THE PUBLISHING HOUSE
OF THE ROMANIAN ACADEMY

PROCEEDINGS OF THE ROMANIAN ACADEMY, Series A,
Volume 18, Number 2/2017, pp. 182–187

MINIMAL INGREDIENTS FOR TURING COMPLETENESS IN MEMBRANE COMPUTING

Bogdan AMAN, Gabriel CIOBANU

Romanian Academy, Institute of Computer Science
Corresponding author: Gabriel CIOBANU, E-mail: gabriel@info.uaic.ro

Dedicated to the 150th anniversary of the Romanian Academy

Abstract. In this paper we provide several computability results looking for minimal ingredients needed to obtain Turing completeness of various bio-inspired computation models (membrane systems). We emphasize the relevance of number two in reaching Turing completeness for several membrane systems.

Key words: membrane systems, Turing completeness.

1. INTRODUCTION

Membrane computing is a known branch of natural computing aiming to abstract computing ideas and formal models from the structure and functioning of living cells. The membrane systems (also known as P systems), are parallel and distributed computing models offering an approach to computational phenomena that is potentially superior to the one provided by conventional systems due to their inherent parallelism and non-determinism [26]. Computation is given by rewriting rules operating over multisets of objects (representing sets of objects with associated quantities) in cell-like compartmental architectures with polarizations (membrane electric charges). Objects represent the formal counterpart of the molecular species (ions, proteins) floating inside cellular compartments, while the evolution rules represent the formal counterpart of chemical reactions.

Three main research directions are usually considered in membrane computing: modelling of biological phenomena, computational power in terms of the classical notion of Turing computability using a minimal quantity of ingredients, and efficiency in algorithmically solving NP-complete problems in polynomial time (by using an exponential quantity of ingredients) [26].

Considering P systems with moving membranes [12], it is possible to model a part of the immune system [11], to get Turing completeness by using a small number of membranes [16], and to solve efficiently hard problems [14]. Moreover, these P systems are related to the process calculus of mobile ambients [9, 10].

Gh. Păun presented the following question (problem F in [25]): "*Can the polarizations be completely avoided? The feeling is that this is not possible, and such a result would be rather sound: passing from no polarization to two polarizations amounts to passing from non-efficiency to efficiency*."

Inspired by this question, we analyze minimal ingredients needed to get the Turing completeness. In particular, we show that two ingredients of some type are enough to obtain Turing completeness. There are several similarities: *two* is the minimal number of regions in a distributed system, two is the number of registers in a Turing complete register machine, and two is the number of states in the smallest unit of memory that is the bit. Focusing on membrane systems, the minimal number of ingredients needed in order to obtain Turing completeness for membrane systems is: *two* polarizations are enough in active membranes (Section 3), *two* catalysts in catalytic P systems (Section 4), either promoters or inhibitors of weight two, as well as *two* membranes with one catalyst and one promoter/inhibitor (Section 5), *two* membranes in splicing P systems (Section 6) and *two* membranes in P systems with symport/antiport rules (Section 7).

1. PREREQUISITES

The set of non-negative integers is denoted by $\mathbb{N}$. Given a finite alphabet $V = \{a_1, \ldots, a_n\}$, the free monoid generated by V under the operation of concatenation is denoted by V^*. The elements of V^* are called strings, and the empty string is denoted by λ. The number of occurrences of a symbol a_i in a string x is denoted by $|x|_{a_i}$, while the length of a string x is denoted by $|x| = \sum_{a_i} |x|_{a_i}$. The Parikh vector associated with a string x with respect to V is $(|x|_{a_1}, \ldots, |x|_{a_n})$. The Parikh image of an arbitrary language L over V is the set of all Parikh vectors of strings in L, and is denoted by $Ps(L)$. For a family of languages FL, the family of Parikh images of languages in FL is denoted by $PsFL$. A multiset over V is a mapping $f: V \to \mathbb{N}$; it can be represented by $\langle a_1^{f(a_1)}, \ldots, a_n^{f(a_n)} \rangle$. The families of regular and recursively enumerable languages are denoted by REG and RE respectively, and the family of Turing computable sets of non-negative integers by NRE. For more details of formal language theory the reader is referred to the handbook in this area [27].

2. P SYSTEMS WITH ACTIVE MEMBRANES

We consider membrane systems with active membranes [23] without division and dissolution rules (these rules are not needed here).

Definition 1. A P system with active membranes of degree d is a tuple

$$\Pi = (O, H, E, \mu, w_1, \ldots, w_d, e_1, \ldots, e_d, R),$$ where:

- O is a finite non-empty alphabet of *objects*;
- H is a finite *set of labels* for membranes;
- E is the set of *electrical charges* for membranes;
- μ is a *membrane structure* with membranes labelled by elements of H;
- $w_1, \ldots, w_d \in O^*$ describe the *initial multisets of objects* placed in μ;
- $e_1, \ldots, e_d$ are the *initial electrical charges* of the d membranes of μ;
- R is a finite set of rules:

(a) $[a \to w]_h^i$, $a \in O, v \in O^*, h \in H, i \in E$ — *object evolution rules*

An object a placed in a membrane labelled by h with polarization i is rewritten into the multiset w.

(b) $a[\,]_h^i \to [b]_h^j$, $a, b \in O, h \in H, i, j \in E$ — *communication rules*

An object a is sent into a membrane labelled by h, becoming b, possibly changing the polarization.

(c) $[a]_h^i \to b[\,]_h^j$, $a, b \ O, h \in H, i, j \in E$ — *communication rules*

An object a, placed into a membrane labelled by h, is sent out of membrane h and becomes b, possibly changing the polarization.

(c$_\lambda$) $[a]_h^i \to b[\,]_h^j$, $a \in O, b \in O^* \cup \{\lambda\}, h \in H, i, j \in E$ — *communication rules*

Same as rule *(c)*, except that b can also be deleted.

The rules of such a P system can be applied in a maximal parallel manner in one step (all rules should be used in parallel to the maximum degree possible), with the following constraint: at most one of the rules of types *(b)*, *(c)* and *(c$_\lambda$)* can to be used for each membrane in a given step. If no rule can be applied, the computation halts. Outputs are associated only with halting computations in the form of the objects sent into the environment during the computation. By $NOP_m(active_n, D)$ and $PsOP_m(active_n, D)$ we denote the family of all numbers, and respectively vectors of numbers computed by P systems with at most m membranes allowing n polarizations, using rules of the types contained in D, where $\varnothing \subset D \subseteq \{a, b, c, c_\lambda\}$.

It is enough to have *two* polarizations to achieve Turing completeness.

THEOREM 1 [5]. $NOP_1(active_2, \{a, c_\lambda\}) = NOP_2(active_2, \{a, c\}) = NRE$, *and*
$PsOP_1(active_2, \{a, c_\lambda\}) = PsOP_2(active_2, \{a, c\}) = PsRE$.

Turing completeness can be obtained without using membrane polarizations, but using division rules and a membrane structure where for each symbol appearing in the simulated matrix grammar there appear *two* membranes [1]. By considering also membrane creation rules, P systems with active membranes using *two* membrane labels and at most *two* objects are Turing complete [6].

3. CATALYTIC P SYSTEMS

Definition 2. A catalytic *P system* is a construct

$$\Pi = (O, C, \mu, w_1, \ldots, w_d, R_1, \ldots, R_d, i_o), \text{ where:}$$

- $d, O, \mu, w_1, \ldots, w_d$ are as in Definition 1;
- $C \subseteq O$ is the alphabet of *catalysts*;
- R_i , $1 \leq i \leq d$, are finite sets of evolution rules over O associated with the regions of μ; the rules have the forms $ca \rightarrow cv$ or $a \rightarrow v$, where c is a catalyst, a is an object from $O \backslash C$, and v is a string over $((O \backslash C) \times \{here, out, in\})$;
- $i_o \in \{0, 1, \ldots, d\}$ indicates the output region of Π.

The objects are transported through membranes according to the targets *in*, *out* and *here*. The evolution is similar with the one of the systems described in the previous section. A membrane system Π is called purely catalytic if every evolution rule involves a catalyst. The families of all sets of numbers or sets of vectors computed by [purely] catalytic P systems with at most m membranes and the set of catalysts containing at most k elements are denoted by $NOP_m([p]cat_k)$ and $PsOP_m([p]cat_k)$, respectively. We use the bracket notation [p] to specify the parameter p to be present. The next result claims that both catalytic P systems with no catalyst and purely catalytic P systems with one catalyst can only generate regular sets.

THEOREM 2. [26] $PsOP_1(cat_0) = PsOP_2(pcat_1) = PsREG$.

According to [4], for the (purely) catalytic P systems it is still open one of the challenging questions in the area of P systems, namely whether we need two or three catalysts to get Turing completeness. Now we have the following result.

THEOREM 3. [17] $NOP_2(cat_2) = NOP_2(pcat_3) = NRE$, and $PsOP_2(cat_2) = PsOP_2(pcat_3) = PsRE$.

Bistable catalysts (*2cat*) represent a special extension of the concept of catalysts able to change between *two* states when being used in a rule, i.e. the catalytic rules are of the form $ca \rightarrow c'v$ and $c'a \rightarrow cv$.

THEOREM 4. [2] $NOP_1(2cat_1) = NRE$.

Mobile catalysts (*mcat*) are allowed to have targets, i.e. the catalytic rules have the form $ca \rightarrow (c, tar)v$ with $c \in C$ and $tar \in \{here, out, in\}$.

THEOREM 5. [19] $NOP_2(mcat_2) = NRE$.

The concept of matter/antimatter was used in P systems: for each object a, the antimatter object a^- exists. We consider catalytic P systems extended by allowing annihilation rules $aa^- \rightarrow \lambda$. Other rules can be applied only if no annihilation rule could bind the corresponding objects. We can prove that these *two* types of objects (matter and antimatter) are powerful enough such that catalysts are not needed [3].

THEOREM 6. NOP_1 *(antim/pri)* $= NRE$ and $PsOP_1$ *(antim/pri)* $= PsRE$.

Using rules of the form $u \rightarrow v$ and no catalyst, the cooperative rules of degree *two* of the form $ab \rightarrow v$ are enough to obtain Turing completeness [24], while non-cooperative rules of the form $a \rightarrow v$ are not [22].

5. P SYSTEMS WITH PROMOTERS AND INHIBITORS

Membrane systems with promoters/inhibitors [15] represent an extension of the usual membrane systems. A rule with promoters of weight k has the form $u \to v|_p$, where $u, v, p \in O^*$ and $|p| \leq k$; such a rule can be applied only in the presence of the multiset of promoters p. The weight k can be interpreted here as the quantity of the promoters. A rule with inhibitors of weight k has the form $u \to v\,|_{\neg i}$, where $u, v, i \in O^*$ and $|i| \leq k$; such a rule can be applied only in the absence of the multiset of inhibitors i.

The difference between catalysts and promoters consists in the fact that catalysts directly participate in rules (but are not modified by them) and the number of applications of a catalytic rule cannot exceed the number of existing catalysts, while in the case of promoters it is possible to apply a rule as many times as possible in the presence of the corresponding promoter. Thus, the catalysts inhibit the high parallelism of the system while the promoters facilitate it (and guide the computation process).

We denote by $PsOP_m(\alpha,\beta)$, where $\alpha \in \{ncoo, coo\} \cup \{cat_k \mid k \geq 0\}$ and $\beta \in \{proR_w, inhR_w\}$, the family of sets of vectors of natural numbers generated by P systems with at most m membranes in which the evolution rules can be either non-cooperative (*ncoo*), cooperative (*coo*) or catalytic by using at most k catalysts (cat_k), as well as either promoters ($proR_w$) or inhibitors ($inhR_w$) of weight at most w.

P systems with inhibitors generate exactly the same family of Parikh sets as ET0L systems. P systems with promoters generate more than the family of Parikh sets of ET0L languages.

THEOREM 7 [28]. $PsOP_1(ncoo, proR_1) \supseteq PsOP_1(ncoo, inhR_1) = PsET0L$.

Using either promoters or inhibitors of weight *two*, the P systems are Turing complete.

THEOREM 8 [8]. $\mathrm{PsOP}_1(ncoo, proR_2) = PsOP_1(ncoo, inhR_2) = PsRE$.

The following Turing completeness results concern membrane systems with *two* membranes, one catalyst and either one promoter or inhibitor.

THEOREM 9 [18]. $PsOP_2(cat_1, proR_1) = PsOP_2(cat_1, inhR_1) = PsRE$.

6. SPLICING P SYSTEMS

Splicing rules are frequently written as $u_1\#u_2\$u_3\#u_4$, where $u_1, u_2, u_3, u_4 \in O$ and $\{\$,\#\} \notin O$. The strings u_1u_2 and u_3u_4 are called splicing sites.

Definition 3. A splicing *P system of degree* $d \geq 1$ is a construct $\Pi = (O, T, \mu, w_1, \ldots, w_d, R_1, \ldots, R_d)$, where:

- O, μ, and w_i are as in Definition 1;
- $T \subseteq O$ is the *terminal alphabet*;
- R_i, for $i \in \{1,\ldots, d\}$, are sets of evolution rules of the form $(r; tar_1, tar_2)$ associated with the compartments 1,..., d of μ, where r is a splicing rule over O and $tar_1, tar_2 \in \{here, in, out\}$ are target indicators.

A configuration of such a system is a tuple $(M_1,\ldots, M_d)$ over O. Formally, for $i \in \{1,\ldots, d\}$, if $x = x_1u_1u_2x_2$, $y = y_1u_3u_4y_2 \in M_i$ and $(r = u_1\#u_2\$u_3\#u_4; tar_1, tar_2) \in R_i$, $x_1, x_2, y_1, y_2, u_1, u_2, u_3, u_4 \in O$, the splicing $(x, y) \vdash_r (z,w)$, $z,w \in O^*$ can take place in i. The strings z and w pass to the compartments indicated by tar_1 and tar_2. A splicing P system is called non-extended if $O = T$. We use $ELSP_m(spl, in)$ for the family of languages generated by splicing P systems having a degree at most m, and $LSP_m(spl, in)$ for the non-extended case. The following theorem claims that splicing P systems with *two* compartments may generate any recursively enumerable language.

THEOREM 10 [26]. $ELSP_2(spl, in) = LSP_2(spl, in) = RE$.

7. SPLICING P SYSTEMS

Definition 4. A P system with antiport and symport rules of degree $d \geq 1$ is a construct

$\Pi = (O, T, E, \mu, w_1, \ldots, w_d, R_1, \ldots, R_d, i_o)$, where:

- $O, \mu, w_1, \ldots, w_d$ and i_o are as in Definition 2, while T is as in Definition 3;
- $E \subseteq O$ is a set of objects found in an unbounded number in the environment;
- R_i, for $i \in \{1,\ldots, d\}$, are sets of evolution rules $(u, out; v, in)$, with $u \neq \lambda$ and $v \neq \lambda$ (antiport rule), and (x, out) or (x, in), with $x \neq \lambda$ (symport rule).

The antiport rule $(u, out; v, in)$ in R_i exchanges the multiset u inside membrane i with the multiset v outside membrane i; the symport rule (x, out) sends the multiset x out of membrane i, and (x, in) takes x in from outside membrane i (if i is the skin membrane, then x must contain at least one symbol not in E).

When using a *minimally parallel* mode *min*, in each transition step we choose at least one rule from every set R_i of rules. $anti_k^s$ (sym_k) indicates that only antiport (symport) rules of weight at most k and size at most s are used, where the weight of an antiport rule $(u, out; v, in)$ is $max\{|u|, |v|\}$ and the size is $|u|+|v|$. The families of sets $N(\Pi)$ and $Ps(\Pi)$ computed by such symport/antiport P systems with at most d membranes with rules of type α working in *min* mode are denoted by $NOP_d(\alpha,min)$ and $PsOP_d(\alpha,min)$, respectively.

The first result concerning P systems with symport/antiport rules working in maximal parallel way claims that systems of *two* membranes using symport and antiport rules of weight *two* are enough for Turing completeness. In [21] the authors claim that they do not know whether this result is optimal, but it seems that simpler systems are not so powerful.

THEOREM 11 [21]. $NOP_2(anti_2, sym_2) = NRE$.

P systems with minimal symport/antiport rules with only *two* membranes are Turing complete [7]. Turing completeness can also be obtained using two membranes when working in the minimally parallel mode [26].

THEOREM 12. $NOP_2(anti_2^3, min) = NOP_2(sym_3, min) = NRE$,

and $PsOP_2(anti_2^3, min) = PsOP_2(sym_3, min) = PsRE$.

Another situation is given by P systems with string objects [22], where strings are used instead of objects and the evolution rules are based on string processing operations (in particular, on rewriting). Under some restrictions over rules, two membranes are enough to generate all *RE* languages.

The P systems were extended to tissue P systems with communication rules, where one rule (if applicable) is used on every channel. Turing completeness can be obtained either with one cell and *two* channels between the single cell and the environment, or with *two* cells and one channel between them [20].

8. CONCLUSION

In this paper we present a survey of computability results in membrane computing. It is worth noting the relevance of number two in reaching Turing completeness for these bio-inspired models; the number of *two* ingredients often represents a delimiting boundary. In many cases, passing from one ingredient to *two* ingredients represents the border between Turing non-completeness to Turing completeness. We are not aware of any other approach that comprehensively investigates when and why this delimiting boundary is found. Here we argued that *two* ingredients are enough to obtain Turing completeness for various P systems: (i) in active membranes with polarizations, *two* polarizations are enough for Turing completeness (neither dissolution nor division rules are used); (ii) in catalytic P systems either *two* catalysts or a bi stable catalyst (a catalyst with *two* states) or *two* mobile catalysts placed in *two* membranes are enough; (iii) in splicing P systems, P systems with symport/antiport rules and P systems with string objects *two* membranes are enough.

In [13] we prove that *two* membranes moving according to the endocytosis and exocytosis rules inside a skin membrane represent the minimum number of mobile membranes needed to obtain Turing completeness.

REFERENCES

1. A. ALHAZOV, *P Systems Without Multiplicities Of Symbol-Objects*, Information Processing Letters, **100**, *3*, pp. 124–129, 2006.
2. A. ALHAZOV, *Number of Protons/Bi-stable Catalysts and Membranes in P Systems*. *Time-Freeness*, Lecture Notes in Computer Science, **3850**, pp. 79–95, 2006.
3. A. ALHAZOV, B. AMAN, R. FREUND, *P Systems with Anti-Matter*, Lecture Notes in Computer Science, **8961**, pp. 66–85, 2014.
4. A. ALHAZOV, R. FREUND, *P Systems with Toxic Objects*, Lecture Notes in Computer Science, **8961**, pp. 99–125, 2014.
5. A. ALHAZOV, R. FREUND, GH. PĂUN, *Computational Completeness of P Systems with Active Membranes and Two Polarizations*, Lecture Notes in Computer Science, **3354**, pp. 82–92, 2005.
6. A. ALHAZOV, R. FREUND, A. RISCOS-NÚÑEZ, *Membrane Division, Restricted Membrane Creation and Object Complexity in P Systems*, International Journal of Computer Mathematics **83**, 7, pp. 529–547, 2006.
7. A. ALHAZOV, Y. ROGOZHIN, *Towards a Characterization of P Systems With Minimal Symport/Antiport and Two Membranes*, Lecture Notes in Computer Science, **4361**, pp. 135–153, 2002.
8. A. ALHAZOV, D. SBURLAN, *Ultimately Confluent Rewriting Systems. Parallel Multiset-Rewriting with Permitting or Forbidding Contexts*, Lecture Notes in Computer Science, **3365**, pp. 178–189, 2005.
9. B. AMAN, G. CIOBANU, *Translating Mobile Ambients into P Systems*, Electronic Notes in Theoretical Computer Science, **171**, *2*, pp. 11–23, 2007.
10. B. AMAN, G. CIOBANU, *On the Relationship Between Membranes and Ambients*, BioSystems, **91**, *3*, pp. 515–530, 2008.
11. B. AMAN, G. CIOBANU, *Describing the Immune System Using Enhanced Mobile Membranes*, Electronic Notes in Theoretical Computer Science, **194**, *3*, pp. 5–18, 2008.
12. B. AMAN, G. CIOBANU, Simple*, Enhanced and Mutual Mobile Membranes*, Transactions on Computational Systems Biology, **XI**, pp. 26–44, 2009.
13. B. AMAN, G. CIOBANU, Turing *Completeness Using Three Mobile Membranes*, Lecture Notes in Computer Science, **5715**, pp. 42–55, 2009.
14. B. AMAN, G. CIOBANU, *Solving a Weak NP-complete Problem in Polynomial Time by Using Mutual Mobile Membrane Systems*, Acta Informatica, **48**, *7–8*, pp. 409–415, 2011.
15. P. BOTTONI, C. MARTÍN-VIDE, GH. PĂUN, G. ROZENBERG. *Membrane Systems with Promoters/Inhibitors*, Acta Informatica, **38**, *10*, pp. 695–720, 2002.
16. G. CIOBANU, S.N. KRISHNA, *Enhanced Mobile Membranes: Computability Results*, Theory of Computing Systems, **48**, *3*, pp. 715–729, 2011.
17. R. FREUND, L. KARI, M. OSWALD, P. SOSÍK, *Computationally Universal P Systems without priorities: Two catalysts are sufficient*, Theoretical Computer Science, **330**, *2*, pp. 251–266, 2005.
18. M. IONESCU, D. SBURLAN, *On P Systems with Promoters/Inhibitors*, The Journal of Universal Computer Science, **10**, *5*, pp. 581–599, 2004.
19. S.N. KRISHNA, A. PĂUN, *Results on Catalytic And Evolution-Communication P Systems*, New Generation Computing, **22**, pp. 377–394, 2004.
20. C. MARTÍN-VIDE, J. PAZOS, GH. PĂUN, A. RODRÍGUEZ-PATÓN, *Tissue P Systems*, Theoretical Computer Science, **296**, pp. 295–326, 2003.
21. A. PĂUN, GH. PĂUN, The *Power of Communication: P Systems with Symport/Antiport*, New Generation Computing, **20**, pp. 295–305, 2002.
22. GH. PĂUN, *Computing with Membranes*, Journal of Computer and System Sciences, **61**, *1*, 108–143, 2000.
23. GH. PĂUN, *P Systems with Active Membranes: Attacking NP-complete Problems*, Journal of Automata, Languages and Combinatorics, **6**, pp. 75–90, 2001.
24. GH. PĂUN, Membrane *Computing. An Introduction*, Springer, 2002.
25. GH. PĂUN, *Further Twenty Six Open Problems in Membrane Computing*, Third Brainstorming Week on Membrane Computing, Fénix Editora, Sevilla, pp. 249–262, 2005.
26. GH. PĂUN, G. ROZENBERG, A. SALOMAA (Eds.), *The Oxford Handbook of Membrane Computing*, Oxford University Press, 2010.
27. G. ROZENBERG, A. SALOMAA (Eds.): *Handbook of Formal Languages*, Springer, 1997.
28. D. SBURLAN, *Further Results on P Systems with Promoters/Inhibitors*, International Journal of Foundations of Computer Science, **17**, 1, pp. 205–221, 2006.

Received June 22, 2015