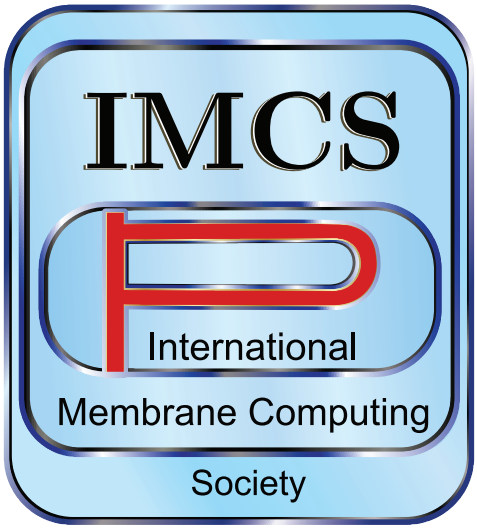
B U L L E T I N

of the International Membrane Computing Society

I M C S



Number 2 December 2016

Bulletin Webpage:

http://membranecomputing.net/IMCSBulletin

Webmaster: Andrei Florea, andrei91ro@gmail.com

Foreword

This is the second issue of our *Bulletin*, completing the first year of the existence of the International Membrane Computing Society.

We hear often the "wise statement" that it is easy to start, not to easy to go on... There are big chances that our project, IMCS and its *Bulletin*, is one further evidence which disproves this statement.

We also know that "even the journey of one thousand miles begins with one step" (Lao Tzu). In what concerns the *Bulletin*, this is already the second step...

This second issue of the *Bulletin* is significantly larger, but it has a similar structure as the first issue, the June 2016 one. A tool for connecting the MC community, a channel through which a broad range of information become available to the many groups working in this research area, from Europe to China, from Canada to New Zealand.

In my opinion, one of the items included deserves a special mentioning: the list of PhD theses completely or only partially dealing with topics in MC. There are more that 85 titles, and this is really a large number. Encouraging is also the evolution in time of the number of theses, which proves the robustness of MC, due especially to the increased number of applications, in a wider range of areas.

It is important to stress that the contents of the *IMCS Bulletin* is conceived as a working material, the *Bulletin* is a "blackboard" where each member of the MC community (and any *future member*, too) can write a *message* to the other members of the community, with this "blackboard" also visible to the entire community and also outside the community: each issue gradually grows and remains available at `http://membranecomputing.net/IMCSBulletin`), also being printed. (If somebody wants to have a printed copy, s/he has to contact the IMCS secretary – see information about the structure of IMCS, including email addresses, in the subsequent pages.)

The "instructions to contributors" are minimal. Any material which any MC researcher considers of interest for the community, helping in achieving the goals of IMCS, is very much welcome and can be submitted at any time to the bulletin editor (myself for a while) or to any member of the Bulletin Committee. In what

concerns the style and format, the previous issues of the *Bulletin* are available as a model. (If an author needs more precise instructions, please contact me. Standard Latex files are sufficient, LNCS style is the best.)

*

The realization of this issue of the *Bulletin of IMCS* owns very much (i) to all contributors, (ii) to the webmaster, Andrei Florea, andrei91ro@gmail.com, and to the MC research group in Politechnica University in Bucharest, led by prof. Cătălin Buiu, where the bulletin is hosted, and (iii) to prof. Gexiang Zhang, the President of IMCS, to his group, and to the Xihua University in Chengdu, China, where the bulletin is printed.

Gheorghe Păun
December 10, 2016

Contents

Letter from The President

Dear IMCS members,

Merry Christmas and Happy New Year to you all!

The first issue of our *Bulletin* has achieved a great success since it was published on the website in June. Tens of hard copies have also been distributed to each research group in the P system community during the 17^{th} International Conference on Membrane Computing (CMC 2016), which was held in Milano, Italy, and The 5^{th} Asian Conference on Membrane Computing (ACMC 2016), which was held at the Universiti Kebangsaan Malaysia (The National University of Malaysia), Bangi, Selangor, Malaysia. The two conferences were rather successful, organized this year under the auspices and with the support of International Membrane Computing Society (IMCS).

First of all, I would like to share the great news that our Honorary President, prof. Gheorghe Păun, received the Order of the Star of Romania (Romania's highest civil Order) on December 1, 2016, which was awarded by the President of Romania in recognition of his life achievements in science. Let us congratulate him on his ground breaking scientific contributions. He also received the title of Honorary Professor of Xihua University, Chengdu, China, in November.

One of our aims is to strengthen the collaborations within the P system community. We are trying our best to accomplish this. Here I just list three examples. ACMC 2016 received an increased number of submissions, 39, this year. Prof. Gheorghe Păun visited China in October and November. During his visit, he delivered a series of lectures on Automata and Formal Languages, consisting of six parts: Chomsky hierarchy, Regulated rewriting, Grammar systems, Lindenmayer (L) systems, Marcus contextual grammars and Splicing (Head, H) systems. More than twenty teachers and students attended the lectures. Another example is that Sergey Verlan and myself started to jointly supervise a PhD student from this

September. This student can obtain two PhD degrees, from Southwest Jiaotong University and University Paris–EST, respectively.

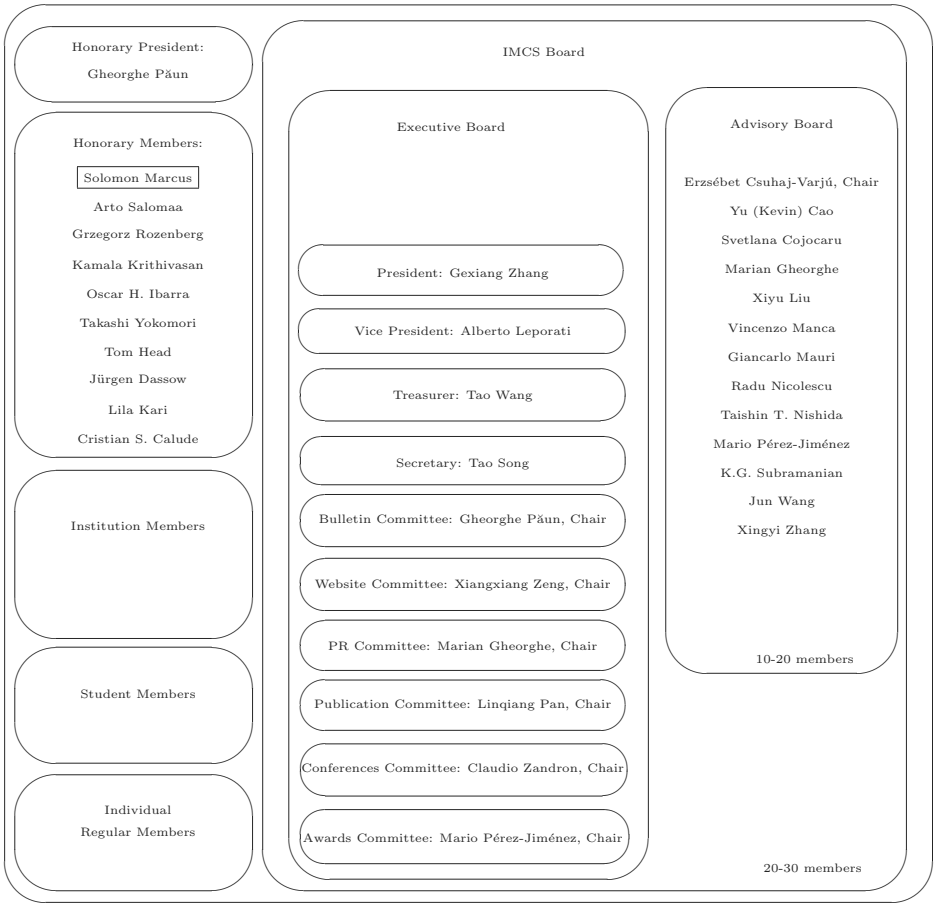
Finally, an encouragement is addressed to all members to submit applications for the three annual IMCS Prizes: (1) The PhD Thesis of the Year, (2) The Theoretical Result of the Year, (3) The Application of the Year.

This issue will appear soon. I would like to take this opportunity to express my deep gratitude to the Chair, prof. Gheorghe Păun, and his committee members, for their efficient and hard work.

Gexiang Zhang
Chengu, China
December 20, 2016

IMCS Matters

Structure of IMCS



The Board of IMCS

The Executive Board:

President: Gexiang Zhang, China, gexiangzhang@gmail.com
Vice President: Alberto Leporati, Italy, alberto.leporati@unimib.it
Treasurer: Tao Wang, China, wangtao2005@163.com
Secretary: Tao Song, China, taosong@hust.edu.cn, songtao0608@hotmail.com
Bulletin Committee Chair: Gheorghe Păun, Romania, gpaun@us.es
Website Committee Chair: Xiangxiang Zeng, China, xzeng@xmu.edu.cn
PR Committee Chair: Marian Gheorghe, U.K. m.gheorghe@bradford.ac.uk
Publication Committee Chair: Linqiang Pan, China, lqpanhust@gmail.com
Conferences Committee Chair: Claudio Zandron, Italy, zandron@disco.unimib.it
Awards Committee Chair: Mario Pérez-Jiménez, Spain, marper@us.es

The Advisory Board:

Erszébet Csuhaj-Varjú, Hungary – Chair, csuhaj@inf.elte.hu
Yu (Kevin) Cao, U.S.A.
Svetlana Cojocaru, Rep. Moldova
Marian Gheorghe, U.K.
Xiyu Liu, China
Vincenzo Manca, Italy
Giancarlo Mauri, Italy
Radu Nicolescu, New Zealand
Taishin T. Nishida, Japan
Mario Pérez-Jiménez, Spain
K.G. Subramanian, India
Jun Wang, China
Xingyi Zhang, China

Honorary President:

Gheorghe Păun, Romania

Honorary Members:

Solomon Marcus, Romania
Arto Salomaa, Finland
Grzegorz Rozenberg, The Netherlands
Kamala Krithivasan, India
Oscar H. Ibarra, U.S.A.
Takashi Yokomori, Japan
Tom Head, U.S.A.
Jürgen Dassow, Germany
Lila Kari, Canada
Cristian S. Calude, New Zealand

Bulletin Committee

Website Committee

PR Committee

Marian Gheorghe, Bradford, U.K. – Chair, M.Gheorghe@bradford.ac.uk
Petros Kefalas, Sheffield, U.K. (International Faculty, Greece), kefalas@city.academic.gr
Savas Konur, Bradford, U.K., S.Konur@bradford.ac.uk
Maciej Koutny, Newcastle, U.K., maciej.koutny@newcastle.ac.uk
Jianhua Xiao, Nankai, China, jhxiao@nankai.edu.cn

Publication Committee

Linqiang Pan, Wuhan, China – Chair, lqpanhust@gmail.com
Marian Gheorghe, Bradford, U.K., m.gheorghe@bradford.ac.uk
Alberto Leporati, Milan, Italy, alberto.leporati@unimib.it
Gheorghe Păun, Bucharest, Romania, gpaun@us.es
Mario Pérez-Jiménez, Seville, Spain, marper@us.es
Gexiang Zhang, Chengdu, China, zhgxdylan@126.com

The main tasks of the Publication Committee are (1) to explore the possibility to initiate a series of MC monographs/collective volumes, (2) to establish a MC international journal, (3) to advise the organizers of CMC, ACMC, BWMC, MC workshops in what concerns the special issues of journals, (4) to help translating MC books in Chinese.

The Publication Committee can become the Editorial Board of the MC series of books, but, of course, the journal should have a much larger Editorial Board.

Conferences Committee

Claudio Zandron, Milan, Italy – Chair, zandron@disco.unimib.it
Henry Adorna, Quezon City, Philippines
Artiom Alhazov, Chişinău, Rep. of Moldova
Bogdan Aman, Iaşi, Romania
Matteo Cavaliere, Edinburgh, Scotland
Erzsébet Csuhaj-Varjú, Budapest, Hungary
Rudolf Freund, Wien, Austria
Marian Gheorghe, Bradford, U.K. – Honorary Member
Thomas Hinze, Cottbus, Germany
Florentin Ipate, Bucharest, Romania
Shankara N. Krishna, Bombay, India
Alberto Leporati, Milan, Italy
Taishin Y. Nishida, Toyama, Japan

Linqiang Pan, Wuhan, China – responsible of ACMC
Gheorghe Păun, Bucharest, Romania – Honorary Member
Mario J. Pérez-Jiménez, Sevilla, Spain
Agustín Riscos-Núñez, Sevilla, Spain
Petr Sosík, Opava, Czech Republic
K.G. Subramanian, Chennai, India
György Vaszil, Debrecen, Hungary
Sergey Verlan, Paris, France
Gexiang Zhang, Chengdu, China

Awards Committee:

Mario Pérez-Jiménez, Seville, Spain – Chair, marper@us.es
Marian Gheorghe, Bradford, U.K., m.gheorghe@bradford.ac.uk
Giancarlo Mauri, Milan, Italy, mauri@disco.unimib.it
Gheorghe Păun, Bucharest, Romania, gpaun@us.es
Linqiang Pan, Wuhan, China, lqpanhust@gmail.com

Rules of functioning:

1. Prizes to be awarded annually: (1) The PhD Thesis of the Year, (2) The Theoretical Result of the Year, (3) The Application of the Year.
2. Each prize consists of diplomas for each co-author, one copy of the Hamangia thinker[1] and one voucher for a discount in the registration fee for the first of BWMC, CMC or ACMC to take place after the prize was voted; the discount will be fixed by the organizing committee of the meeting; in case of several authors, they will choose the one of them to benefit of the voucher.
3. Any registered member of IMCS can be nominated and can receive any of the three prizes. In cases (2) and (3), the prizes are awarded to the authors of a paper or of an application, with at least one of authors being a member of IMCS. The members of the Awards Committee cannot receive any prize, neither they can be coauthors of papers or applications which receive one of prizes (2) and (3).
4. If the Awards Committee considers necessary, each year at most one of the prizes can be awarded ex aequo, to two winners.
5. Any registered member of IMCS can propose a candidate for any prize, by sending to any member of the Awards Committee the relevant information (and any additional information requested by the Awards Committee). Implicitly, the Awards Committee can itself make nominations.

[1] A Neolithic age clay sculpture, about 4000 years BC, found in Romania – see the image at the next page.

6. The nominations for the year Y should reach the Awards Chair before 20 of January of the year Y+1. The Awards Committee members decide the winners by the middle of February, and the prizes are awarded at the first edition of BWMC, CMC or ACMC where the winners participate in.
7. The members of the Awards Committee and the rules of functioning can be changed every year, after March 1, at the proposal of the Chair person or of any member of the IMCS Board, subject to a vote in the IMCS Board.

The IMCS prizes are mainly meant to reward the excellency in MC research, equally focusing on theory and applications, and to encourage young researchers.

The prizes are not subject to competitions, they do not identify *the best* PhD thesis or paper or application, they just point that a certain work/result is of a high value. This does not imply that other works/results are not so. We cannot rank scientific results like in sport, in a mathematical sense.

We only want to call attention to certain works – thus also calling attention to MC and to IMCS.

The prestige of a prize will be given by the prestige of the winners, also on their evolution in time, during their careers.

To reach these goals, we have to be conservative, exigent, transparent in our nominations and, especially, in selection.

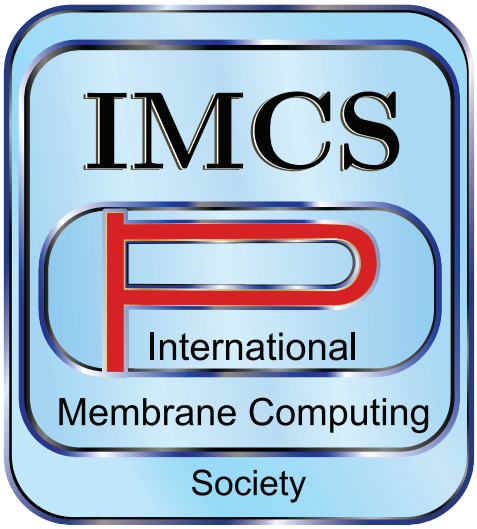
Nominations for the prizes for 2016 are waited for until January 20, 2017, and can be sent at any time, electronically, to any member of the Awards Committee (preferably with a CC to all members).

Constitution of the International Membrane Computing Society – IMCS

Article 1: Name

(1.1.) The name of the Society shall be International Membrane Computing Society, abbreviated IMCS.

(1.2.) The logo of IMCS is the one in the figure below. It should appear in all relevant places, such as IMCS web page, posters, calls, on the cover of the *Bulletin of IMCS*, etc.



Article 2: Objects

(2.1.) The society shall be a nonprofit academic organization, having as its goal to promote the development of membrane computing (MC), internationally, at all levels (theory, applications, software, implementations, connection with related areas, etc.).

(2.2.) A special attention will be paid to the communication/cooperation inside MC community, to connections with other professional scientific organizations with similar aims, and to promoting MC to young researchers.
(2.3.) IMCS will publish proceedings, journals or other materials, printed or electronically, as it sees fit.
(2.4.) IMCS will organize yearly MC meetings, such as the **Conference on Membrane Computing** (CMC), the **Asian Conference on Membrane Computing** (ACMC), the **Brainstorming Week on Membrane Computing** (BWMC), as well as further workshops/meetings, as it sees fit.

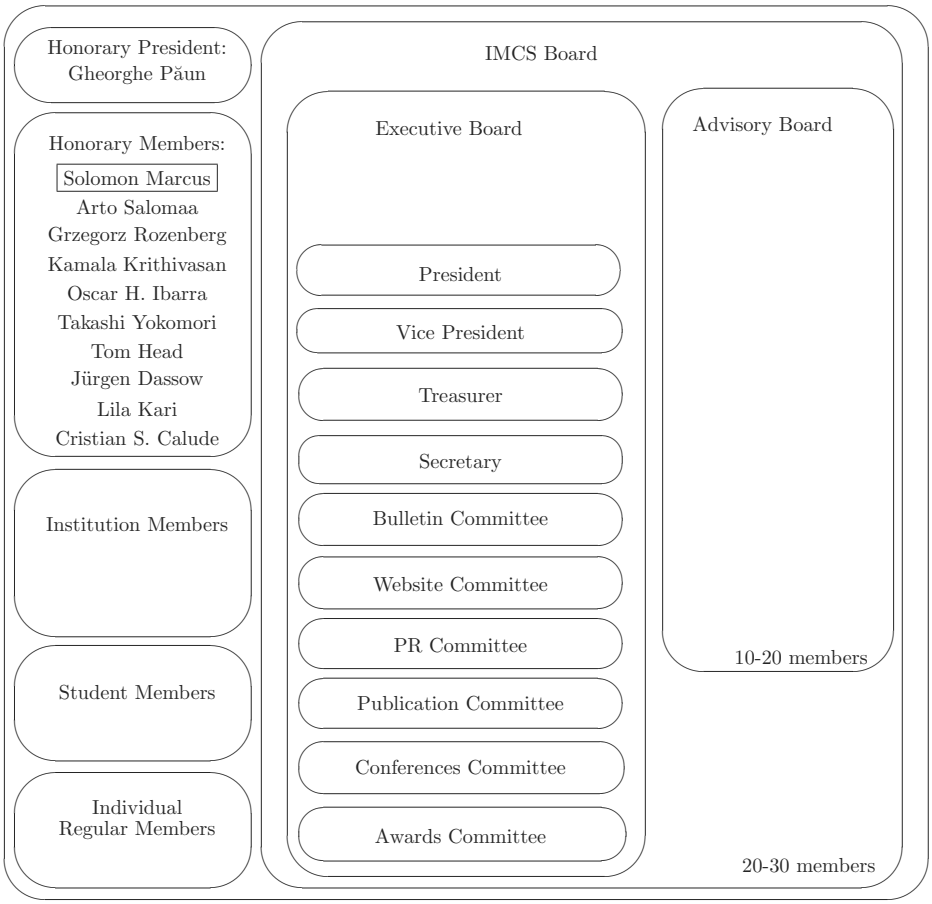
Article 3: Membership

(3.1.) There are four categories of members: **Honorary Members, Regular Members, Student Members**, and **Institution Members**.
(3.2.) The Honorary Members are elected by the IMCS Board (email voting, majority rule). Regular membership is open to all persons interested, on completing a membership form.
(3.3.) Student Members can be undergraduate, master, and PhD students, and they are eligible for various facilities IMCS is planning for students.
(3.4.) Institutions which want to join IMCS and support it can become Institution Members. Any support/sponsorship from an institution will be acknowledged in an appropriate way in IMCS publications.
(3.5.) Any member, of any kind, is supposed to know and accept the Constitution of IMCS.

Article 4: Structure

(4.1.) The structure of IMCS and its governance are as specified in the next figure. The figure also specifies the ten Honorary Members with whom the Society starts (February 2016).
(4.2.) Gheorghe Păun, the founder of MC, is appointed **Honorary President** of IMCS.
(4.3.) The work of IMCS is organized and conducted by the **Board of IMCS**, consisting of the **Executive Board** and the **Advisory Board**. The Advisory Board should have between 10 and 20 members, hence in total the IMCS Board should contain between 20 and 30 members,
(4.4.) The Executive Board consists of four individual positions: **President, Vice President, Treasurer**, and **Secretary**, and six **Committees: Bulletin, Website, PR, Publication, Conferences**, and **Awards Committee**. Each of these six Committees has a chair person. The Advisory Board also has a chair person.
(4.5.) The four individual positions from the Executive Board, the six chair persons of the Committees, the members of the Advisory Board, and the chair of the Advisory Board are elected by the IMCS Board (email voting, majority rule). Each chair of a Committee appoints a number of Committee members as he/she sees fit.

(4.6.) All the elected positions are elected for one year. After one year, a change of an elected person can be proposed by the President or the Vice President of the IMCS Board, by the person itself (resignation), or by two thirds of members of the IMCS Board, and it is voted in the IMCS Board (email voting, majority rule). If there is no change proposal, then the person who occupies any position in the IMCS Board continues in the same position, for one further year.



Article 5: Duties and competencies

(5.1.) The IMCS Board President represents the Society in relation with any external entity, organizes/coordinates the activity of the IMCS Board, initiates voting in the IMCS Board, chairs any panel/meeting of the Society.

(5.2.) The Vice President helps the President in all his/her activity, represents the President when he/she is not available (e.g., in chairing panels/meetings). Every year, the President and the Vice President present a common report

about IMCS activity, first circulating it by email in the IMCS Board and, after possible corrections, posting it in the IMCS web page.

(5.3.) The Treasurer takes care of the income and expenses of IMCS, and each year presents a report in this respect to IMCS Board. This report is analyzed and voted in the IMCS Board (email voting, majority rule).

(5.4.) The Secretary is responsible to keep a track record of the IMCS: memberships, reports, voting results, etc.

(5.5.) The Bulletin Committee takes care of editing the *Bulletin of IMCS*, first accumulating information/materials in an electronic format and then printed, if needed/requested, with a periodicity to be decided by the IMCS Board.

(5.6.) The Website Committee takes care of the IMCS web page, whose structure should be decided by the IMCS Board.

(5.7.) The PR Committee is responsible with developing relationships with other similar organizations and promoting IMCS on various scientific forums, advertising its activity on specialised networks and at international events.

(5.8.) The Publication Committee supervises the publication of proceedings, special issues of journals, collective volumes edited under the auspices of IMCS. Two particular goals of this Committee are to initiate a specialized journal, *International Membrane Computing Journal*, and a specialized series of monographs.

(5.9.) The Conference Committee works as a steering committee for the two MC yearly conferences, CMC and ACMC, looking for venues, suggesting (in cooperation with the organizing committees) program committees and invited speakers, possible sponsors and publications.

(5.10.) The Awards Committee collects nominations and decides the winners of three yearly IMCS Prizes: **(1) The PhD Thesis of the Year, (2) The Theoretical Result of the Year, (3) The Application of the Year**. The Awards Committee has its Rules of functioning, which are voted by the IMCS Board (email voting, majority rule).

Article 6: Voting

(6.1.) Each member of the IMCS Board (between 20 and 30 members) has one vote.

(6.2.) A voting, on any subject, can be initiated by the President, the Vice President, or by two thirds of the IMCS Board members.

(6.3.) The message proposing a vote should specify the issue to be decided in such a way that the alternatives YES and NO are clear. The message should be sent to all members of the IMCS Board, the voting messages of the members should also be sent to all members (full transparency). The voting should last 30 days. If a member is not replying in the first 15 days, the initiator of the vote should contact him/her once again. If a member is not replying even to the second message, then his/her vote is considered *abstaining*.

(6.4.) "Majority rule" means that at least half of the IMCS Board have voted (YES, NO, or abstaining) and the decision is made according to the number of

YES and NO votes which is higher. In case of a draw, the vote of the President is decisive – unless if the President does not decide to repeat the vote, maybe changing the object of the vote.

(6.5.) All ambiguities and uncovered cases should be clarified by discussions in the IMCS Board and, if decided so, proposed as amendments to the Constitution.

Article 7: Panels

(7.1.) On the occasions of IMCS annual meetings, like BWMC, CMC, and ACMC, panels should be organized, chaired by the President, the Vice President or, in their absence, by another member of the IMCS Board designated by the President, to discuss current issues of the Society.

Article 8: Finance

(8.1.) Income: possible membership fees, as decided by IMCS Board, donations, sponsorhips, conference registration fees, participation to research projects.

(8.2.) Expenses: IMCS prizes, students support, *Bulletin of IMCS* hardcopy, maintaining web pages, sponsoring MC conferences – all these and anything else, under the control of the IMCS Board.

Article 9: Amendments

(9.1.) Amendments to IMCS Constitution can be proposed by any member of the IMCS Board, at any time. Any amendment should be discussed and voted in the IMCS Board (email voting, majority rule) and then, if accepted, published in the IMCS web page, thus being available to all members of IMCS.

Article 10: Dissolution

(10.1.) The dissolution of IMCS should be done in two steps: first, a vote in the IMCS Board is organized (email voting, two thirds majority), and, if the dissolution proposal passes, a general vote is organized, where all regular members participate (email voting, two thirds majority; in order the vote to be valid, at least half of the members should vote).

(10.2.) If the Society decides to get dissolved, all remaining assets shall be donated to a similar organization, at the choice of the IMCS Board.

Article 11. Provisory statement

The present Constitution will get provisionally valid by being voted (by email, majority rule), in March 2016, in the IMCS Board, as this Board was constituted by consensus during BWMC 2016 and soon after that. Then, it will be published in the IMCS web page and, as soon as possible, in 2016, it will be voted by all individual members of IMCS (email voting, majority rule). The vote will last one month and to voting will participate all individual members of IMCS, students or regular, registered until the last day of the previous month.

News from MC Research Groups

The Research Group on Modeling, Simulation and Verification of Biological Systems at the University of Pisa

1 Contacts

Dipartimento di Informatica
Università di Pisa
Largo B. Pontecorvo, 3
56127 Pisa, Italy

Group webpage: `http://www.di.unipi.it/msvbio/`

2 Reserch activities

The activity of the research group on modeling, simulation and verification of biological systems started in 2004, with the aim of developing formal notations and analysis techniques for biochemical systems. The class of notations we developed includes term rewriting systems, process calculi and automata-based formalisms. As analysis techniques we studied and applied stochastic simulation, model checking and abstract interpretation.

Subsequently, we considered other classes of biological systems and phenomena, such as cellular pathways (in the context of systems biology), cancer development, and also systems of interest for population biology, ecology and evolution.

Moreover, we conducted research on bio-inspired models of computation (in particular Membrane Systems and Reaction Systems). In particular, we worked on compositional semantics and formal analysis techniques for such bio-inspired models, and on variants of membrane systems for the modeling of populations and ecosystems.

In parallel, we aimed at defining static analyses able to predict all possible evolutions of biological systems modeled in biologically inspired calculi. Such analyses

address relevant properties such as temporal and causal properties and probabilistic termination in stochastic calculi.

Reserch topics

In order of affinity with Membrane Computing:

- Theory of membrane systems [8, 9, 10, 18]
- Formal semantics of membrane systems [11, 12]
- Spatial P systems [7, 6]
- Membrane systems for population and ecosystems modelling [2, 1, 3]
- Reaction Systems [4, 5, 19]
- Modelling formalisms for biochemical systems [13]
- Computational models of cancer [22]
- Cellular pathway components identification [20, 21]
- Analyses of temporal and causal properties for biologically inspired calculi [16, 15, 14]
- Analysis of probabilistic termination in stochastic calculi [17].

Conferences

The group regularly contributes to the organization of the International Symposium "From Data to Models and Back (DataMod)", formerly known as MoKMaSD. DataMod aims at bringing together researchers interested in the combined application of computational modeling methods with data-driven techniques from the areas of knowledge management, data mining and machine learning. Modeling methodologies of interest include automata, agents, Petri nets, process algebras and rewriting systems. Application domains include social systems, ecology, biology, medicine, smart cities, governance, education, software engineering, and any other field that deals with complex systems and large amounts of data.

DataMod (as it was for MoKMaSD) welcomes contributions in which notations and methodologies from membrane computing are exploited for the modeling and analysis of systems in any application domain. Contributions in which such methodologies are applied in synergy with with data-driven techniques would be of particular interest.

The web page of the symposium is: `http://pages.di.unipi.it/datamod/`

3 Components

- Roberto Barbuti (full professor)
  `http://pages.di.unipi.it/barbuti/`
- Francesca Levi (associate professor)
  `http://pages.di.unipi.it/levi/`

- Roberta Gori (assistant professor)
  `http://pages.di.unipi.it/gori/`
- Paolo Milazzo (assistant professor)
  `http://pages.di.unipi.it/milazzo/`
- Giovanni Pardini (postdoc)
  `http://pages.di.unipi.it/pardini/`
- Pasquale Bove (phd student)
- Giovanna Broccia (phd student)

Selected publications

1. R. Barbuti, A. Bompadre, P. Bove, P. Milazzo, and G. Pardini. Attributed probabilistic p systems and their application to the modelling of social interactions in primates. In *International Conference on Software Engineering and Formal Methods*, pages 176–191. Springer, 2015.
2. R. Barbuti, P. Bove, P. Milazzo, and G. Pardini. Minimal probabilistic p systems for modelling ecological systems. *Theoretical Computer Science*, 608:36–56, 2015.
3. R. Barbuti, A. Cerone, A. Maggiolo-Schettini, P. Milazzo, and S. Setiawan. Modelling population dynamics using grid systems. In *International Conference on Software Engineering and Formal Methods*, pages 172–189. Springer, 2012.
4. R. Barbuti, R. Gori, F. Levi, and P. Milazzo. Investigating dynamic causalities in reaction systems. *Theoretical Computer Science*, 623:114–145, 2016.
5. R. Barbuti, R. Gori, F. Levi, and P. Milazzo. Specialized predictor for reaction systems with context properties. *Fundamenta Informaticae*, 147:1–19, 2016.
6. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and G. Pardini. Simulation of spatial p system models. *Theoretical Computer Science*, 529:11–45, 2014.
7. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, G. Pardini, and L. Tesei. Spatial p systems. *Natural Computing*, 10(1):3–16, 2011.
8. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, G. Pardini, and S. Tini. Systolic automata and p systems. In *Computing with New Resources*, pages 17–31. Springer, 2014.
9. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and L. Tesei. Timed p automata. *Fundamenta Informaticae*, 94(1):1–19, 2009.
10. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and S. Tini. Ap systems flat form preserving step-by-step behaviour. *Fundamenta Informaticae*, 87(1):1, 2008.
11. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and S. Tini. Compositional semantics and behavioral equivalences for p systems. *Theoretical Computer Science*, 395(1):77–100, 2008.
12. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and S. Tini. An overview on operational semantics in membrane computing. *International Journal of Foundations of Computer Science*, 22(01):119–131, 2011.
13. R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and A. Troina. The calculus of looping sequences for modeling biological membranes. In *International Workshop on Membrane Computing*, pages 54–76. Springer, 2007.
14. C. Bodei, L. Brodo, R. Gori, D. Hermith, and F. Levi. A global occurrence counting analysis for brane calculi. In *Logic-Based Program Synthesis and Transformation - 25th International Symposium, LOPSTR 2015, Siena, Italy, July 13-15, 2015. Revised Selected Papers*, pages 179–200, 2015.

15. C. Bodei, R. Gori, and F. Levi. Causal static analysis for brane calculi. *Theor. Comput. Sci.*, 587:73–103, 2015.
16. R. Gori and F. Levi. Abstract interpretation based verification of temporal properties for bioambients. *Inf. Comput.*, 208(8):869–921, 2010.
17. R. Gori and F. Levi. An analysis for proving probabilistic termination of biological systems. *Theor. Comput. Sci.*, 471:27–73, 2013.
18. P. Milazzo. Membrane computing: from biology to computation and back. *Isonomia*, 2014.
19. G. Pardini, R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and S. Tini. A compositional semantics of reaction systems with restriction. In *Conference on Computability in Europe*, pages 330–339. Springer, 2013.
20. G. Pardini, P. Milazzo, and A. Maggiolo-Schettini. Identification of components in biochemical pathways: extensive application to sbml models. *Natural Computing*, 13(3):351–365, 2014.
21. G. Pardini, P. Milazzo, and A. Maggiolo-Schettini. Component identification in biochemical pathways. *Theoretical Computer Science*, 587:104–124, 2015.
22. S. Sameen, R. Barbuti, P. Milazzo, A. Cerone, M. Del Re, and R. Danesi. Mathematical modeling of drug resistance due to kras mutation in colorectal cancer. *Journal of theoretical biology*, 389:263–273, 2016.

Research Group on Bio-inspired Computing at Huazhong University of Science and Technology in China

Contact:
Linqiang Pan
School of Automation
Huazhong University of Science and Technology
Wuhan 430074, Hubei, China
lqpan@mail.hust.edu.cn

1 Team Introduction

The Research Group on Bio-inspired Computing at School of Automation (http://english.auto.hust.edu.cn), Huazhong University of Science and Technology (http://english.hust.edu.cn), led by professor Linqiang Pan, focuses on bio-inspired computing such as membrane computing and DNA computing, and related nanotechnology for implementing bio-inspired computing. The group consists of professor Linqiang Pan, associate professors Xiaolong Shi (shixiaolong@hust.edu.cn), Zhihua Chen (chenzhihua@mail.hust.edu.cn), and Zheng Zhang (leaf@mail.hust.edu.cn), lecturer Xueming Liu (xmliu1988@gmail.com). Researchers are always welcome to join our group as post doctors, visiting researchers, or even faculty members.

Now, there are two post doctors in the group: Bosheng Song (boshengsong@hust.edu.cn) working on membrane computing, and Fei Xu (fei_xu@hust.edu.cn) working on DNA nanotechnology.

The group has five PhD students:

1. Zhiqiang Zhang (zhiqiangzhang@hust.edu.cn) working on the computation power of numerical P systems.
2. Cheng He (chenghe@hust.edu.cn) working on multi-objective evolutionary computation.
3. Tingfang Wu (tfwu@hust.edu.cn) working on spiking neural P systems.
4. Yingxin Hu (yingxinhu@hust.edu.cn) working on DNA nanotechnology.

5. Zhiyu Wang (m201372231@hust.edu.cn) working on DNA nanotechnology.

There are twelve people who got PhD from the group:

1. Xiangxiang Zeng (xzeng@xmu.edu.cn) working in Xiamen University; the title of PhD thesis: Research on computational property of spiking neural P systems.
2. Xingyi Zhang (xyzhanghust@gmail.com) working in Anhui University; the title of PhD thesis: Research on the computational power of spiking neural P systems.
3. Yun Jiang (purpleye83@hotmail.com) working in Chongqing Technology and Business University; the title of PhD thesis: Research on the computational power of P systems with network structure.
4. Yunyun Niu (niuyunyun1003@163.com) working in China University of Geosciences; the title of PhD thesis: Research on models and algorithms of membrane computing for solving computational intractable problems.
5. Tao Song (songtao0608@hotmail.com) working in China University of Petroleum; the title of PhD thesis: Research on computational properties and applications of spiking neural P systems.
6. Juanjuan He (hejuanjuan@wust.edu.cn) working in Wuhan University of Science and Technology; the title of PhD thesis: Research on membrane-inspired algorithms and applications.
7. Yansen Su (suyansen1985@163.com) working in Anhui University; the title of PhD thesis: Research the difference of subtypes of non-small cell lung cancer based on molecular level.
8. Keqin Jiang (jiangkq@aqtc.edu.cn) working in Anqing Normal University; the title of PhD thesis: Research on sequential spiking neural P systems based on spike number.
9. Chun Lu (luchun_et@163.com) working in Central China Normal University; the title of PhD thesis: Research on computational properties of membrane systems with proteins.
10. Yuan Kong (kongyuan@scau.edu.cn) working in South China Agricultural University; the title of PhD thesis: Research on the computational power of spiking neural P systems with differential biological backgrounds.
11. Bosheng Song (boshengsong@hust.edu.cn) working in Huazhong University of Science and Technology; the title of PhD thesis: Research on computational property of timed membrane systems.
12. Xueming Liu (xm_liu@hust.edu.cn) working in Huazhong University of Science and Technology; the title of PhD thesis: Research on the robustness and controllability of complex networks.

2 Research interests

Our research interest covers membrane computing, DNA computing, and related nanotechnology. Their short descriptions are as follows:

- **Membrane computing.** Inspired from the structure and the functioning of living cells, some variants of P systems have been proposed, their computation power and applications for modeling biological systems have been investigated. Spiking neural P systems are of a particular interest for us.
- **DNA computing an related nanotechnology.** The group studies the structure and functional properties of DNA molecules, designs self-assembly model and carries out the experimental study of molecular self-assembly. DNA structures as programmable building blocks are used to create self-assembling molecular materials acting as drug delivery vehicle for cancer therapy. DNA nanostructures are designed to implement parallel computational models and algorithms.

3 Selected papers in the last 5 years

The following is a list of selected papers on membrane computing, DNA computing and related nanotechnology published by the group members in the last 5 years.

1. L. Pan, J. Wang, H.J. Hoogeboom, Spiking neural P systems with astrocytes, *Neural Computation*, 24(3) (2012), 805–825.
2. X. Zeng, T. Song, X. Zhang, L. Pan, Performing four basic arithmetic operations with spiking neural P systems, *IEEE Transactions on Nanobioscience*, 11(4) (2012), 366–374.
3. K. Jiang, T. Song, W. Chen, L. Pan, Homogeneous spiking neural P systems working in sequential mode induced by maximum spike number, *International Journal of Computer Mathematics*, 90(4) (2013), 831–844.
4. T. Song, L. Pan, Gh. Păun, Asynchronous spiking neural P systems with local synchronization, *Information Sciences*, 219 (2013), 197–207.
5. L. Valencia-Cabrera, M. Garcia-Quismondo, M.J. Pérez-Jiménez, Y. Su, H. Yu, L. Pan, Modeling logic gene networks by means of probabilistic dynamic P systems. *International Journal of Unconventional Computing*, 9(5-6) (2013), 445–464.
6. X. Shi, W. Lu, Z. Wang, L. Pan, G. Cui, J. Xu, T.H. LaBean, Programmable DNA tile self-assembly using a hierarchical sub-tile strategy, *Nanotechnology*, 25(7) (2014), 075602.
7. T. Song, L.F. Macías-Ramos, L. Pan, M.J. Pérez-Jiménez, Time-free solution to sat problem using P systems with active membranes, *Theoretical Computer Science*, 529 (2014), 61–68.
8. T. Song, L. Pan, Gh. Păun, Spiking neural P systems with rules on synapses, *Theoretical Computer Science*, 529 (2014), 82–95.
9. X. Zeng, X. Zhang, T. Song, L. Pan, Spiking neural P systems with thresholds, *Neural Computation*, 26 (2014), 1340–1361.
10. J. Yang, C. Dong, Y. Dong, S. Liu, L. Pan, C. Zhang, Logic nanoparticle beacon triggered by the binding-induced effect of multiple inputs, *ACS Applied Materials & Interfaces*, 6(16) (2014), 14486–14492.

11. X. Zhang, L. Pan, A. Păun, On the universality of axon P systems, *IEEE Transactions on Neural Networks and Learning Systems*, 26(11) (2015), 2816–2829.
12. X. Shi, C. Chen, X. Li, T. Song, Z. Chen, Z. Zhang, Y. Wang, Size-controllable DNA nanoribbons assembled from three types of reusable brick single-strand DNA tiles, *Soft Matter*, 11(43) (2015), 8484–8492.
13. L. F. Macías-Ramos, B. Song, L. Valencia-Cabrera, L. Pan, M.J. Pérez-jiménez, Membrane fission: A computational complexity perspective, *Complexity*, 21(6) (2016), 321–334.
14. L. Pan, Gh. Păun, B. Song, Flat maximal parallelism in P systems with promoters, *Theoretical Computer Science*, 623 (2016), 83–91.
15. T. Wu, Z. Zhang, Gh. Păun, L. Pan, Cell-like spiking neural P systems, *Theoretical Computer Science*, 623 (2016), 180–189.
16. B. Song, L. Pan, M. J. Pérez-Jiménez, Tissue P systems with protein on cells, *Fundamenta Informaticae*, 144 (2016), 77–107.
17. Z. Zhang, T. Wu, A. Păun, L. Pan, Numerical P systems with migrating variables, *Theoretical Computer Science*, 641 (2016), 85–108.
18. B. Song, L. Pan, M. J. Pérez-Jiménez, Cell-like P systems with channel states and symport/antiport rules, *IEEE Transactions on NanoBioscience*, 15(6) (2016), 555–566.
19. X. Shi, X. Wu, T. Song, X. Li, Construction of DNA nanotubes with controllable diameters and patterns using hierarchical DNA sub-tiles, *Nanoscale*, 8(31) (2016), 14785–14792.
20. B. Song, C. Zhang, L. Pan, Tissue-like P systems with evolutional symport/antiport rules, *Information Sciences*, 378 (2017), 177–193.

RGMCA: Research Group on Membrane Computing and Applications at Xihua University, Chengdu

Hong Peng, Jun Wang

Xihua University
Chengdu 610031,P.R.China
ph.xhu@hotmail.com
wj.xhu@hotmail.com

1 Team Introduction

The research group on membrane computing and applications (RGMCA) at Xihua University is composed of professors, research engineers, technical staff and students from the School of Electrical and Information Engineering and the School of Computer and Software Engineering, Xihua University, Chengdu, Sichuan Province, China. They bring together computer science, engineering, mathematics and other disciplines to develop new P systems models and applications of them.

We also collaborate with the research group of natural computing (RGNC), University of Seville, Seville, Spain (Mario J. Pérez-Jiménez, Agustín Riscos-Núñez), the research group of nature-inspired computation and smart grid Lab., the Southwest Jiaotong University, Chengdu, China (Gexiang Zhang).

Some main members are introduced as follows.

Prof. Hong Peng (ph.xhu@hotmail.com) received his Ph.D. degree in Signal and Information Processing from the University of Electronic Science and Technology of China (UESTC), Chengdu. He is currently a professor in the School of Computer and Software Engineering, Xihua University, China, since 2005. His research interests include membrane computing, machine learning, cloud computing and big data analysis, pattern recognition and image processing, signal processing.

Prof. Jun Wang (wj.xhu@hotmail.com) received her Ph.D. degree in Electrical Engineering from the Southwest Jiaotong University, Chengdu, China. Since 2003, she is currently a professor with the School of Electrical and Information Engineering, Xihua University. Her research interests include membrane computing, electrical automation, and intelligent control.

Dr. Tao Wang (736678662@qq.com) received her Master Degree at Xihua University in 2011. Then, she continued to study in the Southwest Jiaotong University in 2016 and received her PhD degree.

Her research work includes fuzzy spiking neural P systems and fault diagnosis of power systems.

Dr. Xiaoxiao Song (88119843@qq.com) graduated and received his PhD from Chongqing University. He is interested in optimization algorithm research, membrane computing and its application in industry control system.

Dr. Zhang Sun (383623076@qq.com) graduated and received his Ms Degree from Xihua University. Currently, he continues the research in order to complete his PhD. His specialty is hardware design and realization of membrane computing models through FPGA.

Dr. Bing Gou (bingo57350@126.com) graduated received his PhD in 2016 from the Southwest Jiaotong University. In this year, he joined our team and tries to focus on membrane computing model and its application to fault diagnosis of electrical motors control.

Ke Cheng is an M.S. degree student (2511198132@qq.com, Chengdu)
Ming Jun is an M.S. degree student (mingjun.jun@qq.com, Chengdu)
Wenping Yu is an M.S. degree student (2315430785@qq.com, Chengdu)
Guangchun Chen is an M.S. degree student (2303153523@qq.com, Chengdu)
Juan Hu is an M.S. degree student (2421383883@qq.com, Chengdu)

2 Research Interests and Results

(1) **Fuzzy spiking neural P systems and their applications in fault diagnosis**: Fuzzy spiking neural P systems (FSN P systems, in short) are extensions of SN P systems which incorporate fuzzy logic techniques into the framework of SN P systems. The motivation was to make SN P systems a computing framework to model or express fuzzy production rules and to perform fuzzy reasoning. Two kinds of neurons are considered in FSN P systems: proposition neurons and rule neurons. Proposition neurons are used to express fuzzy propositions, while rule neurons are used to describe fuzzy production rules in a knowledge base. Moreover, fuzzy logic is integrated into both proposition and rule neurons.

In recent years, fuzzy logics have been introduced to develop five kinds of FSN P systems: fuzzy spiking neural P systems with linguistic terms (lFSN P systems) [1], fuzzy reasoning spiking neural P systems (FRSN P systems) [2], weighted fuzzy spiking neural P systems (WFSN P systems) [3], adaptive fuzzy spiking neural P systems (AFSN P systems) [4] and fuzzy reasoning spiking neural P systems with trapezoidal fuzzy numbers (tFRSN P systems) [5].

FRSN P systems were firstly used to deal with fault diagnosis problem, where an instance of fault diagnosis of transformers was modeled and analyzed by FRSN P systems [2]. After that the fault diagnosis of power systems has been investigated, in [6, 7, 8], respectively.

(2) **Membrane computing-inspired data clustering**: Membrane clustering algorithms are a class of clustering algorithms which are inspired form the

mechanism of membrane computing models. The main idea is that a data clustering problem can be regarded as an optimization problem. So, the clustering problem can be solved by the use of P systems.
H. Peng, J. Wang and their collaborators have discussed a membrane computing-inspired clustering algorithm [9], in which a differential evolution mechanism is introduced as evolution rules to evolve a set of cluster centers. Then P systems were used to discuss different clustering problems, such as data clustering [10, 11, 12, 13, 14], fuzzy clustering [15, 16] and automatic clustering problems [17, 18]. In addition, P systems were used to induce a decision tree algorithm [19].

(3) **Application of membrane computing in image processing/signal processing**: In recent years our team used membrane computing model to solve a variety of image processing/signal processing problems. Five papers on the topic of application of membrane computing model in image and signal processing have been also published by H. Peng, J. Wang and their collaborators (region-based image segmentation [20, 21], clustering-based image segmentation [22], image thresholding [23, 24, 25, 26], image fusion [27], watermarking [28], and digital filter design [29, 30]).

(4) **Application of membrane computing in micro-grid**: Improved particle swarm optimization, genetic algorithms and artificial fish swarm algorithms based on membrane computing are applied on the operation of distributed power, especially in economic operation of micro-grid and control strategy of micro-grid environmental protection optimization [31, 32, 33, 34, 35]. Ref. [36, 37, 38, 39, 40] applied different improved P systems in different control systems.

3 Publications and Teaching

Our team members have published 40 papers (see references) and received five Chinese patents. There are now more than 8 graduate students working for their Master degree.

References

1. J. Wang, L. Zhou, H. Peng, G. Zhang: An extended spiking neural P system for fuzzy knowledge representation. *Int. J. Innov. Comput. Inform. Control.*, 7(7), 3709-3724 (2011)
2. H. Peng, J. Wang, M.J. Pérez-Jiménez, H. Wang, J. Shao, T. Wang: Fuzzy reasoning spiking neural P system for fault diagnosis. *Information Sciences*, 235, 106-116 (2013)
3. J. Wang, P. Shi, H. Peng, M.J. Pérez-Jiménez, T. Wang: Weighted fuzzy spiking neural P systems. *IEEE Transaction on Fuzzy Systems*, 21(2), 209-220 (2013)
4. J. Wang, H. Peng: Adaptive fuzzy spiking neural P systems for fuzzy inference and learning. *Int. J. Comput. Math.*, 90(4), 857-868 (2013)

5. T. Wang, J. Wang, H. Peng, H. Wang: Knowledge representation and reasoning based on FRSN P system. *In: Proc. WCICA, IEEE*, 849-854 (2011)
6. M. Tu, J. Wang, H. Peng, P. Shi: Application of adaptive fuzzy spiking neural P systems in fault diagnosis of power systems. *Chinese Journal of Electronics*, 23(1), 87-92 (2014)
7. J. Wang, H. Peng, M. Tu, M.J. Pérez-Jiménez, P. Shi: A fault diagnosis method of power systems based on an improved adaptive fuzzy spiking neural P systems and PSO algorithms. *Chinese Journal of Electronics*, 25(2), 320-327 (2016)
8. T. Wang, G. Zhang, J. Zhao, Z., He, J. Wang, M.J. Pérez-Jiménez: Diagnosis of electric power systems based on fuzzy reasoning spiking neural P systems. *IEEE Transaction on Power Systems*, 30(3), 1182-1194 (2015)
9. H. Peng, J. Zhang, Y. Jiang, X. Huang, J. Wang: DE-MC: A membrane clustering algorithm based on differential evolution mechanism. *Romanian Journal of Information Science and Technilogy*, 17(1), 76-88 (2014)
10. L. Qin, F. Cheng, Z. Chen, X. Huang, H. Peng, J. Liu: A hybrid clustering algorithm based on P systems and Immune mechanisms, *ICIC Express Letters*, 9(2), 485-491 (2015)
11. H. Peng, X. Luo, Z. Gao, J. Wang, Z. Pei: A novel clustering algorithm inspired by membrane computing, *The Scientific World Journal*, Vol. 2015, Article ID 929471, 1-9 (2015)
12. J. Jin, H. Liu, F. Wang, H. Peng, J. Wang: Parallel Implementation of P Systems for Data Clustering on GPU. *Bio-Inspired Computing - Theories and Applications, Volume 562 of the series Communications in Computer and Information Science*, vol.562, 200-211 (2015)
13. X. Huang, H. Peng, Y. Jiang, J. Zhang, J. Wang: PSO-MC: a novel PSO-based membrane clustering algorithm, *ICIC Express Letters*, 8(2), 497-503 (2014)
14. Y. Jiang, H. Peng, X. Huang, J. Zhang, P. Shi: A novel clustering algorithm based on P systems, *International Journal of Innovative Computing, Information and Control*, 10(2), 753-765 (2014)
15. H. Peng, J. Wang, M.J. Pérez-Jiménez, A. Riscos-Núñez: An unsupervised learning algorithm for membrane computing. *Information Sciences*, 304, 80-91 (2015)
16. H. Peng, Y. Jiang, J. Wang, M.J. Pérez-Jiménez: Membrane clustering algorithm with hybrid evolutionary mechanisms. *Journal of Software*, 26(5), 1001-1012 (2015)
17. H. Peng, J. Wang, P. Shi, A. Riscos-Núñez, M.J. Pérez-Jiménez: An automatic clustering algorithm inspired by membrane computing, *Pattern Recognition Letters*, 68, 34-40 (2015)
18. H. Peng, J. Wang, P. Shi, M.J. Pérez-Jiménez, A. Riscos-Núñez: An extended membrane system with active membrane to solve automatic fuzzy clustering problems. *International Journal of Neural Systems*, 26(3), 1-17 (2016)
19. J. Wang, J. Hu, H. Peng, M.J. Pérez-Jiménez, A. Riscos-Núñez: Decision tree models induced by membrane systems. *Romanian Journal of Information Science and Technology*, 18(3), 228-239 (2015)
20. Y. Yang, H. Peng, Y. Jiang, X. Huang, J. Zhang: A region-based image segmentation method under P systems, *Journal of Information and Computational Science*, 10(10), 2943-2950 (2013)
21. H. Peng, Y. Yang, Y. Jiang, X. Huang, J. Wang: A region-based color image segmentation method under P systems. *Romanian Journal of Information Science and Technology*, 17(1), 63-75 (2014)

22. J. Zhang, H. Peng, X. Huang, X. Gao: An image pixel clustering algorithm based on P systems, *ICIC Express Letters, Part B: Applications*, 6(7), 1813-1819 (2015)
23. Z. Zhang, H. Peng: Object segmentation with membrane computing, *Journal of Information and Computational Science*, 9(17), 5417-5424 (2012)
24. H. Wang, H. Peng, J. Shao: A thresholding method based on P systems for image segmentation, *ICIC Express Letters*, 6(1), 221-227 (2012)
25. H. Peng, J. Wang, M.J. Pérez-Jiménez, P. Shi: A novel image thresholding method based on membrane computing and fuzzy entropy, *Journal of Intelligent and Fuzzy Systems*, 24(2), 229-237 (2013)
26. H. Peng, J. Wang, M.J. Pérez-Jiménez: Optimal multilevel thresholding with membrane computing. *Digital Signal Processing*, 37, 53-64 (2015)
27. Z. Zhang, X. Yi, H. Peng: A novel framework of tissue membrane systems for image fusion. *Bio-Medical Materials and Engineering*, 24(6), 3259-3266 (2014)
28. H. Peng, J. Wang, M.J. Pérez-Jiménez: The framework of P systems applied to solve optimal watermarking problem. *Signal Processing*, 101, 256-265 (2014)
29. J. Wang, P. Shi, H. Peng: Membrane computing model for IIR filter design. *Information Sciences*, 329, 164-176 (2016)
30. H. Peng, J. Wang: A hybrid approach based on tissue P systems and artificial bee colony for IIR system identification, *Neural Computing and Applications*, 2016. DOI: 10.1007/s00521-016-2201-3.
31. T. Liu, J. Wang, Z. Sun: An improved particle swarm optimization and its application for micro-grid economic operation optimization, *Communications in Computer and Information Science*, 472, 276-280 (2014)
32. Z. Sun, T. Liu, J. Wang, J. Liu, H. Li: An improved particle swarm optimization based on adaptive mutation and P systems for micro-grid economic operation, *Proceedings of the 2015 Chinese Intelligent Automation Conference, Lecture Notes in Electrical Engineering*, 336, 505-512 (2015)
33. J. Luo, J. Wang, P. Shi, H. Luo, Z. Sun, T. Liu: Micro-grid economic operation using genetic algorithm based on P systems, *ICIC Express Letters*, 9(2), 609-617 (2015)
34. M. Tu, J. Wang, X. Song: An artificial fish swarm algorithm based on P systems, *ICIC Express Letters, Part B: Aplications*, 4(3), 747-753 (2013)
35. M. Tu, J. Wang, F. Yang, J. Lu: Application of artificial fish swarm algorithm based on P systems in Economic load dispatch of power systems, *ICIC Express Letters*, 8(2), 409-414 (2014)
36. T. Wang, J. Wang, H. Peng: Optimization of PID controller parameters based on PSOPS algorithm, *ICIC Express Letters*, 6(1), 273-280 (2011)
37. J. Wang, T. Wang, P. Shi: Membrane optimization algorithm based on mutated PSO and its application in nonlinear control systems, *International Journal of Innovative Computing, Information and Control*, 9(7), 2963-2977 (2013)
38. K. Chen, J. Wang, Z. Sun: Programmable logic stage programming using spiking neural P systems, *Journal of Computational and Theoretical Nanoscience*, 12(7), 1292-1299 (2015)
39. K. Chen, J. Wang, Z. Sun, L. Juan, T. Liu: The application of spiking neural P systems for PLC programming, *Communications in Computer and Information Science*, 472, 11-15 (2014)
40. Y. Yang, J. Ming, J. Wang, H. Peng, Z. Sun, and Wenping Yu: The implementation of membrane clustering algorithm based on FPGA, *ACMC2016*, 2016.

The Spanish Thematic Network on Biomolecular and Biocellular Computing

José M. Sempere

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia, Spain
jsempere@dsic.upv.es

Summary. We describe the structure and main activities of the Spanish Thematic Network on Biomolecular and Biocellular Computing (Redbiocom).

1 The network and its research teams

The Spanish Thematic Network on Biomolecular and Biocellular Computing, Redbiocom (http://www.redbiocom.es) was established in the year 2008, and, since then, it has been supported by different Spanish governments through different research supporting programs. It was first coordinated by Prof. Dr. Mario Pérez-Jiménez (Univ. of Sevilla), and, currently, it is coordinated by Dr. José M. Sempere (Univ. Politécnica de Valencia).

Initially, the network was composed by the following Spanish research groups:

- The Research Group of the Universidad Autónoma de Madrid (UAM)
- The Research Group on Computational Biology and Bioinformatics of the Universitat de les Illes Balears (UIB)
- The Research Group of the Universitat de Lleida (ULL)
- The Natural Computing Research Group of the Universidad de Sevilla (US)
- The Research Group on Natural Computing of the Universidad Politécnica de Madrid (UPM-I)
- The Research Group of the Artificial Intelligence Laboratory of the Universidad Politécnica de Madrid (UPM-II)
- The Research Group on Computational Models and Formal Languages of the Universidad Politécnica de Valencia (UPV)

Nowadays, the network has been slightly changed by removing the UAM team and by splitting the UPM-I group into two different groups. In the following, we overview the main activities of the current research groups.

The Research Group on Computational Biology and Bioinformatics at the Universitat de les Illes Balears (UIB)

Group Coordinator: Francesc Rosselló (email: frossello@mac.com)

The research group works in computational biology and bioinformatics. The main topics in which the group is currently working can be enumerated as follows:

1. Structural Bioinformatics
2. Philogenetic Trees and Networks
3. Biomolecular Networks Modeling
4. Biomolecular Processes Modeling

In the last two topics the research group has used and explored membrane computing and computational chemistry as computational tools. The Research Group organized the Brainstorming Workshop on Uncertainty in Membrane Computing in 2004 in Palma de Mallorca.

The Research Group at the Universitat de Lleida (ULL)

Group Coordinator: M. Angels Colomer (email: colomer@matematica.udl.cat)

The main research topics of the group are the following:

1. Ecosystems modeling through Population Dynamic P-systems.
2. Discrete Markov chains research through computational DNA and P systems.
3. Stochastic Modeling of rural problems.

The Research Group on Natural Computing at the Universidad de Sevilla (US)

Group Coordinator: Agustín Riscos-Núñez (email: ariscosn@us.es)

The Research Group on Natural Computing at the Universidad de Sevilla (RGNC) explores the interconnections between Computer Science, Engineering, Mathematics and Biology.

The main research lines of the group are the following:

1. The development of specification languages for complex dynamic systems, specially in multicompartment systems for biological systems (from bacterial colonies to ecosystems).
2. The development of algorithms by formal semantics to simulate and integrate different space-time scales that appear in biological systems.
3. The implementation with high-performance computing of the previous algorithms in a framework of software development.
4. The design of models for systems and synthetic biology in the framework of cellular computing.

5. The research on the theoretical aspects of non-conventional computing, specially with respect to its computational power and efficiency. The main achievements is the formulation of new characterizations of the conjecture $\mathbf{P} \neq \mathbf{NP}$.
6. Applications to various engineering areas, including engineering optimization, power system fault diagnosis, or the study of other complex systems involving data modeling and process interactions.

The RGNC is one of the most referred groups in Membrane Computing. They have organized the Brainstorming Week on Membrane Computing since the year 2004. Two of the members of the group serve in the Program Committee and the Steering Committee of the International Conference on Membrane Computing (CMC). Additionally, Prof. Dr. Mario Pérez-Jiménez (formerly, the RGNC coordinator) is the Awards Committee Chair of the IMCS, and Agustín Riscos-Núñez is member of the IMCS Bulletin Committee. In 2014, the RGNC obtained the NVIDIA CUDA Research Center mention for the Universidad de Sevilla, and it was renewed in 2015.

The Research Group on Natural Computing at the Universidad Politécnica de Madrid (UPM-I)

Group Coordinator: Juan Castellanos (email: `jcastellanos@fi.upm.es`)

The research group on Natural Computing at UPM, focuses its research activities in the following areas:

1. Genetic Algorithms and Evolutionary Computing.
2. Artificial Neural Networks.
3. Molecular Computing (DNA based computing).
4. Cellular Computing and Membrane Computing.

The research group belongs to the European Molecular Computing Consortium (EMCC) and they organize the Natural Information Technologies conference (NIT) that is usually celebrated every year in Madrid. Some members of the group are members of the editorial committee of the *International Journal Information, Theories and Applications* (IJITA).

The Research Group on Mathematical Models and Biocomputing Algorithms at the Universidad Politécnica de Madrid (UPM-II)

Group Coordinator: Fernando Arroyo (email: `farroyo@eui.upm.es`)

This research group was created in 2014, and its members were formerly members of the UPM-I group. The main research topics of the group are the same as the ones of the UPM-I group, together with a strong interest in Formal Languages and Networks of Bio-inspired Processors.

Research Group of the Artificial Intelligence Laboratory at the Universidad Politécnica de Madrid (UPM-III)

Group Coordinator: Alfonso Rodríguez-Patón (email: arpaton@fi.upm.es)

The main research activities of this group are focused in biomolecular computing (DNA based computing) and bacterial computing. Additionally, the group works in membrane computing (P systems), specially in Spiking Neural P systems, and in Systems Biology.

The Research Group on Computation Models and Formal Languages at the Universidad Politécnica de Valencia (UPV)

Group Coordinador: José M. Sempere (email: jsempere@dsic.upv.es)

The research group at UPV works in theoretical aspects of Biomolecular and Membrane Computing (P systems), classical Automata Theory and Formal Languages, Grammatical Inference, and other non-conventional models of computation (such as Networks of Bio-inspired Processors). Some of the theoretical results have been applied to solve practical problems in biosequence processing, such as functional and structural protein prediction, and simulation of biological processes such as the recently developed ARES system to simulate the antibiotic resistance evolution that has been fully designed under the P systems paradigm.

The members of the research group have organized some international events such as the ECAI 2004 Workshop on Symbolic Networks, the 10th International Colloquium on Grammatical Inference (ICGI 2010) and, more recently, the 16th Conference on Membrane Computing (CMC16) in 2015. One of its member, José M. Sempere is a member of the Steering Committee of the International Conference on Membrane Computing (CMC) and member of the *IMCS Bulletin* Committee.

2 Network activities

The main activities of the network are oriented to promote the research activities of its research groups. In addition, it is a valuable instrument to establish collaborations between the groups to propose coordinated research projects in the national research programs and/or international projects with other foreign research groups.

One of the main events that the network has organized is the International School on Biomolecular and Biocellular Computing (http://www.redbiocom.es/ISBBC.html). The first edition of ISBBC was held in Osuna (Sevilla) in 2011, while the second edition was organized in Madrid in 2013 co-located with the NIT 2013 Conference. Currently, the arrangements for the third edition have started and it will be held in June 2017, in Valencia. The school is a three days event for graduated and doctoral students where they can attend talks given by the most prominent researchers of the network together with international invited speakers.

The talks are organized in the areas of Membrane Computing, Systems and Synthetic Biology, Networks of Bio-inspired Processors and Bioinformatics. All the students can apply for grants that cover all the expenses related to attending the school.

Another goal of the network is to serve as a gateway for external collaborations between foreign research groups and the network groups. In this sense, any collaborative project that can involve more than one of the research groups of the network can be addressed to the network.

Any question, proposal or additional information regarding the network can be addressed to José M. Sempere (`jsempere@dsic.upv.es`).

Formal Methods Laboratory (FML) in Iaşi

Romanian Academy, Institute of Computer Science
Blvd. Carol I no.8, 700505 Iaşi, Romania

The Formal Methods Laboratory (http://www.iit.tuiasi.ro/∼fml/) is part of the Institute of Computer Science, Romanian Academy (Iaşi branch). Currently, the team consists of 3 members:

- Gabriel Ciobanu, gabriel@info.uaic.ro
- Bogdan Aman, bogdan.aman@iit.academiaromana-is.ro
- Alexandru Andrei, andrei.alexandru@iit.academiaromana-is.ro.

There are several other collaborators:

- Ross Horne - former member of the team, currently at Nanyang Technological University, Singapore
- Dănuţ Rusu - Associated Professor at A.I. Cuza University of Iaşi
- Cristian Văideanu - Assistant Professor at A.I. Cuza University of Iaşi
- Eneia Todoran - Professor, Technical University of Cluj-Napoca, Romania
- Maciej Koutny - Professor, School of Computing Science, Newcastle University, UK.

Gabriel Ciobanu is a PhD supervisor in Romania from 2001, involved also as evaluator of several PhD theses in Singapore, Germany, France and Italy. Gabriel was a mentor for several former students: Mihai Rotaru (Reuters), Bogdan Tănasă (San Diego), Dorin Paraschiv (Oracle), Daniel Dumitriu (Mainz), Dorin Huzum, Gabriel Moruz (Frankfurt), Sinică Alboaie, Sabin Buraga, Cristian Prisacariu (Oslo), Laura Cornăcel, Călin Juravle (Google Londra), Andreas Resios (Utrecht), Armand Rotaru (UCL Londra), Oana Agrigoroaiei (Stuttgart), Cosmin Bonchiş (UVT Timişoara) and Cornel Izbaşa (Munchen), as well as Rahul Desai, Akash Kumar, K.N. Sridhar, Janardan Mishra and Guo Wenyuan at National University of Singapore.

Research Interests

Since 2002, the group has focused on fundamental research in computer science and mathematics. The research has resulted in theoretical methods, algorithms and

mathematical models. Many papers have been successfully presented at national and international conferences.
The group currently has a major focus on many areas in computer science:

- Membrane Computing (Natural Computing)
- Process Calculi and Distributed Systems
- Logics and Type Systems
- Sets and Multisets

The group has made significant contributions to the area of Membrane Computing, being involved in many publications: books, journal papers, book chapters, conference communications, etc. Many members were actively participating in the main conferences related to natural computing, continuously along the years. Since 2005 the FML has been involved in research projects focusing on Membrane Computing as the main topic. The group has organized 5 of the 6 editions of the Workshop on Membrane Computing and Biologically Inspired Process Calculi (MeCBIC). The second MeCBIC was held in Iaşi in 2008, the third one took place in Bologna (CONCUR 2009), the forth one in Jena (Germany), then in Paris and finally in Newcastle (CONCUR 2012). The proceedings of the MeCBIC workshops have been published as ENTCS volumes 171 and 227 (2006, 2008), EPTCS volumes 11 and 40 (2009, 2010), in arXiv.org (http://arxiv.org/abs/1108.3558) in 2011, and as *Theoretical Computer Science* volume 431 in 2012.

Selected Publications (Chronologically)

1. Andrei Alexandru, Gabriel Ciobanu: *Finitely Supported Mathematics. An Introduction.* Springer 2016, ISBN 978-3-319-42281-7
2. Bogdan Aman, Péter Battyányi, Gabriel Ciobanu, György Vaszil: Simulating P systems with membrane dissolution in a chemical calculus. *Natural Computing* **15**(4): 521-532 (2016)
3. Bogdan Aman, Gabriel Ciobanu: Efficiently solving the bin packing problem through bio-inspired mobility. *Acta Informatica* (2016)
4. Bogdan Aman, Gabriel Ciobanu: Modelling and verification of weighted spiking neural systems. *Theor. Comput. Sci.* **623**: 92-102 (2016)
5. Bogdan Aman, Gabriel Ciobanu: Behavioural observations of cell movements with timing aspects. *Nano Comm. Netw.* **6**(3): 96-102 (2015)
6. Bogdan Aman, Gabriel Ciobanu: Verification of membrane systems with delays via Petri nets with delays. *Theor. Comput. Sci.* **598**: 87-101 (2015)
7. Andrei Alexandru, Gabriel Ciobanu: Generalized multisets: from ZF to FSM. *Computing and Informatics* **34**(5): 1133-1150 (2015)
8. Oana Agrigoroaiei, Gabriel Ciobanu: Rewriting Systems Over Indexed Multisets. *Comput. J.* **57**(1): 165-179 (2014)
9. Gabriel Ciobanu, Dragoş Sburlan: Monitoring changes in dynamic multiset systems. *Fundam. Inform.* **134**(1-2): 67-82 (2014)
10. Gabriel Ciobanu, G. Michele Pinna, Dragoş Sburlan: Power of causal dependencies in rule-based systems. *Journal of Automata, Languages and Combinatorics* **19**(1-4): 45-56 (2014)

11. Gabriel Ciobanu, Dragoş Sburlan: Scenario Based P Systems. *IJUC* **9**(5-6): 351-366 (2013)
12. Bogdan Aman, Gabriel Ciobanu: Properties of enhanced mobile membranes via coloured Petri nets. *Inf. Process. Lett.* **112**(6): 243-248 (2012)
13. Gabriel Ciobanu, Maciej Koutny: *Modelling and Analysis of Biological Systems*, special issue *Theor. Comput. Sci.* **431** (2012)
14. Bogdan Aman, Gabriel Ciobanu: *Mobility in Process Calculi and Natural Computing.* Natural Computing Series, Springer 2011, ISBN 978-3-642-24866-5, pp. 1-194
15. Bogdan Aman, Gabriel Ciobanu: Solving a weak NP-complete problem in polynomial time by using mutual mobile membrane systems. *Acta Inf.* **48**(7-8): 409-415 (2011)
16. Gabriel Ciobanu, Shankara Narayanan Krishna: Enhanced mobile membranes: computability results. *Theory Comput. Syst.* **48**(3): 715-729 (2011)
17. Bogdan Aman, Gabriel Ciobanu: Mutual mobile membranes with objects on surface. *Natural Computing* **10**(2): 777-793 (2011)
18. Oana Agrigoroaiei, Gabriel Ciobanu: Reversing computation in membrane systems. *J. Log. Algebr. Program.* **79**(3-5): 278-288 (2010)
19. Oana Agrigoroaiei, Gabriel Ciobanu: Rule-based and object-based event structures for membrane systems.*J. Log. Algebr. Program.* **79**(6): 295-303 (2010)
20. Bogdan Aman, Gabriel Ciobanu: Computational Aspects of Mobile Membranes, Brane Calculi and Mobile Ambients. *Scholarpedia* **5**(7): 9420 (2010)
21. Gabriel Ciobanu, Viorel Mihai Gontineac: Encodings of multisets. *Int. J. Found. Comput. Sci.* **20**(3): 381-393 (2009)
22. Gabriel Ciobanu, Andreas Resios: Complexity of evolution in maximum cooperative P systems. *Natural Computing* **8**(4): 807-816 (2009)
23. Bogdan Aman, Gabriel Ciobanu: Simple, Enhanced and Mutual Mobile Membranes. *Trans. Computational Systems Biology* **11**: 26-44 (2009)
24. Gabriel Ciobanu, Bogdan Aman: On the relationship between membranes and ambients. *Biosystems* **91**(3): 515-530 (2008)
25. Bogdan Aman, Gabriel Ciobanu: Turing completeness using three mobile membranes. *Lecture Notes in Computer Science* **5715**: 42-55 (2009)
26. Cosmin Bonchiş, Cornel Izbaşa, Gabriel Ciobanu: Information theory over multisets. *Computing and Informatics* **27**(3+): 441-451 (2008)
27. Gabriel Ciobanu, Andreas Resios: Computational complexity of simple P systems. *Fundam. Inform.* **87**(1): 49-59 (2008)
28. Oana Andrei, Gabriel Ciobanu, Dorel Lucanu: A rewriting logic framework for operational semantics of membrane systems. *Theor. Comput. Sci.* **373**(3): 163-181 (2007)
29. Gabriel Ciobanu, Linqiang Pan, Gheorghe Păun, Mario J. Pérez-Jiménez: P systems with minimal parallelism. *Theor. Comput. Sci.* **378**(1): 117-130 (2007)
30. Daniela Zaharie, Gabriel Ciobanu: Distributed evolutionary algorithms inspired by membranes in solving continuous optimization problems. *Lecture Notes in Computer Science* **4361**: 536-553 (2006)
31. Gabriel Ciobanu, Mario J. Pérez-Jiménez, Gheorghe Păun (Eds.): *Applications of Membrane Computing.* Natural Computing Series, Springer 2006, ISBN 978-3-540-25017-3
32. Gabriel Ciobanu, Gheorghe Păun, Mario J. Pérez-Jiménez: On the branching complexity of P systems. *Fundam. Inform.* **73**(1-2): 27-36 (2006)
33. Cosmin Bonchiş, Gabriel Ciobanu, Cornel Izbaşa: Encodings and arithmetic operations in membrane computing. *Lecture Notes in Computer Science* **3959**: 621-630

(2006)
34. Gabriel Ciobanu: *Theory and Applications of P Systems*, special issue *Int. J. Comput. Math.* **83** (2006)
35. Gabriel Ciobanu, Gheorghe Păun, Gheorghe Ştefănescu: P transducers. *New Generation Comput.* **24**(1): 1-28 (2005)
36. Cosmin Bonchiş, Gabriel Ciobanu, Cornel Izbaşa, Dana Petcu: A web-based P systems simulator and its parallelization. *Lecture Notes in Computer Science* **3699**: 58-69 (2005)
37. Daniela Besozzi, Gabriel Ciobanu: A P system description of the sodium-potassium pump. *Lecture Notes in Computer Science* **3365**: 210-223 (2004)
38. Gabriel Ciobanu, Grzegorz Rozenberg (Eds.): *Modelling in Molecular Biology.* Natural Computing Series, Springer 2004, ISBN 3-540-40799-5
39. Gabriel Ciobanu, Guo Wenyuan: P systems running on a cluster of computers. *Lecture Notes in Computer Science* **2933**: 123-139 (2003)
40. Gabriel Ciobanu, Rahul Desai, Akash Kumar: Membrane systems and distributed computing. *Lect. Notes in Computer Science* **2597**: 187-202 (2002)
41. Gabriel Ciobanu, Dorin Paraschiv: P system software simulator. *Fundam. Inform.* **49**(1-3): 61-66 (2002)

Bibliographies

Kernel P Systems Bibliography

Marian Gheorghe

School of Electrical Engineering and Computer Science, University of Bradford
Bradford BD7 1DP, UK
m.gheorghe@bradford.ac.uk

The concept of kernel P systems has been introduced in [5] and its definition revised in [6]. The research covers a broad spectrum of topics, from theory to verifications, applications and simulations. Apart from the papers listed below there is a PhD thesis, by Ciprian Dragomir, in preparation to be submitted.

References

1. M. E. Bakir, S. Konur, M. Gheorghe, I. M. Niculescu, F. Ipate, High Performance Simulations of Kernel P Systems, *Proc. IEEE International Conference on High Performance Computing and Communications, 22 – 24 August, Paris* (S. Khaddaj et al., eds.), 409 – 412, 2014.
2. M. Gheorghe, R. Ceterchi, F. Ipate, S. Konur, Kernel P Systems Modelling, Testing and Verification, *14^{th} Brainstorming Week on Membrane Computing, Sevilla* (submitted, 19 pages), 2016.
3. M. Gheorghe, R. Ceterchi, F. Ipate, S. Konur, Kernel P Systems Modelling, Testing and Verification – Sorting Case Study, *Proc. 17^{th} International Conference on Membrane Computing, Milan, 25 – 29 July, 2016* (A. Leporati, C. Zandron, eds.), 161 – 174, 2016.
4. M. Gheorghe, R. Ceterchi, F. Ipate, S. Konur, Kernel P Systems – Theory and Applications, to be submitted, 2016.
5. M. Gheorghe, F. Ipate, C. Dragomir, Kernel P Systems, *Proc. 10^{th} Brainstorming Week on Membrane Computing* (M. A. Martínez-del-Amor et al., eds.), Fénix Editora, Universidad de Sevilla, 153 – 170, 2012.
6. M. Gheorghe, F. Ipate, C. Dragomir, L. Mierlă, L. Valencia-Cabrera, M. García-Quismondo, M.J. Pérez-Jiménez, Kernel P Systems – Version 1, *Proc. 11th Brainstorming Week on Membrane Computing* (L. Valencia-Cabrera et al., eds.), Fénix Editora, Universidad de Sevilla, 97 – 124, 2013.

7. M. Gheorghe, F. Ipate, S. Konur, Kernel P Systems and Relationships with other Classes of P Systems, in *Multidisciplinary creativity – Homage to Gheorghe Păun on His 65th Birthday* (M. Gheorghe et al., eds.), Spandugino Publishing House, 64 – 76, 2015.
8. M. Gheorghe, F. Ipate, R. Lefticaru, M.J. Pérez-Jiménez, A. Ţurcanu, L. Valencia-Cabrera, M. García-Quismondo, L. Mierlă, 3-COL Problem Modelling Using Simple kernel P Systems, *International Journal of Computer Mathematics*, 90(4), 816 – 830, 2013.
9. M. Gheorghe, S. Konur, F. Ipate, Kernel P Systems and Stochastic P Systems for Modelling and Formal Verification of Genetic Logic Gates, in *Advances in Unconventional Computing* (A. Adamatzky, ed.), Emergence, Complexity and Computation Series, Volume 1: Theory, Springer, 661 – 676, 2015.
10. M. Gheorghe, S. Konur, F. Ipate, L. Mierlă, M. E. Bakir, M. Stannett, An Integrated Model Checking Toolset for Kernel P Systems, *Proc. 16^{th} Conference on Membrane Computing* (G. Rozenberg et al., eds.), *Lecture Notes in Computer Science*, 9504, 153 – 170, 2015.
11. M. Gheorghe, Gh. Păun, M. J. Pérez-Jiménez, G. Rozenberg, Research Frontiers of Membrane Computing: Open Problems and Research Topics, *International Journal of Foundations of Computer Science*, 24, 547 – 624, 2013.
12. F. Ipate, R. Lefticaru, L. Mierlă, L. Valencia-Cabrera, H. Han, G. Zhang, C. Dragomir, M. Gheorghe, Kernel P Systems: Applications and Implementations, *Proc. 8^{th} International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA, 12 – 14 July, Huang Shan* (Z. Yin et al., eds.), 1081 – 1089, 2013.
13. S. Konur, M. Gheorghe, C. Dragomir, L. Mierlă, F. Ipate, N. Krasnogor, Conventional Verification for Unconventional Computing: a Genetic XOR Gate Example, *Fundamenta Informaticae*, 134(1-2), 97 – 110, 2014.
14. S. Konur, M. Gheorghe, C. Dragomir, L. Mierlă, F. Ipate, N. Krasnogor, Qualitative and Quantitative Analysis of Systems and Synthetic Biology Constructs using P Systems, *ACS Synthetic Biology*, 4(1), 83 – 92, 2015.
15. S. N. Krishna, M. Gheorghe, C. Dragomir, Some Classes of Generalised Communicating P Systems and Simple Kernel P Systems, *Proc. 9^{th} International Conference on Computability in Europe, 1 – 5 July, Milan* (P. Bonizzoni et al., eds.), 284 – 293, 2013.
16. R. Lefticaru, L. F. Macías-Ramos, I. M. Niculescu, L. Mierlă, Towards Agent-Based Simulation of Kernel P Systems using FLAME and FLAME GPU, *Proc. Workshop on Membrane Computing, Manchester, 11 – 15 July, 2016* (M. Gheorghe, S. Konur, eds.), Technical Report of the University of Bradford, 58 – 61, 2016.
17. R. Lefticaru, L. F. Macías-Ramos, I. M. Niculescu, L. Mierlă, Agent-Based Simulation of Kernel P Systems with Division Rules using FLAME, *Proc. 17^{th} International Conference on Membrane Computing, Milan, 25 – 29 July, 2016* (A. Leporati, C. Zandron, eds.), 195 – 216, 2016.
18. I. M. Niculescu, M. Gheorghe, F. Ipate, A. Şefănescu, From Kernel P Systems to X-Machines and FLAME, *Journal of Automata, Languages and Combinatorics*, 19(1–4), 239 – 250, 2014.

PhD Theses in Membrane Computing

The following theses were totally or only partially dedicated to membrane computing. The list is mainly compiled from the bibliography given at `http://ppage.psystems.eu`, with some updates provided by Spain, Italy and, especially, China groups. There also are several theses in preparation at this moment. For sure, the list is not complete. We hope to make it as complete as possible in a future version.

1. O. Agrigoroaiei: *Causality and Coordination in Interaction-Based Systems*, Romanian Academy, Iaşi branch, Romania, 2012.
2. A. Alhazov: *Communication in Membrane Systems with Symbol Objects*, Tarragona University, Tarragona, Spain, 2006.
3. A. Alhazov: *Habilitation thesis: Small Abstract Computers*, Academy of Sciences of Moldova, Chişinău, Republic of Moldova, 2013.
4. B. Aman: *Spatial Dynamic Structures and Mobility in Computation*, Romanian Academy, Iaşi branch, Romania, 2009.
5. F. Arroyo-Montoro: *Structures and Biolanguage to Simulate Membrane Computing*, Madrid University, Madrid, Spain, 2004.
6. A. Arteta: *New Techniques for Implementing Membrane Systems*, Technical University of Madrid, Madrid, Spain, 2010.
7. F. Bernardini: *Membrane Systems for Molecular Computing and Biological Modelling*, University of Sheffield, Sheffield, UK, 2005.
8. D. Besozzi: *Computational and Modelling Power of P Systems*, University of Milan-Bicocca, Milano, Italy, 2003.
9. L. Bianco: *Membrane Models of Biological Systems*, Verona University, Verona, Italy, 2007.
10. C. Bonchiş: *Arithmetics and Information Theory in Membrane Computing*, West University, Timişoara, Romania, 2009.
11. R. Brijder: *Models of Natural Computation: Gene Assembly and Membrane Systems*, Leiden University, Leiden, The Netherlands, 2008.

12. R. Castellini: *Algorithms and Software for Biological MP Modeling by Statistical and Optimization Techniques*, Verona University, Verona, Italy, 2010.
13. M. Cavaliere: *Evolution, Communication and Observation: From Biology to Membrane Computing and Back*, Universidad de Sevilla, Seville, Spain, 2006.
14. P. Cazzaniga: *Stochastic Algorithms for Biochemical Processes*, Verona University, Verona, Italy, 2009.
15. H. Chen: *Research on Applications on Membrane Computing*, Chongqing University, Chongqing, China, 2011.
16. Y. Chen: *Formal Modeling and Simulation of Train Flow Based on High Confidence Wireless Communication*, Lanzhou Jiaotong Univ., China, 2014.
17. J. Cheng: *Autonomous Differential Evolution Algorithm: Design and Application*, Southwest Jiaotong Univ., China, 2015.
18. L. Ciencialová: *P Kolonie* (in Czech), Opava University, Opava, Czech Republic, 2008.
19. C. Dragomir:
20. D. Diaz-Pernil: *Sistemas Celulares de Tejidos: Formalización y Eficiencia Computacional*, Universidad de Sevilla, Seville, Spain, 2008.
21. I. Dincă: *Modelling and Specification of Parallel and Distributed Systems*, University of Piteşti, Piteşti, Romania, 2011.
22. G. Franco: *Biomolecular Computing – Combinatorial Algorithms and Laboratory Experiments*, Università degli Studi di Verona, Italy, 2006.
23. P. Frisco: *Theory of Molecular Computing. Splicing and Membrane Systems*, Leiden University, Leiden, The Netherlands, 2004.
24. M. Garcia-Quismondo-Fernandez: *Modelling and Simulation of Real-Life Phenomena in Membrane Computing*, Universidad de Sevilla, Seville, Spain, 2014.
25. C. Graciani-Diaz: *Especificacin y Verificación de Programas Moleculares en PVS*, Universidad de Sevilla, Seville, Spain, 2003.
26. J. He: *Research on Membrane-Inspired Algorithms and Applications*, Huazhong University of Science and Technology, Wuhan, China, 2014.
27. S. Hemalatha, *A Study on Rewriting P Systems, Splicing Grammar Systems and Picture Array Languages*, Anna University, Chennai, India, 2007.
28. L. Hu: *Research on Fault Knowledge Evolution Technologies of Human-Machine Synergy Diagnosis for CNC Machine Tools*, Chongqing Univ., China, 2015.
29. L. Huang: *Research on Membrane Computing Optimization Methods*, Zhejiang University, Hangzhou, China, 2007.
30. A.M. Ionescu: *Membrane Computing: Traces, Neural Inspired Models, Controls*, Tarragona University, Spain, 2008.
31. T. Isdorj-Onolt: *Membrane Computing, Neural Inspirations, Gene Assembly in Ciliates*, Universidad de Sevilla, Seville, Spain, 2007.
32. C. Izbaşa: *Information Theory and Computation over Multisets*, West University, Timişoara, Romania, 2010.
33. J.R. Jack: *Discrete Nondeterministic Modeling of Biochemical Networks*, Louisiana Technical University, Ruston, USA, 2009.

34. Y. Jiang: *Research on the Computational Power of P Systems with Network Structure*, Huazhong University of Science and Technology, Wuhan, China, 2011.
35. K. Jiang: *Research on Sequential Spiking Neural P Systems Based on Spike Number*, Huazhong University of Science and Technology, Wuhan, China, 2014.
36. Y.-B. Kim: *Distributed Algorithms in Membrane Systems*, The University of Auckland, Auckland, New Zealand, 2012.
37. Y. Kong: *Research on the Computational Power of Spiking Neural P Systems with Differential Biological Backgrounds*, Huazhong University of Science and Technology, Wuhan, China, 2015.
38. S.N. Krishna: *Languages of P Systems: Computability and Complexity*, University of Madras, Chennai, India, 2001.
39. L. Lakshmanan: *On the Crossroads of P Systems and Contextual Grammars: Variants, Computability, Complexity and Efficiency*, Indian Institute of Technology, Madras, India, 2003.
40. R. Lefticaru: *Testing Methods Based on Formal Specifications*, University of Piteşti, Romania, 2011.
41. C. Liu: *Research on Optimization Methods of Cell-Like Membrane Computing and Their Applications*, Dalian University of Technology, Dalian, China, 2014.
42. R. Lombardo: *Unconventional Computations and Genome Representations*, Verona University, Italy, 2016.
43. C. Lu: *Research on Computational Properties of Membrane Systems with Proteins*, Huazhong University of Science and Technology, Wuhan, China, 2012.
44. L.F. Macías-Ramos: *Developing efficient simulators for cell machines*, Universidad de Sevilla, Seville, Spain, 2016
45. M. Madhu: *Studies of P Systems as a Model of Cellular Computing*, Indian Institute of Technology, Madras, India, 2003.
46. L. Marchetti: *MP representations of Biological Structures and Dynamics*, Università di Verona, Verona, Italy, 2012.
47. M.A. Martinez-del-Amor: *Accelerating Membrane Systems Simulators Using High Performance Computing with GPU*, Universidad de Sevilla, Seville, Spain, 2013.
48. V.P. Metta: *Modelling and Simulation of Spiking Neural P systems Using Petri Nets*, Thapar University, Patiala, Indian, 2012.
49. P. Milazzo: *Qualitative and Quantitative Formal Modeling of Biological Systems*, Pisa University, Pisa, Italy, 2006.
50. Y. Niu: *Research on Models and Algorithms of Membrane Computing for Solving Computational Intractable Problems*, Huazhong University of Science and Technology, Wuhan, China, 2012.
51. M. Oswald: *P Automata*, Vienna University of Technology, Wien, Austria, 2004.
52. R. Pagliarini: *Modelling and Reverse-Engineering of Biological Phenomena by Means of Metabolic P Systems*, Verona University, Verona, Italy, 2011.

53. A. Păun: *Unconventional Models of Computation: DNA and Membrane Computing*, Western University, London, Canada, 2003.
54. A.B. Pavel: *Development of Robot Controllers Using Membrane Computing*, Politechnica University, Bucharest, Romania, 2015.
55. X. Peng: *Research on Computational Power and Applications of Spiking Neural P Systems*, Central South University, Changsha, China, 2013.
56. I. Pérez-Hurtado de Mendoza: *Desarrollo y Aplicaciones de un Entorno de Programación para Computación Celular: P-Lingua*, Universidad de Sevilla, Seville, Spain, 2010.
57. D. Pescini: *Modelling, Analysis and Stochastic Simulations of Biological Systems*, University of Milano-Bicocca, Milano, Italy, 2007.
58. B. Popa: *Membrane Systems with Limited Parallelism*, Louisiana Technical University, Ruston, USA, 2006.
59. A.E. Porreca: *The Time-Space Trade-Off in Membrane Computing*, Milano Bicocca University, Italy, 2012.
60. A. Ramanujan: *Control Languages in P Systems*, Indian Institute of Technology Madras, Chennai, Indian, 2014.
61. A. Riscos-Núñez: *Programación Celular: Resolución Eficiente de Problemas Numéricos NP-completos*, Universidad de Sevilla, Seville, Spain, 2004.
62. F. Romero-Campero: *P Systems: A Computational Modelling Framework for Systems Biology*, Universidad de Sevilla, Seville, Spain, 2008.
63. Á. Romero-Jimńez: *Complejidad y Universalidad en Modelos de ComputaciÓn Celular*, Universidad de Sevilla, Seville, Spain, 2002.
64. F. Sancho-Caparrini: *Verificación de Programas en Modelos de Computación no Convencionales*, Universidad de Sevilla, Seville, Spain, 2002.
65. D. Sburlan: *Promoting and Inhibiting Contexts in Membrane Computing*, Universidad de Sevilla, Seville, Spain, 2006.
66. T. Song: *Research on Computational Properties and Applications of Spiking Neural P Systems*, Huazhong University of Science and Technology, Wuhan, China, 2013.
67. B. Song: *Research on Computational Property of Timed Membrane Systems*, Huazhong University of Science and Technology, Wuhan, China, 2015.
68. I. Stamatopoulou: *A Formal Framework for the Modelling of Multi-Agent Systems with Dynamic Structure*, University of Thessaloniki, Thessalonica, Greece, 2008.
69. Y. Sun: *Process Modeling and Operation Optimization Methods for Coal Water Slurry Gasification Unit*, East China University of Science and Technology, 2012.
70. C. Ştefan: *Parallel and Distributed Communication Systems: A Formal Approach*, University of Piteşti, Romania, 2012.
71. A. Ţurcanu: *Modelling, Testing and Verification of Biology Inspired Systems*, University of Piteşti, Piteşti, Romania, 2013.

72. L. Valencia-Cabrera: *An Environment for Virtual Experimentation with Computational Models Based on P Systems*, Universidad de Sevilla, Seville, Spain, 2015.
73. C.-I. Vasile: *Distributed Control for Multi-Robot Systems*, Politechnica University, Bucharest, Romania, 2015.
74. C. Vasilica Ferent: *Methodologies for Specification and Validation of Software Systems*, ???, 2011.
75. J. Wang: *Spiking Neural P Systems*, Leiden University, Leiden, The Netherlands, 2011.
76. J. Wang: *Computational Power of Spiking Neural Membrane Systems Inspired by Neural Systems*, Huazhong University of Science and Technology, Wuhan, China, 2013.
77. H. Wu: *P Systems as Formal Models for Distributed Algorithms*, The University of Auckland, Auckland, New Zealand, 2014.
78. J. Xiao: *The Research on Bio-Inspired Encoding Algorithms for DNA Computing*, Huazhong Univ. of Science and Technology, Wuhan, China, 2008.
79. G. Xiong: *Research on Methods for Fault Diagnosis of Power Grids Based on Computational Intelligence*, Huazhong Univ. of Science and Technology, China, 2014.
80. W. Xiong: *Research on Traffic Assignment Model Considering Emission Effects and Solution Algorithm*, Wuhan University of Technology, Wuhan, China, 2009.
81. J. Xue: *Two Classes of Biological Computing and Applications in Data Mining*, Shandong Normal University, Ji'nan, China, 2015.
82. S. Yang: *Research on P systems Based Optimization Algorithms and Applications*, Zhejiang University, Hangzhou, China, 2012.
83. C. Zandron: *A Model for Molecular Computing: Membrane Systems*, University of Milano-Bicocca, Milano, Italy, 2001.
84. X. Zeng: *Research on Computational Property of Spiking Neural P Systems*, Huazhong University of Science and Technology, Wuhan, China, 2011.
85. X. Zhang: *Research on the Computational Power of Spiking Neural P Systems*, Huazhong University of Science and Technology, Wuhan, China, 2009.
86. J. Zhao: *Research on Bio-inspired Optimization Algorithms Based on Membrane Computing and Applications*, Zhejiang University, Hangzhou, China, 2010.

It might be instructive to also have a view of the evolution of the number of PhD theses presented in each year (the increasing tendency of the last years is obvious – for sure, a consequence of the interest for MC in China):

2001	2	2009	6
2002	2	2010	5
2003	5	2011	8
2004	4	2012	10
2005	1	2013	6
2006	6	2014	8
2007	5	2015	8
2008	7	2016	3?

Membrane Computing
16th International Conference, CMC 2015
Valencia, Spain, August 17-21, 2015
Revised Selected Papers
LNCS 9504, Springer-Verlag, Berlin, 2015

Grzegorz Rozenberg, Arto Salomaa, José M. Sempere, Claudio Zandon, editors

Preface

The present volume contains the invited contributions and a selection of papers presented at the 16^{th} International Conference on Membrane Computing (CMC16), that was held in Valencia, Spain, 17-21 August, 2015 (website address: `http://users.dsic.upv.es/workshops/cmc16/`), as well as three selected papers from the Asian Conference on Membrane Computing (ACMC) 2015, held in Anhui University, Hefei, Anhui, China, 12-15 November, 2015 (website address: `http://2015.asiancmc.org/`).

The CMC series started with three workshops organized in Curtea de Argeş, Romania, in 2000, 2001 and 2002. The workshops were then held in Tarragona, Spain (2003), Milan, Italy (2004), Vienna, Austria (2005), Leiden, The Netherlands (2006), Thessaloniki, Greece (2007), and Edinburgh, UK (2008).

The 10^{th} edition was organized again in Curtea de Argeş, in August 2009, where it was decided to continue the series as the Conference on Membrane Computing (CMC). The following editions were held in Jena, Germany (2010), Fontainebleau, France (2011), Budapest, Hungary (2012), Chişinău, Moldova (2013), and Praha, Czech Republic (2014).

A regional version of CMC, the Asian Conference on Membrane Computing, ACMC, started in 2012 in Wuhan (China), and continued in Chengdu, China (2013) and Coimbatore, India (2014).

CMC16 has been organized, under the auspices of the European Molecular Computing Consortium (EMCC), by the Research Group on Computation Models and Formal Languages of the Universitat Politècnica de València and it was supported by the Escuela Técnica de Ingeniería Informática (ETSINF, UPV).

CMC16 consisted of three different parts: the first day was organized as a Tutorial Day, with lectures by Gheorghe Păun, Rudolf Freund, Claudio Zandron, Gyorgy Vaszil, and Agustín Riscos-Núñez. From Tuesday to Thursday the con-

ference continued with standard sessions; invited lectures were given by Ion Petre (Abo Akademi University, Finland), Andrés Moya (Universitat de València, Spain) and Vincenzo Manca (University of Verona, Italy). The last day of the conference was devoted to presentation of extended abstracts and interaction between participants. Based on the votes of the CMC16 participants, the Best Paper Award of this edition was given to Rudolf Freund and Petr Sosík for their paper "On the Power of Catalytic P Systems with One Catalyst".

The editors express their gratitude to the Program Committee, the invited speakers, the authors of the papers, the reviewers, and all the participants for their contributions to the success of CMC16.

November 2015

Grzegorz Rozenberg
Arto Salomaa
José M. Sempere
Claudio Zandron

The Programme Committee of CMC16 consisted of

Artiom Alhazov (Chişinău, Moldova)
Matteo Cavaliere (Edinburgh, Scotland)
Ludek Cienciala (Opava, Czech Republic)
Gabriel Ciobanu (Iaşi, Romania)
Erzsébet Csuhaj-Varjú (Budapest, Hungary)
Giuditta Franco (Verona, Italy)
Rudolf Freund (Wien, Austria)
Marian Gheorghe (Bradford, UK)
Thomas Hinze (Cottbus, Germany)
Florentin Ipate (Bucharest, Romania)
Shankara Narayanan Krishna (Bombay, India)
Alberto Leporati (Milan, Italy)
Vincenzo Manca (Verona, Italy)
Maurice Margenstern (Metz, France)
Giancarlo Mauri (Milan, Italy)
Radu Nicolescu (Auckland, New Zealand)
Linqiang Pan (Wuhan, China)
Gheorghe Păun (Bucharest, Romania and Sevilla, Spain)
Mario de Jesús Pérez Jiménez (Sevilla, Spain)
Dario Pescini (Milan, Italy)
Agustín Riscos-Núñez (Sevilla, Spain)
José M. Sempere (Valencia, Spain) – Co-chair
Petr Sosík (Opava, Czech Republic)
Gyorgy Vaszil (Debrecen, Hungary)
Sergey Verlan (Paris, France)
Claudio Zandron (Milan, Italy) – Co-chair
Gexiang Zhang (Chengdu, Sichuan, China)

Additonal reviewers:

Bogdan Aman (Iaşi, Romania)
Raluca Lefticaru (Bucharest, Romania)
Alin Stefanescu (Bucharest, Romania)
Álvaro Romero Jiménez (Sevilla, Spain)
Andrei Alexandru (Iaşi, Romania)
Miguel Á. Martínez-Del-Amor (Sevilla, Spain)

Contents

Invited Papers

Regular Papers

Membranes Agents – Book Summary

Luděk Cienciala

Research Institute of the IT4Innovations Centre of Excellence,
Faculty of Philosophy and Science, Silesian University in Opava, Czech Republic
ludek.cienciala@fpf.slu.cz

1 Summary

The book *Membránoví agenti – o variantách P kolonií* (Membrane Agents – About Variants of P Colonies) by doc. RNDr. Luděk Cienciala, Ph.D., was published in 2015 by Silesian University in Opava.

In this book the author focused on one of the modern areas in theoretical computer science, computational models inspired by the biochemistry of cells, called membrane or P systems.

The membrane systems were introduced by Gheorghe Păun in 1998 as a distributed parallel computing model inspired by the biochemistry of cells. This model consists of membranes hierarchically embedded in the outermost skin membrane. Each membrane delimits a region which can contain objects; these objects can be seen as symbols from a given alphabet. There is a set of rules associated with every region, and the objects can evolve or be transported to their parent or children regions according to these rules.

P colonies were introduced in 2004 as an abstract computing device evolved from membrane systems; they are composed of independent single membrane agents, reactively acting and evolving in a shared environment.

Different variants of P colonies have been derived from the original model since that time. The book contains not only a brief introduction to membrane systems but also a compendium of different types of P colonies together with a comparison of results obtained in research devoted to these systems.

2 Contents

Advances in Unconventional Computing. Volume 1: Theory, Volume 2: Prototypes, Models and Algorithms. Springer, 2016

Andrew Adamatzky, editor

Director of the Unconventional Computing Centre
UWE, Bristol BS16 1QY UK
andrew.adamatzky@uwe.ac.uk, adamatzky@gmail.com

Unconventional computing is a science in flux. What is unconventional today will be conventional tomorrow. Designs being standard in the past are seen now as a novelty. Unconventional computing is a niche for interdisciplinary science, a cross-breed of computer science, physics, mathematics, chemistry, electronic engineering, biology, materials science and nanotechnology. The aims are to uncover and exploit principles and mechanisms of information processing in, and functional properties of, physical, chemical and living systems to develop efficient algorithms, design optimal architectures and manufacture working prototypes of future and emergent computing devices.

I invited worlds leading scientists and academicians to describe their vision of unconventional computing and to highlight the most promising directions of future research in the field. Their response was overwhelmingly enthusiastic: over 50 chapters were submitted spanning almost all the fields of natural and engineering sciences. Unable to fit over one and half thousands pages into one volume, I grouped the chapters as "theoretical" and "practical". By "theoretical" I mean constructs and algorithms which have no immediate application domain and do not solve any concrete problems, yet they make a solid mathematical or philosophical foundation to unconventional computing. "Practical" includes experimental laboratory implementations and algorithms solving actual problems. Such a division is biased by my personal vision of the field and should not be taken as an absolute truth.

The first volume brings us mind-bending revelations from gurus in computing and mathematics. The topics covered are computability, (non-)universality and complexity of computation; physics of computation, analog and quantum computing; reversible and asynchronous devices; cellular automata and other mathematical machines; P-systems and cellular computing; spatial computation; chemical

and reservoir computing. As a dessert we have two vibrant memoirs by founding fathers of the field.

The second volume is a tasty blend of experimental laboratory results, modelling and applied computing. Emergent molecular computing is presented by enzymatic logical gates and circuits, and DNA nano-devices. Reaction-diffusion chemical computing is exemplified by logical circuits in BelousovZhabotinsky medium and geometrical computation in precipitating chemical reactions. Logical circuits realised with solitons and impulses in polymer chains show advances in collision-based computing. Photo-chemical and memristive devices give us a glimpse into the hot topics of novel hardware. Practical computing is represented by algorithms of collective and immune-computing and nature-inspired optimisation. Living computing devices are implemented in real and simulated cells, regenerating organisms, plant roots and slime moulds. Musical biocomputing and living architectures make the ending of our unconventional journey non-standard.

The chapters are self-contained. No background knowledge is required to enjoy the book. Each chapter is a treatise of marvellous ideas.

Volume 1 (ISBN-13: 978-3319339238)

1. Selim G. Akl: Nonuniversality in Computation: Fifteen Misconceptions Rectified
2. Vladik Kreinovich and Olga Kosheleva: What Is Computable? What Is Feasibly Computable? A Physicists Viewpoint
3. Norman Margolus: The Ideal Energy of Classical Lattice Dynamics
4. Tania Ambaram, Edwin Beggs, Jose Felix Costa, Diogo Pocas and John V. Tucker: An Analogue-Digital Model of Computation: Turing Machines with Physical Oracles
5. Bruce J. MacLennan: Physical and Formal Aspects of Computation: Exploiting Physics for Computation and Exploiting Computation for Physical Purposes
6. Jerome Durand-Lose: Computing in Perfect Euclidean Frameworks
7. Ed Blakey: Unconventional Computers and Unconventional Complexity Measures
8. Mark Burgin: Decreasing Complexity in Inductive Computations
9. Hector Zenil and Jürgen Riedel: Asymptotic Intrinsic Universality and Natural Reprogrammability by Behavioural Emulation
10. Kenichi Morita: Two Small Universal Reversible Turing Machines
11. Dmitry I. Iudin and Yaroslav D. Sergeyev: Percolation Transition and Related Phenomena in Terms of Grossone Infinity Computations
12. Tommaso Bolognesi: Spacetime Computing: Towards Algorithmic Causal Sets with Special-Relativistic Properties
13. Antoine Spicher and Jean-Louis Giavitto: Interaction-Based Programming in MGS
14. Maurice Margenstern: Cellular Automata in Hyperbolic Spaces

15. Genaro J. Martinez, Andrew Adamatzky, and Harold V. McIntosh: A Computation in a Cellular Automaton Collider Rule 110
16. Karl Svozil: Quantum Queries Associated with Equi-partitioning of States and Multipartite Relational Encoding Across Space-Time
17. Cristian S. Calude and Michael J. Dinneen: Solving the Broadcast Time Problem Using a D-wave Quantum Computer
18. Alexis De Vos and Stijn De Baerdemacker: The Group Zoo of Classical Reversible Computing and Quantum Computing
19. Martin Lukac, Michitaka Kameyama, Marek Perkowski, Pawel Kerntopf, and Claudio Moraga: Fault Models in Reversible and Quantum Circuits
20. Hiroshi Umeo: A Class of Non-optimum-time FSSP Algorithms
21. Jia Lee: Universality of Asynchronous Circuits Composed of Locally Reversible Elements
22. Matthew Dale, Julian F. Miller, and Susan Stepney: Reservoir Computing as a Model for In-Materio Computing
23. Zoran Konkoli: On Reservoir Computing: From Mathematical Foundations to Unconventional Applications
24. Paolo Dini: Computational Properties of Cell Regulatory Pathways Through Petri Nets
25. Marian Gheorghe, Savas Konur, and Florentin Ipate: Kernel P Systems and Stochastic P Systems for Modelling and Formal Verification of Genetic Logic Gates
26. Erik Bergh and Zoran Konkoli: On Improving the Expressive Power of Chemical Computation
27. Andrew Schumann: Conventional and Unconventional Approaches to Swarm Logic
28. Christopher Bennett, Aldo Jesorka, Goran Wendin, and Zoran Konkoli: On the Inverse Pattern Recognition Problem in the Context of the Time-Series Data Processing with Memristor Networks
29. Nicholas J. Macias and Lisa J.K. Durbeck: Self-Awareness in Digital Systems: Augmenting Self-Modification with Introspection to Create Adaptive, Responsive Circuitry
30. Gheorghe Păun: Looking for Computers in the Biological Cell. After Twenty Years
31. Cristian S. Calude: Unconventional Computing: A Brief Subjective History

Volume 2 (ISBN-13: 978-3319339207)

1. Matthew R. Lakin, Milan N. Stojanovic, and Darko Stefanovic: Implementing Molecular Logic Gates, Circuits, and Cascades Using DNAzymes
2. Evgeny Katz and Brian E. Fratto: Enzyme-Based Reversible Logic Gates Operated in Flow Cells
3. Sergii Domanskyi and Vladimir Privman: Modeling and Modifying Response of Biochemical Processes for Biocomputing and Biosensing Signal Processing

4. Jerzy Gorecki, Joanna N. Gorecka, Bogdan Nowakowski, Hiroshi Ueno, Tatsuaki Tsuruyama, and Kenichi Yoshikawa: Sensing Parameters of a Time Dependent Inflow with an Enzymatic Reaction
5. Mingzhu Sun and Xin Zhao: Combinational Logic Circuit Based on BZ Reaction
6. James Stovold and Simon OKeefe: Associative Memory in Reaction-Diffusion Chemistry
7. Ben De Lacy Costello and Andrew Adamatzky: Calculating Voronoi Diagrams Using Chemical Reactions
8. Larry Bull, Rita Toth, Chris Stone, Ben De Lacy Costello, and Andrew Adamatzky: Light-Sensitive BelousovZhabotinsky Computing Through Simulated Evolution
9. Ivan Zelinka: On Synthesis and Solutions of Nonlinear Differential Equations – A Bio-Inspired Approach
10. Kohta Suzuno, Daishin Ueyama, Michal Branicki, Rita Toth, Artur Braun, and Istvan Lagzi: Marangoni Flow Driven Maze Solving
11. Jitka Cejkova, Silvia Holler, To Quyen Nguyenova, Christian Kerrigan, Frantisek Stepanek, and Martin M. Hanczyc: Chemotaxis and Chemokinesis of Living and Non-living Objects
12. Mariusz H. Jakubowski, Ken Steiglitz, and Richard Squier: Computing with Classical Soliton Collisions
13. Ken Steiglitz: Soliton-Guided Quantum Information Processing
14. Stefano Siccardi and Andrew Adamatzky: Models of Computing on Actin Filaments
15. Reem Mokhtar, Sudhanshu Garg, Harish Chandran, Hieu Bui, Tianqi Song, and John Reif: Modeling DNA Nanodevices Using Graph Rewrite Systems
16. Hajo Broersma, Julian F. Miller, and Stefano Nichele: Computational Matter: Evolving Computational Functions in Nanoscale Materials
17. Kacper Pilarczyk, Przemyslaw Kwolek, Agnieszka Podborska, Sylwia Gaweda, Marek Oszajca, and Konrad Szacilowski: Unconventional Computing Realized with Hybrid Materials Exhibiting the PhotoElectrochemical Photocurrent Switching (PEPS) Effect
18. Silvia Battistoni, Alice Dimonte, and Victor Erokhin: Organic Memristor Based Elements for Bio-inspired Computing
19. Ella Gale: Memristors in Unconventional Computing: How a Biomimetic Circuit Element Can be Used to Do Bioinspired Computation
20. Xin-She Yang: Nature-Inspired Computation: An Unconventional Approach to Optimization
21. Jeff Jones: On Hybrid Classical and Unconventional Computing for Guiding Collective Movement
22. Nikolaos P. Bitsakidis, Nikolaos I. Dourvas, Savvas A. Chatzichristofis, and Georgios Ch. Sirakoulis: Cellular Automata Ants
23. Krzysztof Pancerz and Andrew Schumann: Rough Set Description of Strategy Games on Physarum Machines

Tutorials, Surveys

Complexity Perspectives on Minimal Cooperation in Cell-like Membrane Systems

Luis Valencia-Cabrera, David Orellana-Martín,
Agustín Riscos-Núñez, Mario J. Pérez-Jiménez

Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
Universidad de Sevilla
Avda. Reina Mercedes s/n, 41012 Sevilla, Spain
E-mail: lvalencia@us.es,dorellana@us.es,ariscosn@us.es,marper@us.es

Summary. Cooperation is doubtless a critical ingredient of a computing model. This paper provides an overview on results showing how forbidding cooperation, or allowing it only a minimum degree, influences the computing power.

In particular, we restrict ourselves to two types of cell-like membrane systems. On one hand, we analyze the efficiency of polarizationless P systems with active membranes without dissolution rules when minimal cooperation is permitted in object evolution rules. On the other hand, cell-like P systems with symport/antiport rules of minimal length are also addressed.

Specifically, assuming that **P** is not equal to **NP**, several frontiers of the efficiency are obtained in these two frameworks.

1 Introduction

The dynamics of cell-like membrane systems is captured by means of rewriting rules that allow the evolution of objects. Any object, alone or together with one more object can be transformed into other objects, can pass through one membrane and can dissolve/divide/separate the membrane where it currently resides [8]. Non-cooperative systems only contain rules abstracting proteins that transport only one solute across the membrane (for example, the glucose transporter in red blood cell membranes).

P systems with active membranes [9] is an interesting particular case of non-cooperative system. In the first version of these systems, electrical charges from the set $\{+, -, 0\}$ are associated with membranes. Besides, a rule associated with a

membrane labelled by h is applicable to a configuration if the object corresponding to its left-hand side is in membrane h of that configuration, and the membrane has a polarization prefixed in the rule. Thus, in some sense, there exists a kind of cooperation/collaboration in order to trigger the rule.

A particular case of cooperative systems are the *cell-like P systems with symport/antiport rules* [7] where there exist rules with the length at least two. In these systems, communication is implemented by means of symport/antiport rules abstracting trans-membrane transport of couples of chemical substances, in the same or in opposite directions.

As it is stated by the so-called Milano theorem [17], no computationally hard problems can be solved in polynomial time without using rules allowing the generation of an exponential number of membranes/cells in polynomial time, unless $\mathbf{P} = \mathbf{NP}$ holds. So, in order to achieve efficiency (the ability to provide polynomial-time solution to computationally hard problems), division rules abstracting the cell division process and separation rules abstracting the membrane fission process are also included.

It is worth pointing out that dissolution rules and division rules for non-elementary membranes are not necessary/relevant to get the computational efficiency in (non-cooperative) P systems with active membranes. Nevertheless, if electrical charges are forbidden, then dissolution rules start to play a relevant role in these systems from a computational complexity point of view. The reason is that only problems in class $\mathbf{P}$ can be efficiently solved by means of families of such systems without dissolution rules, even in the case that division rules for non-elementary membranes are allowed [3]. The situation is similar for cell-like P systems with symport/antiport rules where only a single object is involved in any rule (so, only symport rules with length 1 are permitted).

The main goal of this work is to present recent results concerning the computational efficiency of both polarizationless P systems with active membranes and cell-like P systems with symport/antiport rules when *minimal cooperation* is considered. The term "minimal cooperation" is used in the following sense: in the first case, the left-hand side of each rule has at most a couple of objects; in the second case, at most two objects are involved in each communication (symport/antiport) rule.

2 The computational frameworks

2.1 Polarizationless P systems with active membranes

A polarizationless P system with active membranes $(\Gamma, H, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}, i_{out})$ of degree q, can be viewed as a set of q membranes, labelled by elements of H, arranged in a hierarchical structure μ given by a rooted tree, such that: (a) $\mathcal{M}_1, \ldots, \mathcal{M}_q$ represent the finite multisets of *objects* initially placed in the q membranes of the system; (b) $\mathcal{R}$ is a finite set of rules over Γ associated with the labels; (c) $i_{out} \in H \cup \{env\}$ indicates the output region. The leaves of μ are called

elementary membranes; any other membrane is said to be non-elementary. The semantics of polarizationless P systems with active membranes is defined as usual P systems with active membranes (see [4, 9] for details).

A new kind of models can be considered when separation rules, inspired by the membrane fission process, are used instead of division rules as a mechanism to produce an exponential number of membranes in linear time. These rules are associated with a (prefixed) partition $\{\Gamma_0, \Gamma_1\}$ of the working alphabet Γ and a (prefixed) partition $\{H_0, H_1\}$ of the set of labels H.

- *Separation for elementary membranes*: $[\, a \,]_h \rightarrow [\, \Gamma_0 \,]_h \, [\, \Gamma_1 \,]_h$ for $h \in H \setminus \{i_{out}\}$, $a \in \Gamma$ and h is not the label of the root of μ.
- *Separation for non-elementary membranes*: $[[\,]_{h_0}[\,]_{h_1}]_h \rightarrow [\Gamma_0[\,]_{h_0}]_h \, [\Gamma_1[\,]_{h_1}]_h$, where $h \in H \setminus \{i_{out}\}$ is not the label of the root of μ, $h_0 \in H_0$ and $h_1 \in H_1$.

A separation rule $[\, a \,]_h \rightarrow [\ \Gamma_0 \ ]_h \ [\ \Gamma_1 \ ]_h$ is applicable to a configuration $\mathcal{C}_t$ at an instant t, if there exists a membrane labelled by h in $\mathcal{C}_t$ such that it is an elementary membrane in $\mathcal{C}_t$ and contains object a. When applying such a rule, the membrane is separated into two membranes with the same label; at the same time, object a is consumed and the multiset of objects contained in membrane h gets distributed: the objects from Γ_0 are placed in one membrane, those from Γ_1 are placed in another.

A separation rule $[\,[\,]_{h_0}[\,]_{h_1}]_h \rightarrow [\ \Gamma_0 \ [\,]_{h_0}]_h \ [\ \Gamma_1 \ [\,]_{h_1}]_h$ is applicable to a configuration $\mathcal{C}_t$ at an instant t, if there exists a membrane labelled by h in $\mathcal{C}_t$ such that it contains a membrane labelled by h_0 and another membrane labelled by h_1. When applying such a separation rule to a membrane labelled by h in $\mathcal{C}_t$, that membrane is separated into two membranes with the same label, in such a way that the contents (objects and inner membranes) are distributed as follows: The first membrane receives the objects from Γ_0, and all inner membranes whose label belongs to H_0; while the second membrane receives the objects from Γ_1, and all inner membranes whose label belongs to H_1.

Minimal cooperation in object evolution rules

Next, minimal cooperation in objects evolution rules is introduced in the framework of polarizationless P systems with active membranes. The term "minimal cooperation" is used in the following sense: the left-hand side of such rules consists of two symbols.

Definition 1. *In the context of polarizationless P systems with active membranes, the following kinds of minimal cooperation in object evolution rules are considered.*

- Primary minimal cooperation (**pmc**)*: object evolution rules are of the form* $[\, u \rightarrow v \,]_h$*, where* $h \in H$*,* u, v *are multisets over* Γ*, and* $1 \leq |u|, |v| \leq 2$*, but at least one object evolution rule verifies* $|u| = 2$*.*

- Bounded minimal cooperation (**bmc**)*: object evolution rules are of the form* $[\, u \to v \,]_h$*, where* $h \in H$*,* u, v *are multisets over* Γ*, and* $1 \leq |v| \leq |u| \leq 2$*, but at least one object evolution rule verifies* $|u| = 2$*.*
- Minimal cooperation and minimal production (**mcmp**)*: object evolution rules are of the form* $[\, a \to b \,]_h$*,* $[\, a\, b \to c \,]_h$*, for* $h \in H$ *and* $a, b, c \in \Gamma$*, but at least one object evolution rule is of the second type.*

We denote by $\mathcal{DAM}^0(\alpha, \beta, \gamma, \delta)$ (respectively, $\mathcal{SAM}^0(\alpha, \beta, \gamma, \delta)$) the class of all polarizationless P systems with active membranes and with division rules (respectively, separation rules), where the meaning of parameters α, β, γ and δ is the following:

- If $\alpha = pmc$ (resp. $\alpha = bmc$ or $\alpha = mcmp$) then primary minimal cooperation (resp. bounded minimal cooperation or minimal cooperation and minimal production) in object evolution rules are permitted. If $\alpha = +e$ then any non-cooperative object evolution rule is permitted.
- If $\beta = +c$ (resp. $\beta = -c$) then communication rules are permitted (resp. forbidden).
- If $\gamma = +d$ (resp. $\gamma = -d$) then dissolution rules are permitted (resp. forbidden).
- If $\delta = +n$ (resp. $\delta = -n$) then division/separation rules for elementary and non–elementary membranes are permitted (resp. only division/separation rules for elementary membranes are permitted).

In this context, *counting* membrane systems instead of *recognizer* membrane systems can be considered [10]. The output of these systems will be natural numbers.

2.2 Cell-like P systems with symport/antiport rules

Cell-like P systems with symport/antiport rules were introduced in [7] aiming to abstract the biological phenomenon of trans-membrane transport of couples of chemical substances, in the same or in opposite directions.

A P system with symport/antiport rules of degree q

$$\Pi = (\Gamma, \mathcal{E}, \mu, \mathcal{M}_1, \ldots, \mathcal{M}_q, \mathcal{R}_1, \cdots, \mathcal{R}_q, i_{out})$$

can be viewed as a set of q membranes, labelled by $1, \ldots, q$, arranged in a hierarchical structure μ given by a rooted tree whose root is called the *skin membrane*, such that: (a) $\mathcal{M}_1, \ldots, \mathcal{M}_q$ represent the finite multisets of objects initially placed in the q membranes of the system; (b) $\mathcal{E}$ is the set of objects initially located in the environment of the system, all of them available in an arbitrary number of copies; (c) $\mathcal{R}_1, \cdots, \mathcal{R}_q$ are finite sets of communication rules over Γ ($\mathcal{R}_i$ is associated with the membrane i of μ); and (d) i_{out} represents a distinguished *region* which will encode the output of the system. We use the term *region* i ($0 \leq i \leq q$) to refer to membrane i in the case $1 \leq i \leq q$ and to refer to the environment in the case $i = 0$. The length of rule (u, out) or (u, in) (resp. $(u, out; v, in)$) is defined as $|u|$ (resp. $|u| + |v|$). Division rules and separation rules are introduced in these models

in a similar way that in polarizationless P systems with active membranes, both in syntax and semantics.

For each natural number $k \geq 1$, we denote by $\mathcal{CDC}(k)$ (respectively, $\mathcal{CSC}(k)$) the class of P systems with symport/antiport rules and division rules (respectively, separation rules) such that the length of the communication rules is at most k. When division or separation for non-elementary membranes is permitted then we denote $\mathcal{CD}_{ne}\mathcal{C}(k)$ and $\mathcal{CS}_{ne}\mathcal{C}(k)$, respectively. If the alphabet of the environment is the empty set then we write $\widehat{\mathcal{CDC}}(k)$, $\widehat{\mathcal{CSC}}(k)$, respectively.

3 Proof techniques

The results described at the next Section have been obtained by using three different proof techniques:

- *Dependency graph*: a directed graph (*dependency graph*) G_Π associated with a P system Π is considered in such a way that there exists an accepting computation of Π if and only if there exists a path between two distinguished nodes in the dependency graph associated with it.
- *Algorithmic technique*: a deterministic algorithm $\mathcal{A}$ working in polynomial time that receives as input a P system Π and an input multiset m of Π is considered in such a manner that algorithm $\mathcal{A}$ reproduces the behaviour of **a single** computation of $\Pi + m$.
- *Simulation technique*: a P system Π' simulating (in an efficient way) a given P system Π is constructed, in such a way that there exists an efficient and injective correspondence between accepting computation of Π' and accepting computation of Π.

4 Recent results

The role of minimal cooperation is studied from the computational complexity point of view. In the case of polarizationless P systems with active membranes, objects evolution rules whose left-hand side consists of a couple of objects are considered. In the case of cell-like P systems with symport/antiport rules, minimal cooperation is expressed by the property of the length of communication rules to be equal to two.

4.1 Polarizationless P systems with active membranes

By using the dependency graph technique it has been shown that families of non-cooperative polarizationless P systems with active membranes which make no use of dissolution rules, can only solve tractable problems in an efficient way [3, 14].

Theorem 1. $\mathbf{P} = \mathbf{PMC}_{\mathcal{DAM}^0(+e,+c,-d,+n)} = \mathbf{PMC}_{\mathcal{SAM}^0(+e,+c,-d,+n)}$.

However, when dissolution rules are allowed in that framework, the situation is quite different. If division rules for non-elementary membranes are permitted then **PSPACE**-complete problems can be solved in polynomial time [1].

Theorem 2. $\mathbf{PSPACE} \subseteq \mathbf{PMC}_{\mathcal{DAM}^0(+e,+c,+d,+n)}$.

Nevertheless, it is an open question what happens if division rules only for elementary membranes are permitted in polarizationless P systems with active membranes and with dissolution rules.

Open question: $\mathbf{PMC}_{\mathcal{DAM}^0(+e,+c,+d,-n)} = \mathbf{P}$? (*Păun's conjecture*).

The computational efficiency of polarizationless P systems with active membranes and membrane division can be reached by considering *bounded minimal cooperation*, even in the case that dissolution rules and division rules for non-elementary membranes are forbidden [15].

Theorem 3. $\mathtt{SAT} \in \mathbf{PMC}_{\mathcal{DAM}^0(bmc,+c,-d,-n)}$.

However, if separation rules are used instead of division rules in polarizationless P systems with active membranes which make use of bounded minimal cooperation then, by using the algorithmic technique, the limitation on the efficiency of these systems has been shown [13].

Theorem 4. $\mathbf{PMC}_{\mathcal{SAM}^0(bmc,+c,-d,+n)} = \mathbf{P}$.

Families of polarizationless P systems with active membranes and membrane separation which use *primary minimal cooperation* in object evolution rules can solve **NP**-complete problems in polynomial time, even in the case that dissolution rules and division rules for non-elementary membranes are forbidden [14].

Theorem 5. $\mathtt{SAT} \in \mathbf{PMC}_{\mathcal{SAM}^0(pmc,+c,-d,-n)}$.

The result of Theorem 3 has been improved in the sense that minimal cooperation in object evolution rules producing only a single object suffices to reach the efficiency of polarizationless P systems with active membranes [16].

Theorem 6. $\mathtt{SAT} \in \mathbf{PMC}_{\mathcal{DAM}^0(mcmp,+c,-d,-n)}$.

By considering the corresponding "counting version" of membrane systems from $\mathcal{DAM}^0(mcmp, +c, -d, -n)$, the previous result has been extended to the counting problem $\#SAT$ [10], which is a well known $\#\mathbf{P}$-complete problem [6].

Bearing in mind that minimal cooperation with minimal production in object evolution rules is a particular case of bounded minimal cooperation, we deduce the following result:

Theorem 7. $\mathbf{PMC}_{\mathcal{SAM}^0(mcmp,+c,-d,+n)} = \mathbf{P}$.

4.2 Cell-like P systems with symport/antiport rules

From the complexity point of view, non-cooperative cell-like P systems with symport/antiport rules and polarizationless P systems with active membranes and without dissolution rules have some similarities. In particular, the dependency graph technique used to prove Theorem 1 can be adapted to show that such non-cooperative systems are not efficient unless $\mathbf{P} = \mathbf{NP}$ holds.

Theorem 8. $\mathbf{P} = \mathbf{PMC}_{\mathcal{CD}_{ne}\mathcal{C}(1)} = \mathbf{PMC}_{\mathcal{CS}_{ne}\mathcal{C}(1)}$.

However, if minimal cooperation in communication rules is considered, then computational efficiency can be reached when membrane division rules are used [12].

Theorem 9. $\texttt{HAM} - \texttt{CYCLE} \in \mathbf{PMC}_{\mathcal{CDC}(2)}$.

Concerning membrane separation rules, if the length of communication rules is at most two then, by using the algorithmic technique, it can be shown that separation rules are not enough to provide polynomial-time solutions to computationally hard problems, assuming that $\mathbf{P} \neq \mathbf{NP}$. Nevertheless, P systems with symport/antiport rules with length at most three and with membrane separation are computationally efficient [5].

Theorem 10. $\mathbf{P} = \mathbf{PMC}_{\mathcal{CSC}(2)}$ *and* $\texttt{SAT} \in \mathbf{PMC}_{\mathcal{CSC}(3)}$.

In [11] a polynomial-time solution for the QSAT problem, which is a well known **PSPACE**-complete problem [2], has been given by means of a family of cell-like P systems with communication rules with length at most three which make use of division rules for non-elementary membranes.

Theorem 11. $\texttt{QSAT} \in \mathbf{PMC}_{\mathcal{CD}_{ne}\mathcal{C}(3)}$.

Next, the role of environment associated with P systems with symport/antiport rules is studied from the complexity point of view. Let us recall that in these systems environment provides an arbitrary large amount of objects at the initial configuration.

First, it has been shown that the role of the environment is not relevant in P systems with symport/antiport rules and membrane division by using simulation technique.

Theorem 12. *For each* $k \geq 1$ *we have* $\mathbf{PMC}_{\widehat{\mathcal{CDC}}(k)} = \mathbf{PMC}_{\mathcal{CDC}(k)}$.

Second, it has been shown that P systems with symport/antiport rules, without environment and with membrane separation can only solve problems in class **P** in polynomial time [5] by using algorithmic technique.

Theorem 13. *For each* $k \geq 1$ *we have* $\mathbf{P} = \mathbf{PMC}_{\widehat{\mathcal{CSC}}(k)}$.

Hence, in this kind of P systems the role of environment is important: $\mathbf{P} = \mathbf{PMC}_{\widehat{\mathcal{CSC}}(3)}$ and $\texttt{SAT} \in \mathbf{PMC}_{\mathcal{CSC}(3)}$.

5 Conclusions

In this work, we present an overview of the results related to the computational efficiency of (minimal) cooperation. Informally speaking, one could say that it is not surprising that in most cases models using cooperation prove to be more powerful than their non-cooperative counterparts, by setting an analogy with the existing gap between context-free and context-sensitive rules in Chomsky hierarchy. This paper investigates the impact of (minimal) cooperation in the framework of cell-like membrane systems. That is, the goal is to restrict the cooperation to its limit, while minimizing the rest of ingredients of the model, in such a way that borderlines of efficiency (expressed in terms of the degree of cooperation) are unveiled. More precisely, the paper focuses on two case studies: allowing some "minimal" degree of cooperation in object evolution rules for polarizationless P systems with active membranes, and reducing symport/antiport rules to "minimal" length for cell-like P systems.

There are many other P system models worth studying, which could probably benefit from the results gathered here by following analogous techniques but performing the corresponding adjustments.

Recently, the computational efficiency of polarizationless P systems with active membranes and *division* rules which incorporates minimal cooperation with minimal production in *communication* rules (instead of in object evolution rules) has been studied. The results obtained are similar to the case of minimal cooperation in object evolution rules but with an important restriction: division rules for elementary and non-elementary membranes have been used. Moreover, it has been shown that efficiency can be reached by using minimal cooperation with minimal production only in *send-in* rules (or only in *send-out* rules).

Two main questions remain to be investigated:

- In order to reach the efficiency, can the use of division rules for non-elementary membranes be avoided?
- What about the efficiency of polarizationless P systems with active membranes and *separation* rules which incorporates minimal cooperation with minimal production in communication rules?

We conclude by presenting some additional open questions.

- Is there a family of systems from $\mathcal{DAM}^0(mcmp, +c, -d, +n)$ (or from $\mathcal{SAM}^0(mcmp, +c, -d, +n)$) providing a polynomial-time solution to the QSAT problem?
- $\texttt{QSAT} \in \mathbf{PMC}_{\mathcal{CD}_{ne}\mathcal{C}(2)}$?
- $\texttt{QSAT} \in \mathbf{PMC}_{\mathcal{CS}_{ne}\mathcal{C}(3)}$?

References

1. A. Alhazov, M.J. Pérez-Jiménez. Uniform solution of QSAT using polarizationless active membranes. In J. Durand-Lose and M. Margenstern (eds.) *Machines, Computations, and Universality. Lecture Notes in Computer Science*, 4664 (2007), 122-133.
2. M.R. Garey, D.S. Johnson. *Computers and Intractability A Guide to the Theory of NP-Completeness*. W.H. Freeman and Company, 1979.
3. M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, F.J. Romero-Campero. On the power of dissolution in P systems with active membranes. In R. Freund, Gh. Păun, Gr. Rozenberg, A. Salomaa (eds.) *Membrane Computing*, 6th International Workshop, WMC 2005, Vienna, Austria, July 18-21, 2005, Revised Selected and Invited Papers. *Lecture Notes in Computer Science*, 3850 (2006), 224-240.
4. A. Leporati, C. Ferretti, G. Mauri, M.J. Pérez-Jiménez, C. Zandron. Complexity aspects of polarizationless membrane systems. *Natural Computing*, 8, 4 (2009), 703-717.
5. L.F. Macías-Ramos, B. Song, L. Valencia-Cabrera, L. Pan, M.J. Pérez-Jiménez. Membrane fission: A computational complexity perspective. *Complexity*, 21, 6 (2016), 321-334.
6. C.H. Papadimitriou. *Computational Complexity*. Addison-Wesley Publ. Company, USA, 1994.
7. A. Păun, Gh. Păun. The power of communication: P systems with symport/antiport, *New Generation Computing*, **20**, 3 (2002), 295–305.
8. Gh. Păun. Computing with membranes, *Journal of Computer and Systems Science*, **61**, 1 (2000), 108–143.
9. Gh. Păun. P systems with active membranes: attacking **NP**–complete problems, *Journal of Automata, Languages and Combinatorics*, **6** (2001), 75–90.
10. A. Riscos-Núñez, M.J. Pérez-Jiménez. Counting membrane systems. Attacking #**P**-complete problems. **Submitted**, 2016.
11. B. Song, M.J. Pérez-Jiménez, L. Pan. Efficient solutions to hard computational problems by P systems with symport/antiport rules and membrane division. *Biosystems*, 130 (2015), 51-58.
12. L. Valencia-Cabrera, B. Song, L.F. Macas, L. Pan, A. Riscos-Núñez, M.J. Pérez-Jiménez. Minimal cooperation in P systems with symport/antiport: A complexity approach. In L.F. Macas, Gh. Păun, A. Riscos, L. Valencia (eds) *Proceedings of the Thirteenth Brainstorming Week on Membrane Computing*, 2-6 February, 2015, Sevilla, Spain, pp. 301-323.
13. L. Valencia-Cabrera, D. Orellana-Martín, A. Riscos-Núñez, M.J. Pérez-Jiménez. Minimal cooperation in polarizationless P systems with active membranes. In C. Graciani, Gh. Paun, D. Orellana-Martn, A. Riscos-Núñez, L. Valencia-Cabrera (eds.) *Proceedings of the Fourteenth Brainstorming Week on Membrane Computing*, 1-5 February, 2016, Sevilla, Spain, pp. 327-356.
14. L. Valencia-Cabrera, D. Orellana-Martín, M.A. Martínez-del-Amor, M.J. Pérez-Jiménez, A. Riscos-Núñez. Computational efficiency of minimal cooperation and distribution in polarizationless P systems with active membranes, *Fundamenta Informaticae*, submitted 2016.
15. L. Valencia-Cabrera, D. Orellana-Martín, M.A. Martínez-del-Amor, M.J. Pérez-Jiménez, A. Riscos-Núñez. Polarizationless P systems with active membranes: Computational complexity aspects. *Journal of Automata, Languages and Combinatorics*, 21, 1-2 (2016), 107-123.

16. L. Valencia-Cabrera, D. Orellana-Martín, A. Riscos-Núñez, M.J. Pérez-Jiménez. Reaching efficiency through complicity in membrane systems: dissolution, polarization and cooperation. *Theoretical Computer Science*, submitted 2016.
17. Zandron, C., Ferretti, C., Mauri, G.: Solving NPcomplete problems using P systems with active membranes. In Antoniou, I., Calude, C.S., Dinneen, M.J. (eds.) *Unconventional Models of Computation*, pp. 289301. Springer, Heidelberg (2000)

Polymorphic P Systems: A Survey

Artiom Alhazov[1], Rudolf Freund[2], and Sergiu Ivanov[3,4]

[1] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Str. Academiei 5, Chişinău, MD 2028, Moldova
artiom@math.md
[2] Faculty of Informatics, TU Wien
Favoritenstraße 9–11, 1040 Vienna, Austria
rudi@emcc.at
[3] LACL, Université Paris Est – Créteil Val de Marne
61, av. Général de Gaulle, 94010, Créteil, France
sergiu.ivanov@u-pec.fr
[4] TIMC-IMAG/DyCTiM, Faculty of Medicine of Grenoble
5 avenue du Grand Sablon, 38700, La Tronche, France
sergiu.ivanov@univ-grenoble-alpes.fr

Summary. Polymorphic P systems are a variant of P systems – a multiset-rewriting-based model of computing inspired from the structure and the functioning of the living cell. In polymorphic P systems, rules are not statically defined, but instead are dynamically inferred from the contents of specially designated pairs of membranes. Besides enabling dynamic modifications of the form of the available rules, polymorphism allows for embedding rules into the sides of other rules. In the present article, we compile the results and the conclusions published on polymorphic P systems since 2011 (when they were introduced) and give an extensive list of 11 open questions.

1 Introduction

Membrane computing is a research field originally founded by Gheorghe Păun in 1998 [15]. Membrane computing focuses on membrane systems (also known as P systems) which is a model of computing based on the abstract notion of a membrane. Formally, a membrane is treated as a container delimiting a region; a region may contain objects which are acted upon by the rewriting rules associated with the membrane. Quite often, the objects are plain symbols coming from a finite alphabet, but P systems operating on more complex objects (e.g., strings) sometimes are considered, too. A comprehensive overview of different flavours of membrane systems and their expressive power is given in the 2010 handbook [17]. For a state of the art snapshot of the domain, we refer the reader to the P systems website [20], as well as to the bulletin of the International Membrane Computing Society [19].

As indicated by its name, membrane computing draws inspiration from the structure and functioning of the living cell [16]. Indeed, one can see the cell as a hierarchical arrangement of containers (rooted at the cellular wall) performing biochemical processing. A great variety of abstract representations of substance flow within the membranes as well as across them has been proposed: communication rules, symport/antiport rules, dissolution rules, etc. [17]. On the other hand, P systems have a distinctive computer science touch: objects evolve in discrete, synchronous steps determined by a global clock, the rules defining the behaviour are typically (multiset) rewriting rules. Although computer science and cell biology are arguably distinct domains, they do have one feature in common: the description of the "program" can be modified by the organism itself. In cells, this paradigm (sometimes referred to as "program is data") is represented by mechanisms such as reverse transcription [8], while in computer science this is embodied by the fact that the program is stored in the memory alongside the data it manipulates. It is therefore only natural that there have been quite a number of attempts to represent the "program is data" paradigm in membrane computing. Examples of such possible presentations include generalised P systems [9], rule creation [5], activators [1], inhibiting/deinhibiting rules [7], symport/antiport of rules [6], etc. One notable characteristic of the majority of these cited approaches is that, even though the set of rules associated with a given membrane may evolve, the rules appearing in this set may only be chosen from a finite, statically fixed, collection.

Polymorphic P systems are an implementation of the "program is data" paradigm for membrane systems which does not limit the set of available rules by a finite cardinality and which allows direct tampering with the form of the rules. In polymorphic P systems, rules are not statically defined, but are instead dynamically inferred from the contents of specially designated pairs of membranes. One member of such a pair defines the multiset representing the left-hand side of the rule; the other member defines the right-hand side.

Dynamic definition of rules via pairs of membranes has numerous interesting consequences:

- *natural rule dynamics*: since rules are defined by the contents of membranes, sending an object across a membrane and changing a rule are exactly the same operation;
- *nested rules*: the definition does not restrict the membranes defining the sides of a rule to occupying some particular slots of the membrane hierarchy; in particular, we are allowed to place rules *inside* the membranes defining *other rules*;
- *inexpensive simulation*: there is almost no additional overhead to naively simulating polymorphic P systems on a conventional computer: instead of only moving symbols from membranes to membranes, the simulator should be made capable of also moving symbols into and out of the collections representing the sides of rules.

Polymorphic P systems can thus modify their own "program" in a quite natural way: by moving around symbols. Furthermore, nesting rules quickly gives rise to highly non-trivial behaviour. This, on the one hand, means that polymorphism may be used to solve complex problems (like recognising the factorial language) with relative ease. On the other hand, polymorphic rules may be in quite complex interdependence, which often makes reasoning about their behaviour a quite difficult task.

Polymorphic P systems were originally introduced in the 2011 paper [4]; the subsequent work [11] analyses the power of the minimal variant of polymorphic P systems, in which most of the usual ingredients (cooperative rules, communication rules, etc.) are not allowed. The paper [3] considers general states in P systems; this framework generalises, among other things, the idea of polymorphism.

Unfortunately, apart from the three cited works, polymorphic P systems have received relatively little attention. The goal of the present article is to give an overview of the existing (scanty) results on polymorphic P systems and to list the numerous research directions that are still open.

This article is structured as follows. Section 2 recalls some basic notions from the theory of formal languages and defines conventional P systems as well as polymorphic P systems. The immediately following Section 3 recalls how polymorphism and target indications can be used to achieve fast growth rates and to generate complex languages. The following Section 4 recalls several more specific terms which capture some details of polymorphism more precisely. These terms are used in Section 5 to describe the power of non-cooperative polymorphism without target indications. Section 6 briefly recalls the concept of state in P systems as introduced in [3] and shows how states generalise polymorphism. Finally, Section 7 gives an overview of some of the possible directions of research on polymorphic P systems listing 11 major open questions.

2 Preliminaries

An *alphabet* V is a finite set, and a *multiset* over V is any function $w : V \to \mathbb{N}$; $w(a)$ is the *multiplicity* of a in w. A multiset w is often represented by one of the strings containing exactly $w(a)$ copies of each symbol $a \in V$. The *support* $supp(w)$ of the multiset w is the set of elements appearing in w a non-zero number of times: $supp(w) = \{a \in V \mid w(a) > 0\}$. The set of all multisets over the alphabet V is denoted by V°. The empty multiset is denoted by $\mathbf{0}$ ($supp(\mathbf{0}) = \emptyset$). The *projection* (restriction) of w over a subalphabet $V' \subseteq V$ is the multiset $w|_{V'}$ defined as follows:

$$w|_{V'}(a) = \begin{cases} w(a), & a \in V'; \\ 0, & a \in V \setminus V'. \end{cases}$$

We denote the family of recursively enumerable sets of non-negative integers by NRE.

For further introduction to the theory of formal languages and computability, we refer the reader to [15, 17].

2.1 P Systems: The Basic Model

A *P system* is a construct

$$\Pi = (O, T, \mu, w_1, \ldots, w_n, R_1, \ldots R_n, h_i, h_o),$$

where O is the alphabet of objects, $T \subseteq O$ is the alphabet of terminal objects, μ is the membrane structure injectively labelled by the numbers from $\{1, \ldots, n\}$ and usually given by a sequence of correctly nested brackets, w_i are the multisets giving the initial contents of each membrane i ($1 \leq i \leq n$), R_i is the finite set of rules associated with membrane i ($1 \leq i \leq n$), and h_i and h_o are the labels of the input and the output membranes, respectively ($1 \leq h_i \leq n$, $1 \leq h_o \leq n$).

The *depth* of the membrane structure μ is the height of this structure seen as a tree. Thus, a membrane system which only has one membrane is of depth 1; a membrane system with two nested membranes is of depth 2, etc.

Quite often the rules associated with membranes are multiset rewriting rules (or special cases of such rules). Multiset rewriting rules have the form $u \to v$, with $u \in O^o \backslash \{\mathbf{0}\}$ and $v \in O^o$. If $|u| = 1$, the rule $u \to v$ is called *non-cooperative*; otherwise it is called *cooperative*. In *communication P systems*, rules are additionally allowed to send symbols to the neighbouring membranes. In this case, for rules in R_i, $v \in O \times Tar_i$, where Tar_i contains the symbols *out* (corresponding to sending the symbol to the parent membrane), *here* (indicating that the symbol should be kept in membrane i), and in_h (indicating that the symbol should be sent into the child membrane h of membrane i).

In P systems, rules are often applied in a maximally parallel way: in one derivation step, only a non-extendable multiset of rules can be applied. The rules are not allowed to consume the same instance of a symbol twice, which creates competition for objects and may lead to the P system choosing non-determinstically between the maximal collections of rules applicable in one step.

A computation of a P system is traditionally considered to be a sequence of configurations it can successively pass through, stopping at the halting configuration. A halting configuration is a configuration in which no rule can be applied any more, in any membrane. The result of a computation of a P system Π as defined above is the contents of the output membrane h_o projected over the terminal alphabet T.

While maximal parallelism and halting by inapplicability are staple ingredients, various other derivation modes and halting conditions have been considered for P systems [17].

2.2 Polymorphic P Systems

A *polymorphic P system* is the following construct:

$$\Pi = (O, T, \mu, w_s, \langle w_{1L}, w_{1R}\rangle, \ldots, \langle w_{nL}, w_{nR}\rangle, h_i, h_o)\,,$$

where O is a finite alphabet of objects, T is the subalphabet of terminal objects, μ is a tree structure consisting of $2n+1$ membranes, w_s is the multiset giving the contents of the skin membrane, $\langle w_{iL}, w_{iR}\rangle$ are pairs of multisets giving the contents of membranes iL and iR ($1 \leq i \leq n$), and h_i and h_o are the labels of the input and the output membranes, respectively, with $h_i \in H$ and $h_o \in H \cup \{0\}$, where 0 denotes the environment. We require that, for every $1 \leq i \leq n$, the membranes iL and iR have the same containing (parent) membrane. The *depth* of (the membrane structure of) Π is defined as for conventional P systems: it is the height of μ seen as a tree.

The rules of Π are not statically given in its description and are instead dynamically inferred for each configuration based on the contents of the pairs of membranes iL and iR. Thus, if in a configuration of the system these membranes contain the multisets u and v, respectively, then, in the next step, their parent membrane h will evolve as if it had the multiset rewriting rule $u \to v$ associated with it. If, however, in some configuration, iL is empty, we consider the rule defined by the pair $\langle iL, iR\rangle$ to be *disabled*, i.e. no rule will be inferred from the contents of iL and iR.

Polymorphic P systems evolve by applying the dynamically inferred rules in a maximally parallel way. A *computation* of a polymorphic P system Π is a finite sequence of configurations Π may successively visit, ending in the halting configuration in which no rules can be applied any more in any membrane. Like for other classes of P systems, the output of Π is the contents of the output membrane h_o projected onto the terminal alphabet T.

It follows from the definition of polymorphic P systems that the total set of rules which may be applied during a derivation of such a system Π is unbounded. Indeed, Π may ensure unbounded growth of the contents of a right-hand-side membrane iR. Remark however that the total number of rules available in an evolution step is *statically bounded* by the number of membranes.

Not too many additional ingredients have been considered for polymorphic P systems up to now. The only one actually discussed in [4] is target indications. Target indications for polymorphic P systems were defined by adding the following mapping to the definition of a system:

$$\varphi : \{iR \mid 1 \leq i \leq n\} \to \{in_h \mid h \in H\} \cup \{here, out\}.$$

This mapping assigns a target indication to every right-hand-side membrane. The target indication is interpreted as in the case of conventional P systems.

Note that, by this definition, φ is allowed to assign any target indication in_h to any right-hand-side membrane, which would effectively allow ignoring the hierarchical membrane structure for communication purposes. Nevertheless, in all the examples considered in the present article (and in the other works published to date), targets are always given with respect to the membrane structure of the system.

In the present article we follow the tradition of not explicitly drawing membranes defining invariable rules in graphical illustrations [4, 11]. Thus, if the contents w_{iL} and w_{iR} never change, we graphically represent the defined rule by the usual notation $w_{iL} \to w_{iR}$.

3 The Power of Polymorphism

In this section we showcase the power of polymorphic P systems by giving concrete constructions achieving complex behaviour while retaining low levels of descriptional complexity. We use the term *growth rate* to refer to the formula describing the size of the multiset contained in a membrane of a P system (or the union of the contents of all membranes) as a function of time (number of steps).

3.1 Polymorphism *without* Target Indications

We start by showing how superexponential growth (in time) can easily be achieved using polymorphism alone without any additional ingredients.

Example 1 (superexponential growth [11]). Consider the following polymorphic P system:

$$\Pi_1 = (\{a\}, \{a\}, \mu, a, \langle a, a\rangle, \langle a, aa\rangle, s), \text{ where}$$
$$\mu = [\,[\,]_{1L}[\,[\,]_{2L}[\,]_{2R}]_{1R}]_s.$$

Π_1 has superexponential growth rate. A graphical illustration of Π_1 is given in Figure 1.

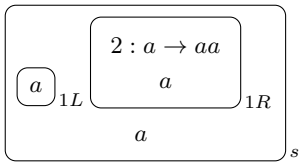


Fig. 1. The polymorphic P system Π_1 with superexponential growth

In the initial configuration, the membranes $1L$ and $1R$ define the rule $a \to a$ in the skin membrane s. Rule 2 in membrane $1R$ is formally represented by the pair of membranes $\langle 2L, 2R\rangle$, but graphically depicted as $a \to aa$, because the contents of $1R$ and $1R$ never change.

In the first derivation step, rule 1 ($a \to a$) is applied in the skin, leaving the contents of the membrane intact, and rule 2 ($a \to aa$) is applied in membrane $1R$, doubling the number of a's; therefore, after the first step, rule 1 will be of the form $a \to aa$. In the second step of the derivation, rule 1 will transform the multiset a in the skin into aa, and rule 2 will double the contents of the right-hand-side

membrane $1R$ once again, thus transforming rule 1 into $a \to a^4$. Consequently, in the third derivation step, rule 1 will *quadruple* the number of a's in the skin.

In general, after k derivation steps, the contents of the right-hand-side membrane $1R$ will be 2^k, and rule 1 will have the form $a \to a^{2^k}$. The number of a's in the skin will be given by the product $1 \cdot 2 \cdot 4 \cdot \ldots \cdot 2^{k-1}$ or, equivalently, by the following formula:

$$2^0 \cdot 2^1 \cdot 2^2 \cdot \ldots \cdot 2^{k-1} = 2^{1+2+\ldots+k-1} = 2^{\frac{k(k-1)}{2}}.$$

The rate of growth of the contents of the skin membrane therefore is superexponential.

Besides showing off impressive growth rates, the previous example also helps to fix an upper bound on how fast a polymorphic P system can grow.

Proposition 1. *[11, Theorem 4],[4, page 7] The growth rate of the contents of any membrane of a polymorphic P system Π of depth d (without target indications) is bounded by the polynomial $Ic^{p(n)}$, where:*

- *I is the total number of objects in the initial configuration of Π,*
- *c is the maximal size of a right-hand side of an invariable rule of Π,*
- *$p(n)$ is a polynomial of degree at most $d-1$.*

Indeed, a polymorphic P system Π achieves superexponential growth by nesting right-hand-side membranes. Each level of nesting contributes a factor to the exponent of the formula expressing the growth rate. Since the depth of the membrane structure is statically bounded, the degree of the polynomial in the exponent is bounded as well, and its form is determined by the initial contents of the membranes.

Even though the P system Π_1 of Example 1 does achieve superexponential growth rates, it never halts and the language it generates is therefore empty. More effort is needed to actually generate the set of the powers of two. A first idea would be embedding the pair of rules $a \to a$ and $a \to \lambda$ into $1L$, thereby making it possible to disable the multiplying rule 1 at an arbitrary moment. This will effectively stop multiplication, but $1R$ will keep evolving, since rule 2 is always applicable. If, additionally, we decide to arbitrarily disable rule 2, it may become disabled before rule 1 (before the contents of $1L$ is erased), which will lead to several multiplications of the contents of the skin by a constant factor.

The construction from the following example illustrates how to properly stop the multiplications. The main idea for achieving rule inapplicability consists in rewriting all instances of a into a different symbol.

Example 2 (powers of two [11]). Consider the following polymorphic P system:

$$\Pi_2 = (\{a,b\}, \{b\}, \mu, a, \langle a,a\rangle, \langle a,aa\rangle, \langle a,a\rangle, \langle a,a\rangle, \langle a,b\rangle, s), \text{ where}$$
$$\mu = [\,[\,]_{1L}[\,[\,]_{2L}[\,[\,]_{3L}[\,[\,]_{4L}[\,]_{4R}[\,]_{5L}[\,]_{5R}]_{3R}]_{2R}]_{1R}]_s.$$

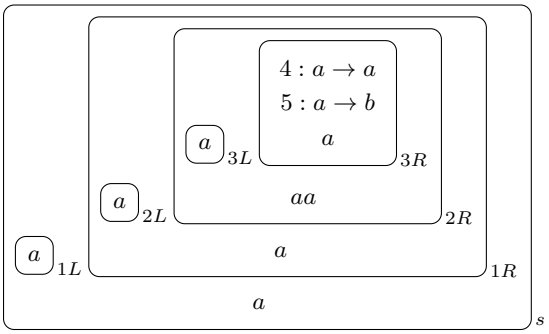


Fig. 2. The P system Π_2 generating the number language $\left\{2^{\frac{n(n-1)}{2}} \;\middle|\; n \in \mathbb{N}, n > 3\right\}$

The number language generated by Π_2 is $N(\Pi_2) = \left\{2^{\frac{n(n-1)}{2}} \;\middle|\; n \in \mathbb{N}, n > 3\right\}$. A graphical representation of Π_2 is given in Figure 2.

Membranes s, $1L$, $1R$, $2L$, and $2R$ of Π_2 have the same initial contents as the corresponding membranes of Π_1. The role of the additional infrastructure in membrane $2R$ of Π_2 is to ensure proper halting.

Π_2 operates in two phases. During the first phase, rule 4 is always applied in membrane $3R$, keeping rule 3 in the form $a \to a$, which does not change the right-hand-side membrane $2R$. In the first phase Π_2 therefore does the same thing as Π_1. The second phase of a computation of Π_2 starts with the application of rule 5 in $3R$, which changes rule 3 into $a \to b$. This, in the next step, changes rule 2 into $a \to bb$. In the following step, all the a's of $1R$ are rewritten into b's, which finally leads to rewriting all the a's in the skin into b's. The second phase takes four steps and rewrites all a's into b's, while keeping the multiplications going. This explains the constraint $n > 3$ in the definition of the number language $N(\Pi_2)$ generated by Π_2: $N(\Pi_2) = \left\{2^{\frac{n(n-1)}{2}} \;\middle|\; n \in \mathbb{N}, n > 3\right\}$.

3.2 Polymorphism *with* Target Indications

Adding target indications to polymorphic P systems allows a more fine-grained control of the evolution. For example, it is relatively easy to generate the factorial number language using only four rules *without* nested right-hand-side membranes.

Example 3 (factorial [4]). Consider the following polymorphic P system with target indications:

$$\Pi_3 = (\{a,b,c,d\}, \{b\}, \mu, abd, \langle a,a\rangle, \langle b,bd\rangle, \langle b,c\rangle, \langle d,a\rangle, \varphi, s), \text{ where}$$
$$\mu = [\,[\,]_{1L}[\,]_{1R}[\,]_{2L}[\,]_{2R}[\,]_{3L}[\,]_{3R}[\,]_{4L}[\,]_{4R}]_s,$$
$$\varphi(1R) = \varphi(2R) = here, \;\; \varphi(3R) = in_{1L}, \;\; \varphi(4R) = in_{1R}.$$

Π_3 generates the number language $N(\Pi_3) = \{n! \mid n \in \mathbb{N}\}$. A graphical representation of Π_3 is given in Figure 3. For rules 3 and 4 whose right-hand-side membranes

$3R$ and $4R$ are attributed targets different from *here*, we use the standard P system notation for targets as a shortcut; for example (c, in_{1L}) stands for "produce a c and send it into membrane $1L$".

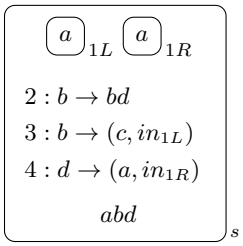


Fig. 3. The polymorphic P system Π_3 generating the factorial number language

Π_3 calculates the factorial in a very straightforward fashion: by successive multiplications. The whole process is controlled by the symbol b which chooses between rules 2 and 3. If rule 2 is applied, b is sustained and an instance of d is produced, which, in the next step, adds an a to the right-hand side of rule 1 (by applying rule 4). Thus, while b chooses rule 2, the right-hand side of rule 1 grows by one a symbol a step, which leads to the multiplication of the contents of the skin by successive natural numbers, starting from 1 in the first step.

Π_3 halts by applying rule 3, which erases the control symbol b and renders rule 1 inapplicable by adding an extra dummy c to $1L$.

Remark that the complexity of computing $n!$ is linear in n. More precisely, Π_3 needs n steps to generate $n!$ ($n \geq 1$). Indeed, rule 1 is applied in every step effecting multiplications by successive natural numbers, and, when rule 1 is inapplicable, we are sure that b has been erased by rule 3 and d by rule 4, which renders all rules inapplicable.

Note that, while this example takes over the example shown at the top of page 9 of [4], there is a minor difference: in Example 3 we add an instance of d to the skin from the very beginning. This avoids an extra "warm-up" step in which the first instance of d is produced by rule 2.

Finally, we would like to point out that Π_3 uses "half" of the appearance checking functionality provided by cooperativity: rule 1 becomes non-cooperative only in the halting configuration and is never applied. As we will later see, rule 1 of Π_3 is a *weakly* non-cooperative rule.

Besides generation of the factorial language, the article [4] also shows how to generate the double exponential language $\left\{2^{2^n} \mid n \in \mathbb{N}\right\}$ in *linear* time using polymorphism coupled with target indications. This language of 2^n-th powers of 2 is obtained by iterated squaring of the number of symbols present in the skin of the system.

Example 4 (double exponential [4]). Consider the following polymorphic P system with target indications:

$$\begin{aligned}
\Pi_4 &= (\{a, b, a', b', c\}, \{a\}, \mu, b^2, \langle a, \lambda\rangle, \langle a, a\rangle, \langle a, c\rangle,\\
&\qquad \langle b, \lambda\rangle, \langle b, a'b'\rangle, \langle a', a\rangle, \langle b', b\rangle, \varphi, s), \text{ where}\\
\mu &= [\ \ [\,[\,]_{2L}[\,]_{2R}[\,]_{3L}]_{1L}\ \ [\,[\,]_{4L}[\,]_{4R}]_{1R}\\
&\qquad [\,]_{5L}[\,]_{5R}\ \ [\,]_{6L}[\,]_{6R}\ \ [\,]_{7L}[\,]_{7R}\ \ ]_s,\\
\varphi(h) &= \begin{cases} here, & h \in \{iR \mid 1 \le i \le 6\},\\ in_{1R}, & h = 7R. \end{cases}
\end{aligned}$$

This P system generates the number language $N(\Pi_4) = \left\{2^{2^n} \;\middle|\; n \in \mathbb{N}\right\}$. A graphical presentation of Π_4 is given in Figure 4.

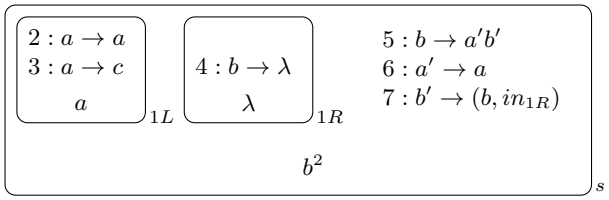


Fig. 4. The polymorphic P system Π_4 generating the language $\left\{2^{2^n} \;\middle|\; n \in \mathbb{N}\right\}$

A computation of Π_4 is structured into three-step phases. A phase starts with an application of rule 5, which splits each b into a b' and an a'. In the second step of a phase, rules 6 and 7 are applied. Rule 6 rewrites every a' into an a; rule 7 removes primes from b' and moves all of the resulting instances of b into $1R$. In the third step of a phase, rule 1 may become applicable, rewriting *each* a in the skin into as many instances of b as there were initially at the start of the phase.

The first two steps of a phase ensure that there are as many instances of b in $1R$ as there are instances of a in the skin. An application of rule 1 therefore *squares* the number of symbols in the skin. In total, if the skin contains the multiset b^k at the beginning of a phase, it will contain the multiset b^{k^2} at the end of the phase.

To allow Π_4 to chain multiple such phases, rule 4 discards all b's immediately after rule 1 is applied at the end of a phase. Remark that the only time when the skin contains a's and when rule 1 is applicable is when $1R$ also contains some instances of b; rule 1 is thus never applied to erase instances of a.

Finally, rules 2 and 3 in $1L$ control the liveliness of Π_4. While the system applies rule 2 keeping an a in $1L$, the other rules may keep on squaring the contents of the skin. When Π_4 chooses to apply rule 3, rule 1 is rendered forever inapplicable, because c never appears in the skin. Rules 5, 6, and 7 may split b's into b' and a, sending some b's into $1R$, but the skin does not contain rules for processing a

any more. Rule 4 will drop the last packet of b's sent in by rule 7 and will stop evolving as well, thus leading to the total halting of Π_4.

The claim that Π_4 produces 2^{2^n} instances of a in a time linear in n follows from the fact that it takes Π_4 only three steps to square the number represented by the symbols in the skin.

Notice that, independently of the moment at which rule 3 is applied to disable rule 1, the first two steps of a phase will always take place. In the third step, however, only rule 4 in $1R$ will be applied, discarding the last instances of b in the system. We can thus give a more accurate value for the time complexity of the work of Π_4: this system takes $3(n+1)$ steps to calculate 2^{2^n}.

Comparing the constructions from the previous two subsections leads to an interesting conclusion: Proposition 1 sets an upper bound on the power of polymorphism *without* target indications which is surpassed by Π_4 *using* target indications. Thus, adding target indications to polymorphic P systems strictly increases their expressive power. This result is formally summarised in Corollary 2 stated in Section 5.

The original article [4] introducing polymorphic P systems gives further examples of constructions with intricate behaviour, like generating the language $\{n! \cdot n^k \mid n, k \in \mathbb{N}\}$ [4, Example 2], deterministically computing the function $n \to 2^{2^n}$ [4, Example 5], deciding the factorial language [4, Example 6], etc. We refer the reader to the cited article for details.

4 Some Formal Aspects of Polymorphism

Now that we have seen several examples showing off the power of polymorphism, we will recall some definitions from [11, Subsection 2.2] which will allow us to describe the expressivity of polymorphic P systems using a more fine-grained language.

Definition 1 (strong non-cooperativity [11]). *Rule i (defined by the pair of membranes $\langle iL, iR\rangle$) of a polymorphic P system Π is said to be* strongly non-cooperative *if, in any evolution, iL contains at most one symbol.*

Definition 2 (weak non-cooperativity [11]). *Rule i (defined by the pair of membranes $\langle iL, iR\rangle$) of a polymorphic P system Π is said to be* weakly non-cooperative *if, whenever it is applied, its left-hand-side membrane iL contains exactly one symbol.*

Weak non-cooperativity therefore allows a rule to have more than one symbol in the left-hand side and only requires that this rule to be not applicable in such situations. For example, all rules of Examples 1, 2, and 4 are strongly non-cooperative, while rule 1 of Example 3 is weakly non-cooperative, because its left-hand side is of the form ac in the halting derivation and because it is never applied after c is added to $1L$.

Definition 3 (left and right polymorphism [11]). *A rule i of a polymorphic P system Π is called* left-polymorphic *if only its left-hand side may change. Rule i is called* right-polymorphic *if only its right-hand side may change. If all rules of Π are left-polymorphic (respectively, right-polymorphic), then Π is called left-polymorphic (respectively, right-polymorphic).*

The effective implication of the definition of left-polymorphic (respectively, right-polymorphic) P systems is that no right-hand-side (respectively, left-hand-side) membrane is allowed to contain any non-trivial rules. This condition is necessary, but not sufficient in case additional ingredients like target indications are considered. Thus, P systems Π_1 and Π_2 from Examples 1 and 2 are right-polymorphic, while P systems Π_3 and Π_4 from Examples 3 and 4 are neither left- nor right-polymorphic.

Finally, we recall the notations for classes of polymorphic P systems and for families of languages generated by them. The output of a P system is a multiset over the terminal alphabet, which can be treated as a vector; the *vector language* generated by a P system Π is denoted by $Ps(\Pi)$. If we only consider the total size of the output multiset, we are discussing the *number language* generated by Π, denoted by $N(\Pi)$.

To denote the family of polymorphic P systems with at most k membranes, with rule disabling (by emptying left-hand-side membranes), and with strongly non-cooperative rules, we will use the notation

$$OP_k(polym_{+d}(ncoo_s)).$$

If no bound is specified on the number of membranes, k is replaced by $*$ or is omitted. If left- or right- polymorphism is allowed only, *polym* is replaced by *lpolym* or *rpolym*, respectively. If rule disabling is not allowed, $+d$ is replaced by $-d$. If weakly non-cooperative rules are allowed, $ncoo_s$ is replaced by $ncoo_w$. If we consider polymorphic P systems of depth limited by a constant d, we add it as a superscript to OP, as follows:

$$OP_k^d(polym_{+d}(ncoo_s)).$$

Finally, if we consider polymorphic P systems with target indications, we add the symbol *tar* to the notation:

$$OP_k(polym_{+d}(ncoo_s, tar)).$$

5 The Computing Power of the Minimal Variant

In this section, we will briefly recall the results of the analysis from [11] which focuses on the polymorphic P systems with (strongly and weakly) non-cooperative rules without any additional ingredients. We explicitly remark that, while the results in [11] are formulated for the generation of *number* languages, many of the arguments actually prove the corresponding (stronger) results for the generation of *vector* languages.

5.1 Strong and Weak Cooperativity

One of the first results shown in [11] concerns the opposition between strong and weak non-cooperativity. It turns out that, when no additional ingredients are allowed, relying on *weak* non-cooperativity does not increase the computing power of polymorphic P systems.

Theorem 1 (strong and weak non-cooperativity [11, Theorem 2]). $PsOP_*(polym_{+d}(ncoo_w)) = PsOP_*(polym_{+d}(ncoo_s))$.

The proof relies on the fact that, whatever the kind of non-cooperativity, the trajectories of different symbols cannot depend on one another. This implies that the time intervals separating two configurations in which a left-hand-side membrane contains exactly one symbol form a regular set, and therefore the effect of whatever happens in such a membrane can be faithfully simulated by a finite state control implemented on single instances of symbols.

This result is rather interesting since weak non-cooperativity is a dynamic condition quite difficult to check. It also allows the following strong statement about the left-hand-side membranes of any weakly non-cooperative polymorphic P system.

Corollary 1 (shallow left-hand sides [11, Corollary 1]). *Given a polymorphic P system $\Pi \in OP_*(polym_{+d}(ncoo))$, it is possible to construct another polymorphic P system Π' such that $N(\Pi') = N(\Pi)$ and all left-hand-side membranes of Π' contain invariable rules.*

Moreover, Theorem 1 allows a relatively easy proof of the fact that disallowing explicit disabling of rules by erasing their left-hand sides does not contribute to the expressive power.

Proposition 2 (rule disabling [11, Proposition 1]). $PsOP_*(polym_{-d}(ncoo_w)) = PsOP_*(polym_{+d}(ncoo_w))$.

The proof is based on the fact that disabling a rule by erasing its left-hand side is equivalent to just replacing the left-hand side by a dummy symbol which never appears in the containing membrane.

Given that strong and weak non-cooperativity conditions are equivalent when no additional ingredients are allowed, and that disabling of rules can be simulated without emptying left-hand-side membranes, we will simply write *polym* instead of $polym_{+d}$ or $polym_{-d}$, and *ncoo* instead of $ncoo_w$ or $ncoo_s$.

5.2 Target Indications

Adding target indications brings a lot of power to polymorphic P system. The following corollary formally states the conclusion formulated after Example 4.

Corollary 2 (targets increase power).
$\{2^{2^n} \mid n \in \mathbb{N}\} \in NOP_*(polym_{+d}(ncoo_w, tar)) \setminus NOP_*(polym_{+d}(ncoo_w))$.

Proof. This statement follows from the limit on the growth rate for polymorphic P systems without target indications established in Proposition 1 and from the existence of Π_4 (Example 4) which has an even faster growth rate.

Interestingly, it looks like adding target indications actually makes a difference between strong and weak non-cooperativity, as it seems to be difficult to achieve the synchronisation needed to generate the factorial language without relying on weak non-cooperativity.

Conjecture 1 (synchronisation by weak non-cooperativity [4, page 8]).
$\{n! \mid n \in \mathbb{N}\} \notin NOP_*(polym_{+d}(ncoo_s, tar))$.

(Recall that Example 3 gives a *weakly* non-cooperative polymorphic P system with target indications which generates the factorial language.)

5.3 Left, Right, and General Polymorphism

We have seen up to now that polymorphic P systems can achieve impressive growth rates by dynamically varying the right-hand sides of their rules. It turns out that, even if we prohibit right-hand sides from changing, (left-) polymorphic P systems are still more powerful than the conventional communication P systems.

Proposition 3 (power of left polymorphism [11, Proposition 2]).
$\{2^n \mid n \in \mathbb{N}\} \in NOP_*(lpolym(ncoo))$.

Figure 5 shows how to generate the language of the powers of two using left-polymorphic rules. Successive doubling of the contents of the skin is effected by rule 1 which is maintained operational by applications of rule 2 in $1L$. The system halts by applying rule 3.

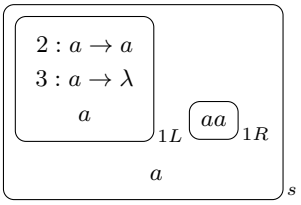


Fig. 5. A left-polymorphic P system generating $\{2^n \mid n \in \mathbb{N}\}$

On the other hand, a very similar number language can be generated by right-polymorphic P systems using the same approach as shown in Example 2: rewriting all the symbols in right-hand-side membranes to "output symbols", thereby rendering the multiplication rule (and all other rules) inapplicable.

Proposition 4 (power of right polymorphism [11, Proposition 4]).
$\{2^n \mid n \in \mathbb{N}, n > 2\} \in NOP_*(rpolym(ncoo))$.

A right-polymorphic P system generating this number language is shown in Figure 6.

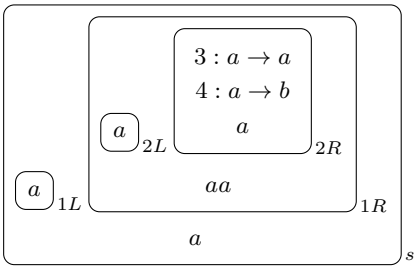


Fig. 6. A right-polymorphic P system generating $\{2^n \mid n \in \mathbb{N}, n > 2\}$

Finally, non-cooperative polymorphic P systems without ingredients are not computationally complete, as they cannot generate the factorial number language.

Theorem 2 (no factorial [4, Theorem 3]).
$\{n! \mid n \in \mathbb{N}\} \notin NOP_*(polym(ncoo))$.

The proof essentially states that, without any additional ingredients, synchronising different membranes is impossible, which means that there is no way of preventing the situation in which the "multiplication" rules stop varying and in which the contents of the output membrane is multiplied by the *same* factor an arbitrary number of times.

5.4 An Infinite Hierarchy

The result concluding the analysis carried out in [11] of polymorphic P systems with non-cooperative rules and no ingredients concerns the relationship between the computational power of such systems and the *depth* of their membrane structures.

Theorem 3 (bounded power [11, Theorem 4]).
$\left\{2^{\binom{n}{d-2}} \;\middle|\; n \in \mathbb{N}, n > d\right\} \notin NOP_*^d(polym(ncoo))$, $d > 1$.

(We remark that Theorem 4 in [11] actually states this result for the language $\left\{2^{\binom{n}{d-1}} \;\middle|\; n \in \mathbb{N}, n > d\right\}$. Using $d-1$ instead of $d-2$ is a typo.)

The proof essentially points out that the absence of interference between non-cooperative rules does not allow generating complex languages other than by growing very fast, which means, given the bound on the growth rate established in

Proposition 1, that the expressive power of such polymorphic P systems is bounded by their depth.

On the other hand, the language $\left\{ 2^{\binom{n}{d-2}} \;\middle|\; n \in \mathbb{N}, n > d \right\}$ *can* actually be generated by a polymorphic P system of depth $d+1$. P system Π_2 (Example 2) is an example for $d+1=5$. This implies that deeper membrane structures confer strictly more expressive power.

Corollary 3 (infinite hierarchy [11, Corollary 3]).
$NOP_*^d(polym(ncoo)) \subsetneq NOP_*^{d+1}(polym(ncoo))$, $d \geq 1$.

The latter statement brings up the existence of an infinite hierarchy in the class of polymorphic P systems with respect to their computing power.

6 P Systems with States: Polymorphism on Steroids

Even though P systems are commonly considered as stateless computing devices, many characterisations of their computational power are built around simulations of some stateful models of computing, such as register machines [12, 17]. This effectively delimits a subcategory of ingredients of the simulating P systems which represent the state of the simulated device. Various ways of representing states have been employed, including membrane polarisations, finite state control of rule applications, toxic objects, rule colours, etc.; Section 2 of [3] gives a comprehensive overview.

In this article, we recall the concept of P systems with states as introduced in [3]. This concept, while not exclusively generalising polymorphism, gives an interesting perspective on dynamically changing rules.

An n-cell *(tissue) P system with states* is the following construct [3]:

$$\Pi = (n, O, O_T, Q, \delta, f_I, f_O, q_i, F, C_i),$$

where n is the number of cells, O is the set of possible objects in the cells of Π, $O_T \subseteq O$ is the set of terminal objects, Q is a (possibly infinite) set of states, δ is the function computing the new state and the new configuration from the current ones, f_I is the input function, f_O is the output function, q_i is the initial state, $F \subseteq Q$ is the set of final states, and $C_i \in O^n$ is the core of the initial configuration.

This definition is a very general take on P systems. Essentially, the defined construct operates on a collection of objects (which can be anything), guided by a state control. To represent conventional P systems, O should be the set of all multisets over a finite alphabet V: $O = V^\circ$. To represent P systems with string objects, O should be the set of all strings over a finite alphabet: $O = V^*$.

The state control embedded in the definition of P systems with states is rather general as well, since the set of states Q is allowed to overlap the set of objects or to be infinite. Still, for the majority of applications, it seems reasonable to restrict

Q to be a computable set. The state transition function should be computable as well, but, in general, it may be of any complexity.

P systems with states are equipped with the "encoding" function f_I and the "decoding" function f_O. No restriction is imposed on the complexity of these functions; in most cases, however, it seems reasonable to require them to be "considerably less complex" than the state transition function δ to avoid the trivial situations in which the bulk of the computation happens in the encoding or the decoding phase.

Finally, the core is a generalisation of the initial contents of the membranes of a P system. It provides the additional "salt" to the input function f_I, which therefore constructs the initial configuration by combining the input (if there is some) with the core.

P systems with states vastly generalise many models of P systems, including models which do not directly use the term "state" in their definition. In particular, the definition of the state transition function is sufficiently general to easily capture the notion of polymorphism, as can be seen in the following example.

Example 5. Consider the following P system with states:

$$\Pi_6 = (1, \{a\}^*, \{a\}^*, \{i, h\}, \delta, f_I, f_O, i, \{h\}, (a^2)).$$

It has one cell, its objects are finite strings of the form a^*, and all of these objects are terminal. Π_6 has two states: the working state i and the halting state h. The state transition function is defined to either apply the "rule" $a \to a^{\text{current length of the string}}$ in a maximally parallel fashion, or to switch nondeterministically to the halting state. Using the notations from [3], this can be written in the following way:

$$\delta(i, (a^m)) = \{\ \ (i,\ \{(a \to a^m, maxpar)\}),\ \ (h, \emptyset)\ \ \}.$$

Π_6 operates almost exactly like Π_4 from Example 4, by iterated squaring. Due to the versality of the state transition function, Π_6 only needs one step to carry out one multiplication.

To make Π_6 actually generate a number language, we will define f_I to be the constant function $f_I = (i, (a^2))$, initialising the derivation of Π_6 in state i with the initial word a^2, and we will define f_O as the function extracting the multiplicity of a: $f_O(h, (a^m)) = m$. With these definitions, the number language generated by Π_6 is exactly the number language generated by Π_4: $L(\Pi_6) = N(\Pi_4) = \left\{2^{2^n} \mid n \in \mathbb{N}\right\}$.

Quite unsurprisingly, the expressivity of P systems with states is a strict superset of the expressivity of polymorphic P systems. Yet, the whole point of the introduction of P systems with states is pointing out new extensions or additional ingredients to existing, less general models. Examples of such extensions include using different derivation modes (minimal parallelism?) or halting conditions (partial adult halting?) with polymorphism.

7 Open Questions

We choose to employ the concluding section of this article to outline some of the possible research directions around polymorphic P systems. For readability, we group the research directions into subsections.

7.1 Expressive Power

The expressive power of polymorphic P systems is largely understudied. The original article [4] introducing the model barely scratched the surface of the problem, pointing out the sufficiently obvious computational completeness in the general case (because polymorphic P systems generalise conventional P systems, which are computationally complete) and focused on showing off the power of the model almost exclusively. The later work [11] considered the most basic variant without cooperativity and communication, and showed that bare polymorphism may induce pretty non-trivial behaviour.

The following list gives some concrete questions concerning the expressivity of polymorphism, in no particular order.

Question 1 (right polymorphism [11]). Are non-cooperative, right-polymorphic P systems less powerful than general polymorphic P systems?

While it is relatively easy to give a positive answer to the counterpart of this question for *left*-polymorphic P systems – such systems have insufficiently deep membrane structures to capture general polymorphism, as shown in Corollary 1 and Theorem 3 – right polymorphism is a more difficult subject to handle. Indeed, Corollary 1 seems to imply that the contribution of the variability of left-hand sides of membranes is rather unimportant. Given that virtually no synchronisation is possible between the right- and the left-hand-side membranes defining a rule, it may well be possible that non-cooperative, right-polymorphic P systems are just as powerful as P systems in which both rule sides are allowed to vary.

Question 2 (upper bounds [11]). What are the upper bounds on the expressive power of non-cooperative (left-, right-) polymorphic P systems?

Subsection 5.3 recalls a series of results giving a hint of the computational power of different kinds of polymorphism, but no actual upper bounds have been set yet. Proving such upper bounds may be quite helpful in improving our understanding of polymorphism.

Question 3 (target indications [11]). What is the expressive power of (strongly, weakly) non-cooperative polymorphic P systems *with* target indications?

Examples 3 and 4 seem to suggest that the contribution of target indications to the computing power of polymorphic P systems is considerable. Adding targets to rules seems to be a change sufficiently fundamental to actually differentiate between strong and weak non-cooperativity (cf. Conjecture 1).

7.2 Better Target Indications

The original article [4] already considers polymorphic rules with targets indications. It turns out that pretty coarse indications, sending the whole right-hand side into a membrane, already allow building interesting behaviour. However, in a typical communication P system, target indications are assigned to *individual* symbols, not to entire right-hand sides. The following question therefore seems very natural to be asked.

Question 4 (finer targets). What is the most natural way to generalise target indications attached to *individual* symbols?

This question is not exactly as trivial as it may sound, because assigning target indications enriches the structure of right-hand sides of rules, making them more than just multisets. In the coarse definition given in [4] and taken over in the present work, the extra structure is factored out into the mapping φ assigning target indications to right-hand-side membranes; this allows a uniform treatment of left- and right-hand sides, as well as of the contents of the skin. Attaching targets to individual symbols may break this symmetry and should be handled carefully. For example, if we consider that symbols like (a, in_h) may appear in a right-hand-side membrane, what happens if such a symbol is moved into a *left*-hand-side membrane? Furthermore, do we allow *nested* target indications, like in $((a, in_h), in_m)$?

The quest for finer target indications hides one more, orthogonal question.

Question 5 (dynamic targets). What is the most natural way to define *dynamic* targets?

Having dynamic targets would mean that a polymorphic P system could change the targets assigned to its right-hand-side membranes or to some particular symbols contained in such a membrane. This question is interesting not only in the context of finer-grained target indications: one may consider, for example, the ways in which a polymorphic P system as defined in this article (i.e., with coarse target indications given by the mapping φ) could dynamically modify the targets assigned to some right-hand-side membranes.

Finally, the classical question on the topology of the membrane structure may also be considered for polymorphic P systems.

Question 6. What is the most natural way to define polymorphic *tissue* P systems?

The conventional definition of tissue P systems relies on unrestricted target indications which allow any rule to place a symbol in any membrane (see [10] for a comprehensive overview). In this case, the membrane structure is typically omitted, since it has no influence on the evolution of the system. One of the implications of having such relaxed target indications in polymorphic P systems would probably be some kind of "flattening" of the membrane structure, since it will no longer be necessary for a rule to be "physically" located within a left- or a right-hand-side membrane of a another rule to be able to modify it.

7.3 Dissolution and Division

Dissolution and division are among the most fascinating aspects of membrane computing. They were introduced to parallel the corresponding processes happening in the living cell, and have become notorious for the complexity of the behaviour they induce. This is particularly true for membrane division which has been shown to allow solving NP-complete problems in polynomial time. Introducing these concepts for polymorphic P systems seems to be a quite challenging task.

Question 7 (dissolution). What is the most natural way of introducing *membrane dissolution* for polymorphic P systems?

In polymorphic P systems, membranes are no longer independent entities and go in pairs defining the dynamic rules. What would be the semantics of dissolving a member of such a pair? Should the other member be dissolved as well (*rule dissolution*)? Should the other member become an ordinary membrane? This direction seems more promising, since dissolving pairs of rule membranes should be equivalent to simply disabling rules.

Defining membrane division is even more challenging.

Question 8 (division). What is the most natural way of introducing *membrane division* for polymorphic P systems?

Just as in the case of dissolution, interdependence within pairs of rule membranes is what renders defining division difficult. What happens to membrane iL if its corresponding membrane iR is divided? Perhaps one of the simplest ways of dealing with this situation would be demoting both iL and the offspring of iR to regular membranes. Yet, considering more complex semantics may be very interesting. For example, we may consider that, dividing iR will also divide iL, in which case one will probably be able to achieve similar computational complexity feats as with P systems with active membranes.

7.4 Increasing the Exoticism

Polymorphic P systems are already a relatively exotic member of the family of P systems, but this should not prevent the enthusiastic from increasing the exoticism of the model by adding other ingredients. One may consider membrane polarisations affecting the semantics of rules, target selection forcing only rules with the same target indications to be picked in a single evolution step [2], or even sending rules across membranes [13]. Various frameworks generalising many models of P systems [3, 10] may be of help in finding the extra ingredients and in properly introducing them alongside polymorphism.

7.5 Applications

One of the most exciting aspects of polymorphic P systems is that, unlike in the case of P systems with active membranes [18], naively simulating polymorphism on conventional computers does not seem to incur additional costs with respect to simulating conventional multiset rewriting rules. This is because, essentially, a naive simulator of multiset rewriting would be moving around collections of symbols; to enable polymorphism, it would just have to support moving symbols into and out of rule sides.

Nevertheless, this reasoning may quickly become irrelevant for specific *optimised* implementations which *expect* the rules to be invariable. One notable example are simulators running on SIMD architectures (e.g., general purpose GPUs), which may have some particular restrictions on how data is to be moved around [14].

Question 9 (optimising simulators). Is polymorphism always easy to be simulated on conventional computers?

On the other hand, applications of polymorphic P systems may be of interest in topics related to complexity.

Question 10 (polymorphism vs. complexity). Can polymorphism be used for solving some complex problems faster?

This question is particularly interesting in the context of comparing polymorphic P systems with P systems with active membranes. The latter model is known to be able to solve NP-complete problems in polynomial time; would something similar be possible with polymorphism? We conjecture a negative answer to the strongest formulation of this question: otherwise, supposing trivial simulation of polymorphism on conventional computers, we could end up solving NP-complete problems in polynomial time using the existing technology.

The conjectured inferiority of the computing power of polymorphism with respect to that of active membranes implies a new question which is arguably more interesting.

Question 11. What are the problems that polymorphic P systems can solve *faster* than conventional P systems?

We have seen that polymorphism allows achieving non-trivial speed-ups in generation of some complex languages. Are there any other examples? If yes, *why* is polymorphism more efficient than conventional P systems?

Overall, polymorphic P systems seem to be a field full of interesting questions and results awaiting exploration. Given the scarcity of publications focusing on this extension of P systems, we conjecture that not all of the questions left unanswered require superhuman powers to be handled.

References

1. Artiom Alhazov. A note on P systems with activators. In Gheorghe Păun, Agustín Riscos-Núñez, Alvaro Romero-Jiménez, and Fernando Sancho-Caparrini, editors, *Second Brainstorming Week on Membrane Computing, Sevilla, Spain, February 2-7 2004*, pages 16–19, 2004.
2. Artiom Alhazov and Rudolf Freund. Variants of small universal P systems with catalysts. *Fundamenta Informaticae*, 138(1-2):227–250, 2015.
3. Artiom Alhazov, Rudolf Freund, Sergiu Ivanov, and Marion Oswald. Observations on P systems with states. In Marian Gheorghe, Ion Petre, Mario J. Pérez-Jiménez, Grzegorz Rozenberg, and Arto Salomaa, editors, *Multidisciplinary Creativity. Hommage to Gheorghe Păun on His 65th Birthday*. Spandugino, 2015.
4. Artiom Alhazov, Sergiu Ivanov, and Yurii Rogozhin. Polymorphic P systems. In Marian Gheorghe, Thomas Hinze, Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa, editors, *Membrane Computing*, volume 6501 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2011.
5. Fernando Arroyo, Angel V. Baranda, Juan Castellanos, and Gheorghe Păun. Membrane computing: The power of (rule) creation. *Journal of Universal Computer Science*, 8:369–381, 2002.
6. Matteo Cavaliere and Daniela Genova. P systems with symport/antiport of rules. In Gheorghe Păun, Agustín Riscos-Núñez, Alvaro Romero-Jiménez, and Fernando Sancho-Caparrini, editors, *Second Brainstorming Week on Membrane Computing, Sevilla, Spain, February 2-7 2004*, pages 102–116, 2004.
7. Matteo Cavaliere, Mihai Ionescu, and Tseren-Onolt Ishdorj. Inhibiting/de-inhibiting rules in P systems. In *Pre-proceedings of the Fifth Workshop on Membrane Computing (WMC5), Milano, Italy, June 2004*, pages 174–183, 2004.
8. John M. Coffin, Stephen H. Hughes, and Harold E. Varmus, editors. *Overview of Reverse Transcription – Retroviruses*. Cold Spring Harbor Laboratory Press, 1997.
9. Rudolf Freund. Generalized P-Systems. In Gabriel Ciobanu and Gheorghe Păun, editors, *Fundamentals of Computation Theory, 12th International Symposium, FCT '99, Iaşi, Romania, August 30–September 3, 1999, Proceedings*, volume 1684 of *Lecture Notes in Computer Science*, pages 281–292. Springer, 1999.
10. Rudolf Freund and Sergey Verlan. A formal framework for static (tissue) P systems. In George Eleftherakis, Petros Kefalas, Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa, editors, *Membrane Computing*, volume 4860 of *Lecture Notes in Computer Science*, pages 271–284. Springer, 2007.
11. Sergiu Ivanov. Polymorphic P systems with non-cooperative rules and no ingredients. In Marian Gheorghe, Grzegorz Rozenberg, Arto Salomaa, Petr Sosík, and Claudio Zandron, editors, *Membrane Computing - 15th International Conference, CMC 2014, Prague, Czech Republic, August 20–22, 2014, Revised Selected Papers*, volume 8961 of *Lecture Notes in Computer Science*, pages 258–273. Springer, 2014.
12. Ivan Korec. Small universal register machines. *Theoretical Computer Science*, 168(2):267–301, 1996.
13. Shankara Narayanan Krishna and Gheorghe Păun. P systems with mobile membranes. *Natural Computing*, 4(3):255–274, 2005.
14. Miguel A. Martínez-del-Amor, Luis F. Macías-Ramos, Luis Valencia-Cabrera, Agustín Riscos-Núñez, and Mario J. Pérez-Jiménez. Accelerated simulation of P systems on the GPU: A survey. In Linqiang Pan, Gheorghe Păun, Mario J. Pérez-Jiménez, and Tao Song, editors, *Bio-Inspired Computing - Theories and Applications:*

9th International Conference, BIC-TA 2014, Wuhan, China, October 16-19, 2014. Proceedings, pages 308–312. Springer, Berlin, Heidelberg, 2014.
15. Gheorghe Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61:108–143, 2000, and *TUCS Report 208*, November 1998 (`www.tucs.fi`).
16. Gheorghe Păun. *Membrane Computing: An Introduction*. Natural Computing Series Natural Computing. Springer, 2002.
17. Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa, eds. *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., New York, NY, USA, 2010.
18. Antonio E. Porreca, Alberto Leporati, Giancarlo Mauri, and Claudio Zandron. P systems with active membranes: Trading time for space. *Natural Computing*, 10(1):167–182, 2011.
19. Bulletin of the International Membrane Computing Society (IMCS). `http://membranecomputing.net/IMCSBulletin/index.php`.
20. The P Systems Website. `http://ppage.psystems.eu/`.

Some Notes on Membrane Computing and Image Processing

Daniel Díaz-Pernil[1], Miguel A. Gutiérrez-Naranjo[2], Hong Peng[3]

[1] CATAM Research Group - Dept. of Applied Mathematics I
University of Seville, Spain
sbdani@us.es
[2] Dept. of Computer Science and Artificial Intelligence
University of Seville, Spain
magutier@us.es
[3] School of Computer and Software Engineering
Xihua University, Chengdu, 610039, China
ph.xhu@hotmail.com

Summary. The application of Membrane Computing techniques to the study of digital images has been a vivid research area in the last years. In this paper, some of the research lines are presented and many of the main published papers are cited in the bibliography.

1 Introduction

Computer vision [131] is one of the most promising challenges for computer scientists in the next years. This research area is placed in the interplay of many disciplines, such as artificial intelligence, pattern recognition, signal processing, neurobiology, psychology or image processing among others. It concerns the automated processing of images from the real world to extract and interpret information on a real time basis.

Roughly speaking, a digital image is a two dimensional surface where each point is associated to a set of features such as bright or color. It is natural to consider only a discrete version of the definition, since only a finite amount of pixels placed in a lattice of integer coordinates is usually considered. The set of features can also have a finite amount of values (e.g., values in a range $\{0, \ldots, 255\}$ for colors). Such discrete amount of data makes digital images appropriate to be dealt with Membrane Computing techniques, but other features can be also considered. One of them is that the treatment of the image can be parallelized and locally solved. Regardless how large is the picture, many of the processes can be performed in parallel in different local areas of the image. Another interesting feature is that the local information needed for an image transformation can also be easily encapsulated in a membrane and represented as a multiset of objects. Such

features, together with the maximal parallelism, have encouraged many researchers to explore the links between Membrane Computing and digital image processing.

Formally, a *2D digital image I of size $n \times m$* ($n, m \in \mathbb{N}$) is a rectangular net of objects (i, j) called *pixels* (*voxels* in 3D images), with $1 \leq i \leq n$ and $1 \leq j \leq m$ (in general, any subset of the integer plane $\mathbb{Z} \times \mathbb{Z}$ can be chosen). Let $\mathcal{C}$, *the alphabet of colors of I*, be the ordered set of all colors in I. We define the size of $\mathcal{C}$, $|\mathcal{C}|$, as the number of colors of this alphabet. Moreover, we will assume that each pixel of I is associated to a color of $\mathcal{C}$. So, we encode the pixel (i, j) with associated color $a \in \mathcal{C}$ as the object a_{ij}. Therefore, the image I can be codified as the set $\{a_{ij} : a \in \mathcal{C} \wedge 1 \leq i \leq n \wedge 1 \leq j \leq m\}$. Another basic concept associated to pixels is the adjacency. In such case a distance is defined and two pixels are adjacent if the distance between them is one. Depending on the chosen distance, we can talk about 4-adjacency or 8-adjacency relation. The different treatments of such mappings (digital images) provide a big amount of current applications in biometrics [1], surveillance [27], medical imaging [3], human fingerprints classification [72], cartography [82], data compression and data storage [59], automated inspection of printed circuit boards [148] or optical character recognition (OCR) [134] among others.

In the literature, one can find many examples of the use of bio-inspired techniques for dealing with problems associated to the treatment of digital images. One of the classic examples is the use of cellular automata [124, 129]. Other efforts are related to artificial neural networks as in [42, 154] or, more recently, deep learning [63]. In this paper, we present some of the main research lines bridging Membrane Computing and digital images. In Section 2, we recall the first attempts of linking both disciplines, mainly based on array grammars and on a graphical interpretation of the information encoded in a P system configuration which allows to associate a picture to it. Section 3 is devoted to one of the main applications of Membrane Computing to image processing, namely the segmentation of images. Segmentation is the process of splitting a digital image into sets of pixels in order to make it simpler and easier to analyze. One of its main uses is the localization of objects and boundaries. Technically, the process consists on assigning a label to each pixel, in such way that pixels with the same label form a meaningful region. Among the applications of segmentation of digital images, we can find the face recognition [153] or location of objects in satellite images (roads, forests, etc.) [50], but probably its main application area is medical imaging [8]. Section 4 is devoted to the skeletonization of images. Skeletonization is one of the approaches for representing a shape with a small amount of information by converting an image into a more compact representation and keeping the meaningful features. The conversion should remove redundant *information*, but it should also keep the basic structure. Other of the most promising applications focuses on algebraic-topological aspects of the images (Section 5) as those related to homology theory or discrete Morse theory (see, e.g., [24, 116]). The paper ends with an example of application (Section 6) and some final conclusions.

2 First Steps

One of the first steps towards bridging Membrane Computing and digital images was considering two dimensional objects. The objects placed in a membrane are usually 0-dimensional or 1-dimensional (in the case of string-objects [46, 107]). The first step for considering images in Membrane Computing is the use of 2-dimensional objects, called arrays. Array grammars have been widely studied in the literature. They can be considered as a straightforward extension of string grammars to two dimensional pictures. Such pictures are sets of symbols placed in the points with integer coordinates of the plan (see, e.g. [28, 48, 123, 143]).

In [15], the model of array-rewriting P systems was presented on the basis of the transition P systems [105]: Rules are of type $\mathcal{A} \to \mathcal{B}(tar)$ where $\mathcal{A}$ is the array to be rewritten, $\mathcal{B}$ is the new one and $tar \in \{here, in, out\}$ indicates the emplacement of the picture where the substitution has been made. Different approaches can be found in [108, 137]. In a natural way, transition P systems were extended to other P systems models, as in [20], where tissue P systems with arrays are used for dealing with the segmentation of images (see Section 3).

For example[4], let us consider a P system with three nested membranes $[\,[\,[\,]_3\,]_2\,]_1$, an alphabet with two symbols a and # (the blank), an initial configuration with membranes 2 and 3 empty and the array $\begin{matrix} a \\ a \end{matrix}$ placed in the membrane 1. Let us consider the sets of rules

$$R_1 = \left\{ \begin{matrix} & \# \\ \# & a \end{matrix} \to \begin{matrix} & a \\ \# & a \end{matrix} (in) \right\},$$

$$R_2 = \left\{ \begin{matrix} a & \# \\ \# & \end{matrix} \to \begin{matrix} a & a \\ \# & \end{matrix} (out), \begin{matrix} a & \# & \# \\ \# & & \end{matrix} \to \begin{matrix} a & a & a \\ \# & & \end{matrix} (in) \right\},$$

$$R_3 = \emptyset.$$

This P system generates all the L-shaped angles with equal arms, each arm being of length at least three. In the literature, there are many approaches setting bridges between array grammars and Membrane Computing (see, e.g. [2, 14, 15, 30, 31, 45, 81, 108, 137]).

Other of the first links between P systems and digital images was the generation of graphical representation of branching structures able to simulate the growing of higher plants. The growth of plants, considered as a function of time, has attracted the attention of scientific community for a very long time. Features such as the bilateral symmetry of leaves, the central symmetry of flowers have been matter of study for computer scientists, mathematicians and life scientists among others. In 1968, Aristid Lindenmayer presented a theoretical framework for studying the development of simple multicellular organisms. The devices introduced in this framework are known as *parallel rewriting systems* or *L-systems*. L-systems were

[4] Adapted from the Example 1 in [15].

introduced for modelling multicellular organisms in terms of division, growth and death of individual cells [84, 85]. Several years later, the range of applications of L-systems was extended to higher plants and complex branching structures [49].

In [52, 53], a first approach for using P systems to simulate the growth and development of living plants was presented. This approach mixes L-systems and P systems, leading in fact to an L-system *factorized* into several units, which are them computed in the compartments delimited by the membranes of the P system.

Later, a new approach [118, 119, 121, 122] was presented. It used the model of *P systems with membrane creation* [91, 104] where an object can produce a whole membrane via the application of a rule. This kind of rules allows to control the development of the membrane structure of a cell-like P system, which has a natural graphical interpretation as a tree-like graph. The multisets placed inside the membranes can be graphically interpreted in terms of color, length or thickness of the corresponding segment in the branching structure, allowing to provide a more and more realistic appearance. Fig. 1, borrowed from [122], shows the graphical representation of four configurations of a P system with membrane creation. The drawn trees reproduce the tree-like structure of the membrane structure of the P systems and the length and thickness of the branches and the corresponding angles are fixed by the multiset of objects placed in the membranes in the corresponding configuration.

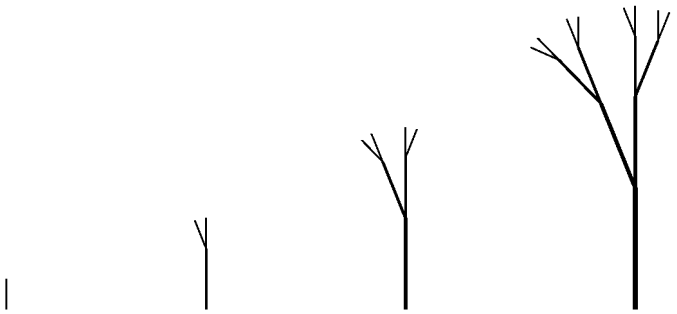


Fig. 1. Graphical interpretation of four configurations of a P system simulating the growing of a plant. Image borrowed from [122].

A close interpretation of the growing of higher plants was made for linking Membrane Computing and fractals [67]. A *fractal* [88] is a shape made of parts similar to the whole in some way. This self-similarity occurs over an infinite range of scales in pure mathematical structures but over a finite range in many natural objects such as clouds, coastlines, surface of tumors or snowflakes. An appropriate use of the membrane creation rules together with the non-determinism intrinsic to P systems and the interpretation of the multisets of objects could be useful in the study of the smooth surface of solid tumours, as pointed out in [68].

3 Segmentation

The study of array grammars or the graphical interpretation of a P system configuration were the first attempts to link Membrane Computing techniques to the study of images, but they cannot be considered as image processing. In this way, one of the most studied image processing techniques studied in Membrane Computing is segmentation.

Segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share certain visual characteristics. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze. Image segmentation is typically used to locate objects and boundaries (lines, curves, etc.) in images. More precisely, image segmentation is the process of assigning a label to every pixel in an image in such a way that those pixels with the same label share certain visual characteristics. These regions are mutually disjoint, well-defined and have the same properties. The purpose of segmenting an image is to identify regions that are then utilized to recognize and understand the image. In the past decades, a large number of image segmentation algorithms have been developed [18, 86, 96]. These algorithms can be roughly classified into three categories: threshold-based segmentation methods, edge-based segmentation methods and region-based segmentation methods. Segmentation has shown its utility in bordering tumors and other pathologies, computer-guided surgery or the study of anatomical structure, but also in techniques which are not thought to produce images but to produce positional information such as electroencephalography (EEG), or electrocardiography (EKG).

In the literature, there exist different techniques to segment an image. Some of them are clustering methods [83, 141], histogram-based methods [140], Watershed transformation methods [139, 147] or graph partitioning methods [149]. Some of the practical applications of image segmentation are medical imaging [141], the study of anatomical structure, locate objects in satellite images (roads, forests, etc.) [132] or face recognition [80] among others.

In the literature, one can find many approaches from Membrane Computing to the problem of segmenting images. These approaches are summarized in Table 1.

Table 1. The studies of P systems-based segmentation

#	Methods	Bibliography
1	Threshold-based segmentation	[21, 26, 36, 99, 100, 101] [102, 113, 142, 152]
2	Region-based segmentation	[25, 74, 103, 146, 145]
3	Edge-based segmentation	[9, 19, 22, 33, 37, 144]
4	Software and hardware implementation	[9, 10, 33, 36, 98, 133]

3.1 Threshold-based Segmentation Methods

Thresholding is widely used as a popular technique in image segmentation [126]. The goal of thresholding is to separate objects from background image or discriminate objects from objects that have distinct gray levels. Its underlying assumption is that an image consists of different regions corresponding to the gray-level ranges. It has been used widely as a tool to segment the gray images, but only a few works on color image segmentation have been reported. The main advantage of this technique lies in its simple computation approach. However, the threshold-based segmentation method ignores the spatial relationship information.

In recent years, P systems have been used to deal with threshold-based segmentation problems. Díaz-Pernil *et al.* [36] developed an image segmentation method on 2D images using P systems, which was applied to medical image segmentation. Christinal *et al.* [21] presented an image segmentation method based on tissue-like P systems, which segmented the images using the 4-neighborhood relation of pixels in the 2D-image. However, they only addressed the segmentation results of artificial images rather than real-life images. Reina-Molina *et al.* [113] proposed a thresholding method based on tissue-like P systems with multiple auxiliary cells. Peña-Cantillana *et al.* [99] presented a tissue-like P systems-based thresholding segmentation method using 4-adjacency. Christinal *et al.* [26] have proposed a variant of P system (tissue-like P system) using the rules to perform a parallel color segmentation of 2D images based on a threshold method. Wang *et al.* [142] proposed an optimal single-level thresholding method based on P systems. Peng *et al.* [100] presented a three-level thresholding method based on cell-like P systems for image segmentation. Zhang *et al.* [152] developed an infrared object segmentation method with Membrane Computing, which was used to obtain the optimal parameters quickly. Peng *et al.* [102] proposed a thresholding method based on tissue-like P systems and fuzzy entropy. Peng *et al.* [101] developed an optimal multi-level thresholding method based on cell-like P system.

3.2 Region-based Segmentation Methods

There are two key approaches regarding the region-based segmentation method: region growing and splitting-merging. Region growing polymerizes images pixels or sub-regions that are considered as seeds into larger regions according to some criteria [16]. The characteristics of pixels and the adjacency of spatial distribution are fully considered in region growing. However, because of its iterative computational process, region growing has a high computing cost. In recent years, P systems have been used to realize several region-based segmentation methods.

Christinal *et al.* [25] proposed a region-based segmentation method for 2D and 3D images with tissue-like P systems. Yang *et al.* [146] developed a region-based segmentation method with Membrane Computing, which effectively segmented gray images. However, the method cannot be extended to color images. Thus, Peng *et al.* [103] presented a region-based method to deal with color image segmentation. Isawasan *et al.* [74] proposed a region-based segmentation method based on tissue

P systems for hexagonal digital images. Yahya *et al.* [145] proposed a region-based method based on tissue P systems for 2D image segmentation.

3.3 Edge-based Segmentation Methods

The edge-based segmentation method is extensively utilized for gray-level image segmentation, which is based on the detection of discontinuity in the gray level. An edge or boundary is a place where there is a more or less abrupt change in the gray level. Among the most used edge detection operators are Roberts operator, Sobel operator, Gauss-Laplace operator and Canny operator. Inspired from the mechanism of P systems, a number of edge-based segmentation methods have been addressed in recent years.

Christinal *et al.* [22] presented an edge-based segmentation method using tissue-like P systems for 2D and 3D images. Díaz-Pernil *et al.* [37] proposed an edge-based segmentation method based on tissue-like P systems to obtain homology groups. Díaz-Pernil *et al.* [33] proposed a parallel implementation of a new algorithm for segmenting images with gradient-based edge detection by using techniques from Membrane Computing. Carnero *et al.* [9] used tissue-like P systems to design an edge-based segmentation method. Christinal *et al.* [19] develop a method to search partially bounded regions with P systems. Yahya *et al.* [144] a tissue-like P system-based edge-based segmentation method for 2D hexagonal images.

3.4 Software and Hardware Implementation

Carnero *et al.* [10] have proposed a new hardware tool including a Field-Programmable Gate Array unit (FPGA) to perform segmentation of digital images for solving edge-based detection and noise removal problem. Their system uses Membrane Computing as well as a hardware programming (VHDL) language to propose an *ad hoc* processor. In another work, Díaz-Pernil *et al.* [36] have proposed a new software tool for segmenting 2D digital images on the basis of tissue-like P system, wherein the object oriented C++ programming language has been used in the implementation part. However, they did not provide a clear explanation regarding the technical aspects of developing the proposed tool.

A bio-inspired Membrane Computing software has been proposed by Peña-Cantillana *et al.* [98] to solve the threshold problem and it has been implemented in Compute Unified Device Architecture (CUDA™), an innovative device architecture (see Section 7). Carnero *et al.* [9] have presented the use of the FPGA to implement tissue-like P system rules for solving segmentation problems. Sheeba *et al.* [133] have proposed tissue-like P system to segment medical image, nuclei of the white blood cells (WBCs) of the peripheral blood smear images in morphology segmentation technique. Their algorithm has been implemented using MATLAB software.

In the work of Díaz-Pernil *et al.* [33] a CUDA™ has been presented to implement tissue-like P system rules for segmenting images by the use of gradient-based

Fig. 2. A hand-written word and its skeletonization. Image borrowed from [38].

edge detection to enhance the traditional methods of segmenting digital images. In [145], Yahya *et al.* used the tissue P systems simulator presented in [7] to check the validity of their design.

4 Skeletonization

Skeletonization in image processing is an approach for representing a shape with a small amount of information by converting an image into a more compact representation and keeping the meaningful features. The conversion should remove redundant *information*, but it should also keep the basic structure. The concept of skeleton was introduced by Blum in [5, 6], under the name of medial axis transformation. The skeleton of an image is useful to characterize objects by a compact representation while preserving the connectivity and topological properties of any image. The most important features concerning a shape are its *topology* (represented by connected components, holes, etc.) and its *geometry* (elongated parts, ramifications, etc.), thus they must be preserved.

Roughly speaking, we can say that the image B is a skeleton of the black and white image A, if the former has fewer black pixels than the latter, preserves its topological properties and, in some sense, keeps its *meaning*. Figure 2 illustrates this idea. The skeletonized image keeps the *meaning* of the original one and it uses fewer black pixels. It keeps the basic geometry of the original image and also its topology. Let us remark that the white regions inside the hand-made words are also white regions in the skeletonized one and the connectedness is preserved.

Skeletonization has been found useful for data compression and pattern recognition in a wide range of applications in the industrial and scientific fields. It is usually considered as a pre-processing step in pattern recognition algorithms, but

its study is also interesting by itself for the analysis of line-based images such as coronary arteries [41], human fingerprints classification [72], cartography [82], data compression and data storage [59], automated inspection of printed circuit boards [148] or optical character recognition (OCR) [134] among others. In many cases, the transformation of all the pixels can be done in parallel, since the *state* of a pixel at the step i only depends on the *states* of a set of pixels at the step $i-1$. Such parallelism in skeletonizing algorithms has been broadly studied (see, e.g., [65, 87, 138, 151]). The development of new hardware architectures has also contributed to new parallel implementations of these algorithms [55, 70, 71].

In [38, 39], Díaz-Pernil *et al.* presented an implementation of the Guo and Hall algorithm [65, 66] for skeletonizing images by using Spiking Neural P systems. In this algorithm, the pixels are examined for deletion in an iterative process. First of all, given an $p \times q$ image, it is divided into two sub-sections. One of the sections is composed by the pixels a_{ij} such that $i+j$ is even. Alternatively, the second sub-section corresponds to the pixels a_{ij} such that $i+j$ is odd. The algorithm consists on two sub-iterations where the removal of redundant pixels from both sub-sections are alternated, i.e., in each step only the pixels of one of the subsections are evaluated for its deletion.

The decision is based on a 3×3 neighborhood. Given a pixel $P0$, a clockwise enumeration $P1, \ldots, P8$ of its eight neighbor pixels is considered, as shown in Figure 3 (a). As usual, for each $i \in \{1, \ldots, 8\}$, Pi is considered as a Boolean variable, with the truth value 1 if Pi is *black* and 0 if Pi is *white*. In each iteration, an evaluated black pixel $P0$ is deleted (changed to white) if and only if a set of conditions are satisfied. The key point in [39] is the use of a compact representation of the neighborhood of a pixel (also used in [125]) and the use of weights associated to the synapses of the SN P system.

A different approach to the skeletonization of images with P systems was presented by Nicolescu [92] where an approach to the problem is presented by using complex objects and actors in the framework of Membrane Computing.

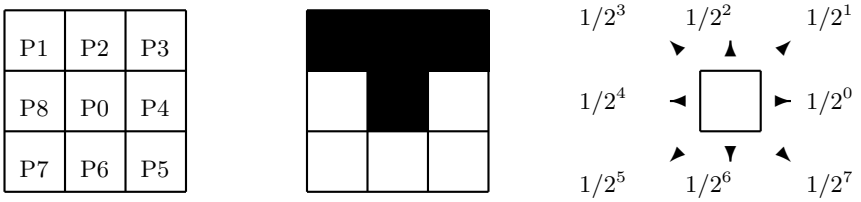


Fig. 3. (Left) Enumeration of the pixels in a 3×3 neighborhood. (Center) 3×3 neighborhood with encoding $[0,0,0,0,1,1,1,1,1]$, or, shortly, $2^4+2^5+2^6+2^7+2^8 = 496$. (Right) Scheme of the weights of the synapses. Figure borrowed from [38].

5 Algebraic-Topological Aspects

A different approach to computer vision can also be obtained from Topology. Topology in Computer Vision is referred to connectivity, in a general way. For example, we look for connected components, holes in these connected components, etc., where the compulsory idea is the connection. In particular, algebraic topology [47] provides techniques and algorithms for dealing digital images from a topological point of view.

The relationship between Algebraic Topology and Natural Computing is not new. In 1996, J. Chao and J. Nakayama [17] connected both areas using Neural Networks by extended Kohonen maps. Some years after, Giavitto *et al.* studied in [54] the topological structure of the Membrane Computing and Ceterchi *et al.* published two works where Digital Image is introduced in the framework of the Membrane Computing [14, 15].

5.1 Effective Homology

Recently, the links between algebraic topology and membrane computing have started to be explored via *Homology Theory* [23, 24, 37]. In such cases black and white images are taken and using labeling techniques the number of black connected components and the number of holes[5] of these connected components are calculated. This information is known as *the Betti numbers* from a 2D picture.

Effective Homology [110, 130, 120], is a algebraic-topological theory mainly based on the computational notion of chain homotopy equivalence, a concept which algebraically connects a cell complex or subdivided object with its homology groups. Roughly speaking, a chain homotopy equivalence can be specified by an operator, called *chain homotopy operator*, working at level of linear combinations of cells which represents an efficient and non-redundant way of connecting cells. For instance, a chain homotopy operator at level of cells of dimension 0 of a cell complex K can be completely described by a directed spanning forest (as many trees as connected components the object has) of the graph subcomplex formed by all the cells of K of dimension 0 and 1. Effective Homology uses chain homotopy operators for capturing homology information and for representing the object in an algebraic-topological way. In fact, this idea is underlying the Eilenberg-MacLane work [43, 44] for computing the homology of *prime* spaces in homotopy theory, and it has been recently used in discrete image context. In [62], a method for computing homology aspects (with coefficients in the finite field $\mathbb{Z}/2\mathbb{Z} = \{0, 1\}$) of a three dimensional digital binary-valued volume V considered over a body-centered-cubic grid is described. The representation used in that paper for a digital image is an algebraic-topological model (AT-model) consisting in two parts:

- *Geometric modeling level:* A cell complex $K(V)$ topologically equivalent to the original volume is constructed. A 3D–cell complex consists of vertices (0–cells),

[5] White connected components surrounded by black connected components

edges (1–cells), faces (2–cells) and polyhedra (3–cells). In particular, each edge connects two vertices, each face is enclosed by a loop of edges, and each 3–cell is enclosed by an envelope of faces;

- *Homology analysis level:* Homology information about $K(V)$ is exclusively codified in terms of a chain homotopy operator [60, 61].

This method has recently evolved to a technique for generating a $\mathbb{Z}/2\mathbb{Z}$-coefficients. It takes an AT-model for a 26–adjacency voxel-based digital binary volume V using a polyhedral cell complex at geometric modeling level [76, 77, 89] and a chain homotopy operator described by a combinatorial vector field (a set of semidirected forests or a discrete differential form) at homology analysis level [111, 112]. For instance, a chain homotopy operator at level of cells of dimension 0 (vertices) of a cell complex $K(V)$ can be completely described by a semidirected spanning forest of the graph subcomplex formed by all the cells of $K(V)$ of dimension 0 and 1.

In Figure 4, a pixel-based digital object O (first picture from the left) is analyzed as a cell complex in which the square pixels are the 0-cells. The 1-cells are edges joining 8-neighbor pixels and these 2-cells are triangles or squares formed by three or four mutually (and in a maximal way) 8-adjacent pixels. Picture (b) describes this cell complex (in dark grey) in which the barycenters of the different cells are drawn (solid circles for the 0-cells, crosses for the 1-cells and solid squares for the 2-cells). The subcomplex formed by the 0 and 1-cells can be seen as a subgraph of the 8-adjacency graph of O. In (c), a spanning tree covering all the vertices of the cell complex is specified. In fact, we consider a subdivision of this tree, having as 0-cells the vertices of the cell complex and the barycenters of the 1-cells belonging to the tree. An arrow in the tree determines the *pairing* of the source (0-cell) and sink (1-cell) cells and, consequently, indicating in this way that both are killed in homology group computation. Let us emphasize that only the top left 0-cell of the complex is not paired. It is a representative cycle (critical 0-cell of the homological process determined by the tree) of the unique connected component that the object has. Finally, in (d) we also draw the trees *covering* the rest of cells. They are semidirected, with arrows from the barycenters of 1-cells to the barycenter of the 2-cells. In terms of a process for computing homology groups, an arrow also means here that its source and sink cells are both *killed*. There is a edge marked in yellow which is not paired with an arrow. This 1-cell is a representative *critical* cell of the one-dimensional homology generator that the object has.

Using Effective Homology Theory as main tool for designing algorithms for computing complexes topological invariants (cohomology ring, (co)homology operations, homotopy groups,...), the problem of decomposing the objects into combinatorial graph-like pieces appears in a natural way. A possible solution to solve the high complexity costs of these processes is provided here by Membrane Computing.

Authors use in [32, 34] a well known tool from Membrane Computing, *promoters*. They are used to speed up the membrane algorithms. In that way, a bigger

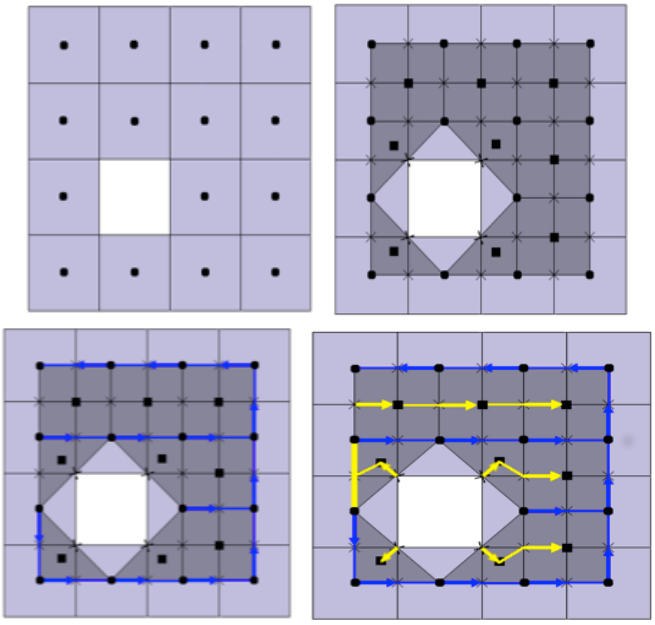


Fig. 4. Example. Pictures (a) left up, (b) right up, (c) left down and (d) right down.

amount of information is handled. Within the Digital Imagery setting, we determine here an Membrane Computing strategy for partially specifying a chain homotopy operator at the level of pixels for a pixel-based digital 2D binary object O. This fundamental data structure in Effective Homology is obtained in terms of a forest spanning every vertex of its associated adjacency graph. Every tree of this forest determines and *localizes* the corresponding connected component. In [32] authors obtain something more than the Betti numbers, they obtain the representative objects of each connected components and the borders of the holes. In this way, they get the homology groups.

Until here, the efficiency of the membrane models with these kind of problems is tested. But, we have to think to a bigger problem: what will happen when we want to work with bigger dimensions? We have two different ways to follow. We could carry on as before, where the simplicity (from algebraic point of view) enables high efficiency. Or, we can introduce new algebraic-topological concepts where the amount of information to deal would be increased.

Reina-Molina *et al.* decided to take this second option in [114, 115, 116]. They present a simulation of the Morse theory algorithms in a parallel way, getting the homology groups of n-D objects.

We have introduced two different theoretical ways to solve problems from Algebraic Topology, but software based on these theories have been development too. On one hand, Peña-Cantillana *et al.* in [32] generate a parallel software using

GPUs by CUDA to get the homology groups of 2D shapes based on techniques of spanning trees generated with membrane models where the promoters are compulsory. On the other hand, Reina-Molina *et al.* use PyCUDA (Python plus CUDA) to solve homological problems in a practical way in [117]. But, the complexity to adapt the Morse theory to parallel algorithms is high. So, this last software is working almost completely parallel.

Many open questions arise in the relation of Membrane Computing and Topology. From Algebraic Topology, we wonder if the Membrane Computing techniques can help for a better understanding of the problems and the design of more effective solutions. From Membrane Computing, a deeper study is necessary in order to explore how specific techniques of the different models can be applied. In particular, the use of priority in the application of rules is a strong requirement. It is worth to study if it can be avoided. From an implementation point of view, the exploration of the different parallel hardware architectures (clusters, grids, FPGA, ...) for the efficient implementation of the algorithms theoretically developed is an open research line.

6 An Example of Application

Image analysis and processing have a great significance in the field of medicine, especially in non-invasive treatment and clinical study. However, with the development of new technologies, larger quantity of data, especially high quality images, is available. Therefore, there is a new necessity of efficient and fast algorithms capable of processing and extracting meaningful features from images in a reasonable time. This is the case of mass screening programs for the early detection of retinal diseases such as glaucoma or diabetic retinopathy. Visual inspection of the large number of images so obtained is a time consuming task for the medical experts. Moreover, CAD (Computer Aided Diagnosis) tools based on retinal image processing developed in the past are limited by the balance between accuracy and complexity due to their sequential programming.

In [35], a fully automatized algorithm based on Membrane Computing techniques for the parallel segmentation of the optic disc in retinal fundus images was presented (see Fig. 5). The optic disc is seen on fundus color photographs as a bright yellowish disc in human retina from where the blood vessels and optic nerves emerge. Its relevance resides in the fact that it is a key point for the diagnosis of a wide variety of diseases such as glaucoma or diabetic retinopathy. Moreover, it is usually taken as a base for detecting other anatomical structures (macula, blood vessels) and retinal abnormalities (microaneurysms, hard exudates, drusens, etc.). Most of the methods found in the literature are semi-automatized. This means that the computer treatment is crucial in the localization and detection of the optic disc, but the human expert is the one who takes the final decision. In this paper, a fully automatized method is presented where no human expert is necessary for the detection of the optic disc.

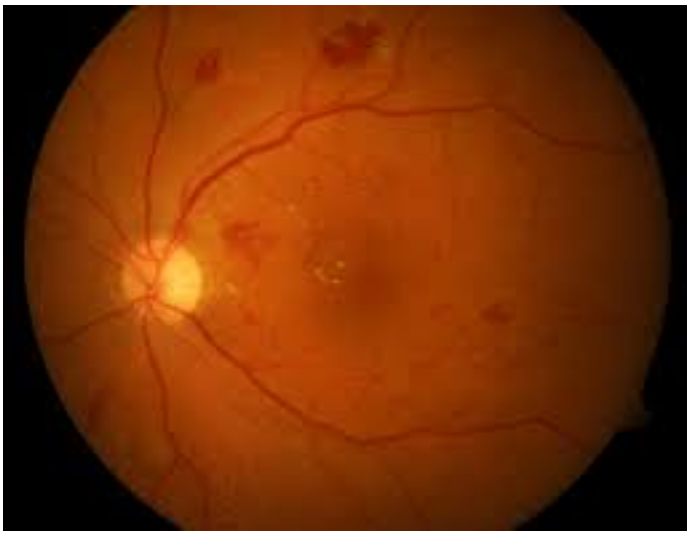


Fig. 5. Retinal image taken from the *Standard Diabetic Retinopathy Database* (DIARETDB1). The optic disc can be located as a yellow disc inside the image.

Changes in the optic disc can indicate the current state and progression of a certain disease while its diameter is usually used as a reference for measuring retinal distances and sizes [128]. Therefore, accurate optic disc localization and detection of its boundary is a principal and basic step for automated diagnosis systems [94].

In [35], a new method has been implemented with the Graphics Processing Units (GPU) technology. Image edges are extracted using a new operator called AGP-color segmentator based on the Membrane Computing approach presented in [33]. It is a Membrane Computing implementation on CUDA™ of the 3×3 and 5×5 versions of the Sobel algorithm [135] for edge detection. In order to choose an appropriate threshold to the binarization, a P system implementation [98] of the Hamadani algorithm [69] is applied. In order to avoid erroneous results, the obtained image is processed by eliminating the eye border. This is performed by applying a threshold on each color plane of the original image with the algorithm presented in [98]. The circular Hough transform is applied in parallel to the image in an interval of radius wide enough to consider all the possible optic discs.

The *Hough transform* is a well-known feature extraction technique used in image analysis. The classical Hough transform was concerned with the identification of lines in the image, but later the Hough transform has been extended to identifying positions of arbitrary shapes, most commonly circles or ellipses [4, 40, 73]. The basic idea behind the Hough transform is to convert the image into a parameter space that is constructed specifically to describe the desired shape analytically.

The reliability of the tool was tested with 129 images from the public databases DRIVE [136] and DIARETDB1 [75] obtaining an average accuracy of 99.6% and a

mean consumed time per image of 7.6 and 16.3 sec. respectively. A comparison with several state-of-the-art algorithms shows that the algorithm represents a significant improvement in terms of accuracy and efficiency.

7 Final Conclusions

Parallelizing classical digital image algorithms is a big challenge in the years to come [29, 97]. Such paralleling is much more complex than the merely simultaneous application of the sequential algorithm to different pieces of the image. The coordination of different simultaneous processes in a whole algorithm is so hard task that commonly the parallel algorithm needs to be re-designed with only slight references to the classical one. Usually, the design of a new parallel implementation not inspired by the sequential one allows an open-mind vision of the problem and the proposal of new creative solutions. Such new parallel solutions needs a strong theoretical support that allows to control, to formalize, to check and to formally verify new algorithms.

As pointed out above, many of the problems in digital images share features very interesting for using these techniques: the information can be split into little pieces and expressed as (multi)sets of objects; the computation steps can be processed by rewriting rules; and the same sequential algorithm must be applied in different regions of the image which are independent and they can be treated locally by a set of processors. All these features lead us to consider Membrane Computing to deal with digital images.

The key point of paralleling classical sequential algorithms is the search of the efficiency and this efficiency is strongly linked to the development of new parallel hardware architectures which allows a realistic implementation of the theoretical advantages of the parallel processes. Different hardware architectures (clusters, grids, FPGA, ...) propose different solutions [78, 79, 93, 127]. One of the most used architectures in the papers presented above has been the Compute Unified Device Architecture, CUDA™. This is a general purpose parallel computing architecture that allows the parallel NVIDIA Graphics Processing Units (GPUs) to solve many complex computational problems in a more efficient way than on a CPU.

GPUs have emerged as general-purpose coprocessors in recent years. Traditionally designed for gaming applications, GPUs offers many computing threads arranged in a Single-program Multiple-data (SPMD) model. The chosen hardware architecture for our parallel implementation has been the Compute Unified Device Architecture, (CUDA^{TM}), which allows the parallel NVIDIA GPUs [155] to solve many complex computational problems in a more efficient way than on a sequential Central Processing Unit (CPU)[6]. This architecture has been widely used in Membrane Computing not only for dealing with images [12, 13].

The choice of this parallel architecture is supported by several reasons. The first one is that the computing language CUDA™ allows programmers a friendly model

[6] For a good overview, the reader can refer to [95].

for implementing easily parallel programs, but the main reason comes from the practical side. In the last years, there exists an increasing interest in the specialized industry for the development of more and more powerful Graphic Processing Units which can be used for general purposes. This interest leads, on the one hand, to a more economically accessible (and hence, more extended) hardware and, on the other hand, to the development of more powerful computational units. The use of this new parallel architecture is currently explored as a tool for paralleling the treatment of digital images [11, 90].

Many other problems related to digital images have been addressed with Membrane Computing techniques. We can cite *smoothing* [99] which study how to enhance an image by removing regions which do not provide relevant information, the approach presented in [150], where Membrane Computing and quantum-inspired evolutionary algorithms are combined or the search of *partially bounded regions* [19]. This problem is also related to the HGB2I problem which consists on calculating the number of connected components and the representative curves of the holes of these components.

Special attention deserves the work by Gimel'farb *et al.* [56, 58], where the *symmetric dynamic programming stereo* (SDPS) algorithm [57] for stereo matching was implemented by using Membrane Computing techniques.

Many other approaches not cited in this paper have also been reported in the literature. This research area is rather active and it provides many lines for future researchers. Many open questions about new P system models, new digital image problems, new real-life application cases or new architectures will need deep studies in the next years.

References

1. Adeoye, O.S.: A survey of emerging biometric technologies. International Journal of Computer Applications 9(10), 1–5 (November 2010)
2. Annadurai, S., Kalyani, T., Dare, V.R., Thomas, D.G.: P systems generating isopicture languages. Progress in Natural Science 18(5), 617 – 622 (2008)
3. Ayache, N.: Medical image analysis and simulation. In: Shyamasundar, R.K., Ueda, K. (eds.) ASIAN. Lecture Notes in Computer Science, vol. 1345, pp. 4–17. Springer (1997)
4. Ballard, D.: Generalizing the Hough transform to detect arbitrary shapes. Pattern Recognition 13(2), 111 – 122 (1981)
5. Blum, H.: An associative machine for dealing with the visual field and some of its biological implications. In: Bernard, E.E., Kare, M.R. (eds.) Biological Prototypes and Synthetic Systems. vol. 1, pp. 244–260. Plenum Press, New York (1962), proceedings of the 2^{nd} Annual Bionics Symposium, held at Cornell University, 1961.
6. Blum, H.: An associative machine for dealing with the visual field and some of its biological implications. Computer and Mathematical Sciences Laboratory, Electronics Research Directorate, Air Force Cambridge Research Laboratories, Office of Aerospace Research, United States Air Force (1962)

7. Borrego-Ropero, R., Díaz-Pernil, D., Pérez-Jiménez, M.J.: Tissue simulator: A graphical tool for tissue P systems. In: Vaszil, G. (ed.) Proceedings of the International Workshop Automata for Cellular and Molecular Computing. pp. 23–34. MTA SZTAKI, Budapest, Hungary (August 2007), satellite of the 16th International Symposium on Fundamentals of Computational Theory
8. Campadelli, P., Casiraghi, E., Esposito, A.: Liver segmentation from computed tomography scans: A survey and a new algorithm. Artificial Intelligence in Medicine 45(2-3), 185–196 (2009)
9. Carnero, J., Díaz-Pernil, D., Gutiérrez-Naranjo, M.A.: Designing tissue-like P systems for image segmentation on parallel architectures. In: Martínez-del-Amor, M.A., Păun, Gh., Pérez-Hurtado, I., Romero-Campero, F.J., Valencia-Cabrera, L. (eds.) Ninth Brainstorming Week on Membrane Computing. pp. 43–62. Fénix Editora, Sevilla, Spain (2011)
10. Carnero, J., Díaz-Pernil, D., Molina-Abril, H., Real, P.: Image segmentation inspired by cellular models using hardware programming. Image-A: Applicable Mathematics in Image Engineering 1(3), 143–150 (2010)
11. Carranza, C., Murray, V., Pattichis, M., Barriga, E.S.: Multiscale AM-FM decompositions with GPU acceleration for diabetic retinopathy screening. In: Image Analysis and Interpretation (SSIAI), 2012 IEEE Southwest Symposium on. pp. 121–124 (2012)
12. Cecilia, J.M., García, J.M., Guerrero, G.D., Martínez-del-Amor, M.A., Pérez-Hurtado, I., Pérez-Jiménez, M.J.: Simulating a P system based efficient solution to SAT by using GPUs. Journal of Logic and Algebraic Programming 79(6), 317–325 (2010)
13. Cecilia, J.M., García, J.M., Guerrero, G.D., Martínez-del-Amor, M.A., Pérez-Hurtado, I., Pérez-Jiménez, M.J.: Simulation of P systems with active membranes on CUDA. Briefings in Bioinformatics 11(3), 313–322 (2010)
14. Ceterchi, R., Gramatovici, R., Jonoska, N., Subramanian, K.G.: Tissue-like P systems with active membranes for picture generation. Fundamenta Informaticae 56(4), 311–328 (2003)
15. Ceterchi, R., Mutyam, M., Păun, Gh., Subramanian, K.G.: Array-rewriting P systems. Natural Computing 2(3), 229–249 (2003)
16. Chang, Y., Li, X.: Adaptive image region-growing. IEEE Trans. Image Processing 3(6), 868–872 (1994)
17. Chao, J., Nakayama, J.: Cubical singular simplex model for 3D objects and fast computation of homology groups. In: 13th International Conference on Pattern Recognition (ICPR'96). vol. IV, pp. 190–194. IEEE Computer Society, IEEE Computer Society, Los Alamitos, CA, USA (1996)
18. Cheng, H., Jiang, X., Sun, Y., Wang, J.: Color image segmentation: advances and prospects. Pattern Recognition 34(12), 2259–2281 (2001)
19. Christinal, H.A., Berciano, A., Díaz-Pernil, D., Gutiérrez-Naranjo, M.A.: Searching partially bounded regions with P systems. In: Pant, M., Deep, K., Nagar, A., Bansal, J.C. (eds.) Proceedings of the Third International Conference on Soft Computing for Problem Solving: SocProS 2013, Volume 1. pp. 45–54. Springer India, New Delhi (2014)
20. Christinal, H.A., Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J.: Array tissue-like P systems. In: Martínez del Amor, M.A., Păun, Gh., Pérez Hurtado, I., Riscos-Núñez, A. (eds.) Eighth Brainstorming Week on Membrane Computing. pp. 37–51. Fénix Editora, Sevilla, Spain (2010)

21. Christinal, H.A., Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J.: Thresholding of 2D images with cell-like P systems. Romanian Journal of Information Science and Technology (ROMJIST) 13(2), 131–140 (2010)
22. Christinal, H.A., Díaz-Pernil, D., Real, P.: Segmentation in 2D and 3D image using tissue-like P system. In: Bayro-Corrochano, E., Eklundh, J.O. (eds.) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications 14th Iberoamerican Conference on Pattern Recognition, CIARP 2009, Guadalajara, Jalisco, Mexico, November 15-18, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5856, pp. 169–176. Springer, Berlin Heidelberg (2009)
23. Christinal, H.A., Díaz-Pernil, D., Real, P.: Using membrane computing for obtaining homology groups of binary 2D digital images. In: Wiederhold, P., Barneva, R.P. (eds.) Combinatorial Image Analysis 13th International Workshop, IWCIA 2009, Playa del Carmen, Mexico, November 24-27, 2009. Proceedings. Lecture Notes in Computer Science, vol. 5852, pp. 383–396. Springer, Berlin Heidelberg (2009)
24. Christinal, H.A., Díaz-Pernil, D., Real, P.: P systems and computational algebraic topology. Mathematical and Computer Modelling 52(11-12), 1982 – 1996 (December 2010), the BIC-TA 2009 Special Issue, International Conference on Bio-Inspired Computing: Theory and Applications
25. Christinal, H.A., Díaz-Pernil, D., Real, P.: Region-based segmentation of 2D and 3D images with tissue-like P systems. Pattern Recognition Letters 32(16), 2206 – 2212 (2011), advances in Theory and Applications of Pattern Recognition, Image Processing and Computer Vision.
26. Christinal, H.A., Díaz-Pernil, D., Real Jurado, P., Selvan, S.E.: Color segmentation of 2D images with thresholding. In: Mathew, J., Patra, P., Pradhan, D.K., Kuttyamma, A.J. (eds.) Eco-friendly Computing and Communication Systems: International Conference, ICECCS 2012, Kochi, India, August 9-11, 2012. Proceedings. pp. 162–169. Springer, Berlin, Heidelberg (2012)
27. Collins, R., Lipton, A., Kanade, T.: Introduction to the special section on video surveillance. Pattern Analysis and Machine Intelligence, IEEE Transactions on 22(8), 745 –746 (aug 2000)
28. Cook, C.R., Wang, P.S.P.: A Chomsky hierarchy of isotonic array grammars and languages. Computer Graphics and Image Processing 8(1), 144 – 152 (1978)
29. Davies, E.: Computer and Machine Vision: Theory, Algorithms, Practicalities. Elsevier Science, Waltham, Mass. (2012)
30. Dersanambika, K.S., Krithivasan, K.: Contextual array P systems. International Journal of Computer Mathematics 81(8), 955–969 (2004)
31. Dersanambika, K.S., Krithivasan, K., Subramanian, K.G.: P systems generating hexagonal picture languages. In: Martín-Vide, C., Mauri, G., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing, International Workshop, WMC 2003, Tarragona, Spain, July 17-22, 2003, Revised Papers. Lecture Notes in Computer Science, vol. 2933, pp. 168–180. Springer, Berlin Heidelberg (2003)
32. Díaz-Pernil, D., Berciano, A., Peña-Cantillana, F., Gutiérrez-Naranjo, M.A.: Bioinspired parallel computing of representative geometrical objects of holes of binary 2D-images, in press
33. Díaz-Pernil, D., Berciano, A., Peña-Cantillana, F., Gutiérrez-Naranjo, M.A.: Segmenting images with gradient-based edge detection using membrane computing. Pattern Recognition Letters 34(8), 846 – 855 (2013)

34. Díaz-Pernil, D., Christinal, H.A., Gutiérrez-Naranjo, M.A., Real, P.: Using Membrane Computing for Effective Homology. Applicable Algebra in Engineering, Communication and Computing 23(5-6), 233–249 (2012)
35. Díaz-Pernil, D., Fondón, I., Peña-Cantillana, F., Gutiérrez-Naranjo, M.A.: Fully automatized parallel segmentation of the optic disc in retinal fundus images. Pattern Recognition Letters 83, Part 1, 99 – 107 (2016), Geometric, Topological and Harmonic Trends to Image Processing
36. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Molina-Abril, H., Real, P.: A bio-inspired software for segmenting digital images. In: Nagar, A.K., Thamburaj, R., Li, K., Tang, Z., Li, R. (eds.) Fifth International Conference on Bio-Inspired Computing: Theories and Applications, BIC-TA 2010, University of Hunan, Liverpool Hope University, Liverpool, United Kingdom / Changsha, China, September 8-10 and September 23-26, 2010. vol. 2, pp. 1377 – 1381. IEEE Computer Society, Beijing, China (2010)
37. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Real, P., Sánchez-Canales, V.: Computing homology groups in binary 2D imagery by tissue-like P systems. Romanian Journal of Information Science and Technology 13(2), 141–152 (2010)
38. Díaz-Pernil, D., Peña-Cantillana, F., Gutiérrez-Naranjo, M.A.: Skeletonizing images by using spiking neural P systems. In: Martínez-del-Amor, M.A., Păun, Gh., Pérez-Hurtado, I., Romero-Campero, F.J. (eds.) Tenth Brainstorming Week on Membrane Computing. vol. I, pp. 91–110. Fénix Editora, Sevilla, Spain (2012)
39. Díaz-Pernil, D., Peña-Cantillana, F., Gutiérrez-Naranjo, M.A.: A parallel algorithm for skeletonizing images by using spiking neural P systems. Neurocomputing 115, 81 – 91 (2013)
40. Duda, R.O., Hart, P.E.: Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM 15(1), 11–15 (1972)
41. Dufresne, T.E., Sarwal, A., Dhawan, A.P.: A gray-level thinning method for delineation and representation of arteries. Computerized Medical Imaging and Graphics 18(5), 343 – 355 (1994)
42. Egmont-Petersen, M., de Ridder, D., Handels, H.: Image processing with neural networks - a review. Pattern Recognition 35(10), 2279–2301 (2002)
43. Eilenberg, S., MacLane, S.: Relations between homology and homotopy groups of spaces. The Annals of Mathematics 46(3), pp. 480–509 (1945)
44. Eilenberg, S., MacLane, S.: Relations between homology and homotopy groups of spaces. II. The Annals of Mathematics 51(3), pp. 514–533 (1950)
45. Fernau, H., Freund, R., Schmid, M.L., Subramanian, K.G., Wiederhold, P.: Contextual array grammars and array P systems. Annals of Mathematics and Artificial Intelligence 75(1-2), 5–26 (2015)
46. Ferretti, C., Mauri, G., Zandron, C.: P systems with string objects. In: Păun et al. [109], pp. 168 – 197
47. Freedman, D., Chen, C.: Algebraic topology for computer vision. Science And Technology (2009)
48. Freund, R.: Array grammars. Tech. Rep. 15/00, Research Group on Mathematical Linguistics, Rovira i Virgili University, Tarragona (2000)
49. Frijters, D., Lindenmayer, A.: A model for the growth and flowering of aster novae-angliae on the basis of table $< 1, 0 >$ L-systems. In: Rozenberg, G., Salomaa, A. (eds.) L Systems, Most of the papers were presented at a conference in Aarhus, Denmark, January 14-25, 1974. Lecture Notes in Computer Science, vol. 15, pp. 24–52. Springer (1974)

50. Gamanya, R., Maeyer, P.D., Dapper, M.D.: An automated satellite image classification design using object-oriented segmentation algorithms: A move towards standardization. Expert Systems with Applications 32(2), 616–624 (2007)
51. García-Quismondo, M., Macías-Ramos, L.F., Păun, Gh., Valencia-Cabrera, L. (eds.): Tenth Brainstorming Week on Membrane Computing, vol. II. Fénix Editora, Sevilla, Spain (2012)
52. Georgiou, A., Gheorghe, M.: Generative devices used in graphics. In: Alhazov, A., Martín-Vide, C., Păun, Gh. (eds.) Preproceedings of the Workshop on Membrane Computing. Technical Report, Vol. 28/03. pp. 266–272. Research Group on Mathematical Linguistics, Universitat Rovira i Virgili, Tarragona, Spain (2003)
53. Georgiou, A., Gheorghe, M., Bernardini, F.: Membrane-based devices used in computer graphics. In: Ciobanu, G., Păun, Gh., Pérez-Jiménez, M.J. (eds.) Applications of Membrane Computing, pp. 253–281. Natural Computing Series, Springer Berlin Heidelberg (2006)
54. Giavitto, J.L., Michel, O.: The topological structures of membrane computing. Fundamenta Informaticae 49(1-3), 123–145 (2002)
55. Gil Montoya, M., Garcia, I.: Implementation of parallel thinning algorithms on multicomputers: analysis of the work load balance. In: Proceedings of the Sixth Euromicro Workshop on Parallel and Distributed Processing, 1998. PDP '98. pp. 257–263 (1998)
56. Gimel'farb, G., Nicolescu, R., Ragavan, S.: P systems in stereo matching. In: Real, P., Díaz-Pernil, D., Molina-Abril, H., Berciano, A., Kropatsch, W. (eds.) Computer Analysis of Images and Patterns, Lecture Notes in Computer Science, vol. 6855, pp. 285–292. Springer Berlin / Heidelberg (2011)
57. Gimel'farb, G.L.: Probabilistic regularisation and symmetry in binocular dynamic programming stereo. Pattern Recognition Letters 23(4), 431–442 (2002)
58. Gimel'farb, G.L., Nicolescu, R., Ragavan, S.: P system implementation of dynamic programming stereo. Journal of Mathematical Imaging and Vision 47(1-2), 13–26 (2013)
59. González, R.C., Woods, R.E.: Digital image processing. Pearson/Prentice Hall (2008)
60. González-Díaz, R., Jiménez, M.J., Medrano, B., Molina-Abril, H., Real, P.: Integral operators for computing homology generators at any dimension. In: Ruiz-Shulcloper, J., Kropatsch, W.G. (eds.) CIARP. Lecture Notes in Computer Science, vol. 5197, pp. 356–363. Springer, Berlin Heidelberg (2008)
61. González-Díaz, R., Jiménez, M.J., Medrano, B., Real, P.: Chain homotopies for object topological representations. Discrete Applied Mathematics 157(3), 490–499 (2009)
62. González-Díaz, R., Real, P.: On the cohomology of 3D digital images. Discrete Applied Mathematics 147(2-3), 245–263 (2005)
63. Goodfellow, I., Bengio, Y., Courville, A.: Deep learning (2016), `http://www.deeplearningbook.org`, book in preparation for MIT Press
64. Graciani, C., Păun, Gh., Romero-Jiménez, A., Sancho-Caparrini, F. (eds.): Fourth Brainstorming Week on Membrane Computing, vol. II. Fénix Editora, Sevilla, Spain (2006)
65. Guo, Z., Hall, R.W.: Parallel thinning with two-subiteration algorithms. Communications of the ACM 32, 359–373 (March 1989)
66. Guo, Z., Hall, R.W.: Fast fully parallel thinning algorithms. CVGIP: Image Understanding 55, 317–328 (May 1992)

67. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J.: Fractals and P systems. In: Graciani et al. [64], pp. 65–86
68. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A., Romero-Campero, F.J.: How to express tumours using membrane systems. Progress in Natural Science 17(4), 449–457 (2007)
69. Hamadani, N.: Automatic target cueing in IR imagery. Master's thesis, Air Force Institute of Technology, WAFP (December 1981)
70. Heydorn, S., Weidner, P.: Optimization and performance analysis of thinning algorithms on parallel computers. Parallel Computing 17(1), 17 – 27 (1991)
71. Holt, C., Stewart, A.: A parallel thinning algorithm with fine grain subtasking. Parallel Computing 10(3), 329 – 334 (1989)
72. Hongbin, P., Junali, C., Yashe, Z.: Fingerprint thinning algorithm based on mathematical morphology. In: 8th International Conference on Electronic Measurement and Instruments, 2007. ICEMI '07. pp. 2–618 – 2–621 (july 2007)
73. Hough, P.V.C.: Machine analysis of bubble chamber pictures. In: International Conference on High Energy Accelerators and Instrumentation. pp. 554 – 558. CERN (1959)
74. Isawasan, P., Venkat, I., Subramanian, K.G., Khader, A.T., Osman, O., Christinal, H.A.: Region-based segmentation of hexagonal digital images using membrane computing. In: Membrane Computing (ACMC), 2014 Asian Conference on. pp. 1–4. IEEE (2014)
75. Kauppi, T., Kalesnykiene, V., Kamarainen, J.K., Lensu, L., Sorri, I., Raninen, A., Voutilainen, R., Uusitalo, H., Kälviäinen, H., Pietilä, J.: The diaretdb1 diabetic retinopathy database and evaluation protocol. In: Proceedings of the British Machine Vision Conference 2007, University of Warwick, UK, September 10-13, 2007. pp. 252–261. British Machine Vision Association (2007)
76. Kenmochi, Y., Imiya, A., Ichikawa, A.: Discrete combinatorial geometry. Pattern Recognition 30(10), 1719–1728 (1997)
77. Kenmochi, Y., Imiya, A., Ichikawa, A.: Boundary extraction of discrete objects. Computer Vision and Image Understanding 71(3), 281–293 (1998)
78. Khalid, N.E.A., Ahmad, S.A., Noor, N.M., Fadzil, A.F.A., Taib, M.N.: Analysis of parallel multicore performance on Sobel edge detector. In: Proceedings of the 15th WSEAS international conference on Computers. pp. 313–318. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA (2011)
79. Khalid, N.E.A., Ahmad, S.A., Noor, N.M., Fadzil, A.F.A., Taib, M.N.: Parallel approach of Sobel edge detector on multicore platform. International Journal of Computers and Communications 5, 236–244 (2011)
80. Kim, S.H., Kim, H.G., Tchah, K.H.: Object oriented face detection using colour transformation and range segmentation. Electronics Letters, IEEE 34, 979–980 (1998)
81. Krishna, S.N., Rama, R., Krithivasan, K.: P systems with picture objects. Acta Cybernetica 15(1), 53–74 (2001)
82. Lee, K.H., Cho, S.B., Choy, Y.C.: Automated vectorization of cartographic maps by a knowledge-based system. Engineering Applications of Artificial Intelligence 13(2), 165 – 178 (2000)
83. Li, X., Zhang, T., Qu, Z.: Image segmentation using fuzzy clustering with spatial constraints based on Markov random field via bayesian theory. IEICE Transactions

on Fundamentals of Electronics, Communications and Computer Sciences 91-A(3), 723–729 (2008)
84. Lindenmayer, A.: Mathematical models for cellular interaction in development: Parts I and II. Journal of Theoretical Biology 18, 280–315 (1968)
85. Lindenmayer, A.: Developmental systems without cellular interactions, their languages and grammars. Journal of Theoretical Biology 30(3), 455 – 484 (1971)
86. Liu, D., Jiang, Z., Feng, H.: A novel fuzzy classification entropy approach to image thresholding. Pattern Recognition Letters 27(16), 1968–1975 (2006)
87. Lü, H.E., Wang, P.S.P.: A comment on a fast parallel algorithm for thinning digital patterns. Communications of the ACM 29(3), 239–242 (Mar 1986)
88. Mandelbrot, B.B.: The Fractal Geometry of Nature. W. H. Freedman and Co., New York (1983)
89. Molina-Abril, H., Real, P.: Advanced homology computation of digital volumes via cell complexes. In: da Vitoria Lobo, N., Kasparis, T., Roli, F., Kwok, J.T.Y., Georgiopoulos, M., Anagnostopoulos, G.C., Loog, M. (eds.) SSPR/SPR. Lecture Notes in Computer Science, vol. 5342, pp. 361–371. Springer, Berlin Heidelberg (2008)
90. Moulik, S., Boonn, W.W.: The role of GPU computing in medical image analysis and visualization. In: Boonn, W.W., Liu, B.J. (eds.) Medical Imaging 2011: Advanced PACS-based Imaging Informatics and Therapeutic Applications. vol. 7967, pp. 79670L–79670L–8 (2011), proceedings of the SPIE
91. Mutyam, M., Krithivasan, K.: P systems with membrane creation: Universality and efficiency. In: Margenstern, M., Rogozhin, Y. (eds.) MCU. Lecture Notes in Computer Science, vol. 2055, pp. 276–287. Springer (2001)
92. Nicolescu, R.: Parallel thinning with complex objects and actors. In: Gheorghe, M., Rozenberg, G., Salomaa, A., Sosík, P., Zandron, C. (eds.) Membrane Computing - 15th International Conference, CMC 2014, Prague, Czech Republic, August 20-22, 2014, Revised Selected Papers. Lecture Notes in Computer Science, vol. 8961, pp. 330–354. Springer (2014)
93. Ogawa, K., Ito, Y., Nakano, K.: Efficient canny edge detection using a GPU. In: Proceedings of the 2010 First International Conference on Networking and Computing. pp. 279–280. ICNC '10, IEEE Computer Society, Washington, DC, USA (Nov 2010)
94. Osareh, A., Mirmehdi, M., Thomas, B., Markham, R.: Automated identification of diabetic retinal exudates in digital colour images. British Journal of Ophthalmology 87(10), 1220–3 (2003)
95. Owens, J.D., Houston, M., Luebke, D., Green, S., Stone, J.E., Phillips, J.C.: GPU Computing. Proceedings of the IEEE 96(5), 879–899 (May 2008)
96. Pal, N.R., Pal, S.K.: A review on image segmentation techniques. Pattern Recognition 26(9), 1277–1294 (1993)
97. Parker, J.: Algorithms for Image Processing and Computer Vision. John Wiley & Sons, New York, NY, USA (2010)
98. Peña-Cantillana, F., Díaz-Pernil, D., Berciano, A., Gutiérrez-Naranjo, M.A.: A parallel implementation of the thresholding problem by using tissue-like P systems. In: Real, P., Díaz-Pernil, D., Molina-Abril, H., Berciano, A., Kropatsch, W.G. (eds.) Computer Analysis of Images and Patterns - 14th International Conference, CAIP 2011, Seville, Spain, August 29-31, 2011, Proceedings, Part II. Lecture Notes in Computer Science, vol. 6855, pp. 277–284. Springer (2011)

99. Peña-Cantillana, F., Díaz-Pernil, D., Christinal, H.A., Gutiérrez-Naranjo, M.A.: Implementation on CUDA of the smoothing problem with tissue-like P systems. International Journal of Natural Computing Research 2(3), 25–34 (2011)
100. Peng, H., Shao, J., Li, B., Wang, J., Pérez-Jiménez, M.J., Jiang, Y., Yang, Y.: Image thresholding with cell-like P systems. In: García-Quismondo et al. [51], pp. 75–88
101. Peng, H., Wang, J., Pérez-Jiménez, M.J.: Optimal multi-level thresholding with membrane computing. Digital Signal Processing 37, 53–64 (2015)
102. Peng, H., Wang, J., Pérez-Jiménez, M.J., Shi, P.: A novel image thresholding method based on membrane computing and fuzzy entropy. Journal of Intelligent and Fuzzy Systems 24(2), 229–237 (2013)
103. Peng, H., Yang, Y., Zhang, J., Huang, X., Wang, J.: A region-based color image segmentation method based on P systems. Romanian Journal of Information Science and Technology 17(1), 63–75 (2014)
104. Pérez-Jiménez, M.J., Riscos-Núñez, A., Romero-Jiménez, A., Woods, D.: Complexity - membrane division, membrane creation. In: Păun et al. [109], pp. 302 – 336
105. Gheorghe Păun: Membrane Computing. An Introduction. Springer-Verlag, Berlin, Germany (2002)
106. Păun, Gh.: Computing with membranes. Tech. Rep. 208, Turku Centre for Computer Science, Turku, Finland (November 1998)
107. Păun, Gh.: Computing with membranes. Journal of Computer and System Sciences 61(1), 108–143 (2000), see also [106]
108. Păun, Gh.: Grammar systems vs. membrane computing: A preliminary approach. In: Pre-Proceedings of the Workshop on Grammar Systems, MTA SZTAKI Budapest. pp. 225 –245 (2004)
109. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford, England (2010)
110. Real, P.: Homological perturbation theory and associativity. Homology, Homotopy and Applications 2(5), 51–88 (2000)
111. Real, P., Molina-Abril, H.: Cell at-models for digital volumes. In: Torsello, A., Escolano, F., Brun, L. (eds.) GbRPR. Lecture Notes in Computer Science, vol. 5534, pp. 314–323. Springer, Berlin Heidelberg (2009)
112. Real, P., Molina-Abril, H., Kropatsch, W.G.: Homological tree-based strategies for image analysis. In: Jiang, X., Petkov, N. (eds.) CAIP. Lecture Notes in Computer Science, vol. 5702, pp. 326–333. Springer, Berlin Heidelberg (2009)
113. Reina-Molina, R., Carnero Iglesias, J., Díaz-Pernil, D.: Image segmentation using tissue-like P systems with multiple auxiliary cells. Image-A: Applicable Mathematics in Image Engineering 2(4), 25–28 (2011)
114. Reina-Molina, R., Díaz-Pernil, D.: Bioinspired parallel 2D or 3D skeletonization. Image-A: Applicable Mathematics in Image Engineering 3(6), 41–4 (2013)
115. Reina-Molina, R., Díaz-Pernil, D., Gutiérrez-Naranjo, M.A.: Cell complexes and membrane computing for thinning 2D and 3D images. In: García-Quismondo et al. [51], pp. 167–186
116. Reina-Molina, R., Díaz-Pernil, D., Real, P., Berciano, A.: Membrane parallelism for discrete Morse theory applied to digital images. Applicable Algebra in Engineering, Communication and Computing 26(1-2), 49–71 (2015)
117. Reina-Molina, R., Díaz-Pernil, D., Real, P., Berciano, A.: Effective homology of k-d digital objects (partially) calculated in parallel. Pattern Recognition Letters 83, 59–66 (2016)

118. Rivero-Gil, E., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J.: Graphics and P systems: Experiments with JPLANT. In: Díaz-Pernil, D., Graciani, C., Gutiérrez-Naranjo, M.A., Păun, Gh., Pérez-Hurtado, I., Riscos-Núñez, A. (eds.) Sixth Brainstorming Week on Membrane Computing. pp. 241–253. Fénix Editora, Sevilla, Spain (2008)
119. Rivero-Gil, E., Gutiérrez-Naranjo, M.A., Romero Jiménez, Á., Riscos-Núñez, A.: A software tool for generating graphics by means of P systems. Natural Computing 10(2), 879–890 (2011)
120. Romero, A., Rubio, J., Sergeraert, F.: Effective homology of filtered digital images. Pattern Recognition Letters 83, 23–31 (2016)
121. Romero-Jiménez, Á., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J.: Graphical modeling of higher plants using P systems. In: Hoogeboom, H.J., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Workshop on Membrane Computing. Lecture Notes in Computer Science, vol. 4361, pp. 496–506. Springer, Berlin Heidelberg (2006)
122. Romero-Jiménez, A., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J.: The growth of branching structures with P systems. In: Graciani et al. [64], pp. 253–265
123. Rosenfeld, A.: Picture Languages. Academic Press, Reading, MA (1979)
124. Rosin, P.L.: Training cellular automata for image processing. IEEE Transactions on Image Processing 15(7), 2076–2087 (2006)
125. Saeed, K., Tabedzki, M., Rybnik, M., Adamski, M.: K3M: A universal algorithm for image skeletonization and a review of thinning techniques. Applied Mathematics and Computer Science 20(2), 317–335 (2010)
126. Sahoo, P., Soltani, S., Wong, A.: A survey of thresholding techniques. Computer Vision, Graphics, and Image Processing 41(2), 233 – 260 (1988)
127. Sanduja, V., Patial, R.: Sobel edge detection using parallel architecture based on FPGA. International Journal of Applied Information Systems 3(4), 20–24 (July 2012), published by Foundation of Computer Science, New York, USA
128. Sekhar, S., Al-Nuaimy, W., Nandi, A.K.: Automated localisation of retinal optic disk using Hough transform. In: Biomedical Imaging: From Nano to Macro, 2008. ISBI 2008. 5th IEEE International Symposium on. pp. 1577–1580. IEEE (2008)
129. Selvapeter, P.J., Hordijk, W.: Cellular automata for image noise filtering. In: World Congress on Nature & Biologically Inspired Computing, 2009. NaBIC 2009. pp. 193–197. IEEE, IEEE (2009)
130. Sergeraert, F.: The computability problem in algebraic topology. Advances in Mathematics 104, 1–29 (1994)
131. Shapiro, L.G., Stockman, G.C.: Computer Vision. Prentice Hall PTR, Upper Saddle River, NJ, USA (2001)
132. Sharma, O., Mioc, D., Anton, F.: Polygon feature extraction from satellite imagery based on color image segmentation and medial axis. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences XXXVII, 235–240 (2008), part B3a, Commission III
133. Sheeba, F., Thamburaj, R., Nagar, A.K., Mammen, J.J.: Segmentation of peripheral blood smear images using tissue-like P systems. Bio-Inspired Computing: Theories and Applications (BIC-TA), 2011 Sixth International Conference on 0, 257–261 (sept 2011)
134. Smith, S.J., Bourgoin, M.O., Sims, K., Voorhees, H.L.: Handwritten character classification using nearest neighbor in large databases. IEEE Transactions on Pattern Analysis and Machine Intelligence 16(9), 915–919 (1994)

135. Sobel, I.E.: Camera models and machine perception. Ph.D. thesis, Dept. of Computer Sciences, Stanford, CA, USA (1970), aAI7102831
136. Staal, J., Abràmoff, M.D., Niemeijer, M., Viergever, M.A., van Ginneken, B.: Ridge-based vessel segmentation in color images of the retina. IEEE Transactions on Medical Imaging 23(4), 501–509 (2004)
137. Subramanian, K.G.: P systems and picture languages. In: Durand-Lose, J.O., Margenstern, M. (eds.) MCU. Lecture Notes in Computer Science, vol. 4664, pp. 99–109. Springer, Berlin Heidelberg (2007)
138. Suzuki, S., Abe, K.: Binary picture thinning by an iterative parallel two-subcycle operation. Pattern Recognition 20(3), 297 – 307 (1987)
139. Tarabalka, Y., Chanussot, J., Benediktsson, J.A.: Segmentation and classification of hyperspectral images using Watershed transformation. Pattern Recognition 43(7), 2367–2379 (2010)
140. Tobias, O.J., Seara, R.: Image segmentation by histogram thresholding using fuzzy sets. IEEE Transactions on Image Processing 11(12), 1457–1465 (2002)
141. Wang, D., Lu, H., Zhang, J., Liang, J.Z.: A knowledge-based fuzzy clustering method with adaptation penalty for bone segmentation of CT images. In: Proceedings of the 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference. vol. 6, pp. 6488–6491 (2005)
142. Wang, H., Peng, H., Shao, J., Wang, T.: A thresholding method based on P systems for image segmentation. ICIC Express Letters 6(1), 221–227 (2012)
143. Wang, P.S.P.: Some new results on isotonic array grammars. Information Processing Letters 10(3), 129 – 131 (1980)
144. Yahya, R.I., Shamsuddin, S.M., Hasan, S., Yahya, S.I.: Tissue-like P system for segmentation of 2D hexagonal images. ARO-The Scientific Journal of Koya University IV(1), 35–42 (2016)
145. Yahya, R.I., Hasan, S., George, L.E., Alsalibi, B.: Membrane computing for 2D image segmentation. International Journal of Advances in Soft Computing and its Application 7(1) (2015)
146. Yang, Y., Peng, H., Jiang, Y., Huang, X., Zhang, J.: A region-based image segmentation method under P systems. Journal of Information & Computational Science 10(10), 2943–2950 (2013)
147. Yazid, H., Arof, H.: Image segmentation using watershed transformation for facial expression recognition. In: IFMBE Proceedings, 4th Kuala Lumpur International Conference on Biomedical Engineering. pp. 575–578 (2008)
148. Ye, Q.Z., Danielsson, P.E.: Inspection of printed circuit boards by connectivity preserving shrinking. IEEE Transactions on Pattern Analysis and Machine Intelligence 10(5), 737–742 (Sep 1988)
149. Yuan, X., Situ, N., Zouridakis, G.: A narrow band graph partitioning method for skin lesion segmentation. Pattern Recognition 42(6), 1017–1028 (2009)
150. Zhang, G.X., Gheorghe, M., Li, Y.: A membrane algorithm with quantum-inspired subalgorithms and its application to image processing. Natural Computing 11(4), 701–717 (2012)
151. Zhang, T.Y., Suen, C.Y.: A fast parallel algorithm for thinning digital patterns. Communications of the ACM 27(3), 236–239 (Mar 1984)
152. Zhang, Z., Peng, H.: Object segmentation with membrane computing. Journal of Information & Computational Science 9(17), 5417–5424 (2012)
153. Zhao, W., Chellappa, R., Phillips, P.J., Rosenfeld, A.: Face recognition: A literature survey. ACM Computing Surveys 35(4), 399–458 (2003)

154. Zhou, Y., Chellappa, R.: Artificial neural networks for computer vision. Research notes in neural computing, Springer-Verlag (1992)
155. NVIDIA Corporation. NVIDIA CUDA^{tm} Programming Guide. `http://www.nvidia.com/` (2012)

P Colonies

Lucie Ciencialová[1], Erzsébet Csuhaj-Varjú[2], Luděk Cienciala[1], and Petr Sosík[1]

[1] Institute of Computer Science and Research Institute of the IT4Innovations Centre of Excellence, Silesian University in Opava, Czech Republic
{lucie.ciencialova, ludek.cienciala, petr.sosik}@fpf.slu.cz
[2] Department of Algorithms and Their Applications, Faculty of Informatics, Eötvös Loránd University, Budapest, Hungary
csuhaj@inf.elte.hu

Summary. P colonies are abstract computing devices modelling communities of very simple reactive agents (cells) living and acting in a joint shared environment. The concept was motivated by colonies, grammar systems based on interplay of very simple agents, on one hand, and by membrane systems, massively parallel computational models inspired by cell biology, on the other hand. Some variants of P colonies also allow the environment to participate actively in the system's evolution. In this paper we summarize the most important results on P colonies, present open problems concerning these constructs, and suggest new research directions in their study.

1 Introduction

In contemporary computer science, there has been a growing demand for reliable and efficient computing devices to describe the behaviour of communities of dynamically changing agents which are in interaction with their shared environment. Multi-agent systems with very simple reactive agents are of special interest, in particularly with respect to their emerging behaviour and the limits of their power.

P colonies, introduced in [25], were motivated by these problems. They are variants of very simple tissue-like P systems, where the agents (the cells) have only one region and they interact with their shared environment by using programs (collections of rules of special form). P systems (or membrane systems), introduced in [30], are a family of computing devices inspired by biology and biochemistry of cells. Colonies of simple formal grammars, also motivating P colonies, were introduced in [23].

During the years, P colonies have been studied in detail; a summary of results can be found in [26].

Although several variants of P colonies have been developed, all of them have some common basic features. Inside each agent (each cell) there is a finite multiset of objects. These objects are processed by a finite set of programs associated to the

agent. The number of objects inside each agent is constant (does not change) during the functioning of the agent community and it is called the capacity of the P colony. The agents share an environment which is represented by a multiset of objects. One type of these objects, called the environmental object, is distinguished, and it is supposed to be in a countably infinite number of copies in the environment. (In the literature, the reader may also find that the environmental symbol appears in an arbitrarily large number of copies in the environment).

Using their programs, the agents can change the objects present at their disposal and can exchange some of their objects with objects present in the environment. These synchronized actions correspond to a configuration change (a transition) of the P colony; a finite sequence of consecutive configuration changes starting from the initial configuration is a computation. The result of the computation is the number of copies of a distinguished object, called the final object, present in the environment in a final configuration of the P colony.

It can easily be seen that the environment is both a communication channel for the agents and a storage for the objects. It plays strategic role in synchronizing the work of the agents during the computation.

One major research topic in the theory of P colonies is the study of their computational power related to their descriptional complexity. These investigations focus on that questions how many components are necessary and how much extent the programs can be simplified to obtain a certain computational power. In addition to these problems, the working modes of P colonies have obtained attention as well, whether or not parallelism in the joint work of the agents plays significant role in increasing the expressive power of P colonies.

The rules of P colonies demonstrate strong similarities with instructions or rules of some well-known computing devices (register machines, rewriting systems based on point mutations, other variants of membrane systems), thus comparisons of these constructs with other classical and non-classical computing devices are also of interest.

P colonies, due to their original motivation, model multi-agent systems (complex systems) acting in an environment. According to the basic definitions, the objects present in the environment have significant role in the change of the states of the agents. Therefore, one of the research directions in the theory is devoted to studying the role of the dynamics of the environment in the behaviour of P colonies, i.e., the case when the objects in the environment are provided step-by-step not only by the actions of the agents but by some special object provider device.

Due to their simplicity and distributed nature, P colonies are convenient tools for modelling complex systems as robot collections, sender and consumer systems, eco-systems. We expect several new areas of applications in the future.

2 Notations

We assume that the reader is familiar with formal language and automata theory, computability, and the basics of membrane computing [31, 29].

Throughout the paper we use the following notions and notations. Let Σ be the alphabet and let Σ^* be the set of all words over Σ (including the empty word ε). The length of a word $w \in \Sigma^*$ is denoted by $|w|$ and the number of occurrences of the symbol $a \in \Sigma$ in w by $|w|_a$.

A multiset of objects M is a pair $M = (V, f)$, where V is an arbitrary (not necessarily finite) set of objects and f is a mapping $f : V \to N$; f assigns to each object in V its multiplicity in M. Each multiset of objects M with the set of objects $V' = \{a_1, \ldots a_n\}$ can be represented as a string w over alphabet V', where $|w|_{a_i} = f(a_i);\ 1 \leq i \leq n$. Obviously, all words obtained from w by permuting the letters represent the same multiset M. Symbol ε represents the empty multiset. The set of all multisets with the set of objects V is denoted by V^*. The cardinality of M, denoted by $|M|$, is defined by $|M| = \sum_{a \in V} f(a)$.

The set of all non-negative integers is denoted by N. We use REG, CF and RE as notations for the families of regular, context-free and recursively enumerable languages. The family of languages accepted by matrix grammars without appearance checking and with erasing rules is denoted by MAT^{ϵ} and the family of interactionless L systems is denoted by $0L$. NRE denotes the family of recursively enumerable set of non-negative integers.

Definition 1. [28] *A register machine is a construct* $M = (m, H, l_0, l_h, P)$ *where:*
- m *is the number of registers,*
- H *is the set of instruction labels,*
- l_0 *is the start label,*
- l_h *is the final label,*
- P *is a finite set of instructions injectively labeled with the elements from the set* H.

The instruction of the register machine are of the following forms:

$l_1 : (ADD(r), l_2, l_3)$	Add 1 to the content of the register r and proceed to the instruction (labeled with) l_2 or l_3.
$l_1 : (SUB(r), l_2, l_3)$	If the register r stores the value different from zero, then subtract 1 from its content and go to instruction l_2, otherwise proceed to instruction l_3.
$l_h : HALT$	Halt the machine. The final label l_h is only assigned to this instruction.

Without loss of generality, one can assume that in each ADD-instruction $l_1 : (ADD(r), l_2, l_3)$ and in each SUB-instruction $l_1 : (SUB(r), l_2, l_3)$ the labels l_1, l_2, l_3 are mutually distinct.

The register machine M computes a set $N(M)$ of numbers in the following way: it starts with all registers empty (hence storing the number zero) with the instruction labelled l_0 and it proceeds to apply the instructions as indicated by the labels

(and made possible by the contents of registers). If it reaches the halt instruction, then the number stored at that time in the register 1 is said to be computed by M and hence it is introduced in $N(M)$. It is known (see e.g. [28]) that in this way we compute all Turing computable sets of numbers.

The register machine is called partially bind if the *SUB*-instruction is executed as follows: If register r stores a non-zero value then this value is decreased by one and the next instruction will be l_2 or l_3, otherwise the computation aborts. When the partially blind register machine enters the final state, the result obtained in the first register is only taken into account if the remaining registers store value zero. The family of sets of non-negative integers generated by partially blind register machines is denoted by NRM_{pb}. The partially blind register machines accept a proper subfamily of NRE.

3 The basic model of P colonies

The original concept of a P colony was introduced in [25] and presented in a developed form in [24, 16].

Definition 2. *A P colony of capacity k, $k \geq 1$, is a construct*
$$\Pi = (A, e, f, v_E, B_1, \ldots, B_n), \text{ where}$$

- *A is an alphabet, its elements are called objects;*
- *$e \in A$ is the basic (or environmental) object of the colony;*
- *$f \in A$ is the final object of the colony;*
- *v_E is a finite multiset over $A - \{e\}$, called the initial state (or initial content) of the environment;*
- *B_i, $1 \leq i \leq n$, are agents, where each agent $B_i = (o_i, P_i)$ is defined as follows:*
  - *o_i is a multiset over A consisting of k objects, the initial state (or the initial content) of the agent;*
  - *$P_i = \{p_{i,1}, \ldots, p_{i,k_i}\}$ is a finite set of programs, where each program consists of k rules, which are in one of the following forms each:*
    - *$a \to b$, $a, b \in A$, called an evolution rule;*
    - *$c \leftrightarrow d$, $c, d \in A$, called a communication rule;*
    - *r_1/r_2, called a checking rule; r_1, r_2 are both evolution rules or both communication rules.*

We add some brief explanations to the components of the P colony.

We first note that throughout the paper, we use term "*object a inside agent B*" and term "*$a \in w$, where w is the state of agent B*" as equivalent.

The first type of rules associated to the programs of the agents, the *evolution rules*, are of the form $a \to b$. This means that object a inside the agent is rewritten to (evolved to be) object b.

The second type of rules, the *communication rules*, are of the form $c \leftrightarrow d$. If a communication rule is performed, then object c inside the agent and object d in

the environment agent swap their location. Thus, after executing the rule, object d appears inside the agent and object c is located in the environment.

The third type of rules are the checking rules. A checking rule is formed from two rules of one of the two previous types. If a checking rule r_1/r_2 is performed, then the rule r_1 has higher priority to be executed over the rule r_2. This means that the agent checks whether or not rule r_1 is applicable. If the rule can be executed, then the agent must use this rule. If rule r_1 cannot be applied, then the agent uses rule r_2.

We note that these types of rules are the basic ones; in some variants of P colonies other types of rules have been also considered. We will discuss them in later sections.

The program determines the activity of the agent: the agent can change its state and/or the state of the environment.

The environment is represented by a finite number (zero included) of copies of non-environmental objects and a countably infinite copies of the environmental object e.

In every step, each object inside the agent is affected by the execution of a program. Depending on the rules in the program, the program execution may affect the environment as well. *This interaction between the agents and the environment is the key factor of the functioning of the P colony.*

The functioning of the P colony starts from its initial configuration (initial state).

The *initial configuration* of a P colony is an $(n+1)$-tuple of multisets of objects present in the P colony at the beginning of the computation. It is given by the multisets o_i for $1 \leq i \leq n$ and by multiset v_E. Formally, the *configuration* of the P colony Π is given by $(w_1, \ldots, w_n, w_E)$, where $|w_i| = k,\ 1 \leq i \leq n$, w_i represents all the objects present inside the i-th agent, and $w_E \in (A - \{e\})^*$ represents all the objects in the environment different from the object e.

At each *step of the computation* (at each transition), the state of the environment and that of the agents change in the following manner: In the *maximally parallel* derivation mode, each agent which can use any of its programs should use one (non-deterministically chosen), whereas in the *sequential* derivation mode, one agent uses one of its programs at a time (non-deterministically chosen). If the number of applicable programs for an agent is higher than one, then the agent non-deterministically chooses one of the programs.

A sequence of transitions is called a *computation*. A computation is said to be halting, if a configuration is obtained where no program can be applied anymore. With a halting computation, we associate a *result* which is given as the number of copies of the objects f present in the environment in the halting configuration.

Because of the non-determinism in choosing the programs, starting from the initial configuration we obtain several computations, hence, with a P colony we can associate a set of numbers, denoted by $N(\Pi)$, computed by all possible halting computations of given P colony.

In the original model (see [25]) the number of objects inside each agent is set to two. Therefore, the programs were formed from only two rules. Moreover, the initial configuration was defined as $(n+1)$-tuple $(ee, \ldots, ee, \varepsilon)$ so at the beginning of the computation the environment of the P colony is "empty", it is without an input information.

The number of agents in a given P colony is called the degree of Π; the maximal number of programs of an agent of Π is called the height of Π and the number of the objects inside an agent is the capacity of Π. The family of all sets of numbers $N(\Pi)$ computed as above by P colonies of capacity at most $c \geq 0$, degree at most $n \geq 0$ and height at most $h \geq 0$, using checking programs, and working in the sequential mode is denoted by $NPCOL_{seq}K(c, n, h)$; whereas the corresponding families of P colonies working in the maximally parallel way are denoted by $NPCOL_{par}K(c, n, h)$. If one of the parameters n and h is not bounded, then we replace it with $*$. If only P colonies using programs without checking rules are considered, then we omit parameter K.

4 Computational power of P colonies

Although P colonies are very simple computing devices, due to their (mainly parallel) working mode and distributed nature they demonstrate large expressive (computational) power. In most cases, computational completeness can be obtained with these constructs even with a very few components and a very few restrictions on the programs. In this section, we briefly summarize some important results concerning their expressive power. Most of the statements are based on simulations of register machines, thus providing further knowledge on the nature of these classical computing devices as well.

To demonstrate a connection between P colonies and register machines, we add an example how the *ADD*-instruction of a register machine can be simulated by a P colony.

Example 1. Let $\Pi = (A, e, f, v_E, B)$ be the P colony with capacity two and let the current content of the agent be $l_1 e$. Let $M = (m, H, l_0, l_h, P)$ be a non-deterministic register machine with m registers. The *ADD*-instruction $l_1 = (ADD(r), l_2, l_3)$ of M can be simulated by the following programs associated with the agent:

P :

$1 : \langle l_1 \to l_1'; e \to a_r \rangle$;
$2 : \langle l_1' \to l_2; a_r \leftrightarrow e \rangle$;
$3 : \langle l_1' \to l_3; a_r \leftrightarrow e \rangle$;

At the beginning, objects l_1 and e are placed inside the agent. The content of register r is encoded to the number of objects a_r placed in the environment. The computation is done in such a way that the agent rewrites its content to $l_1' a_r$ by using the first program (there are two rewriting rules in it). In the second step the agents rewrites l_1' to object corresponding to the label of the next instruction l_2 (or l_3) to be executed and it puts object a_r into the environment.

4.1 Restricted P colonies

By [25], P colonies of capacity two are computationally complete. Furthermore, their programs have special forms: one of the rules is an evolution rule and the other one is either a communication rule or a checking rule with two communication rules.

These variants of P colonies are called *restricted* P colonies.

The family of all sets of numbers computed by restricted P colonies without checking rules and with parameters c, n, h and working modes *par* and *seq*, see above, is denoted by $NPCOL_{par}R(c, n, h)$ or $NPCOL_{seq}R(c, n, h)$, respectively. If the restricted P colonies are with checking rules, then we add K in front of R.

Let us have one more example.

Example 2. Let $\Pi = (A, e, f, v_E, B)$ be the P colony with capacity two and let the current contents of the agent be $l_1 e$. Let $M = (m, H, l_0, l_h, P)$ be a non-deterministic register machine with m registers. The *ADD*-instruction $l_1 = (ADD(r), l_2, l_3)$ can be simulated by the following programs associated with the agent:

P :

$1 : \langle e \to a_r; l_1 \leftrightarrow e \rangle$; $\quad 3 : \langle l_1 \to l_2; d \leftrightarrow e \rangle$;
$2 : \langle e \to d; a_r \leftrightarrow l_1 \rangle$; $\quad 4 : \langle l_1 \to l_3; d \leftrightarrow e \rangle$;

At the beginning, objects l_1 and e are inside the agent. The content of register r is encoded to the number of objects a_r placed in the environment. The computation is done in such a way that at one computational step the agent must rewrite one of the objects inside it and should exchange the other one with an object from the environment. At the first step the agent rewrites e to object a_r and sends object l_1 into the environment. In the second step it rewrites e to auxiliary object d and exchanges object a_r and l_1 from the environment. At the last step the agent rewrites object l_1 to the object corresponding to the label of the next instruction l_2 (or l_3) to be executed and it puts object d into the environment.

For restricted P colonies, using the maximally parallel working mode, the following results hold:

- $NPCOL_{par}KR(2, *, 5) = NRE$ in [16, 25],
- $NPCOL_{par}R(2, *, 5) = NRE$ in [18],
- $NPCOL_{par}K(2, *, 4) = NRE$ in [16],
- $NPCOL_{par}KR(2, 1, *) = NRE$ in [18],
- $NPCOL_{par}R(2, 2, *) = NRE$ in [8].

The reader can easily see that the family of sets of natural numbers computed by restricted P colonies with or without the use of checking rules having at most five programs associated with agent equals to *NRE*. If we remove the restriction on the type of rules in the programs, P colonies need only at most four programs associated with every agent to obtain computational completeness. The difference in the last two results demonstrates the power of checking rules and the power

of synchronized cooperation. To generate *NRE*, the restricted P colonies need only one agent if the agent can use checking rules and two agents if they are not equipped with checking rules.

The maximally parallel application of rules does not necessarily add power, as the following results demonstrate:

- $NPCOL_{seq}KR(2,*,5) = NRE$ in [18],
- $NPCOL_{seq}KR(2,1,*) = NRE$ in [18],
- $NPCOL_{seq}K(3,*,6) = NRE$ in [16, 24].

However, if only *restricted P colonies with the sequential working modes* are considered, the maximal computation power to be obtained is equal to the *recognition power of blind counter machines*, thus significantly reduced, irrespectively from the number of programs and agents in the P colony.

Notice that the property "restricted" demonstrates strong similarity to some normal forms of variants of regulated grammars, where some of the production is used for programming the action and some other production is responsible for its execution. By using some well-organized synchronizing mechanisms, simulation of standard P colonies with restricted ones can be demonstrated, thus, we may consider restricted P colonies as "normal forms" for the family of P colonies.

We note that the idea of restriction can be extended, with prescribing the ratio of evolution and communication rules in the programs of capacity k, $k \geq 2$.

4.2 Homogeneous P colonies

If each program in the P colony consists of rules of the same type (for a P colony with capacity two, this means that the program is formed from two evolution rules, or two communication rules or two checking rules of the same type), then we can call the P colony *homogeneous*.

Indicating by symbol H that homogeneous P colonies are considered, the following results were obtained:

- $NPCOL_{par}KH(2,*,4) = NRE$ in [9],
- $NPCOL_{par}KH(2,1,*) = NRE$ in [9].

As for the previous variants, we provide an example.

Example 3. Let $\Pi = (A, e, f, v_E, B)$ be the P colony with capacity two and let the current content of the agent be $l_1 e$. Let $M = (m, H, l_0, l_h, P)$ be a non-deterministic register machine with m registers. The ADD-instruction $l_1 = (ADD(r), l_2, l_3)$ can be simulated by the following programs associated with the agent:

P :

$1 : \langle l_1 \to l_1'; e \to a_r \rangle$;	$4 : \langle l_1' \to l_2; e \to e \rangle$;
$2 : \langle l_1' \leftrightarrow e; a_r \leftrightarrow e \rangle$;	$5 : \langle l_1' \to l_3; e \to e \rangle$;
$3 : \langle e \leftrightarrow l_1'; e \leftrightarrow e \rangle$;	

At the beginning, objects l_1 and e are placed inside the agent. The content of register r is encoded to the number of objects a_r placed in the environment. The computation is done in such a way that the agent at one computational step must rewrite all object inside it or must exchange all of its objects with objects from the environment. At the first step, the agent rewrites multiset $l_1 e$ to multiset $l'_1 a_r$. In the second step, it sends both objects of multiset $l'_1 a_r$ into the environment. At the third step, it consumes objects of multiset $l'_1 e$ and at the last step the agent rewrites object l_1 to the object corresponding to the label of the next instruction to be executed, namely, l_2 (or l_3).

The results have been recalled so far concern mainly P colonies with agents of capacity at least two. It is a challenging question, whether the work of agents with capacity one, i.e., with agents having one rule in each program can be organized in such way that they obtain the same power as P colonies in the general sense. Notice that in this case the objects play more important rule in the synchronization of the work of the agents. The following results give positive answer to this question.

- $NPCOL_{par}K(1, *, 5) = NRE$ in [10],
- $NPCOL_{par}KH(1, *, 6) = NRE$ in [9],
- $NPCOL_{par}K(1, 4, *) = NRE$ in [8],
- $NPCOL_{par}(1, 6, *) = NRE$ in [13].

Finally, we provide two more interesting results dealing with P colonies with capacity three [16, 4].

- $NPCOL_{par}K(3, *, 3) = NRE$ in [16],
- $NPCOL_{par}H(3, 2, *) = NRE$ in [4].

In Table 1 the reader can find a summarized list of results concerning the computational complete variants of P colonies.

5 P colonies with prescribed teams

P colonies with prescribed teams were introduced in [19]. Unlike the original variants of P colonies, the agents use finite sets of rules called teams instead of programs; with each agent a finite set of teams is given, with priorities (pri) among them. The used rules can be communicating (com), rewriting (rew), and so-called membrane rules (mem). The membrane rules are in a form $a \rightarrowtail b$ (a goes out and becomes b) or $b \rightarrowtail a$ (a goes in and becomes b).

The P colony can work in sequential (seq) or parallel (par) manner. The rules are applied by the team in parallel manner with various stop conditions: $*$ (stop after arbitrary number of derivation steps), $\leq l$, $\geq l$, resp. $= l$ (stop after at most l, at least l resp. after exactly l derivation steps) and t_0 (the team becomes inactive when it is no longer able to work as a team.)

n.	mode of comp.	capacity	degree	height	checking rules / restricted programs / homogeneous programs			
1.	*par*	1	∗	5	K			in [10]
2.	*par*	1	∗	6	K		H	in [9]
3.	*par*	1	4	∗	K			in [8]
4.	*par*	1	6	∗				in [13]
5.	*par*	1	3	∗				in [15]
6.	*par*	2	∗	8				in [16]
7.	*par*	2	∗	5	K	R		in [25, 16]
8.	*seq*	2	∗	5	K	R		in [18]
9.	*par*	2	∗	5		R		in [18]
10.	*par*	2	∗	4	K			in [16]
11.	*par*	2	∗	4	K		H	in [9]
12.	*seq*	2	∗	4	K			in [24]
13.	*seq/par*	2	1	∗	K	R		in [16, 18]
14.	*par*	2	2	∗		R		in [8]
15.	*seq/par*	2	1	∗	K		H	in [9]
16.	*par*	1	3	325	K		H	in [5]
17.	*par*	2	23	5	K	R		in [17]
18.	*par*	2	22	6	K	R		in [17]
19.	*par*	2	22	5	K			in [17]
20.	*par*	2	92	3			H	in [5]
21.	*par*	2	70	5			H	in [5]
22.	*seq/par*	2	1	74	K	R		in [5]
23.	*seq/par*	2	1	66	K			in [5]
24.	*par*	2	2	163			H	in [5]
25.	*par*	2	35	8				in [17]
26.	*par*	2	57	8		R		in [17]
27.	*par*	3	35	7				in [17]
28.	*seq/par*	3	∗	3	K			in [16, 24]
29.	*par*	3	2	∗			H	in [4]

Table 1. Computational complete classes of P colonies

At each step of the computation, the contents of the environment and the contents of every agent changes in the following way: in the maximally parallel derivation mode, each agent which can use any of its teams should use one (non-deterministically chosen) in the mode d, while in the sequential derivation mode, one agent uses one of its teams in the mode d at a time (non-deterministically chosen). As in the usual case, any copy of an object can be involved in only one rule. Using the teams as described above, with all agents acting simultaneously or sequentially, non-deterministically choosing the team(s) to be applied, the P colony changes its configuration.

In [19], the authors showed that the families of P colonies with prescribed teams are computationally complete if some conditions hold. These conditions concern,

for example, the working mode, the number of objects in the agents using rewriting and communication rules, the priority among the teams, the number of teams.

The following table summarizes the list of results on computational completeness of P colonies with prescribed teams which use rewriting and communication rules [19].

(In the tables, below, d indicates that the results hold for any of the modes.)

computational mode	*capacity*	*max. number of sets in the team*	*max. number of rules in the set*	*number of agents*	*number of teams*	*priorities*	*mode*
seq	2	2	1	$*$	6	*pri*	d
par	2	2	1	$*$	5		d
seq	2	2	1	1	$*$	*pri*	d
seq	2	2	2	1	$*$		t_0

The following table contains a list of results from [19] concerning P colonies using membrane rules only.

computational mode	*capacity*	*max. number of sets in the team*	*max. number of rules in the set*	*number of agents*	*number of teams*	*priorities*	*mode*
seq	2	2	1	$*$	12	*pri*	d
seq	2	2	1	1	$*$	*pri*	d
par	2	3	1	$*$	10		d
par	2	3	2	$*$	5		d
seq	1	2	2	1	$*$		t_0
seq	1	2	1	1	$*$	*pri*	t_0

6 P colonies with senders and consumers

In [13] new types of programs for P colonies with two objects inside each agent were introduced. The first of them is a deletion program — $\langle a_{in}; bc \to d \rangle$; using this program an agent consumes one object (a) from the environment and transforms the two objects (b, c) inside the agent into a new one (d). The second type is an insertion program in the form $\langle a_{out}; b \to cd \rangle$. By executing this program, the agent sends to the environment one object (a) and generates two new objects (c, d) from the other object (b).

Example 4. [13] (a) A P colony with one sender cell can generate the Parikh set of a regular language $L \subseteq T^*$. Let $G = (N, T, P, S)$ be a regular grammar such that $L(G) = L$.

For generating the Parikh vectors of the words in L, we use, for each $S \to aB$ of P, the programs $\langle e, out; e \to eS \rangle$, $\langle e, out; S \to aB \rangle$ and then $\langle x, out; A \to aB \rangle, x \in$

T for every $A \to aB$ in P. Finally, for every rule of the form $A \to a$ we need $\langle x, out; A \to aF\rangle, x \in T,\ \langle a, out; F \to FF\rangle,$ where $F \notin T \cup N$.

(b) A P colony with one consumer cell can "consume" the Parikh set of a regular language L. To see this, let $M = (Q, T, \delta, q_0, F)$ be a deterministic finite automaton such that $L(M) = L$.

We need the program $\langle e, in; ee \to q_0\rangle$, and to every transition $\delta(q_i, a) = q_j$ in M, we introduce $\langle a, in; xq_i \to q_j\rangle, x \in T \cup \{e\}$. If $q_j \in F$ in $\delta(q_i, a) = q_j$ we have to add the programs $\langle a, in; xq_i \to F\rangle, x \in T$, where $F \notin Q \cup T$.

In [4] the authors showed that P colonies with one sender and one consumer and some initial content in the environment are computationally complete. In [13] the authors proved that P colonies with senders and consumers with three agents and with only environmental objects in the initial configuration can generate every recursively enumerable set of natural numbers.

- $NPCOL_{sc}(3, *) = NRE$ in [13].
- $NPCOL_{sc}(2, *, ini) = NRE$ in [4, 14].

7 P colonies with evolving environment

The environment is static in the basic model, it can be changed only by the activity of the agents. Eco-P colonies were constructed as a natural extension of P colonies with dynamically evolving environment, the evolution does not depend only on the activity of agents. The mechanism of evolution in the environment is based on an 0L scheme. An 0L scheme is a pair (Σ, P), where Σ is the alphabet of 0L scheme and P is the set of context-free rules. It fulfils the following condition: for all $a \in \Sigma$ there exists $\alpha \in \Sigma^*$ such that $(a \to \alpha) \in P$. For $w_1, w_2 \in \Sigma^*$ we write $w_1 \Rightarrow w_2$ if $w_1 = a_1 a_2 \dots a_n, w_2 = \alpha_1 \alpha_2 \dots \alpha_n$, for $a_i \to \alpha_i \in P, 1 \le i \le n$.

Definition 3. *A generalized P colony with capacity $k \ge 1$ is a construct*

$$\Pi = (A, e, f, v_E, D_E, B_1, \dots, B_n),\ \textit{where}$$

- *A is the alphabet of the generalized P colony, its elements are called objects,*
- *e is the basic (environmental) object of the generalized P colony, $e \in A$,*
- *f is the final object of the generalized P colony, $f \in A$,*
- *v_E is the initial content of the environment, $v_E \in (A - \{e\})^*$,*
- *D_E is an 0L scheme (A, P_E), where P_E is the set of context-free rules,*
- *$B_i,\ 1 \le i \le n$, are the agents, every agent is a construct $B_i = (o_i, P_i)$, where o_i is a multiset over A, it defines the initial state (content) of agent B_i and $|o_i| = k$ and $P_i = \{p_{i,1}, \dots, p_{i,k_i}\}$ is the finite set of programs of three types $(a, b, c, d \in A)$:*
  - *(1) generating program with generating rules $a \to bc$ and transporting rules d out - the number of generating rules is the same as the number of transporting rules.*

(2) consuming program with consuming rules $ab \to c$ *and transporting rules* d *in - the number of consuming rules is the same as the number of transporting rules.*

(3) rewriting/communication program which can contain three types of rules:

- $\diamond$ $a \to b$*, called a rewriting rule,*
- $\diamond$ $c \leftrightarrow d$*, called a communication rule,*
- $\diamond$ r_1/r_2*, called a checking rule; each of* r_1, r_2 *is a rewriting or a communication rule.*

Every agent has only one type of programs. The agent with generating programs is called *sender* and the agent with consuming programs is called *consumer.* The capacity of a P colony with senders and consumers must be an even number.

The *initial configuration* of a P colony is the $(n+1)$-tuple $(o_1, \ldots, o_n, v_E)$, with the same interpretation of the symbols $o_1, \ldots, o_n, v_E$ as in Definition 3. In general, the *configuration* of the P colony Π is defined as $(n+1)$-tuple $(w_1, \ldots, w_n, w_E)$, where w_i represents the multiset of objects inside the i-th agent, $|w_i| = k$, $1 \leq i \leq n$, and $w_E \in (A - \{e\})^*$ is the multiset of objects different from e placed in the environment.

By applying programs, the generalized P colony passes from one configuration to some other configuration. Objects in the environment unaffected by any program in the given step are rewritten by the 0L scheme D_E. (Notice that in this case the 0L scheme is considered as a multiset rewriting mechanism). At each step, every agent tries to find one of its programs to apply. If the number of applicable programs is higher than one, then the agent non-deterministically chooses one program. At each step of, the set of active agents executing a program must be maximal, i.e., no further agent can be added to it.

A sequence of consecutive configurations starting from the initial configuration is called a *computation.* A configuration is *halting* if the P colony has no applicable program. Each halting computation has an associated a *result* – the number of copies of the final object placed in the environment in a halting configuration.

$$N(\Pi) = \{|w_E|_f \mid (o_1, \ldots, o_n, v_E) \Rightarrow^* (w_1, \ldots, w_n, w_E)\},$$

where $(o_1, \ldots, o_n, v_E)$ is the initial configuration, $(w_1, \ldots, w_n, w_E)$ is the final configuration, and $\Rightarrow^*$ denotes reflexive and transitive closure of $\Rightarrow$.

Let $NEPCOL(i, j, h, u, v, w)$ be the family of the sets of numbers computed by generalized P colonies with at most $j \geq 1$ agents with $i \geq 1$ objects inside the agent and with at most $h \geq 1$ programs associated with each agent such that:

$u = check$	if the P colony uses rewriting/communication rules with checking rules
$u = no\text{-}check$	if the P colony uses rewriting/communication rules without checking rules
$u = s/c/sc$	if the P colony contains only sender / only consumer / both sender and consumer agents
$v = pas$	if the rules of 0L scheme are of type $a \to a$ only,
$v = act$	if the set of rules of 0L scheme disposes of at least one rule of another type than $a \to a$,
$w = ini$	if the environment or agents contain initially objects different from e, otherwise w is omitted,

If a numerical parameter is not bounded, we use notation $*$.

Example 5. Let $M = (m, H, l_0, l_h, P)$ be a non-deterministic register machine with m registers. The ADD-instruction $l_1 = (ADD(r), l_2, l_3)$ will be simulated by the following rules:

ENV :	B :
$1 : l_1 i \to a_r l_1' D;$	$5 : \langle Pe \to P; \overline{l_2}\ in \rangle;$
$2 : l_1' \to \overline{l_2 l_3} D;$	$6 : \langle Pe \to P; \overline{l_3}\ in \rangle;$
$3 : \overline{l_2} \to l_2 D;$	$7 : \langle P\overline{l_2} \to P; e\ in \rangle;$
$4 : \overline{l_3} \to l_3 D;$	$8 : \langle P\overline{l_3} \to P; e\ in \rangle;$

The computation is done in such a way that the 0L scheme works in the environment, it adds one to the contents of register r (generates one copy of object a_r - the rule number 1) and generates objects l_2 and l_3, labels of all instructions which will be possibly executed in the next steps of computation of the register machine M (the rule 2). In the next step, consumer agent B takes one of these objects inside the agent - the rule 5 or 6. Then, instruction l_2 or l_3 will be simulated.

Generalized P colonies with two agents (senders and consumers) with passive environment (0L scheme contains the rules of type $a \to a$ only) are computationally complete. If the environment is active, then the family of generalized P colonies is computationally complete if the systems have two consumers and the initial contents of their environment is different from e.

- $NEPCOL(2, 2, *, sc, pas, ini) = NRE$ in [14],
- $NEPCOL(2, 2, *, c, act, ini) = NRE$ in [3],
- $NRM_{pb} \subseteq NEPCOL(2, 1, *, c, act, ini)$ in [15],
- $NEPCOL(1, 2, *, no\text{-}check, act, ini) = NRE$ in [15],
- $NRM_{pb} \subseteq NEPCOL(1, 1, *, check, act, ini)$ in [15].

Generalized P colonies can be related to other variants of P systems. In [15] it was shown that for an arbitrary extended catalytic P system with one catalyst there exists a generalized P colony with checking rules and one agent containing

one objects such that the two constructs determine the same set of numbers. Catalytic P systems are of interest in membrane computing, due to their relevance to chemical catalysts [29].

8 PCol automata

The basic motivation of P colonies was to model multi-agent systems with very simple agents interacting with their shared environment. The interaction was realized in communicating objects, and the description of the result of the activity of the P colony was defined as the multiset of distinguished objects in the environment when no more action could be performed.

Interaction of the environment and the collection of agents can also be described as the sequence of multisets of non-environmental agents that are found in the environment during the computation, i.e., the sequence of computational steps. From this point of view, the concept of a P colony can be extended to the notion of a PCol automaton (a P colony automaton), motivated by P automata from membrane computing [29] and classical finite automata [32].

In reference to the finite automaton, the concept of the P colony was extended by an input tape and the generating device was changed to an accepting one [7]. The agents of the P colony work according to the actual symbol read from the input tape. To do this, they have rules which can "read" the input tape, we call them tape rules or T-rules. The other rules, which are rules of standard P colonies, are called non-tape rules or N-rules. An input symbol is said to be read if at least one agent processed it (by using its corresponding T-rule).

Now we recall the notion of a PCol automaton.

Definition 4. *A PCol automaton of capacity k and with n agents, $k, n \geq 1$, is a construct $\Pi = (A, e, v_E, (o_1, P_1), \ldots, (o_n, P_n), F)$ where*

- *A is an alphabet, the alphabet of the PCol automaton, its elements are called objects;*
- *$e \in A$ is the environmental object of the PCol automaton; $v_E \in (A - \{e\})^*$ is a string representing the multiset of objects different from e, called the initial state of the environment ;*
- *$(o_i, P_i), 1 \leq i \leq n$, is the i-th agent; where*
  - *o_i is a multiset over V, the initial state (contents) of the agent,*
  - *P_i is a set of programs, where every program consists of k rules, each of them is one of the of the following types:*
    - *tape rules of the form $a \xrightarrow{T} b$ or $a \xleftrightarrow{T} b$, called rewriting tape rules and communication tape rules, respectively; or*
    - *non-tape rules of the form $a \rightarrow b$, or $c \leftrightarrow d$, called rewriting (non-tape) rules and communication (non-tape) rules, respectively.*

  and
- *F is a set of accepting configurations of the PCol automaton.*

For each i, $1 \leq i \leq n$, we distinguish tape programs and non-tape programs. The set of tape programs (T-programs), denoted by P_i^T, are formed from one tape rule and $k-1$ non-tape rules, the set of non-tape programs (N-programs) which contain only non-tape rules, is denoted by P_i^N, thus, $P_i = P_i^T \cup P_i^N$ and $P_i^T \cap P_i^N = \emptyset$.

The computation starts in the initial configuration, i.e., when the input word is on the input tape and all agents are in initial state.

For a configuration $(w_E, w_1, \ldots, w_n)$ and an input symbol a, the sets of applicable programs, $\mathcal{P}$, can be constructed. To pass from one configuration to some other one in one step we define the following types of transitions:

- *t-transition* $\Rightarrow_t^a$: There exists at least one set of applicable programs $P \in \mathcal{P}$ such that every $p \in P$ is T-program with T-rule in the form $x \overset{T}{\rightarrow} a$ or $x \overset{T}{\leftrightarrow} a, x \in A$ and the set P is maximal.
- *n-transition* $\Rightarrow_n$: There exists at least one set of applicable program $P \in \mathcal{P}$ such that every $p_i \in P$ is N-program and the set P is maximal.
- *tmin-transition* $\Rightarrow_{tmin}^a$: If exists at least one set of applicable program $P \in \mathcal{P}$ such that there is at least one T-program in P in the form $x \overset{T}{\rightarrow} a$ or $x \overset{T}{\leftrightarrow} a, x \in A$ and possibly N-programs. The set P is maximal.
- *tmax-transition* $\Rightarrow_{tmax}^a$: There exists at least one set of applicable program $P \in \mathcal{P}$ such that P contain as many T-programs (they are in a form $x \overset{T}{\rightarrow} a$ or $x \overset{T}{\leftrightarrow} a, x \in A$) as possible, P can contain N-programs too, and the set P is maximal.

We say that a PCol automaton works in *t (tmax, tmin) mode* of computation if it uses only *t- (tmax-, tmin-) transitions*. It works in *nt (ntmax or ntmin)* working mode if it uses *t-(tmax- or tmin-)* transitions and there is no set of applicable T-programs that can use *n*-transition. PCol automaton works in *init* mode if it performs only *t*-transitions and after reading all the input symbols it makes *n*-transitions.

If the PCol automaton works in *tn*, *tmax* or *tmin* mode, then it reads one input symbol in every step of computation. Consequently, the length of the computation equals to the length of the input string. Notice that this property strongly resembles to some property of ε-free finite automata.

The computation by a PCol automaton may end in a final state. It is successful if the whole input tape is read and the PCol automaton is in some configuration in F.

Let $M = \{t, nt, tmax, ntmax, tmin, ntmin, init\}$.

The *language accepted by a PCol automaton* Π, given as above, is defined as the set of strings which can be read during a successful computation:

$$L(\Pi, mode) = \{\, w \in A^* | (w; v_E, o_1, \ldots, o_n) \text{ can be transformed by } \Pi \text{ into } (\varepsilon; w_E, w_1, \ldots, w_n) \in F \text{ with a computation in mode } mode \in M \}.$$

Let $\mathcal{L}(PColA, mode)$ denote the class of languages accepted by PCol automata in the computational mode $mode \in M$.

Language classes of the Chomsky hierarchy can be described by PCol automata as follows [7].

- For every regular language L there exists a PCol automaton working in the t-mode having only one agent accepting all words from L.
- There exists a context-free language that can be accepted by a PCol automaton with only one agent and working in the t-mode.
- The family of languages accepted by PCol automata with one agent working in the t-mode is a subfamily of the family of context-sensitive languages.

It is open question whether the family of context-sensitive languages is equal to the family of languages accepted by PCol automata with one agent working in the t-mode. Notice that unlike other variants of P colonies PCol automata working in the t-mode are not computationally complete.

In [7], it was shown that class of languages accepted by PCol automata working in the *nt, ntmin* or *ntmax* mode equals to the class of recursively enumerable languages, respectively. The workspace needed to obtain this computational power is provided by the interaction between the agents and the environment.

- $\mathcal{L}(PColA, nt) = RE$ in [7],
- $\mathcal{L}(PColA, ntmin) = RE$ in [7],
- $\mathcal{L}(PColA, ntmax) = RE$ in [7],
- $\mathcal{L}(PColA, init) = NRE$ in [4].

In [22] the concept of a PCol automaton was generalized in the following manner: as in the case of the original model, the agents work in a maximally parallel way, but the agents are allowed to use any of their tape rules at every computation step. Those symbols that are read by the agents from the tape at a certain step of the computation form a multiset. If the symbols in this multiset form a prefix of the not-yet read part of the input, then this prefix of the input is considered to be read. The process continues until the whole string is read and no agent is able to apply any program.

The class of languages accepted by generalized PCol automata where all the communication rules are tape rules are denoted by $\mathcal{L}(genPCol, com - tape)$, the class of languages accepted by generalized PCol automata with all programs having at least one tape rule by $\mathcal{L}(genPCol, all - tape)$, and, finally, the class of languages accepted by generalized PCol automata where the programs are with arbitrary types of rules are denoted by $\mathcal{L}(genPCol, *)$.

In [22] it is shown that

- $\mathcal{L}(genPCol, com - tape) \cup \mathcal{L}(genPCol, all - tape) \subseteq \mathcal{L}(genPCol, *)$,
- $\mathcal{L}(genPCol, com - tape) \cap \mathcal{L}(genPCol, all - tape) - \mathcal{L}(CF) \neq \emptyset$,
- $\mathcal{L}(REG) \subset \mathcal{L}(genPCol, com - tape) \cap \mathcal{L}(genPCol, all - tape)$.

One of the most important results in [22] is

$$\mathcal{L}(genPCol, com - tape) \cup \mathcal{L}(genPCol, all - tape) \subseteq r - 1LOGSPACE,$$

where $r-1LOGSPACE$ is a proper subclass of the class of languages accepted by Turing machines with a one-way input tape using logarithmic space on work tapes.

The authors compare the accepting power of generalized PCol automata to standard P automata, constructs combining properties of antiport P systems and finite automata. For more information on P automata the reader is referred to Chapter 5 of [29].

9 APCol systems

In [6] the authors make one step further in combining properties of P colonies and automata. While the behaviour of the agents of PCol automata is determined both by the string to be processed and the environment consisting of multisets of symbols, in the case of APCol systems (Automaton-like P colonies), the agents act only on the input string. This interaction between the agents of the P colony and the input string is realized by exchanging symbols between the objects of the agents and that of the string (communication rules), and the states of the agents can change both by communication and evolution; the latter one is an application of a rewriting rule to an object. The distinguished symbol, e (in the previous models the environmental symbol) has a special role: whenever it is exchanged by a symbol in the environmental string, this symbol is erased. An evolution rule is of the form $a \rightarrow b$. It means that object a inside the agent is rewritten (evolved) to the object b. The second type of rules are called communication rules. A communication rule is in the form $c \leftrightarrow d$. When this rule is performed, the object c inside the agent and a symbol d in the string are exchanged, so, we can say that the agent rewrites symbol d to symbol c in the input string. If $c = e$, then the agent erases d from the input string and if $d = e$, symbol c is inserted into the string.

The computation in APCol systems starts with an input string, representing the environment, and with each agents having only symbols e in their state. (Note that the initial states of the agents can be chosen not to consist of only e.)

A computational step means a maximally parallel action of the active agents, i.e., agents that can apply their rules. Every symbol can be object of the action of only one agent. The computation ends if the input string is reduced to the empty word, there are no more applicable programs in the system, and meantime at least one of the agents is in so-called final state.

Definition 5. *An Automaton-like P colony (an APCol system, for short) is a construct*

$$\Pi = (A, e, B_1, \ldots, B_n),\ n \geq 1,\ \textit{where}$$

- *A is an alphabet; its elements are called the objects,*
- *$e \in A$, called the basic object,*
- *$B_i,\ 1 \leq i \leq n$, are agents. Each agent is a triplet $B_i = (o_i, P_i, F_i)$, where*
  - *o_i is a multiset over A, describing the initial state (content) of the agent, $|o_i| = 2$,*

- $P_i = \{p_{i,1}, \ldots, p_{i,k_i}\}$ *is a finite set of programs associated with the agent, where each program is a pair of rules. Each rule is in one of the following forms:*
  - $a \to b$, *where* $a, b \in A$, *called an evolution rule,*
  - $c \leftrightarrow d$, *where* $c, d \in A$, *called a communication rule,*
- $F_i \subseteq A^*$ *is a finite set of final states (contents) of agent* B_i.

As in the case of other variants of P colonies, the number of objects inside the agents are called the capacity of the APCol system, which is 2.

During the work of the APCol system, the agents perform programs. Since both rules in a program can be communication rules, an agent can work with two objects in the string in one step of the computation. In the case of program $\langle a \leftrightarrow b; c \leftrightarrow d\rangle$, a substring bd of the input string is replaced by string ac. If the program is of the form $\langle c \leftrightarrow d; a \leftrightarrow b\rangle$, then a substring db of the input string is replaced by string ca. That is, the agent can act only in one place in one step of the computation and the change of the string depends both on the order of the rules in the program and on the interacting objects. In particular, we have the following types of programs with two communication rules:

- $\langle a \leftrightarrow b; c \leftrightarrow e\rangle$ - b in the string is replaced by ac,
- $\langle c \leftrightarrow e; a \leftrightarrow b\rangle$ - b in the string is replaced by ca,
- $\langle a \leftrightarrow e; c \leftrightarrow e\rangle$ - ac is inserted in a non-deterministically chosen place in the string,
- $\langle e \leftrightarrow b; e \leftrightarrow d\rangle$ - bd is erased from the string,
- $\langle e \leftrightarrow d; e \leftrightarrow b\rangle$ - db is erased from the string,
- $\langle e \leftrightarrow e; e \leftrightarrow d\rangle$; $\langle e \leftrightarrow e; c \leftrightarrow d\rangle$, ... - these programs can be replaced by programs of type $\langle e \to e; c \leftrightarrow d\rangle$.

At the beginning of the computation of the APCol system the environment is given by a string ω of objects which are different from e. Consequently, an initial configuration of the APCol system is an $(n+1)$-tuple $c = (\omega; o_1, \ldots, o_n)$ where ω is the input string and the other n components are multisets of strings of objects, given in the form of strings, the initial states the of agents.

A configuration of an APCol system Π is given by $(w; w_1, \ldots, w_n)$, where $|w_i| = 2,\ 1 \leq i \leq n$, w_i represents the state of the i-th agent and $w \in (A - \{e\})^*$ is the string to be processed.

At each step of the (parallel) computation every agent attempts to find one of its programs to use. If the number of applicable programs is higher than one, the agent non-deterministically chooses one of them. At every step of the computation, the maximal possible number of agents have to be active, i.e., have to perform a program.

By computation steps given above, the APCol system performs a transition, i.e., it passes from one configuration to some other configuration. A sequence of configurations started from the initial configuration is called a computation. A configuration is halting if the APCol system has no applicable program. A computation is called accepting if and only if at least one agent is in final state and

the string to be processed is ε. Hence, the string ω is accepted by the APCol Π if there exists a computation by Π such that it starts in the initial configuration $(\omega; o_1, \ldots, o_n)$ and the computation ends by halting in the configuration $(\varepsilon; w_1, \ldots, w_n)$, where at least one of $w_i \in F_i$ for $1 \leq i \leq n$. The language $L_{acc}(\Pi)$ accepted by Π is the set of words over $(A - \{e\})$ which are accepted by Π.

APCol systems are powerful computational devices as it is shown in [6]:

Let A be an alphabet and let $L \subseteq A^*$ be a recursively enumerable language. Let $L' = S \cdot L \cdot E$, where $S, E \notin A$. Then there exists an APCol system Π with two agents such that $L' = L(\Pi)$ holds.

APCol systems can also be used not only for accepting but generating strings. A string w_F is generated by an APCol system Π if there exists a computation starting in an initial configuration $(\varepsilon; ee, \ldots, ee)$ and the computation ends by halting in configuration $(w_F; w_1, \ldots, w_n)$, where $w_i \in F_i$ for at least one w_i, $1 \leq i \leq n$. The language $L_{gen}(\Pi)$ generated by Π is the set of words over $(A - \{e\})$ which are generated by Π.

In particularly important are those variants, where the programs are restricted (as defined for standard P colonies).

We denote by $APCol_{acc}R(n)$ (or $APCol_{acc}(n)$) the family of languages accepted by APCol systems having at most n agents, $n \geq 1$, with restricted programs only (or without this restriction). Analogously, we denote by $APCol_{gen}R(n)$ the family of languages generated by APCol systems having at most n agents, $n \geq 1$, with restricted programs only, and $APCol_{gen}(n)$ denotes the case when the programs are without any restriction.

We may associate sets of numbers to APCol systems working in the generating or the accepting mode in the usual manner.

For an APCol system Π, $NL_{acc}(\Pi)$ and $NL_{gen}(\Pi)$ denote the length sets of $L_{acc}(\Pi)$ and $L_{gen}(\Pi)$, respectively. The family of length sets of languages accepted or generated by restricted APCol systems with at most n agents, $n \geq 1$, is denoted by $NAPCol_xR(n)$, $x \in \{acc, gen\}$, respectively, and $NAPCol_x(n)$ denotes the case when the programs are without any restriction.

The following results were obtained in [6]:

- $NAPCol_{gen}R(2) = NRE$.
- $NRM_{pb} \subseteq NAPCol_{gen}R(1)$.
- $APCol_{gen}R(1) \subseteq MAT^{\epsilon}$.
- $APCol_{gen}R(1) \subset MAT^{\epsilon}$.

10 2D P colonies

In [11] a new model, called 2D P colony was introduced. As in the original model, the P colony is of capacity two and the agents are equipped with sets of the programs formed from rules – communication and evolution. The main change is in the environment. Namely, the authors put the agents into the 2D grid of square cells and they provide the agent with the possibility to move – the motion rule.

The direction of the movement of the agent is determined by the contents of cells surrounding the cell in which the agent is placed.

The program can contain at most one motion rule. To achieve the greatest simplicity in agent behaviour, one other condition was set. If the agent moves, it cannot communicate with the environment. So if the program contains a motion rule, then the other rule is an evolution rule.

Definition 6. *A 2D P colony is a construct*

$$\Pi = (A, e, Env, B_1, \ldots, B_k, f), k \geq 1, \text{ where}$$

- A *is an alphabet of the colony, its elements are called objects,*
- $e \in A$ *is the basic environmental object of the 2D P colony,*
- Env *is a pair* $(m \times n, w_E)$*, where* $m \times n, m, n \in N$ *is the size of the environment and* w_E *is the initial contents of environment, it is a matrix of size* $m \times n$ *of multisets of objects over* $A - \{e\}$*.*
- $B_i,\ 1 \leq i \leq k$*, are agents, each agent is a construct* $B_i = (o_i, P_i, [o, p]),\ 0 \leq o \leq m,\ 0 \leq p \leq n$*, where*
  - o_i *is a multiset over* A*, it determines the initial state (contents) of the agent,* $|o_i| = 2$*,*
  - $P_i = \{p_{i,1}, \ldots, p_{i,l_i}\}, l \geq 1, 1 \leq i \leq k$ *is a finite set of programs, where each program contains exactly* 2 *rules, which are in one of the following forms each:*
    - $a \to b$*, called the evolution rule,* $a, b \in A$*,*
    - $c \leftrightarrow d$*, called the communication rule,* $c, d \in A$*,*
    - $[a_{q,r}] \to s, 0 \leq q, r \leq 2, s \in \{\Leftarrow, \Rightarrow, \Uparrow, \Downarrow\}$*, called the motion rule,*
- $f \in A$ *is the final object of the colony.*

A configuration of the 2D P colony is given by the state of the environment - matrix of type $m \times n$ with multisets of objects over $A - \{e\}$ as its elements, and by the state of all agents - pairs of objects from alphabet A and the coordinates of the agents. An initial configuration is given by the definition of the 2D P colony.

A computational step consists of three parts. The first part lies in determining the applicable set of programs according to the actual configuration of the 2D P colony. There are programs belonging to all agents in this set of programs. In the second part we have to choose one program corresponding to each agent from the set of applicable programs. There is no collision between the communication rules belonging to different programs. The third part is the execution of the chosen programs.

A change of the configuration is triggered by the execution of programs and it involves changing the state of the environment, contents and placement of the agents.

A computation is non-deterministic and maximally parallel. The computation ends by halting when no agent has an applicable program.

The result of the computation is the number of copies of the final object placed in the environment at the end of the computation.

The aim of introducing 2D P colonies is not studying their computational power but monitoring their behaviour during the computation.

In [11] an example for a 2D P colony simulating a kind of cellular automata – Conway's Game of Life ([20]) is presented. The following example is the pattern called beacon (see Fig. 1).

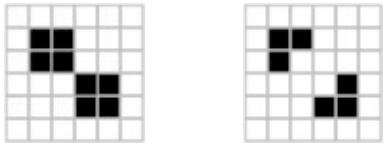


Fig. 1. Pattern beacon changes in two consecutive steps

Let Π_2 be 2D P colony defined as follows: $\Pi_2 = (A, e, Env, B_1, \ldots, B_{16}, f)$, where

- $A = \{e, f, D, S, Z, M, O, L, N\}$,
- $e \in A$ is the basic environmental object of the 2D P colony,
- $Env = (6 \times 6, w_E)$,
- $w_E = \begin{bmatrix} D & D & D & D & D & D \\ D & S & S & D & D & D \\ D & S & S & D & D & D \\ D & D & D & S & S & D \\ D & D & D & S & S & D \\ D & D & D & D & D & D \end{bmatrix}$,
- $B_1 = (ee, P_1, [1,1])$, $B_2 = (ee, P_2, [1,2])$,..., $B_{16} = (ee, P_{16}, [4,4])$,
- $f \in A$ is the final object of the 2D P colony.

The states of the automata are stored inside the cells (D - dead automaton, S - live automaton). There is only one kind of agent in this 2D P colony, so there are sixteen identical agents located in the matrix 4×4 of inner cells with following programs:

The first program is to initialize the agent $\langle e \leftrightarrow e;\ e \rightarrow Z \rangle$;

We sort the programs using the number of copies of object S in the condition of the motion rule.

1. when neighbouring automata are dead - a single program for both dead as well as alive automaton $\left\langle \begin{bmatrix} D & D & D \\ D & e & D \\ D & D & D \end{bmatrix} \rightarrow \Uparrow;\ Z \rightarrow M \right\rangle$.

2. when there is one alive neighbouring automaton - there are eight possible programs for dead and alive automata $\left\langle \begin{bmatrix} S & D & D \\ D & e & D \\ D & D & D \end{bmatrix} \rightarrow \Uparrow;\ Z \rightarrow M \right\rangle$ and seven other combinations.
3. when there are two alive neighbouring automata - twenty-eight programs for alive automata $\left\langle \begin{bmatrix} S & S & D \\ D & S & D \\ D & D & D \end{bmatrix} \rightarrow \Uparrow;\ Z \rightarrow O \right\rangle$ and other twenty-seven combinations.
4. when there are two alive neighbouring automata - twenty-eight programs for dead automata $\left\langle \begin{bmatrix} S & S & D \\ D & D & D \\ D & D & D \end{bmatrix} \rightarrow \Uparrow;\ Z \rightarrow M \right\rangle$ and other twenty-seven combinations.
5. when there are three alive neighbouring automata - fifty-six eight possible programs for dead and alive automata $\left\langle \begin{bmatrix} S & S & S \\ D & e & D \\ D & D & D \end{bmatrix} \rightarrow \Uparrow;\ Z \rightarrow O \right\rangle$ and other fifty-five combinations.
6. when there are four alive neighbouring automata - eight possible programs for dead and alive automata $\left\langle \begin{bmatrix} S & S & S \\ S & e & D \\ D & D & D \end{bmatrix} \rightarrow \Uparrow;\ Z \rightarrow M \right\rangle$ and other sixty-nine combinations.
7. when there are at least five alive neighbouring automata - fifty- eight possible programs for dead and alive automata $\left\langle \begin{bmatrix} S & S & S \\ S & e & S \\ * & * & * \end{bmatrix} \rightarrow \Uparrow;\ Z \rightarrow M \right\rangle$ and other fifty-five combinations.

After executing one of the above programs, all agents move one step forward and rewrite one of their objects e to object M (automaton will be dead) or to object O (automaton will be live). The following programs are for downward movement and for updating the state of an automaton - i.e., the replacement of the object in the cell for an object in the agent to change the state of the automaton.

$\left\langle \begin{bmatrix} * & * & * \\ * & e & * \\ * & * & * \end{bmatrix} \rightarrow \Downarrow;\ O \rightarrow S \right\rangle; \left\langle \begin{bmatrix} * & * & * \\ * & e & * \\ * & * & * \end{bmatrix} \rightarrow \Downarrow;\ M \rightarrow D \right\rangle;$

$\langle e \rightarrow\ L;\ S \leftrightarrow S \rangle\,; \langle e \rightarrow\ L;\ D \leftrightarrow S \rangle\,; \langle S \rightarrow\ e;\ L \rightarrow e \rangle\,; \langle D \rightarrow\ e;\ L \rightarrow e \rangle\,;$
$\langle e \rightarrow\ L;\ S \leftrightarrow D \rangle\,; \langle e \rightarrow\ L;\ D \leftrightarrow D \rangle\,.$

(see Fig.2).

$$\begin{bmatrix} D & D & D & D & D & D \\ D & S(ee) & S(ee) & D(ee) & D(ee) & D \\ D & S(ee) & S(ee) & D(ee) & D(ee) & D \\ D & D(ee) & D(ee) & S(ee) & S(ee) & D \\ D & D(ee) & D(ee) & S(ee) & S(ee) & D \\ D & D & D & D & D & D \end{bmatrix} \begin{bmatrix} D & D & D & D & D & D \\ D & S(eZ) & S(eZ) & D(eZ) & D(eZ) & D \\ D & S(eZ) & S(eZ) & D(eZ) & D(eZ) & D \\ D & D(eZ) & D(eZ) & S(eZ) & S(eZ) & D \\ D & D(eZ) & D(eZ) & S(eZ) & S(eZ) & D \\ D & D & D & D & D & D \end{bmatrix}$$

$$\begin{bmatrix} D & D(eO) & D(eO) & D(eM) & D(eM) & D \\ D & S(eO) & S(eM) & D(eM) & D(eM) & D \\ D & S(eM) & S(eM) & D(eM) & D(eO) & D \\ D & Ds(eM) & D(eM) & S(eO) & S(eO) & D \\ D & D & D & S & S & D \\ D & D & D & D & D & D \end{bmatrix} \begin{bmatrix} D & D & D & D & D & D \\ D & S(eS) & S(eS) & D(eD) & D(eD) & D \\ D & S(eS) & S(eD) & D(eD) & D(eD) & D \\ D & D(eD) & D(eD) & S(eD) & S(eS) & D \\ D & D(eD) & D(eD) & S(eS) & S(eS) & D \\ D & D & D & D & D & D \end{bmatrix}$$

$$\begin{bmatrix} D & D & D & S & S & D \\ D & S(LS) & S(LS) & D(LD) & D(LD) & D \\ D & S(LS) & D(LS) & D(LD) & D(LD) & D \\ D & D(LD) & D(LD) & D(LS) & S(LS) & D \\ D & D(LD) & D(LD) & S(LS) & S(LS) & D \\ D & D & D & D & D & D \end{bmatrix} \begin{bmatrix} D & D & D & D & D & D \\ D & S(ee) & S(ee) & D(ee) & D(ee) & D \\ D & S(ee) & D(ee) & D(ee) & D(ee) & D \\ D & D(ee) & D(ee) & D(ee) & S(ee) & D \\ D & D(ee) & D(ee) & S(ee) & S(ee) & D \\ D & D & D & D & D & D \end{bmatrix}$$

Fig. 2. The sequence of configurations of the 2D P colony simulating beacon

11 Applications of P colonies

Robot controllers

P colonies and PCol automata were introduced as robot controllers in [2]. The authors followed two ideas of controlling a robot with use of P colonies.

The first controller model used the PCol automaton with instructions for the robot on the input tape. The agents have to read the current information from the tape and together with objects in the environment coming from the receptors, they generate objects - commands for actuators.

The agents are assembled into modules. All the modules are controlled by the main control unit. Each input symbol on the input tape represents a single instruction which has to be done by the robot, so the input string is the sequence of the actions which guides the robot in reaching its goal; performing all the actions. In this meaning the computation ends by halting, and it is successful if the whole input tape is read.

The second idea was to use original model of the P colony and put all information to the environment. They also used module-oriented structure of agents, each module performs the individual functions in the control of the robot. The authors constructed a P colony with four modules: The control unit, the left actuator controller, the right actuator controller and the infra-red receptors. The controller (the P colony) is completed by the input and output filter. The input filter codes signals from the robots receptors and spread the coded signal into the environment. In the environment there is the coded signal used by the agents. The output

filter decodes the signal from the environment which the actuator controllers sent into it. Decoded signal is forwarded to the robots actuators.

Surface runoff

2D P colonies appear to be suitable to simulate multi-agent systems. In [1] the authors presented hydrological modelling flow of liquid over the earth's surface using 2D P colonies.

The issue of the flow of liquid over the Earth's surface is studied by experts from two areas - hydrology and geoinformatics. Both of these disciplines work closely together on the issue of the so-called "surface runoff". Surface runoff is the water flow that occurs when the soil is saturated to full capacity and excess water from rain, meltwater, or other sources flows over the land.

Agents in the model have capacity 2, the agent contains two objects. Each of the objects carries the information about the state of the agent. One of the objects stores information about the activity of the agent. At this stage of the simulation it is the information that the agent "flows" down the terrain or the agent is still inactive (belonging to the rainfall that have not fall). The other object stores information about the previous direction of flow. This information can further modify the way of the agent as inertia.

Based on the entered data - the slope surface, a source of fluid and quantity - they simulated the fluid distribution in the environment.

The research continues in [12] where the model can fill sinks (places without output).

12 Open problems and conclusions

We recalled the idea and functioning of the basic model of P colonies. This model was introduced in [25] in 2004. Since that time many papers and studies about the model and its variations have been published.

Almost all these works are focused on describing the computational power of more or less restricted variants of P colonies. Although extensive investigations have been made in this direction, some basic questions have remained open: what about deterministic P colonies, furthermore how to define determinism in P colonies. Notice that determinism can be defined in several levels of the syntax and the semantics of the system. Another interesting question can be the problem of reversibility: how to define reversible P colonies and, thus, reversible computation in P colonies?

Automaton-like P colonies (PCol automata and their variants) are topics of further study as well. Although they have been compared to some classical and non-classical automata variants, precise descriptions of their relation to P automata and to some further variants of classical and non-classical automata would be very useful. Several other, related open problems can be found in [21].

P colonies can also be compared to several types of P systems, in particularly tissue-like constructs. Generalized communicating P systems, where the rules of this tissue-like membrane systems describe the move of pairs of objects from pairs of compartments to new locations (each object of the pair is moved to a new compartment) demonstrate functional similarity to P colonies. Initial steps in this direction have been made in [27].

The second part of this survey was dedicated to the extension of the P colonies called 2D P colonies. This model was found suitable for simulations of multi-agent systems. One of the simulation introduced in [1] is the simulation of surface runoff.

In the future, many ways appear for improving the model of the 2D P colonies. One way is to assign the number to objects in addition to the type. This number will indicate the value of the parameter that the object represents. Another possibility is to extend the environment with mechanism which is able to change the object in the environment independently from the activity of the agents.

The concepts and results reported in this survey demonstrate that P colonies (and their variants) and are simple but very powerful devices. In addition, they can serve as modelling tools as well. The reader is welcome to contribute in exploring this fruitful research area.

Acknowledgments.

This work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPU II) project IT4Innovations excellence in science - LQ1602, and by the Silesian University in Opava under the Student Funding Scheme, project SGS/13/2016. The work of Erzsébet Csuhaj-Varjú was supported by NKFIH (National Research, Development, and Innovation Office), Hungary, grant no. K 120558.

References

1. Cienciala, L., Ciencialová, L., Langer, M.: Modelling of surface runoff using 2D P colonies. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8340, 101–116 (2014)
2. Cienciala, L., Ciencialová, L., Langer, M., Perdek, M.: The abilities of P colony based models in robot control. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 8961, 179–193 (2014)
3. Cienciala, L., Ciencialová, L.: Eco-P colonies. In: Păun, Gh., Pérez-Jiménez, M., Riscos-Núñez, A. (eds.) Pre-Proceedings of the 10th Workshop on Membrane Computing. pp. 201–209. Curtea de Arges, Romania (2009)
4. Cienciala, L., Ciencialová, L.: P colonies and their extensions. In: Kelemen, J., Kelemenová, A. (eds.) Computation, Cooperation, and Life - Essays Dedicated to Gheorghe Păun on the Occasion of His 60th Birthday. Lecture Notes in Computer Science, vol. 6610, pp. 158–169. Springer (2011)

5. Cienciala, L., Ciencialová, L.: Some new results of P colonies with bounded parameters. Natural Computing pp. 1–12 (2016)
6. Cienciala, L., Ciencialová, L., Csuhaj-Varjú, E.: P colonies processing strings. Fundamenta Informaticae 134(1-2), 51–65 (2014)
7. Cienciala, L., Ciencialová, L., Csuhaj-Varjú, E., Vaszil, Gy.: PCol automata: recognizing strings with P colonies. In: Ángel del Amor, M., Păun, Gh., Pérez Hurtado de Mendoza, I., Riscos Núnez, A. (eds.) Eighth brainstorming week on membrane computing. Sevilla, 2010. (RGNC Report 01/2010.). pp. 65–76. Fénix Editora, Sevilla (2010)
8. Cienciala, L., Ciencialová, L., Kelemenová, A.: On the number of agents in P colonies. Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 4860, 193–208 (2007)
9. Cienciala, L., Ciencialová, L., Kelemenová, A.: Homogeneous P colonies. Computing and Informatics 27(3+), 481–496 (2008)
10. Cienciala, L., Ciencialová, L., Langer, M.: Modularity in P colonies with checking rules. In: Gheorghe, M., Păun, Gh., Rozenberg, G., Salomaa, A., Verlan, S. (eds.) Lecture Notes in Computer Science:Membrane Computing: 12th International Conference, CMC 2011, Fontainebleau, France, August 23-26, 2011, Revised Selected Papers, vol. 7184, pp. 104–119. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
11. Cienciala, L., Ciencialová, L., Perdek, M.: 2D P colonies. In: Csuhaj-Varjú, E., Gheorghe, M., Rozenberg, G., Salomaa, A., Vaszil, Gy. (eds.) Lecture Notes in Computer Science: Membrane Computing - 13th International Conference, CMC 2012, Budapest, Hungary, August 28-31, 2012, Revised Selected Papers. Lecture Notes in Computer Science, vol. 7762, pp. 161–172. Springer (2012)
12. Ciencialová, L., Cienciala, L., Perdek, M.: 2D P colonies used in hydrology simulations. International Multidisciplinary Scientific GeoConference Surveying Geology and Mining Ecology Management, SGEM 1(2), 3–10 (2014)
13. Ciencialová, L., Csuhaj-Varjú, E., Kelemenová, A., Vaszil, Gy.: Variants of P colonies with very simple cell structure. International Journal of Computers, Communications and Control 4(3), 224–233 (2009)
14. Ciencialová, L., Cienciala, L., Sosík, P.: Generalized P colonies with passive environment. In: Graciani, C., Orellana-Martín, D., Riscos-Núñez, A., Romero-Jiménez, Á., Valencia-Cabrera, L. (eds.) Fourteen Brainstorming Week on Membrane Computing. pp. 151–162 (2016)
15. Ciencialová, L., Cienciala, L., Sosík, P.: P colonies with evolving environment. In: Leporati, A., Zandron, C. (eds.) Proceedings of the 17th International Conference on Membrane Computing (CMC17). pp. 105–118 (2016)
16. Csuhaj-Varjú, E., Kelemen, J., Kelemenová, A., Păun, Gh., Vaszil, Gy.: Computing with cells in environment: P colonies. Journal of Multiple-Valued Logic and Soft Computing 12(3-4 SPEC. ISS.), 201–215 (2006)
17. Csuhaj-Varjú, E., Margenstern, M., Vaszil, Gy.: P colonies with a bounded number of cells and programs. In: Hoogeboom, H.J., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing, 7th International Workshop, WMC 2006, Leiden, The Netherlands, July 17-21, 2006, Revised, Selected, and Invited Papers. Lecture Notes in Computer Science, vol. 4361, pp. 352–366. Springer (2006)
18. Freund, R., Oswald, M.: P colonies working in the maximally parallel and in the sequential mode. In: Zaharie, D., Petcu, D., Negru, V., Jebelean, T., Ciobanu, G., Cicortas, A., Abraham, A., Paprzycki, M. (eds.) Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC 2005),

25-29 September 2005, Timisoara, Romania. pp. 419–426. IEEE Computer Society (2005)
19. Freund, R., Oswald, M.: P colonies and prescribed teams. International Journal of Computer Mathematics 83(7), 569–592 (2006), http://dx.doi.org/10.1080/00207160601065389
20. Gardner, M.: Mathematical Games: The fantastic combinations of John Conway's new solitaire game "life". Scientific American 223, 120–123 (1970)
21. Gheorghe, M., Păun, G., Pérez-Jiménez, M.J., Rozenberg, G.: Research frontiers of membrane computing: Open problems and research topics. Int. J. Found. Comput. Sci. 24(5), 547–624 (2013), Section 5. P Colonies and dP Automata by Erzsébet Csuhaj-Varjú.
22. Kántor, K., Vaszil, G.: Generalized P colony automata. Jornal of Automata, Language and Combinatorics 19(1–4), 145–156 (2014)
23. Kelemen, J., Kelemenová, A.: A grammar-theoretic treatment of multiagent systems. Cybern. Syst. 23(6), 621–633 (November 1992)
24. Kelemen, J., Kelemenová, A.: On P colonies, a biochemically inspired model of computation. In: Proc. of the 6^{th} International Symposium of Hungarian Researchers on Computational Intelligence, Budapest TECH. pp. 40–56. Hungary (2005), http://conf.uni-obuda.hu/mtn2005/Kelemen.pdf
25. Kelemen, J., Kelemenová, A., Păun, Gh.: Preview of P colonies: A biochemically inspired computing model. In: Workshop and Tutorial Proceedings. Ninth International Conference on the Simulation and Synthesis of Living Systems (Alife IX). pp. 82–86. Boston, Massachusetts, USA (September 12-15 2004)
26. Kelemenová, A.: P Colonies, chap. 23.1, pp. 584–593. In: [29] (2010)
27. Krishna, S.N., Gheorghe, M., Ipate, F., Csuhaj-Varjú, E., Ceterchic, R.: Further results on generalised communicating P systems (2016), submitted
28. Minsky, M.L.: Computation: Finite and Infinite Machines. Prentice-Hall, Inc., Upper Saddle River, NJ, USA (1967)
29. Păun, Gh., Rozenberg, G., Salomaa, A.: The Oxford Handbook of Membrane Computing. Oxford University Press, Inc., New York, NY, USA (2010)
30. Păun, G.: Computing with membranes. J. Comput. Syst. Sci. 61(1), 108–143 (2000)
31. Rozenberg, G., Salomaa, A.: Handbook of Formal Languages: Beyonds words. Handbook of Formal Languages, Springer (1997)
32. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages, Vol. 1: Word, Language, Grammar. Springer-Verlag New York, Inc., New York, NY, USA (1997)

Technical Notes, Open Problems, Inquiries, Answers

Multiset Generalization of Balancing Chemical Reactions

Vincenzo Manca

University of Verona, Italy
vincenzo.manca@univr.it

2-MBP: 2-Multiset Balancing Problem

Two (finite) multisets of multisets M_1, M_2 over a set of atoms X, shortly 2-multisets over X, are balanced when in their respective sums $\sum M_1, \sum M_2$ each atom of X occurs with the same multiplicity. For example the following pair of 2-multisets (subscripts and multiplicative coefficient are the multiplicities of the two levels) are balanced:

$$2a_2b_3 + 3abc_4, 4c_3 + 3b_3a + 2a_2$$

Let us consider a simple chemical reaction, the combustion of methane, which is essentially an oxidation where CH_4 reacts with the Oxygen molecule O_2. The subscript index of atom symbol denotes its multiplicity in the molecule, while the coefficients of molecules, are put on the left of the molecules (multiplicity 1 is implicit when no coefficient is indicated). Therefore, the methane combustion is correctly balanced when a methane molecule combines with two Oxygen molecules, by giving as products a Carbon dioxide molecule plus two water molecules:

$$CH_4 + 2O_2 \rightarrow CO_2 + 2H_2O.$$

We adopt the same style by using abstract atoms denoted by lower case letters $a, b, c \ldots$ Whence the reaction above is of type:

$$ab_4 + 2c_2 \rightarrow ac_2 + 2b_2c.$$

We again remark that the transformation above as any chemical reaction is between a multiset of multisets over an alphabet into another multiset of multisets

over the same alphabet, that is, a (finite) second-order multiset transforms into another second-order multiset (over a finite alphabet). In passing, it is interesting to consider that many chemical laws follow from the simple fact that molecules are finite multisets of atoms and atoms cannot be created or destroyed in chemical reactions (disregarding here the isomeric molecular forms).

Referring to the example above, given the two sets $A = \{ab_4, c_2\}$ and $B = \{ac_2, b_2c\}$, we search for the positive integers multiplicities x_1, x_2 of the elements of A and y_1, y_2 of the elements of B such that the corresponding multisets $x_1ab_4+x_2c_2$ and $y_1ac_2 + y_2b_2c$ respectively have the same number of a, b, c.

In general, the 2-multiset balancing problem can be stated with the following abstract chemical formulation. Given two sets A, B of (abstract) molecules over some alphabet of atoms, written as $\mathbb{T} = A \to B$, find a vector of positive integers (a multiplicity vector) $(u_\mu | \mu \in A \cup B)$ such that the multiset of molecules $u\mathbb{T} = \sum_{\mu\in A} u_\mu\mu \to \sum_{\mu\in B} u_\mu\mu$ is balanced, that is, for every atom a in the alphabet, if $\mu(a)$ is the multiplicity of a in the molecule μ, then:

$$\sum_{\mu\in A} \mu(a)u_\mu = \sum_{\mu\in B} \mu(a)u_\mu$$

This formulation easily transforms in linear algebra notation as the search for two vectors of positive integers U, V, of size m, k respectively, that verify the following equation, where $\mathbb{A}$ and $\mathbb{B}$ are $n \times m$ and $n \times k$ matrices of positive integers coefficients (n is the number of atoms, m and k the cardinalities of the sets of molecules of A and B):

$$\mathbb{A}U = \mathbb{B}V$$

A natural question is: Can any 2-multiset transformation be correctly balanced? The answer is negative, and it is easy to provide examples showing it.

A preliminary analysis of 2-MBP problem suggests that a general strategy for attacking 2-MBP could be computationally very hard. In fact, if is surely complex to find general conditions for deciding the solvability of the problem and for generating its possible solutions, nevertheless it could be useful to discover a number of results that can be used for giving answers for a large class of instances of this problem, especially if this can be done with reasonable computational costs. Another issue of interest is the individuation of well-known problems that can be seen as particular or more general cases of this problem.

Evolutionary Resilience of Membrane Computations

Matteo Cavaliere[1], Alvaro Sanchez[2]

[1] University of Edinburgh
mcavali2@staffmail.ed.ac.uk
[2] Yale University
alvaro.sanchez@yale.edu

Many results in the area of membrane systems highlight, in various forms, an important point: The ability to encode powerful computations is often linked to the ability of the cells to have some sort of synchronization of their responses i.e., the ability to communicate and to exchange (simple) messages [5]. This is, for instance, the case in simple model of agents (simplified cells) [1] – the ability of agents to synchronize allows the population to perform powerful computations, equivalent to a class of register machines.

In many actual biological scenarios, such synchronization is based on the possibility to communicate either through a shared environment or using diffusible signaling molecules [6, 4].

In this last case, at least some of the cells must be able to produce and secrete the diffusible molecules. Such producers are cooperators paying an individual cost to produce a public good: a signal that allows the cells to coordinate their response to a stimulus. The overall computing power of the population is then linked to the fate of those cells that contribute to the production of the signal. However, the public goods nature of the computation is potentially threatened by the endogenous appearance of cheaters, cells that take advantage of the enhanced computing power mediated by public good, but which do not contribute to its production, avoiding paying the costs associated to it. In situations where the cost-benefit breakdown is favorable to these cheaters, cheating cells may spread in the population disrupting the ability of the cells to properly process environmental information [4].

Interestingly, the ability of individual cells to process information can be a possible solution to the cheating problem. This is indeed the case when such information-processing is coupled to the ecological constraints in which the cellular population is embedded [2] and [3].

In this brief note we want to suggest that the area of membrane computing may be a good framework in which one could combine the notion of cellular computation with evolutionary dynamics and study the issue of evolutionary resilience in a systematic and mathematical manner.

For instance, one could look for formal trade-offs between cellular computation, parallelism and evolutionary resilience. How can a computational process be distributed between individual cells and which is the cost in terms of evolutionary resilience? Which is the best strategy to divide the computational labor between the cells to avoid the emergence of detrimental mutants?

These types of questions may be approached by properly extending standard models of membrane computing. For instance, in the context of a model with agents [1], one may add cell division and assume that cells may divide at different speed – cells that are simpler (in terms of the employed internal biochemistry) will divide faster, spreading in the population and altering the proper balance between different types of cells. This would naturally couple the notion of cellular fitness to that of cellular computation. Such idea could also be easily added in classical models of membrane computing with division, known for their ability to solve efficiently (in parallel way) hard computational problems [5]. One could then study which of the obtained results with membrane systems with divisions are evolutionary stable – i.e., they are not compromised by the fact that some cells may execute less internal complex instructions/biochemistry and then divide faster, spreading in the population and, possibly, compromising the overall computation (certainly remains to formulate a good measure for the complexity of the internal cellular instructions, perhaps in terms of classical descriptional complexity).

We are aware that our suggestions are just a starting point of a discussion on this issue but hopefully may help to contribute novel ideas in how to address a novel and intriguing point in membrane computing – cells need to split the computational tasks with the evolutionary constraint to keep the proper balance between the different types of cells, avoiding that some of them would outcompete the others.

References

1. M. Cavaliere, R. Mardare, and S. Sedwards. A multiset-based model of synchronizing agents: Computability and robustness. *Theoretical Computer Science*, 391(3):216–238, feb 2008.
2. M. Cavaliere and J. F. Poyatos. Plasticity facilitates sustainable growth in the commons. *Journal of The Royal Society Interface*, 10(81):20121006–20121006, jan 2013.
3. K. I. Harrington and A. Sanchez. Eco-evolutionary dynamics of complex social strategies in microbial communities. *Communicative & Integrative Biology*, 7(1):e28230, jan 2014.
4. S. A. Levin. Public goods in relation to competition, cooperation, and spite. *Proceedings of the National Academy of Sciences*, 111 (Supplement_3): 10838–10845, July 2014.
5. G. Păun, G. Rozenberg, and A. Salomaa, eds. *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., 2010.
6. T. J. Perkins and P. S. Swain. Strategies for cellular decision-making. *Mol Syst Biol*, 5, nov 2009.

Families of Languages Associated with SN P Systems: Preliminary Ideas, Open Problems

Gheorghe Păun[1], José M. Sempere[2]

[1] Institute of Mathematics of the Romanian Academy
Bucharest, Romania
gpaun@us.es
[2] Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia, Spain
jsempere@dsic.upv.es

Summary. By considering various encodings of the spike train generated by a spiking neural (SN) P system, several related languages are obtained. A few proposals on how to define such a family of languages, some preliminary results, and a series of suggestions for further research are briefly presented.

1 Basic Definitions, Basic Problems

The reader is supposed to be familiar with SN P systems (in general, with membrane computing area), so that we pass directly to introducing the families of languages mentioned above.

The idea is simple: to take the binary language $L_1(\Pi)$ generated by an SN P system Π (the set of all binary strings – also called spike trains – describing halting computations of Π: a symbol 0 is associated with a step when no spike is sent to the environment, and a symbol 1 is associated with a step when at least one spike is sent to the environment, until the system halts), and to encode blocks of k digits, for various natural numbers k, in such a way that various languages $L_k(\Pi)$ are obtained.

Of course, we have to take care of the case when the spike train is not of a length which is a multiple of the considered k. In this case, we add symbols 0 so that the obtained binary string is of a length divisible by k.

More formally, let $B = \{0, 1\}$ be the binary alphabet, let $k \geq 1$ be a natural number, and V_k be an alphabet. Consider a mapping $\varphi_k : B^k \longrightarrow V_k$. For each string $w \in B^*$ we consider the string ${}_kw = w0^t$, where $t = \min\{n \geq 0 \mid |w0^n| \text{ is a multiple of } k\}$.

The string ${}_kw$ can be written in the form ${}_kw = x_1x_2 \dots x_s$, such that $|x_j| = k$ for all $j = 1, 2, \dots, s$. Then, φ_k can be extended to $(B^k)^*$ in the natural way: $\varphi_k(y_1y_2 \dots y_t) = \varphi_k(y_1)\varphi_k(y_2) \dots \varphi_k(y_t)$ for all $y_i \in B^k, 1 \leq i \leq t, t \geq 0$.

Thus, for an SN P system Π and an encoding φ_k as above, we can define the language

$$L_{\varphi_k}(\Pi) = \{\varphi_k({}_kw) \mid w \in L_1(\Pi)\}.$$

This language depends on the encoding φ_k, hence a family of languages can be associated with Π by varying k and the mapping φ_k.

Already at this very general level there appear several research issues. Consider classes of mappings φ_k with various properties and investigate the properties (size, closure, decidability, etc.) of the corresponding families of languages generated by SN P systems. How these properties depend on the SN P systems? Following suggestions and importing notions and questions from the grammar forms area (a very active research area some decades ago, starting with the pioneering paper [2] – see also the corresponding chapter from the first volume of [5]) can be useful.

2 The One-to-one Case

Actually, the present research idea occurred in a framework related to classical communication channels with encoded information, where with every SN P system different languages can be associated that depend on a parameter that fixes a time window to analyze the spikes train at the output.

A natural possibility is to order in a precise way, e.g., lexicographically, the strings in B^k, and to associate with each of them a distinct symbol from an alphabet Σ_k with 2^k elements.

The fact that the encoding is one-to-one is rather restrictive: the passing from the binary language $L_1(\Pi)$ to a given $L_k(\Pi)$ (we omit mentioning the mapping φ) can be done by means of a sequential transducer (a gsm, in the usual terminology, [5]). Conversely, the passage from $L_k(\Pi)$ to $L_1(\Pi)$ is done by an one-to-one (non-erasing) morphism, which implies that the converse passage is done by an inverse morphism.

This observation is interesting enough to be formally formulated:

Proposition 1. *If $L_1(\Pi) \in FL$, where FL is a family of languages closed under gsm mappings or under inverse morphisms, then $L_k(\Pi) \in FL$, for all $k \geq 1$. If FL is closed under non-erasing morphisms and $L_k(\Pi) \in FL$, then also $L_1(\Pi) \in FL$.*

Families as FL above are REG, LIN, CF in the Chomsky hierarchy, hence if $L_1(\Pi)$ is regular/linear/context-free, then also all languages $L_k(\Pi)$ are regular/linear/context-free, respectively, and conversely.

This means that each family $F(\Pi) = \{L_k(\Pi) \mid k \geq 1\}$ contains only languages of the same type in the Chomsky hierarchy (for instance, it is not possible to have a context-free non-regular language $L_k(\Pi)$ together with a regular language $L_j(\Pi)$, for some $k \neq j$.

Of course, if $L_1(\Pi)$ is finite, then all languages $L_k(\Pi)$ are finite, hence the family $F(\Pi)$ is finite, up to a renaming of symbols of alphabets Σ_k.

If, instead, $L_1(\Pi)$ is infinite, then $F(\Pi)$ can be an infinite family, because the alphabet of $L_{k+1}(\Pi)$ might be larger than the alphabet of $L_k(\Pi)$.

This is the case, for instance, for the SN P system Π generating $L_1(\Pi) = \{1^n01^m \mid n, m \geq 1\}$ (which is an infinite regular language).

This assertion seems to be true (*conjecture*) for all SN P systems Π with infinite $L_1(\Pi)$.

3 The Non-injective Case

The previous type-preserving Proposition 1 does not hold in the case of using encodings which are not one-to-one.

Here is an example: Consider Π such that $L_1(\Pi) = \{1^n01^n \mid n \geq 1\}$ (SN P systems are universal, [4], hence any language can be taken as the starting language). Of course, $L_1(\Pi)$ is context-free non-regular.

Consider the encoding $\varphi_k : B^k \longrightarrow \{a, b\}$ defined by $\varphi_k(w) = a$ if $|w|_0 \leq 1$, and $\varphi_k(w) = b$ if $|w|_0 \geq 2$. We get

$$L_k(\Pi) = a^+ \cup a^*b, \text{ for } k \geq 4,$$

and

$$L_k(\Pi) = a^+ \cup a^+b, \text{ for } k = 2, 3.$$

Clearly, the languages $L_k(\Pi), k \geq 2$, are regular, in spite of the fact that $L_1(\Pi)$ is (context-free) non-regular.

The properties of the encoding is crucial for the properties of the obtained language families (this is true in other frameworks, see, e.g., [3] and its references), hence this issue deserves further research efforts.

4 Final Comments

The idea of associating a family of languages with a given P system is rather natural. We have illustrated it here with the case of SN P systems, but the same strategy can be applied for any type of P systems producing a language (such that cell-like P systems with external output, SN P systems generating trace languages [1], etc.).

A more systematic study of this idea is of interest, starting with relevant examples, continuing with "standard" formal language theory questions, and ending with possible applications of this approach (as languages generated by the same P system are "genetically" related, maybe in this way one can capture biological connections/dependencies or other types of relationships).

Of course, further ways to associate a family of languages to a given SN P system, to a given P system in general, remain to be found.

References

1. H. Chen, M. Ionescu, A. Păun, Gh. Păun, B. Popa: On trace languages generated by spiking neural P systems, *Eighth International Workshop on Descriptional Complexity of Formal Systems* (DCFS 2006), June 21-23, 2006, Las Cruces, New Mexico, USA, 94–105.
2. A.B. Cremers, S. Ginsburg: Context-free grammar forms. *J. Computer System Sci.*, 11 (1975), 86–116.
3. E. Csuhaj-Varjú, G. Vaszil: On counter machines versus dP automata. *Membrane Computing. 14th Intern. Conf., CMC 2013, Chişinău, August 2013* (A. Alhazov et al., eds.), LNCS 8340, Springer, Berlin, 2014, 138–150.
4. M. Ionescu, Gh. Păun, T. Yokomori: Spiking neural P systems. *Fundamenta Informaticae*, 71, 2-3 (2006), 279–308.
5. G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages*, 3 vols., Springer-Verlag, Berlin, 1997.

Membrane Systems Analysis

Marian Gheorghe

School of Electrical Engineering and Computer Science, University of Bradford
Bradford BD7 1DP, UK
m.gheorghe@bradford.ac.uk

Membrane systems analysis is a less developed research area in membrane computing although concepts such is causality – of different types – [1, 2, 3] have been introduced and investigated. In relation to testing, the identifiable P systems have been studied [4]. Also, connections with Petri nets, where analysis methods and tools exist in abundance, have been made [5]. However, for this analysis to become effective it is necessary to develop further research into: (a) identifying simpler components of the execution strategy, similar to occurrence nets of various types (simple, behavioural, communicating etc) [6]; and (b) efficient algorithms for implementing these analysis concepts. This should be a research that will involve any membrane systems, but those benefiting from simulation and verification tools are the most appropriate to be considered. This research should address both formal foundations of these concepts as well as evaluation methods and the construction of appropriate tools supporting the analysis.

References

1. N. Busi. Causality in Membrane Systems, *Lecture Notes in Computer Science*, 4860, 160 – 171, 2007.
2. G. Ciobanu, D. Lucanu. Events, Causality, and Concurrency in Membrane Systems, *Lecture Notes in Computer Science*, 4860, 209 – 227, 2007.
3. O. Agrigoroaie, G. Ciobanu. Quantitative Causality in Membrane Systems, *Lecture Notes in Computer Science*, 7184, 62 – 72, 2011.
4. M. Gheorghe, F. Ipate, S. Konur. Testing Based on Identifiable P Systems Using Cover Automata and X-machines, *Information Sciences*, 372, 565 – 578, 2016.
5. J. Kleijn, M. Koutny. Petri Nets and Membrane Computing, Chapter 15 in *The Oxford Handbook of Membrane Computing* (Gh. Păun, G. Rozenberg, A. Salomaa, eds.), Oxford University Press, 389 – 412, 2010.
6. M. Koutny, B. Randell. Structured Occurrence Nets: A Formalism for Aiding System Failure Prevention and Analysis Techniques, *Fundamenta Informaticae*, 97, 41 – 91, 2009.

P System for Two-Level Economic Exchange with Investment

Alan Mehlenbacher

Economics Department, University of Victoria
Victoria BC, V8P 5C2 Canada
amehlen@uvic.ca

Summary. I expand the use of P systems in Economics that was introduced in [1], [2] by developing a model and reporting simulation results for a dynamic two-level exchange model of buyers and sellers with investment. I invite comments, criticisms, and suggestions.

1 Introduction

Economic simulations using mathematical models have the advantage of rigorous specification, but they have been criticized for being unrealistically rational and restricted to a small number of agents with limited heterogeneity. One alternative is multi-agent systems, which have the advantages of accommodating a large number of agents, heterogeneity, non-optimal behaviors, and agent interactions that produce emergent effects. I have used multi-agent systems to analyze complex interactions in auctions [3], [4], and the possible applications are many [5]. However, multi-agent system models have in turn been criticized for having arbitrary specifications.

Another promising alternative is P systems. These models are dynamic, rigorously specified, and produce emergent effects. They can be deterministic or stochastic. They are scalable in that we can develop a model, observe its behaviour by hand for a few steps, and then run relatively easy-to-develop computer simulations. In summary, P systems have the advantages of multi-agent systems, but are more transparently specified and understood by non-programmers. The applications of P systems are many [6], and the first applications of P systems to Economics were process models of vertical integration and spin-off [1] and a dynamic model framework [2]. More recent applications of P systems in animal Ecology [7], [8], [9], [10], [11] have provided additional inspiration for the application of P systems to dynamic Economic models.

My objective in this project was to design and implement a relatively simple, but not trivial, model of a dynamic buyer-seller exchange with two types of investment: investment in technology that lowers seller production costs and investment

in advertising that entices more buyers into the market. Section 2 contains descriptions of the model's structure and objects. Section 3 specifies the rewriting rules, and Section 4 specifies the processes that use the rewriting rules. Section 5 contains simulation results, and Section 6 concludes with a brief description of more sophisticated models currently in development.

2 Model Structure

P systems consist of objects that evolve and interact in compartments called membranes, a term that reflects the use of P systems in simulations of cellular processes. In this section I describe the membranes and the agent and catalyst objects.

2.1 Membranes

In the two-level exchange model there are four membranes: an economy membrane E, that contains a market membrane M, that in turn contains buyer and seller membranes B and S. The membrane hierarchy is $B, S \subset M \subset E$, so that an empty economy string is $\left[\left[\left[\ \right]_B\left[\ \right]_S\right]_M\right]_E$. In the implementation, the membranes must also be preceded by their type: ${}_E\left[{}_M\left[{}_B\left[\ \right]_B{}_S\left[\ \right]_S\right]_M\right]_E$, but I use the briefer notation for readability. The outer economy membrane E contains the market membrane M and individual buyer and seller agents that have exited or not yet entered the market. Buyers and sellers that have entered the market are contained in membranes B and S that are both contained in M. Buyers and sellers move into M in order to execute buy-sell transactions, and then move back into B and S.

2.2 Agent Objects

The agent objects are buyers and sellers that come in different types. This model uses high, medium, and low levels (h, m, l) for types. The membrane B contains buyers b_i that have low or high willingness to pay (demand) $i \in \{l, h\}$, and membrane S contains sellers s_j that have low or high production costs $j \in \{l, h\}$. Buyer type can change from low demand to high demand, depending on the intensity of advertising. Seller type can change from high cost to low cost, depending on the intensity of investment in technology.

2.3 Catalyst Objects

There are catalyst objects a for advertising and r_k for low, medium, or high profit, i.e., $k \in \{l, m, h\}$. For example, n advertising objects in E induce entry of a buyer from E to B; n medium-profit objects in S induce investment in advertising and/or cost-reduction technology; n high-profit objects in E induce entry of a high-cost seller from E to S; and n low-profit objects in S induce exit of a high-cost seller from S to E. There are two possible ways to use catalysts: deterministic and

probabilistic. A deterministic method applies one of the n catalyst objects to induce the rule to fire and then reduces the number of catalyst objects to *n-1*. A probabilistic approach uses the number of catalysts to generate a probability that increases with the number of catalyst objects. In this approach, the probability $\bar{\rho}$ should increase with the number of catalyst objects and have an upper bound bound of 1. This calls for a logistic function, for example $\bar{\rho}= 1/(1 + 2e^{-0.75n})$, which gives an increasing sequence of probabilities $\{0.51, 0.69, 0.83, 0.91, \dots\}$. In this model, I use the probabilistic method since this better reflects the uncertainty in the decisions of economic agents.

2.4 Initial String

The two-level exchange economy is represented by a string consisting of these membranes, agents, and objects. The string evolves through time by applying a repeated sequence of processes that represent economic decisions and interactions. An example of an initial string is $\left[[[b_l b_h b_h]_B [s_l s_h s_h]_S]_M b_l b_l s_h s_h\right]_E$. The seller agents outside the M membrane are high-cost producers since low-cost producers would have entered; and that the buyer agents that have not entered are low demand since a high-demand buyer would have entered. Each time period is modelled by a sequence of six economic processes, and each process consists of a sequence rules selected from a set of six rewriting rules that are described in the next section.

3 Rewriting Rules

Basic P system rewriting rules for cellular process simulations are explained in [1] and [12] among others, and I have adapted these basic rules for modelling Economics systems. Three fundamental rule types are *evolution* rules that rewrite object strings within a membrane, *communication* rules that rewrite object strings across membranes, and *active membrane* rules that rewrite membranes. The two-level economic exchange model uses rules only of the first two types, but rules of the third type will be required for more complex models. In this system, the *evolution* rules are *add*, *txn*, and *type* while the *communication* rules are *tran*, *trap*, and *trat*.

The *txn* rule implements profit-generating transactions between buyer and seller agents that have been randomly paired. The probabilistic *add* rule creates new objects depending on probability determined by a catalyst. For example, an advertising object is induced probabilistically by profit objects. The *type* rule is also a catalyst-driven probabilistic rule that changes the type of an object. For example, the type of a seller is changed from high-cost to low-cost depending on the level of investment in technology.

The *tran* rule is used for deterministic transfer of objects from one membrane to another. This single rule replaces the two common *in* and *out* rules that transfer objects from container membranes to submembranes and vice versa in cellular

process simulations. In Economic models a membrane can contain more than one submembrane and sometimes we require transfer up or down more than one level in the membrane hierarchy. The *trap* rule is a catalyst-driven probabilistic transfer rule that transfers objects from one membrane to another depending on probability determined by a catalyst. The semi-probabilistic *trat* rule is an interesting combination of the probabilistic *type* rule and the deterministic *tran* rule. If there are one or more catalyst objects, an object is transfered from one membrane to another, but before it is transferred its type may be changed depending on probability determined by a catalyst. For example, a buyer enters the market when there is one or more advertising object, but the buyer is more likely to be high-demand if there is more than one advertising object.

The rules are summarized for general use in Table 1. In the rule specifications, membranes are represented by M and N, objects by a and b, specific object types by t and s, and general object types by i and j . Each probabilistic rule is potentially triggered by a catalyst c_t, such as advertising or profit, that is used to calculate a probability threshold $\bar{\rho}$. When a random draw is less than $\bar{\rho}$ the rule fires and the catalyst is deleted.

4 Processes

Each process is described in the text and then summarized in a table that shows each rule, its specification for the two-level exchange model, and the result of applying the rule to a running example with initial state $\left[[[b_l b_h b_h]_B [s_l s_h s_h]_S]_M b_l b_l s_h s_h\right]_E$. In the rule specifications, only the relevant membranes are shown, but the entire model is shown in the example.

4.1 P1: To Market

Each period begins with b and s agents migrating from their home membranes B and S to the market membrane M. There are two *tran* rules, one for the b agents and one for the s agents, as shown in Table 2.

4.2 P2: Transaction

In the Transaction process, b and s agents in M are randomly paired, each bs string in M induces induces a transaction resulting in a profit object, the profit object is transferred to the appropriate membrane, and the b and s agents are transferred back to their home membranes. The profit object r_k is type $k = \{l, m, h\}$ depending on the types for (b,s) being (l, h), (l, l) or (h, h), or (h, l) respectively. To explain, if the buyer has low willingness to pay (low demand) and the seller has high costs, the difference (profit) between the revenue and the cost will be low; if the buyer has low demand and the seller has low costs or vice versa, the difference between

Table 1. Rules Specifications

Rule Template	Description
	Specification
$add\,(M, a_t, c_t)$	Add typed object a_t into M with probability $\bar{\rho}$ that depends on a catalyst c_t.
	$[c_t]_M \rightarrow \bar{\rho}[a_t]_M$
$tran\,(M, N, a_i)$	Transfer all types of a from M to N.
	$[a_i]_M \rightarrow [a_i]_N$
$trap\,(M, N, a_t, c_t)$	Transfer typed object a_t from M to N with probability $\bar{\rho}$ that depends on a catalyst c_t.
	$[a_t c_t]_M \rightarrow \bar{\rho}[a_t]_N$
$trat\,(M, N, a_t, a_s, c_t)$	Change a's type from t to s with probability $\bar{\rho}$ that depends on a catalyst c_t and transfer the resulting a_i from M to N.
	$[a_t c_t]_M \rightarrow \bar{\rho}[a_i]_N$
$txn\,(M, a_i, b_j)$	Execute a profit-generating transaction between randomly paired objects a and b.
	$[a_t a_s b_s b_t]_M \rightarrow [a_t b_s a_s b_t r_t r_s]_M$
$type\,(M, a_t, a_s, c_t)$	Change a's type from t to s with probability $\bar{\rho}$ that depends on a catalyst c_t.
	$[a_t c_t]_M \rightarrow \bar{\rho}[a_s]_M$

Table 2. P1: To Market

Process	Rule	Specification
		Initial: $\left[\left[[b_l b_h b_h]_B [s_l s_h s_h]_S\right]_M b_l b_l s_h s_h\right]_E$
P1a	$tran\,(B, M, b_i)$	$[b_i]_B \rightarrow \left[[\,]_B b_i\right]_M$
		$\left[\left[[\,]_B [s_l s_h s_h]_S b_l b_h b_h\right]_M b_l b_l s_h s_h\right]_E$
P1b	$tran\,(S, M, s_j)$	$[s_j]_S \rightarrow \left[[\,]_S s_j\right]_M$
		$\left[\left[[\,]_B [\,]_S b_l b_h b_h s_l s_h s_h\right]_M b_l b_l s_h s_h\right]_E$

the revenue and the cost will be medium; if the buyer has high demand and the seller has low costs, the seller's profit will be high.

If k=h the profit object is transferred to E to be used as a catalyst for seller entry, and if k= l or m the profit object is transferred to S to be used as a catalyst for seller exit or investment.

The Transaction process (P2) uses rules *txn* and *tran* as shown in Table 3 to implement the transactions between randomly paired buyers and sellers, transfer the profit objects to the appropriate membrane, and transfer the buyers and sellers back to their home membranes.

Table 3. P2: Transactions

Process	Rule	Specification
		Current: $\left[\left[\left[\ \right]_B\left[\ \right]_S b_l b_h b_h s_l s_h s_h\right]_M b_l b_l s_h s_h\right]_E$
P2a	$txn\,(M, b_i, s_j)$	$[b_i s_j]_M \rightarrow [b_i s_j r_k]_M$ *where k=l for i=l,j=h; k=m for i=j; k=h for i=h,j=l*
		$\left[\left[\left[\ \right]_B\left[\ \right]_S b_l s_h b_h s_l b_h s_h r_l r_h r_m\right]_M b_l b_l s_h s_h\right]_E$
P2b	$tran\,(M, E, r_h)$	$[b_i s_j r_h]_M \rightarrow [b_i s_j]_M [r_h]_E$
		$\left[\left[\left[\ \right]_B[r_l r_m]_S b_l s_h b_h s_l b_h s_h\right]_M b_l b_l s_h s_h r_h\right]_E$
P2c	$tran\,(M, S, r_k)$	$[b_i s_j r_k]_M \rightarrow [b_i s_j]_M [r_k]_S$, *for k=l,m*
		$\left[\left[\left[\ \right]_B[r_l r_m]_S b_l s_h b_h s_l b_h s_h\right]_M b_l b_l s_h s_h r_h\right]_E$
P2d	$tran\,(M, B, b_i)$	$[b_i s_j]_M \rightarrow [s_j]_M [b_i]_B$
		$\left[\left[[b_l b_h b_h]_B[r_l r_m]_S s_h s_l s_h\right]_M b_l b_l s_h s_h r_h\right]_E$
P2e	$tran\,(M, S, s_j)$	$[s_j]_M \rightarrow [s_j]_S$
		$\left[\left[[b_l b_h b_h]_B[s_h s_l s_h r_l r_m]_S\right]_M b_l b_l s_h s_h r_h\right]_E$

4.3 P3: Investment

Seller agents can invest in advertising to increase demand and in technology to reduce costs. Since high profits do not provide incentive for investment and low profits do not provide sufficient funding for investment [13][14], investments are more likely to occur when transactions have produced profits of type m . The probability of investment depends on the number n of m profit objects and, for brevity, I denote n occurences of m profit objects as r_m^n.

Profit objects in S induce investment in advertising and/or technology with probability $\bar{\rho}$. If the advertising condition fires, an advertising object a is created in

S and transferred to E. If the advertising condition does not fire and the technology condition fires, the type of one of the high-cost seller agents s_h in membrane S is changed to an l. When either of these processes fires the r_m^n catalyst objects are deleted, signifying that the profits have been invested.

The process of investment in advertising and/or technology is modeled by three rules, *add*, *tran*, and *type* as shown in Table 4. The example assumes that P3a fires (and so P3c does not) so that an advertising object is created in the E membrane. Note that if P3a does not fire, the *tran* rule has a null effect.

Table 4. P3: Investment

Process	Rule	Specification
		Current: $\left[\left[\left[b_l b_h b_h\right]_B \left[s_h s_l s_h r_l r_m\right]_S\right]_M b_l b_l s_h s_h r_h\right]_E$
P3a	$add\,(S, a, r_m^n)$	$[r_m^n]_S \rightarrow \bar{\rho}[a]_S$
		$\left[\left[\left[b_l b_h b_h\right]_B \left[s_h s_l s_h r_l a\right]_S\right]_M b_l b_l s_h s_h r_h\right]_E$
P3b	$tran\,(S, E, a)$	$[a]_S \rightarrow [a]_E$
		$\left[\left[\left[b_l b_h b_h\right]_B \left[s_h s_l s_h r_l\right]_S\right]_M b_l b_l s_h s_h r_h a\right]_E$
P3c	$type\,(S, s_h, s_l, r_m^n)$	$[s_h r_m^n]_S \rightarrow \bar{\rho}[s_l]_S$
		$\left[\left[\left[b_l b_h b_h\right]_B \left[s_h s_l s_h r_l\right]_S\right]_M b_l b_l s_h s_h r_h a\right]_E$

4.4 P4: Seller Entry

A string of one to n high-profit objects, denoted r_h^n, in E induces entry of a high-cost seller from E to S with probability $\bar{\rho}$, and if entry occurs the r_h^n profit objects are deleted. The Seller Entry process uses a *trap* rule as shown in Table 5 to simulate seller entry that is induced by high profits. The example assumes that P4a fires, i.e., $\rho < \bar{\rho}$, so that a seller enters and the high-profit objects are deleted from the E membrane.

4.5 P5: Seller Exit

A string of one to n low-profit objects, denoted r_l^n, in S induces exit of a high-cost seller from S to E with probability $\bar{\rho}$ and, if exit occurs, the r_l^n profit objects are deleted. The Seller Exit process uses a *trap* rule as shown in Table 6 to simulate seller exit that is induced by low profits. The example assumes that the processes fire, i.e., $\rho < \bar{\rho}$, so that a high-cost seller exits from S to E and the low-profit object is deleted from the S membrane.

Table 5. P4: Seller Entry

Process	Rule	Specification
		Current: $\left[\left[\left[b_l b_h b_h\right]_B \left[s_h s_l s_h r_l\right]_S\right]_M b_l b_l s_h s_h r_h\right]_E$
P4a	$trap\,(E, S, s_h, r_h^n)$	$\left[\left[\ \right]_S s_h r_h^n\right]_E \rightarrow \bar{\rho}\left[\left[s_h\right]_S\right]_E$
		$\left[\left[\left[b_l b_h b_h\right]_B \left[s_h s_l s_h s_h r_l\right]_S\right]_M b_l b_l s_h a\right]_E$

Table 6. P5: Seller Exit

Process	Rule	Specification
		Current: $\left[\left[\left[b_l b_h b_h\right]_B \left[s_h s_l s_h s_h r_l\right]_S\right]_M b_l b_l s_h a\right]_E$
P5a	$trap\,(S, E, s_h, r_l^n)$	$\left[\left[s_h r_l^n\right]_S\right]_E \rightarrow \bar{\rho}\left[\left[\ \right]_S s_h\right]_E$
		$\left[\left[\left[b_l b_h b_h\right]_B \left[s_h s_l s_h\right]_S\right]_M b_l b_l s_h s_h a\right]_E$

4.6 P6. Buyer Entry

A string of one to n advertising objects, denoted a^n, in E induces deterministic transfer of a new buyer from E to B and a probabilistic change in type using the semi-probabilistic *trat* rule. The type for this new buyer depends upon the advertising intensity indicated by the number n of advertising objects. If a random draw is less than $\bar{\rho}$, one of the low-demand buyers is changed to high-demand, and this buyer with type i equal to l or h is then transferred from E to B. The example shown in Table 7 assumes that $\rho < \bar{\rho}$ causing P6a to change a low-demand buyer to high demand before transferring from E to B.

Table 7. P6: Buyer Entry

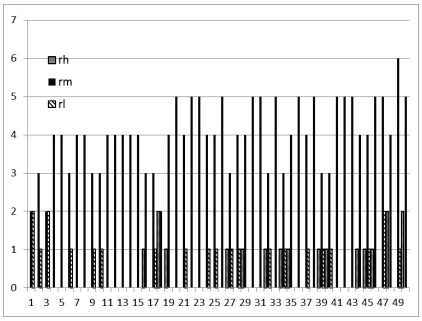
Process	Rule	Specification
		Current: $\left[\left[\left[b_l b_h b_h\right]_B \left[s_h s_l s_h\right]_S\right]_M b_l b_l s_h s_h a\right]_E$
P6a	$trat\,(E, B, b_l, b_h, a^n)$	$\left[b_l a^n\right]_S \rightarrow \bar{\rho}\left[b_i\right]_B$, *for i=l,h*
		$\left[\left[\left[b_l b_h b_h b_h\right]_B \left[s_h s_l s_h\right]_S\right]_M b_l s_h s_h\right]_E$

5 Simulation Results

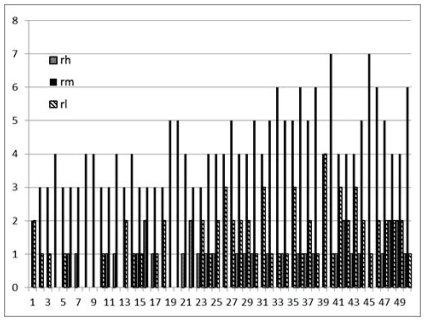
The model was very amenable to implementation in Java, with processes passing parameters to rules. In order to convey some of the variation between simulations, I show results for two of the simulations restricted to 50 periods. Table 8 shows that most transactions result in medium profits, some result in high profit, but only a few result in low profit. The medium profit transactions result in investment in technology that leads to lower costs, as demonstrated in Table 9 by the increasing number of low-cost sellers. Investment in advertising induces buyer entry which increases demand. Table 9 shows the number of buyers of both high and low types increasing steadily. In simulation 1, there are always more low-demand buyers than high-demand buyers, but in simulation 2 the number of high-demand buyers eventually matches the number of low-demand buyers. Table 10 illustrates that most entry is by buyers and that the entry by sellers is somewhat offset by seller exit.

Table 8. Profit

Simulation 1



Simulation 2

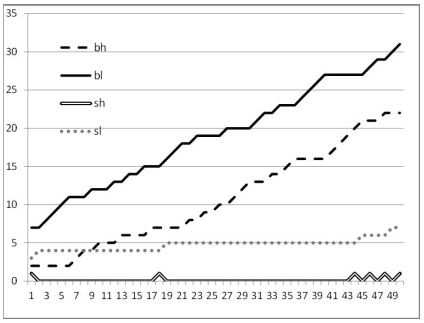


6 Next Steps

This two-level model demonstrates the feasibility of implementing Economic exchange models using P systems. The two-level model can be extended to include objects and types for prices, profits, market concentration, and consumer surplus, as well as enhanced structural features such as membranes for agents, products, and industries. The model can be extended in even more interesting ways by adding another level of exchange so that the three levels represent wholesalers (or manufacturers), retailers (or dealers), and consumers. This three-level model can then be used to model different types of vertical restraints such as resale price maintenance. The rules created for the two-level model are directly usable in the extended

Table 9. Agent Growth

Simulation 1



Simulation 2

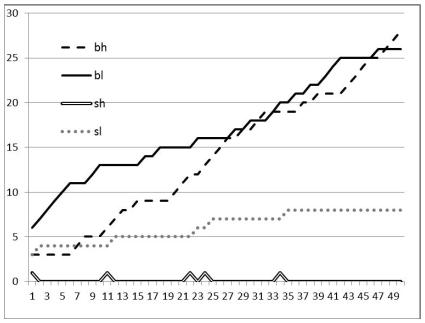
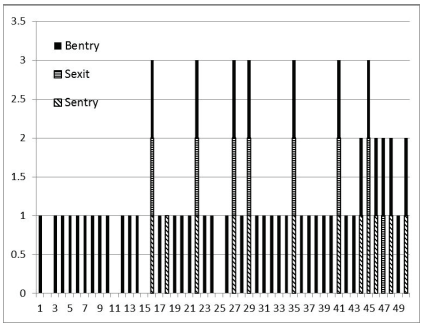
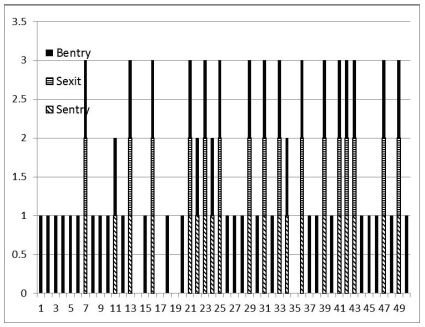


Table 10. Entry and Exit

Simulation 1



Simulation 2



two-level and the three-level models, but new rules are required for transferring membranes, pairing membranes, and updating numeric values.

References

1. G. Păun and R. A. Păun, "Membrane computing as a framework for modeling economic processes," in *Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing (SYNASC'05)*, pp. 8–pp, IEEE, 2005.
2. G. Păun and R. Păun, "Membrane computing and economics: Numerical p systems," *Fundamenta Informaticae*, vol. 73, no. 1, 2, pp. 213–227, 2006.
3. A. Mehlenbacher, "Multiagent system simulations of treasury auctions," *Computational Economics*, vol. 34, no. 1, pp. 67–117, 2009.
4. A. Mehlenbacher, "Multiagent system simulations of signal averaging in english auctions with two-dimensional value signals," *Computational Economics*, vol. 34, no. 2, pp. 119–143, 2009.

5. L. Tesfatsion and K. L. Judd, *Handbook of computational economics: agent-based computational economics*, vol. 2. Elsevier, 2006.
6. G. Păun, G. Rozenberg, and A. Salomaa, *The Oxford handbook of membrane computing*. Oxford University Press, Inc., 2010.
7. M. À. Colomer, A. Margalida, D. Sanuy, and M. J. Pérez-Jiménez, "A bio-inspired computing model as a new tool for modeling ecosystems: the avian scavengers as a case study," *Ecological modelling*, vol. 222, no. 1, pp. 33–47, 2011.
8. M. A. Colomer, I. Pérez-Hurtado, M. J. Pérez-Jiménez, and A. Riscos-Núñez, "Comparing simulation algorithms for multienvironment probabilistic p systems over a standard virtual ecosystem," *Natural Computing*, vol. 11, no. 3, pp. 369–379, 2012.
9. A. Margalida and M. À. Colomer, "Modelling the effects of sanitary policies on european vulture conservation," *Scientific reports*, vol. 2, p. 753, 2012.
10. M. À. Colomer, A. Margalida, and M. J. Pérez-Jiménez, "Population dynamics p system (pdp) models: A standardized protocol for describing and applying novel bio-inspired computing tools," *PloS one*, vol. 8, no. 4, p. e60698, 2013.
11. M. À. Colomer, A. Montori, E. García, and C. Fondevilla, "Using a bioinspired model to determine the extinction risk of calotriton asper populations as a result of an increase in extreme rainfall in a scenario of climatic change," *Ecological Modelling*, vol. 281, pp. 1–14, 2014.
12. G. Păun, "Introduction to membrane computing," in *Applications of Membrane Computing*, pp. 1–42, Springer, 2006.
13. K. Bagwell, "The economic analysis of advertising," *Handbook of industrial organization*, vol. 3, pp. 1701–1844, 2007.
14. C.-Y. Lee, "A new perspective on industry r&d and market structure," *The Journal of Industrial Economics*, vol. 53, no. 1, pp. 101–122, 2005.

Spiking Neural P Systems with Communication on Request

Linqiang Pan[1], Gheorghe Păun[2], Gexiang Zhang[3]

[1] Key Laboratory of Image Information Processing
and Intelligent Control of Education Ministry of China
School of Automation
Huazhong University of Science and Technology
Wuhan 430074, Hubei, China
lqpan@mail.hust.edu.cn

[2] Institute of Mathematics of the Romanian Academy
P.O. Box 1-764, RO-014700 Bucharest, Romania
gpaun@us.es

[3] Robotics Research Center, Xihua University, Chengdu 610039, China
Key Laboratory of Fluid and Power Machinery (Xihua University),
Ministry of Education, Chengdu 610039, China
School of Electrical Engineering, Southwest Jiaotong University,
Chengdu 610031, China
zhgxdylan@126.com

Summary. In the spiking neural P systems investigated so far, the application of evolution rules depends on the contents of a neuron (checked by means of a regular expression), a specified number of spikes are consumed and a specified number of spikes are produced, and then sent to each of the neurons linked by a synapse to the evolving neuron. In this work, we consider the opposite strategy: spikes are requested from neighboring neurons, depending on the contents of the neuron (again, checked by means of a regular expression). No spike is consumed or created, they are only moved along synapses and replicated (when two or more neurons request the contents of the same neuron). It is proved that the obtained computing devices are universal, equivalent with Turing machines, but where two types of spikes are used. It remains open whether the second type of spikes can be avoided without decreasing the computing power.

1 Introduction

Spiking neural (SN) P systems are abstracted from the way the neurons cooperate by sending to each other electrical impulses of identical shape (spikes), which were introduced in [3] and investigated in a large number of papers, see a recent bibliography in [10].

Briefly, SN P systems have the following structure and functioning. Neurons (in the form of a membrane) are placed in the nodes of a graph (whose edges are called synapses) and they contain a number of spikes, identical objects denoted by a, which evolve by means of rules of the form $E/a^c \to a^p$: if the contents of the neuron are described by the regular expression E (over the alphabet $\{a\}$), then c spikes are consumed and p spikes are produced; the produced spikes are sent to all neurons to which a synapse starting from the evolving neuron points out, with the p spikes replicated in such a way that each destination neuron receives p spikes. The system evolves synchronously, in each time unit, each neuron which can use a rule, should use one. When the computation halts, no further rule can be applied, a result is obtained, e.g., in the form of the number of spikes present in a specified neuron in the halting configuration. Many variants of SN P systems were considered. Most of the obtained classes of SN P systems are computationally universal, equivalent in power to Turing machines. An interesting topic is to find small universal SN P systems (first results, much improved after that, can be found in [12]). In certain cases (see, e.g., [4], [9]), polynomial solutions to computationally hard problems can also be obtained in this framework. The interested reader can consult the above mentioned bibliography or the chapter in [13] dedicated to SN P systems.

In terms of parallel-cooperating grammar systems (PCGS), see [1], the "standard" SN P systems are communicating *on command*: the initiative for communication belongs to the emitting neuron. Taking the inspiration from PCGS area, it is natural to consider also the reverse case: the communication *on request*. The spikes should be moved from a neuron to another one when the receiving neuron requests that.

Request-response is an important concept in software engineering. A request-response interaction (also called request-reply) is one of three event based interaction types in an event based system, which is a system in which interactions among the agents in the system are governed by events, principally those interactions that are request-response, message-passing, or publish-subscribe [8]. A request-response interaction happens between two agents. Agent A makes a request to agent B by sending agent B a request indicating the type of request along with the details of the request. Agent B processes the request and responds by sending a reply back to agent A. In a request-response interaction, there are potentially four events [8]: (1) the act of sending the request by agent A; (2) the receipt of the request by agent B; (3) the act of sending the reply by agent B; and (4) the receipt of the reply by agent A. For synchronous request-response interactions, especially those that occur over short periods of time, these four events are normally all combined together and considered one event. There are several instances.

Request-response is one of the basic methods computers use to communicate with each other, in which the first computer sends a request for some data and the second computer responds to the request [18]. For example, browsing a web page is an example of request-response communication. Request-response can be seen as a telephone call, in which someone is called and they answer the call.

Request-response is a message exchange pattern in which a requestor sends a request message to a replier system which receives and processes the request, ultimately returning a message in response [18].

The class of SN P systems we introduce here have only rules for requesting spikes from the neighboring neurons, the action being again dependent on the contents of the neuron. Basically, the rules are of the form $E/Q(a^{n_1}, j_1) \cdots (a^{n_m}, j_m)$, with the meaning that, if the neuron where this rule resides (say, neuron i) has a number of spikes described by the regular expression E, then it asks (this is the meaning of Q) n_1 spikes from neuron j_1, ..., n_m spikes from neuron j_m. If the neurons $j_1, \ldots, j_m$ cannot satisfy the requests (they contain less spikes than requested), then the rule cannot be applied. Also queries of the form (a^∞, j) can be formulated, with the meaning that all spikes from neuron j are requested, no matter how many they are, maybe none. When several neurons request simultaneously spikes from the same neuron, the queries should be identical, and the requested spikes are replicated. Details will be given in the next section.

We want to stress an important feature of this variant of SN P systems, shortly call them *SNQ P systems*: no spike is consumed, they are only moved from a neuron to another one (from this point of view, they remind P systems with symport/antiport rules, [11]); the only way to increase the number of spikes in the system is by replicating the spikes in neurons which receive multiple queries.

Replication is a powerful operation. For instance, in the framework of SNQ P systems we can easily generate non-semilinear sets of numbers. However, we are not able to prove whether SNQ P systems are universal or not (we actually believe that they are not universal). The universality is obtained if we extend the definition by considering spikes of two types (in the proof they are denoted by a and b, with spikes of type b used only for controlling the evolution of the system). It remains open whether or not this second type of spikes can be avoided, without decreasing the computing power. Further research questions are pointed out in the end of the paper.

2 SN P Systems with Communication by Request

We pass now to formally defining the devices briefly described above. The reader is assumed to be familiar with basic elements of membrane computing, e.g., from [13], as well as with some basic notions and notations from language and automata theory, e.g., from [14]. We only mention that V^* denotes the free monoid generated by the alphabet V under the operation of concatenation and the null element λ (the empty string), and that the family of sets of natural numbers computed by Turing machines is denoted by NRE (they are the length sets of recursively enumerable languages, hence the notation).

We directly introduce the devices that we investigate, in the general form (with several types of spikes).

A *spiking neural P system with communication by request* (shortly, SNQ P system), with k types of spikes, is a construct

$$\Pi = (O, \sigma_1, \ldots, \sigma_m, a_{i_0}, out),$$

where:

1. $O = \{a_1, a_2, \ldots, a_k\}$ is an alphabet (a_i is a type of spikes), $k \geq 1$;
2. $\sigma_1, \ldots, \sigma_m$ are *neurons*, of the form $\sigma_i = (a_1^{n_1} a_2^{n_2} \ldots a_k^{n_k}, R_i), 1 \leq i \leq m$, $n_t \geq 0$, $1 \leq t \leq k$, where:
   a) $n_j \geq 0$ is the *initial number of spikes of type* a_j contained in neuron σ_i, $1 \leq j \leq k$;
   b) R_i is a finite set of *rules* of the form E/Qw, with w a finite non-empty list of *queries* of the forms (a_s^p, j) and (a_s^∞, j), $1 \leq s \leq k, p \geq 0, 1 \leq j \leq m$;
3. $a_{i_0}, 1 \leq i_0 \leq k$, is the *type of output spikes* and $out \in \{1, 2, \ldots, m\}$ indicates the *output neuron*.

The meaning of a query (a_s^p, j) is that neuron σ_i requests p copies of a_s from neuron σ_j, while (a_s^∞, j) means that all spikes of type a_s from σ_j, No matter how many they are, are requested by σ_i.

Please note that we have not specified the set of *synapses*, as they are implicitly defined by the rules.

An SNQ P system starts from the initial configuration, described by the numbers of spikes of each type present in each neuron in the beginning of the computation, and proceed by applying the rules in the style of usual SN P systems – synchronously, one rule is applied in each neuron where a rule can be used. E/Qw can be used if both (1) the contents of the neuron are described by the regular expression E (note that the contents of the neuron are interpreted in the multiset sense, all strings obtained by permuting a string describing the number of spikes in the neuron are equivalent to each other) and (2) all queries formulated in w are satisfied (if (a_s^p, j) is a query in w and σ_j contains strictly less than p spikes, then the rule cannot be applied). Note that a query (a_s^∞, j) is always satisfiable, as all spikes a_s from σ_j are requested, no matter how many they are, maybe none.

There is another situation when a rule cannot be applied, namely in the case of *conflicting queries*: if two different neurons i_1, i_2 ask from the same neuron j different numbers of occurrences of the same spike a_s, then the two rules cannot be used simultaneously (but one of them, non-deterministically chosen, can be used). Note that there are two different cases considered, that is, two queries of the form $(a_s^p, j), (a_s^r, j)$ with $p \neq r$, and two queries of the form $(a_s^p, j), (a_s^\infty, j)$, for p a given number.

A delicate point appears when defining the result of a computational step, because of the interplay of the queries. We describe a *step* in terms of several *sub-steps* as follows.

- **Sub-step 1.** In each neuron, we choose a rule to apply, and check its applicability. This means checking three conditions: (i) that the regular expression in the rule corresponds to the contents of the neuron, (ii) that the queries in the rule can be satisfied by the indicated neurons, and (iii) that there are no conflicting queries among the selected rules. If any of these conditions is not

satisfied, then the rules should be changed, or, in the case of the third condition, some of the rules involved in conflicting queries should be omitted. However, the set of selected applicable rules should be maximal, in the sense that no rule can be added to the set without losing the applicability (each neuron which can evolve, should do it).

- **Sub-step 2.** The requested spikes are removed from the neurons where they were present. For each neuron we have three cases: (i) no spike a_s was requested by any other neuron (and then the existing number of spikes a_s remains unchanged), (ii) all spikes of some kind a_s were requested, by at least one other neuron (and then no spike of this kind remains here), or (iii) p spikes of type a_s are requested, by at least one other neuron (and then p is deduced from the number of copies of a_s present in the neuron). Note that because of the fact that the requests are not conflicting, we know precisely how many spikes of each type we have to deduce from each neuron.
- **Sub-step 3.** The queries are satisfied, the requested spikes are moved to the requesting neurons. To the result of Sub-step 2 we add the requested spikes, with the following meaning: if two (or more) neurons request spikes from the same other neuron, then the number of spikes to be submitted to the two (or more) neurons is the same (say, p copies of some a_s), but only p spikes are removed from the emitting neuron, the p spikes are replicated and exactly p spikes are moved to each of the requesting neurons. The same in the case of two or more queries of the form (a_s^∞, j), all spikes present in σ_j are replicated as many times as the number of other neurons having submitted queries to σ_j.

Of course, the three sub-steps together form a step, which lasts for one time unit.

After a computation step as illustrated above, the system passes to a new configuration. A sequence of such *transitions* from a configuration to another one, starting from the initial configuration, is called a *computation*. A computation *halts* if it reaches a configuration where no rule can be applied. The *result* of a halting computation is the number of copies of spike a_{i_0} present in neuron σ_{out} in the halting configuration. For an SNQ P system Π, let us denote by $N(\Pi)$ the set of numbers generated by Π.

Let us also denote by $NSN_kP_m(Q)$ the family of sets $N(\Pi)$ generated by SNQ P systems using at most k types of spikes and at most m neurons. When the numbers k or m can be arbitrary, the corresponding parameter is replaced with $*$.

Note that there are several important differences of SNQ P systems in comparison with usual SN P systems: (1) we use several types of objects, and we still call all of them spikes, (2) there is no interaction with the environment, no spike is sent out, hence we have to consider the result of a computation only in the internal mode (no spike train is defined here), (3) there is no other way to increase the number of spikes than the replication in the case of multiple queries from the same neuron (this corresponds to the case when a neuron in a usual SN P system sends spikes to several neurons to which it has synapses).

It is also worth mentioning the difference from the systems we consider here and those in [15], where the request is done only from the environment, for cell-like

SN P systems, using (in the skin region only) rules of the form $E/\lambda \leftarrow a^p$, with the meaning that p spikes are brought from the environment. Besides these rules, usual spiking rules are used in [15].

3 Preliminary Results

We start the study of SNQ P systems by examining the power of small systems, which have a small number of neurons and of types of spikes. This is also an opportunity to illustrate the previous definitions with a couple of simple examples.

Directly from definitions, it follows that the two parameters k and m induce a double hierarchy of families of sets of numbers:

$$NSN_kP_m(Q) \subseteq NSN_{k+1}P_{m+1}, \text{ for all } k \geq 1, m \geq 1.$$

As we will see in the next section, the hierarchy on the number of types of spikes collapses at the second level (SNQ P systems with two types of spikes are already universal). Because of the universality, the other hierarchy on the number of neurons cannot be infinite, but we do not know its precise height.

The following observation is obvious: systems with only one neuron cannot apply any rule, hence they only generate singleton sets.

Lemma 1. *$NSN_kP_1 = NSN_*P_1 = SING, k \geq 1$, where $SING$ denotes the family of singleton sets.*

Systems with two neurons have also a rather limited power.

Lemma 2. *$NSN_kP_2 = NSN_*P_2 = FIN, k \geq 1$, where FIN denotes the family of finite sets of numbers.*

Proof. In systems with two neurons, spikes cannot be replicated, hence the initial number of spikes cannot be increased, and $NSN_*P_2 \subseteq FIN$.

On the other hand, $FIN \subseteq NSN_1P_2$: consider a finite set of numbers, arranged in the increasing order, $n_1 < n_2 < \ldots < n_s$, and consider the SNQ P system from Figure 1. We use the standard style in representing SN P systems; we also explicitly represent the synapses defined by the queries, as well as the initial spikes present in neurons (if no spike is specified, this means that no spike is present in that neuron in the initial configuration). Also as usual in the area of SN P systems, we identify a neuron with its label, thus equivalently saying "neuron σ_i" and "neuron i".

Each computation takes only one step, one of the elements of the finite set is non-deterministically generated. □

The passage from two neurons to three neurons entails a rather large increase of the generative power, and the explanation resides in the possibility to have replication of spikes, not only permitting the increase of number of spikes, but even an exponential increase.

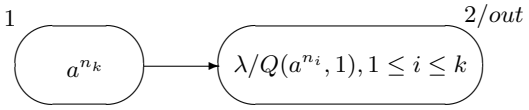


Fig. 1. An SNQ P system generating a finite set

Lemma 3. *The family $NSN_1P_3(Q)$ contains any arithmetical progression.*

Proof. Let us take an arithmetical progression $L = \{n_0 + i \cdot n_1 \mid i \geq 1\}$, for some $n_0 \geq 0$ and $n_1 \geq 1$, and construct the SNQ P system Π from Figure 2.

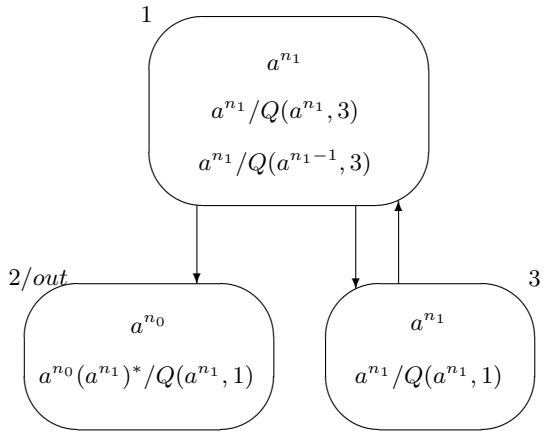


Fig. 2. An SNQ P system generating an arithmetical progression

Formally, the system is:

$$\begin{aligned}
&\Pi = (\{a\}, \sigma_1, \sigma_2, \sigma_3, a, 2), \text{ where}\\
&\sigma_1 = (n_1, \{a^{n_1}/Q(a^{n_1},3),\ a^{n_1}/Q(a^{n_1-1},3)\}),\\
&\sigma_2 = (n_0, \{a^{n_0}(a^{n_1})^*/Q(a^{n_1},1)\}),\\
&\sigma_3 = (n_1, \{a^{n_1}/Q(a^{n_1},1)\}).
\end{aligned}$$

In each step, neurons σ_1 and σ_3 repeatedly exchange n_1 spikes, while neuron σ_2 also requests n_1 spikes from neuron σ_1 (hence the n_1 spikes of neuron σ_1 are duplicated), thus going along the terms of the arithmetical progression. The computation can stop in any moment, by using the second rule of neuron σ_1: neuron σ_1 brings only $n_1 - 1$ spikes inside (hence the query of neuron σ_2 cannot be satisfied) and neuron σ_3 remains with one spike inside, which, together with the n_1 spikes brought from neuron σ_1, do not allow the use of the rule in neuron σ_3. Clearly, $N(\Pi) = L$. □

Lemma 4. *The family $NSN_1P_3(Q)$ contains non-semilinear sets of numbers.*

Proof. Let us consider the SNQ P system Π in Figure 3.

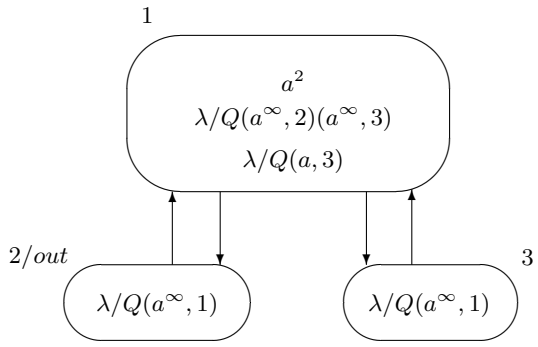


Fig. 3. An SNQ P system generating a non-semilinear set

The neurons $\sigma_1, \sigma_2, \sigma_3$ can use a rule only if they are empty. This is the case in the beginning with neurons σ_2 and σ_3, hence they request all spikes of neuron σ_1. Now neuron σ_1 can request back the spikes from neurons σ_2, σ_3, getting 4 spikes inside. As long as the neurons use their rules asking for all spikes of the partner neurons, the number of spikes present in neuron σ_1 is doubled, and this also happens with the contents of neurons σ_2 and σ_3.

At some step, $\lambda/Q(a, 3)$ is used in neuron σ_1, simultaneously with neurons σ_2, σ_3 requesting the spikes of neuron σ_1. The computation halts, because all neurons have at least one spike inside, hence they can use no rule. The number of spikes from the output neuron is a power of 2, so $N(\Pi) = \{2^n \mid n \geq 1\}$. □

Therefore, the increase of the number of neurons from 1 to 2 and to 3 induces a strict increase of the computing power of SNQ P systems. It remains to be investigated whether the strict increase of computing power is also true for the next levels of the hierarchies $NSN_kP_m(Q) \subseteq NSN_kP_{m+1}$.

4 The Universality of SNQ P Systems

We give now the main result of the paper, the universality of SNQ P systems with two types of spikes (without a bound on the number of neurons). The proof will use the characterization of NRE by means of register machines.

Such a device is a construct $M = (n, H, l_1, l_h, I)$, where n is the number of registers, H is the set of instruction labels, l_1 is the start label (for simplicity, we may assume that l_1 labels an `ADD` instruction, but this is not essential; note that in many places the start instruction is labeled with l_0, but here we prefer to start

from 1), l_h is the halt label (assigned to instruction HALT), and I is the set of instructions; each label from H labels only one instruction from I, thus precisely identifying it. The instructions are of the following forms:

- $l_i : (\texttt{ADD}(r), l_j, l_k)$ (add 1 to register r and then go to one of the instructions with labels l_j, l_k),
- $l_i : (\texttt{SUB}(r), l_j, l_k)$ (if register r is non-empty, then subtract 1 from it and go to the instruction with label l_j, otherwise go to the instruction with label l_k),
- $l_h : \texttt{HALT}$ (the halt instruction).

A register machine M starts with all registers empty (i.e., storing the number zero), applies the instruction with label l_1 and proceeds to apply instructions as indicated by labels (and made possible by the contents of registers); if the machine reaches the halt instruction, then the number n stored at that time in the first register is said to be computed by M. The set of all numbers computed by M is denoted by $N(M)$. If the computation never halts, then no number is generated. It is known that register machines compute all sets of numbers which are Turing computable, hence they characterize NRE (see, e.g., [7]).

Theorem 1. $NRE = NSN_2P_*(Q)$.

Proof. As usual, we only prove the inclusion $\subseteq$, the opposite one can be obtained through a straightforward but cumbersome construction of a Turing machine simulating an SNQ P system, or we can invoke the Turing-Church thesis.

Starting from a register machine $M = (n, H, l_0, l_h, I)$ we construct an SNQ P system Π with two types of spikes, which we denote by a and b, hence $O = \{a, b\}$. We associate one neuron with each register of M (denoted by $1, 2, \ldots, n$), one neuron σ_l with each label $l \in H$, as well as a second neuron, $\sigma_{l_i'}$ with each instruction of M of the form $l_i : (\texttt{SUB}(r), l_j, l_k)$. We also consider the neurons $\sigma_i, 1 \leq i \leq 5$, as mentioned below. (Therefore, the number of neurons depends on the number of registers and labels of M, that is why we cannot bound it in advance.)

If the value of a register r is m, then the corresponding neuron σ_r contains m spikes a. That is, a is the spike indicating the result of the computation, and the output neuron is σ_1 (associated with register 1).

Let us assume that H contains t elements, $l_1, l_2, \ldots, l_t$. Initially, each neuron contain $2t$ copies of b and no copy of a, with the exception of neurons $\sigma_{c_i}, 1 \leq i \leq 5$, which contain the spikes a, b, b, b, b , respectively (see also the construction below).

ADD module: For each instruction $l_i : (\texttt{ADD}(r), l_j, l_k)$ of M we construct the module represented in Figure 4.

In order to simplify the proof, we first assume that we are allowed to also use a query of the form (a, env), with the meaning that one copy of a is requested from the environment – with the environment supposed to contain arbitrarily many copies of a. Later we will remove this kind of rules.

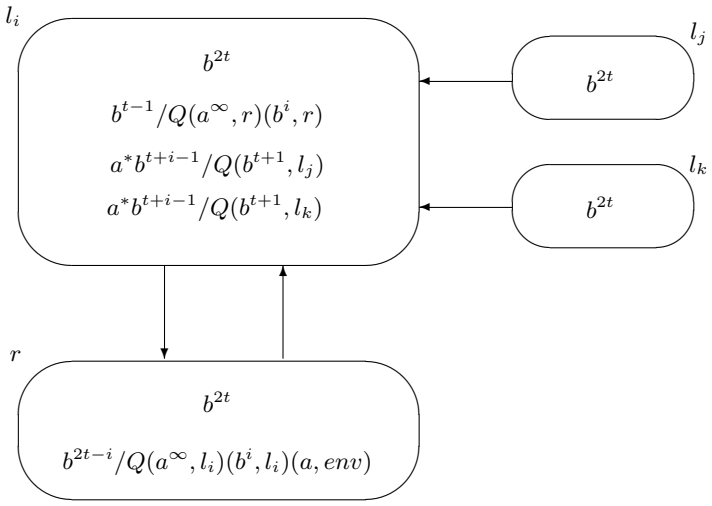


Fig. 4. The ADD module

Such a module is activated when $t+1$ spikes b from neuron l_i are requested by another neuron. (In this construction, the neuron which asks $t+1$ copies of b from σ_{l_i} is σ_{c_1}. If we would accept the query (a, env), hence neurons c_1, c_2, c_3 will be absent, then we have to start with only $t-1$ spikes b in neuron l_1, the starting one.) The neuron (with label) l_i becomes active and it requests from neuron r all copies of a as well as i copies of b. Note that i precisely identifies the label l_i, which precisely identifies the instruction (hence neurons r, l_j, l_k).

In the next step, both neurons l_i and r can apply a rule. In this way, the previous contents of neuron r returns to neuron r, at the same time neuron r requesting one copy of a from the environment, which corresponds to the fact that the register was increased by 1. Simultaneously, neuron l_i uses one of the rules $a^*b^{t+i-1}/Q(b^{t+1}, l_j)$, $a^*b^{t+i-1}/Q(b^{t+1}, l_k)$, non-deterministically chosen, which means that one of the neurons l_j, l_k is activated, while l_i ends with $2t$ copies of b inside, as it was the case at the beginning. The instruction of M is correctly simulated, and one of the instructions with label l_j, l_k will be simulated in the next steps.

It is important to note that, in spite of the fact that several instructions ADD can refer to the same register r (as well as several instructions SUB), this does not lead to wrong computations (i.e., computations in Π not corresponding to computations in M), because the regular expression b^{2t-i} of the rules in neuron r precisely identifies the neuron l_i to which a query is addressed from neuron r.

Let us see now how to avoid the query (a, env). Instead of the query (a, env), we put in neuron r the query (a, c_1), and then we consider the module consisting of three neurons given in Figure 5. Their role is to produce arbitrarily many copies of spike a, keeping them available to neurons with label r corresponding to ADD

instructions, then to request $t+1$ copies of spike b from neuron l_1, thus triggering the simulation of the first instruction in M.

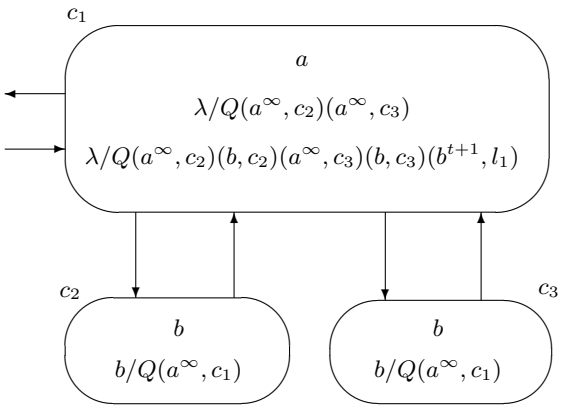


Fig. 5. The module producing arbitrarily many copies of spike a

This module functions in a way similar to the system in Figure 3: as long as neurons c_2, c_3 contains one copy of spike b, they can request the contents of c_1, which then can bring back, doubled, the number of spikes a, repeatedly, until non-deterministically choosing to use the second rule, $\lambda/Q(a^\infty, c_2)(b, c_2)(a^\infty, c_3)(b, c_3)(b^{t+1}, l_1)$. This rule blocks the functioning of neurons c_2, c_3, and also activates neuron l_1.

It is easy to see that this module substitutes the use of the query (a, env), with only one exception: if the module in Figure 5 stops "too early" and there are not enough copies of spike a, as necessary for the simulation of the computation in M. To avoid this situation, we also consider a "trap module": we add the rule $b^{2t-i}/Q(b, c_4)$ to neuron r. If the rule $b^{2t-i}/Q(a^\infty, l_i)(b^i, l_i)(a, c_1)$ cannot be used because neuron c_1 contains no copy of spike a, then this new rule should be used, requesting one copy of spike b from neuron c_4. This neuron is a part of the module in Figure 6.

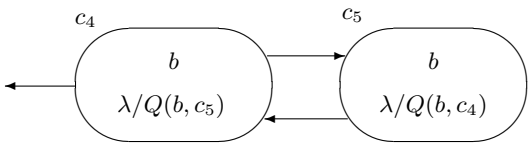


Fig. 6. The trap module

With spikes b inside, neurons c_4 and c_5 cannot use any rule, but after removing the spike b from c_4, neurons c_4 and c_5 will repeatedly ask to each other the remaining spike b, and the computation never stops.

SUB module: For each instruction $l_i : (\texttt{SUB}(r), l_j, l_k)$ of M we construct the module given in Figure 7.

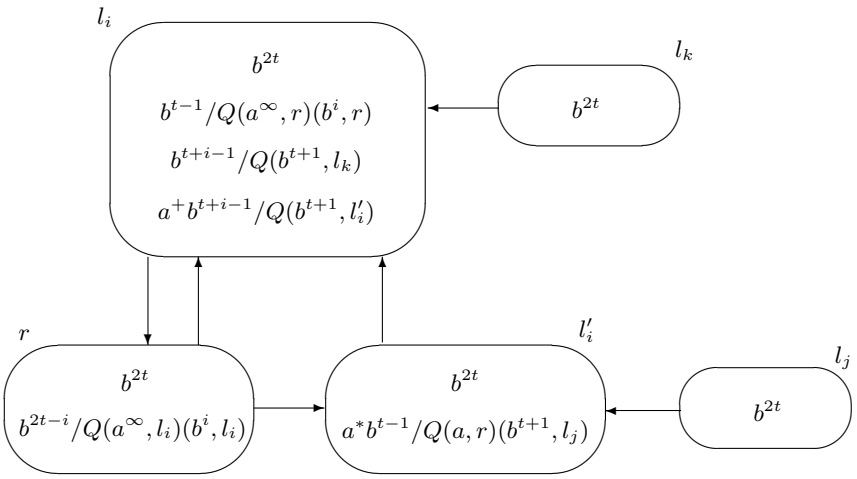


Fig. 7. The SUB module

When neuron l_i "looses" $t+1$ spikes b, it becomes active, and can absorb all spikes a and i spikes b from neuron r. In the next step, both neurons r and l_i can use one rule. If there is no spike a present (corresponding to the fact that register r was empty), then neuron l_i has to use the rule $b^{t+i-1}/Q(b^{t+1}, l_k)$, and neuron l_k is activated. In parallel, neuron r returns to its previous contents (no subtraction was made), neurons l_i', l_j are not modified.

If there is at least one copy of spike a present, the subtraction is performed by activating first neuron l_j' (in parallel, neuron r returns to its previous contents). Neuron l_i' decreases by one the contents of neuron r and activates neuron l_j. The copies of spike a requested by neuron l_i' during a computation remain in this neuron, they are "accepted" by the regular expression of the rule in l_i'.

Again, no unwanted interferences between SUB modules appear, because the label l_i precisely identifies the instruction (hence the module). In this way, the SUB instruction is also correctly simulated.

The simulation of the computation in M continues until the halt instruction is reached. In the neuron associated with l_h there is no rule, hence after activating this neuron, the computation in Π halts. The number of copies of spike a in neuron 1 is the result of computation, hence $N(M) = N(\Pi)$. □

5 A Small Universal SNQ P System

Starting from a universal register machine, as those presented in [5], we will obtain a universal SNQ P system.

In [5], the register machines are used for computing functions, with the universality defined as follows. Let $(\varphi_0, \varphi_1, \ldots)$ be a fixed admissible enumeration of the unary partial recursive functions. A register machine M_u is said to be universal if there is a recursive function g such that for all natural numbers x, y we have $\varphi_x(y) = M_u(g(x), y)$. In [5], several universal register machines are constructed, with the input (the couple of numbers $g(x)$ and y) introduced in specified input registers and the result obtained in another specified register, the output one.

The machine from [5] used in [12] is given in Figure 8. It has 8 registers and 23 instructions. Because here we do not work with numbers encoded in the spike train, as the distance in time between consecutive spikes, but with the multiplicity of spike a in specified neurons, we can have the input and the output of a computation in an SNQ P system defined in the same way as in register machines, hence no input and output module as in [12] is necessary.

$l_0 : (\texttt{SUB}(1), l_1, l_2)$, $l_1 : (\texttt{ADD}(7), l_0)$,
$l_2 : (\texttt{ADD}(6), l_3)$, $l_3 : (\texttt{SUB}(5), l_2, l_4)$,
$l_4 : (\texttt{SUB}(6), l_5, l_3)$, $l_5 : (\texttt{ADD}(5), l_6)$,
$l_6 : (\texttt{SUB}(7), l_7, l_8)$, $l_7 : (\texttt{ADD}(1), l_4)$,
$l_8 : (\texttt{SUB}(6), l_9, l_0)$, $l_9 : (\texttt{ADD}(6), l_{10})$,
$l_{10} : (\texttt{SUB}(4), l_0, l_{11})$, $l_{11} : (\texttt{SUB}(5), l_{12}, l_{13})$,
$l_{12} : (\texttt{SUB}(5), l_{14}, l_{15})$, $l_{13} : (\texttt{SUB}(2), l_{18}, l_{19})$,
$l_{14} : (\texttt{SUB}(5), l_{16}, l_{17})$, $l_{15} : (\texttt{SUB}(3), l_{18}, l_{20})$,
$l_{16} : (\texttt{ADD}(4), l_{11})$, $l_{17} : (\texttt{ADD}(2), l_{21})$,
$l_{18} : (\texttt{SUB}(4), l_0, l_h)$, $l_{19} : (\texttt{SUB}(0), l_0, l_{18})$,
$l_{20} : (\texttt{ADD}(0), l_0)$, $l_{21} : (\texttt{ADD}(3), l_{18})$,
$l_h : \texttt{HALT}$.

Fig. 8. The universal register machine from [5]

Therefore, a direct counting on the modules constructed in the previous proof (8 registers + 23 labels + 13 SUB instructions + 5 neurons in Figures 5 and 6 means a total of 49 neurons) leads to the following result:

Theorem 2. *There is a computing universal SNQ P system with 49 neurons.*

It is highly possible that the number 49 can be slightly improved (by looking to other universal register machines in [5], by possibly saving some neurons by carefully examining the structure of the starting universal register machine, or by using a different construction). This task is left as an open problem to the reader.

6 Further Research Questions

As expected for a fresh kind of model, many questions remain to be investigated. We only mention here a few of them:

1. Can the universality be obtained also for SNQ P systems using only one type of spikes? (Do we have $NRE = NSN_1P_*(Q)$?) We expect a negative answer, and a confirmation of this conjecture would be rather interesting, as not so many classes of P systems are known which are not universal (but are able to compute more than semilinear sets – see the example from Section 3).
2. We have seen that the replication can grow exponentially the number of spikes in linear time. Can this be used in order to solve **NP**-complete problems in polynomial time? We again expect a negative answer – prove such a *Milano theorem* for SNQ P systems (prove that an SNQ P system can be simulated by a Turing machine with a polynomial slow-down, as done in [17] for multiset processing P systems and in [6] for usual SN P systems).
3. Without duplication (and without bringing spikes from the environment) the number of spikes present in the system remains constant, hence only regular sets of numbers can be generated. Can *all* regular sets be generated in this way?
4. Look for normal forms, e.g., in terms of the number of neurons from which a rule can request spikes. In the proof of Theorem 1 we have rules requesting spikes from 1, 2 or even 3 neurons (the case of neuron c_1). Is the universality lost if we bound this number to 1 or 2?
5. An interesting kind of queries seems to be those of the form $(a^{\infty-s}, j)$: take all but s spikes a from neuron σ_j. Of course, if σ_j contains less than s spikes, then the query cannot be satisfied, the rule cannot be applied. Are such queries useful (for instance, in avoiding the use of the second type of spikes in the universality proof)?
6. Although we do not have a spike train associated with a computation in an SNQ P system, we can associate a language to such a system in terms of traces, as in [2]: follow the path of a designated spike from a neuron to another one. The family of these trace languages remains to be investigated.
7. Finally, we point out another natural question: can we remove the regular expressions from the rules and replace them with polarizations associated with the membranes, as done in [16] for standard SN P systems? This made the universality proof in [16] much more difficult, so this is expected also for SNQ P systems – or maybe they will no longer be universal in this case.

Anyway, it is our belief that the spiking neural P systems with communication on request deserve further research efforts.

Acknowledgments

The work of L. Pan was supported by National Natural Science Foundation of China (61033003 and 61320106005) and the Innovation Scientists and Technicians Troop Construction Projects of Henan Province (154200510012). The work of G. Zhang was supported by National Natural Science Foundation of China (61373047 and 61672437) and the Research Project of Key Laboratory of Fluid and Power Machinery (Xihua University), Ministry of Education, P.R. China (JYBFXYQ-1).

References

1. E. Csuhaj-Varjú, J. Dassow, J. Kelemen, Gh. Păun: *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.
2. M. Ionescu, A. Păun, Gh. Păun, M.J. Pérez-Jiménez: Computing with spiking neural P systems: Traces and small universal systems. *DNA Computing. 12th Intern. Meeting on DNA Computing, DNA12, Seoul, Korea, June 2006, Revised Selected Papers* (C. Mao, T. Yokomori, eds.), LNCS 4287, Springer, 2007, 1–16.
3. M. Ionescu, Gh. Păun, T. Yokomori: Spiking neural P systems. *Fundamenta Informaticae*, 71, 2 (2006), 279–308.
4. T.-O. Ishdorj, A. Leporati, L. Pan, X. Zeng, X. Zhang: Deterministic solutions to QSAT and Q3SAT by spiking neural P systems with pre-computed resources. *Theoretical Compututer Science*, 411, 25 (2010), 2345–2358.
5. I. Korec: Small universal register machines. *Theoretical Computer Science*, 168 (1996), 267–301.
6. A. Leporati, C. Zandron, C. Ferretti, G. Mauri: On the Computational Power of Spiking Neural P Systems, *Proceedings of the Fifth Brainstorming Week on Membrane Computing*, Sevilla, Spain, 2007, 227-245.
7. M. Minsky: *Computation – Finite and Infinite Machines*. Prentice Hall, Englewood Cliffs, NJ, 1967.
8. G. Mühl, L. Fiege, P. Pietzuch: *Distributed Event-Based Systems*. Springer-Verlag New York, Inc., 2006.
9. L. Pan, Gh. Păun, M.J. Pérez-Jiménez: Spiking neural P systems with neuron division and budding. *Science China Information Sciences*, 54, 8 (2011), 1596–1607.
10. L. Pan, T. Wu, Z. Zhang: A bibliography of spiking neural P systems, *Bulletin of IMCS*, 1 (June 2016), 63–78 (available at `http://membranecomputing.net/IMCSBulletin/`.
11. A. Păun, Gh. Păun: he power of communication: P systems with symport/antiport. *New Generation Computing*, 20, 3 (2002), 295–306.
12. A. Păun, Gh. Păun: Small universal spiking neural P systems. *BioSystems*, 90, 1 (2007), 48–60.
13. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*. Oxford University Press, 2010.
14. G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.
15. T. Song, L. Pan: Spiking neural P systems with request rules. *Neurocomputing*, 193, 12 (2016), 193–200.

16. T. Wu, A. Păun, Z. Zhang, L. Pan: Spiking neural P systems with polarizations, Submitted, 2015.
17. C. Zandron: *A Model for Molecular Computing: Membrane Systems*. PhD Thesis, Univ. degli Studi di Milano, 2001.
18. https://en.wikipedia.org/wiki/Request

Abstracts of PhD Theses

Author: Christian Bodenstein
(christian.bodenstein@uni-jena.de)

Title: *Theoretical analysis of cyclic processes in biology in the context of Ca^{2+} oscillations, circadian rhythms and synthetic oscillators*

Supervisors:

Ines Heiland, University of Tromso, Department of Arctic and Marine Biology
Thomas Hinze, Brandenburg University of Technology Cottbus, Institute of Computer Science
Stefan Schuster, Friedrich Schiller University Jena, Department of Bioinformatics

Thesis committee:

Peter Dittrich, Friedrich Schiller University Jena, Bio System Analysis Group
Marc Thilo Figge, Leibniz Institute for Natural Product Research and Infection Biology Jena
Ines Heiland, University of Tromso, Department of Arctic and Marine Biology
Thomas Hinze, Brandenburg University of Technology Cottbus, Institute of Computer Science
Peter Ruoff, University of Stavanger, Centre of Organelle Research
Stefan Schuster, Friedrich Schiller University Jena, Department of Bioinformatics

Date of presentation: July 29, 2015

Thesis abstract

Cyclic processes in biology are ubiquitous: they can be found in nearly every organism on earth. The time-scales of these processes vary from microseconds,

in the case of Ca^{2+} oscillations, over hours, in the case of the circadian rhythm, to years in the case of insect eclosion rhythms. Furthermore, the spatial scale is huge: it ranges from cyclic molecular processes within one cell, for example in single cell circadian clocks, over complex oscillator networks of thousands of neurons in the mammalian circadian pacemaker, to population wide predator-prey oscillations. This thesis picks out three examples from this enormous spectrum of biological oscillations and analyzes properties and advantages of these rhythms in a theoretical manner.

The question about the purpose of oscillations in the intracellular Ca^{2+} concentration is still not completely answered. Without doubt, the oscillating signal carries information about the extracellular environment and is decoded into the correct response of the cell. However, the hypothesis of frequency encoded information is nowadays challenged by experimental and theoretical studies showing large variations in the intervals between subsequent Ca^{2+} spikes due to a random spike generation. Therefore, our first study analyzed the relation between the average and steady-state Ca^{2+} concentrations in the cell. Using a mathematical model, we proved that for a convex Ca^{2+} efflux kinetics the average Ca^{2+} concentration is lower, which could be advantageous since Ca^{2+} is toxic. However, for concave kinetics this relation is reversed and the analysis of experimental data revealed that this resembles the natural situation. We believe that the increased average Ca^{2+} concentration could actually support the random Ca^{2+} spike generation and thus, lead to a more regular oscillation. In the second study we analyzed if oscillating Ca^{2+} signals lead to a stronger activation of downstream target proteins compared to constant signals. Indeed we found that this is the case if the kinetics involved in the activation of the target protein possess convex parts, for example due to cooperative Ca^{2+} binding. Similar observations were made before in experiments and a comparison with our theoretical results revealed a high degree of consistence.

A further prominent example of a biological rhythm is the circadian rhythm. The synchronization of this rhythm to light has often been studied, however the rhythm can also be influenced and synchronized by temperature. In a first study we reconstructed a circadian clock model for the unicellular green algae *Chlamydomonas reinhardtii* including changing reaction rates with varying environmental temperature. An important property of the circadian rhythm is that its period is insensitive to environmental temperature changes. Based on results by Ruoff et al. (2003) we manually designed such a temperature compensated oscillator and then analyzed its potential to entrain to an external temperature rhythm with a 24h period. We found that both compensation and entrainment can be contradictory which motivated us to analyze them in more detail. To do so we developed an algorithm to systematically achieve temperature compensation within a specified temperature range in arbitrary oscillator models. This algorithm was then applied to several models proposed in the literature and found that especially minimal models are difficult to compensate, which is probably due to missing parts in these models. In contrast, very detailed models (e.g. the *Arabidopsis model*) showed the problem that their rhythm vanished at higher temperatures. Thus, it is necessary

to take into account experimentally identified temperature insensitive reactions, which can be easily done within our developed algorithm. In another study we investigated the interdependence of temperature compensation and entrainment. We proved analytically that only a temperature compensated rhythm preserves the range of its entrainment under different mean temperatures. Therefore, temperature compensation could have evolved to ensure the same entrainment possibilities over the seasons. In addition, we numerically analyzed literature models with different degrees of temperature compensation calculated by our previously developed algorithm. We found that also the waveform of the temperature signal, sinusoidal or rectangular, influences the entrainment region. While rectangular temperature signals have a broader entrainment region for small amplitudes, they generally lead to period-doubling or chaotic regions for large amplitudes and thus, have a smaller effective entrainment region. This has to be taken into account when designing entrainment experiments since a sinusoidal temperature entrainment signal resembles the natural situation.

We also modeled several hundreds of neurons in the central circadian pacemaker of mammals, the *suprachiasmatic nucleus* (SCN), and analyzed the adaptation of its circadian rhythm to seasonal changes in length of daytime. Experiments in mice found that the overall length of electrical activity of the SCN changes over the season. Using our model we showed that this can be related to properties of the network connecting the neurons. In particular we found that introducing more connections between neurons leads to a well synchronized short neuronal activity characteristic for mice living in winter. In contrast to this reducing the amount of the connections leads to a broadly distributed electrical activity of the neurons characteristic for mice living in summer. A mathematical analysis reveals that the broader distribution is related to the introduction of network communities that are well connected within each other but not between each other. By the very same mechanism our model was able to explain the experimentally observed reduced phase response of mice's clocks in summer. Since the neurons are not well synchronized, their overall phase response to a light pulse is diminished.

With the increasing knowledge about molecular interactions in biological systems and the revelation of underlying principles, the scientific community may start to consider the artificial design of biological systems. The relatively new field of synthetic biology has the aim to design biological circuits from basic modules that fulfill a specified function. In our first two studies we envisioned and implemented a biological *phase locked loop* (PLL). This control system is designed to adapt the frequency of a core oscillator to the frequency of an external signal. Our biochemical implementation of this principle comprises 12 species and the following modules: a core oscillator, modeled by a Goodwin-type oscillator with a circadian rhythm, a multiplication unit, modeled by a dimerization reaction, a low-pass filter, modeled by a linear enzyme cascade and an actuator, modeled by an enzyme degrading one of the oscillator species. Our implementation of the biochemical PLL showed the expected behavior of frequency adaptation. The advantage of our proposed modular design is that each module could be replaced

by another reaction system having a similar function. We found that a significant difference to most analog technical designs is the absence of negative values. Thus, each biological circuit always has to operate around a certain nominal point influenced by an "average" environmental signal. However, this is not the case for circuits performing digital operations. In our third study we implemented a biological frequency divider 1 : 17 using a binary counter with 5 bits. This seemingly strange ratio is motivated by the unusual 13 or 17-year eclosion rhythm of the insect genus *Magicicada*. The bits are implemented as chemical flip-flops using NOR-gates as demonstrated by Hinze et al. (2009). They cycle through 17 different states, where each new state is triggered by an internal or an external annual periodic signal, which could be the availability of nutrients in the soil. Compared to an analog signal integrator, for example, one which triggers an eclosion signal after a certain threshold is reached, the digital design is much more robust to internal and external noise. However, this robustness comes at the cost of approximately 150 involved species. Therefore, we do not expect to find such a device in the naturally evolved organisms. Nevertheless, again the modular implementation allows to replace modules. For example, after replacing the annual signal by different oscillator implementations we observed other frequency divider ratios like 1 : 6, 1 : 5 and 1 : 3. Thus, module replacement could explain the natural occurrence of 13 and 17 year eclosion rhythms.

Contents

Author: Zhiqiang Zhang (zhiqiangzhang@hust.edu.cn)

Title: *Research on the Computational Power of Numerical P Systems*

Supervisor: Linqiang Pan

Key Laboratory of Image Information Processing and Intelligent Control of Education Ministry of China, School of Automation, Huazhong University of Science and Technology, Wuhan 430074, Hubei, China

Date of presentation: November 20, 2016

Thesis abstract

Membrane computing is a branch of natural computing, investigating theoretical computing models inspired by the structure and functioning of living cell (including communications and cooperation of cells in tissues, organ, population of cells and brain) and studying the computing power and computing efficiency of models. The models in membrane computing are called membrane systems or P systems.

One of the most important topics in membrane computing domain is to investigate the computational power of all kinds of variants. Numerical P systems are an important class of P systems, which has a great potential in applications to robot control and economics. The study on numerical P systems is currently in development and the elements affecting the computational power of numerical P systems needs further investigation.

In the thesis, the factors affecting the computational power of numerical P systems are investigated. The main ingredients of a numerical P system include the membrane structure, the numerical variables in each membrane and the evolution rules (each one consists of a production function and a repartition protocol).

The factors investigated in the thesis include the ways of choosing the applied programs (sequential, all parallel and one parallel), the number of enzymatic variables used in enzymatic numerical systems, the control of evolving programs by means of a threshold, using a function to define the repartition coefficients, allowing to variables to migrate from a membrane to another one, and so on.

The detailed content of this dissertation is as follows:

The computational power of enzymatic numerical P systems affected by the working mode (the sequential mode) is investigated. It has been proved that enzymatic numerical P systems working in the all-parallel or one-parallel mode can reach universality. However it is still an open problem whether these systems can

also reach universality when working in the deterministically sequential mode. In the thesis, the universality of deterministically sequential numerical P systems is proved. Besides that, a known universal result on the non-deterministically sequential systems is improved by reducing the number of membranes and programs used in the universal systems. These results show that the computational power of enzymatic numerical P systems is robust with respect to the working mode.

The number of enzymatic variables used in the universal enzymatic numerical P systems is also investigated. It was proved that enzymatic numerical P systems can reach universality in the all-parallel or one-parallel mode. However the number of enzymatic variables needed in the universal systems has not been estimated. In the thesis, it is proved that for enzymatic numerical P systems used as number acceptors working in the all-parallel or one-parallel mode one enzyme is sufficient to reach universality, while for the one-parallel systems used as number recognizers, two membranes are sufficient to reach universality. These results show that the computational power of enzymatic numerical P systems is robust with respect to the number of enzymatic variables: even if the number of enzymatic variables is only one, the system still can reach universality.

The next topic investigated is the computational power of numerical P systems with thresholds. It is an open problem whether the computational power of numerical P systems without thresholds working in the all-parallel or one-parallel mode can reach universality. In the thesis, it is proved that numerical P systems with a threshold, with one membrane and linear production functions, working in the all-parallel or one-parallel mode are universal. These results show that thresholds can improve the computational power of numerical P systems.

Then, numerical P systems with a dynamical repartition protocol are considered. It is an open problem whether the computational power of numerical P systems with constant repartition coefficients working in the all-parallel or sequential mode can reach universality. In the thesis, the coefficients of variables in the repartition protocol are extended from constants to functions. The value of production function distributed to a variable is proportional to the value of its coefficient functions. The computational power of numerical P systems with dynamic repartition protocols working in the all-parallel and deterministically sequential modes is investigated. It is proved that one membrane, constant functions as production functions and linear functions with one variable as coefficient functions suffice to reach universality for the all-parallel systems; linear production function with at most one variable and linear function with at most two variables as coefficient function suffice to reach universality for the deterministically sequential systems. These results show that the function coefficients can improve the computational power of numerical P systems.

The computational power of systems as language generators is an important topic in membrane computing. No result about the computational power of numerical P systems generating languages is reported yet. In this work, the ways of generating languages by numerical P systems are formulated. The families of languages generated by numerical P systems, enzymatic numerical P systems, and

purely enzymatic numerical P systems working in the one-parallel mode are compared with the language families in the Chomsky hierarchy, including the finite, regular, context-free, and recursively enumerable languages.

Finally, we consider numerical P systems whose variables can change the compartments (we call them *migrating* variables). In the standard P systems, a crucial feature of objects is that they can pass through membranes, between regions of the same cell, between cells, or between cells and their environment. This feature is also introduced in numerical P systems, and the new variant is called numerical P systems with migrating variables (MNP systems). The computational power of MNP systems is investigated both as number generators and as string generators, working in the one-parallel or sequential mode. MNP systems used as numbers generators are proved to be universal when working in any of the above two modes. As string recognizers, the generative capacity of such systems is investigated having as a reference the families of languages in the Chomsky hierarchy, and a characterization of recursively enumerable languages is obtained. Migrating variables are an efficient feature for improving the computational power of numerical P systems in the sense of less membranes, lower polynomial degree and less variables in the universal systems.

Contents

Author: Lea Weber (weberlea@b-tu.de)

Title: *Implementation of an artificial evolution for optimal placement of processing units on a freely configurable two-dimensional grid map*

Supervisor:

Thomas Hinze, Brandenburg University of Technology Cottbus,
Institute of Computer Science

Thesis committee:

Uwe Hatnik, Fraunhofer Institute for Integrated Circuits IIS Dresden
Thomas Hinze, Brandenburg University of Technology Cottbus,
Institute of Computer Science
Peter Sauer, Brandenburg University of Technology Cottbus,
Institute of Computer Science

Date of presentation: April 22, 2016

Thesis abstract

Nature's ability in self-organisation is tremendous. For example, fungi create large underground supply and communication networks which are highly efficient as well as very tolerant against disturbances. In a 48 hours test, cultures of fungi were able to replicate infrastructures that engineers and analysts with access to great resources had optimised over decades. The key to this success lays in the process of evolution. Thus, in the program at hand, an artificial evolution was implemented to optimise the layout of a two dimensional grid. The grid consists of coherent paths as well as processing units for objects. They can access the processing units and will either pass through unchanged ("passage") or be combined ("composition"). The whole grid undergoes random mutation for several generations to find new layouts. Aiming for the quickest processing and passage of all objects through the grid, each layout is tested for its total processing duration. The artificial evolution optimises the placement of the processing units using each grid's individual processing duration as fitness criteria. With the tool, the user can configure the paths, the layout and the passing objects according to his problem, only needing to follow few rules.

The thesis at hand describes and explains the structure and algorithms of the tool in detail. The programme is separated into four consecutive parts: At first, the user configures his initial grid. Also, he sets the quantities and types of different processing units and same for the objects passing the grid. When all data is entered, it is edited and saved so that the following computation can easily access it. The tool now has a complete set of data to compute the initial processing duration. Therefore, the next step is to calculate the total number of time steps needed, until all objects have entered the grid, completed their way through by passing all their assigned processing units and exited again. During the process, queues may develop due to limited field capacity. Having obtained the initial fitness, the evolution is started to find an optimal layout. In order to create a population for the first generation, the initial grid is replicated and each version is mutated once. The evolution algorithm then repeatedly assesses each concerning its fitness, eliminates the slowest members of the current population and replaces them by mutating duplicates of the fastest grids. When no more improvements can be found, the resulting grid is shown to the user.

This type of topological optimisation problem is common in nature, science and industry. Due to being formulated abstractly, the software tool may be applied to a variety of scenarios. To demonstrate this flexibility, the thesis presents four case studies: minimisation of production time in a cabinet maker's workshop, quicker grocery shopping at an alternative supermarket, the efficient signal processing in a biological cell and a comparison of different processor architectures. In all cases, the initial processing time can be significantly improved. Interestingly, the resulting grids also show features that can be observed in natural evolution. All in all, the software proves to be a potent tool in optimising the placement of processing units in a two-dimensional grid. It can be found online at: `http://www-user.tu-cottbus.de/weberlea/public_html/gridtool/`

Contents

Author: Sergiu Ivanov (sivanov@colimite.fr)

Title: *On the Power and Universality of Biologically-inspired Models of Computation*

Supervisor:

Sergey Verlan

Assistant professor at the University Paris Est Créteil
LACL, Département Informatique
UFR Sciences et Technologie
Université Paris Est Créteil Val de Marne

Thesis committee:

Jérôme Durand-Lose	Université d'Orléans
Gheorghe Păun	Romanian Academy
Philippe Schnoebelen	CNRS & ENS de Cachan
Enrico Formenti	Université de Nice – Sophia Antipolis
Jean-Louis Giavitto	CNRS & IRCAM
Elisabeth Pelz	Université Paris Est Créteil

Date of presentation: June 23, 2015

Thesis abstract

The present thesis considers the problems of computational completeness and universality for several biologically-inspired models of computation: insertion-deletion systems, networks of evolutionary processors, and multiset rewriting systems. The presented results fall into two major categories: study of expressive power of the operations of insertion and deletion with and without control, and construction of universal multiset rewriting (P) systems of low descriptional complexity.

Insertion and deletion operations consist in adding or removing a subword from a given string if this subword is surrounded by some given contexts. The motivation for studying these operations comes from biology, as well as from linguistics and the theory of formal languages. In the first part of the present work we focus on insertion-deletion systems closely related to RNA editing, which essentially consists in inserting or deleting fragments of RNA molecules. An important feature of RNA

editing is the fact that the locus the operations are carried at is determined by certain sequences of nucleotides, which are always situated to the same side of the editing site. In terms of formal insertion and deletion, this phenomenon is modelled by rules which can only check their context on one side and not on the other. We show that allowing one-symbol insertion and deletion rules to check a two-symbol left context enables them to generate all regular languages. Moreover, we prove that allowing longer insertion and deletion contexts does not increase the computational power. We further consider insertion-deletion systems with additional control over rule applications and show that the computational completeness can be achieved by systems with very small rules.

The motivation for studying insertion-deletion systems also comes from the domain of computer security, for the purposes of which a special kind of insertion-deletion systems called leftist grammars was introduced. In this work we propose a novel graphical instrument for visual analysis of the dynamics of such systems.

The second part of the present thesis is concerned with the universality problem, which consists in finding a fixed element able to simulate the work any other computing device. We start by considering networks of evolutionary processors (NEPs), a computational model inspired by the way genetic information is processed in the living cell, and construct universal NEPs with very few rules. We then focus on multiset rewriting systems (P systems), which model the chemical processes running in the biological cell. For historical reasons, we formulate our results in terms of Petri nets. We construct a series of universal Petri nets and give several techniques for reducing the numbers of places, transitions, inhibitor arcs, and the maximal transition degree. Some of these techniques rely on a generalisation of conventional register machines, proposed in this thesis, which allows multiple register checks and operations to be performed in a single state transition.

Contents

Author: Ciprian Dragomir (c.dragomir@sheffield.ac.uk)
Title: *Formal Verification of P Systems*

Supervisors:

Prof Marian Gheorghe, University of Bradford, UK
Prof Georg Struth, University of Sheffield, UK

Thesis committee:

External examiner: Prof Maciej Koutny, Newcastle University, UK
Internal examiner: Dr Mike Stannett, University of Sheffield, UK

Date of presentation: 30th June 2016

Thesis abstract

Membrane systems, also known as P systems, constitute an innovative computational paradigm inspired by the structure and dynamics of the living cell. A P system consists of a hierarchical arrangement of compartments and a finite set of multiset rewriting and communication rules, which operate in a maximally parallel manner. The *organic* vision of concurrent dynamics captured by membrane systems stands in antithesis with *conventional* formal modelling methods which focus on algebraic descriptions of distributed systems. As a consequence, verifying such models in a mathematically rigorous way is often elusive and indeed counter-intuitive when considering established approaches, which generally require sequential process representations or highly abstract theoretical frameworks. The prevalent investigations with this objective in the field of membrane computing are ambivalent and inconclusive in the wider application scope of P systems.

In this thesis we directly address the formal verification of membrane systems by means of model checking. A fundamental distinction between the agnostic perspective on parallelism, advocated by process calculi, and P systems' emblematic *maximally parallel execution strategy* is identified. On this basis, we establish that traditional process models are decidedly inadequate for expressing P system transitions for the purpose of formal verification. The incompatibility is most eloquently reflected in the state space growth, relative to the system's dynamic components

(processes and compartments), which is *exponential* for process models and *linear* for membrane systems. The observation is decisive for this research project: on one hand it implies the feasibility of model checking P systems, and on the other hand it underlines the suitability of this formal verification technique in the context of membrane computing. Model checking entails an exhaustive state space exploration and does not derive inferences based on the independent instructions comprising a state transition. In this respect, we define a new sequential modelling strategy which is optimal for membrane systems and targets the SPIN formal verification tool.

We introduce elementary P systems, a distributed computational model which subsumes the feature diversity of the membrane computing paradigm and distils its functional vocabulary. A suite of supporting software tools which gravitate around this formalism has also been developed and consists of: 1. the *eps modelling language* for elementary P systems; 2. a parser for the eps specification; 3. a model simulator and 4. a translation tool which targets the Promela specification of the SPIN model checker.

The formal verification approach proposed in this thesis is progressively demonstrated in four heterogeneous case studies, featuring 1. a parallel algorithm applicable to a structured model; 2. a linear time solution to an NP-complete problem; 3. an innovative implementation of the Dining Philosophers scenario (a synchronisation problem) using an elementary P system and 4. a quantitative analysis of a simple random process implemented without the support of a probabilistic model.

Contents

Author: Tao Wang (wangatao2005@163.com)
Title: *Spiking neural P systems and their applications in fault diagnosis of electric power systems*

Supervisor:

Gexiang Zhang
School of Electrical Engineering
Southwest Jiaotong University, Chengdu, 610031, China

Date of presentation: November 18, 2016

Thesis Abstract

When faults occur in electric power systems, a large amount of alarm information will pour into the control center. The abundant information provides various references for fault diagnosis. However, it also makes supervisors too busy to handle timely. Meanwhile, due to the influence of factors such as technology, equipment and environment, in the alarm information is likely to exist uncertainties such as information error, loss and distortion. Thus, computers and artificial intelligence technology are needed when analyzing and processing these alarm information in fault diagnosis of electric power systems. So, information processing ability of fault diagnosis methods is very important. Spiking neural P systems (SNPS) is a special type of neural-like P systems inspired by communication mechanisms between biological neurons. Its computing models are dynamic, discrete, distributed and parallel ones with powerful computing ability and information processing capacity, which are suitable for solving fault diagnosis problems of electric power systems. Therefore, from the way of model abstraction, algorithm design and practical examples and based on the framework and theory of membrane computing and SNPS, this paper focuses on improving the information processing and self-adaption ability of SNPS, and proposes SNPS application models to handle fault diagnosis in electric power systems.

Firstly, based on the demands of knowledge representation and knowledge reasoning in fault diagnosis of electric power systems and considering the function and structures of actual biological networks, this paper designs SNPS application models from the way of real number fuzzy reasoning and proposes two kinds of fuzzy reasoning spiking neural P systems with real numbers (rFRSNPS), in which parameters such as output weights (synaptic weights), firing thresholds and information prejudging parameters are considered and definitions of neurons, synapses,

spikes and firing rules are redefined. The proposed rFRSNPSs called weighted fuzzy reasoning spiking neural P systems with real numbers (rWFRSNPS) and weighted prejudging fuzzy reasoning spiking neural P systems with real numbers (rWPFRSNPS), respectively. Then, reasoning algorithms of rWFRSNPSs and rWPFRSNPSs are designed and fuzzy production rules are modeled based on them, respectively. If the truth values of input neurons and rule neurons are given, then the proposed algorithms can automatically reason out the truth values of other proposition neurons in an rFRSNPS to fulfill the knowledge representation and reasoning. The intuitive graphical modeling process and simple matrix reasoning computing are quite adaptive to complex knowledge representation and reasoning. On a background of transmission network fault diagnosis, this paper designs layered diagnosis models based on rWPFRSNPS for main sections in power systems and proposes a fault diagnosis method based on rWPFRSNPS in which a temporal information processing method based on cause-effect networks is considered. This diagnosis method results in a decrease of matrix dimension, a release of calculation burden and an improvement of model adaptive ability.

In order to improve the ability of SNPS in processing uncertain and imprecise information and considering the actual existence situation and real communication process of knowledge and information, this paper designs SNPS application models from the way of fuzzy number fuzzy reasoning and proposes fuzzy reasoning spiking neural P systems with trapezoidal fuzzy numbers (tFRSNPS), in which trapezoidal fuzzy numbers are considered and definitions of neurons, spikes, ways of pulse accumulation, firing rules and firing conditions are redefined. Then, a reasoning algorithm is designed and fuzzy production rules are modeled based on tFRSNPSs. This paper also proposes a fault diagnosis method based on tFRSNPS in which fault fuzzy production rule sets for main sections in power systems are given. The introduction of trapezoidal fuzzy numbers guarantees that SNPS has a strong ability to express and reason fuzzy knowledge without prejudging the inputs and decreases the fault diagnosis complexity.

Optimization spiking neural P systems (OSNPS), independent of evolutionary operators in evolutionary computation to achieve individual evolution, is a kind of algorithm with superior performances. This paper employs binary strings to represent chromosomes (individuals) and makes the first attempt to solve fault diagnosis of power systems by using the idea of optimization methods in the framework of MC and proposes a fault diagnosis method based on OSNPS.

To verify the practical application performance of SNPS, this paper discusses the applications of rWFRSNPS, rWPFRSNPS, tFRSNPS and OSNPS in fault diagnosis of transmission networks, where the 14 bus power system and 220kV local power system are used to test their feasibility and effectiveness. The experimental results on several cases show that both the proposed three kinds of fuzzy reasoning spiking neural P systems and OSNPS produce prospective diagnosis results with different characteristics.

Contents

Calls to MC Conferences/Meetings

Call for Participation
Fifteenth Brainstorming Week on Membrane Computing (BWMC'17)

Research Group on Natural Computing
Universidad de Sevilla, Spain
January 31 – February 3, 2017

Goal: Similarly to the previous editions, the goal is to gather together researchers interested in Membrane Computing (theory and applications), for exchanging ideas, problems, solutions, for working together, in a friendly framework.

To this aim, the meeting combines joint work sessions (usually on the afternoons), together with short and provocative talks, possibly presenting ongoing work and/or open problems.

Participants are strongly encouraged to circulate open problems and research ideas before the event. Please send to `ariscosn@us.es` your proposals, and we will post them on the BWMC web page.

Researchers from MC community, and from other areas as well, are welcome to submit their open problems, even if they cannot attend the meeting. The Brainstorming Week is an excellent opportunity to discuss about potential collaborations and interdisciplinary approaches.

Dates: January 31, 2017 (Tuesday) - February 3, 2017 (Friday)

Organizing institution:

- Research Group on Natural Computing (RGNC)
- Department of Computer Science and Artificial Intelligence
- Universidad de Sevilla, Spain

Venue:

- E.T.S. Ingeniería Informática, module H, first floor
  Avda. Reina Mercedes s/n. (41012) Sevilla

Web page: News will be communicated through the BWMC'17 web page at the Research Group on Natural Computing: `http://www.gcn.us.es/15bwmc` and also through the P systems web page: `http://ppage.psystems.eu`

Proceedings: As usual, at about two months after the meeting a proceedings volume will be published as a research report of RGNC. Then, a selection of final papers will be published in a special issue of a journal as it was always the case with the previous editions.

Here is the complete list (all journals, except two, are currently indexed in the ISI JCR):

- BWMC 2003: *Natural Computing* 2 (3), 2003, and *New Generation Computing* 22 (4), 2004;
- BWMC 2004: *Journal of Universal Computer Science* 10 (5), 2004, and *Soft Computing* 9 (5), 2005;
- BWMC 2005: *Int. Journal of Foundations of Computer Science* 17 (1), 2006);
- BWMC 2006: *Theoretical Computer Science* 372 (2-3), 2007;
- BWMC 2007: *Int. Journal of Unconventional Computing* 5 (5), 2009;
- BWMC 2008: *Fundamenta Informaticae* 87 (1), 2008;
- BWMC 2009: *Int. Journal of Computers, Control and Communication* 4 (3), 2009;
- BWMC 2010: *Romanian Journal of Information Science and Technology* 13 (2), 2010;
- BWMC 2011: *Int. Journal of Natural Computing Research* 2 (2-3), 2011;
- BWMC 2012: *Int. Journal of Computer Mathematics* 99 (4), 2013;
- BWMC 2013: *Int. Journal of Unconventional Computing* 9(5-6) 2013;
- BWMC 2014: *Fundamenta Informaticae* 134 (1-2), 2014;
- BWMC 2015: *Natural Computing* 15 (4), 2016;
- BWMC 2016: *Theoretical Computer Science* – (to appear)

Registration: In order to register the participants should email Ana M. Ruiz (`anarumez@us.es`). A registration fee of 100 EUR will be requested on arrival at the registration desk. This will cover workshop materials, coffee breaks, lunch, social dinner and proceedings. Several accommodation options are listed on the webpage of the BWMC'17 that need to be booked by the participants.
Important advice: Bookings should be arranged the sooner the better, in order to avoid availability restrictions. In particular, reservations for the University residence should be made not later than January 20th 2017 (by email to `anarumez@us.es`).

Organizing committee:

- Gh. Păun, **Co–Chair**
- M.J. Pérez Jiménez, **Co–Chair**
- C. Graciani
- D. Orellana Martín
- A. Riscos Núñez
- A. Romero Jiménez
- A.M. Ruiz Gómez
- L. Valencia Cabrera

Supported by: Dpt. Ciencias de la Computación e Inteligencia Artificial, Universidad de Sevilla

18th International Conference on Membrane Computing (CMC18)
24-28 July 2017, Bradford (United Kingdom)

FIRST CALL FOR PAPERS

SCOPE AND LOCATION

The Conference on Membrane Computing (CMC) series started in 2000 as the Workshop on Multiset Processing. The first Workshop on Membrane Computing was organized in Curtea de Argeş, Romania, in 2001. In 2010 it was transformed into a conference, CMC11. The last edition, CMC17, was held in Milan, Italy, in 2016. Nowadays a Steering Committee takes care of the continuation of the CMC series which is organized under the auspices of the International Membrane Computing Society (IMCS[1]).

CMC18 is the 18th edition of the International Conference on Membrane Computing series. The conference will take place at the University of Bradford, in Bradford, UK. Bradford is located in the north of England, in North-West Yorkshire, and is a very multicultural city. It has the third largest economy in the Yorkshire, worth over 8.3bn. Bradford has been designed the world's first UNESCO City of Film, it is a thriving cultural destination, with interesting museums and galleries and attractive theatres[2]. The climate is... British, i.e., mild – not too cold in the winter, relatively warm in the summer, with a stable invariant for both: rain! Bradford is famous for some of the finest Asian food in the UK, being crowned Curry Capital of Britain for five consecutive years[3]!

The goal of CMC18 is to bring together researchers working in membrane computing and related fields, in a friendly atmosphere enhancing communication,

[1] http://membranecomputing.net/IMCSBulletin/index.php
[2] https://www.bradford.gov.uk/business/bradford-economy/about-bradfords-economy/
[3] http://www.visitbradford.com/explore/Bradford_Curry_Guide.aspx

cooperation and continuing the tradition of the past meetings. Membrane computing (P systems theory) is an area of computer science aiming to abstract computing ideas and models from the structure and the functioning of living cells, as well as from the way the cells are organized in tissues or higher order structures.

CONFERENCE FORMAT

This edition aims to have the following format, although some changes might occur based on suggestions made by the Steering Committee. It is planned to include: (1) three days of communications with invited speakers and short and long contribution talks according to the papers submitted and the reviews of the Program Committee, and (2) an Interaction Day similar in spirit to the Brainstorming Week on Membrane Computing (BWMC) that is usually organized in Sevilla every year, where the participants will work in a collaborative way to attack open problems and propose new approaches, problems and results. An IMCS General Assembly will be also organized.

SUBMISSION OF PAPERS

Authors are invited to submit papers presenting original, unpublished research in PDF format. There are two tracks for submission:

(1) full paper (of a reasonable length),

(2) extended abstract for poster presentation (maximum four pages). Typical extended abstracts present significant work-in-progress, late-breaking results, or contributions from students new in the field or at the start of their research career.

Only electronic submissions are accepted. Papers should be written in LaTeX and formatted according to the usual LNCS article style which can be downloaded at Springer's LNCS website (`http://www.springer.com/lncs`). Please, include all source files as well as all additional files (figures etc.), and also attach a PDF version of the submission.

Submissions have to be sent through the EasyChair web Page – a link will be circulated soon.

DEADLINES

- Deadline for submissions: 17 April 2017
- Notification of acceptance: 29 May 2017
- Final version: 19 June 2017
- Conference: 24 - 28 July 2017

PROCEEDINGS

A pre-proceedings volume will be available at the conference in electronic format, online, and optionally hard copies. A volume devoted to selected and additionally revised papers will be published in the Lecture Notes in Computer Science series of Springer-Verlag after the event.

STEERING COMMITTEE

Henry Adorna (Quezon City, Philippines)
Artiom Alhazov (Chişinău, Moldova)
Bogdan Aman (Iaşi, Romania)
Matteo Cavaliere (Edinburgh, Scotland)
Erzsébet Csuhaj-Varjú (Budapest, Hungary)
Giuditta Franco (Verona, Italy)
Rudolf Freund (Wien, Austria)
Marian Gheorghe (Bradford, UK) - Honorary Member
Thomas Hinze (Cottbus, Germany)
Florentin Ipate (Bucharest, Romania)
Shankara N. Krishna (Bombay, India)
Alberto Leporati (Milan, Italy)
Taishin Y. Nishida (Toyama, Japan)
Linqiang Pan (Wuhan, China)
Gheorghe Păun (Bucharest, Romania) - Honorary Member
Agustin Riscos-Núñez (Sevilla, Spain)
José M. Sempere (Valencia, Spain)
Petr Sosík (Opava, Czech Republic)
Kumbakonam Govindarajan Subramanian (Penang, Malaysia)
György Vaszil (Debrecen, Hungary)
Sergey Verlan (Paris, France)
Claudio Zandron (Milan, Italy) - Chair
Gexiang Zhang (Chengdu, China)

PROGRAM COMMITTEE

To be announced

ORGANIZING COMMITTEE

Marian Gheorghe (Bradford, UK) - Co-chair
Savas Konur (Bradford, UK) - Co-chair
Raluca Lefticaru (Bradford, UK) - Communication Chair
Dan Neagu (Bradford, UK)

CONTACT INFO

Please do not hesitate to contact us if you have any question.

Marian Gheorghe (email: m.gheorghe@bradford.ac.uk)
Savas Konur (email: s.konur@bradford.ac.uk)

Call for Papers WMC at UCNC

Web Site

https://ucnc2017.csce.uark.edu/workshops/membranessysbio/

The Workshop on Membrane Computing is associated with the 16th International Conference on Unconventional Computation and Natural Computation (UCNC 2017).

June 5-9, 2017, University of Arkansas
Fayetteville, Arkansas, USA
https://ucnc2017.uark.edu/

Workshop

The goal of the Workshop is to bring together researchers working in membrane computing and researchers working in systems and synthetic biology, in a friendly atmosphere enhancing communication and cooperation.

Membrane computing is an area of computer science aiming to abstract computing ideas and models from the structure and the functioning of living cells, as well as from the way the cells are organized and interact in tissues. Synthetic biology is concerned with the design and construction of new biological entities as well as with redesign of existing biological systems.

Systems biology integrates experimental and computational research to study how the behaviour of a biological system results from the interactions of its components.

The Workshop will focus on experimental and theoretical developments in membrane computing and its interface with the areas of systems and synthetic biology.

Invited Speakers

Alvaro Sanchez (Yale University)
Sergey Verlan (University Paris Est, Créteil)

Instructions for Authors

Authors are invited to submit original papers to the email address:
wmc-ucnc2017@ed.ac.uk

Papers must be submitted in Portable Document Format (PDF) and must be prepared in LATEX according to the LNCS format:

http://www.springer.com/computer/lncs/lncs+authors

The papers must not have been submitted simultaneously to other conferences or workshops with published proceedings. All accepted papers will be published in the workshop proceedings and must be presented at the workshop.

The authors of selected papers from the workshop will be invited to submit extended versions for publication in the special issue of *Natural Computing* (Springer).

Important Dates

Submission deadline: 12th February 2017
Notification of Acceptance: 5th March 2017
Workshop: June 6th 2017

Program Committee

M. Cavaliere (co-chair, University of Edinburgh)
D. Genova, University of North Florida
T. Hinze, University of Jena
V. Manca, University of Verona
L. Pan, Huazhong University of Science and Technology
A. Păun, UPM and University of Bucharest
A. Rodríguez-Patón (co-chair, Universidad Politécnica de Madrid)
P. Sosík, Silesian University
X. Zeng, Xiamen University

Reports on MC Conferences/Meetings

Report on CMC17 – The Seventeenth Conference on Membrane Computing

Alberto Leporati, Claudio Zandron

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca
Viale Sarca 336/14, 20126 Milano, Italy
{leporati, zandron}@disco.unimib.it

The 17$^{\text{th}}$ International Conference on Membrane Computing (CMC17) was held in Milan, Italy, from July 25 to July 29, 2016. It has been organized, under the auspices of the International Membrane Computing Society (IMCS) and the European Molecular Computing Consortium (EMCC), by the Research Group on Molecular Computing of the Department of Informatics, Systems, and Communication, at the University of Milano-Bicocca.

CMC17 consisted of three different parts: the first day, representatives of research groups working on membrane computing presented recent research activities of the group, described the composition of the research team, and presented research networks and projects they are involved in.

From Tuesday to Thursday the conference continued with standard sessions. The session of Tuesday, July 26, opened with an obituary for Professor Solomon Marcus, who passed away on March 17, 2016. The obituary, presented by Gheorghe Păun, underlined the various outstanding contributions by Prof. Marcus in different areas: mathematics, computational linguistics, computer science, but also poetics, linguistics, and philosophy, stressing his contributions to membrane computing.

Four invited lectures were delivered during the standard sessions, by researchers working in Membrane Computing since many years. The invited contributions were the following: "Coping with Dynamical Structures for Useful Applications of Membrane Computing", by Thomas Hinze (Cottbus, Germany), "Applications of P Systems in Population Biology and Ecology" by Paolo Milazzo (Pisa, Italy), "Borderlines of Efficiency: What's Up?" by Agustín Riscos-Núñez (Sevilla, Spain), and "Eco-Evo Dynamics and the Role of Cellular Computing" by Matteo Cavaliere (Edinburgh, UK).

In a special dedicated session on Tuesday afternoon (July 26), the first meeting of the International Membrane Computing Society was also held. The goal of the meeting (open to all conference participants) was to introduce the motivations and the work of the society, and to discuss the short and medium term goal and activities.

The last day of the conference (Friday, July 29) was devoted to interaction between participants, to discuss open problems and propose new research topics.

The Best Paper Award of CMC17 was won by Zsolt Gazdag and Gábor Kolonits, for the paper entitled "Remarks on the Computational Power of Some Restricted Variants of P Systems with Active Membranes".

This edition of the conference planned also two Best Student Paper Awards, sponsored by Springer-Verlag with a sum of 250 euro each. The awards were won by the paper "Walking Membranes: Grid-exploring P Systems with Artificial Evolution for Multi-purpose Topological Optimisation of Cascaded Processes", by Thomas Hinze, Lea Louise Weber, and Uwe Hatnik, and by the paper "Remarks on the Computational Power of Some Restricted Variants of P Systems with Active Membranes" by Zsolt Gazdag and Gábor Kolonits.

The volume with the conference pre-proceedings was available at the conference (printed and electronically), while the final proceedings volume, containing a selection of additionally referred papers, will be published by Springer-Verlag in the LNCS series.

Conference webpage: `http://cmc17.disco.unimib.it/`

The Organizing Committee of CMC17 consisted of

- Alberto Leporati (Milan, Italy) – Co-chair,
- Luca Manzoni (Milan, Italy),
- Antonio Enrico Porreca (Milan, Italy), and
- Claudio Zandron (Milan, Italy) – Co-chair.

The Programme Committee of CMC17 consisted of

- Artiom Alhazov (Chişinău, Moldova),
- Bogdan Aman (Iaşi, Romania),
- Lucie Ciencialová (Opava, Czech Republic),
- Erzsébet Csuhaj-Varjú (Budapest, Hungary),
- Giuditta Franco (Verona, Italy),
- Rudolf Freund (Wien, Austria),
- Marian Gheorghe (Bradford, UK),
- Thomas Hinze (Cottbus, Germany),
- Florentin Ipate (Bucharest, Romania),
- Shankara Narayanan Krishna (Bombay, India),
- Alberto Leporati (Milan, Italy) – Co-chair,
- Vincenzo Manca (Verona, Italy),

- Maurice Margenstern (Metz, France),
- Giancarlo Mauri (Milan, Italy),
- Radu Nicolescu (Auckland, New Zealand),
- Linqiang Pan (Wuhan, China),
- Gheorghe Păun (Bucharest, Romania),
- Mario de Jesús Pérez-Jiménez (Sevilla, Spain),
- Antonio E. Porreca (Milan, Italy),
- Agustín Riscos-Núñez (Sevilla, Spain),
- José M. Sempere (Valencia, Spain),
- Petr Sosík (Opava, Czech Republic),
- György Vaszil (Debrecen, Hungary),
- Sergey Verlan (Paris, France),
- Claudio Zandron (Milan, Italy) – Co-chair, and
- Gexiang Zhang (Chengdu, Sichuan, China).

The regular papers presented at the conference were the following:

- Simulating R Systems by P Systems, by Artiom Alhazov, Bogdan Aman, Rudolf Freund, and Sergiu Ivanov
- Purely Catalytic P Systems over Integers and Their Generative Power, by Artiom Alhazov, Omar Belingheri, Rudolf Freund, Sergiu Ivanov, Antonio E. Porreca, and Claudio Zandron
- Semilinear Sets, Register Machines, and Integer Vector Addition (P) Systems, by Artiom Alhazov, Omar Belingheri, Rudolf Freund, Sergiu Ivanov, Antonio E. Porreca, and Claudio Zandron
- Maximal Variants of the Set Derivation Mode, by Artiom Alhazov, Rudolf Freund, and Sergey Verlan
- Computational Power of Protein Networks, by Bogdan Aman and Gabriel Ciobanu
- Comparative Analysis of Statistical Model Checking Tools, by Mehmet Emin Bakir, Marian Gheorghe, Savas Konur, and Mike Stannett
- P Colonies with Evolving Environment, by Lucie Ciencialová, Luděk Cienciala, and Petr Sosík
- Continuation Passing Semantics for Membrane Systems, by Gabriel Ciobanu and Eneia Nicolae Todoran
- Secure Dispersion of Robots in a Swarm Using P Colonies, by Andrei George Florea and Cătălin Buiu
- Remarks on the Computational Power of Some Restricted Variants of P Systems with Active Membranes, by Zsolt Gazdag and Gábor Kolonits
- Kernel P Systems Modelling, Testing and Verification – Sorting Case Study, by Marian Gheorghe, Rodica Ceterchi, Florentin Ipate, and Savas Konur
- Walking Membranes: Grid-exploring P Systems with Artificial Evolution for Multi-purpose Topological Optimisation of Cascaded Processes, by Thomas Hinze, Lea Louise Weber, and Uwe Hatnik

- Agent-Based Simulation of Kernel P Systems with Division Rules Using FLAME, by Raluca Lefticaru, Luis Felipe Macías-Ramos, Ionuţ Mihai Niculescu, and Laurenţiu Mierlă
- Shallow Non-Confluent P Systems, by Alberto Leporati, Luca Manzoni, Giancarlo Mauri, Antonio E. Porreca, and Claudio Zandron
- Data Structures with cP Systems or Byzantine Succintly, by Radu Nicolescu
- Rewriting P Systems with Flat-splicing Rules, by Linqiang Pan, Bosheng Song, and K.G. Subramanian
- The Improved Apriori Algorithm Based on the Tissue-like P System, by Yuzhen Zhao, Xiyu Liu, and Wenxing Sun

The extended abstracts presented at the conference were the following:

- Traces of Computations by Generalized Communicating P Systems, by Ákos Balaskó and Erzsébet Csuhaj-Varjú
- Chemical Term Reduction with Active P Systems, by Péter Battyányi and György Vaszil
- Extensions of P Colonies, by Erzsébet Csuhaj-Varjú
- On Minimal Multiset Grammars, by Giuditta Franco
- Transmission Line Fault Classification Based on Fuzzy Reasoning Spiking Neural P Systems, by Kang Huang
- Recent Results and Problems Concerning P Colony Automata, by Kristóf Kántor and György Vaszil
- A Characterization of Symport/Antiport P Systems Through Information Theory, by José M. Sempere

A Summary of The 5th Asian Conference on Membrane Computing (ACMC 2016)

Linqiang Pan[1], Gexiang Zhang[2], Ravie Chandren Muniyandi[3], Bosheng Song[1]

[1] School of Automation, Huazhong University of Science and Technology, Wuhan 430074, China
[2] School of Electrical Engineering, Southwest Jiaotong University, Chengdu 610031, China
[3] Center for Software Technology and Management, Faculty of Information Science and Technology, National University of Malaysia, Bangi 43600, Malaysia

The 5th Asian Conference on Membrane Computing (ACMC 2016) was held in National University of Malaysia (UKM), Malaysia, from November 14 to November 16, 2016. This conference was organized by a joint effort of the Center for Software Technology and Management (SOFTAM), the Centre for Research Excellence at the Faculty of Information Science and Technology, UKM, and International Membrane Computing Society (IMCS).

ACMC 2016 is one of the flagship conferences on Membrane Computing aiming to provide a high-level international forum for researchers working in membrane computing and related areas, especially for the ones from the Asian region.

ACMC 2016 has received 39 submissions. Each submission was reviewed by at least three programme committee members. Thirty-five papers were accepted for oral presentations during the conference. All the 35 papers have been published in the pre-proceeding of ACMC 2016 and recommended to be published in three international journals and two Springer series after further standard review processes. A selection with 5 papers is combined with the papers from the 14th Brainstorming Week on Membrane Computing (BWMC2016) to be published in the journal *Theoretical Computer Science*. A selection consisting of 10 papers are considered to be included in a special issue of *Romanian Journal of Information Science and Technology*. Three papers are recommended to be published in the Special Issue on Metaheuristic Optimization: Algorithmic Design and Applications of *Journal of Optimization*. Two papers will be published in *Lecture Notes in Computer Science* together with a selection from the 17th International Conference on Membrane Computing (CMC). The remaining 15 papers are being published in the volume of *Communications in Computer and Information Science* of The 11th International Conference on Bio-inspired Computing: Theories and Applications (BIC-TA 2016).

This conference has been organized as a friendly interactive platform with several keynote lectures, oral presentations and specialized discussion sessions, covering a wide range of topics on membrane computing, including theory, applications, implementation and various related aspects such as DNA computing and bioinformatics.

ACMC 2016 was well organized and was a very successful conference. A special thanks is due to the Organising Committee and Supporting Committee at the Universiti Kebangsaan Malaysia, Malaysia, all the authors and participants, the Steering Committee, Programme Committee, and all the editors of the five publications mentioned above.

In the opening ceremony of ACMC 2016, The President of IMCS, Prof. Gexiang Zhang, delivered an introductive report on IMCS and ACMC. The invited talks are summarized as follows:

Invited Talk 1: Real-life Applications with Membrane Computing
Gexiang Zhang, Professor
Southwest Jiaotong University, Chengdu, China
E-mail: zhgxdylan@126.com

This talk reviews the many facts of real-life applications of membrane computing, such as engineering optimization problems solved by membrane-inspired evolutionary algorithms, fault diagnosis of electric power systems with fuzzy reasoning spiking neural P systems, mobile robots membrane controllers designed by using (enzymatic) numerical P systems and ecosystems modeling with population dynamics P systems. This talk also discusses the challenging issues in the real-life applications of membrane computing.

Invited Talk 2: DNA Computing and Splicing P Systems
D. Gnanaraj Thomas, Associate professor
Madras Christian College (MCC), Chennai, India
E-mail: dgthomasmcc@yahoo.com

The aim of this talk is to give the audience a glimpse of the basic notions of DNA Computing and Splicing P Systems along with recent results and developments with potential applications that have appeared in the literature leading to future directions of research.

Invited Talk 3: Spiking Neural P Systems: Insights and Challenges
Henry N. Adorna, Professor
University of the Philippines, Diliman, Quezon City, Philippines
E-mail: hnadorna@dcs.upd.edu.ph

An SN P system is a variant of P systems that was introduced by Ionescu et al. in 2006. Since its introduction different modifications have been added to the original model. Each variant of the original SN P systems is shown to be potentially useful in attacking **NP** problems. Among the newest addition to the SN P systems family is the so-called spiking neural P systems with structural

plasticity or SNPSP systems, which was introduced in 2013. Structural plasticity refers to synapse creation and synapse deletion. An initial topology of an SN P system is given, and other synapses (connections) among neurons are created and deleted for the purpose of computing. This talk reports some results obtained since 2013 and provides some insights and possible directions to pursue in the future.

Invited Talk 4: Biometric Challenges and Bio-Inspired Approaches

Ibrahim Venkat, Senior Lecturer
Universiti Sains Malaysia, Pulau Pinang, Malaysia
E-mail: ibra@usm.my

In this talk, a glimse on the recent state-of-the-art on biometrics and some interesting case studies on recognizing complex faces and gaits is presented. Finally, some potential ideas on the impact of bio-inspired techniques towards the advancement of these biometrics are proposed.

The following is the list of papers delivered during the ACMC 2016.

1. Lian Ye, Ping Guo, A P System Based on Negative Selection for Classification.
2. Wenping Yu, Jun Ming, Jun Wang, Hong Peng, Ke Chen, Chengyu Tao, Fault Diagnosis of Power Systems Using Fuzzy Spiking Neural P Systems with Interval-valued Fuzzy Numbers.
3. Philomenal Karoline, Helen Chandra, Saroja Theerdus Kalavathy, Mary Imelda Jayaseeli, Simulation of Fuzzy Acsh on Membranes with Michaleis-Menten Kinetics.
4. Ping Guo, Wei Xu, A Family P System of Realizing RSA Algorithm.
5. Tingfang Wu, Yanfeng Wang, Suxia Jiang, Xiaolong Shi, Spiking Neural P Systems with Rules on Synapses and Anti-Spikes.
6. Xiyu Liu, Yuzhen Zhao, Wenping Wang, A General Object Oriented Description for Membrane Computing.
7. Kelvin Buno, Francis George Cabarle, Henry Adorna, Marj Calabia, Solving the N-Queens Problem Using dP Systems with Active Membranes.
8. Juan Hu, Guangchun Chen, Hong Peng, Jun Wang, Xiangnian Huang, Xiaohui Luo, Matrix Representation of Parallel Computation for Spiking Neural P Systems.
9. Wenbo Dong, Zhou Kang, Qi Huaqing, Discrete GSO Based on Time Window Division for Solving Multi-objective VRPTW.
10. Yingying Duan, Kang Zhou, Huaqing Qi, MPSO with Exchange-Tree Mechanism for Traffic Network Layout Optimization Problem.
11. Yingying Duan, Kang Zhou, Huaqing Qi, Application of MTabu with Time Classifier in Vehicle Routing Problem with Time Windows.
12. Nestine Hope Hernandez, Francis George Cabarle, Solving Some Computationally Hard Problems Using Numerical P Systems with Thresholds.
13. Xun Wang, Ma Tongmao, Shaohua Hao, Tao Song, Shanchen Pang, Spiking Neural P Systems with Anti-spikes and without Annihilating Priority as Number Generator.

14. Yangyang He, Gexiang Zhang, Kang Huang, Tao Wang, Implementation of Information Fusion Based on Spike Neural P Systems.
15. Ali Maroosi, Ravie Chandren Muniyandi, Criteria for Designing Membrane Systems.
16. Naeimeh Elkhani, Ravie Chandren Muniyandi, Multiple Core Execution of Kernel P System – Multi Objective Binary Particle Swarm Optimization Feature Selection Method for Cancer Classification.
17. Jie Xue, Xiyu Liu, Wenxing Sun, A Density-Grid Based Algorithm by Weighted Spiking Neural P Systems with Antispikes and Astrocyte in Spatial Cluster Analysis.
18. Helen Chandra, Saroja Theerdus Kalavathy, M. Nithya Kalyani, The Computational Power of Array P System with Mate Operation.
19. Zhiqiang Zhang, Yansen Su, Linqiang Pan, The Computational Power of Enzymatic Numerical P Systems Working in the Sequential Mode.
20. Bosheng Song, Yanfeng Wang, Linqiang Pan, Tissue P Systems with Promoters Working in the Flat Maximally Parallel Way.
21. Nurul Liyana Mohamad Zulkufli, Sherzod Turaev, Mohd Izzuddin Mohd Tamrin, Messikh Azeddine, The Computational Power of Watson-Crick Grammars: Revisited.
22. Kazeem Rufai, Ravie Chandren Muniyandi, Zulaiha Ali Othman, Integrated Membrane Computing Framework for Modeling Intrusion Detection Systems.
23. Liu Shuo, Zhou Kang, Zeng Shan, Qi Huaqing, Chen Xing, An Improvement of Small Universal Spiking Neural P Systems with Anti-Spikes.
24. Sureshkumar Williams, Kalpana Mahalingam, Rama Raghavan, Process Guided P System with Graph Productions and Applications.
25. Katrina Gapuz, Ephraim Mendoza, Richelle Ann Juayong, Nestine Hope Hernandez, Francis George Cabarle, Henry Adorna, Solution to Motif Finding Problem in Membranes.
26. Linqiang Pan, Bosheng Song, Atulya K. Nagar, K G Subramanian, Language Generating Flat Splicing P Systems.
27. Yang Yunying, Wang Jun, Peng Hong, The Implementation of Membrane Clustering Algorithm Based on FPGA.
28. Jym Paul Carandang, John Matthew Villaflores, Francis George Cabarle, Henry Adorna, Miguel Ángel Martínez-del-Amor, CuSNP: Spiking Neural P Systems Simulators in CUDA.
29. Raghavan S, Chandrasekaran K, Tools and Simulators for Membrane Computing – A Literature Review.
30. Luis Valencia-Cabrera, Tingfang Wu, Zhiqiang Zhang, Linqiang Pan, Mario J. Pérez-Jiménez, A Software Tool for Computer-Aided Design of Cell-like Spiking Neural P Systems.
31. S. James Immanuel, D.G. Thomas, Robinson Thamburaj, Atulya Nagar, Parallel Contextual Hexagonal Array P Systems.
32. Messikh Azeddine, Youssouf Hamidou Issoufa, Superadiabatic STIRAP: Population Transfer and Quantum Rotation Gates.

33. Saad Alharbi, Ibrahim Venkat, A Genetic Algorithm Based Approach for Solving the Minimum Dominating Set of Queens.
34. Pradeep Isawasan, Ravie Chandren Muniyandi, Ibrahim Venkat, K G Subramanian, Array-rewriting P Systems with Basic Puzzle Grammar Rules and Permitting Features.
35. Rafaa Yahya, Siti Mariyam Shamsuddin, Salah Yahya, Shafatunnur Hasan, Bisan Alsalibi, Ghadah Al-Khafaji, Image Segmentation Using Membrane Computing: A Literature Survey.

The list of participants as well as their email addresses are given below, with the aim of facilitating the further communication and interaction:

1. Akiz Akizur, National University of Malaysia (UKM), Malaysia, akiz.akizur@gmail.com
2. Kelvin C. Buno, University of the Philippines Diliman, Quezon City, Philippines, kcbuno@up.edu.ph
3. Jym Paul Carandang, University of the Philippines Diliman, Quezon City, Philippines, jacarandang@gmail.com
4. Naeimeh Elkhani, National University of Malaysia (UKM), Malaysia, naeimeh.elkhani@gmail.com
5. Ping Guo, Chongqing University, Chongqing, China, guoping@cqu.edu.cn
6. Nestine Hope Hernandez, University of the Philippines Diliman, Quezon City, Philippines, nshernandez@up.edu.ph
7. Bahari Idrus, National University of Malaysia (UKM), Malaysia, bahari@ukm.edu.my
8. Pradeep Isawasan, KDU Penang University College, Malaysia, pradeep.isawasan@gmail.com
9. Richelle Ann Juayong, University of the Philippines Diliman, Quezon City, Philippines, rbjuayong@up.edu.ph
10. M. Nithya Kalyani, Jayaraj Annapackiam College for Women, Theni, India, chandrajac@yahoo.com
11. Philomenal Karoline, Jayaraj Annapackiam College for Women, Theni, India, chandrajac@yahoo.com
12. Xiyu Liu, Shandong Normal University, Ji'nan, China, sdxyliu@163.com
13. Jun Ming, Xihua University, Chengdu, China, mingjun.jun@foxmail.com
14. Ravie Chandren Muniyandi, National University of Malaysia (UKM), Malaysia, ravie@ukm.edu.my
15. Linqiang Pan, Huazhong University of Science and Technology, Wuhan, China, lqpan@mail.hust.edu.cn
16. Hong Peng, Xihua University, Chengdu, China, ph.xhu@foxmail.com
17. Rama Raghavan, National Institute of Technology Karnataka, Mangalore, India, raghavan.sm2005@gmail.com
18. Xiaolong Shi, Huazhong University of Science and Technology, Wuhan, China, shixiaolong@hust.edu.cn

19. Bosheng Song, Huazhong University of Science and Technology, Wuhan, China, boshengsong@hust.edu.cn
20. K. G. Subramanian, Madras Christian College, Chennai, India, kgsmani1948@gmail.com
21. Mohd Izzuddin Mohd Tamrin, International Islamic University Malaysia (UIA), Malaysia, izzuddin@iium.edu.my
22. Jun Wang, Xihua University, Chengdu, China, wj.xhu@foxmail.com
23. Tingfang Wu, Huazhong University of Science and Technology, Wuhan, China, tfwu@hust.edu.cn
24. Wei Xu, Chongqing University, Chongqing, China, xuwei8091@163.com
25. Jie Xue, Shandong Normal University, Ji'nan, China, xiaozhuzhu1113@163.com
26. Salah Yahya, Koya University, Kurdistan, Iraq, salah.ismaeel@koyauniversity.org
27. Lian Ye, Chongqing University, Chongqing, China, ylredleaf@cqu.edu.cn
28. Wenping Yu, Xihua University, Chengdu, China, pean.sl@foxmail.com
29. Zhiqiang Zhang, Huazhong University of Science and Technology, Wuhan, China, zhiqiangzhang@hust.edu.cn
30. Gexiang Zhang, Southwest Jiaotong University, Chengdu, China, zhgxdylan@126.com
31. Yuzhen Zhao, Shandong Normal University, Ji'nan, China, zhaoyuzhen_happy@126.com

Workshop on Membrane Computing at the International Conference on Unconventional Computation and Natural Computation, Manchester, UK, July 11-15 2016

Marian Gheorghe, Savas Konur

School of Electrical Engineering and Computer Science
University of Bradford
Bradford, BD7 1DP, UK

The Workshop on Membrane Computing (WMC), at the International Conference on Unconventional Computation and Natural Computation (UCNC), July 12, 2016, Manchester, UK, is the second event of this type after the one at UCNC 2015 in Auckland, New Zealand. Following the tradition of the 2015 WMC of publishing the Proceedings as a technical report[1], we made the Proceedings available as technical report with the University of Bradford[2].

The Workshop consisted of one invited talk and six contributed presentations (three full papers and three extended abstracts) covering a broad spectrum of topics in Membrane Computing, from computational and complexity theory to formal verification, simulation and applications in robotics. All these papers – see below, but the last extended abstract, are included in the Proceedings.

The invited talk given by Rudolf Freund, "P Systems Working in Set Modes", presented a general overview on basic topics in the theory of Membrane Computing as well as new developments and future research directions in this area.

Radu Nicolescu in "Distributed and Parallel Dynamic Programming Algorithms Modelled on cP Systems" presented an interesting dynamic programming algorithm in a distributed and parallel setting based on P systems enriched with adequate data structure and programming concepts representation. Omar Belingheri, Antonio E. Porreca and Claudio Zandron showed in "P Systems with Hybrid Sets" that P systems with negative multiplicities of objects are less powerful than Turing machines. Artiom Alhazov, Rudolf Freund and Sergiu Ivanov presented in "Extended Spiking Neural P Systems with States" new results regarding the newly introduced topic of spiking neural P systems where states are considered.

[1] http://ucnc15.wordpress.fos.auckland.ac.nz/workshop-on-membrane-computing-wmc-at-the-conference-on-unconventional-computation-natural-computation/

[2] http://hdl.handle.net/10454/8840

"Selection Criteria for Statistical Model Checker", by Mehmet E. Bakir and Mike Stannett, presented some early experiments in selecting adequate statistical model checkers for biological systems modeled with P systems. In "Towards Agent-Based Simulation of Kernel P Systems using FLAME and FLAME GPU", Raluca Lefticaru, Luis F. Macías-Ramos, Ionuţ M. Niculescu, Laurenţiu Mierlă presented some of the advantages of implementing kernel P systems simulations in FLAME. Andrei G. Florea and Cătălin Buiu, in "An Efficient Implementation and Integration of a P Colony Simulator for Swarm Robotics Applications" presented an interesting and efficient implementation based on P colonies for swarms of Kilobot robots.

The Programme Committee members – Erzsébet Csuhaj-Varjú, Alberto Leporati, Radu Nicolescu, Agustín Riscos-Núñez, Mike Stannett, György Vaszil and Gexiang Zhang – have significantly helped with comments and suggestions to the improvement of the contributed papers.

Workshop on Unconventional Computing Systems in commemoration of Yurii Rogozhin Chişinău, Moldova, November 11, 2016

Svetlana Cojocaru, Alexandru Colesnicov, Ludmila Malahov

Institute of Mathematics and Computer Science
5 Academiei str.
Chişinău, MD-2028, Moldova

The2016 edition of the Workshop on Unconventional Computing Systems, in commemoration of Yurii Rogozhin was the third one in this series.

Dr.hab. Yurii Rogozhin (November 11, 1949 † March 10, 2014) was a worldwide known computer scientist with diverse interests, ranged from finding small universal Turing machines to natural computing (e.g., DNA computing by splicing and insertion-deletion, membrane computing).

Yurii had some outstanding results concerning universal Turing machines of a small size. He proved that there are universal Turing machines of the types $UTM(24;2); UTM(10;3); UTM(7;4); UTM(5;5); UTM(4;6); UTM(3;10)$, and $UTM(2;18)$, where by $UTM(m;n)$ we denote the class of universal Turing machines with m states and n symbols. In 2002 Rogozhin and M. Kudlek presented a machine of type $UTM(3;9)$, improving the previous result. These results and results of D. Woods and T. Neary reduce the number of classes $UTM(m;n)$ with an unsettled emptiness problem (i.e., if the class $UTM(m,n)$ is empty) to 41.

A time-varying distributed H system ($TVDH$ system) is a model of biomolecular computing which was introduced by Gh. Păun in 1996 and it has the following feature: at different moments one uses different sets of splicing rules (these sets are called components of the $TVDH$ system) repeatedly. Gh. Păun showed that 7 components are enough in order to generate any recursively enumerable language. In 2004 Yurii and his partners Maurice Margenstern and Serghei Verlan showed that it is possible to construct a $TVDH$ system of degree one which models any type-0 formal grammar. Thus they completely answered the question of constructing $TVDH$ systems of smallest degree which generate any RE language using the parallel nature of molecular computations based on splicing operations.

In the area of membrane computing Yurii Rogozhin has (co)-authored over 50 publications. The first to be mentioned is a series of papers investigating minimal symport/antiport, in particular establishing the ultimate results for the computational completeness with two membranes, proved by very creative and highly non-trivial constructions. He also participated in research of transitional P systems,

as well as of P systems with active membranes. Other topics of Yurii's systematic interest include using the framework of P systems as an additional control mechanism for such operations as insertion/deletion (including exo-operations) and splicing. In particular, he constructed the smallest known universal P system – with only five rules, the proof displaying how to perform multiple tasks by the same rule.

It was Yurii's idea to consider a hybrid computational model combining quantum subsystems and membrane subsystems. Other research directions with practical motivations in its scope are polymorphism, dictionary operations, generating inflections, parsing derivatives and annotating affixes of a natural language.

Since the death of Yurii Rogozhin, Institute of Mathematics and Computer Science in Chişinău, Moldova, holds these commemorative Workshops at a suitable date near Yurii's birthday.

The 2016 Workshop consisted of one invited talk and five contributed presentations. Detailed abstracts of works can be seen on the Institute Web site.[1]

Artiom Alhazov, Omar Belingheri, Rudolf Freund, Sergiu Ivanov, Antonio E. Porreca, and Claudio Zandron in "Semilinear Sets, Register Machines, and Integer Vector Addition (P) Systems" considered P systems working with multisets with integer multiplicities focusing on a model in which rule applicability is not influenced by the contents of the membrane. Authors show that this variant is closely related to blind register machines and integer vector addition systems. Furthermore, they described the computational power of these models.

The next two papers were dedicated to network control theory. **Vladimir Rogojin** in "NetControl4BioMed - Automatic discovery of combined drug therapy" used the computational analysis of the structure of intracellular molecular interaction networks, they being a formal representation of relations between numerous components within cells. Network controllability studies focus on discovering combinations of external interventions that can drive the biological system to a desired configuration. In practice, these studies can be translated into finding a combined multi-drug therapy in order to achieve a desired response from a cell. The author developed a pipeline that finds a minimal set of nodes controlling a given set of targets within a network. The pipeline highlights those control nodes for which there are known FDA approved drugs. The network is generated automatically through querying of a number of pathway databases.

Eugen Czeizler in "How graphs help us fight cancer: structural control of disease networks" applied recent research in the area of network science and network control theory to the analysis of the structure of intra-cellular molecular interaction that can suggest novel therapeutic approaches for systemic diseases like cancer. It was proved that the structural target controllability problem is NP-hard. Search algorithms were improved using several heuristics. The application in the analysis of three types of cancers was demonstrated.

Serghei Verlan and Sergiu Ivanov considered one special class of systems in "Universality of Graph-controlled Leftist Insertion-deletion Systems with

[1] `http://www.math.md/news/2016/12246/`

Two States". They permitted regular expressions as rule contexts and proved that in this case the computational completeness is achieved when additional control mechanisms are used. Then authors investigated simulation of rules with regular contexts by conventional rules of small sizes, and provided a construction that guarantees universality of such systems.

In "Small Asynchronous P Systems with Inhibitors Defining Non-semilinear Sets" **Artiom Alhazov** described 54 systems that are small in different ways like the alphabet size, the number of rules, etc.

The invited talk "Logic in Computer Science, Engineering and Industry" was presented by **Yuri Gurevich** from Microsoft Research. He noted that in software industry, engineers do formal logic daily even though they may not realize that, but, instead of studying logic, they spent a lot of time studying calculus which they use rarely, if ever. Dr. Gurevich illustrated why logic is so relevant and why it is hard for software engineers to pick it up.

Miscellanea

Some Wonders of a Bio-Computer-Scientist*

Gheorghe Păun

Romanian Academy, Bucharest, Romania
gpaun@us.es, curteadelaarges@gmail.com

1. The previous title takes some precautions, which are visible, but still I want to stress them.

First, the autobiographical character. The last two decades – to be precise, the last twenty two years – I was totally dedicated to bio-computability, one can even say that I was *confiscated* by this research area, so fascinating, promising, endless in possibilities of theoretical developments and of applications. The beginning is placed in the spring of 1994, when I read a paper by Tom Head, a wise American who later became a friend and a collaborator of mine, who, already in 1987, proposed a formal language theory model for the recombination operation of ADN under the influence of restriction enzymes and ligase. He has named it the *splicing* operation. I will shortly describe it later. I remember it exactly – I was in Austria, in Graz, participating in a conference. On the one hand, I became immediately enthusiastic about this idea – after two decades of research in formal language theory, I was subconsciously looking for areas to apply this theory, on the other hand, I was somewhat unsatisfied, because Head's formalization was complex, it remained, to my taste, too close to the biological reality. Right then, in the hotel in Graz, I have imagined a simpler definition, closer to the style I was used, I can call it *Salomaa-Marcus style*. Four years after that I have worked in this area. Until 1998, when I had a second thrill, after defining a model inspired from the architecture and the functioning of the living cell, the first one from a families of models which continues to grow. After that, I have totally identified myself with *membrane computing*, a research area which transformed me in a letter and which will be the territory through which I will invite the reader in what follows. (I will sometimes use the abbreviation MC for *membrane computing*.)

* This is the English version of a lecture delivered on December 8, 2016, on the occasion of receiving a Doctor Honoris Causa title from the West University of Timişoara, Romania.

I return to the title. Wonders-questions, not models-results-applications, although the former are connected to the latter. Places where the mathematician who I am by education and the computer scientist with twenty years of theoretical research – I have graduated mathematics in 1974 – remains still, with raised elbows, either not understanding, or expecting something different, or faced with objects coming from biology which, seen through the glasses of the theoretical computer scientist, suggest computability ideas-models totally new for theoretical computer science and, much more, for practical computer science, in the forms we know them from books and from implementations. During billions of years of evolution (I adhere here to the evolutionist paradigm, this is not a suitable place for an evolution version creation discussion, furthermore, for what follows *we do not need any creationistic hypothesis*, to quote a phrase assigned to Laplace, but also to other big names of physics), life has evolved many processes and supports for these processes of a high precision, remarkably subtly and efficient, which can be transferred, at least at the theoretic level, to computer science. How to do that, asks himself the mathematician-computer scientist? With what practical consequences, asks himself the computer scientist interested in applications? Why so many computability aspects, as they were developed up to now, look *non-natural*, far from the "real reality", from the biological one?

Some of these questions get close to the science fiction (SF) frontier of science, but I will not avoid them, as long as the science, physics in particular, does not totally refute them as non plausible. We are placed here under the flag of the possible, not of the probable...

There will also appear questions-wonders which are more precise, more localized. Only part of those which I have reached, significantly fewer than those which were possibly met by other colleagues of bio-computing. That is why the title starts with *some*...

A few remarks about the style. Although I will refer to many notions, from biology and from computer science, although in the end I will provide a pretty comprehensive bibliography (also this one will be mainly autobiographical), the discussion will be informal, without precise references.

In what concerns the bibliography, one of the items is my Reception Discourse in the Romanian Academy, delivered on October 24, 2014, and having the privilege to be followed by a response by Solomon Marcus, the Professor, always with capital initial letter. The present pages can be seen as a continuation of the Discourse, at a more informal-colloquial level, but also as an opportunity to pay my homage to my Professor, who constantly used to promote the *state of wondering, of questioning yourself*. Sadly, the Professor passed away in March 2016.

Above all these, this text is an invitation to the reader to wonder together...

2. Before mentioning any reason to wonder-question ourselves, it could be useful to roughly specify the framework: what/how much biology, what/how much computability are necessary for what follows?

Mainly cellular biology, at a general level: membranes which enclose compartments, "protected reactors" where specific biochemical reactions take place, in-

volving "reactants" (from ions to large macromolecules) which swim in an aqueous solution or they are bound on membranes or on the cytoskeleton, then protein channels which make possible the communication among compartments. Many other ingredients can be added, the biologists know a lot of things about the cell. The monograph *Molecular Biology of the Cell*, by Alberts, Johnson, Lewis, Raff, Roberts, Walter, has over 1500 pages of a large format. A true universe – of a nanometrical scale. The smallest entity about whose alive character there is no doubt (about viruses the opinions are mixed). A complex "factory", at the same time robust and fragile, very precisely and efficiently organized.

Already at this level we get an important question, but one mainly of a biological type: what is making the difference between *alive* and *non-alive*? Chemistry and physics cannot provide a convincing answer. The borderline seems to be defined in terms of organization, information, in terms of *something* which we cannot yet perceive, understand, and model. We may call it *soul*, or *vital breath*, just for moving the question away.

An important detail: the today biology, chemistry, physics are much developed in the study of their object of study in terms of substances and energy, but not also in terms of information. I do not enter into details, this opens the gate to many questions, starting with the definition itself of information, organization, complexity. Can we conceive information without any support, of a substance or energy type, the other two components of what we use to call *matter*? Two more remarks: (1) biology and physics seems to be on the verge of passing to "a new age" – for biology there were proposed some up-dated names, such as *infobiology* and *infobiotics*, and (2) computer science is a part of information theory, in the broad sense, therefore, the computational, algorithmic approach seems to be one of the ways through which the biology and the physics will evolve in the near future. Quantum computing seems to be one of the first steps for physics (Gruska), biocomputability seems to be half of the path for biology.

Let us return to the cell, with a slogan-"equation" launched by Solomon Marcus during one of the first MC meetings, in Curtea de Argeş, Romania, 2001:

Life = DNA software + membrane hardware.

It is stressed here the role played by membranes in the life of cells, thus also motivating the name which I gave to this research area, *membrane computing*; maybe a better name were *cellular computing*.

After the individual cell, I will also incidentally invoke populations of cells, tissues, organs, colonies of bacteria, in the end, also the "thinking cell", the neuron.

Details, in the next sections – here, the computer science basic prerequisites. First of all, Turing computability. The standard framework for most computability research. By Turing-Church Thesis, the highest level of algorithmic computability. At only 24 years, the genial Turing answered Hilbert's question on *what can be mechanically computed*, with a particular application to the decidability in arithmetics, by introducing what is now known under the name of *Turing machine*, the most general and the most convincing (for the contemporary researchers, Church, Kleene, Gödel etc., who tried in various ways to answer Hilbert's question) of the

notion of an algorithm. Then, by the definition of the universal Turing machine (a machine able to simulate any particular Turing machine as soon as a code of the particular machine, together with an input for it, are placed as an input on the tape of the universal Turing machine), and by the theorem proving the existence of the universal Turing machine, Turing was signing the birth certificate of the computers as we have them today: when designing and constructing the first (programmable, as the existence of a universal Turing machine made possible) computers, at the beginning of forties, last century, John von Neumann was explicitly influenced by Turing ideas. From here, the name of Turing-von Neumann computers, from here their sequential character and their theoretical vulnerability to computer viruses (programs and data are placed in the memory, with the possibility of affecting programs by other programs in the same way as data are affected).

At the maximal level, the Turing machine, at the lowest level of computability, the finite automaton, a form of a Turing machine restricted in an extreme extent. These are the two poles of computability to which the bio-inspired computability is constantly referred to.

These are the two poles of the computability *competence*. From a practical point of view, at least equally important is the *performance*, the efficiency. In this way, it enters into the stage the famous distinction between the class of problems which can be solved in polynomial time (with respect to the size of the input) and the class of problems for which no polynomial algorithm is known, but, if somebody proposes a solution (if a solution can be "guessed"), we can check it in polynomial time. The famous complexity classes **P** and **NP**, the (in)famous problem whether or not $\mathbf{P} = \mathbf{NP}$, the first one in the list of "millennium problems" for whose solution the Clay Institute from USA offers one billion dollars.

I will also invoke grammars (especially Chomsky grammars); further details will be mentioned when they will be useful.

3. Let us start with DNA, chronologically the first investigated and a relevant case study.

Today we are learning the language in which God created life, historically claimed Bill Clinton in the summer of 2000, when the reading of the human genome was completed. A language, indeed. Syntax, in the proper meaning of the term. Four "letters", A (adenine), C (cytosine), G (guanine), and T (thymine), placed along two strands, with well specified pairs face to face, the so-called *Watson-Crick complementary pairs* (Watson and Crick have received the Nobel Prize, in 1962, for the discovery of the DNA structure, at the beginning of fifties): A is always in front of T, and C always forms a pair with G. This is the primary structure of DNA, the only one of interest here. The secondary structure, the double helix, and the ternary one, the complex folding, are not relevant for what it follows.

If we raise the temperature of the solution where the DNA molecules are placed, the two strands are separated. If we decrease back the temperature, then the nucleotides look for their Watson-Crick complementary pairs and the double stranded structure is rebuilt. Then, the *restriction enzymes* – somebody called them the most intelligent tools nature provided to the genetic engineers. They are proteins

which recognize a short sequence of nucleotides, a pattern, a *context*, to come closer to linguistics, and cut the DNA molecules, in most cases in the middle of this pattern, in most cases, producing *sticky ends*, staggered strands: one of them is cut in a point, the other one some nucleotides away; the single strands of nucleotides are sticky, if they find their complements, then they get glued to them, completing the double strand.

A lot of further biochemistry can be found around (for instance, ligase, other enzymes, which help the reconstruction of the double strand, or the directionality of the two strands, whose ends are marked with $3'$ and $5'$, the biochemists know why, opposite on the two strands and preserved by the strands after getting separated), and still more computability. Also, similarly much linguistics – it was underlined by Solomon Marcus already in 1974, in a paper which appeared too early, so that it did not receive the audience it deserved.

Now, one first example – and the first surprise.

A result in formal language theory, published already in 1980 by Engelfriet and Rozenberg, says that any language which can be recognized by a Turing machine (equivalently: which can be generated by a Chomsky grammar; in the usual terminology: *recursively enumerable*), hence of the most general type among the computable languages, can be obtained starting from a certain fixed language, that is, independent of the starting language, applying to it a sequential transducer, the simplest type of a transducer, a finite automaton with output (one reads the input string, from left to right, without returns, and for each read symbol one outputs one or more symbols, also depending on a state from a finite set of states). The unique language from which we start, the one in which there are "hidden" all computable languages, is obtained as follows: consider the symbols 0, 1, consider their "complements" $0'$ and $1'$; take a string over the symbols 0, 1 and consider its *twin*, the string obtained by priming all symbols; mix the symbols of the string with the symbols of its twin, in all possible ways, preserving the ordering of symbols in each string – this operation is called *shuffle*. Finally, collect all strings obtained by shuffling all strings over 0, 1 with their twin strings. This is the language we are looking for – called the *twin-shuffle language* over 0, 1.

Four letters, $0, 1, 0', 1'$. Maybe this was the coincidence which suggested to Rozenberg and Salomaa the following operations: take a DNA molecule and "read" it, from left to right, on each strand, with random speeds on the two strands, writing 0 if we met A or T on the upper strand, $0'$ if we met them on the lower strand, 1 if we met C or G on the upper strand, and $1'$ if we met them on the lower strand. Doing this for all possible DNA molecules, we get... exactly the twin-shuffle language over 0, 1! According to Engelfriet and Rozenberg theorem, if we also apply a sequential transducer, then we obtain any computable language (and *all* of them, by varying the transducer). Otherwise stated, everything which can be computed is "codified" in the DNA molecule, what remains to do is to read the molecules as sketched above and then to rewrite these readings with a sequential transducer. A completely unexpected conclusion, difficult to be imagined (without having at hand the theoretical result from 1980).

In a DNA computing book written together with Rozenberg and Salomaa, published by Springer-Verlag in 1998, there are two results which complete the previous observation: on the one hand, we can consider only three nucleotide-symbols, $0, 0', 1$, with the last one being its own "complement" (nature uses to be redundant, nature also loves symmetries), and, on the other hand, we can also take into account the opposite orientation of the two strands, that is, we can read one strand in one direction and the second one in the opposite direction. In both cases, the result is the same, the characterization of the computing power of the Turing machine.

Later, Vincenzo Manca proved that, in a certain precise sense, the double stranded structure of the DNA molecule is *necessary* in order to obtain the computational universality.

We already have here a beautiful wonder – and a comforting example of using an old "purely theoretical" result in a new framework, much closer to the reality.

4. Let us pass now to the splicing operation introduced by Tom Head (in 1987, hence, at the theoretical level, DNA computing started several years before the Adleman experiment, from 1994). Recombination, cut-and-paste, with a complete formal, syntactic formulation. If two restriction enzymes, each one with its own pattern, cut two molecules in such a way that the sticky ends they produce are identical, then the obtained fragments can be recombined, the prefix of the one molecule with the suffix of the other molecule. In this way, we pass from the two starting molecules to two new molecules.

The definition can be simplified, without losing too much from the biochemical significance. First, the Watson-Crick complementarity allows for passing from the double stranded molecules of symbols to usual strings. In this framework, each restriction enzyme recognizes a substring and cuts somewhere in its middle, in a specified position. Two enzymes which produce identical sticky ends lead to a *splicing rule*, which can be written as a quadruple of strings, with the first pair of strings indicating the pattern recognized by the first enzyme and the place where it cuts the string, and analogously the second pair of strings for the second enzyme.

This simplified version of the splicing operation was much investigated in DNA computing. In theory, as the starting point of defining a computing model called, in a paper written together with Rozenberg and Salomaa, *H system*, as a homage to Tom Head. In short, one gives a finite set of string-axioms and a set of splicing rules, one applies the rules to the axioms, then to the resulting strings, and so on, iteratively, possibly also selecting only certain strings, and in this way we generate a language. Like in a Chomsky grammar.

Two are the basic results of this area – and, between them the wonder!

On the one hand, we get a characterization of regular languages, those recognized by finite automata (the result got several proofs, the initial ones pretty complex), on the other hand, a characterization of the computing power of Turing machines (this result belongs to me and the proof, somewhat surprisingly, is simpler than the proofs of the first result mentioned above). The difference is that in

the first result we use a finite set of splicing rules, in the second one we use an infinite set, but the set of rules is represented by a regular language. The universality can be obtained also for a finite set of rules, but with the use of rules controlled in various ways inspired by biology or by the formal language theory: promoters, inhibitors, using a priority relation, imposing a dependence on the sequence of rules used, and so on.

The technical details are not relevant, important is that we compute either at the level of one pole of computability, or at the level of the other pole. Everything or nothing, without equalizing any intermediate level – and there are several such levels, defined in terms of automata or grammars. It is worth noting that also in other frameworks, for instance, in MC, we obtain similar results. May we infer from these results that the intermediate levels are not *natural*? In a certain sense, this is the case, because, if we think to the classes of automata which characterize the context-free and the context-sensitive languages in Chomsky hierarchy, the push-down and the linear bounded automata, respectively, the former have a rather ad-hoc definition, the latter are inspired by complexity theory, the family of languages is defined by taking into consideration a property observed during the computation, not with respect to a static property, belonging to the automata architecture.

5. The discussion can be generalized: what means to compute *in a natural way*? Which data structures to use and which operations on them? Theoretical computer science deals mainly with processing strings of symbols, especially using the *rewriting* operation, replacing a substring, short in comparison with the whole string which it is processed, with another string, also short. On the tape of automata of all types, from the Turing machine to the finite automaton, it is written a string of symbols. The Chomsky grammars, Lindenmayer systems, Marcus contextual grammars, Post systems, Markov algorithms, all these are processing strings. The same for Thue iterated morphisms, actually equivalent with Lindenmayer deterministic systems.

In comparison, what we find in biology? In the case of ADN, the double stranded molecule, and, as operations, the recombination, Tom Head's splicing, the separation of strands and their annealing, only rarely the point mutations, at the level of single nucleotides. Then, in the compartments of a cell, the *multi-set*, the set with multiplicities associated with its elements. In biochemistry, like in chemistry, *the numbers matter*. While the DNA molecule can be considered a string, hence on the DNA string we have positional information, like in the case of numbers written in the Arabic style, in the aqueous solution from the cell compartments the numbers are expressed in unary. Base one is exponentially less efficient than base two – and still this was chosen by nature/biology. A biochemical reaction can be interpreted as a rewriting operation of a multiset: a sub-multiset is replaced by another multiset. However, in the functioning of a cell there appear many other operations, such that those of symport and antiport (I will return to them, as they deserve a more detailed discussion), or operations by which the membranes themselves evolve (division, dissolution, creation of membranes, exo- and endo-cytosis, and so on). Similarly in what concerns the functioning of neurons

and of neural nets. I am careful not to say "the functioning of the brain", because, in spite of the fact that the neurologists know "everything" about the physiology of the brain, nobody can explain how the brain becomes *mind*, the organ where thinking, sensing, sentiments, conscience is (supposed to be) placed. Hypotheses, proposals, speculations there exist, significantly less certainty...

With all these data supports and operations with them one can define computing models, most of them equivalent in power with Turing machines, hence computationally complete, computing everything which can be computed (we take as valid the Turing-Church thesis). The proofs are often, if not always, constructive: we start from a Turing machine or from an equivalent model (normal forms for Chomsky grammars, classes of regulated grammars with context-free rules, register machines, other devices) and we construct an equivalent H system or, in MC, an equivalent P system. Starting the construction from a universal Turing machine (or an equivalent model), the biologically inspired computing model will implicitly be universal, hence programmable.

In theory, everything is wonderful. *In info.* At the practical level, there however appear two problems: the effective implementation, *in vitro*, of the theoretical model, and the question whether such a bio-computer can have also further motivations than proving that *it is possible* to compute in this way. Is such a computer useful, better than an existing electronic computer, at least from one point of view? Possible points of view are many. The possibility to solve problems of **NP** (or higher) complexity in a polynomial time is the first and the most attractive goal, but there are other ones related to the energy efficiency, the possibility to evolve in time, to learn, to self-repair, to deal with imprecise or incomplete data. All these are dreams for computer science, difficult to achieve, but they are common realities in the life of a cell, in biology in general.

Let us consider only one aspect, the massive parallelism, met ubiquitously in biology. The electronic engineers, even if they cannot equalize the biological parallelism, could put together a large number of processors, but, on the one hand, there would appear problems related to the heating of the computer, on the other hand, problems related to the synchronization, the coordination of the processors. It enters the stage the so-called *communication complexity*: the number of bits needed for coordinating the processors becomes comparable with the number of bits used in the computation itself. How the nature has solved these two problems we do not know precisely. Still less we know about how to imitate the nature's solutions.

6. Let us return to the *symport* and *antiport* operations. The membranes which provide the internal structure of the cell are constituted of so-called phospholipid molecules, which have a polarized "head" and a non-polarized "tail", consisting of two fatty acids, hydrophobic. Thrown in water, such molecules self organize spontaneously, forming a sphere with two layers, keeping the fatty acids separated from the water, opposing to the internal and the external water molecules the polarized heads. The common "enemy" and the electrical charges keep the molecules

together, in a structure called *fluid-mosaic*, a model proposed by Singer and Nicolson, fully accepted only in 1972. The phospholipid molecules can move with respect to each other, but they do not allow to large molecules to pass between them from a side to the other side of the membrane; neither the ions can pass, because of their electrical charges. Nature has solved the problem of communication between the interior and the exterior of a vesicle of this type in a rather ingenious way: between the phospholipid molecules there are intercalated proteins which function as transmembrane channels – important to stress, of a *selective* type. Channels for water (the aquaporins of Gheorghe Benga, for which a Nobel Prize was awarded in 2003, but not to him...), natrium-potassium and natrium-calcium pumps, and so on and so forth.

Two special types of such channels are those which support the processes which the biologists call *symport* and *antiport*. In the first case, two molecules, let us identify them by M_1 and M_2, cannot pass separately through the symporter channel, but together they can do it, either entering the inner "reactor", or exiting it. In the second case, M_1 and M_2, placed in separated regions, one inside and the other one outside, cannot separately pass through the protein channel, but together they can do it, one in a direction and the other one in the opposite direction. The protein opens the channel until the molecules pass and then the channel is closed again.

In this manner, we obtain a way to "communicate" among the compartments of a cell – with the mentioning that I call *communication* the passage of objects (molecules, of any type) across a membrane. Starting from multisets of objects placed in the compartments of a cell and using given symport and antiport rules (pairs (M_1, M_2) associated with membranes, with specified directions to move each molecule), we can compute: we iteratively apply the rules, until reaching a configuration where no rule can be applied. (Because the number of objects cannot be modified by symport and antiport rules, we assume that the environment participates in the computation, as an exhaustible source of symbols.) Surprising at the first sight, but not so much at the second sight, we again obtain a characterization of the power of Turing machine.

Universal computation, by communication – here lies the surprise! We compute not by rewriting strings or multisets, but by moving objects across borders defined by means of membranes. Does it makes sense (to try) to construct a computer based on the symport and antiport operations? No, in what it concerns the computing power (Turing-Church Thesis), maybe, from the point of view of efficiency (provided that we will succeed to implement a significant level of parallelism), yes, from a point of view which is not so much taken into consideration by the current technology, although it refers to an important detail: the consumption of energy, heat dissipation. In computer science it is said that the dissipation of heat appears because of erasing. Rewriting means first erasing and after that writing. The operations of symport and antiport do not assume erasing, we only have moving, changing the place. Will these operations diminish/eliminate the energy loss during a computation? This is an appealing hypothesis – but the biologists warn us that, in order to function, many protein channels need chemical energy

(obtained with the help of ATP, adenosine triphosphate, "the energy accumulator of the cell").

Why, at a closer examination, the universality of computing by means of symport/antiport is not so surprising? Theoretical computer science, old results in formal language theory, help us again. There are several characterizations of recursively enumerable languages of the following type: every language which can be recognized by a Turing machine can be obtained starting from a context-sensitive language (recognized by a linearly bounded automaton) and applying to it an operation which remove certain symbols (morphisms, left or right quotients, etc.). In short, context sensitiveness and erasing. Two features which the symport and antiport operations possess: the fact that two molecules evolve simultaneously means context sensitivity, while by "throwing" molecules in the environment or by collecting them in a corner of the cell and ignoring them there, we get the erasing. The only difficulty, for the theoretician, is the proof, the simulation of a computing mechanism equivalent with a Turing machine by means of certain symport and antiport operations associated with the compartments of a given arrangement of membranes. There are many proofs of this type in MC – that is, *in info*, but no *in vivo* or *in vitro* implementation.

7. Remaining close to biology, let us have a look to the way the neurons communicate to each other by means of electrical impulses of an identical shape, the *spikes*. Like any cell, a neuron has a body (soma), from which a "wire" starts, the axon, along which the spikes circulate. On the soma and in the end of the axon there are filaments, by the contact of which one obtains synapses, the links between neurons. We ignore here a lot of architecture and functioning details (some of them were already captured in MC models): the axon is covered by a protecting myelin layer, it is segmented by the so-called Ranvier nodes, a sort of relays amplifying the electrical impulses, the synapses correspond to symport/antiport operations, there also exists another class of cells, the astrocytes, which control/help the neurons activity, depending on the flow of spikes along their axons, and so on. We only keep in mind that we have only one type of "objects", the spike, and that the flow of spikes, their frequency, the distance in time between two consecutive spikes are very important. Of course, the functioning of a neuron, the emission of spikes depends on the neuron contents.

Abstracting all these details, one can define a computing model of the following type. Neurons are placed in the nodes of a graph whose arcs represent the axons/synapses. We start with a given number of spikes placed in each neuron. The neurons also contain *spiking rules*: depending on its contents, a neuron consumes a number of spikes and produces a number of spikes, which are sent to all neurons reached by a synapse which starts in the neuron where the rule was used. One obtains what is called in MC a *spiking neural P system*, in short, an SN P system. In the ten years since these devices were introduced, about 300 papers investigating them were published, from theory (computing power and efficiency) to applications (even in technological decision making, for instance, in a combina-

tion of these models with fuzzy reasoning models – the topic is mainly explored in China).

Like in many other places in bio-computability, also in this framework one obtains either a characterization of the power of finite automata, or of the power of Turing machines. Again, the intermediate levels of computability are not "natural".

Beyond the previous observation, two further reasons of wonder appear in this context.

The first one refers to the way the information is encoded/represented by means of spikes, in the distance between two consecutive electrical impulses. The time as a support of information... Similar to the Morse alphabet, but the goal there is only the transmission of information, here we deal with computations, even at the maximal level, that of Turing machines.

Can we use this observation for constructing computers? If data are encoded in time, using intervals, how will then function the classical time-space trade-off in this framework? Polynomial solutions to **NP**-complete problems are obtained, in theory and also in the DNA computing experiments and in MC theoretical approaches, by using an exponential workspace; in the spiking neurons case, the space and the time are "superposed", the time efficiency seems to be lost.

The second wonder reason is related to a result which looks unexpected at the first sight, namely, that there exist universal (in the universal Turing machine sense) SN P systems which have a relatively small number of neurons, at most of the order of hundreds. The result depends essentially on the type of spiking rules used in neurons and on the number of rules present in each neuron. In short, the number of neurons depends on their complexity, which is rather natural, but the fact that we can obtain a "programmable universal computer" with only about one hundred neurons is unexpected, at least with respect to the billions of neurons in the human brain and even with respect to the algorithmic computability. Clearly, the brain does not has as its goal (only) to compute at the algorithmic level (using the brain as a model for computers and using the computer metaphor in the study of the brain are useful strategies, but with soon to reach limits), but equalizing so easy the computability upper limit is not similarly easy to explain. Are the "neurons" of our models too complex, too powerful, or, on the other hand, the Turing computability is not too comprehensive? It is highly probably that both hypotheses are true. The second assumption is supported also by other characterizations of Turing computability in bio-inspired frameworks, using models which are simple "syntactically" or of small dimensions. The first hypothesis raises the question how to simplify the neurons in such a way not to lose the universality, even if this result is obtained by using a larger number of neurons. There are research efforts in MC following this idea.

8. While it is so easy, in DNA and cellular computing, to reach the level of algorithmic/Turing computability, it is much more difficult to pass beyond "the Turing barrier". This goal pertains to a branch of computability theory called *hypercomputability*, with publications, conferences, dedicated authors and also with skeptical or even acid comments. On the one hand, it is said/hoped that passing

beyond "the Turing barrier" would have much more important practical consequences than a possible proof, even a constructive one, of the equality **P = NP** (a result with a rather low plausibility). On the other hand, up to now, the ideas on which the models able of hypercomutability are based are not too many and not too realistic. In the domain literature there are about one dozen such ideas, but Martin Davis considers all of them *tricks*, supporting a *myth*.

Actually, Turing himself has proposed a version of his machine able of computing more than the Turing machine, by adding an *oracle*, which can answer (for free) questions, maybe questions which pass beyond the competence of the Turing machines. Several other ideas are based on squeezing infinity or real numbers in the construction or in the functioning of the machine. Because real numbers "are more than computable numbers" (more precisely: the set of computable numbers has a denumerable cardinality, while the cardinality of the set of real numbers is strictly larger), something uncomputable is introduced in advance in the model, hence there is no surprise to get a model which is more powerful than a usual Turing machine.

The same with the introduction of the infinity, even a denumerable one, with the following remark of interest for our discussion. Adleman experiment, of solving the problem whether a Hamiltonian path there exists in a graph, proceeds along the following main steps: one first generates all paths in the graph, then one eliminates all paths which do not pass through a number of nodes equal with the number of nodes in the graph (a selection according to the length of DNA molecules), then one removes the paths which are not passing through node 1, node 2, etc. What remains in the end are the DNA molecules which encode Hamiltonian paths. If no molecule survives, then this means that no Hamiltonian path exists. Also in other experiments one proceeds in a similar manner, that is, one starts with a large set, easy to be generated, and one removes iteratively elements which cannot be solutions. Otherwise stated, it is not constructed a solution, but one eliminates non-solutions. "Sculpturing!" Computing by carving! In terms of languages, we start from a general set of strings, maybe the total language over a given alphabet, and we eliminate repeatedly strings. *We remove the complement of the language we want to identify.* Not generating a language, as the grammars are doing, not accepting, as the automata are doing, but rejecting the strings which we do not want.

The family of Turing computable languages is not closed under the operation of taking the complement. Therefore, by "carving" we can "compute" languages which are not Turing computable. To this aim we need to eliminate an infinite set of strings. If in each step we remove a finite set of strings or even an infinite but regular one, in a finite number of steps we do not get out of the family of regular languages. Thus, either there is a step when we remove a complex language, or the "computation" lasts an infinite number of steps. Hypercomputability, but, agreeing with Martin Davis, the procedure does not look as an implementable (hyper)algorithm...

It is worth noting that the physicists are not refuting as totally infeasible the hypotheses on which the hypercomputability is grounded. Here is an SF scenario from which the physicists are not removing the S: let us assume that time has a 2-dimensional description, in spite of the fact that we, the humans, perceive only one dimension of it; let us assume that a human uses a computer which, after performing a number of steps along the dimension of the time which can be sensed by the user, starts to work on the orthogonal direction of time, performing an arbitrary number of steps, all of them invisible to the user, and then comes back and continues on the human visible dimension. Irrespective how many steps were made on the orthogonal direction of time, the user receives the result in a time which is essentially shorter for him.

We departed from biology, let us return to the cell. Nature creates membranes with two main goals: to localize the reactions and the reactants (enclosing them in "protected reactors"), and for creating reactors of a small size, where the reactants are sufficiently close to each other in order to collide and react, helped by the Brownian movement. Thus, smaller means faster. Let us generalize, exaggerating "a little bit", and assume that in a membrane placed inside another membrane the reactions are twice faster than the reactions in the membrane above. Let us create membranes inside membranes, repeatedly. (In MC there are rules of this type, for increasing the membrane architecture.) The reactions are accelerated from top down in an exponential way. This corresponds to the *accelerated automata*, already investigated in computer science since many years: the first step of a computation is performed in one time unit, the second one in half of a time unit, each step which follows in a time which is half of the time the previous step was performed. In this way, in two time units marked on a clock external with respect to the automaton, one performs an infinity of steps of the computation. The whole computation ends in at most two external time units. Accelerated automata of this type can solve problems beyond the power of Turing machines (such as, the halting problem, provided already by Turing as an example of a problem without an algorithmic solution). Exactly this can be done also by an accelerated P system – the result appears in a paper written together with Cristian Calude, now in New Zealand. Biological motivation, hypercomputability, but Martin Davis is again right: the membrane hierarchy should be arbitrarily deep in order to achieve an arbitrarily powerful acceleration...

9. Let us quit the hypercomputability and pass to... *fypercomputability*! This term occurred, as a game with letters-words, replacing the front *h* of hypercomputability with *f*, from *fast*. This is the main expectation from bio-computability, because it is one of the main "barriers" facing the "Turing-von Neumann" computers: the impossibility to solve **NP**-hard problems in a feasible (polynomial) time, not to speak also about **NP**-complete problems (a problem is **NP**-complete if each **NP**-hard problem can be reduced to it in a polynomial time; therefore, if an **NP**-complete problem could be solved in polynomial time, then all **NP**-hard problems would be solved in polynomial time, that is, $\mathbf{P} = \mathbf{NP}$).

However, the exponentials are not at all easy to handle... Here is a simple example, but with a strong didactic impact. Let us assume that we have a problem, for instance, dealing with graphs, of an exponential complexity, say, of the order of 3^n; let us assume that the existing computers can solve the problem in a reasonable time for graphs with 100 nodes. Assume now that the technology promises us or even offers us new computers which are 1000 times faster than the current ones. A spectacular progress, not at all easy to be obtained. A problem which is now solved in 1000 seconds will be solved with the new technology in one second. Good, but if now we can deal with graphs of 100 nodes, which will be the size of graphs which can be handled with the improved computers? The answer is disappointing – graphs with only 106–107 nodes – for the simple reason that 3^7 is greater than 1000. A great technological progress leads to a derisory advance in what concerns the size of the problems which can be solved. Not the technology is the way to cope with the exponential complexity, we need to look for something else, basically new. The parallelism is a path towards fypercomputability, and there are two main strategies: starting from the beginning of the computation with an exponential space (as in the Adleman experiment), or creating such an exponential space during the computation.

In MC, in most cases one follows the second strategy: using biological operations, such as the membrane division or the string replication, one creates an exponential workspace in a linear time, then, using this workspace, one solves in a polynomial time **NP**-complete problems. However, an exponential number of objects can also be obtained without involving the membrane division, but using only rules of the form $a \to aa$, repeatedly, in a parallel manner. Starting from one copy of a, in n steps we get 2^n copies of a. It is rather interesting to see that this manner of obtaining an exponential workspace is not sufficient in order to reach the desired efficiency: the so-called *Milano Theorem*, proved by Claudio Zandron in his PhD thesis (the first European PhD thesis in MC, presented in 2001; the first one in the world was presented one year before, in India), says that any P system, even using rules of the form of $a \to aa$, can be simulated in polynomial time by a Turing machine. Therefore, if such a system would solve **NP**-complete problems in polynomial time, then also the Turing machine would do it, which will imply that $\mathbf{P} = \mathbf{NP}$. Consequently (unless $\mathbf{P} = \mathbf{NP}$), the membrane division is necessary for fypercomputability.

Again, we have encountered here a subtle and surprising aspect. An exponential number of objects, placed in a single membrane, are not sufficient for exponentially speeding up the computation (in P systems), but if we separate these objects in different membranes, also exponentially many, the fypercomputation jump is obtained. The difference is made by the localization, the use of different rules in different membranes. In this way, there appears a parallelism of a more complex nature, with compartments which evolve in parallel, each of them processing, also in parallel, its own objects. I do not know whether this difference/phenomenon has appeared also in other frameworks – anyway not in the classic complexity theory,

where the parallelism is not present, because the working model is the Turing machine, a sequential model.

10. Actually, the computational complexity theory, in its classic form, has several "gaps" from the point of view of bio-computability. I return to the way Adleman has solved the Hamiltonian path problem: he started from a given graph, hence from an instance of the problem, has constructed an ad-hoc "computer", using DNA molecules directly depending on the chosen graph, test tubes and other lab instruments, and he has found the solution (in a linear time, as the number of biochemical operations, some of them of a massive parallelism). In the complexity theory, such an approach is not allowed, it is required to start from the problem itself, and the algorithm/program should be written in a polynomial time, taking as parameters the size of the instances, not the instances themselves. In the general algorithm obtained in this way (one says that it is *uniform*) one introduces the instance to be solved. The idea is to prevent introducing the solution of an instance in the algorithm which pretends to solve the problem, but it simply provides the answer. Then, even for an uniform algorithm, one requests that the time of writing it is polynomial, so that it is not possible to work on solving the problem during the programming phase.

At the beginning of MC, a compromise between the two alternatives was chosen in the papers which proposed polynomial solutions to **NP**-complete problems: solutions which were not necessarily uniform, but at least "honest", starting from instances, but with the algorithm constructed in a polynomial time. They have been called *semi-uniform* solutions. And now the problems appear. How the complexity classes with respect to semi-uniform solutions look like and, more interesting, which are their relations with respect to the classic complexity classes? Of course, the semi-uniform classes are larger than the uniform ones. Are they strictly larger? Interesting enough, there were obtained results of both types, equality as well as strict inclusion, depending on the definition of the considered classes. I do not enter into further details, they become soon too technical.

Still further questions appear.

The brain (and the liver, and other organs and tissues) has a huge number of cells, not all of them working in each moment (at least this is what it was said some years ago). When a task appears, a problem for the brain, a food/beverage difficult to process for the liver, further cells are called to work. Can we use such a strategy in computer science, that is, to start from pre-computed resources, of an arbitrary size, without containing "too much" information, so that one cannot hide there a pre-computed solution of the problem, then to activate the "computer" according to the difficulty of the problem? This idea was followed in the framework of SN P systems, where the cell/neuron division does not seem to be biologically realistic, but it is natural to start from an arbitrarily large network of neurons, provided in advance, without containing spikes, and to introduce the problem to be solved, suitably encoded, in a small number of neurons, such that the network is activated as much as necessary, and it provides then the answer. Also in this area the theoretical developments are missing. When can we say that

the initial resources are pre-computed in a honest way, they do not contain too much information? How the associated complexity classes can look like, which is their relation with the existing complexity classes?

Maybe even more interesting: can the internet be used as a pre-computed support for computations, of an arbitrary size? In a certain sense, internet has been already used in this way – it was the case of the SETI project, maybe of other projects, known or not.

Something more: the algorithms considered in the complexity theory are deterministic, what it is happening in a test tube or in a cell is far from that. However, the result is either "reliable", in the sense that it is obtained with a probability close to 1 (Adleman has used sufficiently many DNA molecules so that he was "sure" that all paths in the graph were constructed in the first phase of the experiment), either the computation, even if it is non-deterministic, is *confluent*, with two possibilities: *strongly confluent*, that is, after a non-deterministic phase, the computation converges to a unique configuration, and after that the continuation is deterministic, or *weakly/logically confluent*, that is, irrespective which was the followed path, the answer is the same. In MC, most of the solutions were initially semi-uniform and (strongly or weakly) confluent, after that for a while the solutions were uniform and confluent, while in the last time most of the considered solutions are uniform and deterministic. Once again, there appears here a challenge for the complexity theory, to clarify the relationships between complexity classes defined using non-deterministic solutions, confluent solutions (in the two versions), and the deterministic solutions.

11. All these are challenges for the theory – also having some practical interpretations and consequences. However, let us refer also to some much more specific aspects, at least at the fist sight closer to applicative computer science. Let us compare the computer programs, as they are usually conceived, as precise sequences of instructions, with a well specified order in which they are executed, maybe with the ordering controlled by using labels, *go to* instructions, conditional instructions, cycles, with the "programs" in a cell and the completely different way they are executed. In a cell, the molecules, present in the form of multisets, react/evolve by the means of *possible* reactions. Instead of a sequence of instructions, we have a set of reactions, a non-ordered set, without any organization, which are applied to "data" (to the molecules) in a concurrent manner, each possible reaction in competition with other possible reactions in finding molecules to process, with many reactions developing concomitantly if sufficient reactants are available and if the necessary conditions are met for the reactions to take place (temperature, salinity, acidity, the presence of certain catalysts and promoters, the absence of inhibitors, and so on). There appear probabilities, stoichiometric coefficients which predict-control the frequency of applying the reactions, depending on the population of reactants and the reaction conditions, there appears the dependence on promoters and inhibitors, but not the precise chain of instruction from a program in Algol-Fortran-Pascal-Basic or in any programming language closer to our days. In a large extent, the functioning of a cell is much more similar to the functioning

of a Chomsky grammar than to the functioning of an automaton, of a Turing machine, where the states control the used instructions, maybe uniquely identifying them, in the case of deterministic automata.

Can we learn something at least interesting if not even useful for computability from these aspects?

Similarly in a tissue or in an organ, and still more in a colony of bacteria, there appears a functioning "in a community" which is not at all similar to the functioning of a set of processors put to work together. In biology, the parallelism is not total, the processes are not perfectly synchronized, in spite of the fact that there are rather precise "biological clocks", circadian cycles, annual seasons. For instance, the bacteriologists do not fully understand the way in which the bacteria realize the so-called *quorum sensing*, the "silent communication" before reaching a threshold after which the bacteria become aggressive. One discusses also in computer science about asynchronous computing, about amorphous computing, but nature is much ahead along these directions.

I end with another aspect of a practical interest: who is computing in Adleman experiment (and in many other similar experiments), who is the computer? The DNA molecules or... Adleman itself, the biochemist, the human? Without the human control, the molecules will not do what we expect from them to do, they will do almost nothing useful for us. It is true, a robotic system can replace the human operator. In this framework, "the computer" is a hybrid, and this is the plausible form of the computer of the future.

12. Of course, there are many other wonders-questions which are worth mentioning. In what concerns the DNA, we have remained at the level of the primary structure, while neither for the protein channels we have mentioned the fact that *the shape* plays a fundamental role, the coupling between surfaces is crucial for the execution of the symport/antiport operations.

Actually, one can imagine computations based on the matching of certain shapes (*computing by shapes*), a sort of puzzle game with specific rules and restrictions, which is again universal and which can get (purely theoretical for the time being) "implementations" in terms of DNA biochemistry.

Similarly, at the level of the brain there are many things to be discussed. An old hypothesis claims that there are two parts of the brain, one which is conscious, controllable, and one which is sub/inconscious; the first one sends problems to the latter one, which proposes solutions, which are evaluated by the cortex and, if they are real solutions, then the problem is solved, otherwise the problem is pushed once again to the subconscious part – and the process is iterated. There is here a dialogue between a deterministic component and a non-deterministic one, again something new for computer science. Can such a process be modeled, for instance, in terms of SN P systems? Has computer science something to learn from this supposed functioning of the brain? (Let us remember that **NP** is the class of problems for which a solution proposed non-deterministically can be deterministically checked in polynomial time, without counting any time for the "guessing"

of the solution, therefore a "bi-automaton, with one deterministic and one non-deterministic module", with the non-deterministic component working in no time, could be of a real practical interest.)

Then, concerning the learning: adaptation, evolution, learnability are common facts in biology, almost absent in the electronic technology, with the learning, at least, intimately present in the *neural computing*, a branch of bio-computing. Even if the type of learning here looks reductionistic, limited to the tuning of certain numerical coefficients (weights) on synapses linking "neurons" of a rather abstract form, this approach proves to be surprisingly efficient, having unexpectedly good results, comparable with the results which another area of bio-computing has, the evolutionary computing – without escaping the so-called *no free lunch theorems*. Recently, the neural computing has obtained a performance of a historical importance: one of the best GO players in the world was defeated, in March 2016, by *AlphaGO*, a..., a..., there is a problem here, as *AlphaGO* is not a program, it is not a computer (of the kind it was used twenty years ago in the game of chess, when Kasparov was defeated by *Deep Blue* computer). This time, billions of GO games among human players were stored on some thousands of computers, and a program based on neural computing has learned from these games and then from games played alone, the program against itself, so that, in the end, the South Korean GO player Lee Sedol, 9-dan, was defeated. The achievement belongs to a Google team, a detail of importance, because a routine to efficiently search through the thousand computers was also important.

The result is remarkable. For many years it was said that GO is the ultimate challenge for the artificial intelligence. Now, this frontier is already behind us. I confess that I have expected a bigger enthusiasm in the bio-computing community after this event.

Of course, now the challenge is to extend the strategy used by *AlphaGO* to other domains than GO, and I am sure that there are works in this direction.

I stop here, confident that the reader has his/her own wonders-questions in front of biology and in front of biology relations with other disciplines, from engineering to computer science. Anyway, the progresses in this area, of the collaboration of biology with computer science, should not be underestimated...

References

1. L.M. Adleman: Molecular computation of solutions to combinatorial problems. *Science*, 226 (Nov. 1994), 1021–1024.
2. B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter: *Molecular Biology of the Cell*. 4th Edition, Garland Science, New York, 2002.
3. P. Bottoni, G. Mauri, P. Mussio, Gh. Păun: Computing with shapes. *Journal of Visual Languages and Computing*, 12, 4 (2001), 601–626.
4. C. Calude, Gh. Păun: Bio-steps beyond Turing. *BioSystems*, 77 (2004), 175–194.
5. B.J. Copeland: Hypercomputation, *Minds and Machines*, 12, 4 (2002), 461–502.
6. M. Davis: The myth of hypercomputation. In *Alan Turing: Life and Legacy of a Great Thinker*, C. Teuscher, ed., Springer-Verlag, Heidelberg, 2003, 195–211.

7. J. Engelfriet, G. Rozenberg: Fixed point languages, equality languages, and representations of recursively enumerable languages. *Journal of the ACM*, 27 (1980), 499–518.
8. J. Gruska: *Quantum Computing*. McGraw-Hill, New York, 1999.
9. T. Head: Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors. *Bulletin of Mathematical Biology*, 49 (1987), 737–759.
10. T. Head, Gh. Păun, D. Pixton: Language theory and molecular genetics. Chapter 7 in vol. 2 of *Handbook of Formal Languages*, G. Rozenberg, A. Salomaa, eds., Springer-Verlag, Berlin, 1997, 295–360.
11. A.M. Ionescu, Gh. Păun, T. Yokomori: Spiking neural P systems. *Fundamenta Informaticae*, 71 (2006), 279–308.
12. T.-O. Ishdorj, A. Leporati: Uniform solutions to SAT and 3-SAT by spiking neural P systems with pre-computed resources. *Natural Computing*, 7, 4 (2008), 519–534.
13. V. Manca: On the logic and geometry of bilinear forms. *Fundamenta Informaticae*, 64 (2005), 261–273.
14. V. Manca: *Infobiotics. Information in Biotic Systems*. Springer-Verlag, Berlin, 2013.
15. S. Marcus: Linguistic structures and genetic devices in molecular genetics. *Cahiers de Linguistique Théorique et Appliquée*, 11, 2 (1974), 77–104.
16. L. Pan, T. Wu, Z. Zhang: A bibliography of spiking neural P systems, *Bulletin of IMCS*, 1 (June 2016), 63–78.
17. Ch.P. Papadimitriou: *Computational Complexity*. Addison-Wesley, Reading, Mass., 1994.
18. A. Păun, Gh. Păun: The power of communication: P systems with symport/antiport. *New Generation Computing*, 20, 3 (2002), 295–305.
19. A. Păun, Gh. Păun: Small universal spiking neural P systems. *BioSystems*, 90 (2007), 48–60.
20. Gh. Păun: On the splicing operation. *Discrete Applied Mathematics*, 70 (1996), 57–79.
21. Gh. Păun: Regular extended H systems are computationally universal. *Journal of Automata, Languages, and Combinatorics*, 1, 1 (1996), 27–36.
22. Gh. Păun: Computing by carving. *Soft Computing*, 3, 1 (1999), 30–36.
23. Gh. Păun: Computing with membranes. *Journal of Computer and Systems Sciences*, 61, 1 (2000), 108–143, and Turku Centre for Computer Science-TUCS Report No. 208 (1998).
24. Gh. Păun: P systems with active membranes: Attacking NP-complete problems. *Journal of Automata, Languages, and Combinatorics*, 6, 1 (2001), 75–90.
25. Gh. Păun: *Membrane Computing. An Introduction*. Springer-Verlag, Berlin, 2002.
26. Gh. Păun: Towards "fypercomputations" (in membrane computing). In *Languages Alive. Essays Dedicated to Jürgen Dassow on the Occasion of His 65 Birthday*, H. Bordihn, M. Kutrib, B. Truthe, eds., LNCS 7300, Springer-Verlag, Berlin, 2012, 207–221.
27. Gh. Păun: *Looking for Computers in the Biological Cell. After Twenty Years* (in Romanian). Reception Discourse in Romanian Academny, The Publ. House of the Romanian Academy, Bucharest, 2014.
28. Gh. Păun, G. Rozenberg, A. Salomaa: *DNA Computing. New Computing Paradigms*. Springer-Verlag, Berlin, 1998.
29. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*. Oxford Univ. Press, 2000.
30. M.J. Pérez-Jiménez, A. Riscos-Núñez, Á. Romero-Jiménez, D. Woods: Complexity – Membrane division, membrane creation. Chapter 12 in *The Oxford Handbook of*

Membrane Computing, Gh. Păun, G. Rozenberg, A. Salomaa, eds., Oxford Univ. Press, 2010, 302–336.
31. G. Rozenberg, A. Salomaa: *Watson-Crick Complementarity, Universal Computations and Genetic Engineering.* Technical Report 96-28, Dept. of Computer Science, Leiden University, Oct. 1996.
32. G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages*, 3 vols., Springer-Verlag, Berlin, 1997.
33. A.M. Turing: On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of London Mathematical Society*, Ser. 2, 42 (1936), 230–265; with a correction, 43 (1936), 544–546.
34. J.D. Watsonm, F.H. Crick: Molecular structure of nucleic acids. *Nature*, 4356 (April 25, 1953), 737–738.
35. C. Zandron: *A Model for Molecular Computing: Membrane Systems.* PhD Thesis, Universitá degli Studi di Milano, 2001.
36. G. Zhang, J. Cheng, T. Wang, X. Wang, J. Zhu: *Membrane Computing. Theory and Applications* (in Chinese). Science Press, Beijing, 2015.
37. The website of membrane computing: `http://ppage.psystems.eu.`
38. The website of *Bulletin of IMCS* (International Membrane Computing Society): `http://membranecomputing.net/IMCSBulletin/.`