Towards an Object-Oriented Programming Language for Physarum Polycephalum Computing

Andrew Schumann[1] and Krzysztof Pancerz[1,2]

[1] University of Information Technology and Management
Sucharskiego Str. 2, 35-225 Rzeszów, Poland
aschumann@wsiz.rzeszow.pl

[2] University of Management and Administration
Akademicka Str. 4, 22-400 Zamość, Poland
kpancerz@wszia.edu.pl

Abstract. In the paper, we present foundations of a new object-oriented programming language for Physarum polycephalum computing. Both, theoretical foundations and assumptions for a language specification are considered. Physarum polycephalum is a one-cell organism. In the phase of plasmodium, its behavior can be regarded as a biological substrate that implements the Kolmogorov-Uspensky machine which is the most generalized and nature-oriented version of a mathematical machine. The proposed language will be used for developing programs for Physarum polycephalum by the spatial configuration of stationary nodes (inputs).

Keywords: Physarum polycephalum, unconventional computing, nature-inspired computing, object-oriented programming language, Kolmogorov-Uspensky machine

1 Introduction

Physarum polycephalum is a one-cell organism belonging to Physarales, subclass Myxogastromycetidae, class Myxomycetes and division Myxostelida. In the phase of plasmodium, it looks like an amorphous giant amoeba with networks of protoplasmic tubes. It feeds on bacteria, spores and other microbial creatures (substances with potentially high nutritional value) by propagating towards sources of food particles and occupying these sources. A network of protoplasmic tubes connects the masses of protoplasm. As a result, the plasmodium develops a planar graph, where the food sources or pheromones are considered as nodes and protoplasmic tubes as edges. This fact allows us to claim that plasmodium behavior can be regarded as a biological implementation of Kolmogorov-Uspensky machines [7]. The modification of locations of nutrients (food sources) causes a storage modification of plasmodium. Hence, the plasmodium may be used for developing a biological architecture of different abstract automata such as Kolmogorov-Uspensky machines [16, 22], Tarjan's reference machine [21], and

Schönhage's storage modification machines [19, 20]. In *Physarum Chip Project: Growing Computers From Slime Mould* [11] supported by FP7 we are going to implement programmable amorphous biological computers in plasmodium of Physarum. This abstract computer is called *slime mould based computer*.

One of the paths of our research in this area concerns creating a new programming language that simulates plasmodium behavior. The following main tasks can be distinguished in the first step of this path:

1. Constructing the programming language on the basis of storage machines. The static storage structure is represented by a two-dimensional configuration of point-wise sources of chemo-attractants and chemo-repellents.
2. Constructing the programming language on the basis of the Kolmogorov-Uspensky machine (KUM), where edges are represented by protoplasmic strands.
3. Developing programs represented by the spatial configuration of stationary nodes (treated as inputs of the programs). Outputs of the programs may be recorded optically.

The rest of the paper is organized as follows. In Section 3, we give foundations of specification of a new language. Assumptions of specification are preceded by a theoretical background of Physarum automata (see Section 2).

2 Physarum Automata

Plasmodium's active zones of growing pseudopodia interact concurrently and in a parallel manner. At these active zones, three basic operations stimulated by nutrients and some other conditions can be observed: fusion, multiplication, and direction operations. The *fusion* $Fuse$ means that two active zones A_1 and A_2 both produce new active zone A_3 (i.e. there is a collision of the active zones). The *multiplication* $Mult$ means that the active zone A_1 splits into two independent active zones A_2 and A_3 propagating along their own trajectories. The *direction* $Direct$ means that the active zone A is not translated to a source of nutrients but to a domain of an active space with a certain initial velocity vector v. These operations, $Fuse$, $Mult$, $Direct$, can be determined by the following stimuli:

- The set of attractants $\{N_1, N_2, \ldots\}$. Attractants are sources of nutrients or pheromones, on which the plasmodium feeds. Each attractant N is characterized by its position and intensity. It is a function from one active zone to another.
- The set of repellents $\{R_1, R_2, \ldots\}$. Plasmodium of Physarum avoids light and some thermo- and salt-based conditions. Thus, domains of high illumination (or high grade of salt) are repellents such that each repellent R is characterized by its position and intensity, or force of repelling. In other words, each repellent R is a function from one active zone to another.

Such plasmodium behavior can be presented as an implementation of some abstract automata.

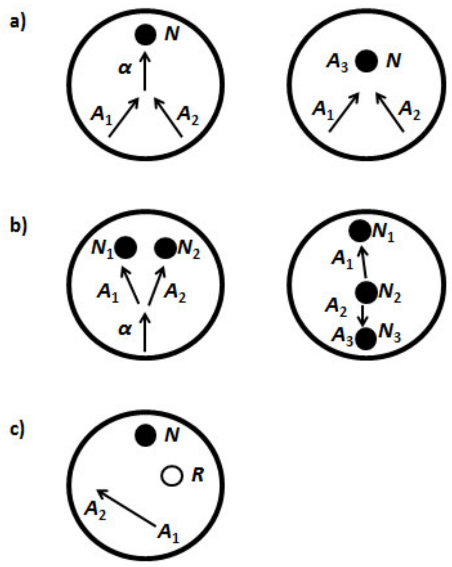

Fig. 1. The stimulation of the following operations in Physarum automata: (a) fusion, (b) multiplication, and (c) direction, where A_1, A_2, A_3 are active zones, N, N_1, N_2, N_3 are attractants, α is a protoplasmic tube, R is a repellent.

2.1 Physarum Cellular Automata

Recall that a cellular automaton is a 4-tuple $\mathcal{A} = \langle \mathbf{Z}^d, S, u, f \rangle$, where (1) $d \in \mathbf{N}$ is a number of dimensions, and the members of $\mathbf{Z}^d$ are referred to as cells, (2) S is a finite set of elements called the states of an automaton $\mathcal{A}$, the members of $\mathbf{Z}^d$ take their values in S, (3) $u \subset \mathbf{Z}^d \setminus \{0\}^d$ is a finite ordered set of n elements, $u(x)$ is said to be a neighborhood for the cell x, (4) $f : S^{n+1} \to S$ that is f is the local transition function (or local rule). As we see an automaton is considered on the endless d-dimensional space of integers, i.e., on $\mathbf{Z}^d$. Discrete time is introduced for $t = 0, 1, 2, \ldots$ For instance, the cell x at time t is denoted by x^t. Each automaton calculates its next state depending on states of its closest neighbors. The cellular automata thus represent locality of physics of information and massive-parallelism in space-time dynamics of natural systems.

In abstract cellular automata, cells are physically identical. They can differ just by one of the possible states of S. In case of Physarum, cells can possess different topological properties. This depends on intensity of chemo-attractants and chemo-repellents. The intensity entails the natural or geographical neighborhood of the set's elements in accordance with the spreading of attractants or repellents. As a result, we obtain Voronoi cells. Let us define what they are mathematically. Let $\mathbf{P}$ be a nonempty finite set of planar points and $|\mathbf{P}| = n$. For points $p = (p_1, p_2)$ and $x = (x_1, x_2)$ let $d(p, x) = \sqrt{(p_1 - x_1)^2 + (p_2 - x_2)^2}$ denote their Euclidean distance. A planar Voronoi diagram of the set $\mathbf{P}$ is a partition of the plane into cells, such that for any element of $\mathbf{P}$, a cell corresponding to a unique point p contains all those points of the plane which are closer to p in respect to the distance d than to any other node of $\mathbf{P}$. A unique region

$$vor(p) = \bigcap_{m \in \mathbf{P}, m \neq p} \{z \in \mathbf{R}^2 \colon d(p,z) < d(m,z)\}$$

assigned to a point p is called a *Voronoi cell* of the point p. Within one Voronoi cell a reagent has a full power to attract or repel the plasmodium. The distance d is defined by the intensity of reagent spreading. A reagent attracts or repels the plasmodium and the distance, on which it is possible, corresponds to the elements of a given planar set $\mathbf{P}$. When two spreading wave fronts of the two reagents meet, this means that on the board of meeting the plasmodium cannot choose its one further direction and splits (see Figure 2).

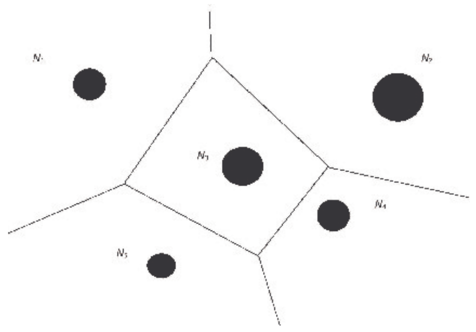

Fig. 2. The Voronoi diagram for Physarum, where different attractants have different intensity and power.

The direction of protoplasmic tubes is defined by concentrations of chemo-attractants or chemo-repellents in Voronoi neighborhood. Each dynamics of protoplasmic tube can be characterized at time step t by its current position x_t and the angle α_t.

2.2 Physarum Kolmogorov-Uspensky Machines

Let Γ be an alphabet, k a natural number. We say that a tree is (Γ, k)-tree, if one of nodes is designated and is called *root* and all edges are directed. Each node is labeled by one of the signs of Γ and each edge from the same node is labeled by different numbers $\{1, \ldots, k\}$ (so, each node has not more than k edges). We see that by this definition of (Γ, k)-tree, the pseudopodia growing from the one active zone, where all attractants are labeled by signs of Γ, and protoplasmic tubes are labeled by numbers of $\{1, \ldots, k\}$, is a (Γ, k)-tree.

Let r be the maximal possible path of $(\Gamma; k)$-tree. We can always design Physarum Voronoi diagrams (using attractants and repellents) for inducing different numbers r and appropriate local properties. The $(\Gamma; k)$-tree limited by

r is called $(\Gamma; k)$-complex. Programming in Kolmogorov-Uspensky machines is considered as transforming one $(\Gamma; k)$-complex to another with the same r by changing nodes and edges using some rules. In case of Physarum implementation of Kolmogorov-Uspensky machines programming is presented as transforming one Voronoi diagram into another with the same r by dynamics of Physarum (e.g. when some attractants become eaten by Physarum).

The simpler version of the Kolmogorov-Uspensky machines is presented by Schönhage's storage modification machines.

2.3 Physarum Schönhage's Storage Modification Machines

These machines consist of a fixed alphabet of input symbols, Γ, and a mutable directed graph with its arrows labeled by Γ. The set of nodes X, identified with attractants is finite, as well. One fixed node $a \in X$ is identified as a distinguished center node of the graph. It is the first active zone of growing pseudopodia. The distinguished node a has an edge x such that $x_\gamma(a) = a$ for all $\gamma \in \Gamma$. That is, all pointers from the distinguished center node point back to the center node. Each $\gamma \in \Gamma$ defines a mapping x_γ from X to X. Each word of symbols in the alphabet Γ is a pathway through the machine from the distinguished center node.

Schönhage's machine modifies storage by adding new elements and redirecting edges. Its basic instructions are as follows:

- Creating a new node: **new** W. The machine reads the word W, following the path represented by the symbols of W until the machine comes to the last symbol in the word. It causes a new node y, associated with the last symbol of W, to be created and added to X. Adding a new node means adding a new attractant within a Physarum Voronoi diagram.
- A pointer redirection: **set** W **to** V. This instruction redirects an edge from the path represented by word W to a former node that represents word V. It means that we can remove some attractants within a Physarum Voronoi diagram.
- A conditional instruction: **if** $V = W$ **then instruction** Z. It compares two paths represented by words W and V and if they end at the same node, then we jump to instruction Z, otherwise we continue. This instruction serves to add edges between existing nodes. It corresponds to the splitting or fusion of Physarum.

3 Foundations of Specification of an Object-Oriented Programming Language for Physarum Polycephalum

The plasmodium of Physarum polycephalum functions as a parallel amorphous computer with parallel inputs and parallel outputs. Data are represented by spatial configurations of sources of nutrients. Therefore, we can generally assume that a program of computation is coded via configurations of repellents and attractants. The plasmodium of Physarum polycephalum is a computing substrate. In [10], Adamatzky underlined that Physarum does not compute. It obeys physical, chemical and biological laws. Its behavior can be translated to the language of computations.

In this section, we deal with foundations of specification of a new object-oriented programming language for Physarum polycephalum computing on the basis of using a Voronoi diagram for implementing Kolmogorov-Uspensky machines. In an object-oriented programming (OOP) paradigm, concepts are represented as objects that have data fields (properties describing objects) and associated procedures known as methods. The OOP approach assumes that properties describing objects are not directly accessible by the rest of the program. They are accessed by calling special methods, which are bundled in with the properties. This approach has been implemented in our new language. Moreover, we have referred to conventions used in the JavaBeans API [4], i.e., the object properties must be accessible using *get*, *set*, and *is* (used for Boolean properties instead of *get*). They are called accessor methods. For readable properties, there are getter methods reading the property values. For writable properties, there are setter methods allowing the property values to be set or updated.

Our new language has been proposed as a prototype-based programming language like, for example, Self [1], JavaScript and other ECMAScript implementations [2]. Unlike traditional class-based object-oriented languages, it is based on a style of object-oriented programming in which classes are not present. Behavior reuse is performed via a process of cloning existing objects that serve as prototypes. This model is also known as instance-based programming.

Table 1. Main objects identified in Physarum polycephalum computing

Object	Properties
Layer	*id*, *size*, *elements*
Physarum	*id*, *position*, *intensity*
Attractant	*id*, *position*, *intensity*
Repellent	*id*, *position*, *intensity*

The main objects identified in Physarum polycephalum computing are collected in Table 1. We assume that a computational space is divided into two-dimensional computational layers on which Physarum polycephalum, as well as attractants and repellents, can be scattered. Our approach allows interaction between elements placed on different layers. This property enables us to use, in the future, the multi-agent paradigm in Physarum polycephalum computing. The user can define, in the computational space, as many computational layers as needed. For each layer, its size can be determined individually. We apply the point-wise configuration of elements scattered on the layers. Therefore, for each element (Physarum, attractant, repellent), its position can be determined using two integers (coordinates). As it was mentioned in Section 2, attractants and repellents are characterized by the property called intensity. This property plays an important role in creation of the Voronoi cells. For each attractant and repellent, the intensity is a fuzzy value from the interval $[0, 1]$, where 1 denotes the maximal intensity, while 0 the minimal intensity, i.e., a total lack of impact of a given attractant or repellent on Physarum polycephalum. The force of attracting (repelling) of Physarum is a combination of intensity of attractants (repellents) and distances between plasmodium and attractants (repellents), respectively.

Let $p = (p_1, p_2)$ and $x = (x_1, x_2)$ be points on the layer where Physarum and attractant (repellent), respectively, are located. To create the Voronoi cells, we can use the following measure modyfying a distance, which is commonly used:

$$f(p, x) = \frac{1}{\varepsilon(x)} \sqrt{(p_1 - x_1)^2 + (p_2 - x_2)^2},$$

where $\varepsilon(x)$ is the intensity of attractant (repellent) placed at x. It means that the Voronoi cells cover the force of attracting (repelling) of plasmodium instead of simple distances between it and attractants (repellents). In the current version of the language, the Voronoi cells are built within layers only.

Analogously to layers, the user can create and scatter on layers as many elements as needed.

Below, we present an exemplary fragment of a code in our language responsible for creating the layer and elements, setting individual properties of elements and scattering elements on the layer.

```
l1=new Layer;
p1=new Physarum;
a1=new Attractant;
a2=new Attractant;
a3=new Attractant;
a4=new Attractant;
l1.add(p1);
p1.setPosition(800,200);
l1.add(a1);
a1.setPosition(500,150);
a1.setIntensity(0.7);
l1.add(a2);
a2.setPosition(500,350);
a2.setIntensity(0.5);
l1.add(a3);
a3.setPosition(400,250);
a3.setIntensity(0.6);
l1.add(a4);
a4.setPosition(600,250);
a4.setIntensity(0.5);
```

For experiments with Physarum polycephalum computing, a specialized computer tool (*PhyChip Programming Platform*) is being developed using the Java environment. The tool consists of two main modules:

1. *Code creation and compilation module.* For generating the compiler of our language, the Java Compiler Compiler (JavaCC) tool [3] is used. JavaCC is the most popular parser generator for use with Java applications.
2. *Simulation module.* It enables the user to perform time simulation of growing pseudopodia, i.e., to run the program.

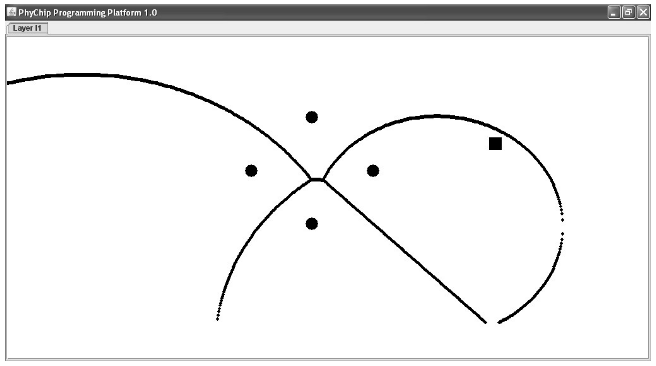

Fig. 3. The Voronoi cells for 4 attractants defined in the exemplary program generated in our tool (attractants are marked with dots whereas Physarum with a square).

In Figure 3, we have shown the Voronoi cells generated in our computer tool for 4 attractants ($a1$, $a2$, $a3$, $a4$) with different intensity assigned to them, defined in the exemplary program. Attractants are marked with dots whereas Physarum with a square. The measure defined earlier has been used to create cells. It is easy to see that Physarum is attracted first of all by the most right attractant.

4 Summation

In the paper, we have outlined theoretical foundations as well as assumptions for a new object-oriented programming language for Physarum polycephalum computing. The next mile steps in our research are the following: implementation of operations based on the π-calculus model [17] of processes and extension of the programming platform to the agent-oriented programming language for computation with raw plasmodium.

Acknowledgments

This research is being fulfilled by the support of FP7-ICT-2011-8.

References

1. Self, http://selflanguage.org/
2. ECMAScript, http://www.ecmascript.org/
3. JavaCC, http://java.net/projects/javacc/
4. JavaBeans. Tech. rep., Sun Microsystems (1997)
5. Adamatzky, A.: Reaction-diffusion algorithm for constructing discrete generalized voronoi diagram. Neural Network World 6, 635–643 (1994)
6. Adamatzky, A.: Computing in Nonlinear Media and Automata Collectives. Institute of Physics Publishing (2001)
7. Adamatzky, A.: Physarum machine: implementation of a kolmogorov-uspensky machine on a biological substrate. Parallel Processing Letters 17(4), 455–467 (2007)
8. Adamatzky, A.: Growing spanning trees in plasmodium machines. Kybernetes 37(2), 258–264 (2008)
9. Adamatzky, A., De Lacy Costello, B., T., S.: Universal computation with limited resources: Belousov-zhabotinsky and physarum computers. International Journal of Bifurcation and Chaos 18(8), 2373–2389 (2008)
10. Adamatzky, A.: Physarum Machines: Computers from Slime Mould. World Scientific (2010)
11. Adamatzky, A., Erokhin, V., Grube, M., Schubert, T., Schumann, A.: Physarum chip project: Growing computers from slime mould. International Journal of Unconventional Computing 8(4), 319–323 (2012)
12. Adamatzky, A., Costello, B.D.L., Asai, T.: Reaction-Diffusion Computers. Elsevier Science, Amsterdam (2005)
13. Chopard, B., Droz, M.: Cellular Automata Modeling of Physical Systems. Cambridge University Press (2005)
14. De Lacy Costello, B., Ratcliffe, N., Adamatzky, A., Zanin, A.L., Liehr, A.W., Purwins, H.G.: The formation of voronoi diagrams in chemical and physical systems: Experimental findings and theoretical models. International Journal of Bifurcation and Chaos 14(7), 2187–2210 (2004)
15. Ilachinski, A.: Cellular Automata: A Discrete Universe. World Scientific (2001)
16. Kolmogorov, A.: On the concept of algorithm. Uspekhi Mat. Nauk 8(4), 175–176 (1953)
17. Milner, R., Parrow, J., Walker, D.: A calculus of mobile processes, I. Information and Computation 100(1), 1–40 (1992)
18. Pavlović, D., Escardó, M.: Calculus in coinductive form. In: Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science. pp. 408–417 (1998)
19. Schönhage, A.: Real-time simulation of multi-dimensional turing machines by storage modification machines. Project MAC Technical Memorandum 37, MIT (1973)
20. Schönhage, A.: Storage modification machines. SIAM Journal on Computing 9(3), 490–508 (1980)
21. Tarjan, R.: Reference machines require non-linear time to maintain disjoint sets. Tech. Rep. STAN-CS-77-603 (1977)
22. Uspensky, V.: Kolmogorov and mathematical logic. The Journal of Symbolic Logic 57, 385–412 (1992)