

Physarum Chip: Growing Computers from Slime Mould. Logical Aspects

Andrew Schumann, Krzysztof Pancerz



UNIVERSITY of INFORMATION
TECHNOLOGY and MANAGEMENT
in Rzeszow, POLAND



Project is realized under 7th Framework Programme
co-financed from the European Commission, CORDIS and FET Proactive funds.

Physarum Chip: Growing Computers from Slime Mould. Logical Aspects

Andrew Schumann, Krzysztof Pancerz

Rzeszow, Szczecin 2016

**The publication has been financed by the
University of Information Technology and Management in Rzeszow**

Title: Physarum Chip: Growing Computers from Slime Mould. Logical Aspects

Authors: Andrew Schumann, Krzysztof Pancerz

Reviewers:

Prof. Andrew Adamatzky (Bristol, UK)

Dr Andrzej Szela (Rzeszow, Poland)

Publishers:

University of Information Technology and Management in Rzeszow

St. Sucharskiego 2, 35-225 Rzeszów, Poland

www.wsiz.rzeszow.pl, www.ksiegarnia.wsiz.pl

email: wsiz@wsiz.rzeszow.pl

Scientific Publishing House IVG

St. Cyfrowa 6, 71-441 Szczecin, Poland

www.wydawnictwoivg.pl

email: biuro@wydawnictwoivg.pl

ISBN 978-83-62062-73-7 eBook

ISBN 978-83-64286-58-2 eBook

© Copyright by Andrew Schumann and Krzysztof Pancerz

Printed in Poland by Scientific Publishing House IVG



**UNIVERSITY of INFORMATION
TECHNOLOGY and MANAGEMENT**
in Rzeszow



Scientific Publishing House IVG

PREFACE	4
INTRODUCTION	7
PART I. MATERIALS FROM PERIODIC REPORTS	8
1. Storage modification machine	8
2. Programming of <i>Physarum</i> storage modification machine	16
3. Formalisation of <i>Physarum</i> storage modification machine	31
4. Game-theoretic interface for <i>Physarum</i> storage modification machine	38
5. Published works	42
PART II. LOGICS OF PHYSARUM MACHINES. MATERIALS FROM CONFERENCE PRESENTATIONS. SELECTED SLIDES	47
1. Context-based games	47
2. Logic gates	61
3. p-Adic valued universe	69
4. Go games	76
PART III. PROGRAMMING OF PHYSARUM MACHINES. MATERIALS FROM CONFERENCE PRESENTATIONS. SELECTED SLIDES	82
1. Ladder diagrams and Petri net models	82
2. Rough-set extensions of transition systems	103
3. Object-oriented language	118
CONCLUSIONS AND FUTURE WORK	133

Preface

Physarum polycephalum we have studied in the project *Physarum Chip: Growing Computers from Slime Mould* belongs to the species of order *Physarales*, subclass *Myxogastromycetidae*, class *Myxomycetes*, division *Myxostelida*. It is commonly known as a true, a cellular or multi-headed slime mould. *P. polycephalum* has a complex life cycle. Plasmodium is a 'vegetative' phase, a single cell with a myriad of diploid nuclei. The plasmodium is visible to the naked eye. The plasmodium looks like an amorphous yellowish mass with networks of protoplasmic tubes. The plasmodium behaves and moves as a giant amoeba. It feeds on bacteria, spores and other microbial creatures and micro-particles. When foraging for its food the plasmodium propagates towards sources of food particles, surrounds them, secretes enzymes and digests the food. Typically, the plasmodium forms a network of protoplasmic tubes connecting the masses of protoplasm at the food sources which has been shown to be efficient in terms of network length and resilience. When several sources of nutrients are scattered in the plasmodium's range, the plasmodium forms a network of protoplasmic tubes connecting the masses of protoplasm at the food sources.

The main objectives of the project are to design and fabricate a distributed biomorphic computing device built and operated by slime mould of *Physarum polycephalum*. A Physarum chip (see Figure 1, 2) is a network of processing elements made of the slime mould's protoplasmic tubes coated with conductive substances; the network is populated by living slime mould. A living network of protoplasmic tubes acts as an active non-linear transducer of information, while templates of tubes coated with conductor act as fast information channels.

The Physarum chip will have parallel inputs (optical, chemo- and electro-based) and outputs (electrical and optical). The Physarum chip will solve a wide range of computation tasks, including optimisation on graphs, computational geometry, robot control, logic and arithmetical computing. The slime mould-based implementation is a bio-physical model of future nano-chips based on biomorphic mineralisation.

We envisage that research and development centred on novel computing substrates, as self-assembled and fault-tolerant fungal networks will lead to a revolution in the bio-electronics and computer industry. Combined with conventional electronic components in a hybrid chip, *Physarum* networks will radically improve the performance of digital and analog circuits.

Figure 1. Phyrasum chip.

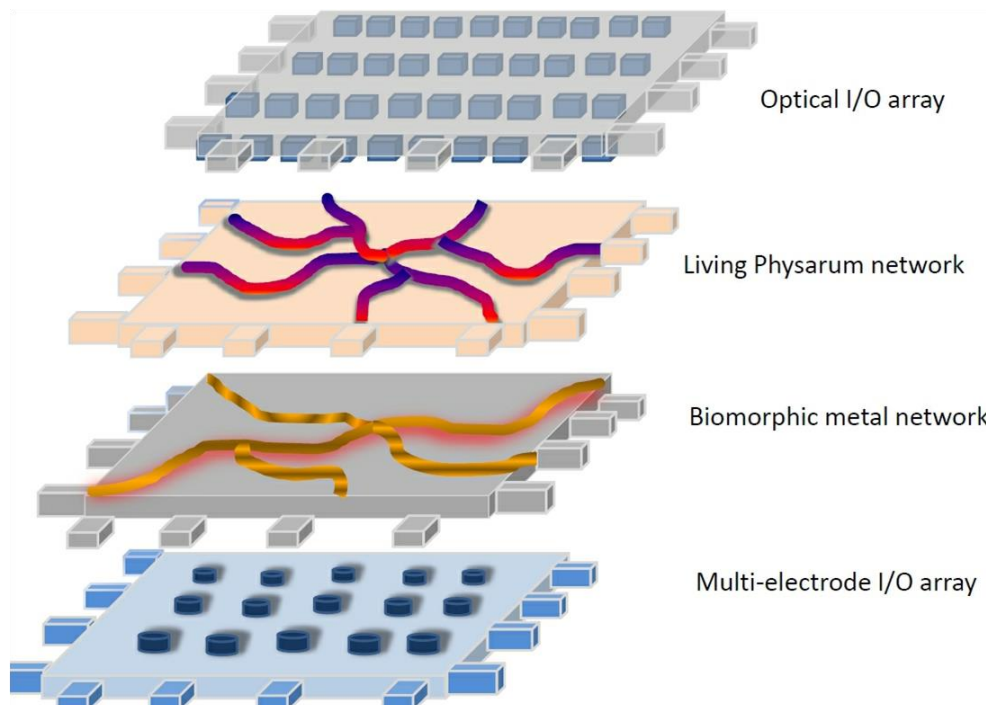
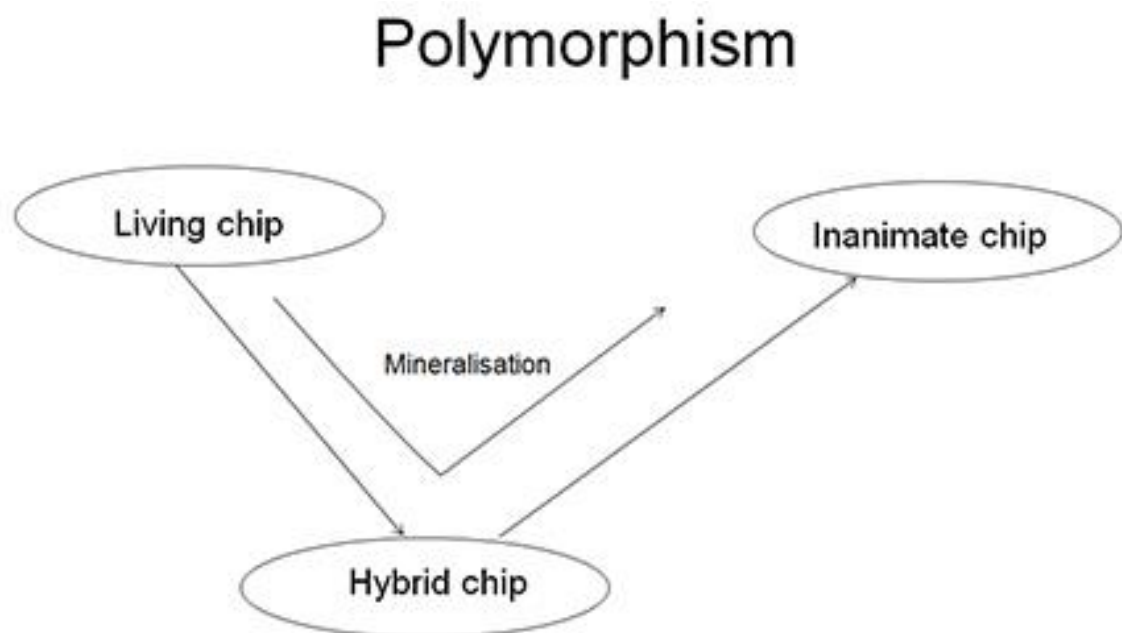


Figure 2. Polymorphism of Physarum chip.



Taking into account the enormous and growing interest of research centres and commercial laboratories in the recent experimental implementations of chemical, molecular and biological computers, we can predict that in the next 20-30 years, networks of slime mould mineralised and/or coated with compound substances will become a widespread commodity and a very promising component of novel information processing circuits.

This research has been supported by the Seventh Framework Programme (FP7-ICT-2011-8) and carried out under the leadership of prof. Andrew Adamatzky (Bristol, UK).

For more details please see <http://www.phychip.eu/>

*Andrew Adamatzky,
Victor Erokhin,
Martin Grube,
Theresa Schubert,
Andrew Schumann*

Introduction

Symbolic-logical, mathematical and programming aspects of the Physarum chip have been studied by Andrew Schumann and Krzysztof Pancerz in Rzeszow, Poland, and this book contains some materials from both periodic reports including a list of published works (Part I) and from conference presentations (Part II and Part III). In Part II we consider logics of Physarum machines and in Part III we consider a programming of *Physarum* machines. Part I and III are written jointly by Andrew Schumann and Krzysztof Pancerz. Part II is written solely by Andrew Schumann. This book does not cover all results obtained by us, but just some results which are mentioned in the periodic reports and presented at conferences. We are grateful for our collaboration to Andrew Adamatzky, Martin Grube, Jeff Jones, Andrei Khrennikov, Jan Woleński, and Ludmila Akimova with whom we have written some papers jointly. In this book we have used some experimental photos taken by our colleagues and published in some joint papers (the references to these papers are given with the photos properly).

The aim of this book is to help the reader to learn our main ideas implemented in the project fast, without reading long argumentation and mathematical or programming details. This book contains main logical ideas, philosophical presuppositions, and mathematical results used in designing the Physarum chip.

*Andrew Schumann,
Krzysztof Pancerz*

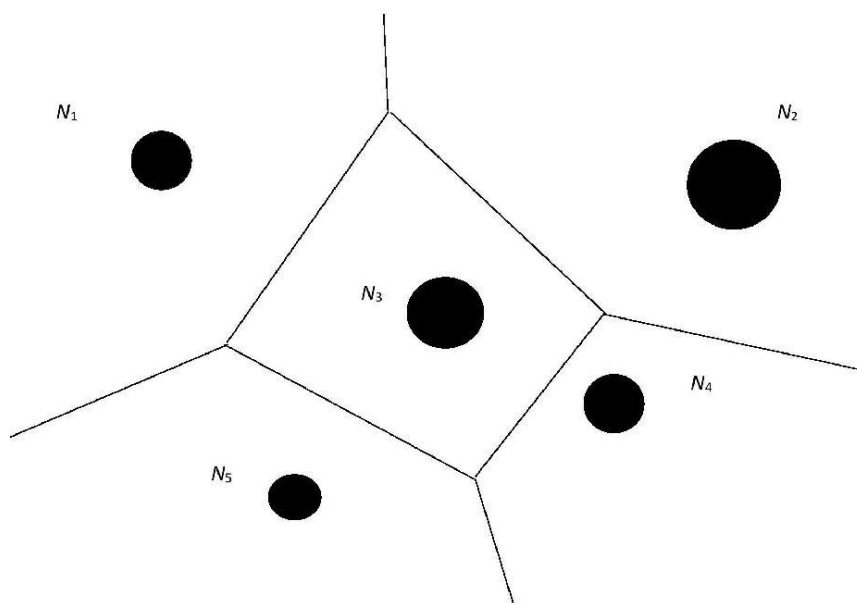
Part I. Materials from Periodic Reports

1. Storage modification machine

In the plasmodium behaviour we can implement different abstract automata such as Kolmogorov-Uspensky machines, Schönhage's storage modification machines, etc. In the meanwhile, plasmodium's active zones of growing pseudopodia are considered the key phenomenon of all these automata. These zones interact concurrently and in a parallel manner. At these active zones, three basic operations stimulated by nutrients (attractants) and some other conditions can be observed and defined as basic: fusion, multiplication, and direction operations. The *fusion* means that two active zones A and B either produce new active zone C (i.e. there is a collision of the active zones) or just a protoplasmic tube. The *multiplication* means that the active zone A splits into two independent active zones B and C propagating along their own trajectories. The direction means that the active zone is not translated to a source of nutrients but to a domain of an active space with certain initial velocity vector.

Attractants and repellents involved in the stimulation of plasmodium gives a topology which can be defined as a Voronoi diagram (see Fig. 3). Within one Voronoi cell a reagent has a full power to attract or repel the plasmodium. The distance is defined by intensity of reagent spreading like in other chemical reactions simulated by Voronoi diagrams. A reagent attracts or repels the plasmodium and the distance on that it is possible corresponds to the elements of a given planar set. When two spreading wave fronts of two reagents meet, this means that on the board of meeting the plasmodium cannot choose its one further direction and splits. Within the same Voronoi cell two active zones will fuse.

Figure 3. The Voronoi diagram for *Physarum polycephalum*, where different attractants have different intensity and power.



Let us consider an example of automata. In *Schönhage's storage modification machines* we deal with a fixed alphabet of input symbols, G , and a mutable directed graph with its arrows labelled by G and identified with possible protoplasmic tubes. The set of nodes X , identified with attractants, is finite. One fixed node a from 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 $x_g(a)=a$ for all g from G . That is, all pointers from the distinguished center node point back to the center node. Each g from G defines a mapping x_g from X to X in accordance with directions of growing pseudopodia; $x_g(b)$ is the node found at the end of the edge starting at b labelled by g . Each word of symbols in the alphabet G is a pathway through the machine from the distinguished center node. For example $ABBC$ would translate to taking path A from the start node, then path B from the resulting node, then path B , then path C . With respect to the word $ABBC$, the plasmodium moves.

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 ; its location in relation to the other nodes and pointers is determined by W . If W is the empty string, this has the effect of creating a new center node a , linked to the old a . For example, **new AB** creates a new node that is reached by following the B pointer from the node designated by A . The growing pseudopodia from active zone A to active zone B corresponds to

this word AB . Adding a new node B means adding a new attractant denoted by B 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 . If W is the empty string, then this has the effect of renaming the center node a to be the node indicated by V . Notice that **set W to V** means removing nodes and the edges incident to $W \setminus V$. So, we can remove some attractants denoted by $W \setminus V$ 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 else continue. This instruction serves to add edges between existing nodes. It corresponds to the splitting (multiplication) or fusion (fusion) of *Physarum*.

Thus, a *program of Physarum Schönhage's storage modification machine* is any action transforming sets X of nodes for growing pseudopodia with the alphabet G into other sets X' of nodes for growing pseudopodia with the same alphabet G which carries out by instructions **new W ; set W to V ; if $V=W$ then instruction Z** .

Within *Physarum Schönhage's storage modification machines* we can implement different logical systems such as Aristotelian syllogistics. In the *Physarum* implementation of Aristotelian syllogistics, all data points are denoted by appropriate syllogistic letters as attractants. A data point S is considered empty if and only if an appropriate attractant denoted by S is not occupied by plasmodium. We have syllogistic strings of the form SP with the following interpretation: S is P , and with the following meaning: SP is true if and only if S and P are neighbours and both S and P are not empty, otherwise SP is false. By this definition of syllogistic strings, we can define atomic syllogistic propositions as follows:

- *SaP*. In formal syllogistics: there exists A such that A is S and for any A , if A is S , then A is P . In *Physarum model*: there is a plasmodium at A and for any A , if A is connected to S by a protoplasmic tube, then A is connected to P by a protoplasmic tube.
- *SiP*. In formal syllogistics: there exists A such that both ' A is S ' is true and ' A is P ' is true. In *Physarum model*: there exists plasmodium at A such that A is connected to S by a protoplasmic tube and A is connected to P by a protoplasmic tube.
- *SeP*. In formal syllogistics: for all A , ' A is S ' is false or ' A is P ' is false. In *Physarum model*: for all plasmodia at A , A is not connected to S by a protoplasmic tube or A is not connected to P by a protoplasmic tube.
- *SoP*. In formal syllogistics: for any A , ' A is S ' is false or there exist A such that ' A is S ' is true and ' A is P ' is false. In *Physarum model*: for any plasmodia at A , A is not connected to S by a protoplasmic tube or there exists A such that A is connected to S by a protoplasmic tube and A is not connected to P by a protoplasmic tube.

Physarum strings of the form xy , yx are interpreted as particular affirmative propositions “Some x are y ” and “Some y are x ” respectively (i.e. as *SiP*), strings of the form $[xy]$, $[yx]$, $x[y]$, $y[x]$ are interpreted as universal negative propositions “No x are y ” and “No y are x ” (i.e. as *SeP*). A universal affirmative proposition “All x are y ” (i.e. *SaP*) are presented by a complex string $xy \& x[y]$. The sign $\&$ means that we have strings xy and $x[y]$ simultaneously and they are considered the one complex string.

Hence, a spatial expansion of plasmodium is interpreted as a set of syllogistic propositions. The universal affirmative proposition $xy \& x[y]$ means that the plasmodium at the place x goes only to y and all other directions are excluded. The universal negative proposition $x[y]$ or $[xy]$ means that the plasmodium at the place x cannot go to y and we know nothing about other directions. The particular affirmative proposition xy means that the plasmodium at the place x goes to y and we know nothing about other directions. Syllogistic conclusions allow us to mentally reduce the number of syllogistic propositions showing plasmodium's propagation.

The implementation of Aristotelian syllogistic as a particular version of system codified within *Physarum* Schönhage's storage modification machines shows us how it is difficult to guarantee only one direction of the growing plasmodium. In the most cases the plasmodium aims to move to different directions. In other words, while in Aristotelian syllogisms we are concentrating on one direction of many *Physarum* motions, therefore we are dealing with acyclic directed graphs with fusions of many protoplasmic tubes towards one data point, in the most cases of *Physarum* behaviour, not limited by repellents, we observe a spatial expansion of *Physarum* protoplasm in all directions with many cycles. Under these circumstances it is more natural to define all the basic syllogistic propositions in the way they would satisfy the inverse relationship, when all converses are valid: ‘All S are P ’ = ‘All P are S ’ and so on. In other words, then we can draw more natural conclusions for protoplasmic tubes which are decentralized and have some cycles. The formal syllogistic system over propositions with such properties is constructed in [A2]. This system is called *performative syllogistics*. Its atomic syllogistic propositions are defined as follows:

- *SaP*. In formal performative syllogistics: there exist A such that A is S and for any A , A is S and A is P . In *Physarum* model: there is a string AS and for any A which is a neighbour for S and P , there are strings AS and AP . This means that we have a massive-parallel occupation of region, where the cells S and P are located.
- *SiP*. In formal performative syllogistics: for any A , both ‘ A is S ’ is false and ‘ A is P ’ is false. In *Physarum* model: for any A which is a neighbour for S and P , there are no strings AS and AP . This means that the plasmodium cannot reach S from P or P from S immediately.
- *SeP*. In formal performative syllogistics: there exist A such that if ‘ A is S ’ is false, then ‘ A is P ’ is true. In *Physarum* model: there exists A which is a neighbour for S and P such that there is a string AS or there is a string AP . This means that the plasmodium occupies S or P , but surely not the whole region, where the cells S and P are located.

- *SoP. In formal performative syllogistics*: for any A , ‘ A is S ’ is false or there exist A such that ‘ A is S ’ is false or ‘ A is P ’ is false. *In Physarum model*: for any A which is a neighbour for S and P there is no string AS or there exist A which is a neighbour for S and P such that there is no string AS or there is no string AP . This means that the plasmodium does not occupy S or there is a neighbour cell which is not connected with S or P by a protoplasmic tube.

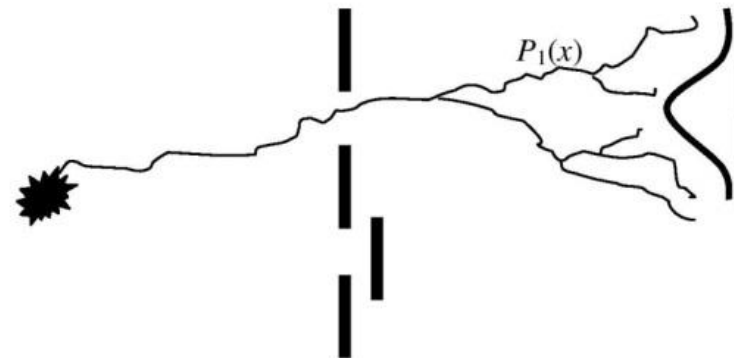
In the performative syllogistics we can analyse the collective dimension of behaviour. Within this system we can study how the plasmodium occupies all possible attractants in any direction if it can only see them. So, this system shows logical properties of a massive-parallel behaviour (i.e. the collective dimension of behaviour). One of the most significant notions involved in this implementation of performative syllogistics in *Physarum* topology is a *neighbourhood*. We can define a distance for the neighbourhood differently, i.e. we can make it longer or closer. So, from different neighbourhoods it will follow that we deal with different universes of discourse.

The *Physarum* performative syllogistics is a very simple logical system with massive-parallel conclusions. This system can be defined within *Physarum* Schönhage's storage modification machines. Nevertheless, it can be defined within a super-computing approach as well. The matter is that in the plasmodium behaviour we face a quantum uncertainty which says about that the plasmodium “calculates” much more, than conventional abstract automata such as *Physarum* Schönhage's storage modification machines.

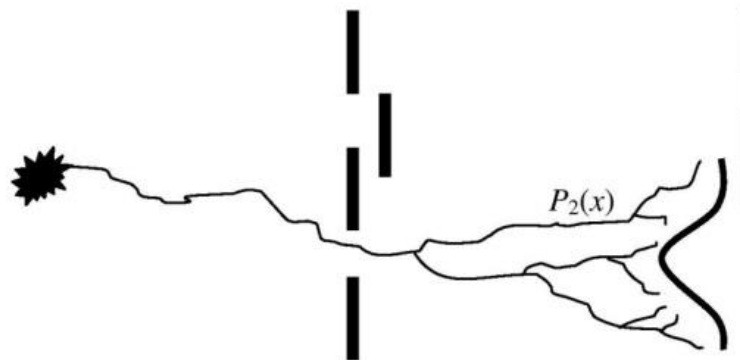
So, if we perform the double-slit experiment for *Physarum polycephalum*, we detect the very similar or even the same self-inconsistencies showing that we cannot approximate atomic individual acts of *Physarum* as well as it is impossible to approximate single photons. Indeed, to approximate atomic acts of *Physarum* we can carry out the double-slit experiment for *Physarum* to show that propagating protoplasmic tubes can be considered a collective behaviour at a time, too.

Let us take the first screen with two slits which are covered or opened and the second screen behind the first at which attractants are distributed evenly. Before the first screen there is an active zone of plasmodium. Then let us perform the following three experiments: (i) slit 1 is opened, slit 2 is covered; (ii) slit 1 is covered, slit 2 is opened; (iii) both slit 1 and 2 are opened (see Fig. 4). In the first (second) experiment protoplasmic tubes arrive at the screen at random in a region somewhere opposite the position of slit 1 (slit 2). We have a curve $P_1(x)$ (respectively, $P_2(x)$) which is interpreted probabilistically: $P_i(x) dx$ is equal the probability of tubes arriving at the screen in some region $(x, x + dx)$, where $i = 1, 2$.

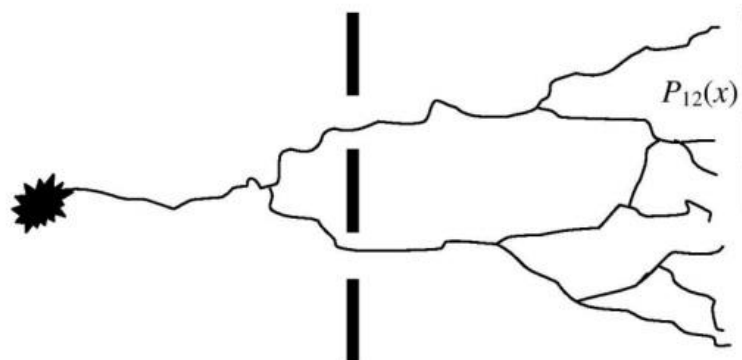
Figure 4. The result of reaching plasmodium protoplasmic tubes at a screen when (a) only slit 1 is open; (b) only slit 2 is open; (c) both slits are open. The curves $P_1(x)$, $P_2(x)$, $P_{12}(x)$ represent the intensity of the tubes passing through the slits [A11], [A15].



a



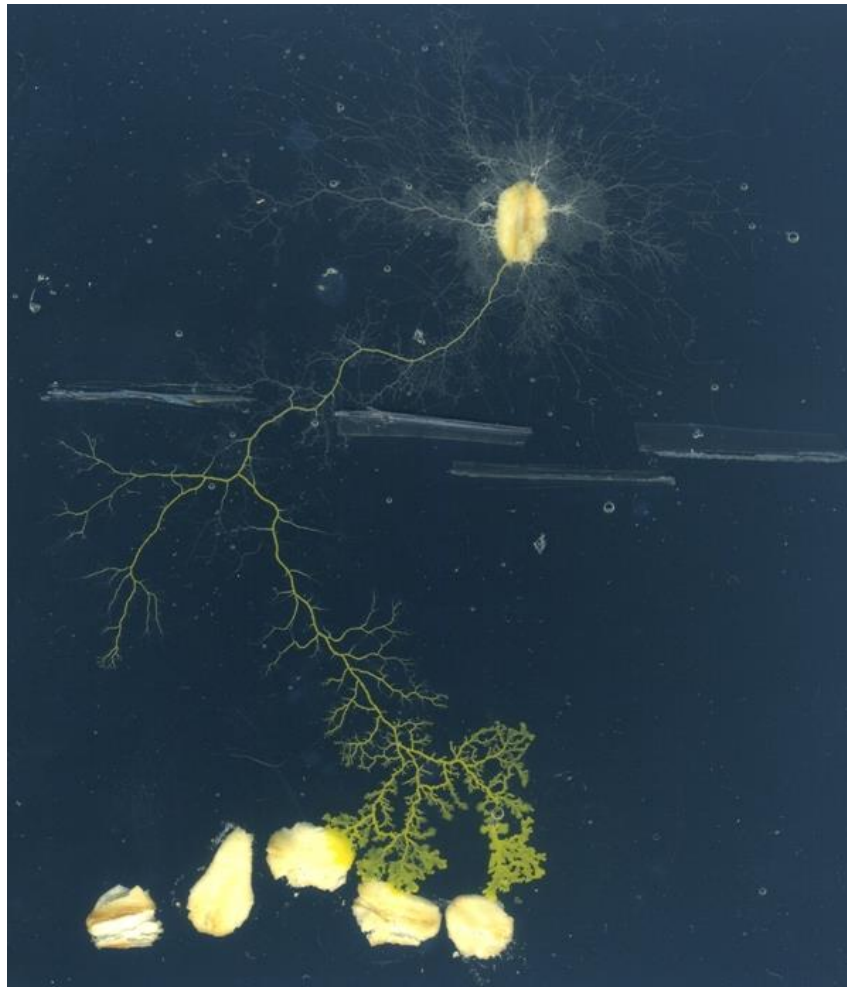
b



c

The difference from the experiments with particles consists in that tubes split before the second screen (see Fig. 4a, 4b) and we always have several tubes split from the one and reaching the screen in some region simultaneously (see Fig. 5). Let us denote all tubes landing at the second screen by A , thereby all tubes that pass through slit 1 by A_1 and all tubes that pass through slit 2 by A_2 . Now we can check if there is a partition of set A in case of *Physarum* into sets A_1 and A_2 . We open both slits. Then we see that the plasmodium behaves like electrons, namely it can propagate just one tube passing through either slit 1 or slit 2 or it can propagate two tubes passing through both slits simultaneously. In the second case, these tubes split before the second screen and do not always occur at the same place, i.e. they appear to occur randomly across the whole screen (Fig. 4c).

Figure 5. The real experiment with the *Physarum* plasmodium when only slit 1 is open, see [A15].



Thus, the total probability $P(A)$ corresponding to the intensity of plasmodium reaching the screen is not just the sum of the probabilities $P(A_1)$ and $P(A_2)$. This means that the plasmodium has the fundamental property of electrons discovered in the double-slit experiment. However, we can rather state that we observe the individual-collective duality in the behaviour, not only the particle-wave duality. Does it mean that electrons have the same duality instead of the particle-wave duality? In any case the experiment with *Physarum* shows more fundamental property of behaviour than the quantum uncertainty observed with electrons. This property is said to be the *individual-collective duality*.

In order to involve the individual-collective duality of plasmodia in our calculations, we can define a new abstract object – *wave set* [M1]. The wave set includes as many versions of behaviour of the whole system as possible. It looks like the quantum bit (or qubit for short). Let us remember that a classical bit can be 0 or 1. The qubit exists in various superpositions of 0 and 1 at the same time. There is an uncertainty of either 0 or 1, but qubit behaves like both 0 and 1. In other words, each quantum state is described in terms of classical states, associating two numbers with each: absolute amplitude (between 0 and 1) which when squared gives you the probability of obtaining that classical state when you measure the system, and a complex phase which governs interference effects. The classical states 0 and 1 are the top point and bottom point of the sphere, while the other states are in superposition. The points around the equator represent states where there is an equal probability of measuring 0 or 1. As a result, we can represent the state of a qubit as simply the point on the surface of a sphere, known as a *Bloch sphere*.

We sketched a formal logical language for extending any formal structure to wave sets [A8], [A13]. For example, using this language it is possible to extend Boolean algebra or group theory to systems on wave sets. Calculations within the media of wave sets can be defined as a version of super-computing. Notice that particular cases of Kolmogorov-Uspensky machines or Schönhage's storage modification machines are presented by game trees. Within the super-computing approach over wave sets we can define game structures for an instable environment [A8]. These structures describe the massive-parallel behaviour of plasmodia.

2. Programming of *Physarum* storage modification machine

2.1. *Logical-philosophical assumptions*

If we understand *Physarum* computing as super-computing over wave sets, we can use algorithms of *Physarum* computing for behavioural sciences such as behavioural economics, behavioural logic, game theory, decision theory, etc. In business intelligence the majority of expert systems used to analyse an organization's raw data appeal to statistical and econometric tools. In their possible applications they are extremely limited by some fundamental assumptions about the characters of material laws. First of all, it is assumed that the system of the material universe consists of primary bodies (atoms) and their combinations and relationships described by mathematical equalities, in particular it is supposed that each atom bears its own separate and independent effect so that the total state is being compounded of a number of separate effects detected in the proceeding state. In other words, in order to explore the total state we should present an appropriate proceeding state as an abstract machine such as Kolmogorov-Uspensky machine or Schönhage's storage modification machine.

Rene Descartes was one of the first thinkers who have put forward the assumption that wholes can be studied due to laws of connection between their individual parts described by maths, i.e. wholes are subject to different laws in proportion to the differences of their parts and these proportions can be analysed mathematically. This one of the main presuppositions of mathematical tools in science is called *measurability* and *additivity* of reality. Due to this assumption modern physics can have obtained all its results. For discovering the material universe it has appealed to *additive measures* such as mass, force, energy, temperature, etc. Economics and conventional business intelligence tries to continue this empiricist tradition and in statistical and econometric tools they deal only with the measurable aspects of reality. They try to obtain additive measures in economics and studies of real intelligent behaviour, also.

Nevertheless, there is always the possibility that there are important variables of economic systems which are unobservable and non-additive in principle. We should understand that statistical and econometric methods can be rigorously applied in economics just after the presupposition that the phenomena of our social world are ruled by stable causal relations between variables. However, let us assume that we have obtained a fixed parameter model with values estimated in specific spatio-temporal contexts. Can it be exportable to totally different contexts? Are real social systems governed by stable causal mechanisms with atomistic and additive features?

Thus, in statistical and econometric tools of business intelligence we accept only phenomena with causal connections measured by additive measures. Nevertheless, in the social world we deal with symbolic interactions studied by *non-additive labels* (symbolic meanings or symbolic values). For accepting the variety of such phenomena we should avoid additivity of basic labels.

Non-additivity of phenomena does not mean that they cannot be studied mathematically. There are some rigorous approaches such as p-adic probability theory which allow us to do it. There has been developed even p-adic quantum mechanics started from the publication of Igor Volovich in 1987. In p-adic quantum mechanics, p-adic probabilities are applied instead of real ones.

The most significant feature of p-adic probabilities (or more generally, non-Archimedean probabilities or probabilities on infinite streams) is that they do not satisfy additivity. On the one hand, the p-adic analogies of the central limit theorem in real numbers face the problem that the normalized sums of independent and i.i.d. random variables do not converge to a unique distribution, there are many limit points, therefore there is no connection with the usual bell type curve. In other words, in p-adic distributions we cannot build up the Gauss curve as fundamental notion of statistics and econometrics. On the other hand, the power set over infinite streams like p-adic numbers is not a Boolean algebra in general case. In particular, there is no additivity – we cannot obtain a partition for any set into disjoint subsets whose sum gives the whole set [M1], [A8], [A9]. Using p-adic (non-Archimedean) probabilities we can disprove Aumann's agreement theorem [A8], [A9] and develop new mathematical tools for game theory, in particular define context-based games by means of coalgebras or cellular automata [A10], [A12]. In these context-based games we can appeal just to non-Archimedean probabilities. These games can describe and formalize complex reflexive processes of behavioural finances (such as short selling or long buying).

The programming language for *Physarum* behaviour we have constructed, on the one hand, simulates the *Physarum* behaviour and, on the other hand, shows which mathematical tools can be implemented in its behaviour. In particular, we consider *Physarum polycephalum* as simulation model for context-based games and behavioural finances.

Let A be a set of any nature. It is built up over atoms. Its powerset denoted by $\mathbf{P}(A)$ is defined as a family of all subsets of A . Let U be the universe consisting of things as atoms. Every member of $\mathbf{P}(U)$ is called *event*. According to Descartes, the material universe is measurable. This means that each event E may have a characteristic number. Let this number $P(E)$ be called *probability measure* of E . Hence, $P(\cdot)$ is regarded as a *set function* (i.e., a function with sets constituting its domain).

The probability measure satisfies the following three axioms: (i) *measurability*; (ii) *certainity*; (iii) *additivity*.

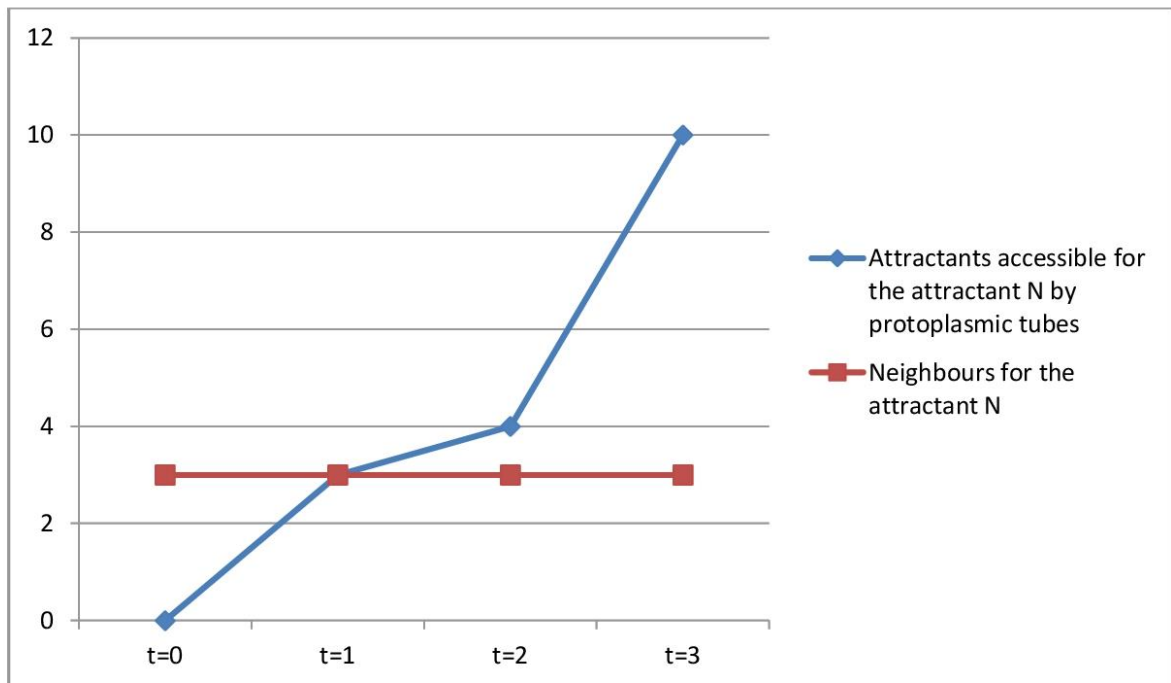
In statistical and econometric tools of business intelligence these axioms are basic, too. However, if we would like to involve quantitative methods to analysing non-additive labels of symbolic interactions, we should avoid these axioms. In symbolic interactions we cannot define additive measures. Conventionally, probability measures run over real numbers of the unit $[0,1]$ and its domain is a Boolean algebra of $\mathbf{P}(U)$ with atoms.

Let us suppose that the sample space U is not fixed, but changes continuously. It can grow, be expanded, decrease or just change in itself. In this case we will deal not with atoms as members of U , but with streams. Let us denote this non-stable set by U^* and call it a *wave set*. The powerset $\mathbf{P}(U^*)$ cannot be a Boolean algebra [M1], [A8], [A9].

We can consider *Physarum* behaviours within a certain topology of attractants and repellents as wave sample set U^* . *Physarum* behaves by plasmodia which can have a form of either waves or protoplasmic tubes. Plasmodia grow from active zones concurrently and in a parallel manner toward attractants. At the same time, they avoid repellents.

Assume that there are two neighbour attractants a and b . We say that there is a string ab or ba if both attractants a and b are occupied by the plasmodium. As a result, we observe a continuous expansion of the set of strings. It can be regarded as U^* (see Fig. 6).

Figure 6. The number of members of U which satisfy properties ‘Attractants accessible for the attractant N by protoplasmic tubes’ and ‘Neighbours for the attractant N ’ for time $t=0, 1, 2, 3$.



The wave set U^* consists of streams of data for different time $t=0, 1, 2, 3, \dots$. Let us show how we can build up U^* constructively. Suppose that A, B are subsets of U and $A :=$ ‘Attractants accessible for the attractant N by protoplasmic tubes’ and $B :=$ ‘Neighbours for the attractant N ’. The property B is verified on the same number of members of U for any time $t=0, 1, 2, 3, \dots$. Nevertheless, the property A is verified on a different number of members of U for different time $t=0, 1, 2, 3, \dots$. This means that we can deal with infinite streams:

$$/A^*/ := (/A/ \text{ for } t=0; /A/ \text{ for } t=1; /A/ \text{ for } t=2, \dots);$$

$$/B^*/ := (/B/ \text{ for } t=0; /B/ \text{ for } t=1; /B/ \text{ for } t=2, \dots);$$

$$/U^*/ := (/U/ \text{ for } t=0; /U/ \text{ for } t=1; /U/ \text{ for } t=2, \dots);$$

where $/X/$ means a cardinality number of X . Notice that if $/U/ = p - 1$, then $/A^*/, /B^*/, /U^*/$ cover p-adic integers.

Thus,

$$A^* := (A \text{ for } t=0; A \text{ for } t=1; A \text{ for } t=2, \dots);$$

$$B^* := (B \text{ for } t=0; B \text{ for } t=1; B \text{ for } t=2, \dots);$$

$$U^* := (U \text{ for } t=0; U \text{ for } t=1; U \text{ for } t=2, \dots).$$

They are wave sets which can be defined as families of infinite streams mutually dependent on each other. Cardinalities of those unconventional sets are non-Archimedean numbers. Then we can define the probability of A^* by the standard proportional relation:

$$P(A^*) := P_{S^*}(A^*) = n(A^*)/N,$$

where

$$/S^*/ = N, n(A^*) = /A^* \& S^*/.$$

These probabilities are non-Archimedean [A8]. In this way, on the medium of plasmodia we can disprove Aumann’s agreement theorem and many other statements of classical game theory. In other words, we can define on the medium of plasmodia knowledge operators, game strategies, game rules, etc.

Let us assume that attractants are regarded as payoffs for *Physarum* and active zones of pseudopodia are regarded as players. Let us show how we can define a zero-sum game. Andrew Adamatzky has performed promising experiments showing that there are cases when sets B^*_i ($:=$ ‘Attractants occupying by agent i ’) are disjoint. Let us suppose that we have only two agents. The first is presented by usual *Physarum polycephalum* plasmodia. The second by another species called *Badhamia utricularis* plasmodia (*PhyX* for short). *Physarum* grows definitely faster than *PhyX* and overtakes more flakes at the same time than the latter (see Fig. 7). Only if the inoculum was “fatter” for *PhyX*, *PhyX* might grow faster. Moreover, if the invasive growth front of *PhyX* is well nourished by oat it easily overgrows the opposing tube system of *Physarum*. So, at the microscopic level we can find out that in most observations *Physarum* could grow into branches of *PhyX*, while *PhyX* could grow over *Physarum* strands [P6], [P13]. We can see that somehow *Physarum* feeds on small branches of *PhyX* (see Fig. 7).

Thus, in case of *Physarum* and *PhyX* we observe a competition in the small branches. For them some knowledge operators are disjoint.

Figure 7. The experiment with two agents: fronts of growing pseudopodia of *Physarum polycephalum* (Phys) and *Badhamia utricularis* (Phyx), see [P6], [P13].

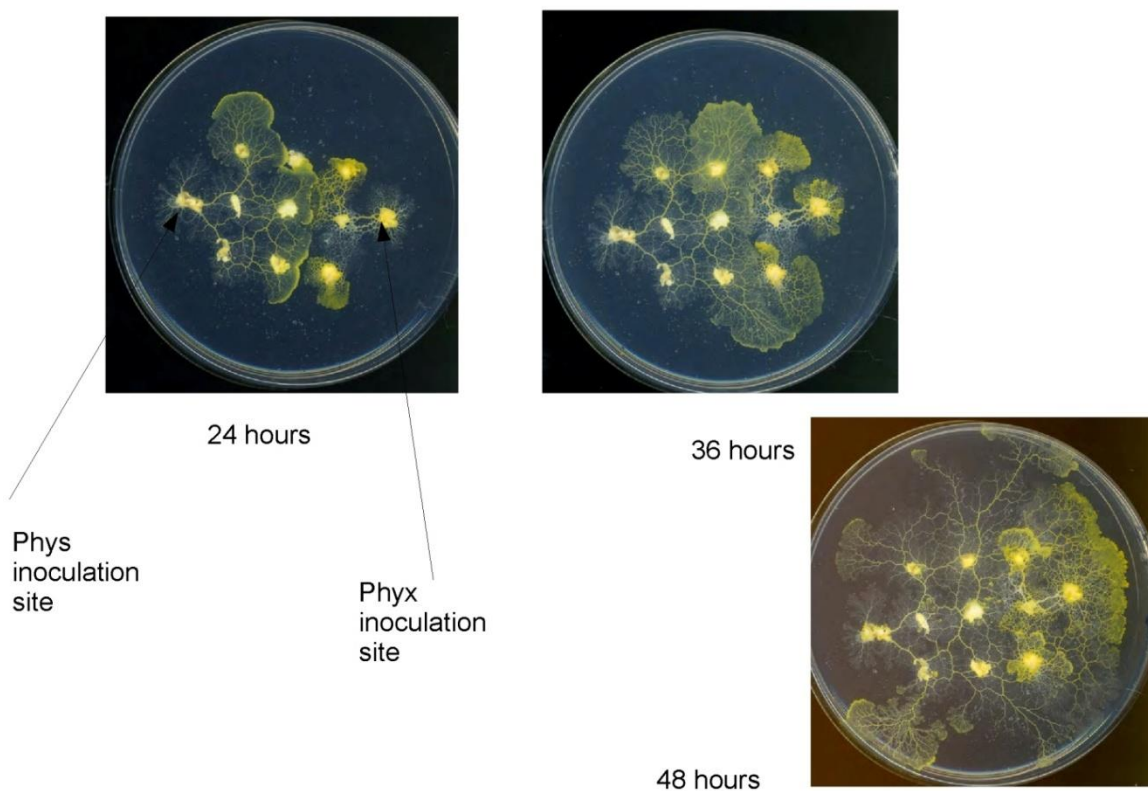
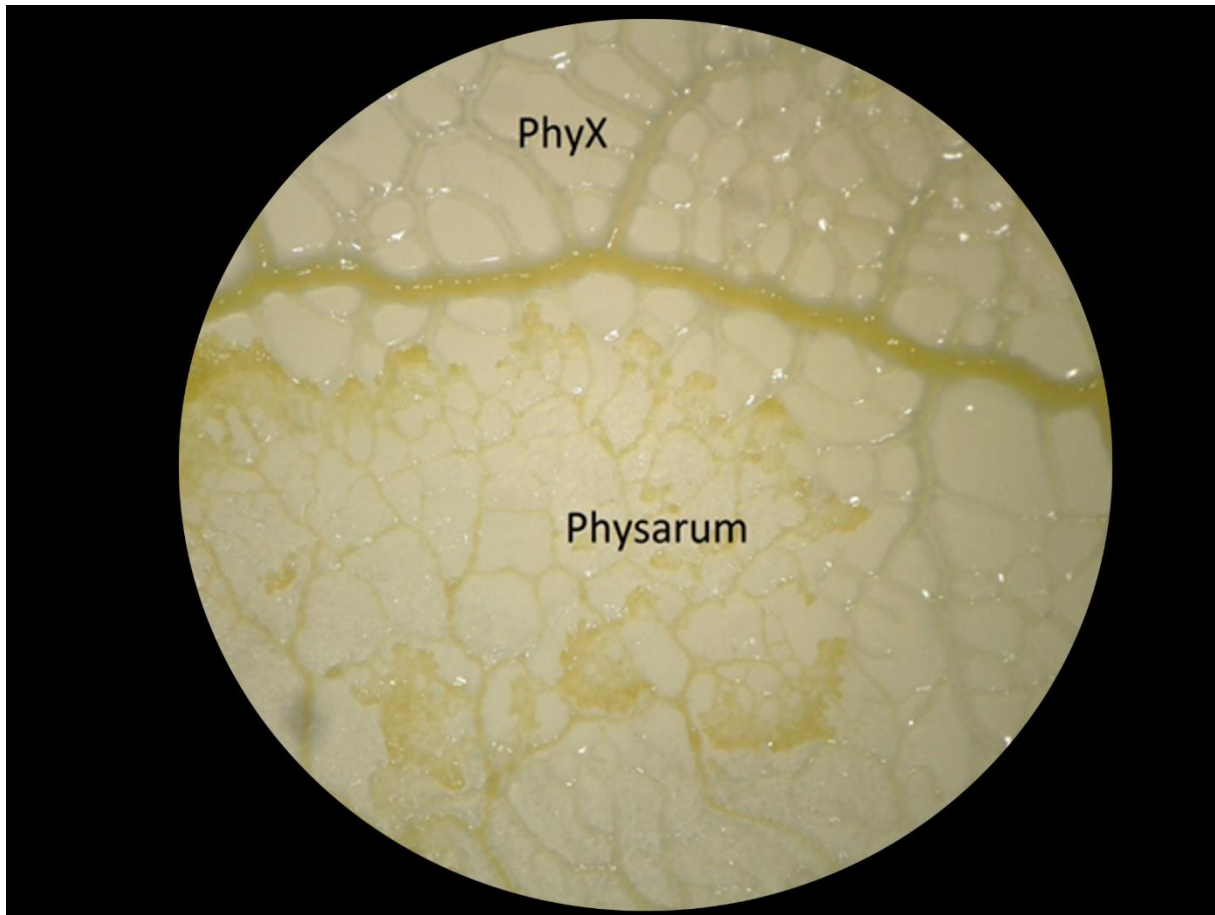
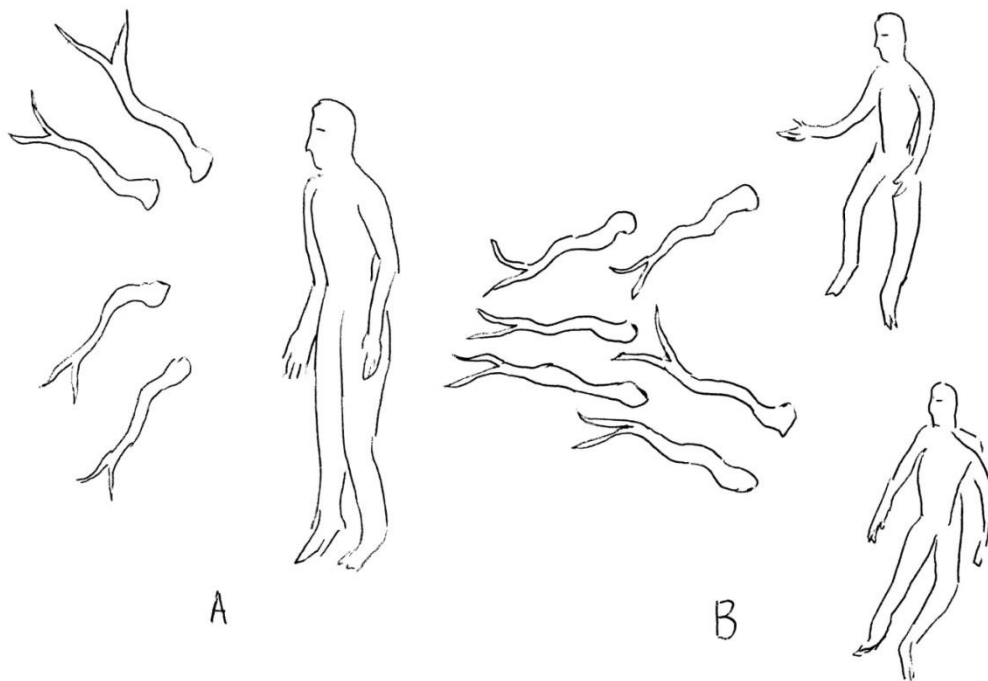


Figure 8. The experiment with two agents: *Physarum polycephalum* could grow into branches of *Badhamia utricularis* (PhyX), see [P6], [P13].



Thus, storage modification machines on plasmodia are not conventional. First of all, we assume the principle of individual-collective duality [P15], [A15], therefore we avoid the notion of elementary (atomic) acts. As a result, we use the notion of wave sets in computation and appeal to non-Archimedean probabilities in probability theory and game theory. For example, let us consider cercariae, free-swimming larvae of pubertal generation parasitizing vertebrate animals. They are capable to insinuate into skin of human being who is for them a casual host, invoking at him/her an allergic reaction, the so-called cercarial dermatitis. In other words, human beings can play role of attractants for cercariae collectives (see Fig. 9). The behaviour of collectives of trematode larvae (miracidiae and cercariae) can be considered intelligent. For describing that behaviour, we appealed to the behavioural logic of *Physarum polycephalum*. We detected that the behaviour of one-cell organism of *Physarum* and the behaviour of local group of cercariae have identical patterns. So, these patterns may be considered an effect of the individual-collective duality. In some situations at a time, living agents may behave individually, but in the same situations at another time, they may behave collectively.

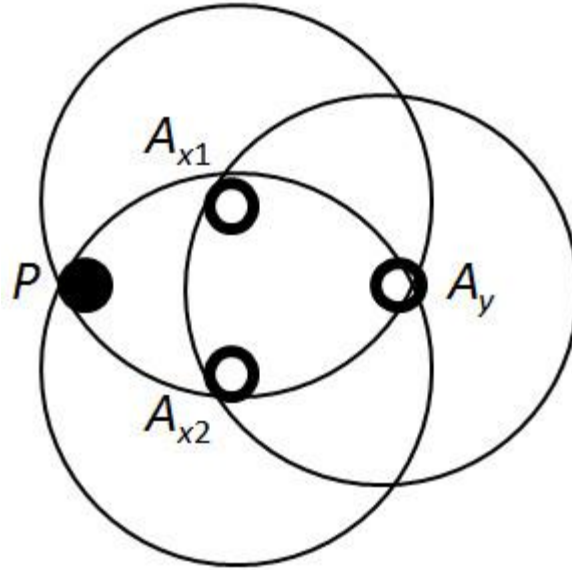
Figure 9. The stimulation of the following operations in cercariae motions: (A) the fusion of cercariae collectives, (B) the multiplication of cercariae collective, where the human beings are attractants and cercariae collectives behave like *Physarum* protoplasmic tubes which perform fusion or multiplication.



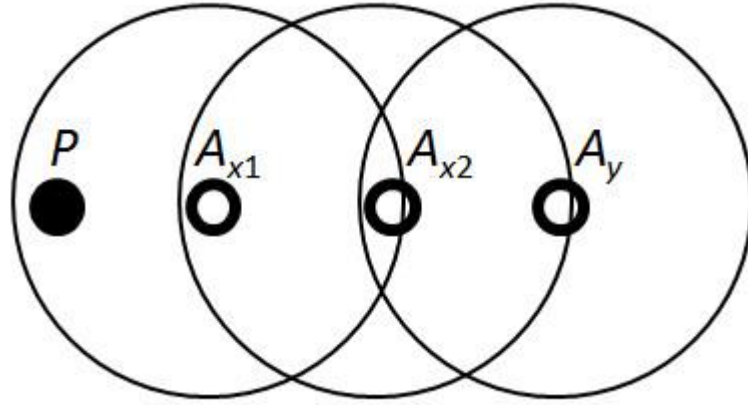
2.2. Programming assumptions

We have obtained a basis of new object-oriented programming language for *P. polycephalum* computing. Within this language we can check possibilities of practical implementations of storage modification machines on plasmodia and their applications to behavioural science such as behavioural economics and game theory. The proposed language can be used for developing programs for *P. polycephalum* by the spatial configuration of stationary nodes. Geometrical distribution of stimuli can be identified with a low-level programming language for *Physarum* machines.

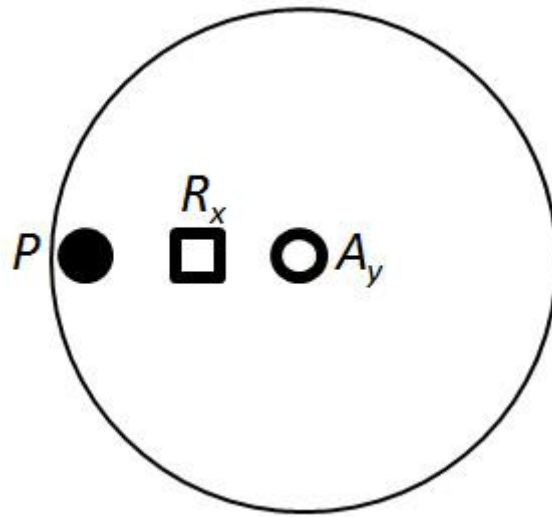
At the beginning, we have proposed to construct logic gates through the proper geometrical distribution of stimuli for *P. polycephalum*. This approach has been adopted from the ladder diagram language widely used to program Programmable Logic Controllers (PLCs). Flowing power has been replaced with propagation of plasmodium of *P. polycephalum*. Plasmodium propagation is stimulated by attractants and repellents. Rungs of the ladder can consist of serial or parallel connected paths of *Physarum* propagation. A kind of connection depends on the arrangement of regions of influences of individual stimuli. If both stimuli influence *Physarum*, we obtain alternative paths for its propagation. It corresponds to a parallel connection (i.e., the OR gate).



If the stimuli influence *Physarum* sequentially, at the beginning only the first one, then the second one, we obtain a serial connection (i.e., the AND gate).



The NOT gate is imitated by the repellent avoiding *Physarum* propagation.



In the proposed approach, we have assumed that each attractant (repellent) is characterized by its region of influence (ROI) in the form of a circle surrounding the location point of the attractant (repellent), i.e., its center point. The intensity determining the force of attracting (repelling) decreases as the distance from it increases. A radius of the circle can be set assuming some threshold value of the force. The plasmodium must occur in a proper region to be influenced by a given stimulus. This region is determined by the radius depending on the intensity of the stimulus. Controlling the plasmodium propagation is realised by activating/deactivating stimuli.

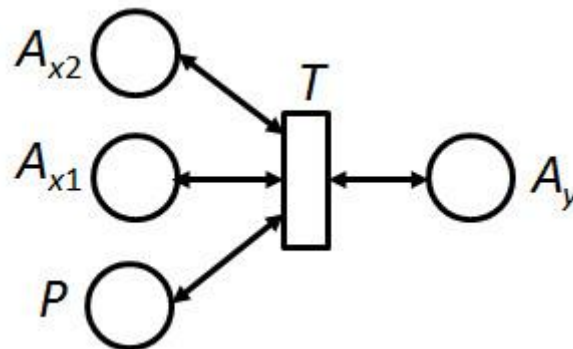
Logic values for inputs have the following meaning in terms of states of stimuli: 0 – attractant/repellent deactivated, 1 – attractant/repellent activated. Logic values for outputs have the following meaning in terms of states of stimuli: 0 – absence of *P. polycephalum* at the attractant, 1 – presence of *P. polycephalum* at the attractant.

At the second stage, we have adopted more abstract models than distribution of stimuli to program *P. polycephalum* machines which can be identified with programming in the high-level language. The choice fell on *Petri nets*. Petri nets were first developed by C.A. Petri. Petri nets are a powerful graphical language for describing processes in digital hardware. We have shown how to build Petri net models, and next implement as *P. polycephalum* machines, of basic logic gates AND, OR, NOT, and simple combination circuits. In our approach, we use Petri nets with inhibitor arcs. Inhibitor arcs are used to disable transitions. Inhibitor arcs test the absence of tokens in a place. A transition can only be if all its places connected through inhibitor arcs are empty. This ability of Petri nets with inhibitor arcs is used to model behaviour of repellents. Plasmodium of *Physarum* avoids light and some thermo- and salt-based conditions and this fact can be modelled by inhibitor arcs. The Petri net model (code in the high-level language) can be translated into the code in the low-level language, i.e., geometrical distribution of attractants and repellents of the *Physarum* machine.

In the proposed Petri net models, we can distinguish three kinds of places:

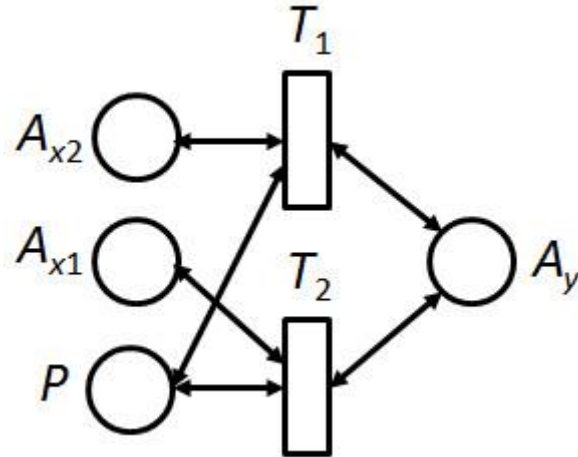
- (1) Places representing *P. polycephalum*. For such a place, the presence of the token means that plasmodium of *Physarum* is present in the origin point. Otherwise, the absence of the token means that there is no plasmodium.
- (2) Places representing input attractants or repellents. For such a place, the presence of the token means that the attractant/repellent is activated. Otherwise, the absence of the token means that attractant/repellent is deactivated.
- (3) Places representing output attractants. For such a place, the presence of the token denotes the present of *Physarum* at the attractant (*Physarum* occupies the attractant). Otherwise, the absence of the token denotes the absence of *Physarum polycephalum* at the attractant.

In the AND gate, the transitions T represents the flow (propagation) of plasmodium from the origin place to the output attractant. T is enabled to fire if both attractants are activated.

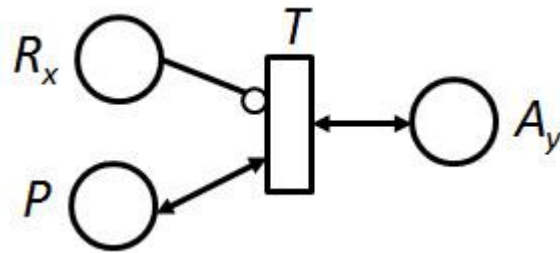


In the OR gate, the transitions T_1 and T_2 represent the alternative flows of plasmodium from the origin place to the output attractant. T_1 is enabled to fire if the first attractant is activated.

T_2 is enabled to fire if the second attractant is activated.

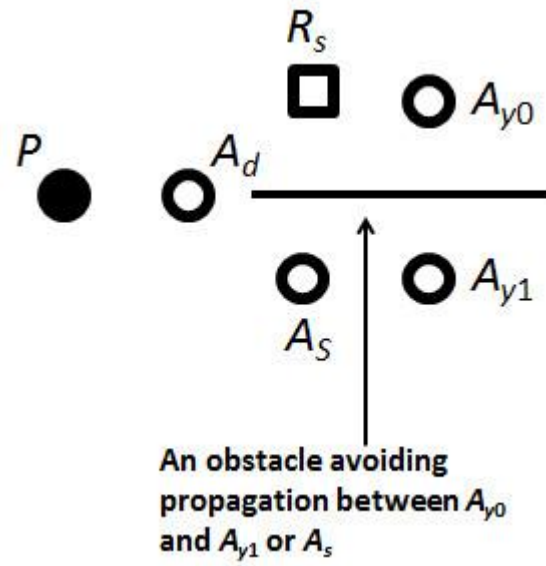


In the NOT gate, the transition T represents the flow (propagation) of plasmodium from the origin place to the output attractant. T is enabled to fire if the repellent is deactivated.



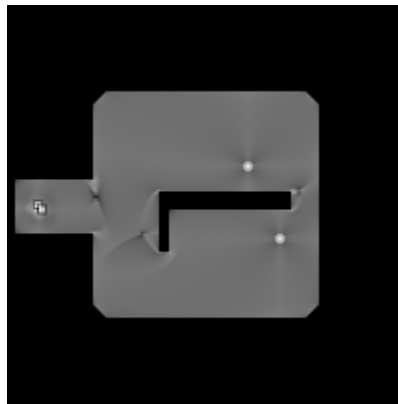
Then, experimentally, we have built a *P. polycephalum* demultiplexer based on the ladder diagram structure. A demultiplexer is a device taking a single input signal and selecting one of many data-output lines, which is connected to the single input. So, using the ladder diagram approach, we can determine geometrical distribution of attractants and repellents for the 1-to-2 demultiplexer [P4], see Fig. 6.

Figure 6. The 1-to-2 demultiplexer.

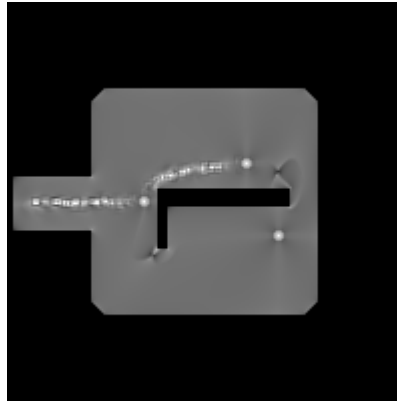


Experiments have been performed using the experimental environment for a particle model of *P. polycephalum*. Pictures taken by Jeff Jones show how *Physarum polycephalum* was propagated in each situation, see [P4].

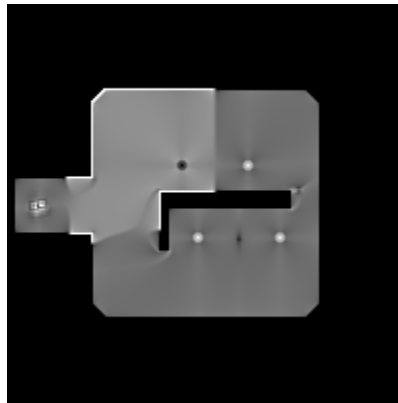
(1) $s = 0$ and $d = 0$: uneventful, because there is no data regardless of switch position.



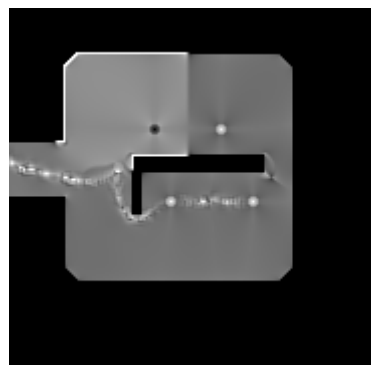
- (2) $s = 0$ and $d = 1$: no repellent causes the stream to go to the upper output attractant, the model does not grow down because it is outside the region of influence of the lower output attractant.



- (3) $s = 1$ and $d = 0$: uneventful, because there is no data regardless of switch position.



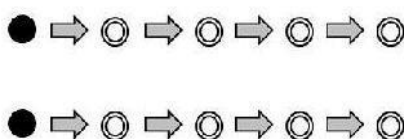
- (4) $s = 1$ and $d = 1$: the repellent causes selection of the lower path to the lower output attractant.



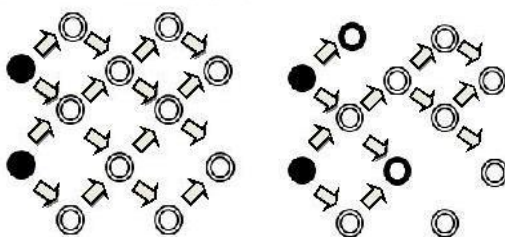
Using the object-oriented programming language for *P. polycephalum* computing, different logical systems from behavioural logic to illocutionary logic and from context-based games to epistemic logic on non-Archimedean probabilities are implementable. Let us consider how we have implemented reflexive games as a kind of context-based games. Attractants may be regarded as *payoffs* for *Physarum*. Plasmodia occupy attractants step by step. By different localizations of attractants we can affect on *Physarum* motions differently. We can interpret the stimuli for *Physarum* motions as Boolean functions on payoffs. Boolean functions are designed by logical gates mentioned above.

The game is designed as a tunnel consisting of attractants and repellents combined as logical gates. Plasmodia propagate towards the tunnel and face different lines. The first line of logical gates in the tunnel which is occupied by plasmodia implement Boolean functions at time $t = 0$. The second line of logical gates implement Boolean functions at time $t = 1$. The third line of logical gates implement Boolean functions at time $t = 2$, etc. The number of plasmodia before the tunnel corresponds to the number of propositional streams we are going to implement. Each plasmodium is active within a neighbourhood consisting of an appropriate set of attractants and depending on ranges of influence. Thus, the number of plasmodia before the tunnel corresponds to the number of neighbourhoods as well.

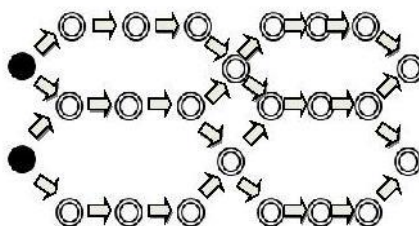
Looking for maximal payoffs (this tunnel consists of 2 neighbourhoods):



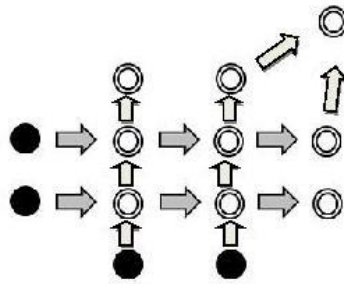
Looking for minimal payoffs (this tunnel consists of 2 neighbourhoods, too):



Saddle Point:



Equilibrium in Dominant Strategies:



Thus, the idea of extensions of abstract automata to super-computing over wave sets allows us to develop new approaches in probability theory and behavioural sciences from behavioural finance to game theory.

3. Formalisation of *Physarum* storage modification machine

3.1. *Process algebra*

In the object-oriented programming language for simulating the plasmodium motions we are based on *process-algebraic* formalizations of *Physarum* storage modification machine [P8]. So, we consider some instructions in *Physarum* machines in terms of process algebra like: add node, remove node, add edge, remove edge [P9], [P8]. Adding and removing nodes can be implemented through activation and deactivation of attractants, respectively. Adding and removing edges can be implemented by means of repellents put in proper places in the space. An activated repellent can avoid a plasmodium transition between attractants. Adding and removing edges can change dynamically over time. To model such behaviour, we propose a high-level model, based on timed transition systems [P9].

In this model we define the following four basic forms of *Physarum* transitions (motions): direct (direction: a movement from one point, where the plasmodium is located, towards another point, where there is a neighbouring attractant), fuse (fusion of two plasmodia at the point, where they meet the same attractant), split (splitting plasmodium from one active point into two active points, where two neighbouring attractants with a similar power of intensity are located), and repel (repelling of plasmodium or inaction).

In *Physarum* motions, we can perceive some ambiguity influencing on exact anticipation of states of *Physarum* machines in time. In case of splitting plasmodium, there is some uncertainty in determining next active points (attractants occupied by plasmodium) if a given active point is known. This uncertainty does not occur in case of direction, where the next active point is uniquely determined. To model ambiguity in anticipation of states of *Physarum* machines, we propose to use rough set theory. Analogously to the lower and upper approximations, we define the lower and upper predecessor anticipations of states in the *Physarum* machine [A16]. Behaviour of *Physarum* machines can also be modelled using Bayesian networks with probabilities defined on rough sets [A16].

Thus, we propose some timed and probabilistic extensions of standard process algebra to implement timed and rough set models of behaviour of *Physarum* machines in our new object-oriented programming language, called here the *Physarum* language, for *Physarum polycephalum* computing [A16], [P9], [P8]. In this language we define Bayesian networks on rough sets.

3.2. Computation on trees

Computations on tree are usually represented by spatial algorithms like Kolmogorov-Uspensky machines. Theoretically, Turing machines, Kolmogorov-Uspensky machines, Schönhage's storage modification machines, and random-access machines have the same expressibility power. In other words, the class of functions computable by these machines is the same. Unfortunately, the computational power of their implementations on the *Physarum polycephalum* medium is low [A12].

The point is that not every computable function can be simulated by plasmodium behaviours: first, the motion of plasmodia is too slow (several days are needed to compute simple functions such as 5-bit conjunction, 3-bit adder, etc., but the plasmodium stage of *Physarum polycephalum* is time-limited, therefore there is not enough time for realizations, e.g., of thousands-bit functions); second, the more attractants or repellents are involved in designing computable functions, the less accuracy of their implementation is, because the plasmodium tries to be propagated in all possible directions and we will deal with directed graphs and other problems; third, the plasmodium is an adaptive organism that is very sensitive to environments, therefore it is very difficult to organize the same laboratory conditions for calculating the same k -bit functions, where k is large.

To make computations on tree more expressive we propose a syllogistic system of propagation. This system can logically simulate a massive-parallel behaviour in the propagation of collectives of parasites. In particular, this system simulates the behaviour of collectives of *Trematode* larvae (miracidia and cercariae) [A14]. Also, this syllogistic system of propagation can be used as basic logical theory for quantum logic [A11]. In this theory we can build non-well-founded trees for which there cannot be logical atoms [A11]. This theory is much more expressive than standard spatial algorithms in simulating the plasmodium motions [P10].

So, protoplasmic trees can be interpreted as syllogistic trees. In this way while Aristotelian syllogistic may describe concrete directions of *Physarum* spatial expansions, performative syllogistic proposed by us [P7] may describe *Physarum* simultaneous propagations in all directions. Therefore, while for the implementation of Aristotelian syllogistic we need repellents to avoid some possibilities in the *Physarum* propagations, for the implementation of performative syllogistic we do not need them [P7]. Performative syllogistic has a p-adic valued semantics and satisfies p-adic valued probabilities [P10]. The syllogistic can be extended to a more general theory of context-based games. This theory is proposed in [P6], [P13]. Within this theory we can define algorithms for computing on protoplasmic tree.

Computations on protoplasmic tree are understood as a kind of extension of concurrent processes defined in concurrent games proposed by Samson Abramsky. This extension is called context-based processes and they are defined in the theory of *context-based games* proposed by us [P6], [P13].

Thus, we define some unconventional algorithms on non-well-founded trees to make calculations on plasmodia more expressive [P10]. These algorithms can be used for constructing an alternative quantum logic (without logical atoms) [A11] and a simulation model for propagating parasites [A14].

3.3. *Cellular automata*

The universe, where *Physarum* lives, consists of cells possessing different topological properties according to the intensity of chemo-attractants and chemo-repellents. The intensity entails the natural or geographical neighbourhood of the set's elements in accordance with the spreading of attractants or repellents. As a result, we obtain Voronoi cells [A17]. In this structure we can implement cellular automata. Taking into account the fact that the plasmodium is very sensitive to the environment and can change its strategies we can extend the standard notion of cellular automata and assume that transition rules can change in an appropriate neighbourhood in accordance with some outer conditions at time step $t = 0, 1, 2 \dots$. The theory of these automata with changing transition rules is proposed in [A10], [A13].

The plasmodium implements *cellular automata with changing transition rules*. Within these automata we can define context-based concurrent formal theories [A13].

3.4. *p*-Adic logic and arithmetics

The slime mould is considered a natural fuzzy processor with fuzzy values on the set of p -adic integers. The point is that in any experiment with the slime mould we deal with attractants which can be placed differently to obtain different topologies and to induce different transitions of the slime mould. If the set A of attractants, involved into the experiment, has the cardinality number $p - 1$, then any subset of A can be regarded as a condition for the experiment such as “Attractants occupied by the plasmodium”. These conditions change during the time, $t = 0, 1, 2, \dots$, and for the infinite time, we obtain p -adic integers as values of fuzzy (probability) measures defined on conditions (properties) of the experiment. This space is a semantics for p -adic valued fuzzy syllogistics we constructed for describing the propagation of the slime mould [A18]. This syllogistics can be extended to a *p*-adic valued logic and *p*-adic valued arithmetics [A15]. Within this logic we can develop a context-based game theory [P13]. All these logical tools can be implemented on plasmodia by conventional and unconventional reversible logic gates [P12], [A19].

We have proposed to use p -adic valued fuzziness and probabilities for measuring behaviours which cannot be measured additively. Then we constructed a natural deductive system for describing all possible experiments with *P. polycephalum*. This system is p -adic many-valued [A18]. We considered possibilities for applying p -adic valued logic BL to the task of designing the *Physarum* chip. If it is assumed that at any time step t of propagation the slime mould can discover and reach not more than $p-1$ attractants, then this behaviour can be coded in terms of p -adic numbers. As a result, this behaviour implements some p -adic valued arithmetic circuits and can verify p -adic valued logical propositions [A18].

We have offered two unconventional arithmetic circuits: adder and subtractor defined on finite p -adic integers [B4]. Adder and subtractor are designed by means of spatial configurations of several attractants and repellents which are stimuli for the plasmodium behaviour. As a result, the plasmodium could form a network of protoplasmic veins connecting attractants and original points of the plasmodium. Occupying new attractants is considered in the way of adders and leaving some attractants because of repelling is considered in the way of subtractors. On the basis of p -adic adders and subtractors we can design complex p -adic valued arithmetic circuits within a p -adic valued logic proposed by us [B4].

So, p -adic valued logic and p -adic valued arithmetic are implementable on plasmodia.

3.5. *Non-well-founded formal theories*

To formalize *Physarum* computing we define a kind of simple actions of labelled transition systems which cannot be atomic; consequently, their compositions cannot be inductive [A13]. Their informal meaning is that in one simple action we can suppose the maximum of its modifications. Such actions are called *hybrid*. Then we propose two formal theories on hybrid actions (the hybrid actions are defined there as non-well-founded terms and non-well-founded formulas): group theory and Boolean algebra [A13]. Both theories possess many unusual properties such as the following one: the same member of this group theory behaves as multiplicative zero in respect to some members and as multiplicative unit in respect to other members [A13]. The group theory proposed by us can be used as the new design method to construct reversible logic gates on plasmodia [P11]. In this way, we should appeal to the so-called non-linear permutation groups. These groups contain non-well-founded objects such as infinite streams and their families.

The theory of non-linear permutation groups proposed by us can be used for designing reversible logic gates on any behavioural systems [P11]. The simple versions of these gates are represented by logic circuits constructed on the basis of the performative syllogistic [A17]. It seems to be natural for behavioural systems and these circuits have very high accuracy in implementing. Our general motivation in designing logic circuits in behavioural systems without repellents is as follows: in this way, we can present behavioural systems as a calculation process more naturally; we can design devices, where there are much more outputs than inputs, for performing massive-parallel computations in the bio-inspired way; we can obtain unconventional (co)algorithms by programming behavioural systems.

Hence, to formalize *Physarum* computing we propose new formal theories such as theory of non-linear permutation groups to design unconventional reversible logic gates [P11].

3.6. Reversible logic gates

We considered different ways of designing reversible logic gates on *P. polycephalum* motions using controlling stimuli such as attractants and repellents [A19]. Repellents are needed because of uncertainty in the direction of plasmodium propagation to eliminate some directions as unimportant. In this way, we can construct conventional reversible logic gates: the CNOT gate, the FREDKIN gate, the TOFFOLI gate, etc. Combinations of reversible logic gates are regarded as matrix multiplications. Nevertheless, the plasmodium in its networking can permanently change its decisions and without repellents we have an extension of matrix multiplication group theory. Within this extension we can design unconventional reversible logic gates, where the number of inputs and outputs is uncertain [A19]. For designing logic gates we have proposed to use Petri net models that can be treated as a high-level description [P20]. Petri net models enable us to reflect propagation of protoplasmic veins of the plasmodium in consecutive time instants (step by step).

3.7. Actin filament networks

We have proposed actin filament networks where inputs are different stresses and outputs are formations and destructions of filaments, on the one hand, and as assemblies and dis-assemblies of actin filament networks, on the other hand [P24]. Hence, under different external conditions we observe dynamic changes in the length of actin filaments and in the outlook of filament networks. As we see, the main difference of actin filament networks from others including neural networks is that the topology of actin filament networks changes in responses to dynamics of external stimuli. Some new filaments/processors can appear in one conditions and they can disappear in other conditions.

3.8. Solving some computation problems

By modelling the plasmodium behaviour in the Physarum chip we can simulate some patterns of collective intelligent behaviours of animal or insect groups: flocks of birds, colonies of ants, schools of fish, swarms of bees, etc. for which there are ever emergent patterns which cannot be reduced to a linear composition of elementary subsystems properly [B22], [B5]. In swarm intelligence the Travelling Salesman Problem can be solved: more shorter distance between cities (pieces of food for the plasmodium), more attracting they are, as well as the Generalized Assignment Problem can be solved: the tubes of the plasmodium are regarded as agents, the nutrient sources as tasks, the amount of nutrient as profit, and the distance as cost [B22], [B5]. We show that by using p-adic integers we can code different emergent patterns so that the implementation of some unconventional algorithms of p-adic arithmetics and logic can be more applicable than conventional automata [B5].

4. Game-theoretic interface for *Physarum* storage modification machine

4.1. Chemical interface

In moving, the plasmodium switches its direction or even multiplies in accordance with different bio signals attracting or repelling its motions, e.g. in accordance with pheromones of bacterial food, which attract the plasmodium, and high salt concentrations, which repel it. So, the plasmodium motions can be controlled by different topologies of attractants and repellents so that the plasmodium can be considered a programmable biological device in the form of a timed transition system, where attractants and repellents determine the set of all plasmodium transitions [P14]. Furthermore, we can define p-adic probabilities on these transitions and, using them, we can define a knowledge state of plasmodium and its game strategy in occupying attractants as payoffs for the plasmodium [P14].

We can regard the task of controlling the plasmodium motions as a game and we can design different interfaces in a game-theoretic setting for the controllers of plasmodium transitions by chemical signals [P14].

4.2. Bio-inspired game theory

We have proposed a bio-inspired game theory on plasmodia, i.e. an experimental game theory, where, on the one hand, all basic definitions are verified in the experiments with *Physarum polycephalum* and *Badhamia utricularis* and, on the other hand, all basic algorithms are implemented in the object-oriented language for simulations of plasmodia. Our results allow us to claim that the slime mould can be a model for concurrent games and context-based games. In context-based games, players can move concurrently as well as in concurrent games, but the set of actions is ever infinite. In our experiments, we follow the following interpretations of basic entities:

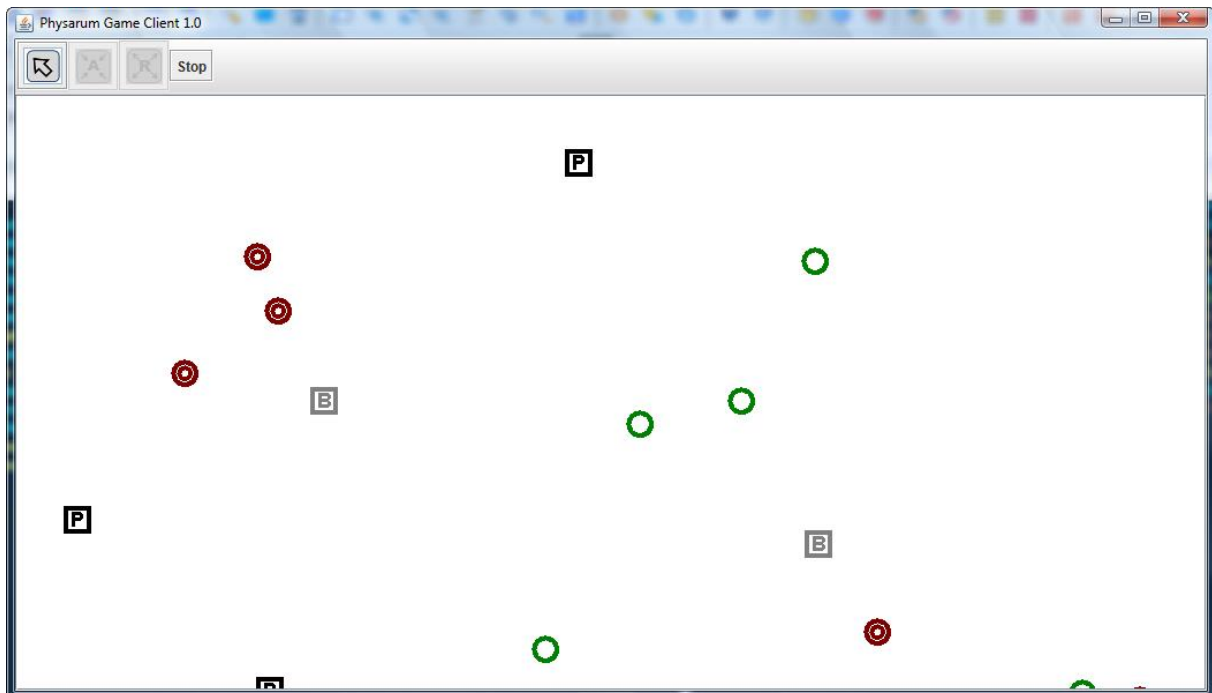
- Attractants as payoffs;
- Attractants occupied by the plasmodium as states of the game;
- Active zones of plasmodium as players;
- Logic gates for behaviours as moves (available actions) for the players;
- Propagation of the plasmodium as the transition table which associates, with a given set of states and a given move of the players, the set of states resulting from that move.

In the *Physarum* game theory we can demonstrate creativity of primitive biological substrates of plasmodia. The point is that plasmodia do not strictly follow spatial algorithms like Kolmogorov-Uspensky machines, but perform many additional actions. So, the plasmodium behaviour can be formalized within strong extensions of spatial algorithms, e.g. within concurrent games or context-based games.

4.3. Zero-sum games

We have proposed a game-theoretic visualization of morphological dynamics with non-symbolic interfaces between living objects and humans. These non-symbolic interfaces are more general than just sonification and have a game-theoretic form. The user interface for this game is designed on the basis of the following game steps: first, the system of *Physarum* language generates locations of attractants and repellents; second, we can chose n plasmodia/agents of *Physarum polycephalum* and m plasmodia/agents of *Badhamia utricularis*; third, we obtain the task, for example to reach as many as possible attractants or to construct the longest path consisting of occupied attractants, etc.; fourth, we can chose initial points for *Physarum polycephalum* transitions and initial points for *Badhamia utricularis* transitions; fifth, we start to move step by step; sixth, we define who wins, either *Physarum polycephalum* or *Badhamia utricularis*, see Fig. 10.

Figure 10. Game theoretic visualization of morphological dynamics.



Hence, we propose a game-theoretic interface from *Physarum* to humans.

Within this interface we have designed the zero-sum game between plasmodia of *Physarum polycephalum* and *Badhamia utricularis*, the so-called PHY game [P14], [B6]. To simulate *Physarum* games, we have created a specialized software tool. It is a part of the larger platform developed, using the Java environment. The tool works under the client-server paradigm. The server window contains:

- a text area with information about actions undertaken,
- a combobox for selecting one of two defined situations,
- start and stop server buttons.

Communication between clients and the server is realized through text messages containing statements of our object-oriented programming language, called the *Physarum* language. The locations of attractants and repellents are determined by the players during the game. At the beginning, origin points of *Physarum polycephalum* and *Badhamia utricularis* are scattered randomly on the plane. During the game, players can place stimuli. New veins of plasmodia are created. The server sends to clients information about the current configuration of the *Physarum* machine (localization of origin points of *Physarum polycephalum* and *Badhamia utricularis*, localization of stimuli as well as a list of edges, corresponding to veins of plasmodia, between active points) through the XML file [B6].

The game designed by us shows aesthetic aspects of plasmodium patterns and can be attractive for the Internet users. We assume that bio-inspired games wake new interests in designing new games and new game platforms.

4.4. Go games

We have simulated the motions of *P. polycephalum* plasmodium by the game of Go [P18], [B3]. We have considered two syllogistic systems implemented as Go games: the Aristotelian syllogistic and performative syllogistic. In the Aristotelian syllogistic, the locations of black and white stones are understood as locations of attractants and repellents, respectively. In the performative syllogistic, we consider the locations of black stones as locations of attractants occupied by plasmodia of *P. polycephalum* and the locations of white stones as locations of attractants occupied by plasmodia of *Badhamia utricularis*. The Aristotelian syllogistic version of Go game is a coalition game. The performative syllogistic version of Go game is an antagonistic game. We described selected functionality of the current version of a new software tool, called *PhysarumSoft*, developed for programming *Physarum* machines and simulating *Physarum* games [P19]. The tool was designed for the Java platform. We proposed a rough set approach for description of a strategy game created on the *Physarum* machine. The strategies of such a game are approximated on the basis of a rough set model, describing behaviour of the *Physarum* machine, created according to the VPRSM (Variable Precision Rough Set Model) approach [P16], [P21], [P23], [B6].

4.5. Neural properties of slime mould

There are two main properties of neural networks: lateral activation and lateral inhibition. The same properties are observed in *Physarum polycephalum* networks. We have generalized our studies of lateral inhibition effects in *P. polycephalum* behaviour in the way of constructing new syllogistics and modal logics [A20]. So, we have shown that there are two main possibilities of pairwise comparisons analysis in computer science: first, pairwise comparisons within a lattice, in this case these comparisons can be measurable by numbers (this one corresponds to lateral inhibition); second, comparisons beyond any lattice, in this case these comparisons cannot be measurable in principle (this one corresponds to lateral activation of neural networks). We have shown that the first approach to pairwise comparisons analysis is based on the conventional square of opposition and its generalization, but the second approach is based on unconventional squares of opposition [A20].

Furthermore, the first approach corresponds to lateral inhibition in transmission signals and the second approach corresponds to lateral activation in transmission signals. For combining lateral inhibition and lateral activation in the same behaviour we introduced the notion of the so-called context-based games to describe rationality of the slime mould [P15]. In these games we assume that, first, strategies can change permanently, second, players cannot be defined as individuals performing just one action at each time step. They can perform many actions simultaneously.

5. Published works

Monographs:

- [M1] **Schumann, A.:** *Unconventional Logics in Decision Making*, University of Information Technology and Management, Rzeszow 2014.

Proceedings:

- [P1] **Schumann, A.:** *Unconventional logic for massively parallel reasoning*, Human System Interaction (HSI), The 6th International Conference on 6-8 June 2013, IEEE Xplore, 2013, pp. 298–305, DOI 10.1109/HSI.2013.6577839.
- [P2] **Schumann, A., Pancerz, K.:** *Towards an Object-Oriented Programming Language for Physarum Polycephalum Computing*, [in:] Szczuka, M., Czaja, L., Kacprzak, M. (eds.): *Proceedings of the Workshop on Concurrency, Specification and Programming (CS&P)*, September 25–27, pp. 389–397, Warsaw - Poland 2013.
- [P3] **Schumann, A., Akimova, L.:** *Simulating of Schistosomatidae (Trematoda: Digenea) Behaviour by Physarum Spatial Logic*, Annals of Computer Science and Information Systems, vol.1. Proceedings of the Federated Conference on Computer Science and Information Systems. IEEE Xplore, pp. 225–2301, 2013.
- [P4] **Schumann, A., Pancerz, K., Jones, J.:** *Towards Logic Circuits Based on Physarum Polycephalum Machines: the Ladder Diagram Approach*, [in:] Cliquet, A., Plantier, G., Schultz, T., Fred, A., Gamboa, H. (eds.): *Proceedings of the 7th International Conference on Biomedical Electronics and Devices (BIODEVICES'2014)*, Angers, March 3–6, pp. 165–170, France 2014.
- [P5] **Schumann, A.:** *Game-Theoretic Aspects of Illocutionary Logic*, [in:] *The Eighth Smirnov's Readings: Proceedings of International Scientific Conference*, 19–21 June, pp. 86–88, Moscow 2013.
- [P6] **Schumann, A., Pancerz, K., Adamatzky, A., Grube, M.:** *Context-Based Games and Physarum Polycephalum as Simulation Model*, [in:] *Workshop Unconventional Computation in Europe*, Unconventional Computation & Natural Computation, University of Western Ontario, Ontario - Canada, July 14–18, London 2014.
- [P7] **Schumann, A.:** *Physarum Syllogistic L-Systems*, [in:] *Future Computing 2014, The Sixth International Conference on Future Computational Technologies and Applications*, 2014.
- [P8] **Pancerz, K., Schumann, A.:** *Principles of an Object-Oriented Programming Language for Physarum Polycephalum Computing*, [in:] *Proceedings of the 10th International Conference on Digital Technologies (DT'2014)*, Zilina, July 9–11, pp. 273–280, Slovakia 2014.
- [P9] **Schumann, A., Pancerz, K.:** *Timed Transition System Models for Programming Physarum Machines: Extended Abstract*, [in:] Popova-Zeugmann, L. (ed.): *Proceedings of the Workshop on Concurrency, Specification and Programming (CS&P)*, Chemnitz, September 29 – October 1, pp. 180–183, Germany 2014.
- [P10] **Schumann, A.:** *p-Adic Valued Fuzzyness and Experiments with Physarum Polycephalum*, FSKD 2014, IEEE Xplore, pp. 466–472, 19–21 Aug. 2014.
- [P11] **Schumann, A.:** *Non-Linear Permutation Groups on Physarum Polycephalum*, ICSAI 2014, IEEE Xplore, pp. 246 – 251, 15–17 Nov. 2014.

- [P12] **Schumann, A.:** *Reversible Logic Gates on Physarum Polycephalum*, International Conference of Numerical Analysis and Applied Mathematics, September 22–28, Special Volume of the AIP Conference Proceedings 2014.
- [P13] **Schumann, A., Pancerz, K., Adamatzky, A., Grube, M.:** *Bio-Inspired Game Theory: The Case of Physarum Polycephalum*, ACM, BICT, 1–3 Dec. 2014.
- [P14] **Schumann, A., Pancerz, K.:** *Interfaces in a Game-Theoretic Setting for Controlling the Plasmodium Motions*, BIOSTEC (Biosignals), Scitepress 2014.
- [P15] **Schumann, A.:** *Rationality in the Behaviour of Slime Moulds and the Individual-Collective Duality*, AISB Convention 2015.
- [P16] **Schumann, A.:** *Bayesian Networks on Transition Systems*, 5th World Congress and School on Universal Logic, June 20–30, 2015.
- [P17] **Schumann, A.:** *p-Adic Mathematics in Behavioural Models. The Case of Physarum Polycephalum*, International Conference on p-adic Mathematical physics and its applications, p-ADICS, 07–12.09, Belgrade, Serbia 2015.
- [P18] **Schumann, A.:** *Go Games on Plasmodia of Physarum Polycephalum*, *Annals of Computer Science and Information Systems*, Proceedings of the Federated Conference on Computer Science and Information Systems, IEEE Xplore, pp. 607–614, 2015.
- [P19] **Schumann, A., Pancerz, K.:** *PhysarumSoft – a Software Tool for Programming Physarum Machines and Simulating Physarum Games*, *Annals of Computer Science and Information Systems*. Proceedings of the Federated Conference on Computer Science and Information Systems, IEEE Xplore, pp. 615–626, 2015.
- [P20] **Schumann, A., Pancerz, K.:** *Petri Net Models of Simple Rule-Based Systems for Programming Physarum Machines*, [in:] Suraj, Z., Czaja, L. (eds.): *Proceedings of the 24th International Workshop on Concurrency, Specification and Programming*, CEUR Workshop Proceedings 1492, CEUR-WS.org, September 28–30, pp. 155–160, Rzeszow - Poland 2015.
- [P21] **Schumann, A., Pancerz, K.:** *Roughness in Timed Transition Systems Modeling Propagation of Plasmodium*, [in:] Ciucci, D., Wang, G., Mitra, S., Wei-Zhi, W. (eds.): *Rough Sets and Knowledge Technology - 10th International Conference, RSKT, held as part of the International Joint Conference on Rough Sets, IJCRS, November 20-23, Proceedings*, Lecture Notes in Computer Science 9436, Springer, pp. 482–491, Tianjin - China 2015.
- [P22] **Schumann, A.:** *From Swarm Simulations to Swarm Intelligence*, 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), ACM, 2015 (accepted).
- [P23] **Pancerz, K., Schumann, A.:** *A Rough Set Version of the Go Game on Physarum Machines*, 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS), ACM, 2015 (accepted).
- [P24] **Schumann, A.:** *Toward A Computational Model of Actin Filament Networks*, Proceedings of the International Conference on Bio-inspired Systems and Signal Processing, SCITEPRESS, 2016 (accepted).

Articles in Journals:

- [A1] 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), pp. 319–323, 2012.
- [A2] Schumann, A.: *On Two Squares of Opposition: the Leśniewski’s Style Formalization of Synthetic Propositions*, “Acta Analytica”, 28, pp. 71–93, 2013, DOI10.1007/s12136-012-0162-4.
- [A3] Schumann, A.: *Anti-Platonic Logic and Mathematics*, “Journal of Multiple-Valued Logic and Soft Computing”, 21(1–2), pp. 53–88, 2013.
- [A4] Schumann, A., Adamatzky, A.: *Logical Modelling of Physarum Polycephalum*, Analele Universitatii de Vest, Timisoara, Seria Matematica – Informatica XLVIII, 3, pp. 175–190, 2010.
- [A5] Schumann, A., Adamatzky, A.: *Physarum Spatial Logic*, New Math. and Nat. Computation, vol. 7, no. 3, pp. 483–498, 2011.
- [A6] Schumann, A.: *Proof-Theoretic Cellular Automata as Logic of Unconventional Computing*, “International Journal of Unconventional Computing”, vol. 8, no. 3, pp. 263–280, 2012.
- [A7] Schumann, A., Pancerz, K.: *Towards an Object-Oriented Programming Language for Physarum Polycephalum Computing: A Petri Net Model Approach*, “Fundamenta Informaticae”, vol. 133, no. 2–3, pp. 271–285, 2014.
- [A8] Schumann, A.: *Reflexive Games and Non-Archimedean Probabilities, p-Adic Numbers*, “Ultrametric Analysis and Applications”, vol. 6, issue 1, pp. 66–79, January 2014.
- [A9] Schumann, A.: *Probabilities on Streams and Reflexive Games*, “Operations Research and Decisions”, vol. 24, no. 1, pp. 71–96, 2014.
- [A10] Schumann, A.: *Payoff Cellular Automata and Reflexive Games*, “Journal of Cellular Automata”, 9 (4), pp. 287–313, 2014.
- [A11] Schumann, A., Khrennikov, A.: *Quantum Non-Objectivity from Performativity of Quantum Phenomena*, “Physica Scripta”, no. T163, pp. 15, 2014.
- [A12] Schumann, A.: *Towards Slime Mould Based Computer*, “New Mathematics and Natural Computation”, (accepted).
- [A13] Schumann, A.: *Towards context-based concurrent formal theories*, “Parallel Processing Letters”, 25, 2015, 1540008.
- [A14] Schumann, A., Akimova, L.: *Syllogistic System for the Propagation of Parasites. The Case of Schistosomatidae (Trematoda: Digenea)*, “Studies in Logic, Grammar and Rhetoric”, 40 (1), pp. 303–319, 2015.
- [A15] Schumann, A., Adamatzky, A.: *The Double-Slit Experiment with Physarum Polycephalum and p-Adic Valued Probabilities and Fuzziness*, “International Journal of General Systems”, 27 Jan 2015, DOI: 10.1080/03081079.2014.997530.
- [A16] Pancerz, K., Schumann, A.: *Rough Set Models of Physarum Machines*, “International Journal of General Systems”, 27 Jan 2015, DOI: 10.1080/03081079.2014.997529.
- [A17] Schumann, A., Adamatzky, A.: *Physarum Polycephalum Diagrams for Syllogistic Systems*, “IfCoLog Journal of Logics and their Applications”, 2 (1), pp. 35–68, 2015.
- [A18] Schumann, A.: *p-Adic valued logical calculi in simulations of the slime mould behaviour*, “Journal of Applied Non-Classical Logics”, 2015, DOI: 10.1080/11663081.2015.1049099.
- [A19] Schumann, A.: *Conventional and unconventional reversible logic gates on Physarum polycephalum*, “International Journal of Parallel, Emergent and Distributed Systems”, 2015, DOI: 10.1080/17445760.2015.1068775.
- [A20] Schumann, A., Woleński, J.: *Two Squares of Oppositions and Their Applications in Pairwise Comparisons Analysis*, “Fundamenta Informaticae”, 2016, (accepted).

- [A21] **Schumann, A.:** *Physarum Polycephalum Syllogistic L-Systems and Judaic Roots of Unconventional Computing*, “Studies in Logic, Grammar and Rhetoric”, 44(57), pp. 181–201, 2016, DOI: 10.1515/slgr-2016-0011.
- [A22] **Schumann, A., Pancerz, K., Szelc, A.:** *The Swarm Computing Approach to Business Intelligence*, “Studia Humana”, 4 (3), pp. 41–50, 2015, DOI: 10.1515/sh-2015-0019.

Chapters in Books:

- [B1] **Schumann, A., Akimova, L.:** *Process Calculus and Illocutionary Logic for Analyzing the Behavior of Schistosomatidae (Trematoda: Digenea)*, [in:] Pancerz, K., Zaitseva, E. (eds.): *Computational Intelligence, Medicine and Biology - Selected Links*. Studies in Computational Intelligence 600, Springer 2015, ISBN 978-3-319-16843-2.
- [B2] **Pancerz, K., Schumann, A.:** *Some Issues on an Object-Oriented Programming Language for Physarum Machines*, [in:] Bris, R., Majernik, J., Pancerz, K., Zaitseva, E., (eds.): *Applications of Computational Intelligence in Biomedical Technology*. Studies in Computational Intelligence 606, Springer 2016, ISBN 978-3-319-19146-1.
- [B3] **Schumann, A.:** *Syllogistic Versions of Go Games on Physarum Polycephalum*, [in:] Adamatzky, A. (ed.): *Advances in Physarum Machines*, vol.21 of the series Emergence, Complexity and Computation, Springer 2016, pp. 651-685, ISBN 978-3-319-26662-6.
- [B4] **Schumann, A., Pancerz, K.:** *p-Adic Computation with Physarum*, [in:] Adamatzky, A. (ed.): *Advances in Physarum Machines*, vol.21 of the series Emergence, Complexity and Computation, Springer, pp. 619-649, 2016, ISBN 978-3-319-26662-6.
- [B5] **Schumann, A.:** *Conventional and Unconventional Approaches to Swarm Logic*, [in:] Adamatzky, A. (ed.): *Advances in Unconventional Computing*, Springer 2016, (accepted).
- [B6] **Pancerz, K., Schumann, A.:** *Rough Set Description of Strategy Games on Physarum Machines*, [in:] Adamatzky, A. (ed.): *Advances in Unconventional Computing*, Springer 2016, (accepted).

About the project in media:

Many-Valued Logics and Slime Moulds.

<http://richardzach.org/2015/06/16/many-valued-logics-and-slime-moulds/>

Defining games on the medium of a one-cell organism.

<http://blog.eai.eu/defining-games-on-the-medium-of-a-one-cell-organism/>

Awards:

Best paper: *Bio-Inspired Game Theory: The Case of Physarum Polycephalum*

<http://bionetics.org/2014/show/exhibit-and-demo>

Our workshops:

1st International Workshop on Biological, Chemical and Physical Computations (BCPC'15).

CFP: BICT 2015 Special Track on Biological Computing and Bio-medical Informatics (BCBI).

Organizing committees:

9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS).

The Seventh International Conference on Future Computational Technologies and Applications, FUTURE COMPUTING 2015.

Part II. Logics of *Physarum* Machines.

Materials from Conference Presentations. Selected Slides

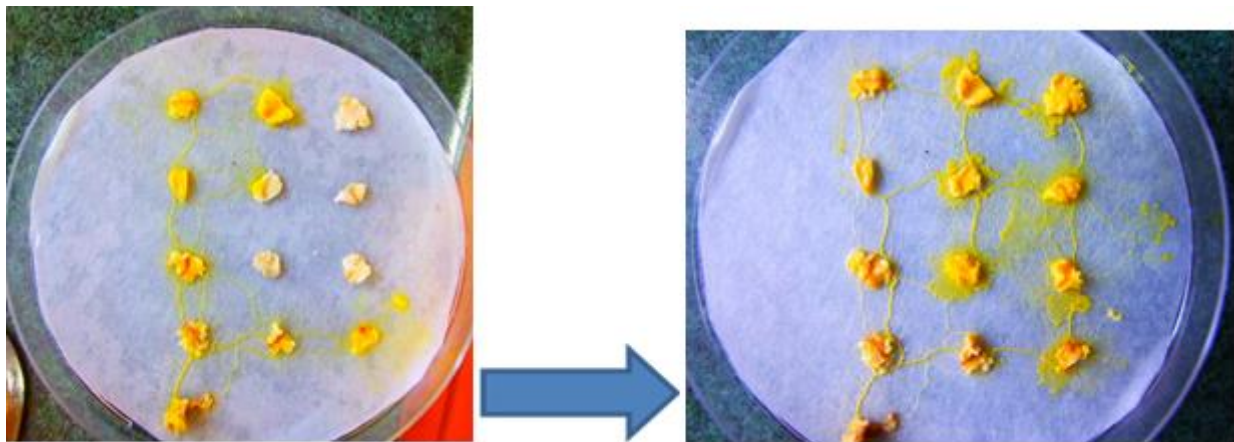
1. Context-based games

1.1. *Plasmodium transition system*

The life cycle of *Physarum polycephalum* consists of the following phases: (i) The plasmodium (networks of protoplasmic veins) as the main vegetative phase, when *Physarum* searches for food by propagating protoplasmic veins. When it finds food, it secretes enzymes to digest it. (ii) The sclerotium as a dormant phase, protecting *Physarum* for long periods of time, when environmental conditions cause the plasmodium to desiccate during feeding. If favourable conditions resume, the plasmodium reappears again and searches for food. (iii) Stalks of sporangia form from the plasmodium as a reproductive phase, when meiosis occurs and spores are formed. This phase begins in case there is not enough food for *Physarum*. (iv) Amoeboid swarm cells as a motile phase, when the spores germinate. The swarm cells fuse together to form a new plasmodium. And the cycle continues.

In computer science, the following fact was taken into account: the plasmodium spread by networks can be programmable and thereby it may simulate different simple engineering tasks such as designing transport systems. Notice that the *Physarum* motions are a kind of *natural transition systems*, $\langle \text{States}, \text{Edg} \rangle$, where States is a set of states presented by attractants and $\text{Edg} \subseteq \text{States} \times \text{States}$ is a transition of plasmodium from one attractant to another. The point is that the plasmodium looks for attractants, propagates protoplasmic tubes towards them, feeds on them and goes on. As a result, a transition system is being built, see Fig. 11.

Figure 11. Plasmodium transition system. *Source*: Schumann, A., Adamatzky, A.: *Physarum Polyccephalum Diagrams for Syllogistic Systems*, “IfCoLog Journal of Logics and their Applications”, 2 (1), 2015, pp. 35-68.



1.2. *Physarum spatial logic*

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 are considered as nodes and protoplasmic tubes as edges. This fact allows us to claim that plasmodium behaviour can be regarded as a biological implementation of Kolmogorov-Uspensky machines. The modification of locations of nutrients (source foods) causes a storage modification of plasmodium.

Entities of *Physarum spatial logic*:

- The set of *attractants* $\{A_1, A_2, \dots\}$ are sources of nutrients, on which the plasmodium feeds. It is still subject of discussion how exactly plasmodium feels presence of attractants.
- The set of *repellents* $\{R_1, R_2, \dots\}$. Plasmodium of *Physarum* avoids light and some thermo- and salt-based conditions. Thus, domains of high illumination are repellents such that each repellent R is characterized by its position and intensity of illumination, or force of repelling.
- The set of protoplasmic tubes $\{C_1, C_2, \dots\}$. Typically, plasmodium spans sources of nutrients with protoplasmic tubes/veins. The plasmodium builds a planar graph, where nodes are sources of nutrients, e.g. oat flakes, and edges are protoplasmic tubes. $C(\alpha)$ means a diffusion of growing pseudopodia α .

Physarum spatial logic contains the following basic operators: *Nil* (inaction), \cdot (prefix), $|$ (cooperation), \backslash (hiding), $\&$ (reaction/fusion), $+$ (choice), a (constant or restriction to a stable state), $A(\cdot)$ (attraction), $R(\cdot)$ (repelling), $C(\cdot)$ (spreading/diffusion). Let $T = \{a, b, \dots\}$, the set of all actions (evidently, this set is finite), be considered as a set of names. A name refers to a communication link or channel. With every $a \in T$ we associate a complementary action \bar{a} . Let us suppose that a designates an input port and \bar{a} designates an output port. Any behaviour of *Physarum* is considered as outputs and any form of outside control and stimuli as appropriate inputs. For instance, T' is to be regarded just as the set of output ports and thereby $T \setminus T'$ contains ports that can be interpreted under different conditions as input ports or output ports. So, for each $X \in \{A_1, A_2, \dots\} \cup \{R_1, R_2, \dots\}$, we take that $X(\gamma) = \delta$ is a function from T' to T' such that X is a stimulus for γ and X makes an output $\delta \in T'$ along γ . Evidently that $X(\text{Nil}) = \text{Nil}$. Let $X(\gamma)$ denote an input and an output.

Some properties of the operations *Nil* (inaction), \cdot (prefix), $|$ (cooperation), \backslash (hiding), $\&$ (reaction/fusion), $+$ (choice), where P_1, P_2, P_3 are different processes of plasmodium:

$$(P_1 + P_2) + P_3 = P_1 + (P_2 + P_3)$$

$$P_1 + P_2 = P_2 + P_1$$

$$(P_1 + P_2) \backslash L = (P_1 \backslash L) + (P_2 \backslash L)$$

$$P + \text{Nil} = P$$

$$(P_1 \& P_2) \& P_3 = P_1 \& (P_2 \& P_3)$$

$$P + P = P$$

$$P_1 \& P_2 = P_2 \& P_1$$

$$P \& P = P$$

$$(P_1 \& P_2) \backslash L = (P_1 \backslash L) \& (P_2 \backslash L)$$

$$P \& \text{Nil} = \text{Nil}$$

$$P_1 \& (P_2 + P_3) = (P_1 \& P_2) + (P_1 \& P_3)$$

$$P_1 + (P_2 \& P_3) = (P_1 + P_2) \& (P_1 + P_3)$$

$$(P_1 | P_2) | P_3 = P_1 | (P_2 | P_3)$$

$$P_1 | P_2 = P_2 | P_1$$

$$P | \text{Nil} = P$$

$$\text{Nil} \backslash L = \text{Nil}$$

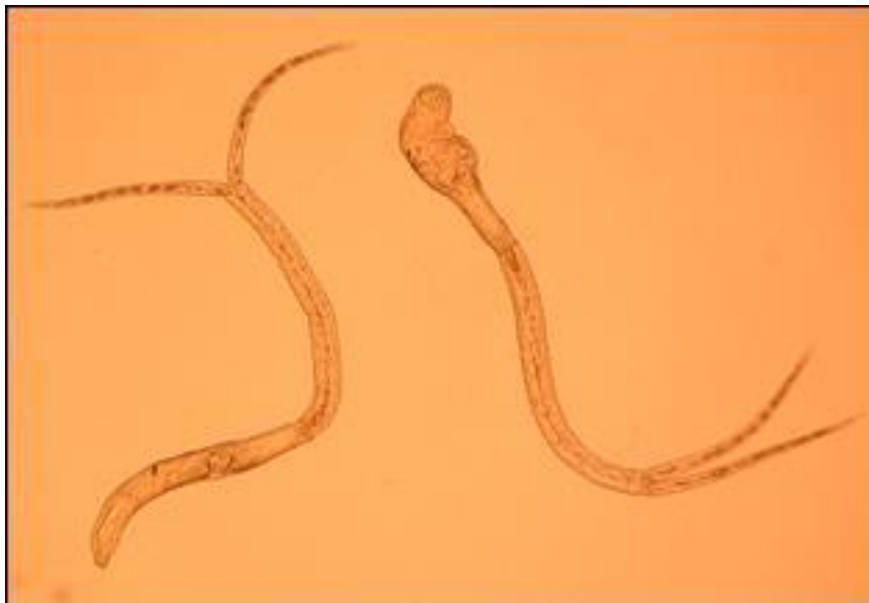
$$(P \backslash L_1) \backslash L_2 = P \backslash (L_1 \cup L_2)$$

1.3. Spatial logic for Schistosomatidae (Trematoda : Digenea)

All representatives of subclass *Digenea* Carus, 1863 (Platyhelminthes: *Trematoda*) are exclusively endoparasites of animals. The digenean life cycle has the form of heterogony, i.e. there is a natural alternation of amphimictic (usually synarmophytous) and parthenogenetic stages. At these stages digeneae have different outward, different means of reproduction and different adaptation to different hosts. The interchange of hosts is necessary for a successful realization of digenean flukes life cycle. The majority of representatives of this subclass have a complete life cycle with participation of three hosts: intermediate, additional (metacercarial) and definitive. Molluscs are always the first intermediate hosts, while different classes of vertebrate animals are definitive hosts.

Avian schistosomatidae have been observed on all continents, including Europe. In a pubertal state they parasitize birds, however they are capable to incorporate into a human organism as nonspecific host. After they penetrate human skin, where they perish, they invoke thereby allergic dermatitis. The fact of incorporation of these larvae into nonspecific hosts invokes interest to avian schistosomatidae, see Fig. 12. Therefore the simulation of behaviour of their local groups can be interesting from a medicine view, because it allows to perceive better features of digenean behaviour. Simulating their behavior is possible by means of *Physarum* spatial logic as we can show.

Figure 12. Schistosomatidae (Trematoda: Digenea). Source: Schumann, A., Akimova, L.: *Process Calculus and Illocutionary Logic for Analyzing the Behavior of Schistosomatidae (Trematoda: Digenea)*, [in:] Pancerz, K., Zaitseva, E. (eds.): *Computational Intelligence, Medicine and Biology - Selected Links*. Studies in Computational Intelligence 600, Springer 2015, ISBN 978-3-319-16843-2



The life cycle of all representatives of the family Schistosomatidae is identical, it passes with participation of two hosts: Miracidiae \Rightarrow molluscs as hosts, then: Cercariae \Rightarrow birds and humans as hosts

From an egg which got to water from an organism of definitive host, a miracidia hatches. It is a free-swimming larval stage of parthenogenetic generation of Schistosomatidae. Miracidia for a short span should find a mollusc of a certain kind to insinuate into it, otherwise it perishes. Molluscs, thus, are attractants for miracidiae. More precisely, in respect to miracidiae the chemotaxis as attractant holds (miracidia moves towards a chemical signal proceeding from a mollusc). Other kinds of attractants for miracidiae are presented by light (there is a positive phototaxis) and gravitation (negative geotaxis). We will designate all miracidian attractants by $A^m_1, A^m_2, \dots, A^m_n$. Miracidian repellents have not been detected still, i.e. $\{R^m_1, R^m_2, \dots, R^m_n\} = \emptyset$.

In a body of mollusc, a miracidia undergoes metamorphosis and it is transformed into a mother sporocyst in which daughter sporocysts educe. In the latter then cercariae start to be formed. This state can be called the miracidian diffusion. We will designate these diffusions by $C^m_1, C^m_2, \dots, C^m_n$.

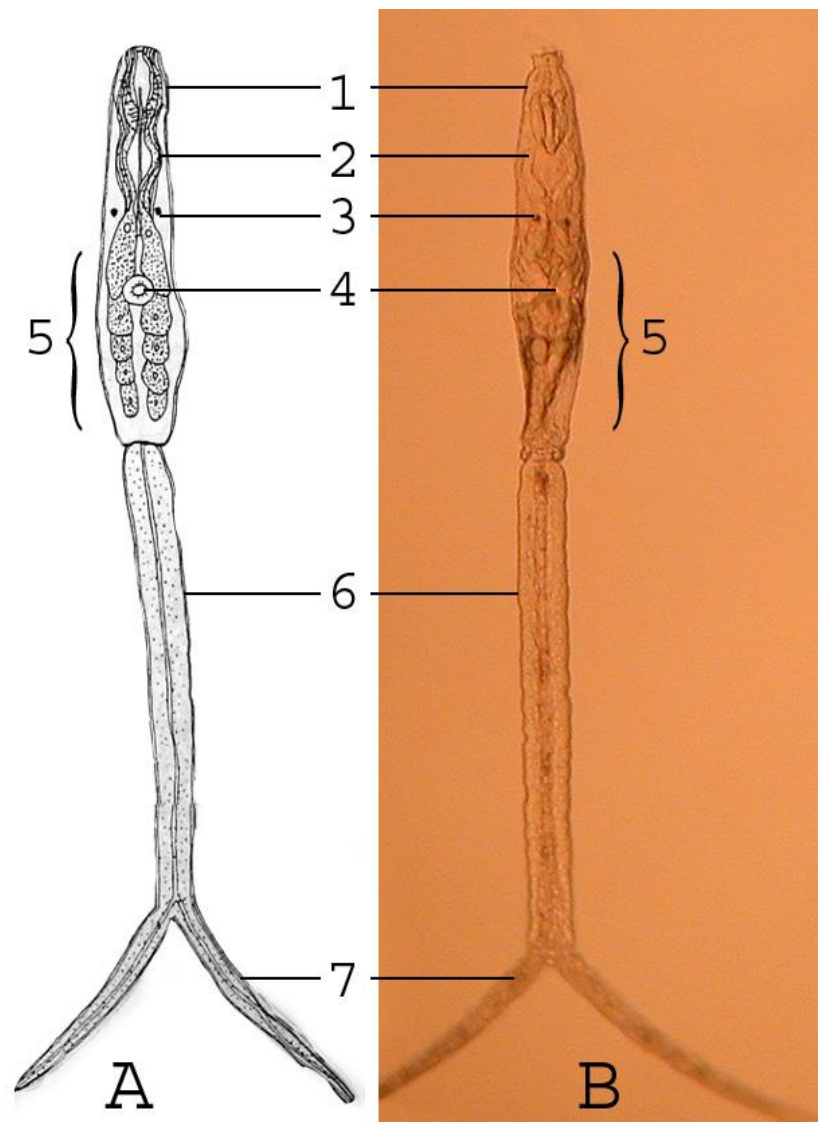
Now we can construct a version of *Physarum* spatial logic for simulating the behaviour of local groups of miracidiae. The processes have the following syntax which is defined in the way of *Physarum* logic:

$$P, Q ::= Nil \mid \gamma.P \mid A^m(\gamma).P \mid .P \mid \\ C^m(\gamma).P \mid (P \mid Q) \mid P \setminus Q \mid P \& Q \mid P + Q \mid a$$

For the simulation we need also to have two sets of actions T and T^m , where T contains actions of *Physarum* plasmodium, T^m includes actions of local group of miracidiae. These sets should have the same number of members (the same cardinality), namely we should have the same number of active zones (growing pseudopodia and active miracidiae), the same number of attractants, and the same number of diffusions (motions of protoplasmic tubes towards food and miracidian motions towards chemical signals of eventual hosts to transform into a maternal sporocyst). For instance, if we have five molluscs in one experimental dish with water and a suspension of miracidiae, then we can try to simulate the miracidian processes by *Physarum* spatial logic for stimuli of five nutrient sources with similar localizations as that for molluscs.

Cercariae are free-swimming larvae of pubertal generation parasitizing vertebrate animals. They are capable to insinuate into skin of human being who is for them a casual host, invoking at him/her an allergic reaction, the so-called cercarial dermatitis.

Figure 13. The constitution of *Trichobilharzia szidati*. A. The schematic structure of cercaria (from Ginetsinskaya, 1968), B. The photo of cercaria. 1. Penetration organ, 2. Penetration gland ducts, 3. Pigmented eye-spots, 4. Ventral sucker, 5. Penetration glands (5 pairs), 6. Tail stem, 7. Furcae. *Source*: Schumann, A., Akimova, L.: *Process Calculus and Illocutionary Logic for Analyzing the Behavior of Schistosomatidae (Trematoda: Digenea)*, [in:] Pancerz, K., Zaitseva, E. (eds.): *Computational Intelligence, Medicine and Biology - Selected Links*. Studies in Computational Intelligence 600, Springer 2015, ISBN 978-3-319-16843-2.



Cercariae actively react to changes in intensity of illumination (shadings) and to turbulence of water. These external factors, corresponding to possible appearances of definitive hosts in water, stimulate the cercarial transition from a resting state into actions that enlarge their chances to meet hosts. Cercariae possess a chemotaxis in relation to specific hosts. On body surfaces larvae of the genus *Trichobilharzia* have chemo-receptors which receive appropriate chemical signals proceeding from a skin of potential host. The similarity of compound of fatty acids of bird and human skin leads to that cercariae equally react to the bird and human appearance in water: they move in their direction, and then they are attached to skin and begin penetration into it. So, the chemotaxis from a skin of potential hosts (surface lipids of skin of human being and swimming bird), the positive phototaxis, the negative geotaxis and the water turbulence present cercarial attractants of different degree of appeal. We will designate these attractants by $A^c_1, A^c_2, \dots, A^c_n$. In some cases, for example children, cercarial larvae can “chip” skin and be brought by venous blood to lungs, invoking there haemorrhages and inflammation.

At the same time, repellents for cercariae have not been found yet. Cercariae simply freely float and as soon as they appear in that part where there is the reacting material they perish. Thus, $\{R^c_1, R^c_2, \dots, R^c_n\} = \emptyset$. In definitive hosts cercariae reach diffusion states. We will designate these diffusions by $C^c_1, C^c_2, \dots, C^c_n$.

The behaviour of local groups of cercariae can be simulated within a version of *Physarum* spatial logic, where the processes have the following syntax:

$$P, Q ::= Nil \mid \gamma.P \mid A^c(\gamma).P \mid .P \mid$$

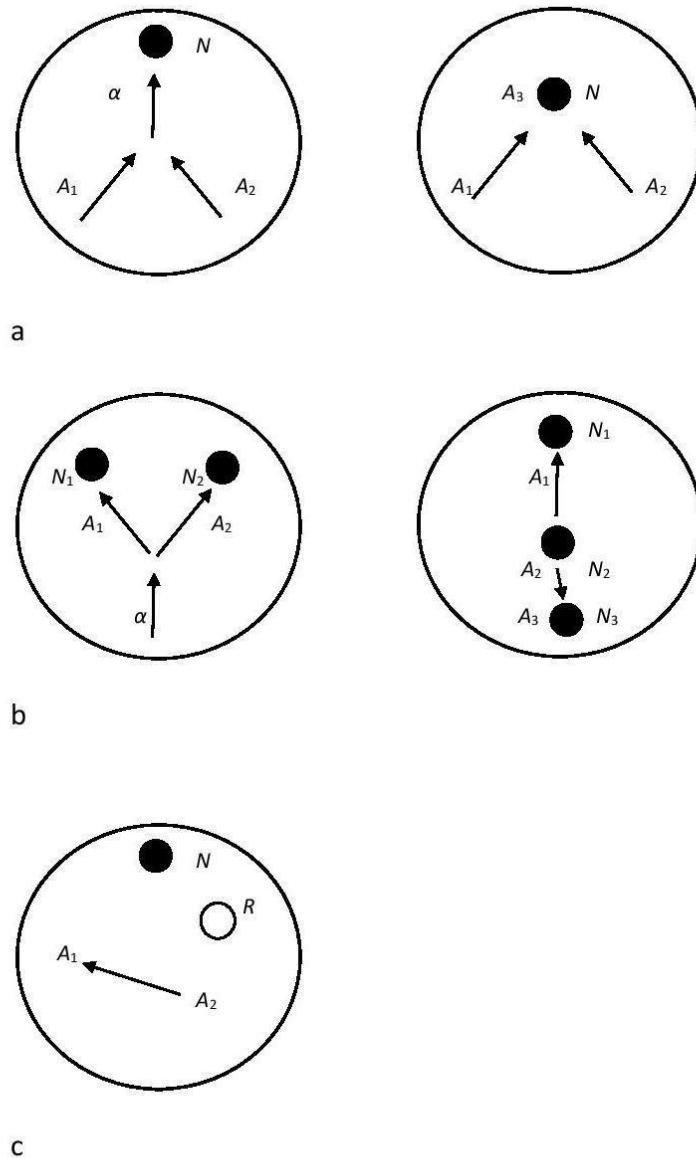
$$C^c(\gamma).P \mid (P \mid Q) \mid P \setminus Q \mid P \& Q \mid P + Q \mid a$$

The sets of actions T and T^c , where T consists of actions of *Physarum plasmodium*, T^c contains actions of local group of cercariae, should have the same number of members. For example, if we have three human beings in one lake with cercariae, then we can simulate the cercarial processes by *Physarum* spatial logic where three nutrient sources with similar localizations as that for human beings act as stimuli. Hence, the behaviour of local groups of cercariae is another biological implementation of Kolmogorov-Uspensky machines. It can build planar graphs as well.

1.3. Concurrent games

So, *Physarum polycephalum* motions are a kind of natural transition systems, $\langle \text{States}, \text{Edges} \rangle$, where States is a set of states presented by attractants and $\text{Edges} \subseteq \text{States} \times \text{States}$. Its basic transitions are pictured in Fig. 14.

Figure 14. Basic transitions of *Physarum polycephalum* plasmodia, where N is an attractant, R is a repellent, A is plasmodium. Operations: (a) Fusion; (b) Split; (c) Repellent.



Timed transition systems on plasmodia can be considered games within the so-called *concurrent game approach*. Thus, we can control the plasmodium motions as a game. As a consequence, user interfaces for the controllers of plasmodium propagations can have a natural form of game-theoretic setting.

For example, in the game *Physarum polycephalum* vs. *Badhamia utricularis*, we have two players (the first plays for the *Physarum polycephalum* plasmodia, the second for the *Badhamia utricularis* plasmodia). The game soft system places attractants and repellents automatically. Then the players choose which attractants are occupied before the game and which rules of the game hold (to reach as many as possible attractants or to construct the longest path consisting of occupied attractants, etc.). Then the system shows who wins and who loses.

It is known due to the experiments performed by Andrew Adamatzky and Martin Grube that if there are only two agents of the plasmodium game, where the first agent is presented by a usual *Physarum polycephalum* plasmodium and the second agent by its modification called a *Badhamia utricularis* plasmodium, then both start to compete with each other. In particular, the *Physarum polycephalum* plasmodium grows faster and could grow into branches of *Badhamia utricularis*, while the *Badhamia utricularis* plasmodium could grow over *Physarum polycephalum* veins, see Fig. 15, 16.

Figure 15. *Physarum polycephalum* (**Phys** for short) vs. *Badhamia utricularis* (**Phyx** for short). Source: Schumann, A., Pancerz, K., Adamatzky, A., Grube, M.: *Bio-Inspired. Game Theory: The Case of Physarum Polycephalum*, BICT 2014, 1-3 Dec. 2014, ACM.

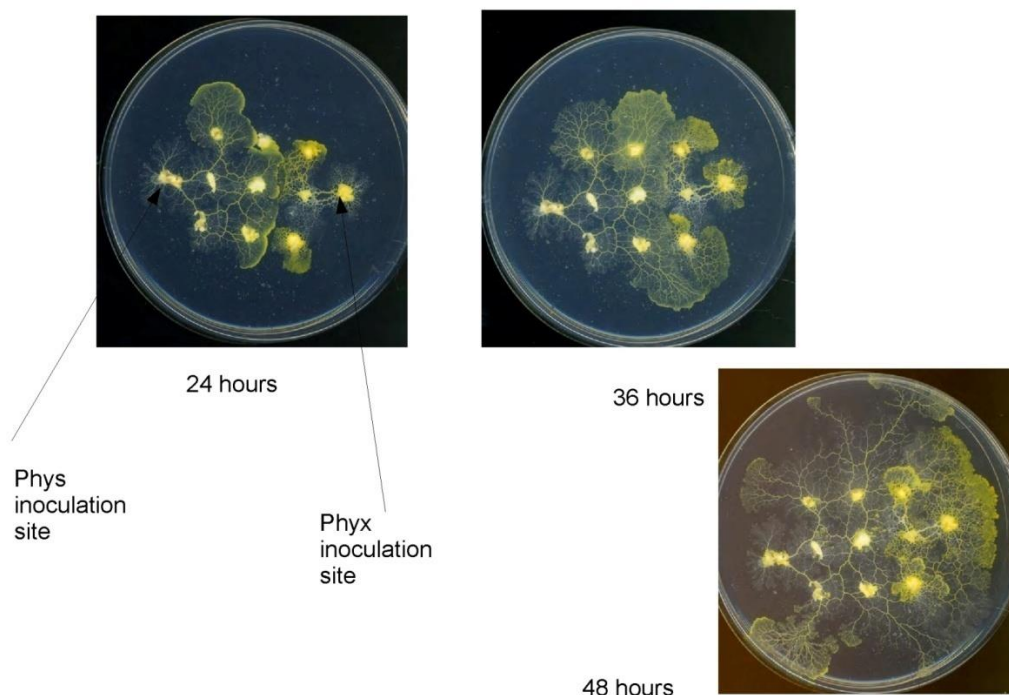
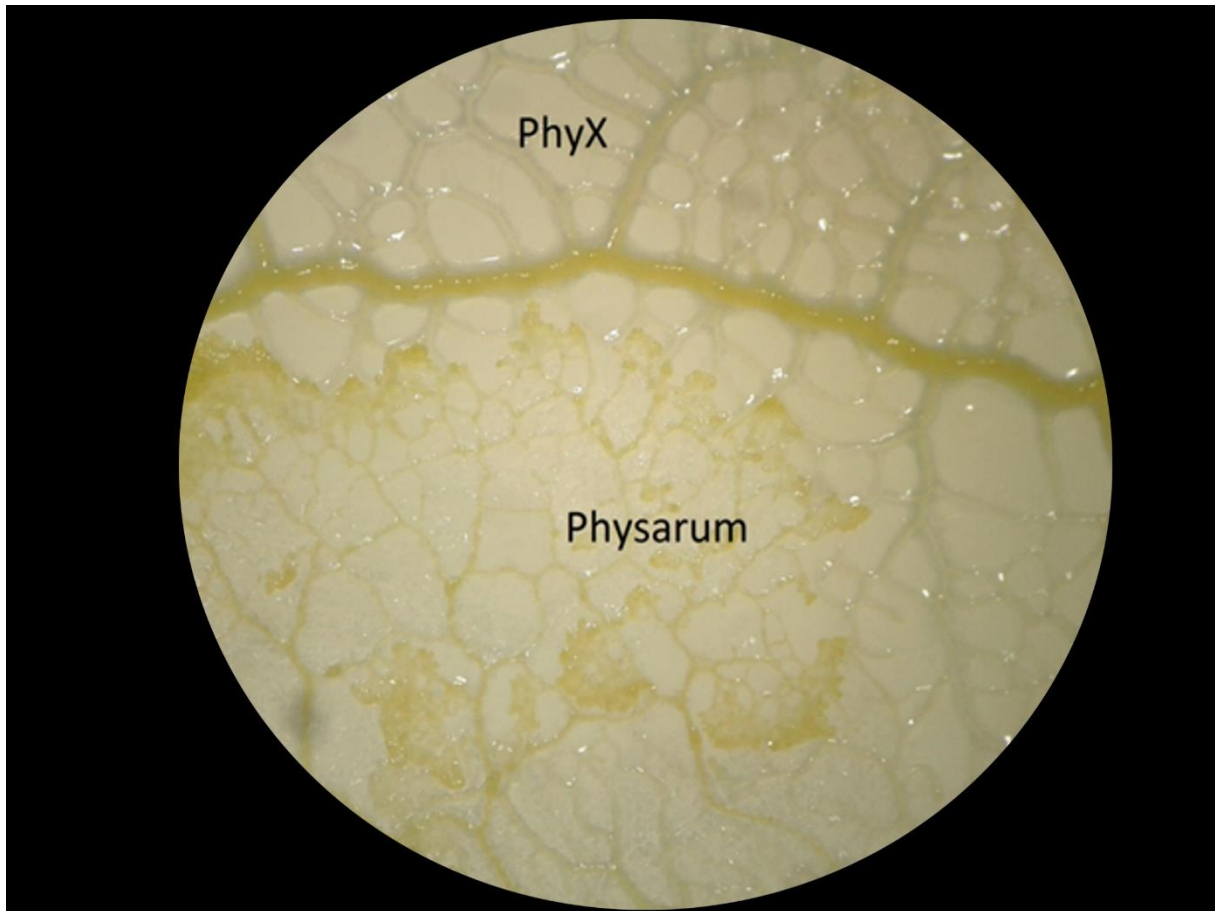


Figure 16. *Physarum polycephalum* (**Physarum** for short) vs. *Badhamia utricularis* (**PhyX** for short).
Source: Schumann, A., Pancerz, K., Adamatzky, A., Grube, M.: *Bio-Inspired. Game Theory: The Case of Physarum Polycephalum*, BICT 2014, 1-3 Dec. 2014, ACM.



Physarum polycephalum and *Badhamia utricularis* demonstrate an intelligent behaviour with intentionality and efficiency, although they do not have nervous systems at all. In particular, they demonstrate the ability to memorize and anticipate repeated events. Furthermore, by means of plasmodium behaviour, it is possible to simulate the behaviour of some collectives such as collectives of parasites. Thus, the complex intelligent behaviour of plasmodium is biologically unexplained still and shows the limits of our understanding what natural intelligence is.

The server window of our game soft for the game *Physarum polycephalum* vs. *Badhamia utricularis* contains, see Fig. 17 – 21:

1. text area with information about actions undertaken,
2. combobox for selecting one of the possible general game strategies:
 - by stimuli placement,
 - by plasmodium movement rules.
3. start and stop server buttons.

Figure 17. The server window of the game *Physarum polycephalum* vs. *Badhamia utricularis*.

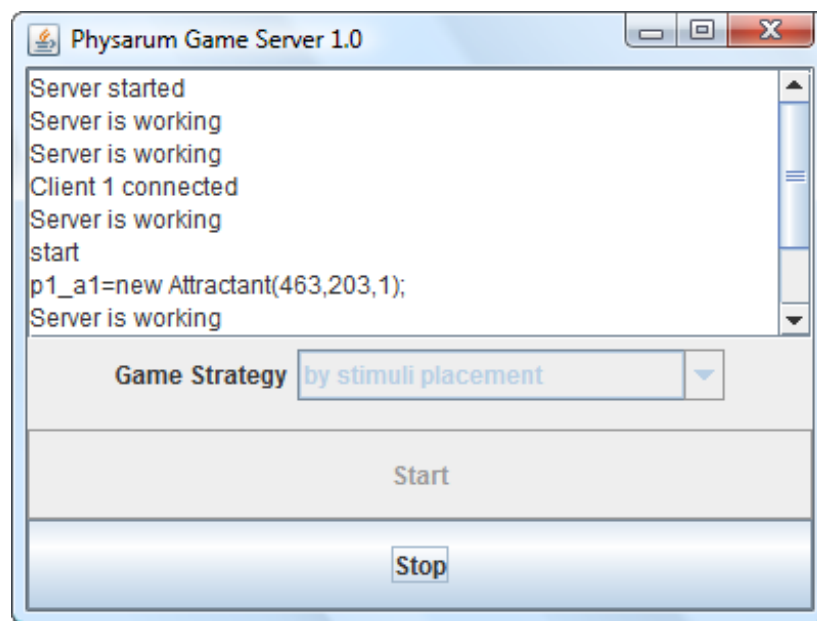


Figure 18. The initial client window at the first game strategy (by stimuli placement). At the beginning, *Physarum* and *Badhamia* are scattered randomly on the plane.

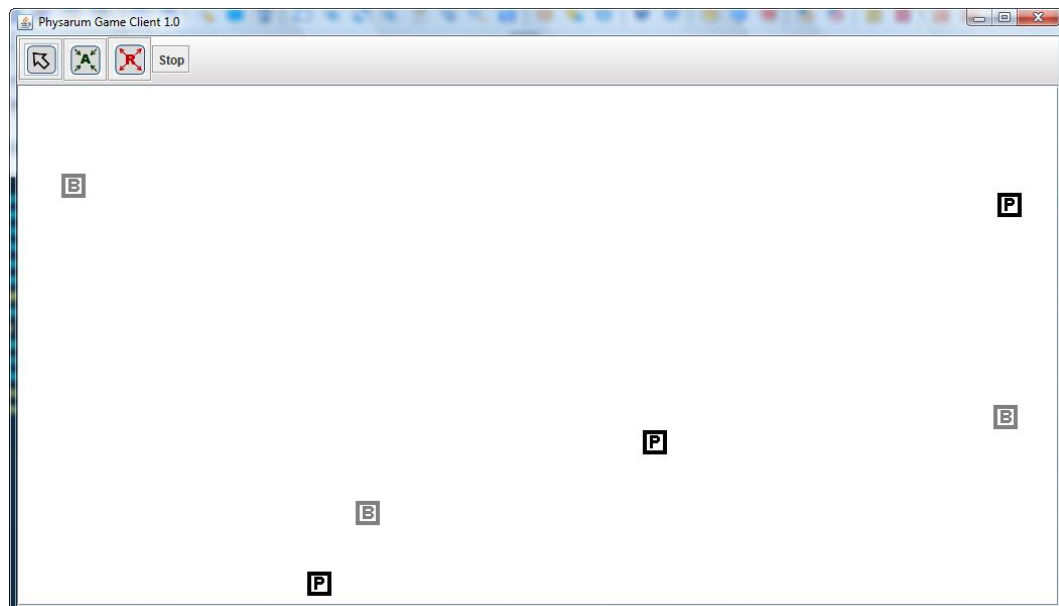


Figure 19. The client window at the first game strategy (by stimuli placement) after several player movements. The player can place stimuli. New veins of plasmodium are created.

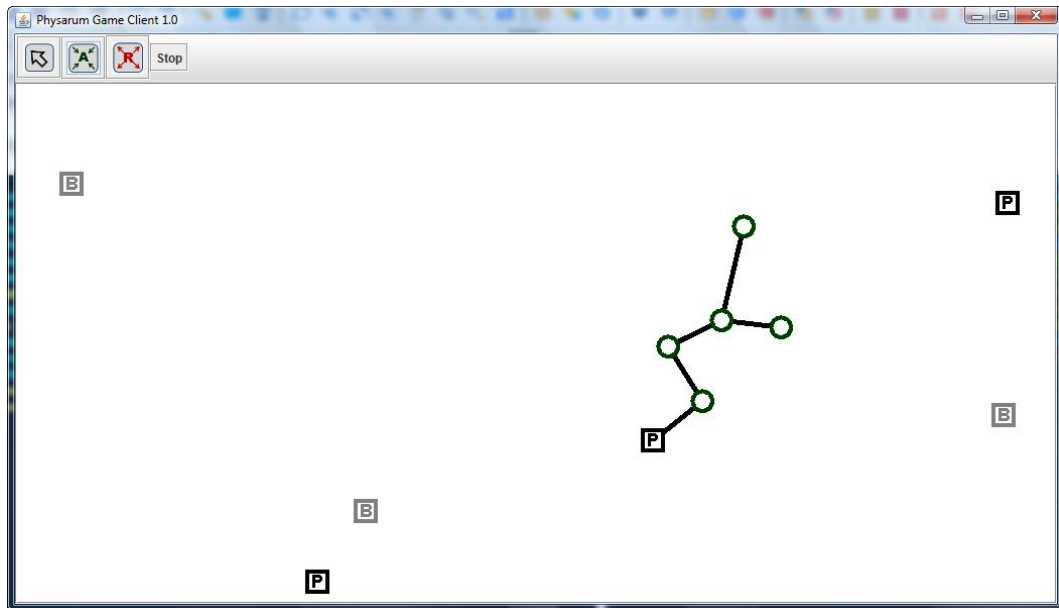


Figure 20. The client window at the first game strategy (by stimuli placement) after further several player movements. The player can place other stimuli. New veins of plasmodium are created, but some of them are annihilated.

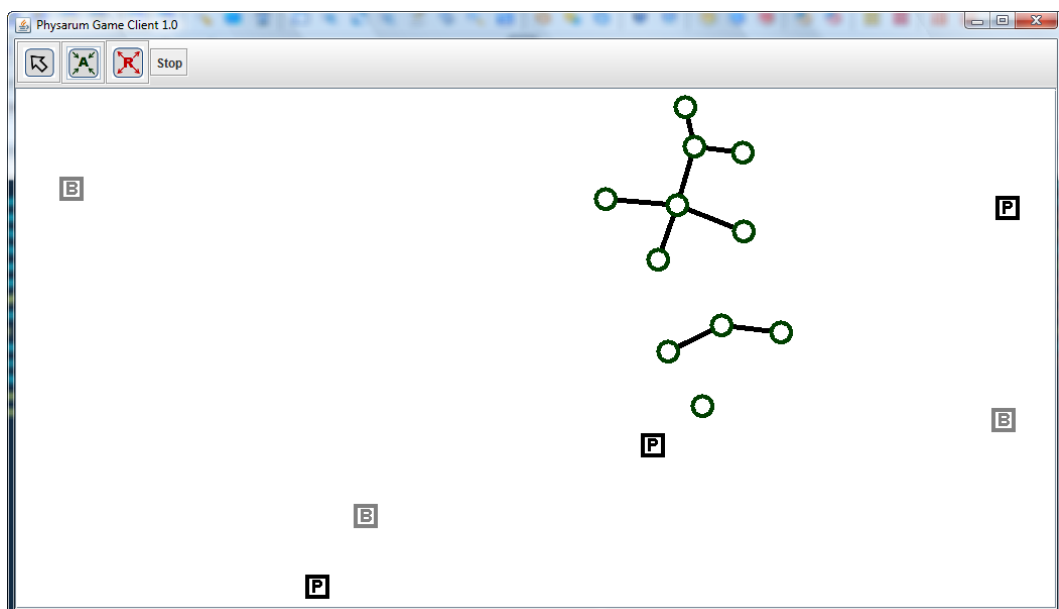
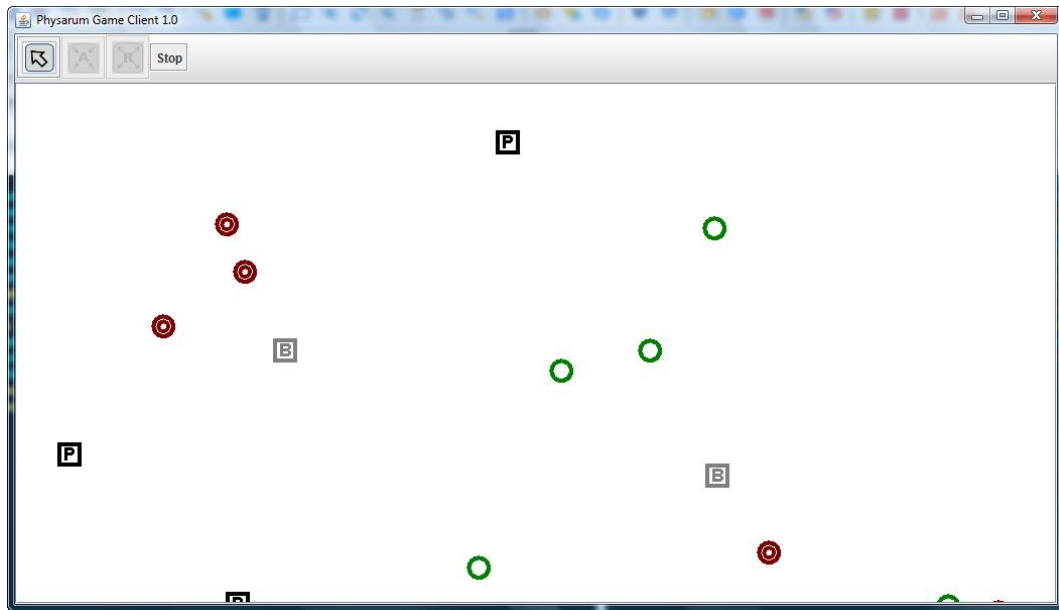
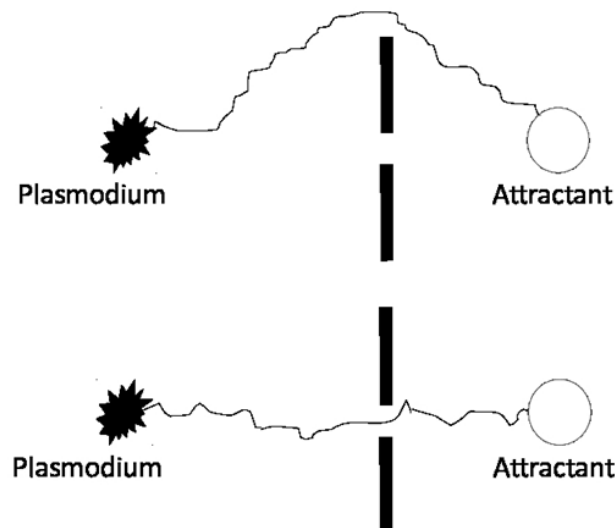


Figure 21. The initial client window at the second game strategy (by plasmodium movement rules). At the beginning, *Physarum* and *Badhamia* as well as stimuli are scattered randomly on the plane.



However, not always concurrent games defined above can describe real motions of *Physarum polycephalum* plasmodia. The problem is that in their motions we cannot approximate atomic acts in all cases, because plasmodia can move differently under the same conditions, see Fig. 22.

Figure 22. Hybrid act. The plasmodium faces the barrier with one slit and in the first experiment it combines two acts at once: repelling and direction. More precisely, we deal here with a transition system with only one stimulus and one passable barrier may have the following three simple actions: (i) pass through, (ii) avoid from left, (iii) avoid from right. But in essence, we deal only with one stimulus and, therefore, with one action, although this action has the three modifications defined above.



As a result, we need to define the so-called *context-based games*, where each *game* can be assumed *infinite*, because its rules can change. Players in context-based games can *change their strategies* and the set of actions is infinite for each player. *Resistance points* for players are reduced to the payoffs if all actions are well-founded. For any game there is *performative efficiency*, when hybrid actions of players belong to the interval of expected modifications.

Thus, the plasmodium motion is an intelligent way of constructing expanding networks for solving complex tasks. This motion has the form of transitions determined by locations of attractants and repellents. On these transitions, it is possible to define p-adic probabilities which are used for defining a knowledge state of plasmodium and its game strategy in occupying attractants as payoffs for the plasmodium. Consequently, the task of controlling the plasmodium motions is considered a game. So, we deal with a kind of *experimental game theory* that is characterized as follows:

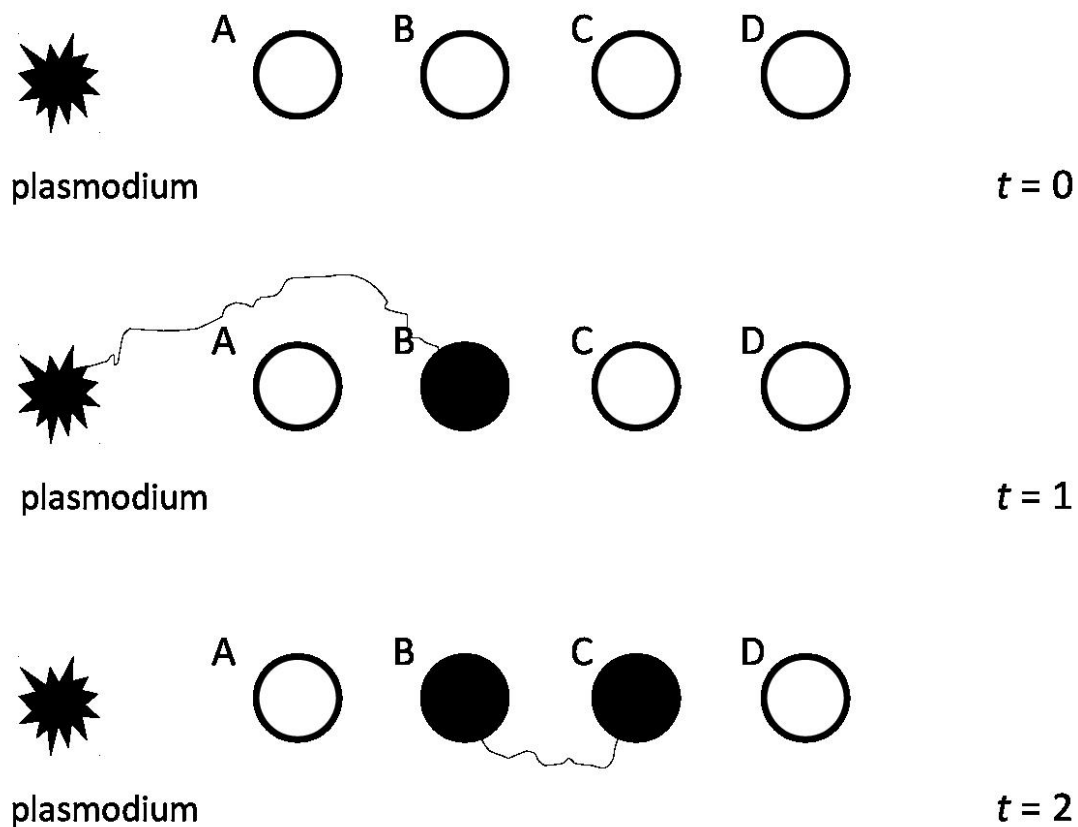
- Attractants are considered payoffs,
- Attractants occupied by the plasmodium are considered states of the game,
- Active zones of plasmodium are considered players,
- Logic gates for behaviours are considered moves (available actions) for the players,
- Propagation of the plasmodium are considered the transition table which associates, with a given set, of states and a given move of the players, the set of states resulting from that move.

2. Logic gates

2.1. Sequential logic gates

Let logic variables x_1, x_2, \dots, x_n be presented by attractants. Each variable x_i can be of a truth value either 0 or 1: it is 0 if x_i is not occupied by the plasmodium and it is 1 if x_i is occupied. Let us denote an ordered collection consisting of all elements of a set $\{x_1, x_2, \dots, x_n\}$ by $[x_1x_2 \dots x_n]$. It is a string. We can replace the variables with their values to obtain a binary combination. For example, variables $\{A, B, C, D\}$ of Fig. 23 have truth values $A = 0, B = 1, C = 1, D = 0$ at $t = 2$ and we receive the binary string $[0110]$.

Figure 23. One of the possible experiments when the plasmodium propagates a protoplasmic tube to attractants A, B, C, D . A is the nearest attractant and it is expected that it will be occupied first. However, the plasmodium occupies B at step $t = 1$ and then C at step $t = 2$.



One single variable can change its values. Moreover, the plasmodium can leave an occupied attractant B so that an appropriate value of B will be then 0. The sequence $\langle 011 \rangle$ means that the variable B is presented by three values fixed during time, since the step $t = 0$ and finishing at $t = 2$.

Thus, our Fig. 23 can be regarded as a truth table structured as follows. There are horizontally located binary strings $X_0 = [0000]$, $X_1 = [0100]$, $X_2 = [0110]$ fixed at time t_0 , t_1 , t_2 respectively. All these strings are meanings of $[ABCD]$ at different time. Also, there are vertically located binary sequences $A(t) = \langle 000 \rangle$, $B(t) = \langle 011 \rangle$, $C(t) = \langle 001 \rangle$, $D(t) = \langle 000 \rangle$.

Now let us define a logical switching as an action to change the value from 0 to 1 or from 1 to 0. For instance, by passing from the binary string X_0 to X_1 , variable B changes its value from logical zero to unity. At the same time, other variables do not change their truth values and, therefore, create a stable background for switching B . In (V. Vasyukevich, *Asynchronous Operators of Sequential Logic*, 2011), the operation of venjunction is defined as the function which is read as follows: “switching x on the background y ” and takes a unity value in the case of switching of variable x from zero to unity at the constant unity value of variable y . It is supposed that the unity value of function remains until x or y reduces to zero. On the basis of venjunction and some other Boolean functions, it is possible to implement asynchronous sequential logic gates on the medium of *Physarum polycephalum*.

Notably that in asynchronous binary sequences there are no external control of time points t_0 , t_1 , t_2 , ..., i.e. there is no external synchronizer which would set these points. The main disadvantage of asynchronous sequential logic gates on slime mould is that there is an analogy between the uncertainty of plasmodium propagation and the uncertainty of position and momentum of quanta. So, in Fig. 23 we expect that the plasmodium occupies the nearest attractant first, but it can be different. Therefore we cannot predict what will be occupied at once and what further. This means that it is difficult to manage switching of logic variables. Nevertheless, we can use a reversible gate which does not erase any information when it acts, because we fix a transformation of the whole computation system.

2.2. Reversible logic gates

In reversible logic gates there is always a unique input associated with a unique output and vice versa. So, any n -bit reversible gate specifies how to map each distinct bit string input of the length n into a distinct bit string output of the same length. For example, the NOT gate is reversible: it is a 1-input and 1-output gate that simply inverts the bit value 0 to 1 and 1 to 0 (see table 1). In the case of the CNOT gate (the 2-bit controlled-NOT gate), the four possible input bit strings are 00, 01, 10, 11 and these are mapped into 00, 01, 11, and 10 respectively. Thus, each reversible gate fixes a specification for how to permute the 2^n possible bit strings inputs expressible in n bits. An n -bit reversible gate is considered an array whose rows and columns are indexed by the 2^n possible bit strings expressible in n bits. The (i,j) -th element of this array has the truth value 1 if and only if the input bit string corresponding to the i -th row is mapped to the output bit string corresponding to the j -th column. In this way we obtain a permutation matrix.

Some implementations of reversible logic gates on plasmodia are pictured in Fig. 24 – 29.

Figure 24. The NOT gate on slime mould, where repellents are denoted by triangles, attractants by circles, black circles are attractants occupied by the plasmodium, white circles are attractant which are not occupied, black triangles are activated repellents, white triangles are deactivated repellents:
 (a) mapping 1 into 1, which is 0; (b) mapping 0 into 1, which is 1; (c) mapping 1 into 0, which is 1; (d) mapping 0 into 0, which is 0.

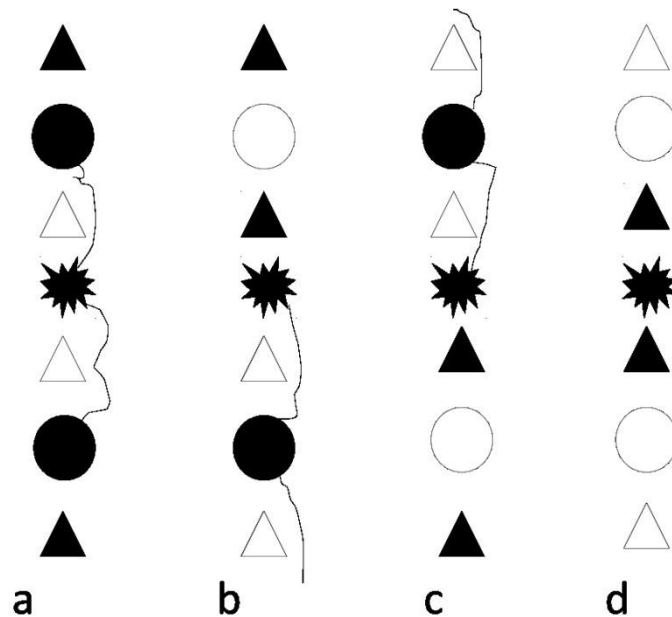


Figure 25. The CNOT gate on slime mould: the string $[ab]$ is transformed into the string $[cd]$ and vice versa.

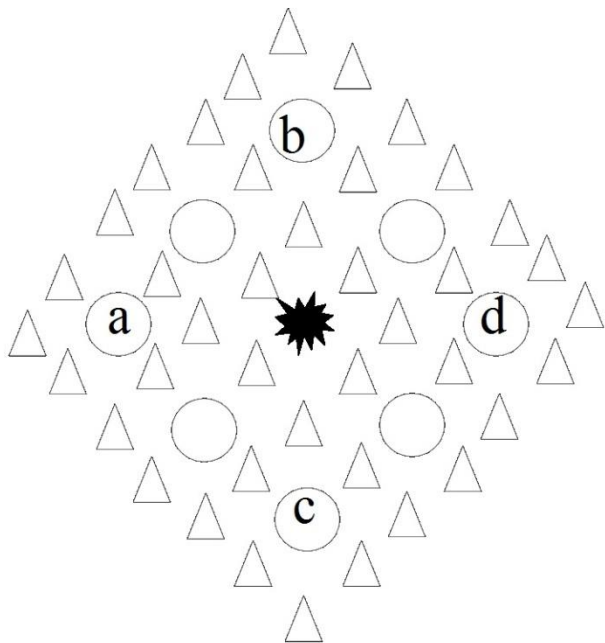


Figure 26. The CNOT gate: mapping 00 into 00.

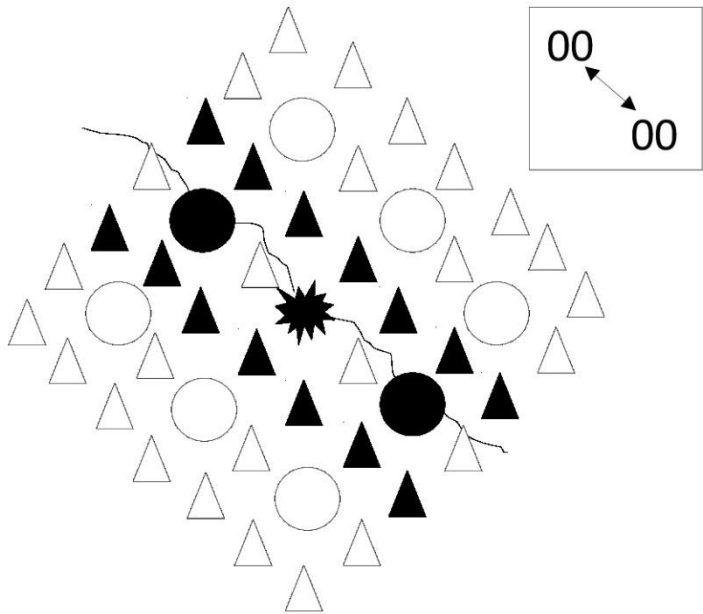


Figure 27. The CNOT gate: mapping 01 into 01.

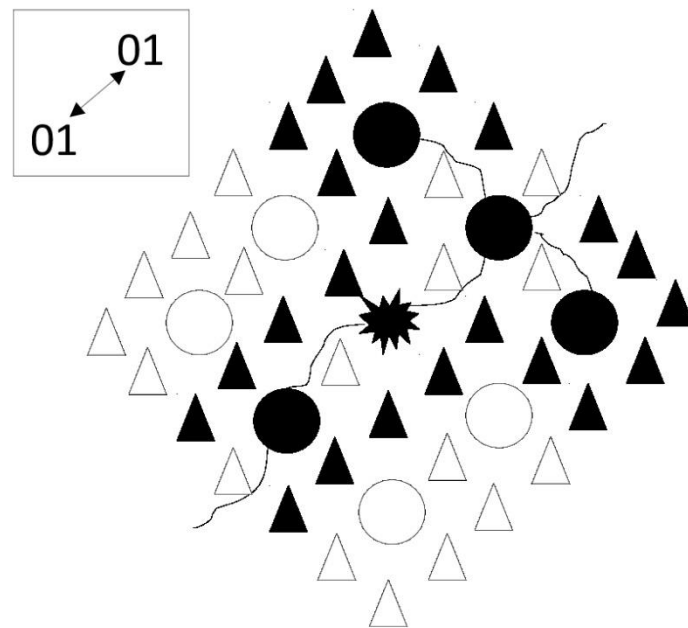


Figure 28. The CNOT gate: mapping 11 into 10.

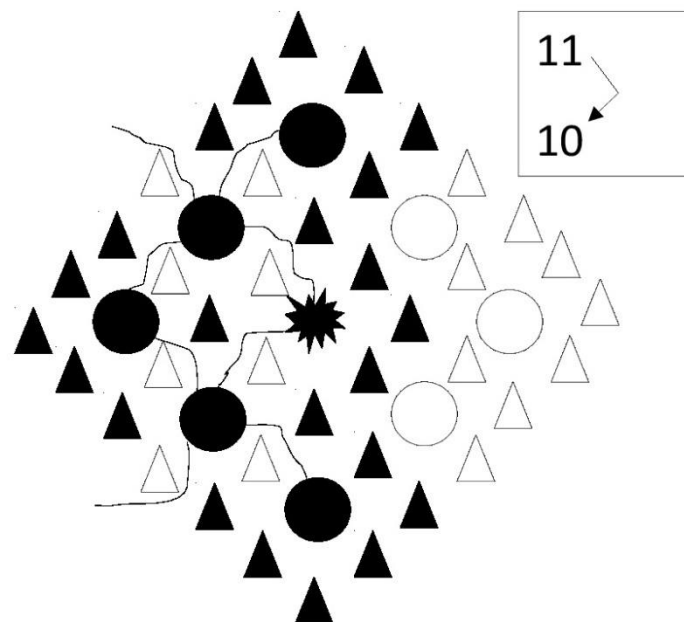
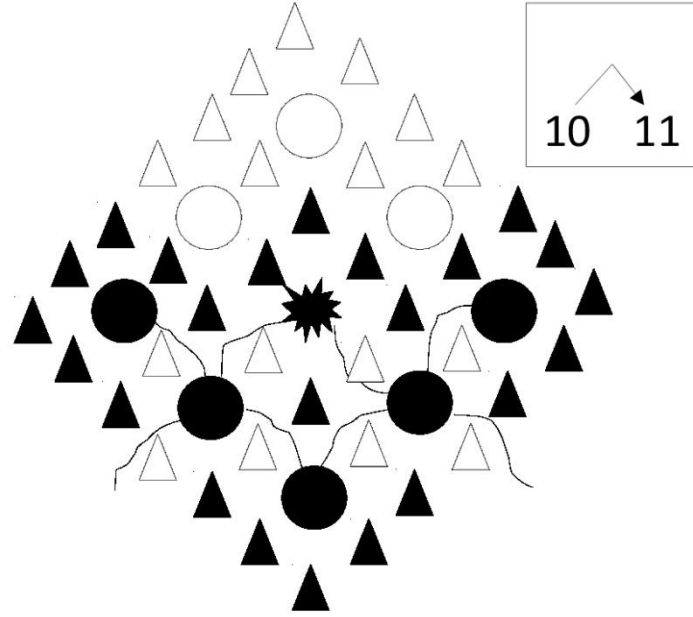


Figure 29. The CNOT gate: mapping 10 into 11.



2.3. Non-linear permutation groups

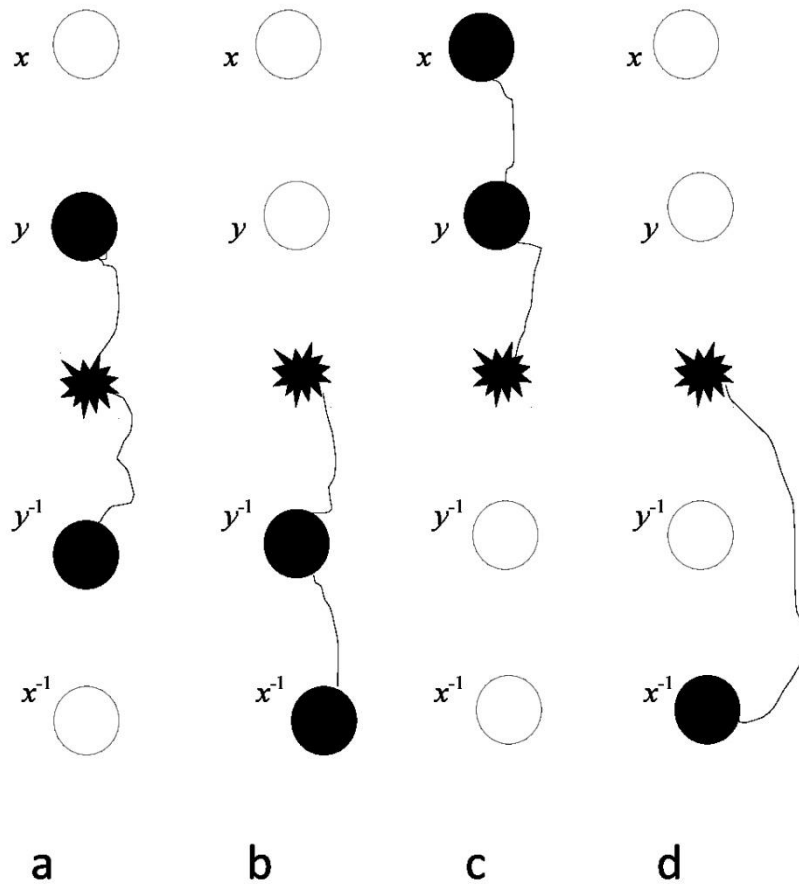
Assume that G is a set of various motions (actions). On this set we can define a linear composition: if x, y are two motions in G , then the motion xy is a result of their composition. Suppose, the operation $x \cdot y$ satisfies the law of associativity, G contains hereafter the unit member e and inverse member x^{-1} . The unit member means an identity motion, when nothing moves. The inverse member x^{-1} of x is a motion which is opposite to x . Obviously, it means that $e \cdot x = x \cdot e = x$ and $x^{-1} \cdot x = x \cdot x^{-1} = e$. So, G is a group and every motion in G causes the setting of linear sequences. For example, the equilateral triangle has the three motions to come back to its place, i.e. the three rotations by 120, 240, and 360 around the center of the triangle mapping every vertex (a_1, a_2, a_3) of the triangle to another one (to a_2, a_3, a_1 , respectively). Indeed, for describing all motions of the triangle we have $x_1 \cdot x_2 \cdot x_3 = e$, where x_i is the i -th rotation by 120, 240, or 360, $i = 1, 2, 3$.

If X is a set (e.g. it is the set of vertexes), then a permutation group may be defined as a group homomorphism h from G to the symmetric group on X . The motion assigns a permutation of X to each element of G in such a way that the permutation of X assigned to: the identity element of G is the identity transformation of X (e.g. the identity rotation of the equilateral triangle); a product $g \cdot h$ of two elements of G is the composition of the permutations assigned to g and h .

Reversible logic gates combinations can be defined within a permutation group as multiplication of matrices.

On the *Physarum polycephalum* motions, logic variables of reversible gates are expressed by attractants. So, a string of the length n is a sequence of n attractants. Usually, the plasmodium occupies the whole region in many directions simultaneously. Let us assume that all attractants stand in line (see Fig. 30). So, the plasmodium can move only to one of the two possible directions: either up or down. Suppose that we do not have repellents at all. We can try to define the plasmodium motions as a group G of actions. Each attractant will be denoted by a member of G in the following way: (i) all attractants above are enumerated from a_0 to a_n and all attractants below are enumerated from b_0 to b_n ; (ii) each b_i is equal a_i^{-1} (this means that each a_i is equal to b_i^{-1}), $i = 0, \dots, n$; (iii) the member of G holds if it is occupied by the plasmodium. Hence, each motion a_i is opposite to the motion b_i . The linear composition $x \cdot y$ at time t is defined for all $x, y \in G$ as follows: $x \cdot y$ occurs at time t if and only if both x and y are occupied by the plasmodium at this time.

Figure 30. Non-linear permutation groups.



Now, let us answer the question, whether G is a group indeed. To be a group, the following assumption has to hold on G : the plasmodium chooses a direction of motions (up, down or up and down simultaneously) just at its start point. It cannot change its strategy at the points x, y, x^{-1}, y^{-1} . Nevertheless, this assumption is very strong and to realize it we must appeal to repellents, which block any come-back of plasmodium. Thus, the plasmodium can permanently change its decisions at the points x, y, x^{-1}, y^{-1} . In this case, x can become opposite to y^{-1} (i.e. y^{-1} to x as well) and y can become opposite to x^{-1} (x^{-1} to y), although first x was opposite to x^{-1} and y to y^{-1} . Let us assume that the plasmodium changes its choices all the time at all points. Then we obtain a strong extension of the group. Instead of usual members of G we will deal with mutually defined streams: (i) $x = x \cdot y^{-1}$ if the plasmodium comes back from x to y^{-1} ; (ii) $y^{-1} = y^{-1} \cdot x$ if the plasmodium comes back from y^{-1} to x ; (iii) $y = y \cdot x^{-1}$ if the plasmodium comes back from y to x^{-1} ; (iv) $x^{-1} = x^{-1} \cdot y$ if the plasmodium comes back from x^{-1} to y .

Furthermore, the plasmodium can move back at points x and y^{-1} (respectively, at points y and x^{-1}) simultaneously. This motions will be denoted by $(\}x, y^{-1}\{)$ (respectively, by $(\}y, x^{-1}\{)$), where $x = xy^{-1}$ and $y^{-1} = y^{-1}x$ (respectively, $y = yx^{-1}$ and $x^{-1} = x^{-1}y$). The objects $(\}x, y^{-1}\{)$, $(\}y, x^{-1}\{)$ are called *wave sets*. The composition of logic devices consisting just of attractants standing in line is not a conventional permutation group. This new group will be called non-linear permutation group. This group contains infinite streams and wave sets.

3. p-Adic valued universe

3.1. Double-slit experiment

There are many spatial presentations of algorithms such as Kolmogorov-Uspensky machines which can be approximated in the plasmodium. However, these approximations cannot be used for projecting an unconventional computer on programmable behaviour of *Physarum polycephalum*. The matter is that we should know how we can extract atomic acts of plasmodium from its possible actions. In this case we would be able to define a natural transition system of *Physarum polycaphalum*, $\langle \text{States}, \text{Edg} \rangle$, as a Kolmogorov-Uspensky machine and then to construct a computer by using the conventional notion of algorithms. Nevertheless, we cannot extract atomic acts as a linear sequence in the meaning of algorithms. This fact can be proven by the double-slit experiment.

Due to experiments performed by our colleagues, we know some basic acts of plasmodium, see Fig. 14: (i) *Direction*: the plasmodium moves towards the attractant; (ii) *Fuse*: the two plasmodia fuse after meeting the same attractant; (iii) *Split*: the plasmodium splits in front of many attractants.

The principal question is that whether we can consider these acts of plasmodium as its atomic acts. To answer that question we can carry out the double-slit experiment, see Fig. 31 – 33.

Figure 31. The double-slit experiment: slit 1 is opened, slit 2 is covered.

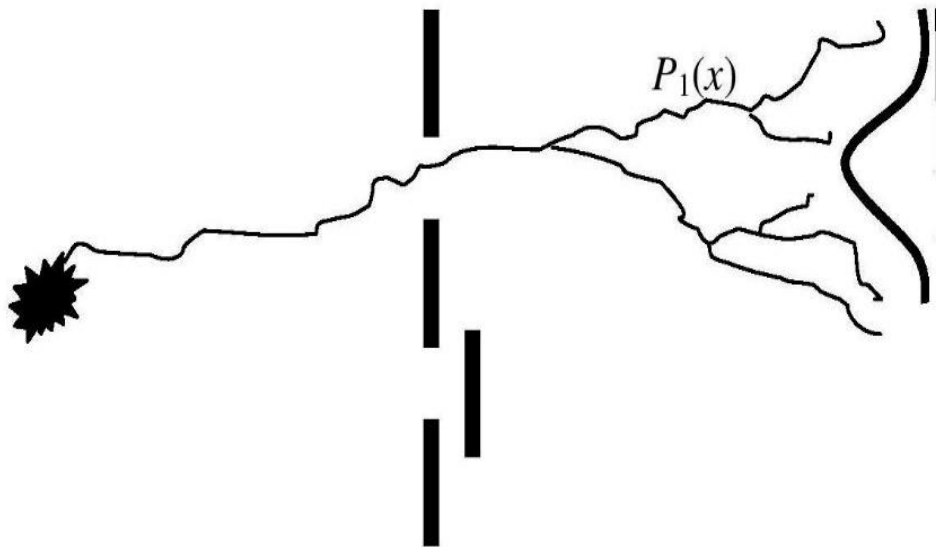


Figure 32. The double-slit experiment: slit 1 is covered, slit 2 is opened.

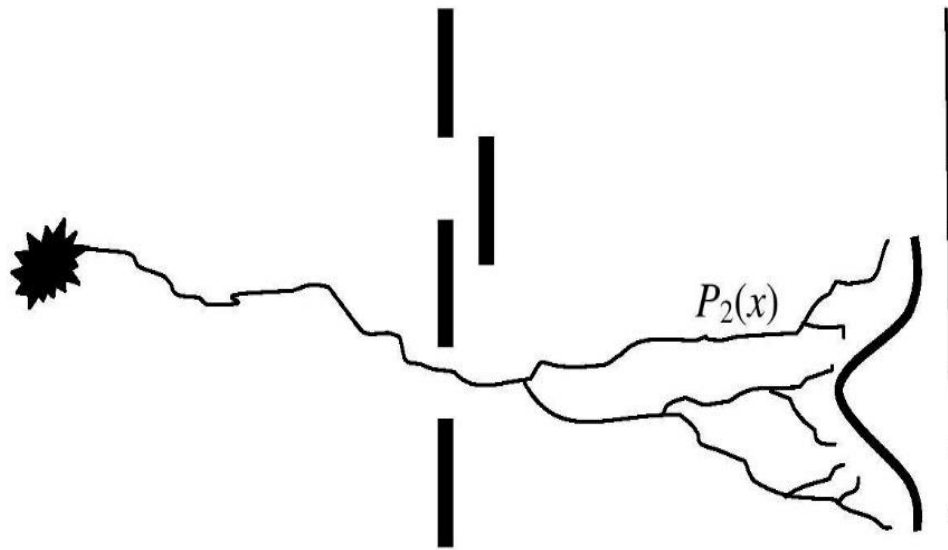
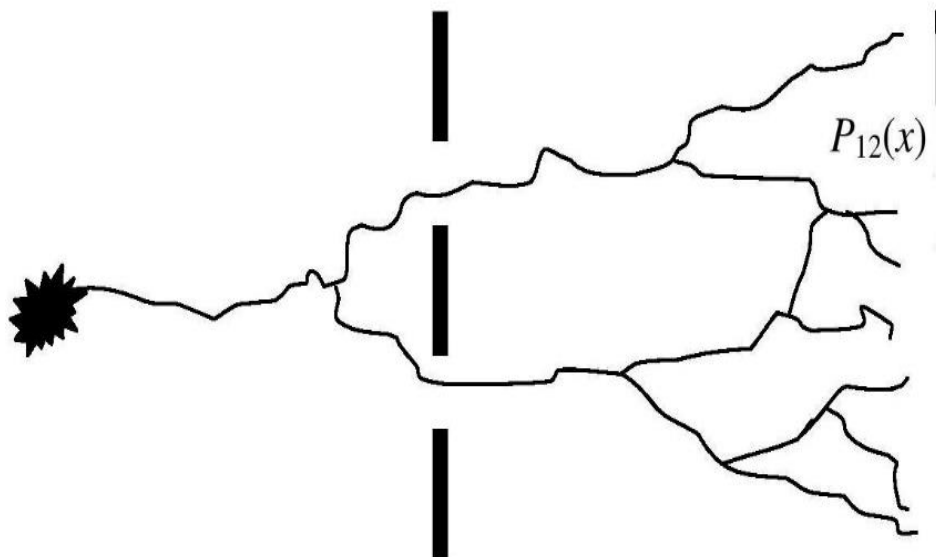


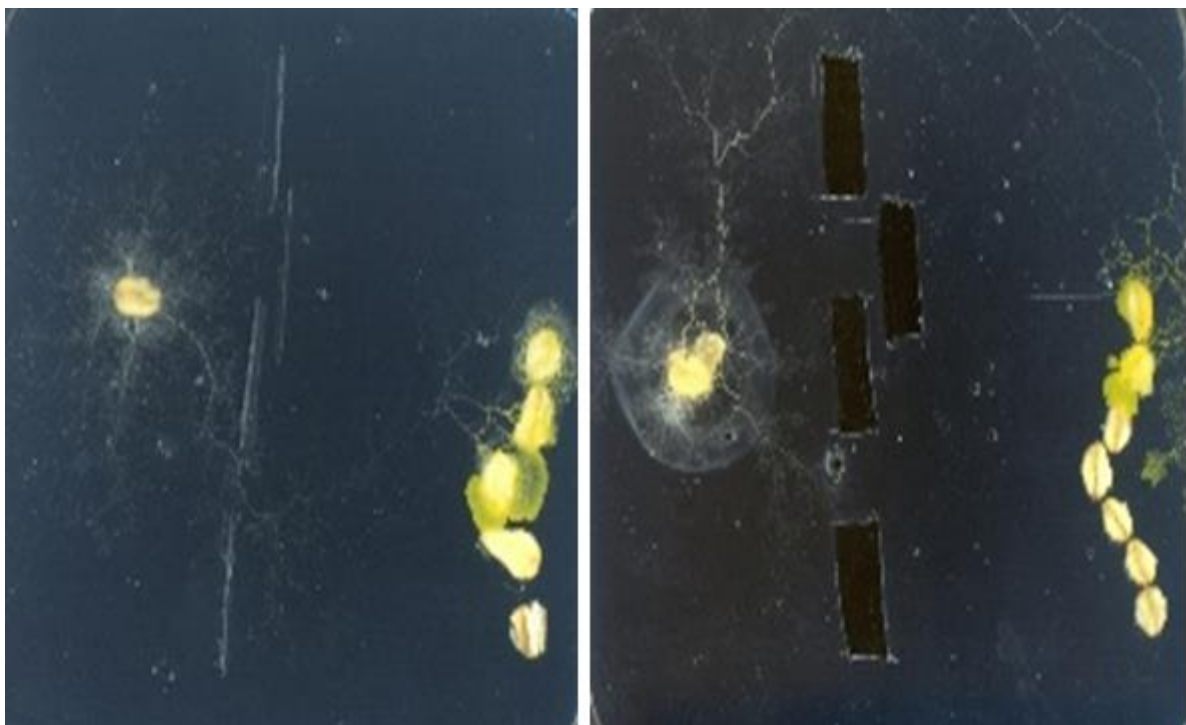
Figure 33. The double-slit experiment: both slit 1 and 2 are opened.



The results of experiments mean that the plasmodium has the fundamental property of photons discovered in the double-slit experiment, i.e. we face a self-inconsistency, too, according to that it is impossible to approximate not only single photons, but also single actions of *Physarum polycephalum*. Indeed, we observe the one act of *Physarum* that is partly *Direction*, partly *Fuse*, and partly *Split*. So, this one act of plasmodium is not individual, but collective. This allows us to state that we face the individual-collective duality of *Physarum polycephalum* plasmodium. In other words, there are no atomic acts for *Physarum polycephalum*. Its acts are parallel: in the one act, the plasmodium can do many things at once and it can always perform many and many acts concurrently. Please see also Fig. 22.

Real double-slit experiments with *P. polycephalum* plasmodia were performed by prof. Andrew Adamatzky, see Fig. 34.

Figure 34. Real double-slit experiment with *Physarum*. Source: Schumann, A., Adamatzky, A.: *The Double-Slit Experiment with Physarum Polycephalum and p-Adic Valued Probabilities and Fuzziness*, International Journal of General Systems, 27 Jan 2015, DOI: 10.1080/03081079.2014.997530.



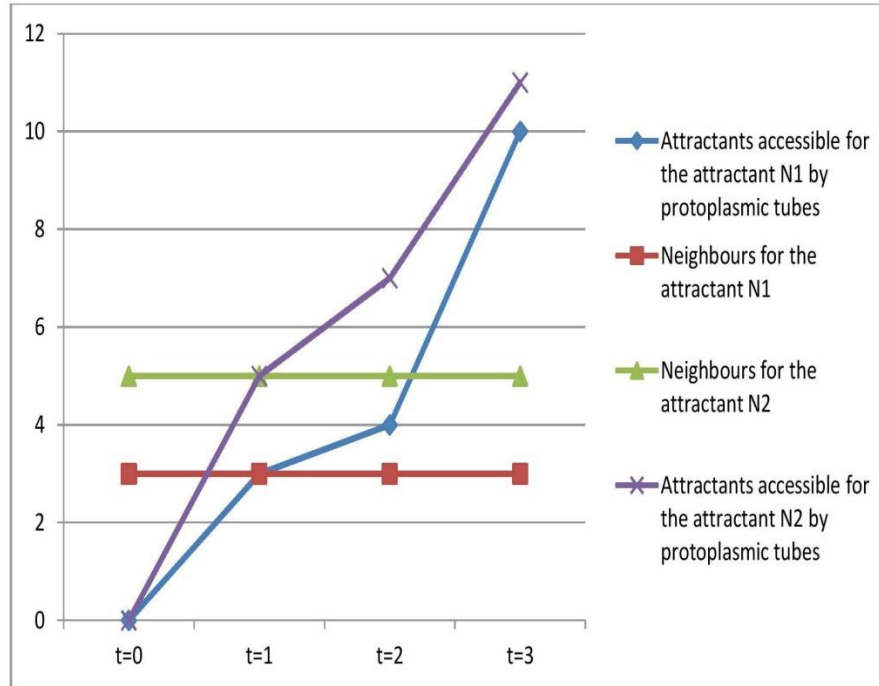
3.2. Growing Sample Space

In probability theory, the universe is understood as a sample space Ω such that every member of $\mathbf{P}(\Omega)$ is called event. Conventionally, probability measures run over real numbers of the unit $[0, 1]$ and their domain is a Boolean algebra of $\mathbf{P}(\Omega)$ with atoms (singletons) denoting simple (atomic) events. In the previous section, we have just shown that in the *Physarum* behaviour we cannot find out atomic events/actions, therefore we cannot define additive measures in the universe Ω . This means that if we wanted to define a domain of probability measures to calculate probabilities on the plasmodium, we could not appeal to the Boolean algebra of $\mathbf{P}(\Omega)$. Otherwise, our measures would be additive.

So, in our case, the powerset of $\mathbf{P}(\Omega)$ for the universe Ω must be non-atomistic. It would be possible if the sample space Ω was not fixed, but it changed, continuously at $t = 0, 1, 2, \dots$. In other words, let us assume that Ω is a set at $t = 0$, which can grow, be expanded, decrease or just change in itself at next steps. As a consequence, we will deal not with atoms as members of Ω , but with streams of the form $\langle a_0, a_1, a_2, \dots \rangle$, where $a_0 \in \Omega$ at $t = 0$, $a_1 \in \Omega$ at $t = 1$, $a_2 \in \Omega$ at $t = 2$, etc.

Let us denote this instable set, i.e. the set of all these infinite streams, by Ω^ω and call it a *wave set*. The powerset $\mathbf{P}(\Omega^\omega)$ is not a Boolean algebra, e.g. it is not atomistic, see Fig. 35.

Figure 35. Growing sample space.



p-Adic probabilities in the growing sample space Ω are defined as follows. Let $|\Omega| = p - 1$ be a cardinality number, A, B, \dots be subsets of Ω . Assume $A :=$ “Attractants accessible for the attractant N_1 by protoplasmic tubes” and $B :=$ “Neighbours for the attractant N_1 ”, etc. Now, let us define the cardinality number of $X^\omega \subseteq \Omega^\omega$ as follows: $|X^\omega| := (|X| \text{ for } t = 0; |X| \text{ for } t = 1; |X| \text{ for } t = 2, \dots)$, where $|X|$ means a cardinality number of X . Notice that if $|\Omega| = p - 1$, then $|A^\omega|$, $|B^\omega|$, and $|\Omega^\omega|$ cover p-adic integers. Then the simplest way to define p-adic probabilities is as follows:

$$P(A^\omega) = |A^\omega|$$

$$\text{or } P(A^\omega) = |A^\omega| / |\Omega^\omega|$$

Notice that in p-adic metric, $|\Omega^\omega| = -1$

Hence, we use a *p-adic valued fuzzy measure* in space Ω defined thus. A function $P(\Omega) \rightarrow \mathbf{Z}_p$ is a p-adic valued fuzzy measure if it verifies the following properties: (i) $P(\emptyset) = 0$; (ii) if $A, B \in \mathbf{P}(\Omega)$, then $P(A \cup B) = \sup(P(A), P(B))$; (iii) if $A, B \in \mathbf{P}(\Omega)$, then $P(A \cap B) = \inf(P(A), P(B))$; (iv) $P(\Omega) = -1$; (v) if $A \in \mathbf{P}(\Omega)$, then $P(\neg A) = -1 - P(A)$. This measure is defined for time $t = 0, 1, 2, \dots$ and shows, how a set $A \in \mathbf{P}(\Omega)$ changes from the start point $t = 0$. Recall that \mathbf{Z}_p denotes the set of p-adic integers, i.e. numbers of the form:

$$n = \alpha_0 + \alpha_1 \cdot p + \dots + \alpha_k \cdot p^k + \dots = \sum_{k=0}^{\infty} \alpha_k \cdot p^k,$$

where $\alpha_k \in \{0, 1, \dots, p - 1\}$ for all natural number k . The p-adic number sometimes has the following notation $n = \dots \alpha_k \dots \alpha_1 \alpha_0$.

Notice that the p-adic numbers differ a lot from the real numbers, e.g. they are not linear-ordered and any distance on them is non-Archimedean. Accordingly, the p-adic valued fuzzy measure possesses many unique properties.

Agent i 's knowledge structure is a function \mathbf{P}_i which assigns to each $a \in \Omega^\omega$ a non-empty subset of Ω^ω , so that each world a belongs to one or more elements of each \mathbf{P}_i , i.e. Ω^ω is contained in a union of \mathbf{P}_i , but \mathbf{P}_i are not mutually disjoint.

$$K_i A^\omega = \{a : A^\omega \subseteq \mathbf{P}_i(\omega)\}$$

If the number of neighbouring attractants does not exceed $p - 1$, see Fig. 36 – 38, then we can consider the plasmodium motions as strings which can be analyzed within a p -adic valued logic, see:

- Schumann, A., Adamatzky, A.: *The Double-Slit Experiment with Physarum Polycephalum and p -Adic Valued Probabilities and Fuzziness*, “International Journal of General Systems”, 27 Jan 2015, DOI: 10.1080/03081079.2014.997530.
- Schumann, A.: *p -Adic valued logical calculi in simulations of the slime mould behaviour*, “Journal of Applied Non-Classical Logics”, 2015, DOI: 10.1080/11663081.2015.1049099.

Figure 36.4-adic valued strings.

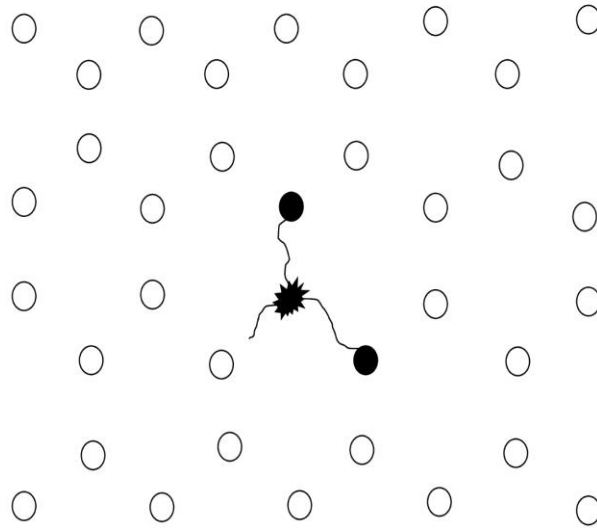


Figure 37. 4-adic valued strings.

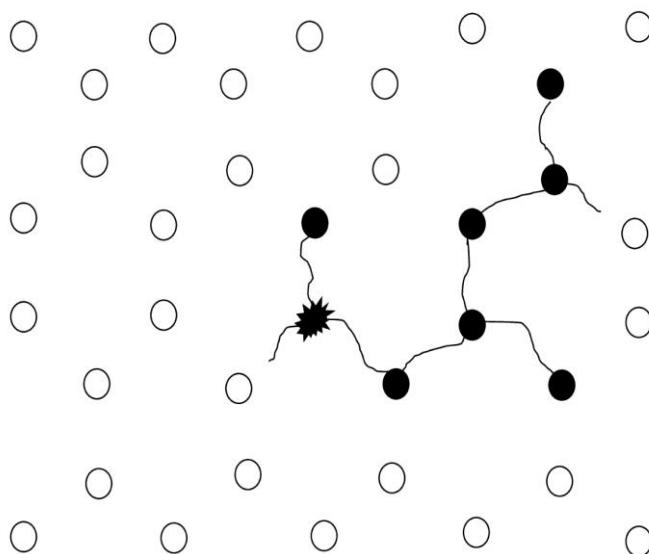
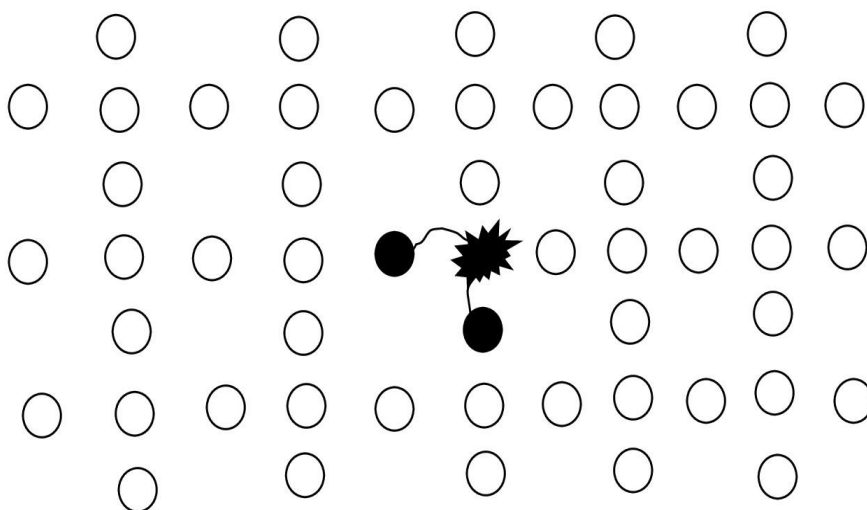


Figure 38. 5-adic valued strings.



4. Go games

Go game is a board game with two players (called Black and White) who alternately place black and white stones, accordingly, on the vacant intersections (called points) of a board with a 19x19 grid of lines. Black moves first. Stones are placed until they reach a point where stones of another colour are located. There are the following two basic rules of the game: (1) each stone must have at least one open point (called liberty) directly next to it (up, down, left, or right), or must be part of a connected group that has at least one such open point; stones which lose their last liberty are removed from the board; (2) the stones must never repeat a previous position of stones. The aim of the game is in surrounding more empty points by player's stones. At the end of game, the number of empty points player's stones surround are counted, together with the number of stones the player captured.

Let we have a 5-adic valued universe for plasmodium motions with a Voronoi topology, see Fig. 39. Then we can implement Go games on plasmodia, see Fig. 40.

Figure 39. The six Voronoi cells in accordance with the four attractants denoted by the black stones and with the two repellents denoted by the white stones. The plasmodium located in the center of the picture connects the three attractants by the three protoplasmic tubes. It cannot see the fourth attractant because of the two repellents.

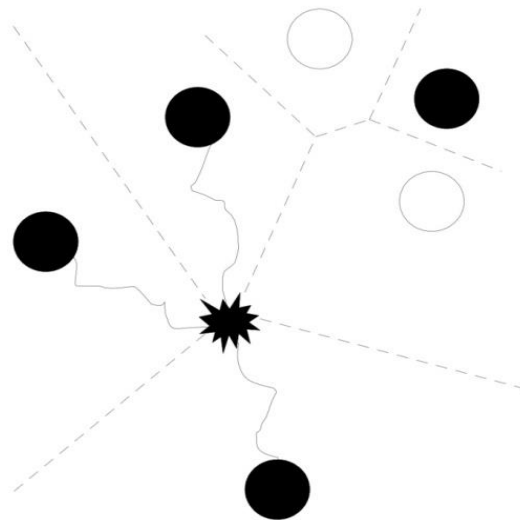
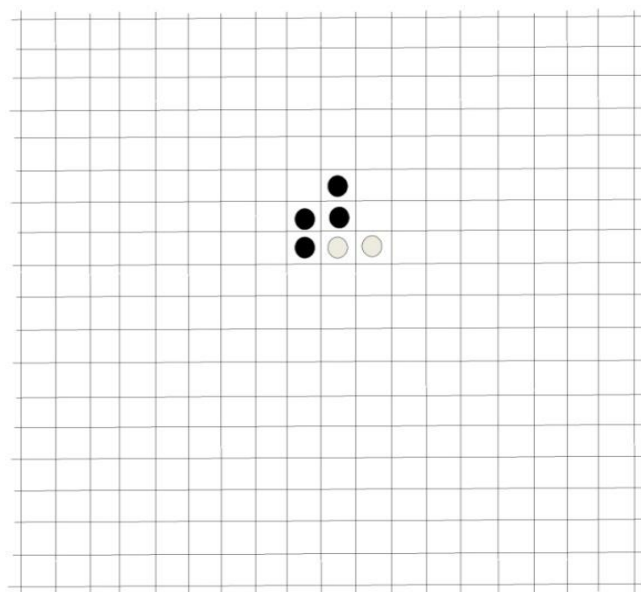


Figure 40. The Go game board with two white stones designating repellents and four black stones designating attractants.



In Go games, syllogistic letters S, P, M, \dots are interpreted as attractants as follows: a data point S is considered empty if and only if an appropriate attractant denoted by S is not occupied by plasmodium. Let us define syllogistic strings of the form SP at time t with the following notation: ' S is(t) P ,' and with the following meaning: SP at t is true if and only if S and P are neighbour cells and both S and P are not empty at t , otherwise SP is false. Thus, this definition represents syllogistic reasoning as *Physarum* labelled transition system $\langle \text{States}; \text{Edg} \rangle$, where $S, P, M, \dots \in \text{States}$ and true propositions ' S is(t) P ', $\dots \in \text{Edg}$.

Using the definition of syllogistic strings, we can define simple syllogistic propositions as follows:

- 1) 'All S are P at time t ' ($Sa(t)P$): there is a string AS at time t and for any A which is a neighbour for S and P , there are strings AS and AP . This means that we have a massive-parallel occupation of region at t , where the cells S and P are located.
- 2) 'Some S are P at time t ' ($Si(t)P$): for any A which is a neighbour for S and P at t , there are no strings AS and AP . This means that the plasmodium cannot reach S from P or P from S immediately at t .
- 3) 'No S are P at time t ' ($Se(t)P$): there exists A at time t which is a neighbour for S and P such that there is a string AS or there is a string AP . This means that the plasmodium occupies S or P , but surely not the whole region at time t , where the cells S and P are located.
- 4) 'Some S are not P at time t ' ($So(t)P$): for any A which is a neighbour for S and P at time t there is no string AS or there exists A which is a neighbour for S and P such that there is no string AS or there is no string AP .

p-Adic probabilities are semantics for our p-adic fuzzy (non-Aristotelian) syllogistic. Let $A_t\{S,P\}$ mean “attractants that are neighbours of attractants S and P at time t ” and $O_t\{S,P\}$ mean “attractants that are neighbours of attractants S and P and are occupied by the plasmodium at time t ”. The cardinalities of $A_t\{S,P\}$ and $O_t\{S,P\}$ are finite p-adic numbers like that: $\alpha_k \dots \alpha_3 \alpha_2 \alpha_1 \alpha_0$. It means that p-adic fuzzy measure P takes these values on $A_t\{S,P\}$ and $O_t\{S,P\}$.

Then we can define the following semantic rules:

$Sa(t)P$ is true if and only if $P(A_t\{S,P\}) = P(O_t\{S,P\})$;

$Si(t)P$ is true if and only if $P(O_t\{S,P\}) = 0$;

$Se(t)P$ is true if and only if $P(O_t\{S,P\}) > 0$;

$So(t)P$ is true if and only if $P(A_t\{S,P\}) > P(O_t\{S,P\})$.

In the Aristotelian Go games we appeal to the following interpretation of atomic propositions:

SaP .

In the formal syllogistic: there exists A such that A is S and for any A , if A is S , then A is P . *In the Go game model:* there is a cell A containing the black stone and for any A , if AS is true, then AP is true. *In the Physarum model:* there is a plasmodium in the cell A and for any A , if AS is true, then AP is true.

SiP .

In the formal syllogistic: there exists A such that both AS is true and AP is true. *In the Go game model:* there exists a cell A containing the black stone such that AS is true and AP is true. *In the Physarum model:* there exists a plasmodium in the cell A such that AS is true and AP is true.

SeP .

In the formal syllogistic: for all A , AS is false or AP is false. *In the Go game model:* for all cells A containing the black stones, AS is false or AP is false. *In the Physarum model:* for all plasmodia A , AS is false or AP is false.

SoP .

In the formal syllogistic: for any A , AS is false or there exists A such that AS is true and AP is false. *In the Go game model:* for all cells A containing the black stones, AS is false or there exists A such that AS is true and AP is false. *In the Physarum model:* for any plasmodia A , AS is false or there exists A such that AS is true and AP is false.

In this Aristotelian model for verifying all the basic syllogistic propositions, we will use the following four cells: x , y , x' , y' of the game board with the 19x19 grid of lines, where x' means all cells which differ from x , but they are neighbours for y , and y' means all cells which differ from y and are neighbours for x . These cells express appropriate meanings of syllogistic letters. The corresponding universe of discourse will be denoted by means of the following diagram:

x	y'
y	x'

Then atomic propositions are pictured by the diagrams of Fig. 41, 42.

Figure 41. Aristotelian existence propositions and their negations.

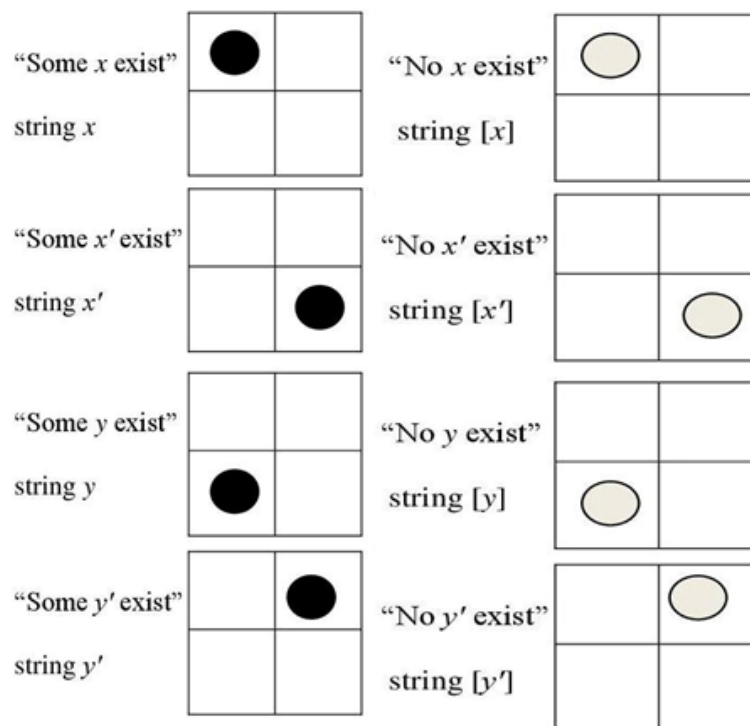
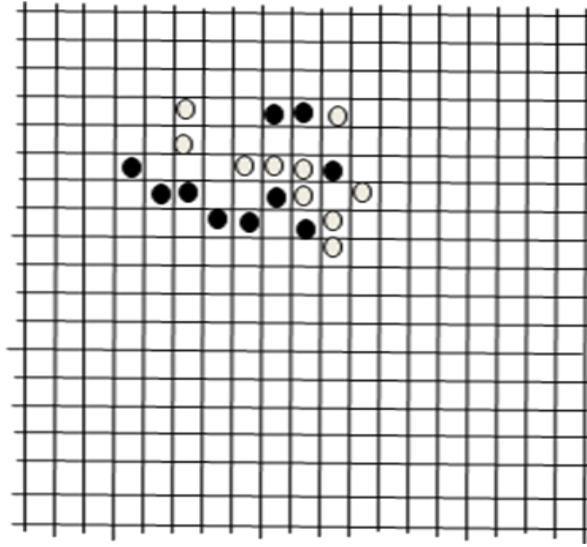


Figure 42. Aristotelian atomic propositions.

<p>“Some xy exist” = “Some x are y” = “Some y are x”; strings xy and yx</p>		<p>“All x are y” = “No x are y”; string xy & $x[y]$</p>	
<p>“Some xy' exist” = “Some x are y” = “Some y' are x”; strings xy' and $y'x$</p>		<p>“All x are y” = “No x are y”; string xy' & $x[y]$</p>	
<p>“Some $x'y$ exist” = “Some x' are y” = “Some y are x”; strings $x'y$ and yx'</p>		<p>“All x' are y” = “No x' are y”; string $x'y$ & $x'[y]$</p>	
<p>“Some $x'y'$ exist” = “Some x' are y” = “Some y' are x”; strings $x'y'$ and $y'x'$</p>		<p>“All x' are y” = “No x' are y”; string $x'y'$ & $x'[y]$</p>	
<p>“No xy exist” = “No x are y” = “No y are x”; strings $[xy]$ and $[yx]$</p>		<p>“All y are x” = “No y are x”; string yx & $y[x]$</p>	
<p>“No xy' exist” = “No x are y” = “No y' are x”; strings $[xy']$ and $[y'x]$</p>		<p>“All y are x” = “No y are x”; string yx' & $y[x]$</p>	
<p>“No $x'y$ exist” = “No x' are y” = “No y are x”; strings $[x'y]$ and $[yx']$</p>		<p>“All y' are x” = “No y' are x”; string $y'x$ & $y'[x]$</p>	
<p>“No $x'y'$ exist” = “No x' are y” = “No y' are x”; strings $[x'y']$ and $[y'x']$</p>		<p>“All y' are x” = “No y' are x”; string $y'x'$ & $y'[x]$</p>	
<p>“Some x are y”, “Some x are y”; strings xy, yx, xy', $y'x$</p>		<p>“Some y are x” “Some y are x”; strings xy, yx, $x'y$, yx'</p>	
<p>“Some x' are y”, “Some x' are y”; strings $x'y'$, $y'x'$, $x'y$, yx'</p>		<p>“Some y' are x” “Some y' are x”; strings xy', $y'x$, $x'y'$, $y'x'$</p>	

Let us consider a game of Go at time step 10, i.e. when White and Black players have placed the 10 white stones and the 10 black stones respectively. Let this game be pictured in Fig. 43. Each Voronoi cell is denoted from $S_{1,1}$ to $S_{18,18}$. So, in Fig. 43 syllogistic letters $S_{6,4}$, $S_{7,5}$, $S_{7,6}$, $S_{8,7}$, $S_{8,8}$, $S_{7,9}$, $S_{8,10}$, $S_{6,11}$, $S_{4,9}$, $S_{4,10}$ are understood as non-empty and syllogistic letters $S_{4,6}$, $S_{5,6}$, $S_{6,8}$, $S_{6,9}$, $S_{6,10}$, $S_{4,11}$, $S_{7,10}$, $S_{7,12}$, $S_{11,8}$, $S_{12,8}$ as empty. As a result, we can build some true syllogistic propositions in this universe like that: ‘Some $S_{7,5}$ are $S_{7,6}$ ’, ‘Some $S_{8,7}$ are $S_{8,8}$ ’, ‘Some $S_{4,9}$ are $S_{4,10}$ ’, ‘No $S_{4,6}$ are $S_{5,6}$ ’, ‘No $S_{6,8}$ are $S_{6,9}$ ’, ‘No $S_{6,9}$ are $S_{6,10}$ ’, ‘No $S_{6,10}$ are $S_{7,10}$ ’, ‘No $S_{11,8}$ are $S_{12,8}$ ’, etc.

Figure 43. The Aristotelian Go game on plasmodia at time step 10.



Hence, the plasmodium motion is an intelligent way of constructing expanding networks for solving complex tasks. This motion has the form of transitions determined by locations of attractants and repellents. On these transitions, it is possible to define p-adic probabilities which are used for defining a knowledge state of plasmodium and its game strategy in occupying attractants as payoffs for the plasmodium. Consequently, the task of controlling the plasmodium motions is considered a Go game.

Part III. Programming of *Physarum* Machines. Materials from Conference Presentations. Selected Slides

1. Ladder diagrams and Petri net models

1.1. Ladder diagrams

A network of protoplasmic tubes connects the masses of protoplasm and, as a result, the plasmodium develops a planar graph, where the food sources or pheromones are considered as nodes and protoplasmic tubes as edges.

Physarum polycephalum networks have been used to build the so-called *Physarum* machines. A *Physarum* machine is a programmable amorphous biological computer experimentally implemented in the vegetative state of true slime mould of *Physarum polycephalum*. The operations in *Physarum* machines can be determined by the following stimuli:

- (i) The set of attractants $\{A_1, A_2, \dots\}$. Attractants are sources of nutrients or pheromones, on which the plasmodium feeds.
- (ii) The set of repellents $\{R_1, R_2, \dots\}$. Plasmodium of *Physarum* avoids light and some thermo- and salt-based conditions.

Formally, a structure of the *Physarum* machine is defined as a triple $\text{PM} = \{P, A, R\}$, where:

- $P = \{P_1, P_2, \dots, P_k\}$ is a set of original points of plasmodium.
- $A = \{A_1, A_2, \dots, A_m\}$ is a set of attractants.
- $R = \{R_1, R_2, \dots, R_n\}$ is a set of repellents.




A dynamics of the *Physarum* machine over time can be described by the family of protoplasmic veins propagating by plasmodium.

Ladder logic is the most popular programming language used to program Programmable Logic Controllers (PLCs). The language was developed from the electromechanical relay system-wiring diagrams. Programs in the ladder logic language are written graphically in the form of the so-called ladder diagrams. The notation assumes that contacts are controlled by discrete (binary) inputs and coils control discrete (binary) outputs.

We can distinguish three main types of elements of ladder diagrams:

- *Normally open contact* (NOC). It passes power (i.e., it is *on*) if the binary input assigned to it has value 1. Otherwise, it does not pass power (i.e., it is *off*), see Table 1.
- *Normally closed contact* (NCC). It passes power (*on*) if the binary input assigned to it has value 0. Otherwise, it does not pass power (*off*), see Table 1.
- *Coil* (C). If it is passing power (i.e., it is *on*), a value of the binary output assigned to it is set to 1. Otherwise (i.e., it is *off*), a value of the binary output assigned to it is set to 0, see Table 1.

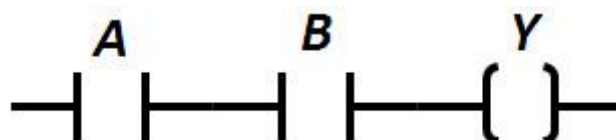
Table 1. Basics of Ladder Diagrams.

Symbol	Meaning
	Normally open contact
	Normally closed contact
	Coil

Now, we can define basic logic gates by ladder diagrams.

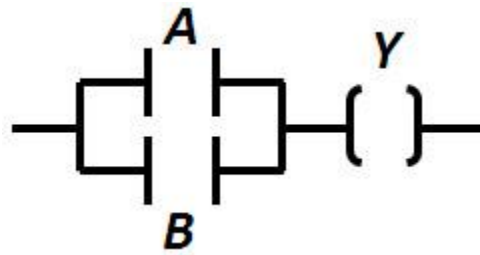
The *ladder diagram AND gate*. The coil is *on* ($Y = 1$) if and only if both contacts are *on*. It is satisfied if $A = 1$ and $B = 1$. Otherwise, the coil is *off*, see Fig. 44.

Figure 44. The ladder diagram AND gate.



The *ladder diagram OR gate*. The coil is *on* ($Y = 1$) if at least one contact is *on*. It is satisfied if $A = 1$ or $B = 1$. Otherwise, the coil is *off*, see Fig. 45.

Figure 45. The ladder diagram OR gate.



The *ladder diagram NOT gate*. The coil is *on* ($Y = 1$) if a contact is *on*. $A = 0$ causes the contact is switched on. The coil is *off* ($Y = 0$) if a contact is *off*. $A = 1$ causes the contact is switched off, see Fig. 46.

Figure 46. The ladder diagram NOT gate.



1.2. Logic circuits

Thus, on plasmodia we can define logic circuits. In the meanwhile, output logic values are represented in Table 2 and input logic values are represented in Table 3.

Table 2. Output logic values.






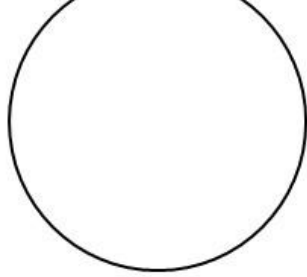

Boolean value	Representation
0	Attractant/repellent deactivated
1	Attractant/repellent activated

Table 3. Input logic value.

Boolean value	Representation
0	Absence of <i>Physarum</i> at the attractant
1	Presence of <i>Physarum</i> at the attractant

For designing logic circuits on slime mould we use some diagrams, their basic symbols are defined in Table 4.

Table 4. Basic symbols used in diagrams for logic circuits.

Symbol	Meaning
	<i>Physarum</i>
	Attractant deactivated
	Attractant activated
	Repellent deactivated
	Repellent activated
	Region of influence
	Direction of plasmodium propagation

A distribution of stimuli for the *AND gate*: the plasmodium of *Physarum polycephalum* P can be propagated to the output attractant A_y if and only if both attractants A_{x1} and A_{x2} are activated, see Fig. 47 and Table 5.

Figure 47. Distribution of stimuli for the AND gate.

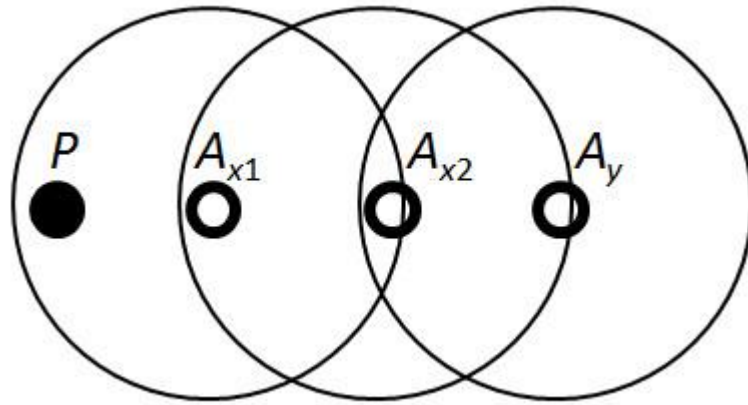
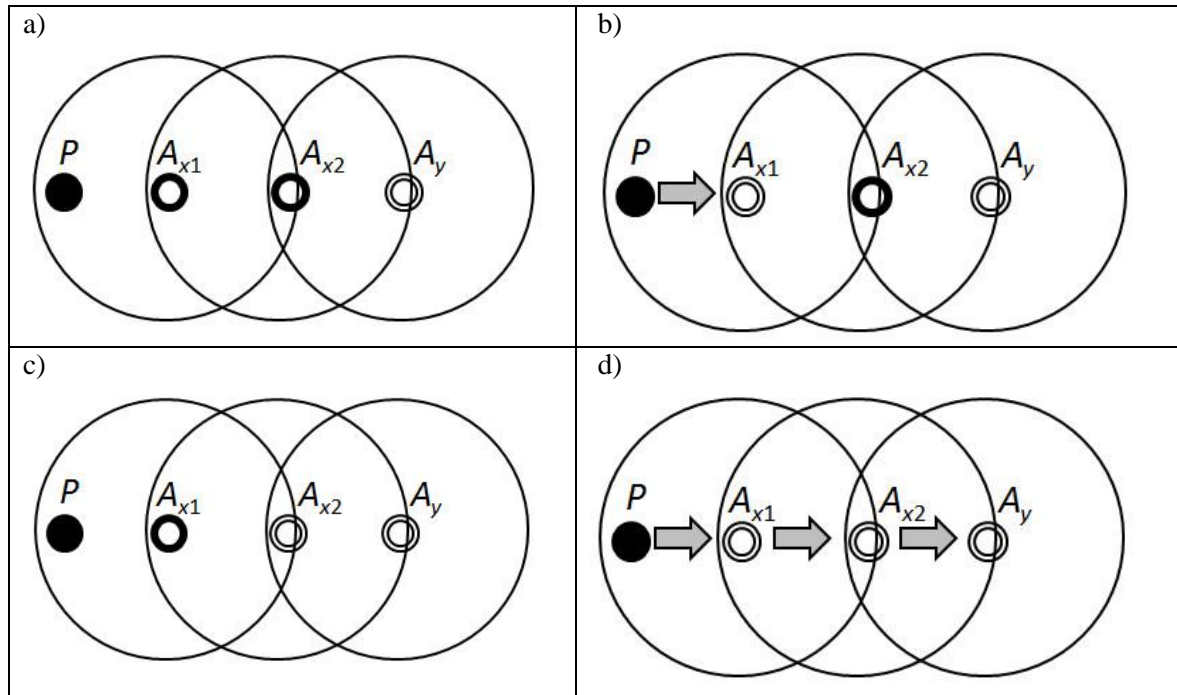


Table 5. States of the AND gate for all input combinations.



A distribution of stimuli for the *OR gate*: the plasmodium of *Physarum polycephalum* P can be propagated to the output attractant A_y if one of attractants A_{x1} or A_{x2} is activated, see Fig. 48 and Table 6.

Figure 48. Distribution of stimuli for the OR gate.

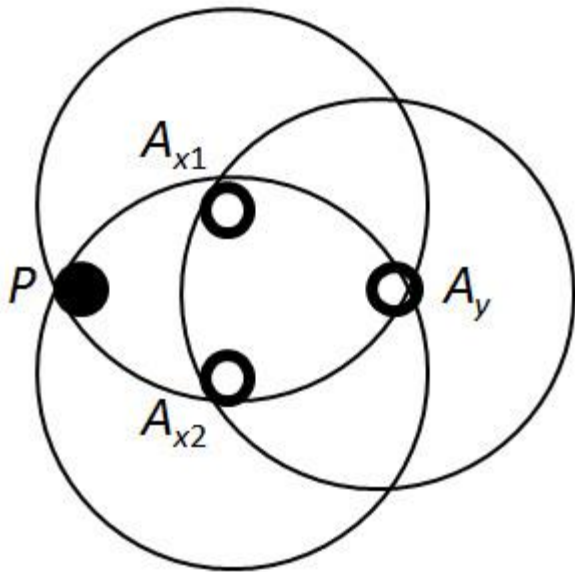
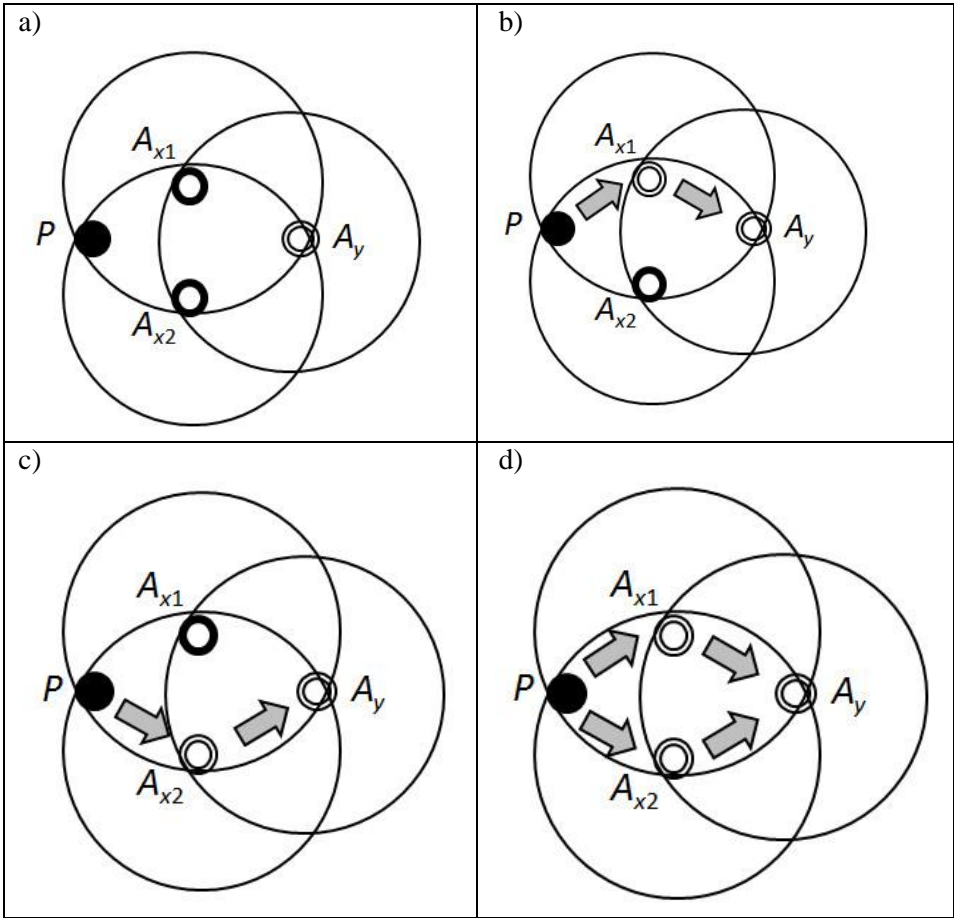


Table 6. States of the OR gate for all input combinations.



The NOT gate behaviour is simulated by the repellent. A distribution of stimuli for the NOT gate: if the repellent R_x is activated (i.e., the input value is 1), then it avoids plasmodium to be attracted by the output attractant A_y , see Fig. 49 and Table 7.

Figure 49. Distribution of stimuli for the NOT gate.

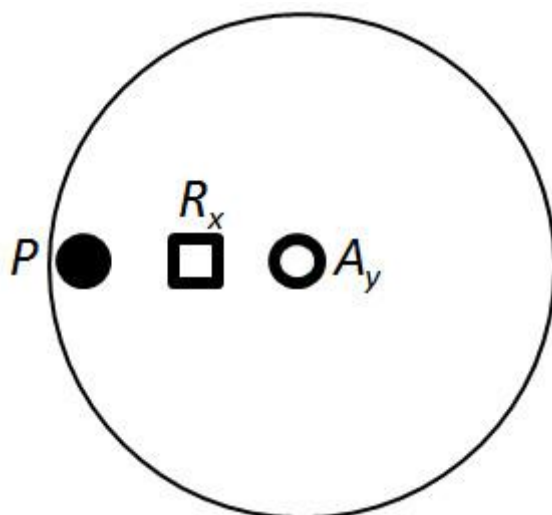
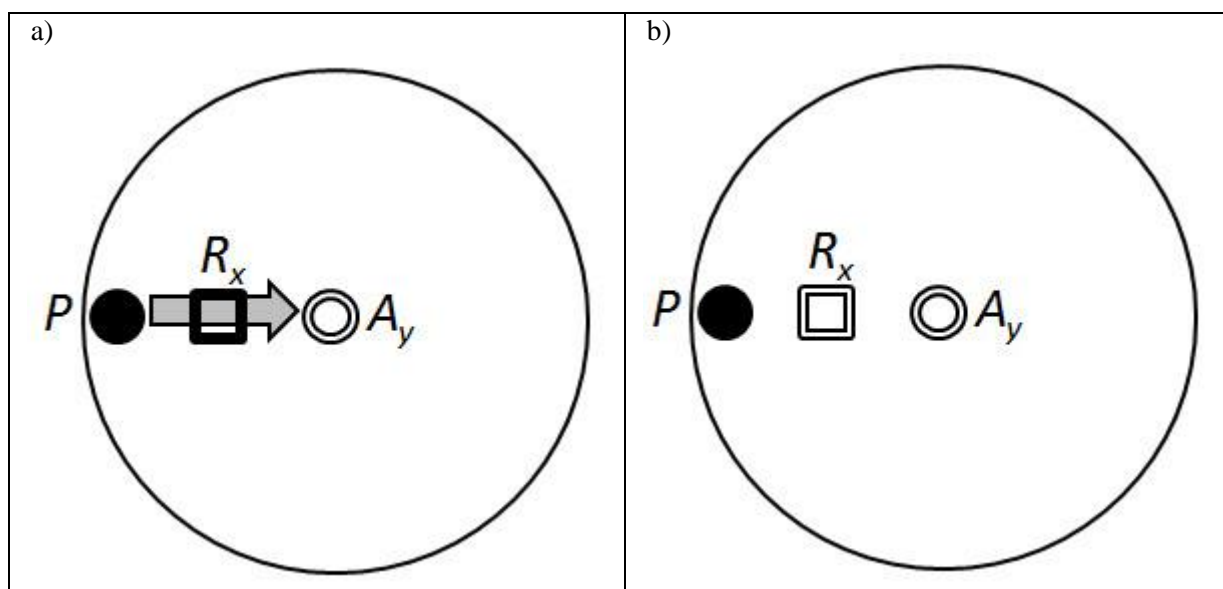


Table 7. States of the NOT gate for all input combinations.



1.3. Petri net models

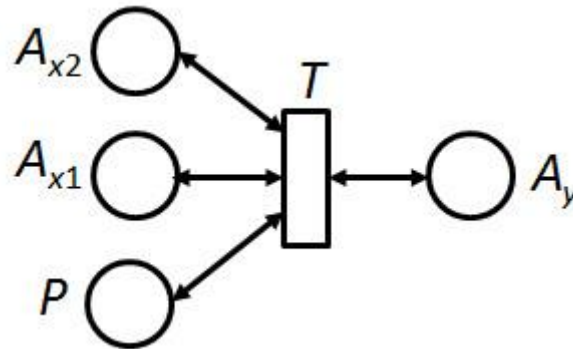
In our approach, we use Petri nets with inhibitor arcs. Inhibitor arcs are used to test for absence of tokens in a place. A transition can only fire if all its inhibitor places are empty. Graphically, an inhibitor arc connects a place to a transition and the arc ends with an empty circle on the transition side. Moreover, a capacity limit equal to 1 is assigned to each place.

A marked Petri net with inhibitor arcs is a five-tuple $MPN = (Pl, Tr, A, w, m)$, where:

- Pl is the finite set of places,
- Tr is the finite set of transitions,
- $A = A_o \cup A_i$ such that $A_o \subseteq (Pl \times Tr) \cup (Tr \times Pl)$ is the set of ordinary arcs from places to transitions and from transitions to places whereas $A_i \subseteq (Pl \times Tr)$ is the set of inhibitor arcs from places to transitions,
- $w: A \rightarrow \{1, 2, 3, \dots\}$ is the weight function on the arcs,
- $m: Pl \rightarrow \{0, 1, 2, \dots\}$ is the initial marking function on the places.

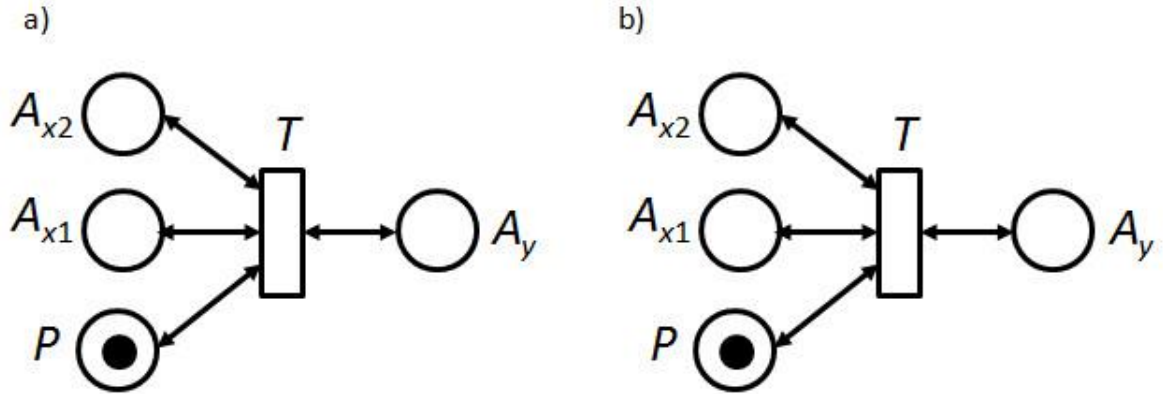
The Petri net model of the AND gate for *Physarum polycephalum* computing is pictured in Fig. 50.

Figure 50. The Petri net model of the AND gate for *Physarum polycephalum* computing: A_{x_1} is a place representing the attractant controlled by the input x_1 ; A_{x_2} is a place representing the attractant controlled by the input x_2 ; A_y is a place representing the attractant corresponding to the output y , P is a place representing *Physarum polycephalum*.



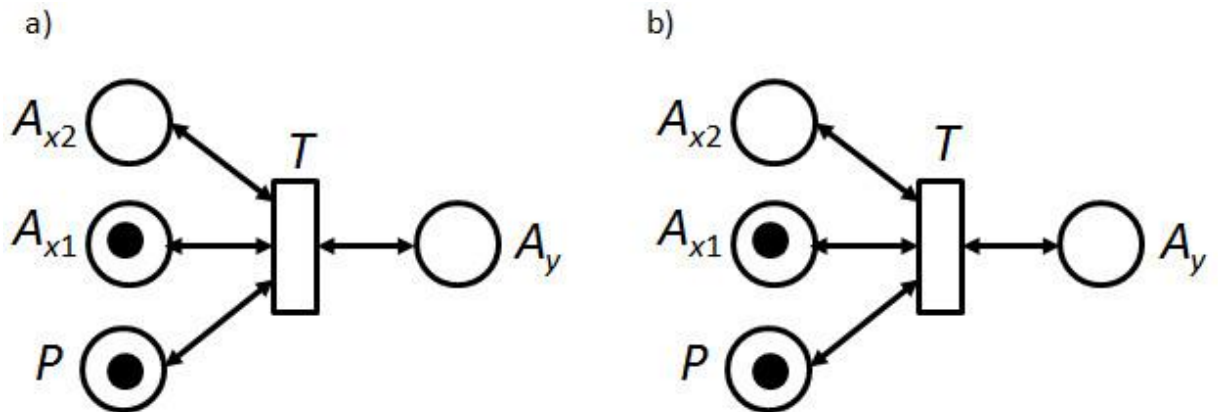
The Petri net model behaviour of the AND gate for $x_1 = 0$ and $x_2 = 0$, (a) before transition firing, (b) after transition firing is pictured in Fig. 51.

Figure 51. Petri net model behaviour of the AND gate (0, 0).



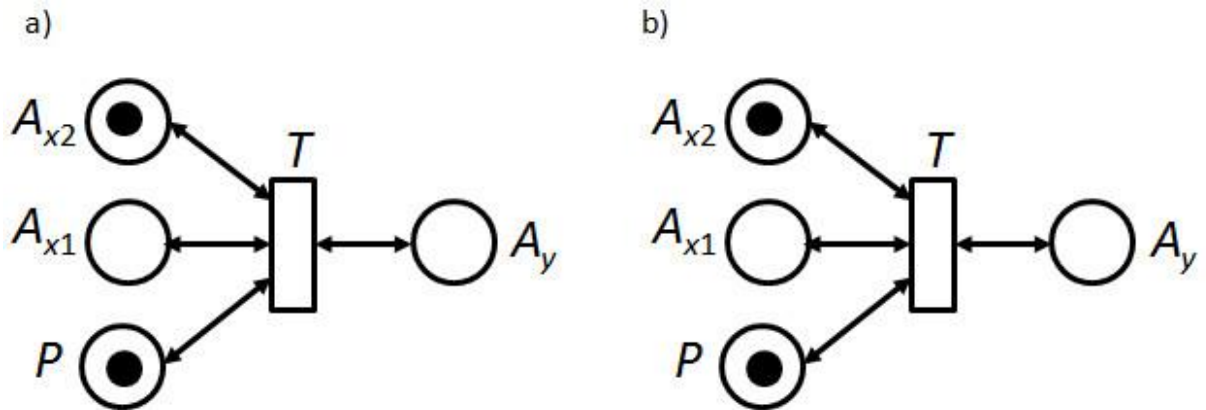
The Petri net model behaviour of the AND gate for $x_1 = 1$ and $x_2 = 0$, (a) before transition firing, (b) after transition firing is pictured in Fig. 52.

Figure 52. Petri net model behaviour of the AND gate (1, 0).



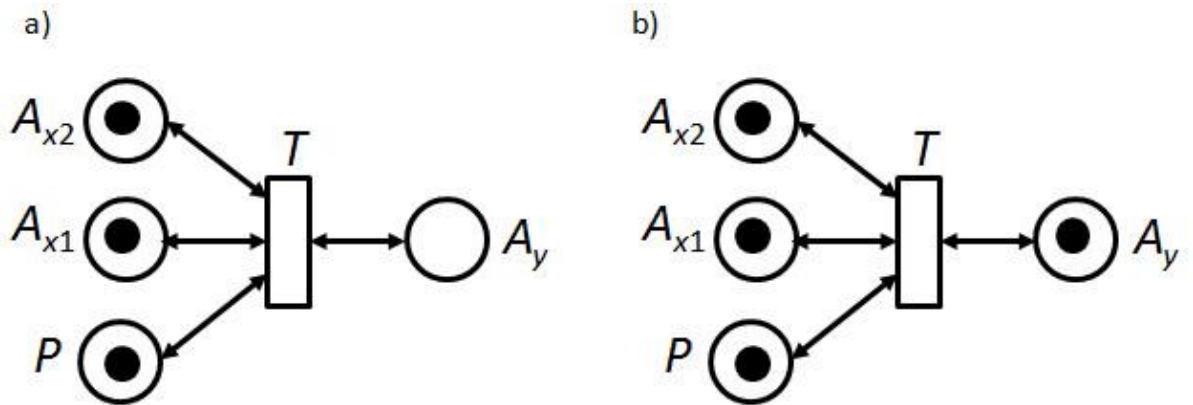
The Petri net model behaviour of the AND gate for $x_1 = 0$ and $x_2 = 1$, (a) before transition firing, (b) after transition firing is pictured in Fig. 53.

Figure 53. Petri net model behaviour of the AND gate (0, 1).



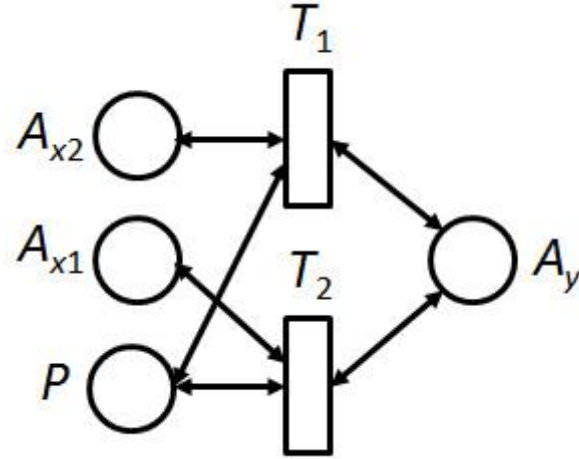
The Petri net model behaviour of the AND gate for $x_1 = 1$ and $x_2 = 1$, (a) before transition firing, (b) after transition firing is pictured in Fig. 54.

Figure 54. Petri net model behaviour of the AND gate (1, 1).



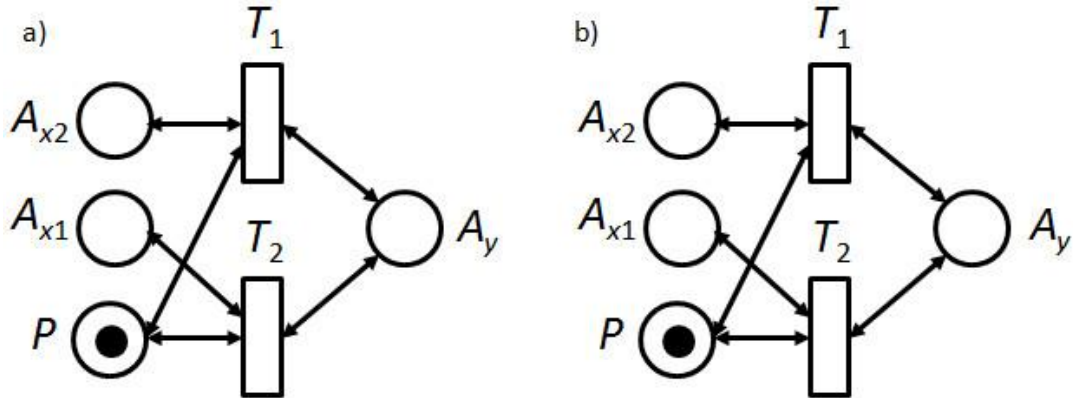
The Petri net model of the OR gate for *Physarum polycephalum* computing is pictured in Fig. 55.

Figure 55. Petri net model of the OR gate for *Physarum polycephalum* computing: A_{x_1} is a representing the attractant controlled by the input x_1 ; A_{x_2} is a place representing the attractant controlled by the input x_2 ; A_y is a place representing the attractant corresponding to the output y ; P is a place representing *Physarum polycephalum*.



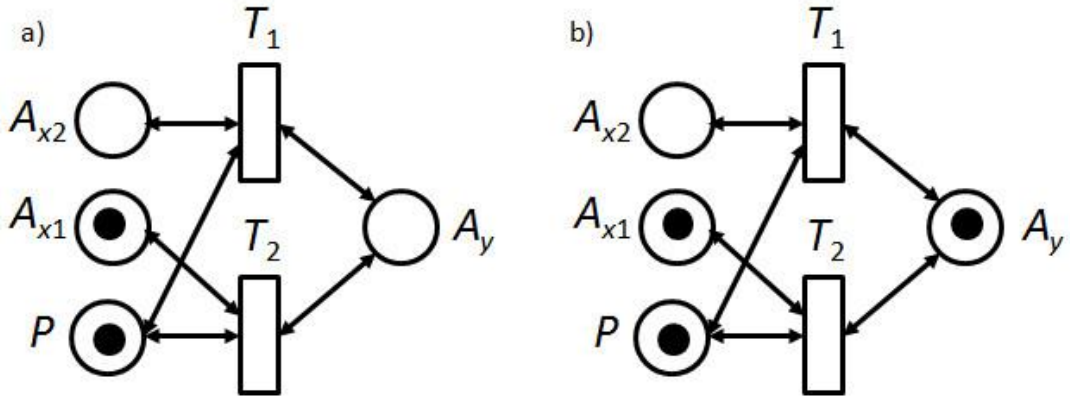
The Petri net model behaviour of the OR gate for $x_1 = 0$ and $x_2 = 0$, (a) before transition firing, (b) after transition firing is pictured in Fig. 56.

Figure 56. Petri net model behaviour of the OR gate (0, 0).



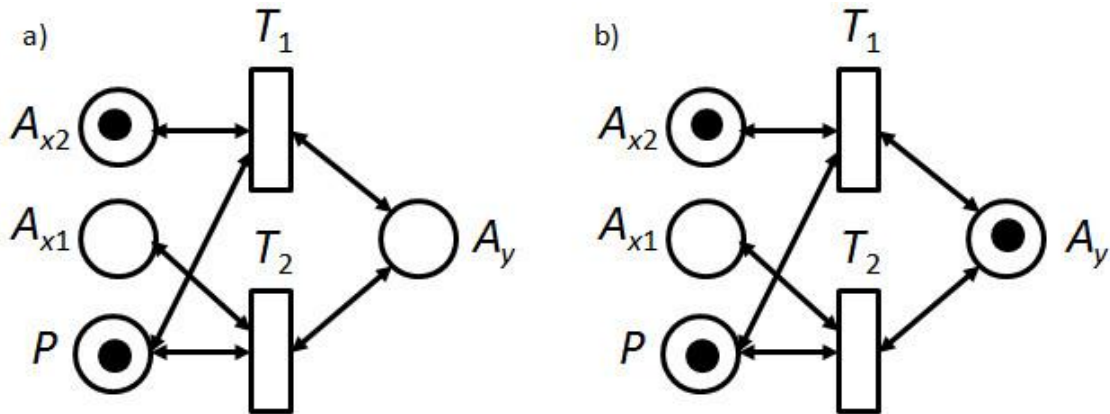
The Petri net model behaviour of the OR gate for $x_1 = 1$ and $x_2 = 0$, (a) before transition firing, (b) after transition firing is pictured in Fig. 57.

Figure 57. Petri net model behaviour of the OR gate (1, 0).



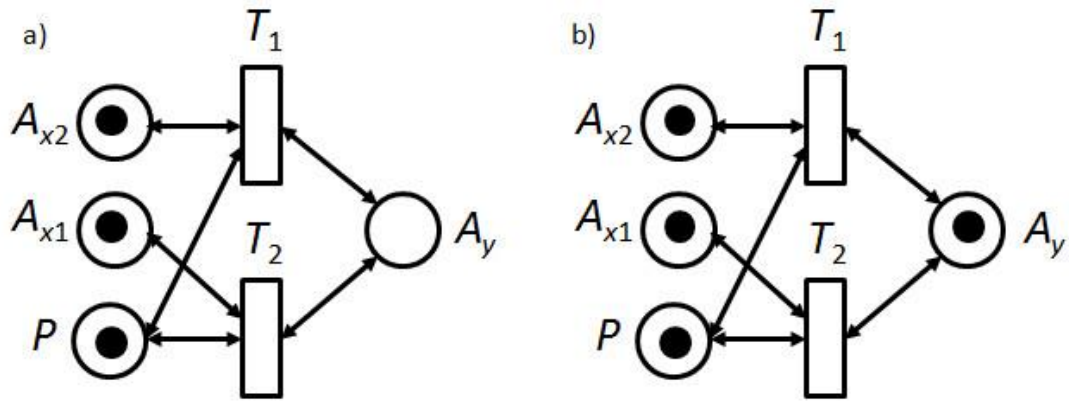
The Petri net model behaviour of the OR gate for $x_1 = 0$ and $x_2 = 1$, (a) before transition firing, (b) after transition firing is pictured in Fig. 58.

Figure 58. Petri net model behaviour of the OR gate (0, 1).



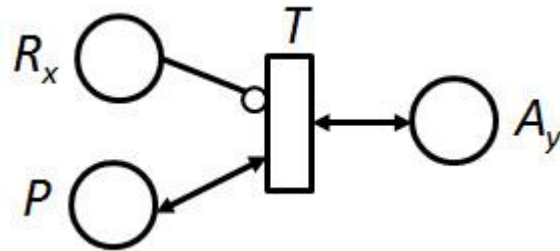
The Petri net model behaviour of the OR gate for $x_1 = 1$ and $x_2 = 1$, (a) before transition firing, (b) after transition firing is pictured in Fig. 59.

Figure 59. Petri net model behaviour of the OR gate (1, 1).



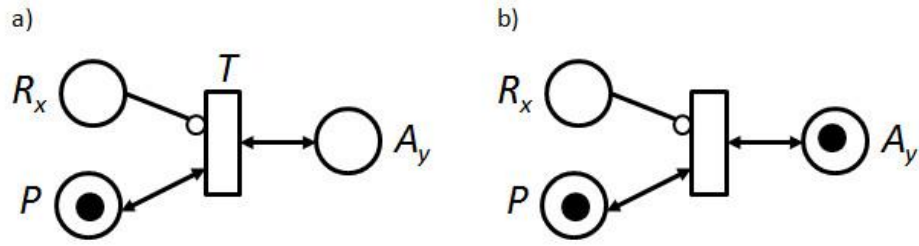
The Petri net model of the NOT gate for *Physarum polycephalum* computing is pictured in Fig. 60.

Figure 60. The Petri net model of the NOT gate for *Physarum polycephalum* computing: A_x is a place representing the attractant controlled by the input x ; A_y is a place representing the attractant corresponding to the output y ; P is a place representing *Physarum polycephalum*.



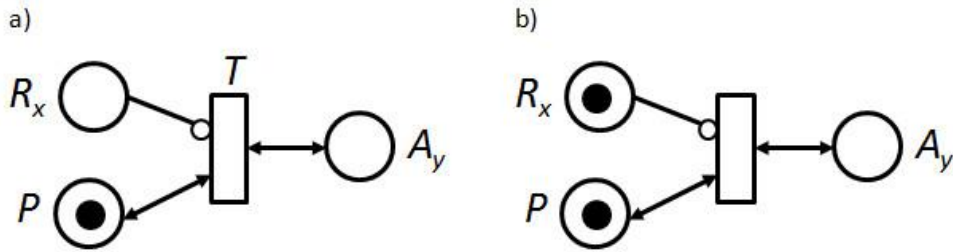
The Petri net model behaviour of the NOT gate for $x=0$, (a) before transition firing, (b) after transition firing is pictured in Fig. 61.

Figure 61. Petri net model behaviour of the NOT gate (0).



The Petri net model behaviour of the NOT gate for $x=1$, (a) before transition firing, (b) after transition firing is pictured in Fig. 62.

Figure 62. Petri net model behaviour of the NOT gate (1).



1.4. Example: 1-to-2 demultiplexer

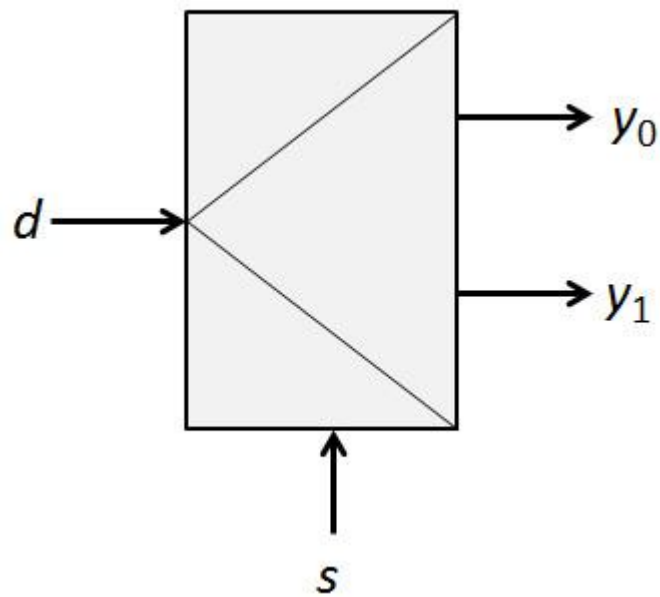
Using ladder diagrams and Petri net diagrams we can design different circuits like 1-to-2 demultiplexer, see Fig. 63. This operation can be described as follows:

if $s = 0$, then $y_0 = d$,

if $s = 1$, then $y_1 = d$.

The functional specification can be written as $y_0 = \bar{s}d$ and $y_1 = sd$.

Figure 63.1-to-2 demultiplexer.



The ladder diagram model of the 1-to-2 demultiplexer is pictured in Fig. 64, its Petri net model is pictured in Fig. 65.

Figure 64.1-to-2 demultiplexer by the ladder diagram.

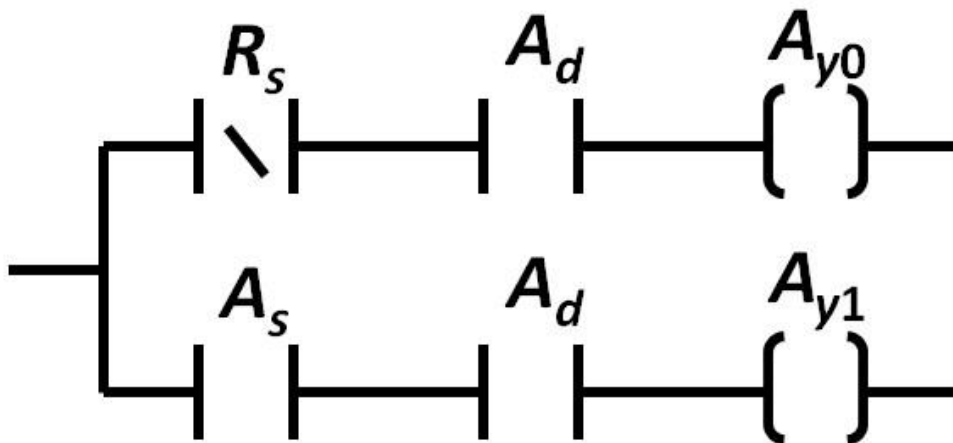
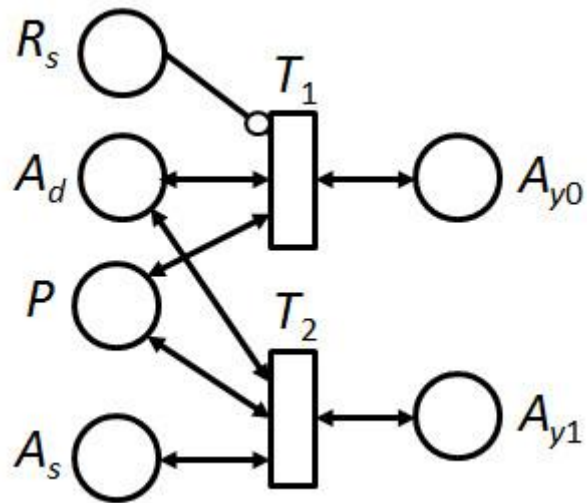


Figure 65. Petri net model of the 1-to-2 demultiplexer.



The Petri net model behaviours of the 1-to-2 demultiplexer for different s and d are pictured in Fig. 66 –69.

Figure 66. Petri net model behaviour of the 1-to-2 demultiplexer for $s=0$ and $d=0$.

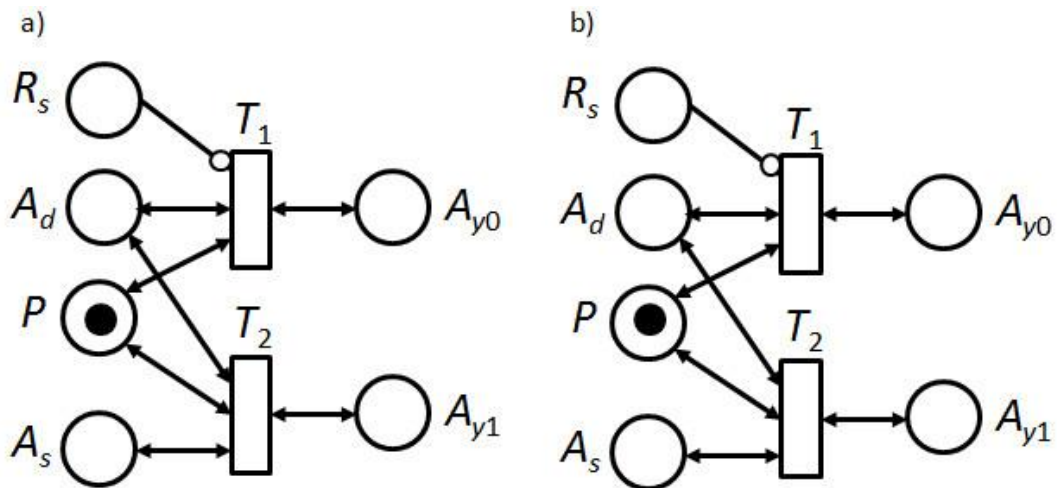


Figure 67. Petri net model behaviour of the 1-to-2 demultiplexer for $s=0$ and $d=1$.

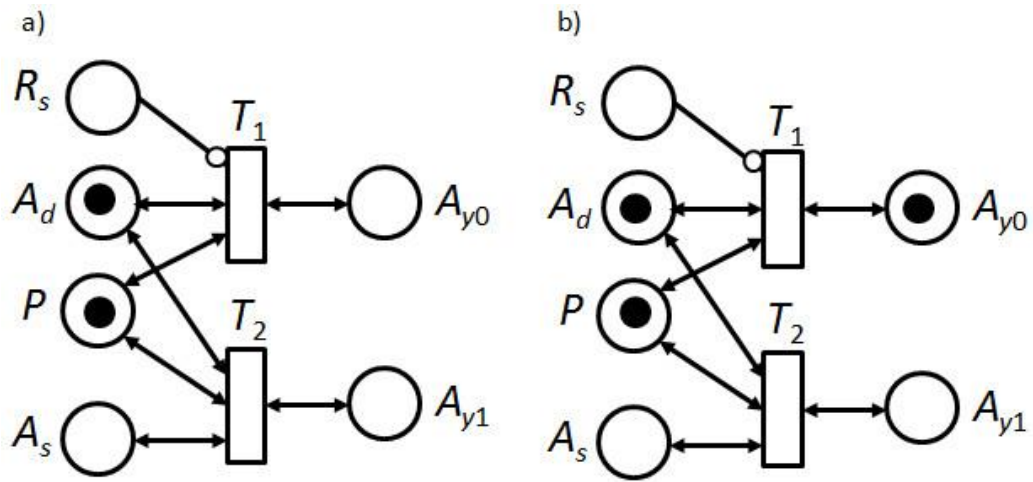


Figure 68. Petri net model behaviour of the 1-to-2 demultiplexer for $s=1$ and $d=0$.

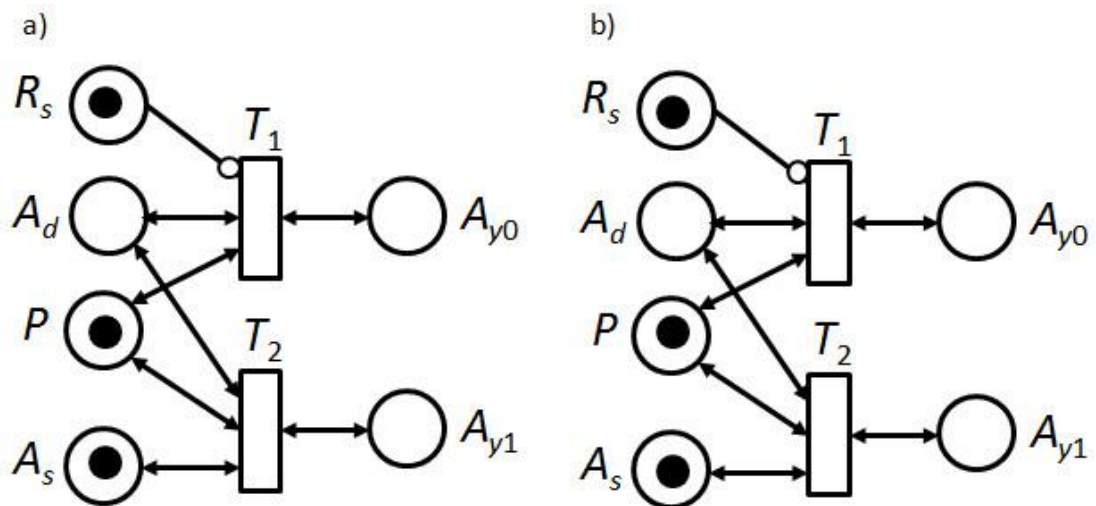
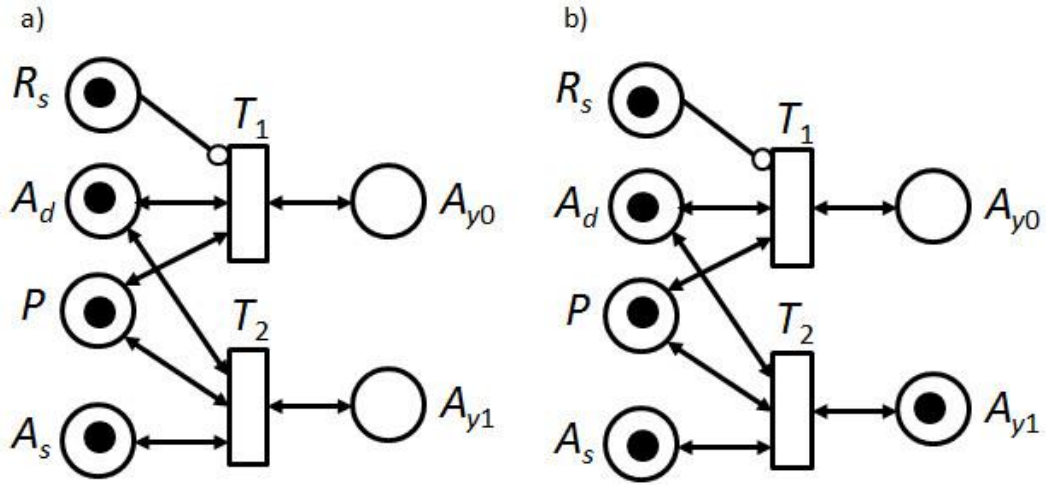
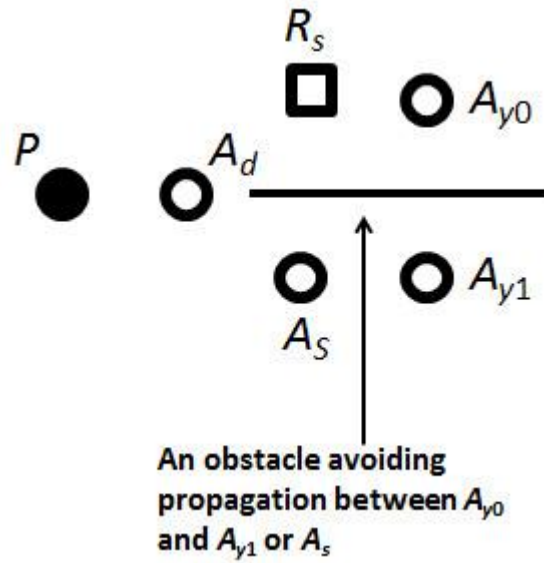


Figure 69. Petri net model behaviour of the 1-to-2 demultiplexer for $s=1$ and $d=1$.



A distribution of stimuli for the 1-to-2 demultiplexer is represented in Fig. 70.

Figure 70. Distribution of stimuli for the 1-to-2 demultiplexer.



The experimental environment for implementation of the 1-to-2 demultiplexer using a particle model of *Physarum polycephalum* is given in Fig. 71 – 72.

Figure 71. Experimental environment for implementation 1-to-2 demultiplexer.

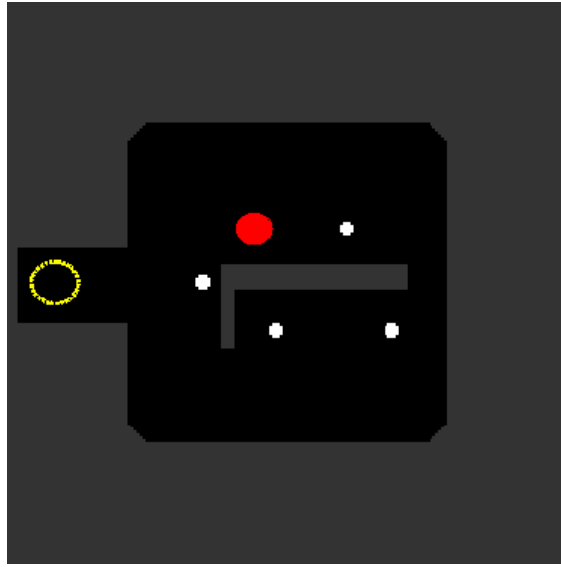
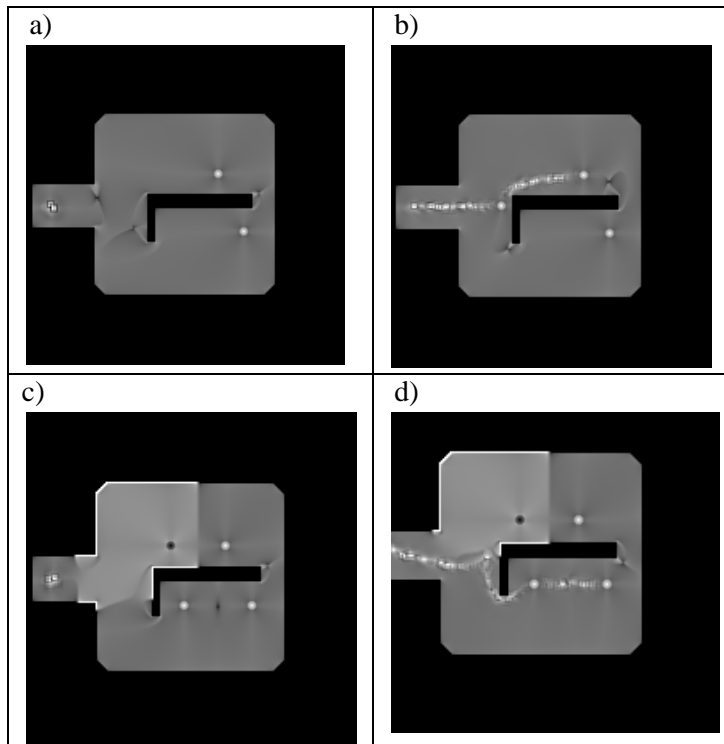


Figure 72. Results of experiments: (a) for $s = 0$ and $d = 0$, (b) for $s = 0$ and $d = 1$, (c) for $s = 1$ and $d = 0$, (d) for $s = 1$ and $d = 1$.



1.5. Example: half adder

The half adder adds two single binary digits a and b . It has two outputs, sum (s) and carry (c). The functional specification of the half adder can be written as:

$$s = \bar{a}b + a\bar{b},$$

$$c = ab.$$

Its ladder diagram is pictured in Fig. 73, its Petri net model is pictured in Fig. 74, and a spatial distribution of attractants and repellents for the half adder is pictured in Fig. 75.

Figure 73. The ladder diagram model of the half adder.

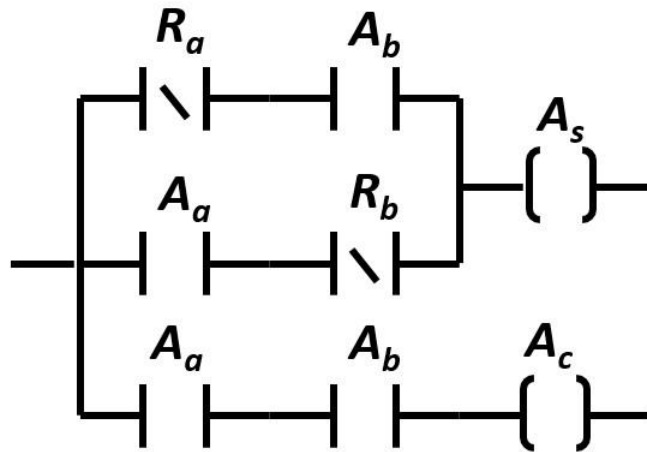


Figure 74. The Petri net model of the half adder.

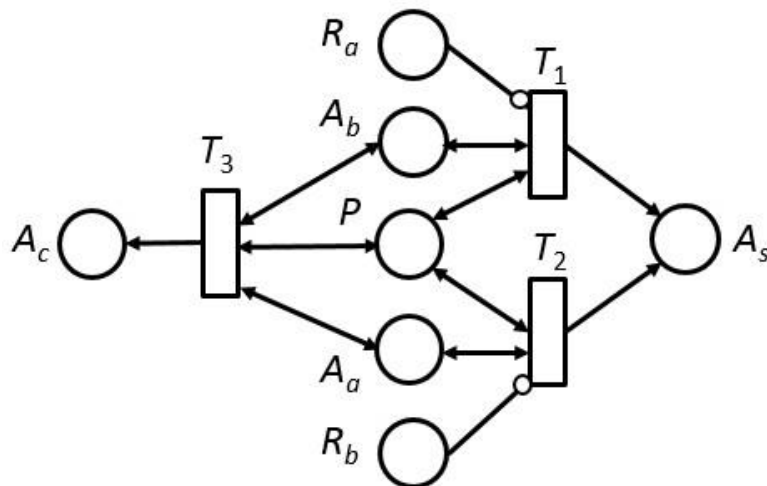
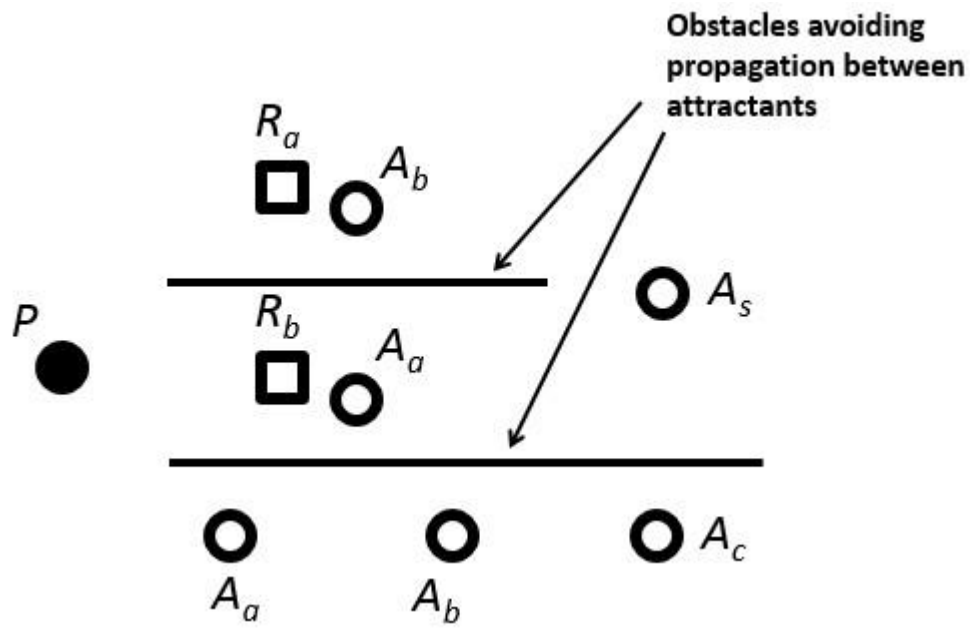


Figure 75. Spatial distribution of attractants and repellents for the half adder.



2. Rough-set extensions of transition systems

2.1. Transition systems

A transition system is a quadruple $TS = (S, E, T, I)$, where:

- S is the non-empty set of states,
- E is the set of events,
- $T \subseteq S \times E \times S$ is the transition relation,
- $I \subseteq S$ is the set of initial states.

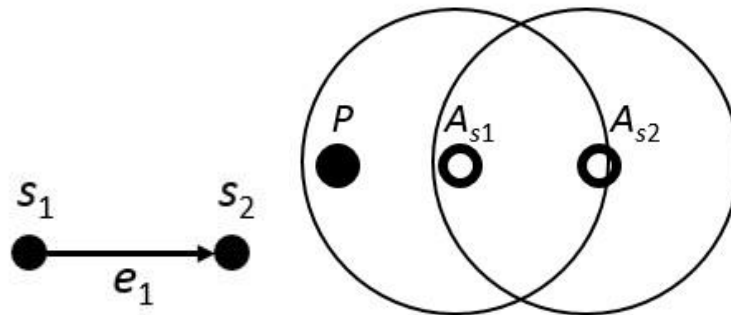
Usually, transition systems are based on actions which may be viewed as labelled events. If $(s, e, s') \in T$ then the idea is that TS can go from s to s' as a result of the event e occurring at s .

Motions of plasmodium of *Physarum polycephalum* are a kind of natural transition systems. States are presented by attractants and events are transitions from one attractant to another. We can define the following three basic forms of *Physarum* transitions (motions): direction, fusion, and splitting.

Direct, direction: a movement from one place, where the plasmodium is located, towards another place, where there is a neighbour attractant, see Fig. 76.

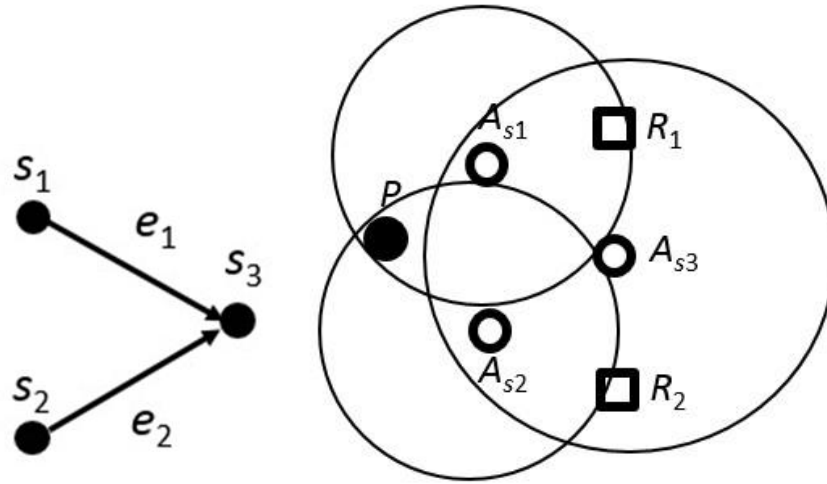
Figure 76. Implementation of the *Direct* motion: If both attractants A_{s_1} and A_{s_2} are activated, then plasmodium propagates from the origin place to the attractant A_{s_1} , and next to the attractant

A_{s_2} .



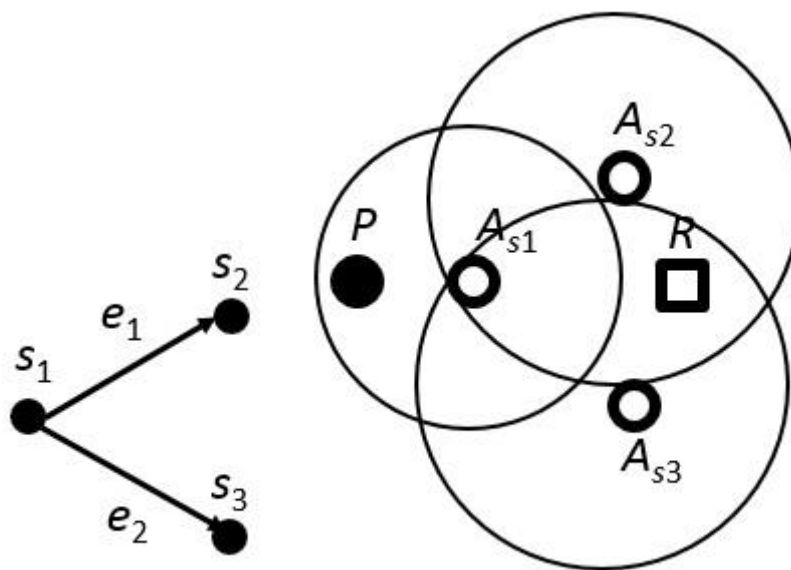
Fuse, fusion of two plasmodia at the place, where they meet the same attractant, is pictured in Fig. 77.

Figure 77. Implementation of the *Fuse* motion: If two plasmodia occupy the attractants A_{s1} and A_{s2} , respectively, then they are turned by the repellents R_1 and R_2 to the attractant A_{s3} , where they are fused.



Split, splitting of plasmodium from one active place into two active places, where two neighbour attractants with a similar power of intensity are located, is represented in Fig. 78.

Figure 78. Implementation of the *Split* motion: if plasmodium occupies the attractant A_{s1} , then it is turned by the repellent R to the attractants A_{s2} and A_{s3} . It means that plasmodium is split at A_{s1} .



A simple example of the transition system $TS = (S, E, T, s_0)$ represents phase changes, see Fig.79, 80.

Figure 79. Phase changes.

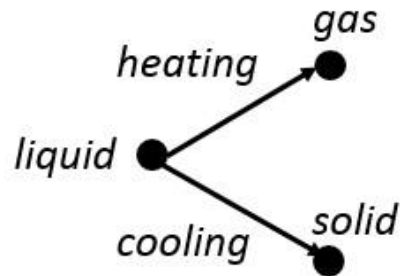
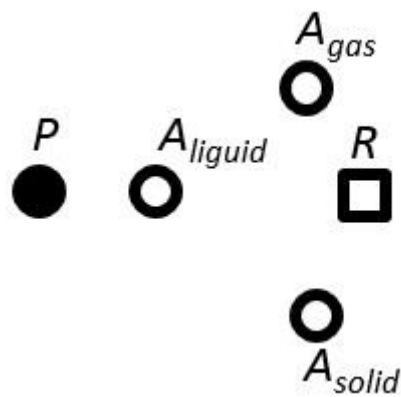


Figure 80. Spatial distribution of attractants and repellents for the transition system TS of Fig. 79.



2.2. Non-deterministic transition systems and rough sets

If there exists the state $s \in S$ in the transition system TS such that $\text{card}(\text{Post}(s)) > 1$, where card the cardinality of the set is, then TS is called a *non-deterministic transition system*. One can see that in non-deterministic transition systems we deal with ambiguity of direct successors of some states, i.e., there exist states having no uniquely determined direct successors. In the presented approach, we propose to manage this ambiguity using rough set theory.

Rough sets were proposed in: Pawlak Z.: *Rough Sets. Theoretical Aspects of Reasoning about Data*, Kluwer Academic Publishers, Dordrecht 1991.

The Variable Precision Rough Set Model (VPRSM) was proposed in: Ziarko, W.: *Variable precision rough set model*, "Journal of Computer and System Sciences", 46(1), pp. 39-59, 1993.

Let us define rough sets. Let $U \neq \emptyset$ be a finite set of objects we are interested in, R be any equivalence relation over U , and $[u]_R$ be an equivalence class of any $u \in U$. With each subset $X \subseteq U$ and any equivalence relation R over U , we associate two subsets:

$$\underline{R}(X) = \{u \in U : [u]_R \subseteq X\},$$

$$\overline{R}(X) = \{u \in U : [u]_R \cap X \neq \emptyset\},$$

called the R -lower and R -upper approximation of X , respectively.

Standard sets are defined by inclusion. Let U be the universe and $A, B \subseteq U$. The standard set inclusion is defined as

$$A \subseteq B \quad \text{if and only if} \quad \forall_{u \in A} u \in B.$$

A majority set inclusion is defined as follows. Let U be the universe, $A, B \subseteq U$ and $0 \leq \beta < 0.5$. The majority set inclusion is defined thus:

$$A \overset{\beta}{\subseteq} B \quad \text{if and only if} \quad 1 - \frac{\text{card}(A \cap B)}{\text{card}(A)} \leq \beta,$$

W. Ziarko proposed some relaxation of the original rough set approach, called the Variable Precision Rough Set Model (VPRSM). The VPRSM approach is based on the notion of majority set inclusion.

By replacing the standard set inclusion with the majority set inclusion in definitions of approximations, we obtain the following two subsets:

$$\underline{R}^\beta(X) = \{u \in U : [u]_R \overset{\beta}{\subseteq} X\},$$

$$\overline{R}^\beta(X) = \{u \in U : \frac{\text{card}([u]_R \cap X)}{\text{card}([u]_R)} > \beta\},$$

called the R_β -lower and R_β -upper approximation of X , respectively.

For each state $s \in S$ in the transition system TS , we can determine its direct successors and predecessors. Let:

$$Post(s, e) = \{s' \in S : (s, e, s') \in T\},$$

$$Pre(s, e) = \{s' \in S : (s', e, s) \in T\},$$

then the set $Post(s)$ of all *direct successors* of the state $s \in S$ is given by

$$Post(s) = \bigcup_{e \in E} Post(s, e)$$

and the set $Pre(s)$ of all *direct predecessors* of the state $s \in S$ is given by

$$Pre(s) = \bigcup_{e \in E} Pre(s, e).$$

Let $TS = (S, E, T, I)$ be a transition system and $X \subseteq S$. The lower predecessor anticipation $Pre_*(X)$ of X is given by

$$Pre_*(X) = \{s \in S : Post(s) \neq \emptyset \text{ and } Post(s) \subseteq X\}.$$

The lower predecessor anticipation consists of all states from which TS surely goes to the states in X as results of any events occurring at these states.

Let $TS = (S, E, T, I)$ be a transition system and $X \subseteq S$. The *upper predecessor anticipation* $Pre^*(X)$ of X is given by

$$Pre^*(X) = \{s \in S : Post(s) \cap X \neq \emptyset\}.$$

The upper predecessor anticipation consists of all states from which TS possibly goes to the states in X as results of some events occurring at these states. It means that TS can also go to the states from outside X .

If we use majority set inclusion, we obtain the generalized notion of the β -lower predecessor anticipation. Let $TS = (S, E, T, I)$ be a transition system and $X \subseteq S$. The β -lower predecessor anticipation $Pre_*^\beta(X)$ of X is given by

$$Pre_*^\beta(X) = \{s \in S : Post(s) \neq \emptyset \text{ and } Post(s) \overset{\beta}{\subseteq} X\}.$$

The β -lower predecessor anticipation consists of each state from which TS goes, in most cases to the states in X as results of events occurring at these states.

2.3. Timed transition systems and rough sets

Timed Transition Systems $TTS = (S, E, T, I, l, u)$ consists of:

- an underlying transition system $TS = (S, E, T, I)$,
- a minimal delay function (a lower bound) $l: E \rightarrow N$ assigning a nonnegative integer to each event,
- a maximal delay function (an upper bound) $u: E \rightarrow N \cup \{\infty\}$ assigning a nonnegative integer or infinity to each event.

We assume, for timed transition systems, that the events may occur only at discrete time instants. Therefore, whenever time instant t is used, it means that $t = 0, 1, 2, \dots$

Let $TTS = (S, E, T, I, l, u)$ be a timed transition system. Let

$$Post_t(s, e) = \{s' \in S : (s, e, s') \in T \wedge l(e) \leq t \leq u(e)\}$$

and

$$Pre_t(s, e) = \{s' \in S : (s', e, s) \in T \wedge l(e) \leq t \leq u(e)\},$$

then the set $Post_t(s)$ of all direct successors of the state $s \in S$ at t is given by

$$Post_t(s) = \bigcup_{e \in E} Post_t(s, e)$$

and the set $Pre_t(s)$ of all direct predecessors of the state $s \in S$ at t is given by

$$Pre_t(s) = \bigcup_{e \in E} Pre_t(s, e).$$

If there exists the state $s \in S$ in the timed transition system TTS at the time instant t such that $card(Post_t(s)) > 1$, where $card$ is the cardinality of the set, then TTS is called a non-deterministic timed transition system at t . In non-deterministic timed transition systems, we deal with ambiguity of direct successors of some states, i.e., at some time instants, there exist states having no uniquely determined direct successors.

Let $TTS = (S, E, T, I, l, u)$ be a timed transition system and $X \subseteq S$. The lower predecessor anticipation $\underline{Pre}_t(X)$ of X at the time instant t is given by

$$\underline{Pre}_t(X) = \{s \in S : Post_t(s) \neq \emptyset \wedge Post_t(s) \subseteq X\}.$$

The lower predecessor anticipation $\underline{Pre}_t(X)$ consists of all states from which TTS surely goes to the states in X as results of any events occurring at these states at the time instant t .

Let $TTS = (S, E, T, I, l, u)$ be a timed transition system and $X \subseteq S$. The upper predecessor anticipation $\overline{Pre}_t(X)$ of X at the time instant t is given by

$$\overline{Pre}_t(X) = \{s \in S : Post_t(s) \cap X \neq \emptyset\}.$$

The upper predecessor anticipation $\overline{Pre}_t(X)$ consists of all states from which TTS possibly goes to the states in X as results of some events occurring at these states at the time instant t . It means that TTS can also go at t to the states from outside X .

The β -lower predecessor anticipation $\underline{Pre}_t^\beta(X)$ of X at the time instant t is given by

$$\underline{Pre}_t^\beta(X) = \{s \in S : Post_t(s) \neq \emptyset \wedge Post_t(s) \overset{\beta}{\subseteq} X\}.$$

The β -lower predecessor anticipation of X at t consists of each state from which TTS goes, in most cases (i.e., in terms of the majority set inclusion) to the states in X as results of events occurring at these states at the time instant t .

If

$$\forall_{t \in \{0,1,2,\dots\}} s \in \underline{Pre}_t(X),$$

then s is said to be a continuous strict anticipator of states from X . It means that s always anticipates (i.e., at each time instant) states from X .

If s is not a continuous strict anticipator of states from X , but

$$\exists_{t \in \{0,1,2,\dots\}} s \in \underline{Pre}_t(X),$$

then s is said to be an interim strict anticipator of states from X . It means that s sometimes (not always) anticipates states from X .

If s is not a continuous and interim strict anticipator of states from X , but

$$\bigvee_{t \in \{0,1,2,\dots\}} s \in \underline{Pre}_t^\beta(X),$$

then s is said to be a continuous quasi-anticipator of states from X .

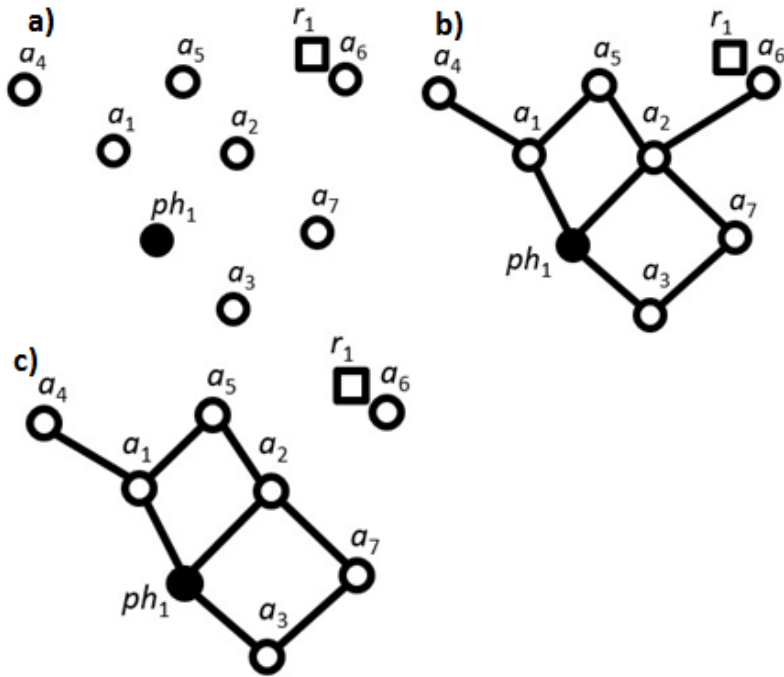
If s is not a continuous and interim strict anticipator and continuous quasi-anticipator of states from X , but

$$\bigvee_{t \in \{0,1,2,\dots\}} s \in \underline{Pre}_t^\beta(X),$$

then s is said to be an interim quasi-anticipator of states from X .

Let us consider an exemplary *Physarum* machine $\mathbf{PM} = \{P, A, R\}$, where $P = \{ph_1\}$, $A = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$, and $R = \{r_1\}$, see Fig. 81.

Figure 81. Example of *Physarum* machine.



Another example. Let us assume that we are interested in the set $X = \{s_5, s_6, s_8\}$ of goal states and $\beta = 0.5$, see Fig. 82. For X and $t < 5$, we obtain:

$$\underline{Pre}_t(X) = \{s_2, s_4\},$$

$$\overline{Pre}_t(X) = \{s_2, s_3, s_4\},$$

$$\text{but } \underline{Pre}_t^{0.5}(X) = \{s_2, s_3, s_4\}.$$

For X and $t \geq 5$, we obtain:

$$\underline{Pre}_t(X) = \{s_2, s_3, s_4\},$$

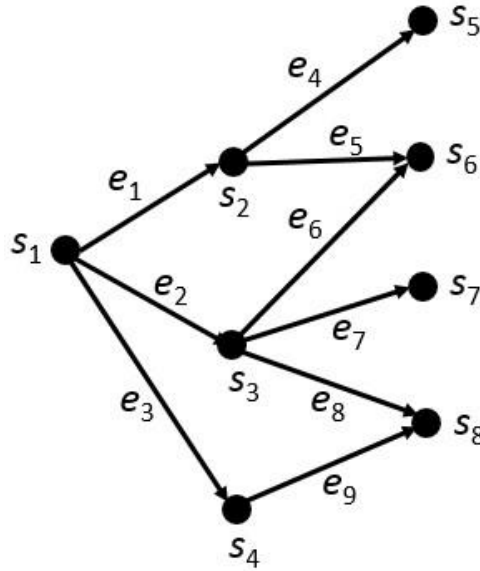
$$\overline{Pre}_t(X) = \{s_2, s_3, s_4\}.$$

It means that s_2 and s_4 are continuous strict anticipators of states from X , s_3 is an interim strict anticipator of states from X but also s_3 is a continuous quasi-anticipator of states from X .

Figure 82. A timed transition system model of PM:

$$l(e_1) = l(e_2) = l(e_3) = l(e_4) = l(e_5) = l(e_6) = l(e_7) = l(e_8) = l(e_9) = 0,$$

$$u(e_1) = u(e_2) = u(e_3) = u(e_4) = u(e_5) = u(e_6) = u(e_8) = u(e_9) = \infty, \text{ and } u(e_7) = 4.$$



2.4. Models of simple rule-based systems

We show that biological substrate in the form of *Physarum polycephalum* can be used to simulate simple rule-based systems. To extort a proper behaviour from the substrate, appropriate distribution of stimuli (attractants and/or repellents) is required. A biological computing device implemented in the plasmodium of *Physarum polycephalum* is said to be a *Physarum* machine. To model behaviour of the substrate and then program *Physarum* machine, we propose to use Petri net models that can be treated as high-level description.

There are various knowledge representation methods that have been developed to make real-world knowledge suitable for being processed by computers. One of the most popular knowledge representation systems are the rule-based ones. Rules can be easily interpreted by humans. Formally, rules can be presented in the framework of propositional logics.

Four types of the composite production rules can be distinguished:

Type 1: IF d_{i_1} AND d_{i_2} AND ... AND d_{i_k} , THEN d_j .

Type 2: IF d_i , THEN d_{j_1} AND d_{j_2} AND ... AND d_{j_k} .

Type 3: IF d_{i_1} OR d_{i_2} OR ... OR d_{i_k} , THEN d_j .

Type 4: IF d_i , THEN d_{j_1} OR d_{j_2} OR ... OR d_{j_k} .

Source: Looney, C., Alfize, A.: *Logical controls via Boolean rule matrix transformations*. *IEEE Transactions on Systems, "Man and Cybernetics"* 17(6), pp. 1077-1082, 1987.

In the literature, one can find a lot of approaches using Petri nets as a model of rule-based systems:

- Chen, S.M., Ke, J.S., Chang, J.F.: *Knowledge representation using fuzzy Petri nets*, "IEEE Transactions on Knowledge and Data Engineering" 2(3), pp. 311-319, 1990.
- Konar, A.: *Cognitive Engineering: A Distributed Approach to Machine Intelligence*, Springer-Verlag, London 2005.
- Suraj, Z.: *A survey of selected classes of logical Petri nets*, [in:] Wróbel, Z., Marszał-Paszek, B., Paszek, P. (eds.): *Monografia jubileuszowa dla uczczenia 45 lat pracy naukowej Prof. A. Wakulicz-Deji.*, Wyd. Uniwersytetu Śląskiego, pp. 157-181, Katowice 2013.
- Szpyrka, M., Szmuc, T.: *D-nets - Petri net form of rule-based systems*, "Foundations of Computing and Decision Sciences" 31(2), pp. 175-190, 2006.

Structures of Petri nets reflect structures of rule-based systems. Petri net models enable us to reflect propagation of protoplasmic veins of the plasmodium in consecutive time instants (step by step). Petri nets are used by us as one of the high-level models in our new object-oriented programming language, called the *Physarum* language, for *Physarum* machines, see:

- Schumann, A., Pancerz, K.: *Towards an object-oriented programming language for Physarum polycephalum computing: A Petri net model approach*, Fundamenta Informaticae 133(2-3), pp. 271-285, 2014.
- Pancerz, K., Schumann, A.: *Some issues on an object-oriented programming language for Physarum machines*, [in:] Bris, R., Majernik, J., Pancerz, K., Zaitseva, E. (eds.): *Applications of Computational Intelligence in Biomedical Technology*, Studies in Computational Intelligence, vol. 606, Springer International Publishing, pp. 185-199, Switzerland 2016.

In the proposed Petri net models of *Physarum* machines, we can distinguish several kinds of places:

- Places representing *Physarum polycephalum*.
- Places representing control stimuli (attractants or repellents) corresponding to propositions in antecedent parts of rules.
- Places representing auxiliary stimuli (attractants) corresponding to partial results of evaluation of logical expressions in antecedent parts of composite production rules.
- Places representing output stimuli (attractants) corresponding to propositions in consequence parts of rules.

The meaning of tokens in places representing control stimuli is represented in Table 8.

Table 8. Tokens meaning – control stimuli.

Token	Meaning	Evaluation of proposition
Present	Stimulus activated	<i>true</i> (for attractants), <i>false</i> (for repellents)
Absent	Stimulus deactivated	<i>false</i> (for attractants), <i>true</i> (for repellents)

The meaning of tokens in places representing auxiliary stimuli is represented in Table 9.

Table 9. Tokens meaning – auxiliary stimuli.

Token	Meaning	Partial evaluation of expression
Present	Stimulus occupied by plasmodium	<i>true</i>
Absent	Stimulus not occupied by plasmodium	<i>false</i>

The meaning of tokens in places representing output stimuli is represented in Table 10.

Table 10. Tokens meaning – output stimuli.

Token	Meaning	Evaluation of proposition
Present	Stimulus occupied by plasmodium	<i>true</i>
Absent	Stimulus not occupied by plasmodium	<i>false</i>

In our models of simple rule-based systems, we have implemented an idea of flowing power used in ladder diagrams to model digital circuits. Flowing power is replaced with propagation of plasmodium of *Physarum polycephalum*. The same idea was used by us to construct logic gates through the proper geometrical distribution of stimuli in *Physarum* machines.

Source: Schumann, A., Pancerz, K., Jones, J.: *Towards logic circuits based on Physarum polycephalum machines: The ladder diagram approach*, [in:] Cliquet, Jr., Plantier, A., Schultz, G., Fred, T., Gamboa, A. (eds.): *Proceedings of the International Conference on Biomedical Electronics and Devices (BIODEVICES'2014)*, pp. 165-170, Angers - France 2014.

In general, we can distinguish two techniques to control behaviour of *Physarum polycephalum*:

- Repellent-based, see Fig. 83, 84, 85, 86.
- Attractant-based.

Source: Adamatzky, A.: *Physarum Machines: Computers from Slime Mould*, World Scientific, Singapore 2010:

Figure 83. A Petri net model of a rule of type 1: the repellent-based control approach.

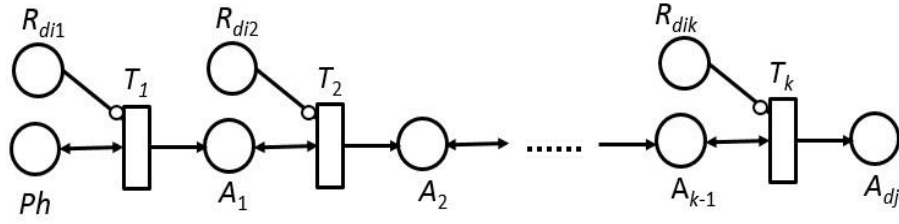
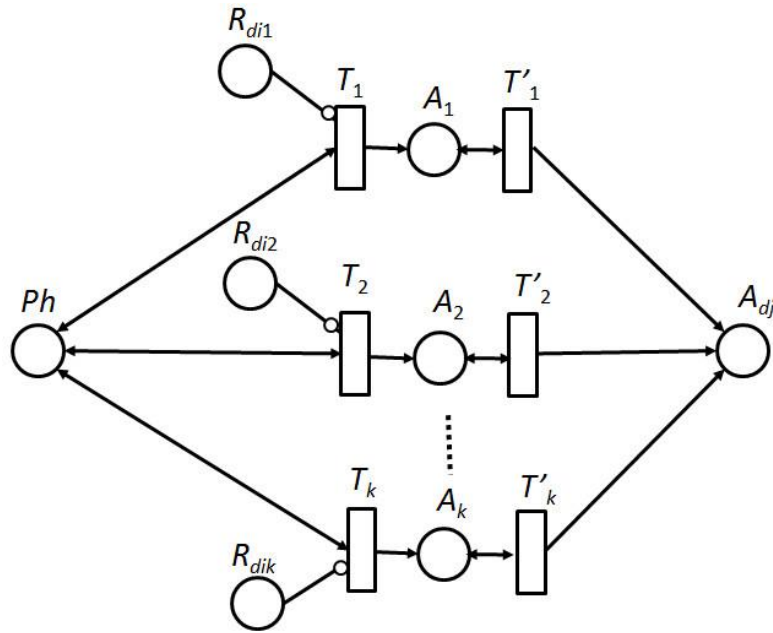


Figure 84. A Petri net model of a rule of type 3: the repellent-based control approach.



So, a high-level model (Petri net model) is translated by us into the low-level language, i.e., spatial distribution of stimuli (attractants and/or repellents). Such distribution can be treated as a program for the *Physarum* machine, see Fig. 85, 86.

Figure 85. The structure of the *Physarum* machine for a production rule of type 1.

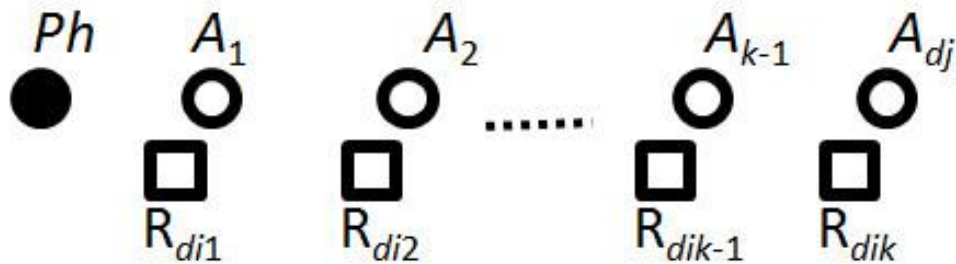
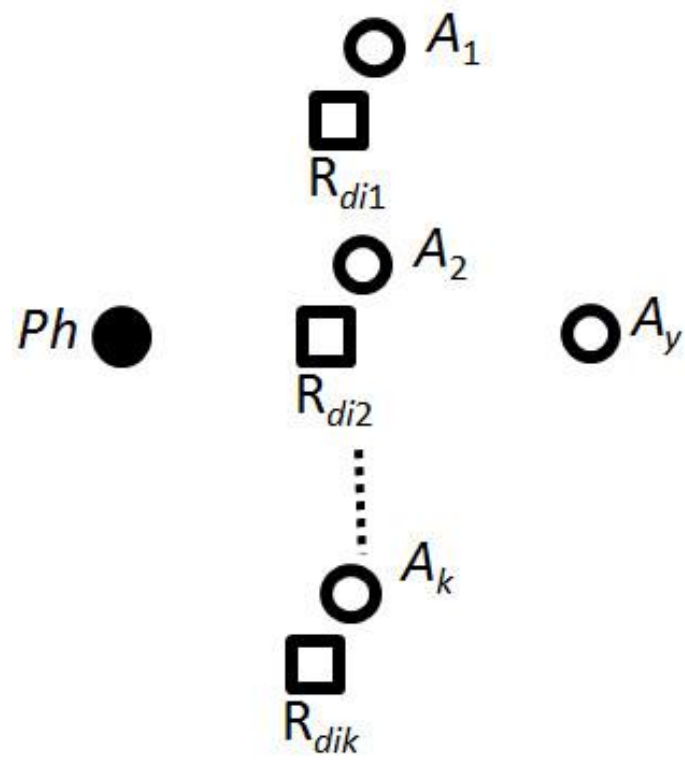


Figure 86. The structure of the *Physarum* machine for a production rule of type 3.



3. Object-oriented language

The proposed language can be used for developing programs for *Physarum* machines by the spatial configuration of stimuli. Spatial distribution of stimuli can be identified with a low-level programming language for *Physarum* machines. The created programming language uses the prototype-based approach called also the class-less or instance-based approach.

There are inbuilt sets of prototypes corresponding to both the high-level models used to describe behaviour of *Physarum polycephalum*, e.g., ladder diagrams, Petri nets, transition systems and the low-level model, i.e., distribution of stimuli.

The plasmodium of *Physarum polycephalum* functions are considered as a parallel amorphous computer with parallel inputs and parallel outputs. We can generally assume that a program of computation is coded via configurations of repellents and attractants.

3.1. Tools and technologies

The grammar of the language is described in the XBNF (eXtended Backus-Naur Form) notation. The parser is created using JavaCC (Java Compiler Compiler), the Java parser generator. The programming and simulation platform is created in the Java environment.

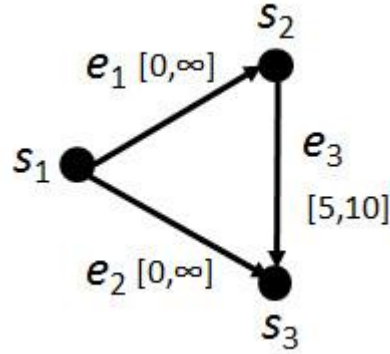
Main objects are defined in the object-oriented (OOP) language for *Physarum polycephalum* computing and their selected methods.

Table 11. OOP language for *Physarum polycephalum* computing.

Model	Object	Selected Methods
Low-Level	<i>Layer</i>	<i>setSize, add</i>
Low-Level	<i>Physarum</i>	<i>setPosition</i>
Low-Level	<i>Attractant</i>	<i>setPosition, setIntensity</i>
Low-Level	<i>Repellent</i>	<i>setPosition, setIntensity</i>
Ladder Diagram	<i>LD.Rung</i>	<i>setExpression</i>
Petri Net	<i>PN.Transition</i>	<i>setDescription</i>
Petri Net	<i>PN.Place</i>	<i>setDescription, setRole</i>
Petri Net	<i>PN.Arc</i>	<i>setAsInhibitor</i>
Transition System	<i>TS.State</i>	<i>setDescription, setAsInitial</i>
Transition System	<i>TS.Event</i>	<i>setDescription</i>
Transition System	<i>TS.Transition</i>	

Let us consider a simple timed transition system shown as a graph structure with the following timing constraints: $l(e_1) = 0$, $u(e_1) = \infty$, $l(e_2) = 0$, $u(e_2) = \infty$, $l(e_3) = 5$, $u(e_3) = 10$, see Fig. 87.

Figure 87. Time Transition System – graph structure.



The code in our created language has the following form:

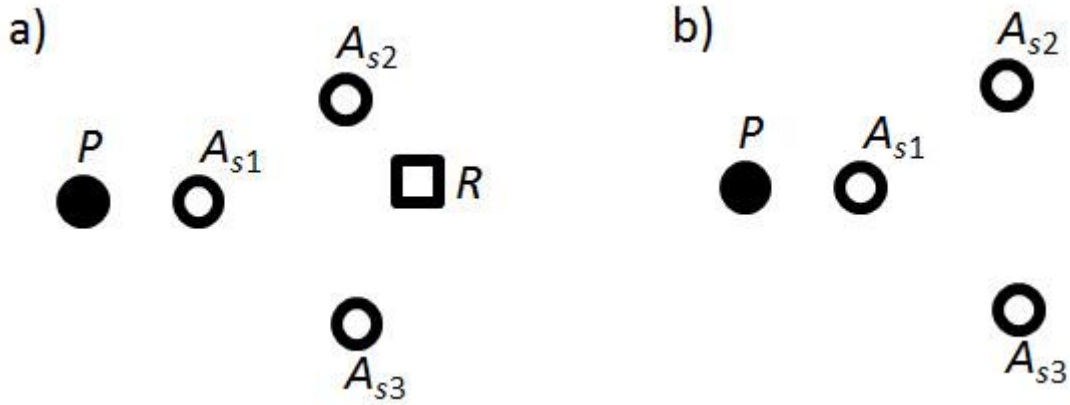
```

#TRANSITION_SYSTEM
s1=new TS.State("s1");
s1.setAsInitial;
s2=new TS.State("s2");
s3=new TS.State("s3");
e1=new TS.Event("e1");
t1=new TS.Transition(s1,e1,s2);
e2=new TS.Event("e2");
t2=new TS.Transition(s1,e2,s3);
e3=new TS.Event("e3");
e3.setTimingConstraints(5,10);
t3=new TS.Transition(s2,e3,s3);

```

As a result of programming the *Physarum* machine, we obtain spatial configurations of stimuli: (a) for the time instant $\tau = 4$, (b) for the time instant $\tau = 8$, where P is *Physarum*, A_{s_1} , A_{s_2} , A_{s_3} are attractants, and R is a repellent, see Fig. 88. The event e_3 is allowed only if actual time $t \in \{5, 6, \dots, 10\}$.

Figure 88. Spatial configuration of stimuli.



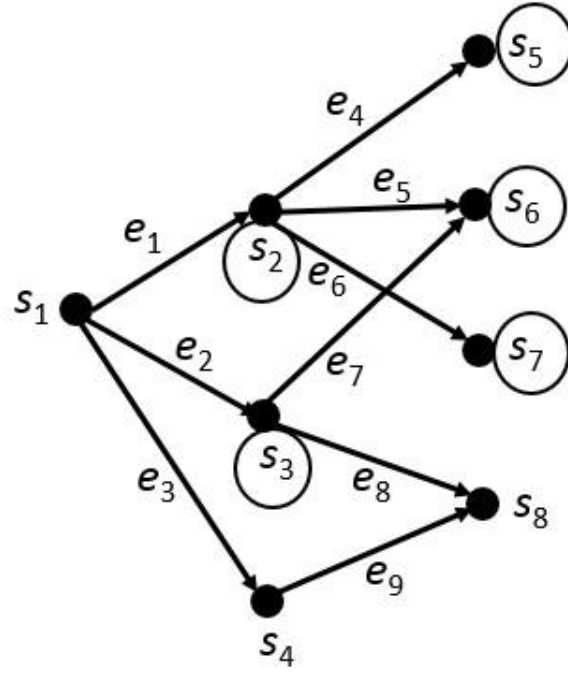
3.2. Rough set models of *Physarum* games

We describe a rough set approach for description of a strategy game created on the *Physarum* machine.

The strategies of such a game are approximated on the basis of a rough set model, describing behaviour of the *Physarum* machine, created according to the VPRSM (Variable Precision Rough Set Model) approach.

As an example, let us consider a transition system describing the game, where player 1 plays for *Physarum polycephalum* and occupies attractants s_2, s_3, s_5, s_6, s_7 and player 2 plays for *Badhamia utricularis* and occupies attractants s_4, s_8 , see Fig. 89.

Figure 89. Transition system presenting a game.



States corresponding to attractants activated by the first player are circled. All other states are assumed to be activated by the second player. Hence, $S^1 = \{s_2, s_3, s_5, s_6, s_7\}$ and $S^2 = \{s_4, s_8\}$. So, $\text{card}(S^2) < \text{card}(S^1)$, but it does not mean that player 1 wins.

If we assume $\beta = 0$ (i.e., the most rigorous case), then, for player 1, $\text{Pre}_*(S^1) = \{s_2\}$ and $\sigma^1 = 1$, and for player 2 $\text{Pre}_*(S^2) = \{s_4\}$ and $\sigma^2 = 1$. Hence, nobody wins, because:

$$\text{Post}(s_1) = \{s_2, s_3, s_4\} \text{ and } \text{Post}(s_1) \dot{\cup} S^1,$$

$$\text{Post}(s_2) = \{s_5, s_6, s_7\} \text{ and } \text{Post}(s_2) \subseteq S^1,$$

$$\text{Post}(s_3) = \{s_6, s_8\} \text{ and } \text{Post}(s_3) \dot{\cup} S^1,$$

$$\text{Post}(s_4) = \{s_8\} \text{ and } \text{Post}(s_4) \dot{\cup} S^1,$$

$$\text{Post}(s_1) = \{s_2, s_3, s_4\} \text{ and } \text{Post}(s_1) \dot{\cup} S^2,$$

$$\text{Post}(s_2) = \{s_5, s_6, s_7\} \text{ and } \text{Post}(s_2) \dot{\cup} S^2,$$

$$\text{Post}(s_3) = \{s_6, s_8\} \text{ and } \text{Post}(s_3) \dot{\cup} S^2,$$

$$Post(s_4) = \{s_8\} \text{ and } Post(s_4) \subseteq S^2.$$

If we assume $\beta = 0.4$ (i.e., more relaxed case), then $Pre_*^{0.4}(S^1) = \{s_1, s_2\}$, $Pre_*^{0.4}(S^2) = \{s_8\}$ and $\sigma^1 = 2$, $\sigma^2 = 1$. This means that player 1 wins, because:

$$Post(s_1) = \{s_2, s_3, s_4\} \text{ and } Post(s_1) \overset{0.4}{\subseteq} S^1,$$

$$Post(s_2) = \{s_5, s_6, s_7\} \text{ and } Post(s_2) \overset{0.4}{\subseteq} S^1,$$

$$Post(s_3) = \{s_6, s_8\} \text{ and } Post(s_3) \overset{0.4}{\dot{\subseteq}} S^1,$$

$$Post(s_4) = \{s_8\} \text{ and } Post(s_4) \overset{0.4}{\dot{\subseteq}} S^1,$$

$$Post(s_1) = \{s_2, s_3, s_4\} \text{ and } Post(s_1) \overset{0.4}{\dot{\subseteq}} S^2,$$

$$Post(s_2) = \{s_5, s_6, s_7\} \text{ and } Post(s_2) \overset{0.4}{\dot{\subseteq}} S^2$$

$$Post(s_3) = \{s_6, s_8\} \text{ and } Post(s_3) \overset{0.4}{\dot{\subseteq}} S^2,$$

$$Post(s_4) = \{s_8\} \text{ and } Post(s_4) \overset{0.4}{\subseteq} S^2.$$

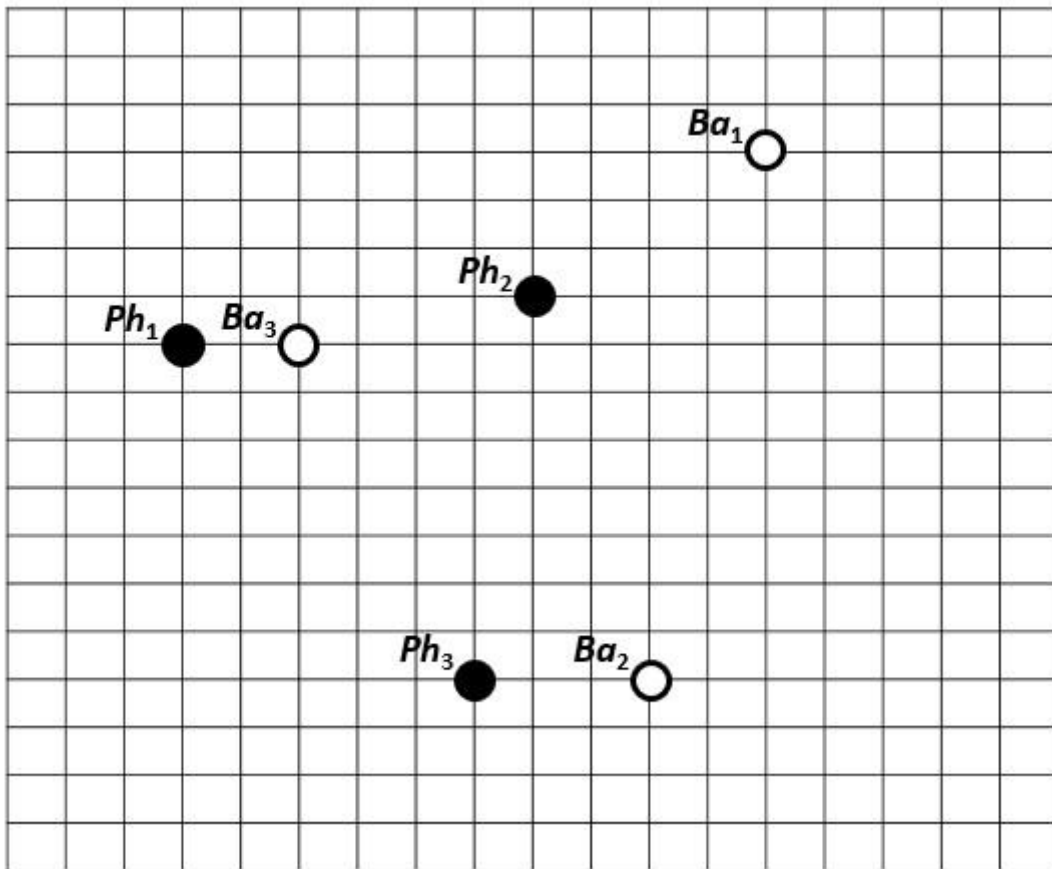
3.3. Go games

Go is a game, originated in ancient China, in which two people play with a Go board and Go stones. In general, the two players alternately place black and white stones, on the vacant intersections of a board with a 19×19 grid of lines, to surrounding territory. Whoever has more territory at the end of the game is the winner.

The plasmodium behaviour can model an ancient Chinese game called Go. We describe implementation of the Go game on the *Physarum* machines. A special version of the game is presented, where payoffs are assessed by means of the measure defined on the basis of rough set theory.

So, let a board for the Go game, with a 19×19 grid of lines, be in use. The set of all intersections of the grid is denoted by I . At the beginning, the fixed numbers of original points of both the plasmodia of *Physarum polycephalum* and the plasmodia of *Badhamia utricularis* are randomly deployed on intersections, see Fig. 90.

Figure 90. Example of stimulated Go game.



During the game, the two players alternately place attractants on the vacant intersections of the board. The first player plays for the *Physarum polycephalum* plasmodia, the second one for the *Badhamia utricularis* plasmodia. The plasmodia look for attractants, propagate protoplasmic veins towards them, feed on them and go on. The attractants occupied by plasmodia of *Physarum polycephalum* are treated as black stones whereas the attractants occupied by plasmodia of *Badhamia utricularis*, as white stones.

Each intersection $i \in I$ is identified by two coordinates x and y . This fact will be denoted by $i(x, y)$. For each intersection $i(x, y)$, we can distinguish its adjacent surroundings:

$$\begin{aligned} \text{Surr}(i) &= \{i'(x', y') \in I : \\ &\quad (x' = x-1 \vee x' = x+1) \wedge (x' \geq 1) \wedge (x' \leq 19) \\ &\quad \wedge \\ &\quad (y' = y-1 \vee y' = y+1) \wedge (y' \geq 1) \wedge (y' \leq 19)\}. \end{aligned}$$

Formally, during the game, at given time instant t , we can distinguish three kinds of intersections in the set I_t of all intersections:

- I_t° – a set of all vacant intersections at t .
- I_t^\bullet – a set of all intersections occupied by plasmodia of *Physarum polycephalum* at t (black stones).
- I_t° – a set of all intersections occupied by plasmodia of *Badhamia utricularis* at t (white stones).

One can see that $I_t = I_t^\circ \cup I_t^\bullet \cup I_t^\circ$, where I_t° , I_t^\bullet , and I_t° are pairwise disjoint.

The lower surroundings approximation $\text{Surr}_*(I_t^\pi)$ of I_t^π is given by configuration of the Go game.

$$\text{Surr}_*(I_t^\pi) = \{i \in I_t^\pi : \text{Surr}(i) \neq \emptyset \wedge \text{Surr}(i) \subseteq I_t^\pi\},$$

where π is either \bullet or \circ . The lower surroundings approximation consists of all intersections, occupied by plasmodia π , whose all of not vacant adjacent intersections are also occupied by π . Each intersection $i \in I$ such that $i \in \text{Surr}_*(I_t^\pi)$ is called a full generator of the payoff of the player playing for the plasmodia π .

The β -lower surroundings approximation. By replacing the standard set inclusion with the majority set inclusion in the definition of the lower surroundings approximation (according to the VPRSM approach), we obtain the β -lower surroundings approximation:

$$\text{Surr}_*^\beta(I_t^\pi) = \{i \in I_t^\pi : \text{Surr}(i) \neq \emptyset \wedge \text{Surr}(i) \overset{\beta}{\subseteq} I_t^\pi\},$$

Each intersection $i \in I$ such that $i \in \text{Surr}_*^\beta(I_t^\pi)$ is called a full quasi-generator of the payoff of the player playing for the plasmodia π .

On the basis of lower surroundings approximations, we define a measure assessing pay-offs of the players:

- (i) For the first player playing for the *Physarum polycephalum* plasmodia, the payoff measure has the form:

$$\Theta^\bullet = \text{card}(\text{Surr}_*(I_t^\bullet)).$$

- (ii) For the second player playing for the *Badhamia utricularis* plasmodia, the payoff measure has the form:

$$\Theta^\circ = \text{card}(\text{Surr}_*(I_t^\circ)).$$

In a more relaxed case, we have respectively:

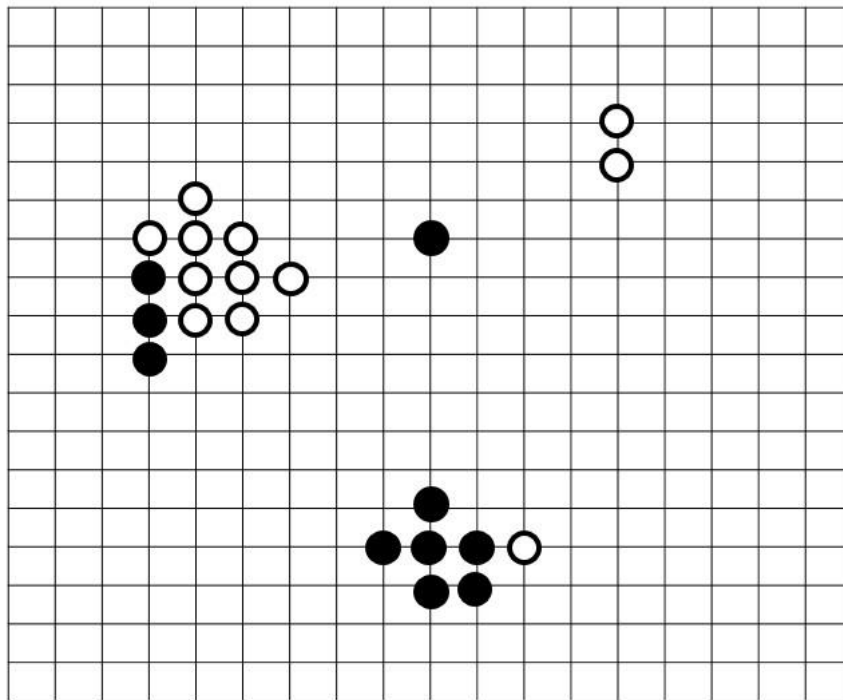
$$\Theta^\bullet = \text{card}(\text{Surr}_*^\beta(I_t^\bullet)).$$

and

$$\Theta^\circ = \text{card}(\text{Surr}_*^\beta(I_t^\circ)).$$

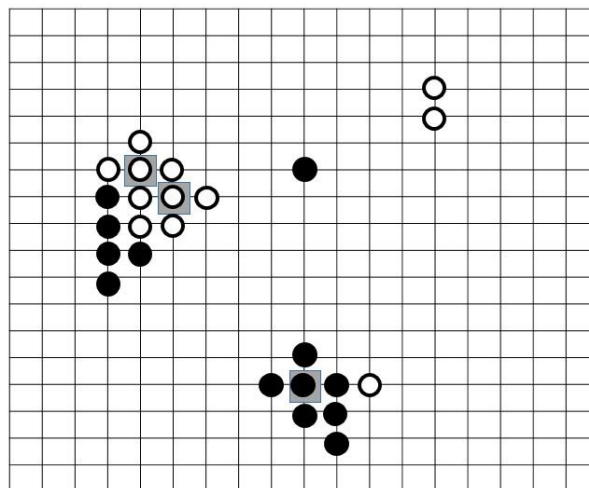
Let us consider an illustrative configuration of the Go game after several moves, see Fig. 91.

Figure 91. Go configuration 1.



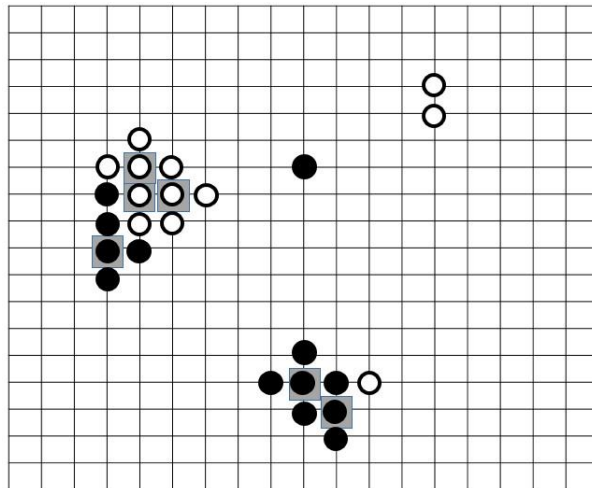
If we appeal to the case of a standard definition of rough sets, then intersections belonging to lower surroundings approximations are marked with grey rectangles, see Fig. 92, and the second player, playing for the *Badhamia utricularis* plasmodia, wins.

Figure 92. Go configuration 2.



If we use the case of the VPRSM approach (i.e., a more relaxed case), for $\beta = 0.25$, then intersections belonging to lower surroundings approximations are marked with grey rectangles, see Fig. 93, and no player wins.

Figure 93. Go configuration 3.



3.4. *PhysarumSoft*

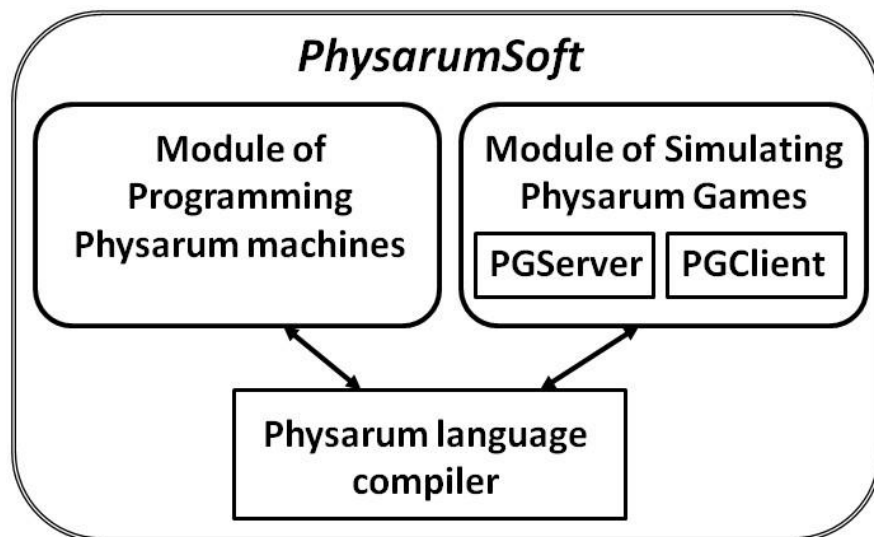
The current version of a new software tool, called *PhysarumSoft*, has been developed for:

- Programming *Physarum* machines.
- Simulating *Physarum* games.

This tool was designed for the Java platform.

A general structure of *PhysarumSoft* is pictured in Fig. 94.

Figure 94. *PhysarumSoft*.



We can distinguish three main parts of *PhysarumSoft*:

- *Physarum* language compiler.
- Module of programming *Physarum* machines.
- Module of simulating *Physarum* games.

The *Physarum* language is an object-oriented high-level programming language. For generating the compiler of the language, the Java Compiler Compiler (JavaCC) tool was used. A compiler translates the high-level code describing the model of *Physarum* machine into the spatial distribution (configuration) of stimuli (attractants, repellents) controlling propagation of protoplasmic veins of the plasmodium. A grammar of our language was described in:

- Pancerz, K., Schumann, A.: *Some issues on an object-oriented programming language for Physarum machine*, [in:] *Applications of Computational Intelligence in Biomedical Technology, Studies in Computational Intelligence*, vol. 606, pp. 185-199, Springer International Publishing.

The main features of *PhysarumSoft* are as follows:

- Portability. The created tool can be run on various software and hardware platforms.
- User-friendly interface.
- Modularity. The project of *PhysarumSoft* and its implementation covers modularity. It makes the tool extend in the future easily.

We have proposed several high-level models used in programming *Physarum* machines:

- Schumann, K., Pancerz, J.: *Towards logic circuits based on Physarum polycephalum machines: The ladder diagram approach*. Proc. of BIODEVICES'2014, pp. 165-170.
- Pancerz, K., Schumann, A.: *Principles of an object-oriented programming language for Physarum polycephalum computing*. Proc. of DT' 2014.
- Schumann, A., Pancerz, K.: *Timed transition system models for programming Physarum machines: Extended abstract*. Proc. of CS&P' 2014.
- Schumann, A., Pancerz, K.: *Towards an object-oriented programming language for Physarum polycephalum computing: A Petri net model approach*. Fundamenta Informatiae 133(2-3), pp. 271-285, 2014.

Let us consider an exemplary timed transition system *TTS* pictured in Fig. 95. Its code is represented in Fig. 96 and the result of compilation is represented in Fig. 97.

Figure 95. An exemplary timed transition system *TTS*.

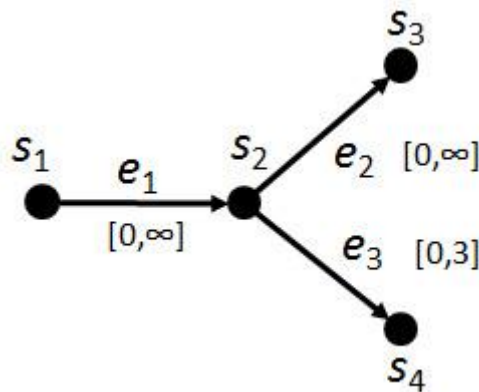
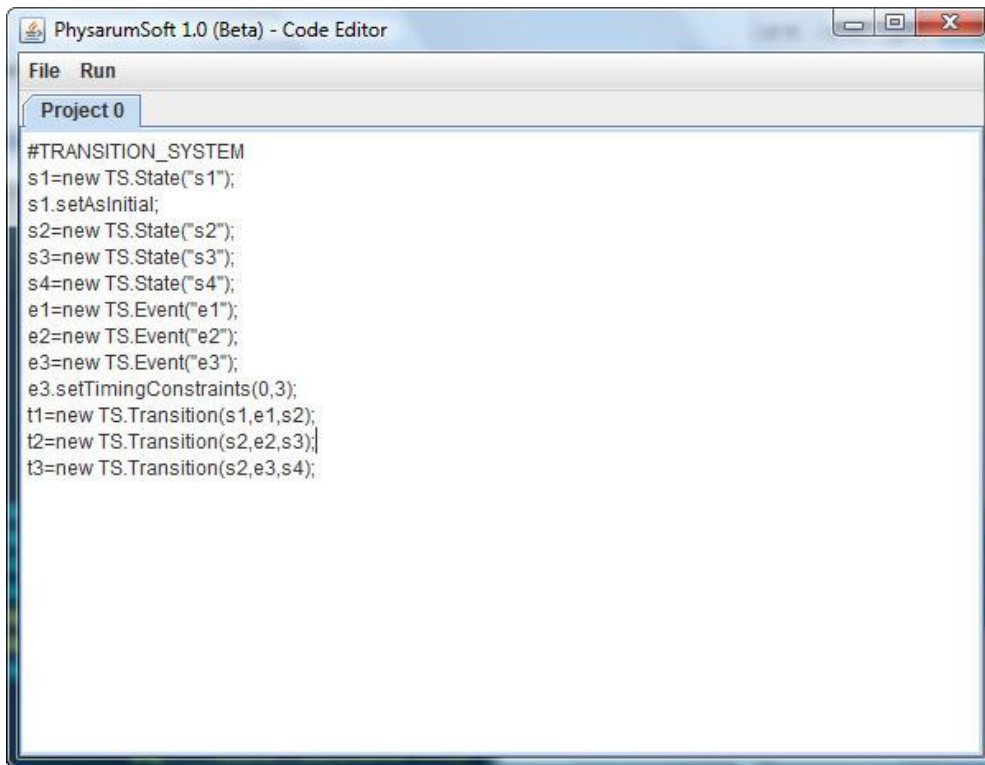
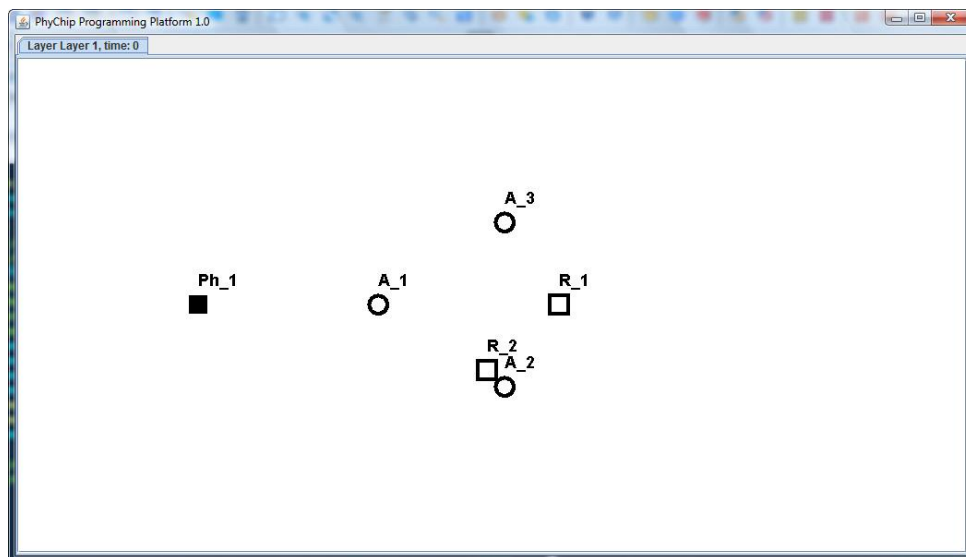


Figure 96. The code for the model in the form of *TTS*.



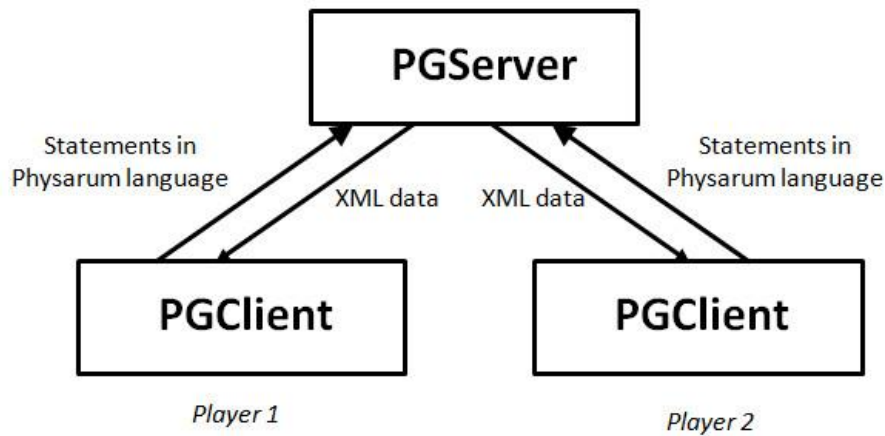
```
#TRANSITION_SYSTEM
s1=new TS.State("s1");
s1.setAsInitial;
s2=new TS.State("s2");
s3=new TS.State("s3");
s4=new TS.State("s4");
e1=new TS.Event("e1");
e2=new TS.Event("e2");
e3=new TS.Event("e3");
e3.setTimingConstraints(0,3);
t1=new TS.Transition(s1,e1,s2);
t2=new TS.Transition(s2,e2,s3);
t3=new TS.Transition(s2,e3,s4);
```

Figure 97. The results of compilation of *TTS*.



The *Physarum* game simulator works under the client-server paradigm, see Fig. 98.

Figure 98. The client-server paradigm.



In the *Physarum* game simulator, we have two players:

- the first plays for the *Physarum polycephalum* plasmodia,
- the second plays for the *Badhamia utricularis* plasmodia.

The main window of PGServer is pictured in Fig. 99.

Figure 99. Main window of PGServer.

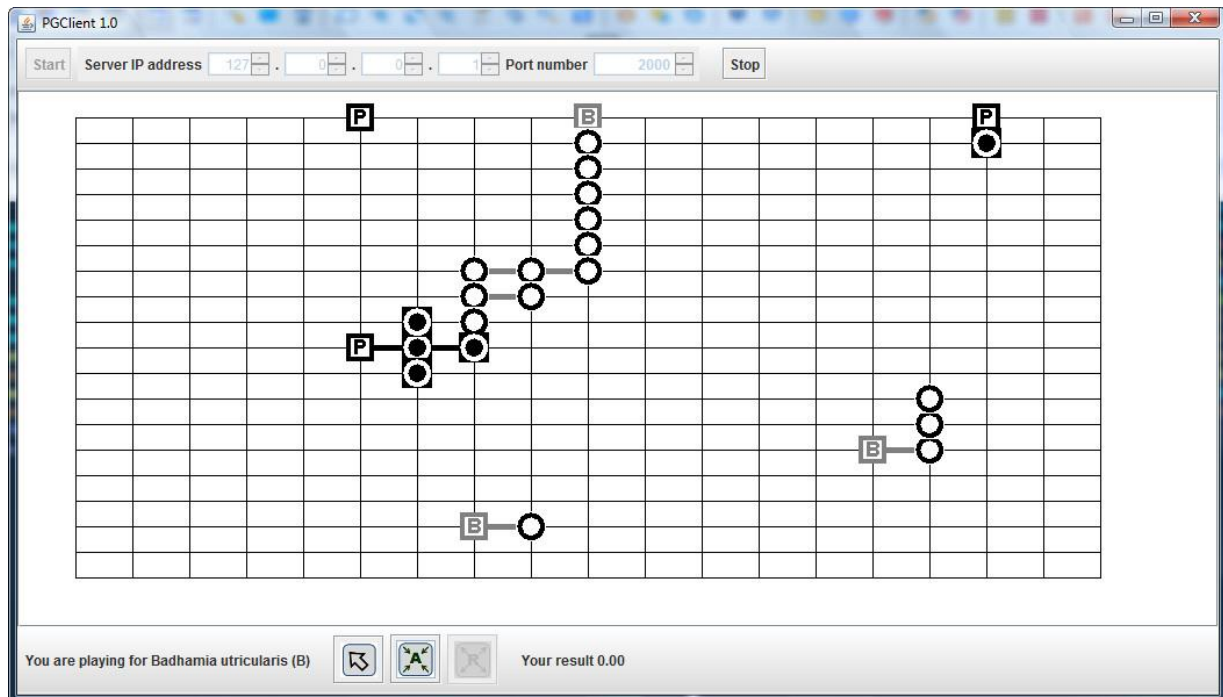


The user can:

- select the port number on which the server listens for connections,
- start and stop the server,
- set the game:
 - a *Physarum* game with strategy based on stimulus placement,
 - a *Physarum* game with strategy based on stimulus activation,
 - a rough set version of the Go game,
- shadow information about actions undertaken.

The main window of PGClient is pictured in Fig. 100.

Figure 100. The main window of PGClient.



The user can:

- set the server IP address and its port number,
- start the participation in the game,
- put attractants at the vacant intersections of a board,
- monitor the current state of the game as well the current assessment of payoffs.

Conclusions and future work

Four classes of devices can be implemented with *Physarum*: morphological processors, electronic circuits, micro-fluidic circuits, intra-cellular circuits, and artificial actin filament networks.

Morphological processors represent results of computation with configurations of protoplasmic networks. Data are represented as a spatial configuration of attractants and repellents. Halting of computation is detected by stable state aided by compressibility changes. The results of computation are represented by morphology of the slime mould. The morphological processors can implement p-adic arithmetics if the slime mould can see not more than $p - 1$ attractants at each step of its propagation at all places of its directions. In case the slime mould can be well controlled in its motions (by using repellents or by distributing attractants on a nutrient-poor substrate), functions on finite p-adic integers can be implemented, i.e. we deal with standard arithmetics. In case the slime mould moves massive-parallelly (e.g. on a nutrient-rich substrate or many attractants are localized near each other) functions on infinite p-adic integers can be implemented, i.e. we deal with non-Archimedean metrics. The morphological processors solve a substantial range of problems from graph optimisation (e.g. shortest path, spanning trees, transport networks, space exploration, risk averse behaviour), combinatorial optimisation (Travelling Salesman Problem), and computational geometry (e.g. tessellations, triangulations, hulls) and collision-based computing (e.g. ballistic logical gates). The morphological processors are parallel: the data space is explored by numerous growth cones simultaneously. These processors can implement concurrent and context-based games. So, they can be used as universal behavioural models for simulating group or swarm behaviour. The time complexity of the computation is determined by a diameter of the physical data set. Thus computation of one problem can proceed for days if a size of the data set is tens of centimetres. The processors can be scaled down only to a one element of data per square centimetre. The morphological processors are not reusable but can be sequentially arranged for more complex applications, e.g. using inexpensive DIY slime ‘cartridges’. Thus, it could be considered as a minor cost, like computer media, such as a floppy disk or USB pen drive.

Micro-fluidic circuits employ *Physarum* response to tactile stimulation. When a fragment of a protoplasmic tube is touched – flow of cytoplasm through this fragment halts temporarily. The cytoplasm is rerouted via bypassing fragments. We have constructed few logical gates and memory devices in micro-fluidic circuits. The tube’s responds to mechanical stimulation in seconds; the response lasts for up to a minute. This determined the maximum frequency of *Physarum* fluidic circuits: 0.02 Hz. A simple gate fits in 2-3 sq mm domain: this is maximum resolution achieved. The micro-fluidic devices are faster than morphological processors and the gates are reusable. Technological disadvantages are that the slime mould must be kept alive yet prevented from uncontrollable sprouting of its protoplasmic tubes and speed of the computing is rather inappropriate.

Intra-cellular circuits employ vesicles as carriers of information. Values of logical variables are encoded in presence/absence of the vesicles. A computation is implemented via collision of the vesicles. Trajectories of the vesicles after collision represent results of the computation. A density of the vesicle based computing circuits is limited by an average diameter of an intra-cellular vesicle: 50 nm. Technological disadvantage of the intra-cellular vesicle based circuits is lack of programmability and strong dependence on physiological state of the slime mould implementing the computation.

There are two subclasses of **electronic circuits** made from *Physarum*: raw and hybrid. Raw circuits use living *Physarum* interfaced with conventional hardware. The implement computation by encoding chemical, tactile, optical or electrical stimuli to frequency of oscillations of extracellular potential (frequency-based logical gates, tactile sensors, chemical sensors, colour sensors) or by modulating input voltage and current (voltage divider, low-pass filters, oscillators, memristors, opto-electronic gates). Hybrid electronic circuits employ *Physarum* coated or loaded with functional materials. The transform input voltage/current to output voltage/current. Examples are memristors, Schottky diodes, transistors. Minimal size of electronic circuits is limited by diameter of protoplasmic tube: minimum 0.1 mm. Frequency of oscillation based circuits is 0.02 Hz. The electrical analog circuits show a response time typical for conventional electronic devices.

Artificial actin filament networks employ transmitting information by means of building and rebuilding actin filament networks in responses to dynamics of intra-cellular and extra-cellular stimuli. Some new processors (filaments) can appear in one conditions and they can disappear in other conditions. This system is much more complex, than artificial neural networks, where we have a fixed number of processors (neurons).

PhyChip software development and interfacing

A new hardware substrate must be supported by nascent methods of programming these devices. To this end we have successfully demonstrated methods of converting a large number of classical algorithms for spatial implementation by PhyChip and its models. The methods exploit parallel morphological adaptation that is embedded within the physical substrate. An important, and often overlooked factor is how to successfully interface classical computing devices with the novel substrate. Our collected results have achieved this interfacing by means of electrical, chemical, tactile, thermal, magnetic and optical stimuli respectively. Data can be presented to present the *Physarum* chip problem configuration, to instantiate computation and to dynamically control its evolution using closed-loop mechanisms, to analyse the current state of computation in real-time, and to decide when to halt the computation. To some extent it remains an open problem as to exploring the range of computation which can and cannot be performed by (or rather perhaps, are ill suited for) *Physarum* chip. In such cases, the methods of interfacing we have explored will enable us to tackle these problems head on.

Symbolic and logical models of *Physarum* computing

We have explored symbolic and logical models of *Physarum* computing to provide spatial propagative computing with new methods. First of all, we have implemented some timed, probabilistic, syllogistic, modal, game-theoretic, rough-set-theoretic extensions of standard process algebra to develop simulation models of the slime mould behaviour. These models allow us to describe also some ambiguities in *Physarum* propagation that influence exact anticipation of states of *Physarum* machines. Then, we have applied some tools of non-well-founded mathematics such as coalgebra and p-adic mathematics to involve a massive parallelism in the plasmodium propagation into computational processes. In this direction, first, we have proposed a theory of hybrid actions (an extension of labelled transitions system to a system with a non-well-founded set of labels) and, second, we have constructed a bio-inspired game theory. The methods proposed by us may enable us to tackle problems whose time and space demands are ill-suited to spatial computing approaches. Moreover, to model computational tasks for *Physarum* machines and to check game-theoretic strategies of slime mould in its occupation of food, we have developed a new object-oriented programming language, called a *Physarum* language. The *Physarum* language is a prototype-based language consisting of inbuilt sets of prototypes corresponding to the high-level models used for describing behaviour of *Physarum* (e.g., ladder diagrams, probabilistic and timed transition systems, Petri nets, concurrent games, context-based games).

In our project we have developed a range of prototypes for bio-inspired computing but we recognized several challenges before these prototypes can be developed into technical applications to solve practical computing problems, which center around questions such as scalability, reusability, reproductivity. We therefore consider this a theme for future application for unconventional computing projects (using biological material as substrates) in the framework of the FET programme. On the other side, we envision that biological information processing as displayed by slime moulds will inspire new forms of distributed algorithms to solve parameter-rich optimization problems. These new algorithms may then conveniently be implemented with classical electronic devices. We thus suggest that future calls in the FET programme might on one hand focus on the discovery of nature-based information processing paradigms as exemplified by diverse biological organisms, and novel ways – conventional and unconventional – to implement these paradigms as technical solutions.

Andrew Adamatzky,
Victor Erokhin,
Martin Grube,
Theresa Schubert,
Andrew Schumann



Andrew Schumann works at the University of Information Technology and Management in Rzeszow, Poland. His research focuses on logic and philosophy of science with an emphasis on non-well-founded phenomena: self-references and circularity. He contributed mainly to research areas such as reasoning under uncertainty, probability reasoning, non-Archimedean mathematics, as well as their applications to cognitive science. He is engaged also in unconventional computing, decision theory, logical modeling of economics.



Krzysztof Pancerz received his M.Sc. in 1998 from the Rzeszow University of Technology in Electrical Engineering and Ph.D. in 2006 from the Institute of Computer Science, the Polish Academy of Sciences in Computer Science. His research interests concern computational intelligence, knowledge engineering, unconventional computing, and computer-aided diagnosis. He has published over 70 research papers in international journals, monographs and conference proceedings.

Physarum Chip: Growing Computers from Slime Mould. Logical Aspects

In the summer we can find out a bright yellow slimy blob on flower beds. It is a slime mould called *Physarum polycephalum*. It is a huge single cell that contains millions of nuclei which are divided at the same time. This large cell is expanded in many directions simultaneously, but very slowly, about 1mm per hour. *Physarum polycephalum* is not animal, plant, fungus, or bacterium. Meanwhile, it is so intelligent. It can solve complex computational problems, e.g. it can optimise transport routes. In this book we propose logics to design selfgrowing computers on the medium of *Physarum polycephalum*. These computers represent a kind of distributed intelligence.