P Systems with Anti-Membranes

Artiom Alhazov\(^1\), Rudolf Freund\(^2\), Sergiu Ivanov\(^3\)

1 Vladimir Andrunachievici Institute of Mathematics and Computer Science
   Academiei 5, Chișinău, MD-2028, Moldova
   artiom@math.md

2 TU Wien, Institut für Logic and Computation
   Favoritenstraße 9-11, 1040 Wien, Austria
   rudi@uncc.at

3 IBISC, Université Évry, Université Paris-Saclay
   23, boulevard de France, 91034 Évry, France
   sergiu.ivanov@univ-evry.fr

Summary. The concept of a matter object being annihilated when meeting its corresponding anti-matter object is taken over for membranes as objects and anti-membranes as the corresponding annihilation counterpart in P systems. Natural numbers can be represented by the corresponding number of membranes with a specific label. Computational completeness in this setting then can be obtained with using only elementary membrane division rules, without using objects.

1 Introduction

The basic model of P systems as introduced in [12] can be considered as a distributed multiset rewriting system, where all objects – if possible – evolve in parallel in the membrane regions and may be communicated through the membranes. Overviews on the field of P systems can be found in the monograph [13] and the handbook of membrane systems [14]; for actual news and results we refer to the P systems webpage [16] as well as to the Bulletin of the International Membrane Computing Society.

Computational completeness (computing any partial recursive relation on non-negative integers) can be obtained with using cooperative rules or with catalytic rules (possibly) together with non-cooperative rules. We recall that non-cooperative rules have the form \(a \rightarrow w\), where \(a\) is a symbol and \(w\) is a multiset, catalytic rules have the form \(ca \rightarrow cw\), where the symbol \(c\) is called the catalyst, and cooperative rules have no restrictions on the form of the left-hand side.
only one catalyst is needed, for example, see [6, 8, 9]. In [2, 1], another concept to avoid cooperative rules is investigated: for any object \( a \) (matter), its anti-object (anti-matter) \( a^\prime \) is considered together with the corresponding annihilation rule \( a \rightarrow \lambda \), which is assumed to exist in all membranes; this annihilation rule is assumed to be a special non-cooperative rule having priority over all other rules in the sense of weak priority (e.g., see [3], i.e., other rules then also may be applied if objects cannot be bound by some annihilation rule any more). For spiking neural P systems, the idea of anti-matter has been introduced in [11] with anti-spikes as anti-matter objects. In [5] the power of anti-matter for solving NP-complete problems is exhibited.

Although, as expected (for example, compare with the Geffert normal forms, see [15]), the annihilation rules are rather powerful, it is still surprising that using matter/anti-matter annihilation rules as the only non-cooperative rules, with the annihilation rules having weak priority, computational completeness can already be obtained without using any catalyst, see [2, 1], whereas usually at least one catalyst is needed even when using other control mechanisms, for example, see [2].

Natural numbers can be represented by the corresponding number of membranes with a specific label. Hence, in this paper we take over the idea of anti-objects for membranes, i.e., for every membrane \( [x] \) we take the anti-membrane \( [x^\prime] \), and the membrane/anti-membrane annihilation rule \( [x] \rightarrow [x^\prime] \rightarrow \lambda \). In the simplest case, we only use elementary membranes, but objects, and membrane division, i.e., rules of the form \( [x] \rightarrow [y] \rightarrow [z] \), possibly also allowing membrane renaming rules of the form \( [x] \rightarrow [y] \rightarrow [z] \) or membrane deletion rules of the form \( [x] \rightarrow [x^\prime] \rightarrow [\lambda] \). In this setting, computational completeness can then be obtained with using only elementary membrane division rules, without using objects, together with anti-membranes and membrane/anti-membrane annihilation rules.

2 Prerequisites

The set of integers is denoted by \( \mathbb{Z} \), and the set of non-negative integers by \( \mathbb{N} \). Given an alphabet \( V \), a finite non-empty set of abstract symbols, the free monoid generated by \( V \) under the operation of concatenation is denoted by \( V^* \). The elements of \( V^* \) are called strings, the empty string is denoted by \( \lambda \), and \( V^\ast \{ \lambda \} \) is denoted by \( V^+ \). For an arbitrary alphabet \( V = \{ a_1, \ldots, a_n \} \), 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 \( |x| = \sum_{a_i \in V} |x|_{a_i} \). The Parikh vector associated with \( x \) w.r.t. \( a_1, \ldots, a_n \) is \( (|x|_{a_1}, \ldots, |x|_{a_n}) \). The Parikh image of an arbitrary language \( L \) over \( \{ a_1, \ldots, a_n \} \), 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 \( PSFL \), while for families of languages over a one-letter (d-letter) alphabet, the corresponding sets of non-negative integers (d-vectors with non-negative components) are denoted by \( NFL \) (\( N^dFL \)).

\((|x|_{a_1}, \ldots, |x|_{a_n}) = (f(a_1), \ldots, f(a_n))\). In the following we will not distinguish between a vector \((m_1, \ldots, m_n)\), a multiset \((a_1^{m_1}, \ldots, a_n^{m_n})\) or a string \( x \) having \(|x|_{a_1}, \ldots, |x|_{a_n} = (m_1, \ldots, m_n)\). Fixing the sequence of symbols \( a_1, \ldots, a_n \) in an alphabet \( V \), in advance, the representation of the multiset \((a_1^{m_1}, \ldots, a_n^{m_n})\) by the string \( a_1^{m_1} \cdots a_n^{m_n} \) is unique. The set of all finite multisets over an alphabet \( V \) is denoted by \( V^* \).

The family of regular and recursively enumerable string languages is denoted by \( REG \) and \( RE \), respectively. For more details of formal language theory the reader is referred to the monographs and handbooks in this area as [4] and [15].

Register machines

A register machine is a tuple \( M = (m, B, l_0, l_f, P) \), where \( m \) is the number of registers, \( B \) is a set of labels, \( l_0 \in B \) is the initial label, \( l_f \in B \) is the final label, and \( P \) is the set of instructions bijectively labeled by elements of \( B \). The instructions of \( M \) can be of the following forms:

- \( l_i : (ADD)(j), l_2, l_3, B \), with \( l_i \in B \setminus \{ l_0, l_f \} \), \( l_2, l_3 \in B, 1 \leq j \leq m \).
  
  Increases the value of register \( j \) by one, followed by a non-deterministic jump to instruction \( l_2 \) or \( l_3 \). This instruction is usually called increment.

- \( l_i : (SUB)(j), l_2, l_3, B \), with \( l_i \in B \setminus \{ l_0, l_f \} \), \( l_2, l_3 \in B, 1 \leq j \leq m \).
  
  If the value of register \( j \) is zero then jump to instruction \( l_3 \); otherwise, the value of register \( j \) is decreased by one, followed by a jump to instruction \( l_2 \).
  
  The two cases of this instruction are usually called zero-test and decrement, respectively.

- \( l_h : (HALT) \). Stops the execution of the register machine.

A configuration of a register machine is described by the contents of each register and by the value of the current label, which indicates the next instruction to be executed. Computations start by executing the instruction \( l_0 \) of \( P \), and terminate with reaching the HALT-instruction \( l_h \). For useful results on the computational power of register machines, we refer to [10].

3 P Systems with Active Membranes and Anti-Membranes

For using anti-matter as a frontier of tractability, we refer to [5], where some standard definition of P systems with active membranes can be found. Here we consider a special rather restricted model, where no objects are used and inside the skin membrane only the following types of rules for elementary membranes are used:

- Elementary membrane division \( [x] \rightarrow [y] \rightarrow [z] \)
- The elementary membrane \( [x] \) is divided into two membranes, possibly changing the label \( h \) of the parent membrane \( [x] \) to two new labels \( h', h'' \) for the two new membranes \( [y] \) and \( [z] \), respectively.

where \( f(a_1), \ldots, f(a_n) \) is the non-cooperative rule having priority over all other rules in the sense of weak priority (e.g., see [3], i.e., other rules then also may be applied if objects cannot be bound by some annihilation rule any more). For spiking neural P systems, the idea of anti-matter has been introduced in [11] with anti-spikes as anti-matter objects. In [5] the power of anti-matter for solving NP-complete problems is exhibited.

Although, as expected (for example, compare with the Geffert normal forms, see [15]), the annihilation rules are rather powerful, it is still surprising that using matter/anti-matter annihilation rules as the only non-cooperative rules, with the annihilation rules having weak priority, computational completeness can already be obtained without using any catalyst, see [2, 1], whereas usually at least one catalyst is needed even when using other control mechanisms, for example, see [2].

Natural numbers can be represented by the corresponding number of membranes with a specific label. Hence, in this paper we take over the idea of anti-objects for membranes, i.e., for every membrane \( [x] \) we take the anti-membrane \( [x^\prime] \), and the membrane/anti-membrane annihilation rule \( [x] \rightarrow [x^\prime] \rightarrow \lambda \). In the simplest case, we only use elementary membranes, but objects, and membrane division, i.e., rules of the form \( [x] \rightarrow [y] \rightarrow [z] \), possibly also allowing membrane renaming rules of the form \( [x] \rightarrow [y] \rightarrow [z] \) or membrane deletion rules of the form \( [x] \rightarrow [x^\prime] \rightarrow [\lambda] \). In this setting, computational completeness can then be obtained with using only elementary membrane division rules, without using objects, together with anti-membranes and membrane/anti-membrane annihilation rules.

3 P Systems with Active Membranes and Anti-Membranes

For using anti-matter as a frontier of tractability, we refer to [5], where some standard definition of P systems with active membranes can be found. Here we consider a special rather restricted model, where no objects are used and inside the skin membrane only the following types of rules for elementary membranes are used:

- Elementary membrane division \( [x] \rightarrow [y] \rightarrow [z] \)
- The elementary membrane \( [x] \) is divided into two membranes, possibly changing the label \( h \) of the parent membrane \( [x] \) to two new labels \( h', h'' \) for the two new membranes \( [y] \) and \( [z] \), respectively.
4 Results

As a first result, we observe that rules changing membrane label, i.e., \( \lvert h \rvert \rightarrow \lvert h' \rvert \), and elementary membrane deletion rules, i.e., \( \lvert h \rvert \rightarrow \lambda \), are not needed and can be replaced by using only elementary membrane division and suitable membrane/anti-membrane annihilation rules.

**Lemma 1.** Rules changing membrane label, i.e., \( \lvert h \rvert \rightarrow \lvert h' \rvert \), and elementary membrane deletion rules, i.e., \( \lvert h \rvert \rightarrow \lambda \), can be simulated by elementary membrane division and membrane/anti-membrane annihilation rules.

**Proof.** A rule changing the membrane label, i.e., \( \lvert h \rvert \rightarrow \lvert h' \rvert \), can be simulated by the rules \( \lvert h \rvert \rightarrow \lvert h' \rvert \), \( \lvert h \rvert \rightarrow \lambda \), \( \lvert h' \rvert \rightarrow \lambda \), and \( \lvert h \rvert \rightarrow \lambda \), where \( h', g, g' \) are new labels.

An elementary membrane deletion rules, i.e., \( \lvert h \rvert \rightarrow \lambda \), can be simulated by the rules \( \lvert h \rvert \rightarrow \lambda \) and \( \lvert h \rvert \rightarrow \lambda \), where \( g, g' \) are new labels.

A PAMS only using elementary membrane division and membrane/anti-membrane annihilation rules is called a PAMS in normal form. As an immediate consequence of the preceding lemma we obtain the following normal form theorem:

**Theorem 1.** For every PAMS \( \Pi \) we can construct a PAMS \( \Pi' \) in normal form such that \( N_\delta(\Pi) = N_\delta(\Pi') \) and \( P_{\delta}(\Pi) = P_{\delta}(\Pi') \), with \( \delta \in \{ \text{gen, acc} \} \).

We now show that PAMSs characterize the sets \( NRE \) and \( PsRE \), respectively.

The main proof idea is as used very often in the area of P systems – is to simulate (the computations of) register machines, as carried out in a similar way in [1] for P systems with anti-matter.

**Theorem 2.** For any \( Y \in \{ N, Ps \} \) and \( \delta \in \{ \text{gen, acc} \} \), \( Y_\delta(PAMS) = YRE \).

**Proof.** Let \( M = (m, B, l_0, l_1, P) \) be a register machine. We now construct a PAMS \( \Pi \) which simulates (the computations of) \( M \):

- \( \Pi = (H \cup \{0\}, \lvert 0, w_0, R \rvert); \)
- \( H = \{ \langle r, \langle 1 \leq r \leq m \rangle \cup \{ l' \mid l \in B \} \cup \{ \#', \#'' \} \} \) is the set of labels for the elementary membranes inside the skin membrane;
  - the label \( r, 1 \leq r \leq m \), is for the copies of membrane \( \lvert \_ \rvert \), representing the contents of register \( r \); the labels \( r' \) are for the corresponding anti-membranes;
  - in the generative case, initially the skin membrane contains only the elementary membrane \( \lvert 0 \rvert \); in the accepting case, suitable copies of membranes for representing the input vector are to be added;
- \( R \) contains the rules described in the following.

The contents of register \( r \) is represented by the number of copies of the elementary membrane \( \lvert \_ \rvert \), \( 1 \leq r \leq m \), and for each membrane \( \lvert \_ \rvert \), we also consider...
5 Conclusion

In this paper, we have taken over the idea of matter and anti-matter objects in P systems to P systems with active membranes, now considering membranes and anti-membranes as the objects interacting with each other in annihilation rules, which we assumed to have weak priority over all other rules. We have investigated a restricted model of P systems with active membranes, without any objects in the whole system and instead only elementary membranes in the skin membrane. In this model, natural numbers are represented as copies of elementary membranes with a specific label. In such a variant of P systems with active membranes, computations of register machines can be simulated by using only (a special variant of) elementary membrane division rules and membrane/anti-membrane annihilation rules.

There are several other interesting variants of P systems allowing for introducing anti-membranes and membrane/anti-membrane annihilation rules. For example, instead of membranes inside the skin membrane, we may consider tissue-like P systems where the skin is replaced by the environment and the labeled membranes now correspond to labeled cells which may interact with each other in cell/anti-cell annihilation rules.

On the other hand, in a more general model, we need not restrict ourselves to elementary membranes interacting with each other in membrane/anti-membrane annihilation rules. In fact, we may consider a variant where in such a reaction only the outermost membranes of two non-elementary membranes react, emitting the interior membrane structure into the skin membrane. In such a variant, non-elementary membrane division becomes relevant, as well as rules allowing for putting a new membrane around a given membrane structure, i.e., rules of the form \([\lambda]_k \rightarrow [\lambda]_l\). Finally, as it is common in P systems with active membranes, in addition objects may be added and guide the membrane rules (yet still evolution rules for the objects may be forbidden). Such variants remain to be investigated in some future papers based on this introductory one.

Acknowledgements

The ideas for this paper came up in the inspiring atmosphere of the Brainstorming Week on Membrane Computing in Sevilla this year.

References

Synchronization of Rules in Membrane Computing

Bogdan Aman, Gabriel Ciobanu

1. Alexandru Ioan Cuza University of Iaşi, Romania
2. Romanian Academy, Institute of Computer Science
bogdan.aman@it.academiaromana-is.ro, gabriel@info.uain.ro

Abstract. We adjust the most used evolution strategy in membrane systems, namely that of maximal parallelism, by imposing an additional synchronization between rules. A rule synchronizing with a set of rules can be applied only if each rule from the set can be applied at least once. For membrane systems working in the accepting mode, this synchronization is powerful enough to provide the computational completeness without any other ingredient (no catalysts, promoters or inhibitors, for instance). The modelling power of synchronization is described by simulating the basic arithmetic operations (addition, subtraction, multiplication and division).

1 Introduction

Membrane systems (also known as P systems) are able to model parallel distributed systems inspired by structure and behaviour of biological cells [18]. A membrane system can be represented as a hierarchical structure of regions (membranes) contained inside a unique outermost membrane called skin. In this paper, we consider the class of P systems defined in [19] in which the various regions of the membrane structure contain multisets of objects and sets of evolution rules. Every region has its own task such that all regions work in parallel to achieve the general task of the entire system; the specific rules of each region modify its objects. The evolution of the initial class of P systems is given by the maximal parallelism in applying the rules [18]. The maximal parallelism ensures that the multiset of applicable rules chosen in a computation step cannot be further extended by adding further rules. This feature was preserved in many of the variants defined in the last twenty years, being a useful feature in obtaining computational completeness. Choosing the rules to be applied in a maximally parallel way is done non-deterministically, by respecting also some restrictions (e.g., priority relation among rules) or value-based criteria (e.g., the guards used in adaptive P systems [7] or kernel P systems [17]). Various results and classes of membrane systems (motivated by different features