Computer Science Journal of Moldova, vol.**18**, no.2(53), 2010

Investigations on Natural Computing in the Institute of Mathematics and Computer Science

Artiom Alhazov Elena Boian Liudmila Burtseva
Constantin Ciubotaru Svetlana Cojocaru
Alexandru Colesnicov Valentina Demidova
Sergiu Ivanov Veaceslav Macari Galina Magariu
Ludmila Malahova Vladimir Rogojin Yurii Rogozhin
Tatiana Tofan Sergey Verlan Tatiana Verlan

Abstract

We describe the investigations on natural computing in the Institute of Mathematics and Computer Science of the Academy of Sciences of Moldova during last fifteen years. Most of these investigations are inspired by results and ideas belonging to Corresponding Member of the Romanian Academy **Gheorghe Păun.**

1 Introduction

In this paper we present a short overview of investigations on natural computing carried out in the Institute of Mathematics and Computer Science of Academy of Sciences of Moldova during last fifteen years.

Exactly fifteen yeas ago one of the authors of this paper, Prof. Yurii Rogozhin has started his study in the scope of natural computing. His studies and studies of his colleagues in the Institute were inspired by scientific activity of Corresponding Member of the Romanian Academy Gheorghe Păun.

This overview includes investigations at the Institute on DNA computing, membrane computing, insertion-deletion systems and other models of biocomputing. The most of results obtained by scientists of the Institute of Mathematics and Computer Science were presented at different international workshops and conferences on natural computing and published in prestigious international journals. Three PhD theses and one Habilitation thesis were defended since 2004 and their results are reflected in this volume of the journal.

2 DNA Computing

2.1 Test Tubes Systems

Molecular computers have been attracting many people from chemistry, biology and computer science. A major break through was a concrete molecular computer by Adleman [1] that could solve instances of the travelling-salesman-problem.

In a remarkable paper T. Head [59] draw the connections between molecular computers and formal language theory. The molecules from biology are replaced by words over a finite alphabet and the chemical reactions are replaced by the *splicing* operation. An H system specifies a set of rules used to perform splicing and a set of initial words or axioms. A splicing rule may be applicable to two molecules. It breaks both molecules at fixed locations, defined by the splicing rule, and recombines the initial string of one broken molecule with the final string of the other one. The computation is done by applying iteratively the rules to the set of words until no more new words can be generated. This corresponds to a bio-chemical experiment where we have enzymes (splicing rules) and initial molecules (axioms) which are put together in a tube and we wait until the reaction stops.

T. Head's smooth connection to formal language theory brought this field to the attention of many people from formal language theory. E.g., Gh. Păun, G. Rozenberg and A. Salomaa [89] asked what classes of formal languages are derivable by molecular computers depending on certain classes in formal language theory that the initial molecules

and enzymes belong to.

One of such results says that any regular language is derivable from finitely many initial molecules with finitely many splicing rules. E. Csuhaj-Varjù, L. Kari, Gh. Păun [47] modified T. Head's concept slightly to systems of n test tubes. Here, any test tube is an H-system with an additional filter. In a single macro-step any test tube generates new molecules according to its set of starting molecules and its set of splicing rules. Afterwards, the outcome of all test tubes is poured into the filters of all test tubes. Those molecules that may pass the filter of test tubes $i, 1 \leq i \leq n$, form the new starting molecules for the i-th test tube for the next macro-step.

This new process, filtering results of one test tube into another, increases the computational capability of molecular computers. Let us call a system of n test tubes *finite* if initially any test tube contains (arbitrarily many) copies of molecules from a finite set of molecules and possesses only finitely many splicing rules.

It is known [89, 1] that a finite 1-test-tube-system generates only regular sets of molecules. However, finite 2-test-tube-system may generate more complicated non-regular sets [47]. C. Ferretti, G. Mauri, C. Zandron [54] have shown that any recursively enumerable (r.e.) set of molecules is derivable in a finite 9-test-tube-system (or in a finite 6-test-tube-system if one allows for a rather simple encoding of the molecules to be generated).

These results have implications for molecular computers as r.e. languages have many undecidable properties. E.g., the membership problem is in general not decidable for r.e. languages. This means that there exists no algorithm $\mathcal{A}$ which can tell, when presenting a word w and an r.e. languages L to $\mathcal{A}$, whether w belongs to L or not. Further, there is a fixed language, U, such that there exists no algorithm $\mathcal{A}$ which can tell, when presenting a word w to $\mathcal{A}$, whether w is an element of U or not.

Thus, a trivial consequence of the result of [54] is that there exists no algorithm which can compute which molecules may be generated in a finite 6-test-tube-system. I.e., the results of a finite 6-test-tube-system cannot be algorithmically predicted in general. We improved

this result by showing how to generate any r.e. language in a finite 3-test-tube-system [96]. Thus, there is no way to predict the outcome of the reactions of only three test tubes starting with molecules and enzymes from finite set of molecules and enzymes.

This question is still open for finite 2-test-tube-systems.

2.2 TVDH Systems

Head splicing systems (H systems) were one of the first theoretical models of biomolecular computing and they were introduced by T. Head [59].

Ordinary H systems are not very powerful and a lot of other models introducing additional control elements were proposed. One of these well-known models are time-varying distributed H systems (TVDH systems) introduced by Gh. Păun in [90] as another theoretical model of biomolecular computing, based on splicing operations.

He started from the biological observation that at each moment there is a set of active enzymes which behave depending on conditions of the environment. If the environment (temperature, acidity or other parameter) changes, then the set of active enzymes also change. In the proposed model, the set of splicing rules changes periodically. More exactly, the model contains a set of words, the axioms, and a finite number of sets of splicing rules, the components. At each step, the current words are spliced once using the rules of the current component, and the result of this splicing forms the set of words for the next iteration.

We remark that this elimination procedure is very powerful and it permits to obtain a big computational power. If the elimination procedure is changed by permitting several splicings to be applied, then another model similar to TVDH systems is obtained: Enhanced Time-Varying distributed H systems, see [73, 75, 103, 104].

It is worthy to note that TVDH systems with one component generate all RE languages. This result highlights the importance of the elimination and shows that for ordinary H systems only small modifications are needed in order to pass from regularity to rationality, see

also [105].

A study of TVDH systems from a computational point of view may be found in [102] where TVDH systems for main arithmetic operations (addition, multiplication, exponentiation and division) and for the Ackermann function were explicitly given. A computer simulator of TVDH systems was also developed [101].

Moreover, the simple structure of TVDH systems permit to use these systems as target for universality proofs for systems based on splicing. For example, in [79] a simulation of the TVDH system from [78] is done. In [106] there is an example of simulation of TVDH systems with one component by splicing membrane systems (P systems) [100].

TVDH systems have a very simple structure and a powerful control. These features stimulated several articles investigating the computational power of these systems. In [92, 91] Gh. Păun showed that TVDH systems are computationally complete by constructing a system having 7 components that simulate any type-0 grammar.

Subsequent articles decreased consequently the number of components needed to obtain the computational completeness. This was done in two ways. In the first case TVDH systems simulating Turing machines and tag systems were constructed. We remark that in this case the proof is strictly sequential: only one molecule which encodes the tape (or the working word) and the state of the machine shall be present in the system. In 1998 M. Margenstern and Yu. Rogozhin showed first that TVDH systems with 2 components are able to do universal computations, see [70, 71]. Their proof was based on a simulation of tag systems [45, 82] and the obtained system is quite complicated. Later, a proof based on a simulation of Turing machines was proposed [71].

Shortly after that the same authors showed that with 2 components it is possible to generate all recursively enumerable languages [72, 74]. Such generation was done in the following way. It is known that for any RE language $\mathcal{L} = \{w_1, w_2, \dots\}$ there is a Turing machine that, given an input 01^n, $n > 0$, will compute $01^{n+1}w_n$. The system that they constructed behaves in the following way. First, a simulation of corresponding Turing machine on the input 01^n is done. After that

special rules cut off word w_n and the system restarts the simulation of the Turing machine on the new input 01^{n+1}. In this way any RE language is generated word by word.

The same authors obtained in 2001 a very important result: TVDH systems with one component are universal [77]. The core of the proof consists in a simulation of tag systems. In the same year they proposed a system having one component that generates all RE languages [76]. The proof is based on the same ideas that were presented in previous paragraph.

Another way to show the computational completeness of TVDH systems consists in simulation of type-0 grammars. This introduces a parallelism in computations because several evolutions of molecules are made in the same time. Also this case is more complicated than the sequential one and needs much more accuracy. Almost all results in this case are based on "rotate-and-simulate" method.

The article of Gh. Păun [92] is an example of this technique. In 1999 A. Păun showed that it is possible to simulate the work of an arbitrary grammar with 4 components [88]. This result was improved by M. Margenstern and Yu. Rogozhin who showed that a type-0 grammar can be simulated with 3 components, see [73].

We improved the above results by showing that it is possible to generate any recursively enumerable language in a parallel way with 2 components, see [78]. Finally, the final point was reached by showing that it is possible to generate all recursively enumerable languages in a parallel way with one component, see [79]. The last result was obtained by using the method of directing molecules, see [104], which is a modification of the "rotate-and-simulate" method for systems based on splicing.

3 Membrane (P) Systems

The research area of membrane computing originated as an attempt to formulate a model of computation motivated by the structure and functioning of a living cell - more specifically, by the role of membranes in compartmentalization of living cells into protected reactors.

Therefore, initial models were based on a cell-like (hence hierarchical) arrangement of membranes delimiting compartments, where multisets of chemicals (called objects) evolve according to given evolution rules. These rules were either modeling chemical reactions and had the form of (multiset) rewriting rules, or they were inspired by other biological processes, such as passing objects through membranes (either in symport or antiport fashion), and had the form of communication rules. These initial models were then modified by incorporating various additional features motivated by considerations rooted in biology, mathematics, or computer science.

The next important step in the development of research in membrane computing was to also consider other (nonhierarchical) arrangements of membranes. While hierarchical (cell-like) arrangements of membranes correspond to trees, tissue-like membrane systems consider arbitrary graphs as underlying structures, with membranes placed in the nodes while edges correspond to communication channels (see [94, 87]).

3.1 Transitional P Systems

We introduced a new approach to study the family of languages generated by the transitional membrane systems without cooperation and without additional ingredients ([28, 29, 30, 31]). The fundamental nature of these basic systems makes it possible to also define the corresponding family of languages in terms of derivation trees of context-free grammars. We also compare this family to the well-known language families and discuss its properties.

We considered some theoretical tasks for P systems. In particular, we considered a new variant of the halting condition in P systems ([20, 55]), i.e., a computation in a P system is already called halting if not for all membranes a rule is applicable anymore at the same time, whereas usually a computation is called halting if no rule is applicable anymore in the whole system. This new variant of partial halting is especially investigated for several variants of P systems using membrane rules with permitting contexts and working in different transition modes,

especially for minimal parallelism.

Both partial halting and minimal parallelism are based on an arbitrary set of subsets from the set of rules assigned to the membranes.

We considered the problem of synchronizing the activity of all membranes of a P system ([9, 23]). After pointing at the connection with a similar problem dealt with in the field of cellular automata where the problem is called the firing squad synchronization problem, FSSP for short, we provided two algorithms to solve this problem. One algorithm is non-deterministic and works in $2h + 3$ steps, the other one is deterministic and works in $3h + 3$ steps, where h is the height of the tree describing the membrane structure. We introduced a new derivation mode for P systems that permits to make a look-ahead on the next configuration and check for some forbidding conditions on it [108]. The interesting point is that the software implementation of this mode needs very small modifications to the standard algorithm of rule assignment for maximally parallelism. As benefits of this mode some non-deterministic proofs become deterministic.

As an example we present a generalized communicating P system that accepts numbers 2^n in n steps in a deterministic way. Another example shows that in the deterministic case this mode is more powerful than the maximally parallel derivation mode. Finally, this mode gives a natural way to define P systems that may accept or reject a computation.

3.2 Communication P Systems

Communication P systems [52, 56, 98, 110] are inspired by the idea of communicating substances through membrane channels of a cell. Molecules may go in the same direction together – *symport* – or some of them may leave while at the same time other molecules enter the cell – *antiport*. Communicating objects between membrane regions is a powerful tool yielding computational completeness with one membrane using antiport rules or symport rules of size three, i.e. involving three objects, in the maximally parallel mode. As register machines can be simulated in a deterministic manner, P systems with antiport rules or

symport rules can accept any recursively enumerable set of (vectors of) natural numbers in a deterministic way.

In tissue P systems, the objects are communicated through channels between cells. In each transition step we apply only one rule for each channel, whereas at the level of the whole system we work in the maximally parallel way. Computational completeness can be obtained with a rather small number of objects and membranes or cells, in the case of tissue P systems even with copies of only one object.

The computational power of P systems with antiport rules or symport rules involving copies of only one object remains one of the most challenging open questions.

The concept of P systems with antiport and/or symport rules can be generalized to systems using membrane rules evolving multisets of objects on both sides of the membrane even depending on permitting contexts (also called promoters) and/or forbidden contexts (also called inhibitors).

P systems with communication rules can also be used as language generators – we take the sequences of terminal objects sent out to the environment as the strings generated by the system. A generalized communicating P system, or a GCPS for short, corresponds to a graph where each node, called a cell, contains a multiset of objects which – by communication – may move between the cells.

The communication rules are rather restricted, any rule identifies four cells, two input cells and two output cells, such that a pair of objects from the two input cells moves synchronously to the two output cells. The form of a communication rule is $(a;i)(b;j) \rightarrow (a;k)(b;l)$ where a and b are objects and i,j,k,l are numbers that identify the input and the output cells. Such a rule means that an object a from cell i and an object b from cell j move synchronously to cell k and cell l, respectively. It can easily be seen that these very simple communication rules can also be interpreted as interaction rules.

Depending on the relation of i,j,k,l, nine restricted variants of communication rules (modulo symmetry) can be distinguished. (For example, $i \neq j \neq k \neq l$ is one of these restrictions, called a parallel-shift rule). When the GCPS has only one type of these restricted

rules, we speak of generalized communicating P systems with minimal interaction, a GCPSMI for short.

We considered generalized communicating P systems which use only one type of the above interaction operations [51]. We proved that in 7 of these cases computational completeness is obtained, i.e., the corresponding GMPCSs are able to determine any recursively enumerable set of non-negative integers; the only exception determines only finite singletons of natural numbers.

The constructions in the proofs also demonstrate that this large expressive power can be obtained by P systems with relatively small numbers of cells and simple graph architectures. We also proved [52] that GCPSs still remain computationally complete if they are given with a singleton alphabet of objects and with one of the restricted types of rules: parallel-shift, join, presence-move, and chain.

3.3 Polymorphic P systems

We introduced a variant of the multiset rewriting model of P systems where the rules of every region are defined by the contents of interior regions, rather than being explicitly specified in the description of the system [33]. This idea is inspired by the von Neumann's concept of "program is data" and also related to the research direction proposed by Gh. Păun about the cell nucleus.

Membrane computing is a fast growing research field opened by Gh. Păun in 1998. It presents a formal framework inspired from the structure and functioning of the living cells.

In this paper we define yet another, relatively powerful, extension to the model, which allows the system to dynamically change the set of rules, not limited to some finite prescribed set of candidates. There are three motives for this extension:

- first, our experience shows that "practical" problems need "more" computing potential than just computational completeness;

- second, we attempt to import a very important computational

ingredient into P systems, this time from the conventional computer science;

- third, this extension correlates with the biological idea that different actions are carried out by different objects, which can be acted upon as well.

Let us first explain these motives. Most papers of the field belong to the following categories:

1. introducing different models and variants,
2. studying the computational power of different models depending on what ingredients are allowed and on the descriptional complexity parameters,
3. studying the computational efficiency of solving intractable problems (supercomputing potential) depending on the ingredients,
4. using membrane computing to represent and model various processes and phenomena, including but not limited to biology,
5. other applications.

There is a surprisingly big gap between the sets of ingredients needed to fulfill requirements in directions 2, 3, and the sets of ingredients demanded by other applications. For instance, very weak forms of cooperation between objects are often enough for the computational completeness, but many "practical" problems cannot be solved in a satisfactory way under the same limitations. This leads to the following question. What is implicitly required in most "practical" problems? We will mention just a few of these requirements below.

A) Determinism or at least confluence. Clearly, the end user wants to obtain the answer to the specified problem in a single run of a system instead of examining infinitely many computations. This is a strong constraint, e.g., catalytic P systems and P systems with minimal symport/antiport are universal, while in the deterministic case non-universality is published for the first ones and

claimed for the latter ones. Informally speaking, less computational power is needed to just compute the result than it is to also enforce choice-free behavior of the system.

B) Input/output. Most of the universality results are formulated as generating languages or accepting sets of vectors, or in an even more restricted setup. There is no need to deal with input in the first case, and in the latter case the final configuration itself is irrelevant (except yes or no in case of the efficiency research). On the other side, both input and output are critical for most applications.

C) Representation. Clearly, any kind of discrete information can be encoded in a single integer in some consistent way. However, a much more transparent data representation is typically required; even the intermediate configurations in a computation are expected to reflect a state of the object in the problem area.

D) Efficiency. Suppose numbers are represented by multiplicities of certain objects. The number of steps needed to multiply two numbers by plain (cooperative) multiset processing is proportional to the result. If the multiset processing can be controlled by promoters/inhibitors/priorities, then the number of steps needed for multiplication is proportional to one of the arguments. However, many applications would ask for a multiplication to be performed in a constant number of steps. Similar problems appear for string processing.

E) Data structures. Membrane computing deals with multisets distributed over a graph, while conventional computers provide random memory access and pointer operations, allowing much more complex structures to be built.

Some of these implicit requirements originate because the user wants a solution which is at least as good as the one that can be provided by conventional computers.

We introduced a new feature into the membrane computing. This time the inspiration is not biological, but rather is from the area of conventional computing.

Suppose we want to be able to manipulate the rules of the system during its computation. A number of papers has been written about this but in most of them the rules are predefined in the description of the system. The most natural way to manipulate the rules is to represent them as data, treat this data as rules, and manipulate it as usual in P systems, in the spirit of von Neumann's approach. In membrane systems, the data consists of multisets, so objects should be treated as description of the rules. Informally, a rule j in a region i can be represented by the contents of membranes jL and jR inside i. Changing the contents of regions jL and jR results in the corresponding change of the rule j. We call such P systems polymorphic, by analogy with polymorphic or self-modifying computer programs.

At the same time, if a membrane system is an abstraction inspired by the biological cell, one can view inner regions as an abstraction inspired by the cell nucleus; their contents correspond to the genes encoding the enzymes performing the reactions of the system.

The simplicity of the proposed model is that we consider the natural encoding, i.e., no encoding at all: the multisets describing the rules are represented by exactly themselves. Therefore, we are addressing a problem informally stated by Gh. Păun "Where Is the Nucleus?" by proposing a computational variant based on one simple difference: the rules are taken from the current configuration rather than from the description of the P system itself.

3.4 Insertion-Deletion (P) Systems

We considered models of biocomputing based on insertion and deletion operations of small size [80, 21, 22, 35, 36, 57, 68, 65, 66, 64, 67, 69, 81]. The insertion and the deletion operations originate from the language theory, where they where introduced mainly with linguistic motivation. In general form, an insertion operation means adding a substring to a given string in a specified (left and right) context, while a deletion

operation means removing a substring of a given string from a specified (left and right) context. A finite set of insertion-deletion rules, together with a set of axioms provide a language generating device: starting from the set of initial strings and iterating insertion-deletion operations as defined by the given rules we get a language.

In the last years, the study of these operations has received a new motivation from molecular computing, because, from the biological point of view, insertion-deletion operations correspond to mismatched annealing of DNA sequences. As expected, insertion-deletion systems are quite powerful, leading to characterizations of recursively enumerable languages. This is not quite surprising as the corresponding device contains two important ingredients needed for the universality: the context dependency and the erasing ability. However, as it was shown in [80], the context dependency may be replaced by insertion and deletion of strings of sufficient length, in a context-free manner. If the length is not sufficient (less than two) then such systems are decidable and a characterization of them was shown by S.Verlan in [107].

Similar investigations were continued in [5, 4] on insertion-deletion systems with one-sided contexts, i.e. where the context dependency is present only from the left (right) side of all insertion and deletion rules. These articles also give some combinations of rule parameters that lead to systems, which are not computationally complete. However, if these systems are combined with the distributed computing framework of P systems, then their computational power may strictly increase [64, 68].

In [21, 35] we study P systems with context-free insertion and deletion rules of one symbol. We show that this family is strictly included in MAT, however some non-context-free languages may be generated. If Parikh vectors are considered, then the corresponding family equals to PsMAT. When a priority of deletion over insertion is introduced, PsRE can be characterized, but in terms of language generation such systems cannot generate a lot of languages because there is no control on the position of an inserted symbol. If one-sided contextual insertion or deletion rules are used, then this can be controlled and all recursively enumerable languages can be generated.

The same result holds if a context-free deletion of two symbols is

allowed.

3.5 Splicing (P) Systems

It is known that H systems are not very powerful, so, a lot of other models introducing additional control elements were proposed. Another extension of H systems was done using the framework of P systems (see [109]). In a formal way, splicing P systems can be considered like a graph, whose nodes contain sets of strings and sets of splicing rules. Every rule permits to perform a splicing and to send the result to some other node. Since splicing P systems generate any recursively enumerable language, it is clear that there are universal splicing P systems.

Like for small universal Turing machines, we are interested in such universal systems that have a small (smallest) number of splicing rules. A first result was obtained by Yu.Rogozhin and S.Verlan in [97] where a universal splicing P system with 8 rules was shown. Similar investigations for P systems with symbol-objects were done in [11, 39] and the latter article constructs a universal antiport P system with 23 rules.

In [38] we provided a new construction for splicing P systems and proved the remarkable fact that 6 splicing rules are powerful enough for the universality. In [34] we presented a series of small universal devices (splicing systems):

- two universal time-varying distributed H systems: of degree 2 with 15 rules and of degree 1 with 17 rules,
- and also three universal splicing test tube systems with 3 or 2 test tubes and 10 rules.

Test tube systems based on splicing, introduced in E. Csuhaj-Varjú et al. in [47] are symbol processing mechanism with components (test tubes) working as splicing schemes in the sense of T. Head, and communicating through redistribution of the contents of the test tubes via filters. These systems with finite initial contents of the tubes and finite sets of splicing rules associated to each component are computationally complete; they characterize the family of recursively enumerable

languages. The existence of universal test tube distributed systems was obtained on this basis, hence there is the theoretical possibility to design universal programmable computers with the structure of such a system.

Since 1996, a lot of variants of test tube systems have been introduced and studied, using filtering of the strings by patterns. A natural question is whether or not such systems with other types of filtering mechanisms, based on techniques widely used in laboratory, can be defined. We introduced a new variant of test tube systems, length-separating test tube systems, based on splicing where the communication of the words among the test tubes is based on filtering by their lengths, motivated by the gel electrophoresis laboratory technique [50]. However, we remark that filtering by length can also be done by other methods, like size exclusion chromatography, that permits to separate molecules depending on their size.

Gel electrophoresis is a technique for separation of molecules which is widely used in the laboratory. It is usually performed for analytical purposes at the final stage of the experiment. Its formal counterpart, the length separation, is a standard tool in DNA computing. For example, it is used in the third step of the Adleman's experiment in 1994. There are also algorithms like in F. Guarnieri et al. [58] based on the length separation which use it at the end of the computation in order to confirm or select the result. In Y. Khodor et al. [63], a method called length-only discrimination based on the generate-and-search approach but relying on the length of the sequence is presented and experimentally confirmed.

We use the length separation in another way. We consider a distributed system and we use the length separation for the communication between the components of the system. Since such systems work iteratively, the length separation, used at each step, becomes one of the main ingredients of the model. In an informal way, our model corresponds to the following experiment.

Let us suppose that there is a set of test tubes. Each of these test tubes may transform DNA molecules (cut, ligate, multiply etc). The tubes are selective and they can do their transformations only

on specific molecules (for example, in a tube DNA molecules may be cut with a specific enzyme, hence only molecules having a corresponding site will be modified). Taking a tube, we may put some amount of DNA molecules into it. After the transformation, all molecules from a tube are put in a gel electrophoresis. After the separation, the gel is cut at some points corresponding to some molecular lengths. Hence, molecules will be grouped by some length intervals. After that, molecules are extracted from the gel and distributed among other test tubes depending on their molecular length interval.

The above process may be iterated and, since all tubes are organized in a network, some interesting transformations may be done. The initial DNA molecules are put in some fixed tube, and the transformed molecules are collected in the output tube. To separate molecules to be transmitted from one tube to another one, we define different conditions. These conditions are exclusive, namely, each string (molecule) found in a tube can be forwarded to only one tube. All molecules in the test tube are communicated to some tube (depending on the underlying graph of the test tube system, might remain at the original tube). One type of these conditions are variants of communication by fixed (bounded) length where strings of length equal to (or at most equal to or at least equal to) a fixed constant are communicated to another tube. In terms of gel electrophoresis this corresponds to the cut at some specific points depending on the marker molecules.

Other types of conditions involve molecules of maximal and minimal size as well as their negations (not maximal and not minimal). From the gel point of view, this corresponds to the selection of the first or the last molecule from the gel (the other molecules correspond to the negation variant). We study the computational power of these constructs. We show that the length separating test tube systems, even with very restricted size parameters, are able to simulate the Turing machines. This result holds in general case as well as for systems only using communication conditions based on separation of molecules by maximal (resp. minimal) and not maximal (resp. not minimal) length and on the ability checking whether the word (the molecule) differs from the empty word. These results correspond to our expectations,

due to the nature of the splicing operation.

If we restrict the communication conditions to select molecules with fixed or bounded length, then the computational power of the corresponding systems is not known. Although using appropriate communication predicates our construction has the power of the Turing machines, this does not help in efficiently solving practical problems. For example, given a particular molecule, can we design a system that will perform a particular transformation on it? Moreover, this transformation should be efficient, i.e., it shall be done in the smallest possible number of steps, involving the smallest number of high-cost operations. This problem is difficult to solve.

Here we provided a theoretical framework that could be used to describe and possibly answer the above questions. However, we mainly focus on the network structure and length filtering; the operation-related improvements remain to be further investigated.

3.6 Reversibility and P Systems

Membrane computing is a formal framework of distributed parallel computing. We studied the reversibility and maximal parallelism of P systems from the computability point of view [32, 24, 25, 37]. The notions of reversible and strongly reversible systems are considered.

The universality is shown for reversible P systems with either priorities or inhibitors, and a negative conjecture is stated for reversible P systems without such control. Strongly reversible P systems without control have shown to only generate sub-finite sets of numbers; this limitation does not hold if inhibitors are used.

Another concept considered is strong determinism which is a syntactic property, as opposed to the determinism typically considered in membrane computing. Strongly deterministic P systems without control only accept sub-regular sets of numbers, while systems with promoters and inhibitors are universal.

3.7 P Systems with Active Membranes

Membrane systems are a convenient framework of describing polynomial-time solutions to certain intractable problems in a massively parallel way. Division of membranes makes it possible to create an exponential space in linear time, suitable for attacking problems in NP and even in PSPACE. Their solutions by so-called P systems with active membranes have been investigated in a number of papers since 2001, later focusing on solutions by restricted systems (see, for example, The Oxford Handbook of Membrane Computing, ed. by G. Paun, G. Rozenberg, A. Salomaa [87]). The description of rules in P systems with active membranes involves membranes and objects; the typical types of rules are object evolution, object communication, membrane dissolution, membrane division.

Our goal was to implement methods of P systems with active membranes in **computer algebra** and particularly, to generalize the approach from decisional problems to the computational ones, by considering a #P-complete (pronounced sharp-P complete) problem of computing the permanent of a binary matrix [5, 14].

Commutative and non-commutative Computer Algebra Systems were analyzed. The systems were analyzed taking into account efficiency, termination of calculations, and some technical details of their implementation. Existing methods of parallel calculations used with Computer Algebra Systems were also examined. We studied a series of problems in the matrix theory (permanent calculation), Grobner base theory (determination if the algebra has the finite dimension, checking if a given set of polynomials forms the Grobner base of the given algebra). We used P-lingua system [95] to simulate elaborated algorithms. We found that the said system does not meet all necessities to simulate monomials manipulations. We proposed to extend it with the mechanism of rule parameterization.

The domain of Computer Algebra is characterized by problems of the high calculation complexity. Therefore, the interest in effective methods of their solution is justified. The results of our research demonstrate that natural calculations make an effective mechanism to solve

problems in this domain, in particular, for non-commutative algebras where the computing processes can be infinite.

Other our goal was to implement methods of P systems with active membranes in **mathematical linguistics**.

Solving most problems of natural language processing is based on using certain linguistic resources, represented by corpora, lexicons, etc. Usually, these collections of data constitute an enormous volume of information, so processing them requires much computational resources. A reasonable approach for obtaining efficient solutions is that based on applying parallelism; this idea has been promoted already in 1970's. Many of the stages of text processing (from tokenization, segmentation, lematizing to those dealing with natural language understanding) can be carried out by parallel methods. This justifies the interest to the methods offered by the biologically inspired models, and by membrane computing in particular.

However, there are some issues that by their nature do not allow complete parallelization, yet exactly they are often those "computational primitives" that are inevitably used during solving major problems, like the elementary arithmetic operations are always present in solving difficult computational problems. Among such "primitives" in the computational linguistics we mention handling of the dictionaries, e.g., dictionary lookup and dictionary update. Exactly these problems constitute the subject of our work.

In our approach we speak about dictionary represented by a prefix tree and P systems with active membranes that are a convenient framework of describing computations on trees [17, 16, 46]. In [13, 15, 41, 42, 43] we formalised inflection process for the Romanian language using the model of P systems with cooperative string replication rules, which will make it possible to automatically build the morphological lexicons as a base for different linguistic applications.

3.8 Networks of Evolutionary Processors

Motivated by some models of massively parallel computer architectures (see [53] and [60]), networks of language processors have been

introduced in 1997 by E. Csuhaj-Varjú and A. Salomaa [48]. Such a network can be considered as a graph where the nodes are sets of productions and at any moment of time a language is associated with a node. In a *derivation* step, any node derives from its language all possible words as its new language. In a *communication* step, any node sends those words to other nodes that satisfy an output condition given as a regular language, and any node takes those words sent by the other nodes that satisfy an input condition also given by a regular language. The language generated by a network of language processors consists of all (terminal) words which occur in the languages associated with a given node.

Inspired by biological processes, J. Castellanos, C. Martín-Vide, V. Mitrana and J. Sempere introduced in [44] a special type of networks of language processors which are called networks with evolutionary processors because the allowed productions model the point mutation known from biology. The sets of productions have to be substitutions of one letter by another letter or insertions of letters or deletion of letters; the nodes are then called substitution node or insertion node or deletion node, respectively.

It was shown by A. Alhazov et al. in [3] that networks of evolutionary processors are universal in that sense that they can generate any recursively enumerable language and that networks with three nodes are sufficient to get all recursively enumerable languages. The proof uses one node of each type (and intersection with a monoid). Therefore it is a natural question to study the power of networks with evolutionary processors where the nodes have only two types, i. e.,

(i) networks with deletion nodes and substitution nodes (but without insertion nodes),

(ii) networks with insertion nodes and substitution nodes (but without deletion nodes), and

(iii) networks with deletion nodes and insertion nodes (but without substitution nodes).

We investigated the power of such systems and studied the number

of nodes sufficient to generate all languages which can be obtained by networks of the type under consideration. We prove that networks of type (i) and (ii) produce only finite and context-sensitive languages, respectively. Every finite, context-sensitive or recursively enumerable language can be generated by a network of type (i) with one node, by a network of type (ii) with two nodes or by a network of type (iii) with two nodes, respectively [19].

Particularly interesting variants of these devices are the so-called hybrid networks of evolutionary processors (HNEPs), where each language processor performs only one of the above operations on a certain position of the words in that node. Furthermore, the filters are defined by some variants of random-context conditions, i.e., they check the presence/absence of certain symbols in the words. These constructs can be considered both language generating and accepting devices, i.e., generating HNEPs (GHNEPs) and accepting HNEPS (AHNEPs).

In [49] E. Chuhaj-Varjú et al. showed that, for an alphabet V, GHNEPs with $27 + 3 \cdot card(V)$ nodes are computationally complete. A significant improvement of the result can be found in [6, 7], where we proved that GHNEPs with 10 nodes (irrespectively of the size of the alphabet) obtain the universal power. Recently [8, 18] we improved this result and showed that any recursively enumerable language can be generated by a GHNEP having 7 nodes and it can be accepted by an AHNEP with the same number of nodes. We also show that the families of GHNEPs and AHNEPs with 2 nodes are not computationally complete. Although the sharpness of the upper bounds is not verified, we considerably improved the previous results. The gap between universality and non-universality for GHNEPs now is very small (it is the same as for the famous PCP problem). In [10] we completed investigation of HNEPs with one node and presented a precise description of languages generated by them.

We considered new variant of HNEP, so called Obligatory Network of Evolutionary Processors (OHNEP shortly) [12]. The differences between HNEP and OHNEP are:

1) in using deletion and substitution operations: a node discards a string if no operations in node are applicable to string (in HNEP

case this string remains in the node),

2) an underlying graph is directed graph (in HNEP case this graph is undirected).

We underline that both differences are natural.

The first one allows us to have the uniform definitions of the operations on a string, as opposed to considering two cases as in HNEPs (it is the set of results of the applications of the operation to all possible positions; the case when there are no such positions yields the empty set by definition).

The second difference, that of generalization of the underlying graph to be directed, is natural from the computational point of view; moreover, since the loops are typically not considered, it also seems relevant from the viewpoint of the biological motivation that the communicating channels are directed.

These differences allow proofing universality of OHNEP with nodes with only one operation, without input and output filters and using only insertion operation at the left end and deletion operation at the right end of a string. This interesting fact stresses the importance of structure of HNEP in order to reach universality.

On the other hand we can avoid substitution operation. Notice that this feature of OHNEP to discard a string if this string does not participate at the operations has counterpart in DNA computing area, TVDH systems also discard strings if they do not participate at splicing operations ([73]). A task to find a minimal number of nodes of universal OHNEP is open. A variant of OHNEP with underlying complete graph is not considered yet.

An implementation of HNEPs and OHNEPs in mathematical linguistics is also interesting task to investigate. The constructions demonstrate that distributed architectures of very small size, with uniform structure and with components based on very simple language theoretic operations are sufficient both to generate and to recognize any recursively enumerable language.

3.9 Other Models of Natural Computing

A number-conserving cellular automaton (NCCA) is a cellular automaton whose states are integers and whose transition function keeps the sum of all cells constant throughout its evolution. It can be seen as a kind of modeling of the physical conservation laws of mass or energy.

We showed a construction method of NCCAs with radius 1/2 [61]. The local transition function is expressed via a single unary function which can be regarded as "flows" of numbers. In spite of the strong constraint, we constructed NCCAs with radius 1/2 that simulate any cellular automata with radius 1/2 or any NCCA with radius 1. We also consider the state complexity of these non-splitting simulations ($4n^2 + 2n + 1$ and $8n^2 + 12n - 16$, respectively). These results also imply existence of intrinsically universal NCCA with radius 1/2.

A reversible logic element is a primitive from which reversible computing systems can be constructed. A rotary element is a typical 2-state 4-symbol reversible element with logical universality, and we can construct reversible Turing machines from it very simply.

There are also many other reversible elements with 1-bit memory. So far, it is known that all the 14 kinds of non-degenerate 2-state 3-symbol reversible elements can simulate a Fredkin gate, and hence they are universal. We showed that all these 14 elements can "directly" simulate a rotary element in a simple and systematic way [84, 85, 86].

Acknowledgments All authors acknowledge the support by the Science and Technology Center in Ukraine, project 4032.

References

[1] L.M.Adleman: *Molecular computation of solutions of combinatorial problems*, Science, 226, pp. 1021–1024, 1994.

[2] A.Alhazov: *Maximally Parallel Multiset-Rewriting Systems: Browsing the Configurations.* In: M.A. Gutiérrez-Naranjo, A. Riscos-Núñez, F.J. Romero-Campero, D. Sburlan: RGNC re-

port 01/2005, University of Seville, *Third Brainstorming Week on Membrane Computing*, Fénix Editora, Sevilla, 2005, pp. 1–10.

[3] A.Alhazov, C.Martín-Vide, Yu.Rogozhin: *On the number of nodes in universal networks of evolutionary processors*, Acta Inf. 43 (2006) pp. 331–339.

[4] A.Alhazov: *Ciliate Operations without Context in a Membrane Computing Framework.* Romanian Journal of Information Science and Technology, vol.10, no.4 (2007), pp. 315–322.

[5] A.Alhazov, L.Burtseva, S.Cojocaru, Y.Rogozhin: *Computing solutions of #P-complete problems by P systems with active membranes.* In: Proceedings of Ninth Workshop on Membrane Computing (WMC9), Edinburgh, UK, July 28 - 31, 2008, pp. 59–70.

[6] A.Alhazov, E.Csuhaj-Varjú, C.Martín-Vide, Yu.Rogozhin: *About Universal Hybrid Networks of Evolutionary Processors of Small size.* In: Pre-proceedings of the 2nd International Conference on Language and Automata, Theory and Applications, LATA 2008, March 13 - 19, 2008, Rovira i Virgili University, Tarragona, Spain. Technical Report of GRLMC No. 36/08, Universitat Rovira i Virgili, Tarragona, Spain (2008), pp. 43–54.

[7] A.Alhazov, E.Csuhaj-Varjú, C.Martín-Vide, Y.Rogozhin: *About Universal Hybrid Networks of Evolutionary Processors of Small Size.* Lecture Notes in Computer Science, Springer, 5196, pp. 28–39, 2008.

[8] A.Alhazov, E.Csuhaj-Varjú, C.Martín-Vide, Y.Rogozhin: *Computational Completeness of Hybrid Networks of Evolutionary Processors with Seven Nodes.* In: Proceedings of the Workshop DCFS 2008, Descriptional Complexity of Formal Systems, Charlottetown, Prince Edward Island, Canada, July 16-18, 2008.

[9] A.Alhazov, M.Margenstern, S.Verlan: *Fast synchronization in P systems. In: Proceedings of Ninth Workshop on Membrane Computing (WMC9)*, Edinburgh, UK, July 28 - 31, 2008, pp. 59–70.

[10] A.Alhazov, Yu.Rogozhin: *About Precise Characterization of Languages Generated by Hybrid Networks of Evolutionary Processors with One Node.* Computer Science Journal of Moldova, 16, no.3 (48) (2008), pp. 364–376.

[11] A.Alhazov, S.Verlan: *Minimization Strategies for Maximally Parallel Multiset Rewriting Systems.* Technical Report of Turku Centre for Computer Science, Turku, Finland, no. 862, (2008).

[12] A.Alhazov, G. Bel-Enguix, Yu.Rogozhin: *Obligatory Hybrid Networks of Evolutionary Processors.* In: Proc. of the First International Conference on Agents and Artificial Intelligents, ICAART 2009, Porto, Portugal, 19-21 January, 2009, pp. 613–618.

[13] A.Alhazov, E.Boian, S.Cojocaru, Yu.Rogozhin: *Modelling Inflections in Romanian Language by P Systems with String Replication.* In: Proc. of the 10th Workshop on Membrane Computing, WMC10, Curtea de Arges (Romania), August 24 - 27, 2009, pp. 116–128.

[14] A.Alhazov, L.Burtseva, S.Cojocaru, Yu.Rogozhin: *Solving PP-Complete and #P-Complete Problems by P Systems with Active Membrane.* Lecture Notes in Computer Science, Springer, 5391 (2009), pp. 108–117.

[15] A.Alhazov, E.Boian, S.Cojocaru, Yu.Rogozhin: *Modelling Inflections in Romanian Language by P Systems with String Replication.* Computer Science Journal of Moldova, vol.17, no.2(50), 2009, pp. 160–178.

[16] A.Alhazov, S.Cojocaru, L.Malahova, Yu.Rogozhin: *Dictionary Search and Update by P Systems with String-Objects and Active Membranes.* International Journal of Computers, Communications and Control, Vol. IV, No. 3 (2009), pp. 206–213.

[17] A.Alhazov, S.Cojocaru, L.Malahova, Yu.Rogozhin: *Dictionary Search and Update by P Systems with String-Objects and Active Membranes.* In: Proc. of the 7th Brainstorming Week on Membrane Computing, Sevilla, Spain, February 2-February 6, 2009. Universidad de Sevilla, RGNC REPORT 1/2009, pp. 1–8.

[18] A.Alhazov, E.Csuhaj-Varjú, C.Martín-Vide, Yu.Rogozhin: *On the size computationally complete hybrid networks of evolutionary processors.* Theoretical Computer Science, Elsevier, 410 (2009), pp. 3188–3197.

[19] A.Alhazov, J.Dassow, C.Martín-Vide, Yu.Rogozhin, B.Truthe: *On Networks of Evolutionary Processors with Nodes of Two Types.* Fundamenta Informaticae, IOS Press, 91(1) (2009), pp. 1–15.

[20] A.Alhazov, R.Freund, M.Oswald, S.Verlan: *Partial Halting and Minimal Parallelism Based on Arbitrary Rule Partitions.* Fundamenta Informaticae, IOS Press, 91(1) (2009), pp. 17–34.

[21] A.Alhazov, A.Krassovitskiy, Yu.Rogozhin, S.Verlan: *P Systems with Minimal Insertion and Deletion.* In: Proc. of the 7th Brainstorming Week on Membrane Computing, Sevilla, Spain, February 2 - February 6, 2009. Universidad de Sivilla, RGNC REPORT 1/2009, pp. 9–21.

[22] A.Alhazov, A.Krassovitskiy, Yu.Rogozhin, S.Verlan: *A Note on P Systems with Small-Size Insertion and Deletion.* In: Proc. of the 10th Workshop on Membrane Computing, WMC10, Curtea de Arges (Romania), August 24 - 27, 2009, pp. 534–537.

[23] A.Alhazov, M.Margenstern, S.Verlan: *Fast Synchronization in P Systems.* In: D.W. Corne, P. Frisco, Gh. P?un, G. Rozenberg, A. Salomaa: Membrane Computing - 9th International Workshop, WMC 2008, Edinburgh, Revised Selected and Invited Papers, Lecture Notes in Computer Science vol. 5391, Springer, 2009, pp. 118–128.

[24] A.Alhazov, K.Morita: *A Short Note on Reversibility in P Systems.* In: Proc. of the 7th Brainstorming Week on Membrane Computing, Sevilla, Spain, February 2-February 6, 2009. Universidad de Sevilla, RGNC REPORT 1/2009, pp. 23–28.

[25] A.Alhazov, K.Morita: *On Reversibility and Determinism in P systems.* In: Proc. of the 10th Workshop on Membrane Comput-

ing, WMC10, Curtea de Arges (Romania), August 24 - 27, 2009, pp. 129–139.

[26] A.Alhazov, I.Petre, V.Rogojin: *The parallel complexity of signed graphs: decidability results and an improved algorithm.* Theoretical Computer Science, 410, Elsevier, (2009, pp. 2308–2315.

[27] A. Alhazov, C.Ciubotaru, Yu.Rogozhin, S.Ivanov: *The Family of Languages Generated by Non-Cooperative Membrane Systems.* In: M. Gheorghe, Th. Hinze, Gh. Păun: *Preproceedings of the Eleventh Conference on Membrane Computing, CMC11*, Jena, Verlag ProBusiness Berlin, 2010, pp. 37–51, and *Lecture Notes in Computer Science* 6501, to appear.

[28] A.Alhazov, C.Ciubotaru, Yu.Rogozhin, S.Ivanov: *The Membrane Systems Language Class.* LA symposium, RIMS Kôkyûroku Series vol. 1691, Kyoto University, 2010, pp. 44–50.

[29] A.Alhazov, C.Ciubotaru, Yu.Rogozhin, S.Ivanov: *Introduction to the Membrane Systems Language Class.* In: Proceedings of the 3rd International Conference "Telecommunications, Electronics and Informatics", ICTEI 2010, vol. II, Chişinău, 2010, pp. 19–24.

[30] A.Alhazov, C.Ciubotaru, Yu.Rogozhin, S.Ivanov: *The Membrane Systems Language Class.* In: Proceedings of the Eighth Brainstorming Week on Membrane Computing, Sevilla, Spain, February 1-5, 2010, RGNC Report 1/2010, Fenix Editora, Sevilla, 2010, pp. 23–36.

[31] A.Alhazov, C.Ciubotaru, Y.Rogozhin, S.Ivanov: *The family of Languages Generated by Non-Cooperative Membrane Systems.* In: Proceedings of the Eleventh International Conference on Membrane Computing. Friedrich Schiller University Jena, Germany, 24-27 August, 2010, pp. 37–52.

[32] A.Alhazov, R.Freund, K.Morita: *Reversibility and Determinism in Sequential Multiset Rewriting.* Unconventional Computation 2010, Tokyo, Lecture Notes in Computer Science, Springer, vol. 6079, 2010, pp. 21–31.

[33] A.Alhazov, S.Ivanov, Y.Rogozhin: *Polymorphic P Systems.* In: Proceedings of the Eleventh International Conference on Membrane Computing. Friedrich Schiller University Jena, Germany, 24-27 August, 2010, pp. 53–66.

[34] A.Alhazov, M.Kogler, M.Margenstern, Yu.Rogozhin, S.Verlan: *Small Universal TVDH and Test Tube Systems.* International Journal of Foundations of Computer Science, 2010 (in press).

[35] A.Alhazov, A.Krassovitskiy, Yu.Rogozhin, S.Verlan: *P systems with minimal insertion and deletion.* Theoretical Computer Science, Elsevier, 2010 (in press).

[36] A.Alhazov, A.Krassovitskiy, Yu.Rogozhin, S.Verlan: *Small Size Insertion and Deletion Systems.* In: Mathematics, Computing, Language, and Life: Frontiers in Mathematical Linguistics and Language Theory, Vol.2. Scientific Applications of Language Methods. World Scientific, 2010 (in press).

[37] A.Alhazov, K. Morita: On Reversibility and Determinism in P Systems. Workshop on Membrane Computing, WMC10, Curtea de Arges, 2009, Lecture Notes in Computer Science, Springer 5957, 2010, pp. 158–168.

[38] A.Alhazov, Y.Rogozhin, S.Verlan: *A Small Universal P Systems.* In: Proceedings of the Eleventh International Conference on Membrane Computing. Friedrich Schiller University Jena, Germany, 24-27 August, 2010, pp. 67–74.

[39] A.Alhazov, S.Verlan: *Minimization Strategies for Maximally Parallel Multiset Rewriting Systems.* arXiv:1009.2706v1 [cs.FL] 14 Sep 2010, http://arxiv.org/abs/1009.2706.

[40] C.H. Bennett, *Logical Reversibility of Computation*, IBM J. Res. Develop. 6, pp. 525–532, 1973.

[41] E.Boian, C.Ciubotaru, S.Cojocaru, A.Colesnicov, V.Demidova, L.Malahov: *P-systems application for solution of some problems in computational linguistics.* Proceedings of the International

Conference ICT+ "Information and Communication Technologies - 2009". May 18-21, 2009, Chisinau, Republic of Moldova. pp. 30–33, ISBN 978-9975-66-134-8. (in Romanian)

[42] E.Boian, S.Cojocaru, A.Colesnicov, C.Ciubotaru, L.Malahova: *Using the P systems in computer linguistic applications (in Russian).* In: Proceedings of the International Conference "Horizons of Applied Linguistics and Linguistics Technologies", Kiev, Ukraine, 21 - 26 September, 2009. p.53.

[43] E.Boian, S.Cojocaru, V.Macari, G.Magariu, T.Verlan: On simulation of inflection process in Romanian language by P-systems with string replication. In CD: Proceedings of the ECIT2010 - 6th European Conference on Intelligent Systems and Technologies. Iasi, Romania, October 07-09, 2010.

[44] J.Castellanos, C.Martń-Vide, V.Mitrana, J.Sempere: *Solving NP-complete problems with networks of evolutionary processors,* In: Proc. IWANN, Lecture Notes in Computer Science 2084, Springer-Verlag, Berlin, 2001, pp. 621–628.

[45] J.Cocke, M.Minsky: *Universality of tag systems with P=2,* Journal of the ACM, 11(1), 1964, pp. 15–20.

[46] S.Cojocaru, E.Boian: *Determination of inflexional group using P systems.* Computer Science Journal of Moldova, vol.18, no.1(52), 2010, pp. 70–81.

[47] E.Csuhaj-Varjù, L.Kari, G.Păun, *Test Tube distributed system based on splicing*, Computer and AI, 2–3, pp. 211–232, 1996.

[48] E.Csuhaj-Varjú, A.Salomaa: *Networks of parallel language processors,* In: New Trends in formal Language Theory (Gh. Păun, A. Salomaa, Eds.), Lecture Notes in Computer Science 1218, Springer-Verlag, Berlin, 1997, pp. 299–318.

[49] E.Csuhaj-Varjú, C.Martín-Vide, V.Mitrana: *Hybrid networks of evolutionary processors are computationally complete,* Acta Inf. 41 (2005), pp. 257–272.

[50] E.Csuhaj-Varjú, S.Verlan: *On length-separating test tube systems.* Natural computing, Springer, vol.7, no.2, 2008, pp. 167–181.

[51] E.Csuhaj-Varjú, S.Verlan: *Power and Size of Generalized Communicating P Systems with Minimal Interaction Rules.* In: Proc. of the 10th Workshop on Membrane Computing, WMC10, Curtea de Arges (Romania), August 24 - 27, 2009, pp. 547–551.

[52] E.Csuhaj-Varjú, G.Vaszil, S.Verlan: *On Generalized Communicating P Systems with One Symbol.* In: Proceedings of the Eleventh International Conference on Membrane Computing. Friedrich Schiller University Jena, Germany, 24-27 August, 2010, pp. 137–154.

[53] S.E.Fahlmann, G.E.Hinton, T.J.Seijnowski: *Massively parallel architectures for AI: NETL, THISTLE and Boltzmann machines,* In: Proc. AAAI National Conf. on AI, William Kaufman, Los Altos, 1983, pp. 109–113.

[54] C.Ferretti, G.Mauri, C.Zandron: *Nine Test Tubes Generate any RE Language*, personal communication.

[55] R.Freund, S.Verlan: *(Tissue) P systems working in the k-restricted minimally parallel derivation mode.* In: Proceedings of International Workshop on Computing with Biomolecules, August 27th, 2008, Wien, Austria, pp. 43–52.

[56] R.Freund, A.Alhazov, Y.Rogozhin, S.Verlan: *Communication P systems.* In: The Oxford Handbook of Membrane Computing, ed. by Gh.Păun, G.Rozenberg, A.Salomaa, Oxford University Press, 2010.

[57] R.Freund, M.Kogler, Yu.Rogozhin, S.Verlan: *Graph-Controlled Insertion-Deletion Systems.* In: Proceedings DCFS 2010. EPTCS 31, 2010, pp. 88–98, doi:10.4204/EPTCS.31.11.

[58] F.Guarnieri, M.Fliss, C.Bacroft: *Making DNA add.* Science (1996) 273(12):220–223.

[59] T.Head *Formal Language Theory and DNA: An Analysis of the Generative Capacity of Specific Recombinant Behaviors*, Bulletin of Mathematical Biology, Vol. 49, No. 6, pp. 737–759, 1987.

[60] W.D.Hillis: *The Connection Machine.* MIT Press, Cambridge, 1985.

[61] K.Imai, A.Alhazov: *On Universality of Radius 1/2 Number-Conserving Cellular Automata.* Unconventional Computation 2010, Tokyo, Lecture Notes in Computer Science, Springer, vol. 6079, 2010, pp. 45–55.

[62] S.Ivanov, V.Macari: *CUDA in Simulating P Systems.* In: Proceedings of the 3rd International Conference "Telecommunications, Electronics and Informatics" ICTEI 2010, Volume II, pp. 198–201.

[63] Y.Khodor, J.Khodor, T.F.Jr.Knight: *Experimental confirmation of the basic principles of length-only discrimination.* In: Jonoska N, Seeman NC (eds) DNA7. Lecture notes in computer science, vol 2340. Springer-Verlag, Berlin (2002), pp. 223–230.

[64] A.Krassovitskiy, Y.Rogozhin, S.Verlan: *One-sided Insertion and Deletion: Traditional and P Systems Case.* In: Proceedings of International Workshop on Computing with Biomolecules, August 27th, 2008, Wien, Austria, pp. 53–64.

[65] A.Krassovitskiy, Y.Rogozhin, S.Verlan: *Further results on insertion-deletion systems with one-sided contexts.* In: Preproceedings of the 2nd International Conference on Language and Automata, Theory and Applications, LATA 2008, March 13 - 19, 2008, Rovira i Virgili University, Tarragona, Spain. Technical Report of GRLMC No. 36/08, Universitat Rovira i Virgili, Tarragona, Spain (2008), pp. 347–358.

[66] A.Krassovitskiy, Y.Rogozhin, S.Verlan: *Further results on insertion-deletion systems with one-sided contexts.* Lecture Notes in Computer Science, Springer, 5196, pp. 333–344, 2008.

[67] A.Krassovitskiy, Y.Rogozhin, S.Verlan: *Computational Power of P Systems with Small Size Insertion and Deletion Rules.* In: Proc. of the International Workshop on The Complexity of Simple Programs, University College Park, Ireland, December 6th and 7th, 2008, pp. 137–148.

[68] A.Krassovitskiy, Yu.Rogozhin, S.Verlan: *Computational Power of P Systems with Small size Insertion and Deletion Rules.* EPTCS 1, 2009, pp. 108–117, doi:10.4204/EPTCS.1.10.

[69] A.Krassovitskiy, Yu.Rogozhin, S.Verlan: *Computational power of insertion-deletion (P) systems with rules of size two.* Natural Computing, Springer, DOI 10.1007/s11047-010-9208-y, 2010 (in press).

[70] M.Margenstern, Yu.Rogozhin: *A Universal Time-Varying Distributed H-System of Degree 2,* Preliminary Proceedings of Forth International Meeting on DNA Based Computers, June 15-19, 1998, University of Pennsylvania, U.S.A., 1998.

[71] M.Margenstern, Yu.Rogozhin: *A Universal Time-Varying Distributed H System of Degree 2*, Biosystems, 52, 1999, pp. 73–80.

[72] M.Margenstern, Yu.Rogozhin: *Generating All Recursively Enumerable Languages with a Time-Varying Distributed H System of Degree 2*, Technical report, Institut Universitaire de Technologie de Metz, 1999, Publications du G.I.F.M.

[73] M.Margenstern, Yu.Rogozhin: *About time-varying distributed H systems,* DNA Computing: 6th International Workshop on DNA-Based Computers, DNA 2000, Leiden, The Netherlands, June 13-17, 2000, LNCS, Springer, Revised Papers (A. Condon, G. Rozenberg, Eds.), 2054 (2000), pp. 53–62.

[74] M.Margenstern, Yu.Rogozhin: *Time-Varying Distributed H systems of Degree 2 Generate All Recursively Enumerable Languages*, in: Where Do Mathematics, Computer Science and Biology Meet (C. Martin-Vide, V. Mitrana, Eds.), Kluwer Academic, Dortrecht, 2000, pp. 399–407.

[75] M.Margenstern, Yu.Rogozhin: *Extended Time-Varying Distributed H Systems - Universality Result*, Proceedings of The 5th World Multi-Conference on Systemics, Cybernetics and Informatics, Industrial Systems, SCI 2001, Orlando, Florida USA, July 22-25, 2001, IX, 2001.

[76] M.Margenstern, Yu.Rogozhin: *Time-Varying Distributed H Systems of Degree 1 Generate All Recursively Enumerable Languages*, In: Words, Semigroups, and Transductions (M. Ito, G. Păun, S. Yu, Eds.), World Scientific, Singapore, 2001, ISBN 981-02-4739-7, pp. 329–340, Festschrift in Honor of Gabriel Thierrin.

[77] M.Margenstern, Yu.Rogozhin: *A Universal Time-Varying Distributed H System of Degree 1.,* DNA Computing: 7th International Workshop on DNA-Based Computers, DNA7, Tampa, FL, USA, June 10-13, 2001. Revised Papers (N. Jonoska, N. C. Seeman, Eds.), 2340, Springer Verlag, Berlin, Heidelberg, New York, 2002.

[78] M.Margenstern, Yu.Rogozhin, S.Verlan: *Time-Varying Distributed H Systems of Degree 2 Can Carry Out Parallel Computations,* DNA Computing: 8th International Workshop on DNA-Based Computers, DNA8, Sapporo, Japan, June 10-13, 2002. Revised Papers (M. Hagiya, A. Ohuchi, Eds.), 2568 (2002), pp. 326–336.

[79] M.Margenstern, Y.Rogozhin, S.Verlan: *Time-varying distributed H systems with parallel computations: the problem is solved*, DNA Computing: 9th International Workshop on DNA Based Computers, DNA9, Madison, WI, USA, June 1-3, 2003. Revised Papers (J. Chen, J. Reif, Eds.), 2943, Springer, 2004.

[80] M.Margenstern, Gh.Păun, Yu.Rogozhin, S.Verlan: *Context-free insertion-deletion systems.* Theoretical Computer Science, Elsevier, vol.330, issue 2 (2005), pp. 339–348.

[81] A.Matveevici, Yu.Rogozhin, S.Verlan: *Insertion-Deletion Systems with One-Sided Contexts.* Lecture Notes in Computer Science, Springer, vol. 4664 (2007), pp. 205–217.

[82] M.Minsky: *Computations: Finite and Infinite Machines,* Prentice Hall, Englewood Cliffts, NJ, 1967.

[83] V.Mitrana, I.Petre, V.Rogojin: *Accepting splicing systems.* Theoretical Computer Science, Elsevier, 411, 25, 2010, pp. 2414–2422.

[84] K.Morita, Ts.Ogiro, A.Alhazov, Ts.Tanizawa: *Non-degenerate 2-State Reversible Logic Elements with Three or More Symbols Are All Universal.* In: Proceedings of 2nd Workshop on Reversible Computation, July 2nd - 3rd, 2010, Bremen, Germany, pp. 27–34.

[85] Ts. Ogiro, A. Alhazov, Ts. Tanizawa, K. Morita: *Universality of 2-State 3-Symbol Reversible Logic Elements - A Direct Simulation Method of a Rotary Element.* In: Proceedings of the 4th International Workshop on Natural Computing, Himeji, 2009, pp. 220–227.

[86] Ts. Ogiro, A. Alhazov, Ts. Tanizawa, K. Morita: *Universality of 2-State 3-Symbol Reversible Logic Elements - A Direct Simulation Method of a Rotary Element.* Natural Computing, PICT 2, Springer Japan, part 3, 2010, pp. 252–259.

[87] *The Oxford Handbook of Membrane Computing,* ed. by Gh.Păun, G.Rozenberg, A.Salomaa. Oxford University Press, 2010.

[88] A.Păun: *On Time-Varying H Systems,* Bulletin of EATCS, 67, February 1999, pp. 157–164.

[89] Gh.Păun, G.Rozenberg, A.Salomaa: *Computing by splicing*, TCS 168, pp. 321–336, 1996.

[90] Gh.Păun: *DNA computing: distributed splicing systems, Structures in Logic and Computer Science.* A Selection of Essays in Honor of A. Ehrenfeucht (J. Mycielsky, G. Rozenberg, A. Salomaa, Eds.), 1261, Springer Verlag, Berlin, Heidelberg, New York, 1997.

[91] Gh.Păun, G.Rozenberg, A.Salomaa: *DNA Computing: New Computing Paradigms,* Springer Verlag, Berlin, Heidelberg, New York, September 1998, ISBN 3-540-64196-3.

[92] Gh.Păun: *DNA computing based on splicing: universality results,* Proceedings of the Second Internetional Colloquium on Universal Machines and Computations, Metz, France (M. Margenstern, Ed.), I, IUT de Metz, 1998.

[93] Gh.Păun, G.Rozenberg, A.Salomaa, Eds.:*Handbook of Membrane Computing.* Oxford University Press, 2010.

[94] Gh.Păun: Membrane Computing. An Introduction, Springer, 2002.

[95] $http://www.p-lingua.org/wiki/index.php/Main_Page$

[96] L.Priese, Yu.Rogozhin, M.Margenstern: *Finite H-systems with 3 Test Tubes are not Predictable.* In: Proceedings of Pacific Simposium on Biocomputing, 3, Kapalua, Maui, January 1998, Hawaii, USA (R.Altman, A.Dunker, L.Hanter, T.Klein eds.), World Sci.Publ., Singapure (1998), pp. 545–556.

[97] Yu.Rogozhin, S.Verlan: *On the Rule Complexity of Universal Tissue P Systems.* LNCS, vol. 3850, pp. 356–362, Springer (2006).

[98] Yu.Rogozhin, S.Verlan: *New choice for small universal devices: Symport/Antiport P systems.* EPTCS 1, 2009, pp. 235-242, doi:10.4204/EPTCS.1.23.

[99] G.Rozenberg, A.Salomaa, Eds.:*Handbook of Formal Languages*, vol. 1–3, Springer, 1997.

[100] *P systems webpage.* `http://ppage.psystems.eu/`

[101] *TVDHsim: Time-Varying Distributed H Systems Simulator.* http://lita.sciences.univ-metz.fr/ verlan/

[102] S.Verlan: *Calculs Moleculaires: les Syst'emes Distributes 'a Changement de Phase,* Master Thesis, Universite de Metz, 2001.

[103] S.Verlan: *On Enhanced Time-Varying Distributed H Systems,* Computer Science Journal of Moldova, 10(3), 2002, pp. 263–279, Kishinev.

[104] S.Verlan: *A Frontier Result on Enhanced Time-Varying Distributed H Systems with Parallel Computations,* Preproceedings of DCFS'03, Descriptional Complexity of Formal Systems, Budapest, Hungary, July 12-14, 2003, 2003.

[105] S.Verlan: *Communicating Distributed H Systems with Alternating Filters,* in: Aspects of Molecular Computing. Essays Dedicated to Tom Head on the Occasion of His 70th Birthday (N. Jonoska, Gh. Păun, G. Rozenberg, Eds.), vol. 2950 of LNCS, Springer Verlag, Berlin, Heidelberg, New York, 2004, pp. 367–384.

[106] S.Verlan: *Head Systems and Applications to Bio-Informatics,* Ph.D. Thesis, University of Metz, 2004.

[107] S. Verlan: *On minimal context-free insertion-deletion systems.* In C. Mereghetti, B. Palano, G. Pighizzini, and D. Wotschke, editors, *Seventh International Workshop on Descriptional Complexity of Formal Systems, June 30 - July 2, 2005 Como, Italy. Proceedings.*, 285-292, 2005. Technical repport no. 06-05, University of Milan. In publication in *Journal of Automata Languages and Combinatorics.*

[108] S.Verlan: *Look-Ahead Evolution for P Systems.* In: Proc. of the 10th Workshop on Membrane Computing, WMC10, Curtea de Arges (Romania), August 24 - 27, 2009, pp. 507–513.

[109] S.Verlan, P.Frisco: *Splicing P Systems.* In: The Oxford Handbook of Membrane Computing, ed. by Gh.Păun, G.Rozenberg, A.Salomaa, Oxford University Press, 2010.

[110] S.Verlan, Y.Rogozhin: *New choice for small universal devices: Symport/antiport P systems.* In: International Workshop on The Complexity of Simple Programs, University College Park, Ireland, December 6th and 7th, 2008, pp. 305–314.

A. Alhazov[1,2], E. Boian[1], L. Burtseva[1],
C. Ciubotaru[1], S. Cojocaru[1], A. Colesnicov[1],
V. Demidova[1], S. Ivanov[1,3], V. Macari[1],
G. Magariu[1], L. Malahova[1], V. Rogojin[1,4],
Yu. Rogozhin[1], T. Tofan[1], S. Verlan[1,5], T. Verlan[1]

Received November 1, 2010

[1] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
5 Academiei str., Chişinău, MD-2028, Moldova

[2] FCS, Department of Information Engineering
Graduate School of Engineering, Hiroshima University
Higashi-Hiroshima 739-8527 Japan

[3] Technical University of Moldova, Faculty of Computers,
Informatics and Microelectronics,
Ştefan cel Mare 168, Chişinău MD-2004 Moldova

[4] Biomedicum Helsinki,
B524a P.O.Box 63 (Haartmaninkatu 8) 00014
UNIVERSITY OF HELSINKI

[5] LACL, Departement Informatique
UFR Sciences et Technologie
Universite Paris XII
61, av. General de Gaulle
94010 Creteil, France

E–mails:
Dr. Artiom Alhazov: artiom@math.md,
Dr. Elena Boian: lena@math.md,
Dr. Liudmila Burtseva: burtseva@math.md,
Dr. Constantin Ciubotaru: chebotar@math.md,
Dr.hab. Svetlana Cojocaru: Svetlana.Cojocaru@math.md,
Dr. Alexandru Colesnicov: kae@math.md,
Valentina Demidova: demidova@math.md,
Sergiu Ivanov: sivanov@math.md,
Veaceslav Macari: vmacari@yandex.ru,
Dr. Galina Magariu: gmagariu@math.md,
Ludmila Malahova: mal@math.md,
Dr. Vladimir Rogojin: vladimir.rogojin@helsinki.fi,
Dr.hab. Yurii Rogozhin: rogozhin@math.md,
Tatiana Tofan: ttofan@math.md,
Dr.hab. Sergey Verlan: verlan@univ-paris12.fr,
Tatiana Verlan: tverlan@math.md