long time ago, nowadays the use of the respective processes is seen in the (nano)technology (cf. [18]). Of course, by this the respective models place themselves out of the scope of AL since e.g. the computational universality (in the sense of Turing) which seems to be the Holy Grail of these and similar models, is not a fundamental property of the living systems. Contrary to that, the ability of self-reproduction and darwinian evolution — the basic elements of life — are not captured by these models.

In the sequel we will focus onto those computational aspects of life that have been omitted in the above mentioned models. We propose a model that is designed so as to satisfy a minimal set of conditions needed for life to exist. It should be stressed that our goal is not modelling a real life — nevertheless we will be inspired by recent ideas and knowledge about mechanisms of real life. Henceforth we firmly stay on the platform of artificial life. Currently, there is no generally accepted definition of life (cf. [15]) that would capture the full richness of life. In contrast to that, the case of minimal life seems to be better understood. Here, the prevailing and generally agreed-on operational definition could go as follows [16]: a (cellular) system is alive if it is both autonomously replicating and subject to darwinian evolution. Autonomous replication is understood as continued growth and division which is reliant on the input of small molecules and energy only, and does not depend on the products of preexisting living systems. Darwinian evolution requires essential biological aspects of genetic variation and its phenotypic expression as variation in survival and reproduction.

We will call our model of minimal living system a bacteroid. It is an abstract model that is inspired by our ideas on the emergence of the first protocells (cf. [16]). The bacteroid is a hybrid system combining two components in its design. The first one corresponds to so-called epistemic work of the bacteroid at hand. That is, this component deals with bacteroid's information gleaning, processing and exploiting. The other component takes care of the bacteroid's "body" growth, maintenance, and multiplication, i.e., ontic work of a bacteroid [7]. The epistemic work amounts to the computational mechanism of a bacteroid while the ontic work amounts to the non-computational one. The computational mechanism is modelled by a so-called multitransducer with a reconfigurable set of initial states which by itself is a novel computational mechanism. It corresponds to a genom and controls all internal and external activities of a bacteroid as well as its phenotypic expression. The body of a bacteroid is created by a membrane that encompasses the multitransducer and supports bacteroid's sensors and motor units (if any). The growth and fission of a bacteroid is governed by self-assembly and other processes obeying physical laws. Neither the self-assembly processes nor the physical laws are under the computa-

tional control. In a sense the respective non-computational mechanism evokes an oracle that is used by a Turing machine in order to obtain the results that principally cannot be obtained by a machine itself. The properties of both the bacteroidal body and its behavior depend on the multitransducer's work; thus, a survival of a bacteroid and its evolution depends on the mutual cooperation and evolution of its epistemic and ontic components.

When compared with the existing models consisting of a computational mechanism encapsulated in a membrane, such as P-systems, the bacteroid differs in several issues.

- First, it is designed to solve a different kind of problems than most P-systems are: these are problems of survival, reproduction, and darwinian evolution;
- its computational mechanism is based on the formalism of finite automata rather than on the rewriting systems¹;
- the time to realize transitions in finite automata can vary;
- the automata can switch their initial states on and off;
- the automata controlling the bacteroid's work in an asynchronous manner and this is essential for bacteroid's work;
- a bacteroid produces material for its "body" (especially its membrane) construction and for the construction of its genetic information;
- this information becomes a subject of the bacteroid's own information processing;
- the bacteroid's membrane and its sensor and motor organs are constructed by self-assembly processes that are not under a computational control;
- last but not least, in the bacteroid's activity, development and reproduction we count on the effects of physical laws.

The main results of the paper can be summarized as follows. First, a proof that in a universe allowing existence of suitable objects with self-assembly properties and obeying certain physical laws, bacteroids fulfilling all properties of minimal life can be designed. Second, a more evolved variant of a bacteroid is shown to possess a rudimentary cognitive ability: it can prefer a certain type of nutrition. Third, a motile bacteroid is constructed so that it features a variant of so-called chemotaxis meaning that it actively seeks to return into areas abounding with nutrition. Finally, it is pointed out that the lineages of bacteroids exhibit a super-Turing computational power.

The paper consists of four main sections. The first section, the introduction, gives a brief overview of the history of the subject from the viewpoint of computational models and summarizes the main results of the paper. The second section

¹In the context of the formal languages theory the idea of multiset processing by finite transducers occurred already in [12].

introduces a model of a bacteroid. Its computational part is represented by a multitransducer whose definition is given in the first subsection. The embodiment of the multitransducer into the membrane and the mechanisms of membrane growth and division are described in the second subsection. This section is closed by a short discussion concerning the realness of the proposed model. The third section gives a brief overview of the main results. First, a protobacteroid is defined fulfilling the conditions of minimal life. Second, it is shown that a so-called selective bacteroid exhibits certain primitive cognitive abilities by distinguishing between two kinds of nutrition. The next result deals with a motile bacteroid that actively returns to areas with plenty of nutrition. The last result concerns the super-Turing ability of lineages of bacteroids. The fourth section contains the conclusion summarizing the main achievements and gives ideas for further research.

The paper describes the research under development, and so far neither the model nor the formalism and nor the terminology are definitive; similarly, the results and their interpretation only start to emerge. In addition to the page limitations this is the main reason why the paper has the form of an extended abstract.

2 Basic model: the bacteroid

Interactive finite multitransducer First, we will concentrate on the epistemic 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 the automata processing multisets of 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 objects that are available for processing at each time — see the description of multitransducer’s activity in the sequel. Thus, from the computational viewpoint a multitransducer is a highly parallel information processing device.

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

Definition 2.1 *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 $I \times S \rightarrow S \times O \times S \times \{0, 1\} \times \mathbf{N}$ 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 2.1. 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 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 it 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 the 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.

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 a computation. Rather, depending on the course of computation this set can change over time: some states can lose 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 as a part of the multitransducer’s definition. 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 intended use of this mechanism is to model the gene switching in real cells. The second point of 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 will be used in tuning the synchronization among various automata².

It seems natural to require 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 in its left hand side the processing will be prolonged by whatever transition that applies to the new state and the symbol read at that very moment. In this way the processing prolongs 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, then a new processing can be launched. In what follows instead of the term “string” we will often use the term “object” to denote either a symbol or a string of symbols.

So far we have not counted on evolution of a multitransducer — a change of its activity can only be made by changing its transition function. It is obvious that at the very beginning of a computation the activity of individual states is given by set B . Depending on the inputs read in subsequent steps and their order, (recall that automata work asynchronously) the activity of states can change.

Embodying the multitransducer In our intended application in minimal life modelling we will have to consider different multitransducers (or even off-springs of a transducer) as agents acting in the same environment. So far this has not been possible since all of them would behave as a single multiset of automata controlled by the union of all transition functions of all multitransducers. Thus we need a way to separate individual transducers from each other. For such a purpose (but also for other purposes — cf. [13] for the current views on embodied cognition) we will endow each multitransducer with a “body” .

A membrane will represent the simplest model of an agent’s body. Its purpose will be to protect the control and information mechanisms of an agent from the environmental influence, to provide a support for agent’s perception–motor units, and, last but not least, to enable the agent’s development (especially its growth and multiplication). The membrane will be modelled as a three–dimensional spherical structure consisting of special objects called *tiles*. A membrane is constructed from tiles “all by itself”, by self–assembly under certain physio–chemical circumstances. That is, under reasonable circumstance 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 become spontaneously incorporated into it. In this way a membrane can grow. When roughly doubling its size a membrane can tear in two approximately equal

²The speed parameter can be avoided at the expense of allowing epsilon transitions in the formal definition of a multitransducer.

parts that both spontaneously again organize into membranes. Henceforth, both properties of the membrane growing and splitting are a consequence of physical (and hence chemical) laws acting in the given environment. The pace of membrane growth depends, among other things, on the supply of tiles which are generated, as we will see, by automata from elements that are not endowed by self-assembly property. Perception and motor units can also find themselves in a membrane. Their presence is controlled by automata and their number also depends on the supply of the respective material.

The activities of a multiset of automata comprising a multitransducer 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 $I \times S \rightarrow S \times O \times S \times \{0, 1\} \times \mathbf{N}$. Of course, such a program resembles a genome residing inside a cell. Similarly, as a genome it consists of instruction for production of various objects that can further be used for membrane, receptor or motor units construction. The program also contains instructions for initiating and terminating an object or unit construction, or for their de-assembly. 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 one by which it accepts “stuff” (objects) from which a tape’s copy is to be constructed.³ Of course, the copying process does not destroy the original tape.

In its activity, especially in the membrane growth and splitting, and in organ development, a multitransducer counts on the self-assembly ability of some objects and on the functioning of physical laws as we have seen it in the case of tiles. In greater detail we will describe the corresponding mechanisms in the sequel.

A *bacteroid* is a generic name for a system consisting of a multitransducer and a membrane surrounding it. Similarly as in the case of real bacteria also bacteroids differ in their genotype (that, roughly, corresponds to the way a multitransducer controls a bacteroid in) and phenotype (that corresponds to the “physical” equipment of a bacteroid — what its membrane is like, what are its organs, where they are located, what are their properties, etc.). The simplest of all bacteroids is a so-called *protobacteroid* possessing no sensory or motor organs and consisting of only a membrane encompassing a program tape. The reason for the existence of such an entity is this: inside the membrane from the objects floating in the environment and freely

³When it comes to modelling of self-reproduction of living organisms some authors seem to claim that in addition to a genetic information copying mechanism, a genetic description of an organism must also include a description of a self-replication mechanism (this is the essence of the well-known von Neumann’s result on self-reproducing automata). In our setting this would mean that a part of the multitransducer’s description should be devoted to the “recipe” how to build 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, from the evolutionary point of view this seems to be unnecessarily complicated and therefore less probable.

permeating this membrane, a protobacteroid's multitransducer synthesizes the tiles that are incorporated, by self-assembly, into an already existing membrane. The respective synthesizing automata will be called *tile automata*. In parallel with this process a replication of the bacteroid's tape is in progress. This process is accomplished by a so-called *copying automaton*. The new tape is also constructed from objects that permeate the membrane.

Now we describe the mechanism of the bacteroid's fission. Assume that after the copying operation is finished both the new and original tape are rolled into separate balls. Further assume that the membrane encompassing both balls is not stretched, it is rather larger than necessary in order to tightly encompass the balls. That means that the resulting shape of the protobacteroid in this phase starts to resemble a sort of a three-dimensional eight and at the moment, when the interior of the membrane between the two balls has a sufficiently small diameter, the self-assembly mechanism of tiles will split the membrane into two parts, each containing one copy of the program tape rolled into a ball. Hence, two protobacteroids emerge. In order to make this work properly there must be a proper synchronization between the copying process and that of the membrane growth. This timing is achieved via the proper settings of the speed parameters in transitions of the respective automata; the corresponding parameter optimization is again a result of an evolution.

Note that in contrast to the P-systems a bacteroid is defined in a way similar to real bacteria, with the help of a single membrane. In P-systems there is the idea of localization by means of embedded membranes that protect the encapsulated objects from "reactions" from outside. In our setting a similar effect is achieved by automata that do not "drop" an object until its processing is finished. Moreover, they can "keep" an object as long as necessary by adjusting the speed of its production. The additional flexibility is offered by initial state switching, that is, we can switch off certain automata to prevent them in processing some objects. Of course, the latter mentioned mechanism is crucial for phenotypic differentiation at the level of either unicellular or multicellular organisms.

Even from the above informal description one can see that for its existence a protobacteroid needs a hierarchy of objects with various properties. We start with simple 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 (e.g. tiles). These self-assembly objects further self-organize into complex structures (such as membranes) that, obeying the physical laws, are endowed with still other emergent properties not possessed by their parts (e.g., membrane splitting). Moreover, the physical laws should work "as needed" for a bacteroid to operate correctly. The self-assembly property similarly as the physical laws are "present" all the time without a need to be invoked; what is done in a bacteroid is harnessing these non-computational phenomena for the purpose of artificial life as we shall see in Section 3.

Bacteroids and real bacteria A few words concerning the realness of the proposed computational and non-computational mechanism of bacteroids are in place. In fact, the computational mechanisms are a radical simplification of so-called central dogma of molecular biology (cf. [5]) that states that each gene in the DNA molecule (genom)

carries the information needed to construct one protein, which, acting as an enzyme, controls one chemical reaction in the cell. In our model one transition instruction corresponds to a gene and the automaton realizing that transition corresponds to that protein. The objects correspond to chemical compounds and computations over objects correspond to chemical reactions. That means that the objects can be seen as physical data representation, they are their “embodiment”. Both computations and self-assembly correspond to the construction of more complex objects. In real life the genom is wound up onto a group of proteins called histones. In real cells gene switching is probably carried out by chemical switches (cf. operon hypothesis) that “cover” the required gene in the genom and thus the gene cannot operate in the same way as before. The communication within a cell works with the help of chemical or other signals along so-called signalling pathways. Self-assembly of certain types of molecules (e.g. of amphiphilic molecules into three-dimensional vesicles) is an experimentally verified fact (cf. [6]). The translocation of input objects via the membrane, or via specialized pores or channels, respectively (see the case of selective bacteroids in the sequel), also corresponds to reality where it is a consequence of the Brownian motion and physio-chemical properties of the participating molecules (both in the membrane and those translocating it). For more information about the self-organization and its molecular realization see the survey paper [18]. The theory of self-assembly is in its beginning (cf. [1]). So far, mostly square- or cube-shaped objects have been considered from which it is not possible to construct all organs found in real bacteria. Nevertheless, even such results point to the fact that in principle the self-assembly is possible and the resulting shape of the assembled objects depends on the shape and properties of the parts from which the complex is built. Last but not least, these results clearly demonstrate that mathematic modelling of self-assembly is possible. Along these lines, a formal definition of the shape and properties of self-assembly objects, in a way similar to that used e.g. in [14], could be introduced into the definition of a bacteroid.

To get an idea about the complexity of real bacteria we mention the following figures. The number of genes in real bacteria starts with approximately 300 in the case of the simplest bacteria and for other bacteria averages around 2000 genes. The notoriously known bacteria *Escherichia Coli* has 4288 genes and utilizes about 20 signal pathways for information transfers inside a cell. It can choose from among approximately 30 types of chemoreceptors; all of them need not be always present but can be “synthesized” on demand. At each time in average there are about 1500 of chemoreceptors of 5 different types expressed in *E. Coli*'s membrane.

3 Main results

Protobacteroids and minimal life The minimal life definition includes the idea of the organism that fulfills the minimal requirements for it to be considered alive. An entity satisfying less requirements cannot be seen as a living system. It seems that the currently favored operational definition asks for meeting two conditions [16]: a minimal life system must be both autonomously replicating and subject to darwinian evolution. Autonomous replication means a continuous growth and division which

is reliant on the input of simple input objects only, and does not depend on the product of other living systems. Darwinian evolution requires variations in survival and reproduction resulting from the genetic variation and its phenotypic expression.

As far as our protobacteroid is concerned, from its very design it is easily seen that the first condition of the above mentioned definition is satisfied. Under a sufficient supply of input objects the protobacteroid survives and multiplies; in this process we assumed that the input objects are “simpler” than the derived ones since the latter are “computed” from the former ones. The darwinian evolution is ensured by insisting on the copying mechanism being unreliable. This means that it is likely that the copying automaton makes an error when copying the tape — not only “typos”, but perhaps such that some program segments get not copied at all while some others may get duplicated. In this way some mutations may immediately lead to a loss of functionality of a protobacteroid while others may open, perhaps in a long run, a way to the increased rate of survivability. It is interesting to note that the copying automaton (that is, the “mutation” mechanism it implements) itself can become a subject of a mutation. In this way various innovation strategies can be realized. Consequently, we see that the protobacteroids fulfill the definition of a minimal artificial life, indeed.

For our system of minimal life to work it has been essential that the “future” of a bacteroid depends on the proper interplay between its epistemic and ontic parts that in turn depends on the proper interaction with the world. Without explicitly introducing a membrane and its structure and, as we shall see later, a possibility of its further development (by accommodating sensor and motor organs in it), we cannot make the multitransducer’s evolution dependent on the functioning of its “body”. Thus, unlike as in the case of P-systems where the membrane appearance is a matter of one instruction, in bacteroids membrane growth and development is a process dependent on information processing abilities (and vice versa) and consuming a substantial amount of information processing power (see the case of selective bacteroids in the sequel). In fact the mutual dependence among the information processing, body control and body evolution, “powered by” physical laws, has been the main animation trick for a bacteroid to become “alive” in our model.

Selective behavior In order to exist and survive, protobacteroids did not make any use of the information on the environment. They were not actively selective — they tried to “consume” whatever had permeated their membranes. We show that so-called *selective bacteroids* that take into account information from the environment get an evolutionary advantage.

Consider a bacteroid \mathcal{S} in an environment consisting of two types of nutrients that are modelled by objects of type A and B , respectively. \mathcal{S} can consume both types of nutrients, but a nutrient of type A is more nutritious than type B — e.g., from type A bacteroid \mathcal{S} can produce substantially more tiles than from type B nutrient. The protobacteroid from the previous subsection — let us call it \mathcal{P} — can exist in such an environment without any problems, since it can “eat” both types of food substances. Nevertheless, in an environment that abounds in type A nutrients, a bacteroid preferring this type could be evolutionary privileged in a long run. Note that the theory of formal languages states that there is no finite state

automaton accepting strings in which *as* prevail over *bs*. Thus, our problem cannot be solved at the level of individual automata: some cooperation among them would be necessary. In fact, the solution we will propose is an application of a random sampling technique.

There is no problem in constructing a Mealy automaton \mathcal{A} accepting solely *as* and \mathcal{B} accepting both *as* and *bs* (i.e., either *a* or *b*). Both types of automata produce at their outputs the same elements they accepted. Call such automata (translocation) *channels*, or *pores* and place them into the membrane of \mathcal{S} . Thus, the pores let selectively pass the objects from the environment into the membrane. Obviously, \mathcal{A} can be active all the time (it makes no sense to reject any *as*). What makes sense is to switch off \mathcal{B} in the case when there is plenty of *as* in the environment. In order to do so let us equip \mathcal{S} with the following additional organs: assume that in the membrane of \mathcal{S} there are *receptors* reacting to the “concentration” of *as* in the mixture of nutrients of both types in the environment of the bacteroid at hand. Each receptor is a special automaton \mathcal{R} that in parallel scans a fixed number of “random” objects (nutrients) appearing at its input ports. For instance, if we want to detect that with a high probability the concentration of *as* is not under 50%, then it is enough for each receptor to scan just two objects to see whether at least one of them is of type *A*. If and only if this is the case the receptor deactivates channels \mathcal{B} (by deactivating their initial states) preventing thus *bs* entering the membrane. Afterwards, \mathcal{R} leaves free the objects it scanned without changing them (or \mathcal{R} “consumes” the objects it scanned and via its output it produces them again into the environment). Thus, the “program” controlling \mathcal{S} might contain the following segment:

- \mathcal{A} is always active allowing translocation of *as* into the membrane;
- upon detecting at least one *a*, \mathcal{R} closes pores \mathcal{B} s; otherwise \mathcal{R} opens \mathcal{B} s.

In order to see that the above described mechanism works as we intended it is important to realize that each receptor opens/closes all channels of the respective type and that all receptors work asynchronously. That is, as long as there is a majority of *as* in the immediate environment of \mathcal{S} , the majority of receptors send the signal for closing pores \mathcal{B} most of the time and therefore \mathcal{S} consumes type *A* nutrients most of the time. Otherwise, the pores \mathcal{B} are kept open most of the time and \mathcal{S} accepts nutrients of both types. Note that in this way a selective bacteroid can switch between the two modes of behavior, but in the long run the more efficient behavior keeps prevailing. The switching moments depend on the construction of the receptors (to what concentration of *as* they react) and on other parameters (e.g. on the time needed to open or close the pores) that are difficult to analyze; nevertheless, the task of switching mechanism optimization is taken over by evolution, by trial and error process. Similarly, the “upgrade” from \mathcal{P} to \mathcal{S} is also a result of evolution. The new organs — pores and receptors — are built by self-assembly processes. The task of the evolution is to “discover” the availability and usefulness of such an upgrade. Note that the behavior just sketched bears the hallmarks of a cognitive behavior — the selective bacteroid discriminates between “good” and “bad”.

Chemotaxis Motile real bacteria (such as *Escherichia Coli*) also display another type of “intelligent” behavior — they keep swimming roughly in the direction of

the largest nutrient gradient. This is called *chemotaxis*. Such behavior can also be modelled within our framework. First of all, we will need a further evolutionary innovation — *flagellar motors* (cf. [9]). In reality, these are tiny (nanoscale) motors powered by ions that rotate the flagella sticking out from the membrane. Rotating the flagella in one direction causes a move of a bacteria along a direct line. Rotating the flagella in an opposite direction causes so-called “tumbling” of a bacteria — it turns randomly in a three-dimensional space so that its next move will proceed along a new, randomly chosen direction. Now assume that a bacteroid, \mathcal{M} , is an upgraded version of a selective bacteroid equipped by flagellar motors (emerging by evolution and constructed by self-assembly). As before, let type A nutrient be “much better” than type B nutrient and assume that the amount of a nutrient consumed depends on the distance the bacteroid has travelled while consuming. Consider the behavior of \mathcal{M} according to the following program segment:

- L: **while** nutrient is OK **do** keep eating and running in the same direction **od**;
- otherwise make a stop, do not eat and rotate randomly;
- then make a short trip in the new direction;
- **go to** L

Clearly, realization of the above described behavior can be seen as a simple type of chemotaxis: when running out of an area with type A nutrient \mathcal{M} tries to return to that area making short trips in random directions. The duration of a short trip is controlled by a special timer that makes use of a speed parameter in its transitions (see the definition of transition function from Definition 2.1).

Computational power It was stressed in the Introduction that achieving the Turing universality had not been the main goal of our design. Keeping the descriptive complexity of our model in a finite range without allowing a use of any additional memory cannot give rise to a device with a universal computing power. Instead of isolated bacteroids we can consider evolutionary sequences of bacteroids in which the successor of each bacteroid is its offspring emerging by a random genetic mutation and surviving under unpredictable conditions (whose description is not a part of the sequence model). Since there is always a subset of states being “transferred” to the next generation, such a sequence of bacteroids corresponds to a so-called *lineage of automata* as introduced in [17]. A lineage of bacteroids can start with a protobacteroid and its subsequent members will correspond to a single random path in its bacteroid’s evolutionary tree. Obviously, there is no way to algorithmically specify such a lineage. Considering computations of lineages of bacteroids leads to so-called *non-terminating interactive evolutionary computations* whose power gets beyond that of the classical Turing machine. This has been proved in a series of recent papers by van Leeuwen and Wiedermann, most notably in [17]. It is not clear whether the notion of a lineage can naturally be modelled by P-systems.

4 Conclusion

We have proposed an abstract embodied hybrid system — a bacteroid — that in its activity couples a computational mechanism with that of self-organization. In the field of AL this seems to represent the first system provably exhibiting minimal life within an artificial universe ruled by laws allowing existence of suitable self-assembly objects. From the computational point of view a bacteroid can be seen as a multiset of communicating asynchronous automata performing a specific sort of real-time multiset cognitive processing that leads to a purposeful behavior (i.e., autonomous growth and replication, and evolution) in the given environment. The computing power of a bacteroid is primarily used to control its body and its organs which are all seen as a result of non-computationally controlled self-assembly processes. In the bacteroid's activity the stress is put on cognitive task solution by relying on a proper interactive coupling of sensory and motor actions. The reproduction involves a learning aspect due to the nature of darwinian evolution. The context of minimal life leads to a consideration of entirely new types of problems whose formulation, in terms of classical finite state automata framework, has not been possible. In addition to the problem of nutrition selectivity and chemotaxis considered here there is a lot of other problems inspired by real bacteria (cf. [8]) that seem to be amenable to a modelling in a similar spirit as sketched in this paper. These problems and extension of our modelling to the case of multicellular organisms represent a possible avenue for further research. Such efforts can shed new light on the computational complexity related to construction and epistemic work of minimal life systems and consequently to the synthesis *de novo* of real artificial organisms (cf. [2], [6], [16]).

Acknowledgement Thanks go to G. Paun for his comments on the earlier version of the paper.

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] Benenson, Y., Paz-Elizur, T., Adar, R., Keinan, E., Livneh, Z., Shapiro, E.: Programmable and autonomous computing machines made of biomolecules. *Nature* 414 (2001), 430-434.
- [3] Brooks, R.: The relationship between matter and life. *Nature*, Vol. 409, 18 January 2001, pp. 409–411
- [4] Brooks, R.: Beyond Computation. *EDGE* 132, June (2002) (http://www.edge.org/3rd_culture/brooks_beyond/beyond_index.html).
- [5] Central dogma of molecular biology:
cf. <http://www.bartleby.com/59/21/centraldogma.html>.
- [6] Hanczyc, M. M., Fukijawa, S. M., Szostak, J. W.: Experimental Models of Primitive Cellular Compartments: Encapsulation, Growth, and Division. *Science*. 302 (2003), 618-622.

- [7] Kováč, L., Nosek, J., Tomáška, L.: An Overlooked Riddle of Lifes Origin: Energy Dependent Nucleic Acid Unzipping. *J. Mol. Evol.* Vol. 57 Suppl 1:S182-9, 2003.
- [8] Lengeler, J. W., Müller, B. S., di Primio, F.: Neubewertung kognitiver Leistungen im Lichte der Fähigkeiten einzelliger Lebewesen. *Kognitionswissenschaft* 8 (2000) 160-178.
- [9] Musgrave, I.: Evolution of the Bacterial Flagellum. In: *Why Intelligent Design Fails: A Scientific Critique of the Neocreationism*. Young, M., and Edis, T. (Eds.), forthcoming from Rutgers University Press, Piscataway, N.J. 2004.
- [10] Paun, G., Rozenberg, G.: A Guide to Membrane Computing. *Theoretical Computer Science* 287 (2002), 73-100.
- [11] Paun, G.: *Membrane Computing. An Introduction*, Springer, 2002
- [12] Paun, G., Thierrin, G.: Multiset processing by means of systems of sequential transducers, In: *Proc. Workshop on Implementing Automata WIA99*, Potsdam, August 1999, LNCS 2214, Springer 2001, 140-157
- [13] Pfeifer, R., Scheier, C.: *Understanding Intelligence*. The MIT Press, Cambridge, 1999, 697 s.
- [14] Rothmund, P, Winfree, E.: The program-size complexity of self-assembled squares (extended abstract). In *Proceedings of the thirty-second annual ACM symposium on Theory of computing*, pages 459-468. ACM Press, 2000.
- [15] Shenhav, B., Lancet, D.: Prospects of a computational origin of life endeavor. *Origins of Life and Evolution in the Biosphere*, Kluwer Academic Publishers, Vol. 34, pp. 181–194, 2004
- [16] Szostak, J.W., Bartel, D.P., Luisi, P.L.: Synthesizing Life. *Nature* 409 (2001) 389-390.
- [17] Wiedermann, J., van Leeuwen, J.: The Emergent Computational Potential of Evolving Artificial Living Systems. *Ai Communications* 15, 4 (2002), 205-216.
- [18] Winfree, E.: DNA Computing by Self-Assembly. *National Academy of Engineering Website, The Bridge* 33, 4 (2003), 31-38.