(Tissue) 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, Chişinău, MD-2028, Moldova
E-mail: artiom@math.md

[2] TU Wien, Institut für Logic and Computation
Favoritenstraße 9–11, 1040 Wien, Austria
E-mail: rudi@emcc.at

[3] IBISC, Université Évry, Université Paris-Saclay
23, boulevard de France, 91034 Évry, France
E-mail: 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. A similar result can be obtained for tissue P systems with cell division rules and cell / anti-cell annihilation rules. In both cases, as derivation modes we may take the standard maximally parallel derivation modes as well as any of the maximally parallel set derivation modes (non-extendable (multi)sets of rules, (multi)sets with maximal number of rules, (multi)sets of rules affecting the maximal number of 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 $a \to w$, where a is a symbol and w is a multiset,

catalytic rules have the form $ca \to cw$, where the symbol c is called the catalyst, and cooperative rules have no restrictions on the form of the left-hand side. Without additional control mechanisms, at least two catalysts are needed, see [7]. Using specific control mechanisms, as for example, rule labels or target agreement, 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^- is considered together with the corresponding *annihilation rule* $aa^- \to \lambda$, 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 $[\ ]_h$ we take the anti-membrane $[\ ]_{h^-}$ and the membrane / anti-membrane annihilation rule $[\ ]_h[\ ]_{h^-} \to \lambda$. In the simplest case, we only use elementary membranes, but no objects, and elementary membrane division, i.e., rules of the form $[\ ]_h \to [\ ]_{h'}[\ ]_{h''}$, possibly also allowing membrane renaming rules of the form $[\ ]_h \to [\ ]_{h'}$ or membrane deletion rules of the form $[\ ]_h \to \lambda$. In this setting, computational completeness then can be obtained with using only elementary membrane division rules, without using objects, together with anti-membranes and membrane / anti-membrane annihilation rules.

Natural numbers can also be represented by the corresponding number of cells with a specific label. Hence, a similar computational completeness result can also be obtained for tissue P systems with cell division rules and cell / anti-cell annihilation rules.

In both cases, as derivation modes we may take the standard maximally parallel derivation modes as well as any of the maximally parallel set derivation modes (non-extendable (multi)sets of rules, (multi)sets with maximal number of rules, (multi)sets of rules affecting the maximal number of objects).

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 λ, and $V^*\backslash\{\lambda\}$ 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 with respect to $a_1, \ldots, a_n$ 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).

A (finite) multiset over a (finite) alphabet $V = \{a_1, \ldots, a_n\}$, is a mapping $f : V \to \mathbb{N}$ and can be represented by $\langle a_1^{f(a_1)}, \ldots, a_n^{f(a_n)} \rangle$ or by any string x for which $(|x|_{a_1}, \ldots, |x|_{a_n}) = (f(a_1), \ldots, f(a_n))$. In the following we will not distinguish between a vector $(m_1, \ldots, m_n)$, a multiset $\langle a_1^{m_1}, \ldots, a_n^{m_n} \rangle$ 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 $\langle a_1^{m_1}, \ldots, a_n^{m_n} \rangle$ by the string $a_1^{m_1} \ldots a_n^{m_n}$ 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 $M = (m, B, l_0, l_h, P)$, where m is the number of registers, B is a set of labels, $l_0 \in B$ is the initial label, $l_h \in B$ 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:

- $l_1 : (ADD(j), l_2, l_3)$, with $l_1 \in B\backslash\{l_h\}$, $l_2, l_3 \in B$, $1 \leq j \leq m$.
  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*.
- $l_1 : (SUB(j), l_2, l_3)$, with $l_1 \in B\backslash\{l_h\}$, $l_2, l_3 \in B$, $1 \leq j \leq m$.
  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. We here 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 $[\,]_h \to [\,]_{h'}[\,]_{h''}$
the elementary membrane $[\,]_h$ is divided into two membranes, possibly changing the label h of the parent membrane $[\,]_h$ to two new labels h', h'' for the child membranes $[\,]_{h'}$ and $[\,]_{h''}$

changing membrane label $[\,]_h \to [\,]_{h'}$
the label h of the elementary membrane $[\,]_h$ is changed to h'

elementary membrane deletion $[\,]_h \to \lambda$
the elementary membrane $[\,]_h$ is deleted

membrane / anti-membrane annihilation $[\,]_h[\,]_{h^-} \to \lambda$
the elementary membrane $[\,]_h$ and its corresponding anti-membrane $[\,]_{h'}$ annihilate each other

Formally, a *P system with active membranes and anti-membranes* (a *PAMS* for short) is a construct $\Pi = (H \cup \{0\}, [\,]_0, w_0, R)$ where H is the set of *membrane labels* used in the membrane rules specified in R, $[\,]_0$ denotes the skin membrane enclosing the initial set of elementary membranes w_0 with labels from H, and R is the set of rules of the forms described above, with the labels of the elementary membranes taken from H.

In any computation step of Π a multiset of rules is chosen from the set R in such a way that no further rule can be added to it so that the obtained multiset would still be applicable to the existing membranes in the skin membrane. We emphasize that membrane / anti-membrane annihilation rules have weak priority over all other rules, i.e., as long as membrane / anti-membrane annihilation rules may bind some membranes, other rules are not allowed to yet be taken into the multiset of rules constructed to be maximal.

A *configuration* of the system can be represented by the membranes inside the skin membrane. Starting from a given *initial configuration* and applying evolution rules as described above, we get *transitions* among configurations; a sequence of transitions forms a *computation*. A computation is *halting* if it reaches a configuration where no rule can be applied any more.

In the *generative case*, a halting computation has associated a result, in the form of the number of membranes with the same labels present in the skin membrane; their numbers represents a vector of natural numbers. In the *accepting case*, all (vectors of) non-negative integers are accepted whose input, given as the corresponding numbers of membranes in the skin membrane in addition to w_0, leads

to a halting computation. The set of non-negative integers and the set of (Parikh) vectors of non-negative integers generated/accepted as results of halting computations in Π are denoted by $N_\delta(\Pi)$ and $Ps_\delta(\Pi)$, respectively, with $\delta \in \{gen, acc\}$. The corresponding families of sets of non-negative integers and the sets of vectors of non-negative integers generated/accepted by PAMSs are denoted by N_δ(PAMS) and Ps_δ(PAMS), respectively.

4 Tissue P Systems with Cell Division and Anti-Cells

Instead of considering elementary membranes inside the skin membrane, we may also consider cells floating in a common environment. Then instead of anti-membranes, we consider anti-cells, i.e., cells with the anti-label. Again, we here consider a special rather restricted model, where no objects are used and only the following types of rules for cells in the tissue P system are used:

cell division $\bigcirc_h \rightarrow \bigcirc_{h'} \bigcirc_{h''}$
the cell $\bigcirc_h$ is divided into two cells, possibly changing the label h of the parent cell $\bigcirc_h$ to two new labels h', h'' for the child cells $\bigcirc_{h'}$ and $\bigcirc_{h''}$

changing cell label $\bigcirc_h \rightarrow \bigcirc_{h'}$
the label h of cell $\bigcirc_h$ is changed to h'

cell deletion $\bigcirc_h \rightarrow \lambda$
the cell $[\ ]_h$ is deleted

cell / anti-cell annihilation $\bigcirc_h \bigcirc_{h^-} \rightarrow \lambda$
the cell $\bigcirc_h$ and its corresponding anti-cell $\bigcirc_{h^-}$ annihilate each other

Formally, a *tissue P system with anti-cells* (a *tPAMS* for short) is a construct $\Pi = (H, w_0, R)$ where H is the set of *cell labels* used in the rules specified in R, w_0 is the initial set of cells with labels from H, and R is the set of rules of the forms described above, with the labels of the cells taken from H.

In any computation step of Π a multiset of rules is chosen from the set R in such a way that no further rule can be added to it so that the obtained multiset would still be applicable to the existing cells. We emphasize that again we assume cell / anti-cell annihilation rules to have weak priority over all other rules, i.e., as long as cell / anti-cell annihilation rules may bind some cells, other rules are not allowed to yet be taken into the multiset of rules constructed to be maximal.

A *configuration* of the system can be represented by the currently existing cells. Starting from a given *initial configuration* and applying evolution rules as described above, we get *transitions* among configurations; a sequence of transitions forms a *computation*. A computation is *halting* if it reaches a configuration where no rule can be applied any more.

In the *generative case*, a halting computation has associated a result, in the form of the number of cells present in the system; their numbers represents a vector of natural numbers. In the *accepting case*, all (vectors of) non-negative integers are accepted whose input, given as the corresponding numbers of initial cells in addition to w_0, leads to a halting computation. The set of non-negative integers and the set of (Parikh) vectors of non-negative integers generated/accepted as results of halting computations in Π are denoted by $N_\delta(\Pi)$ and $Ps_\delta(\Pi)$, respectively, with $\delta \in \{gen, acc\}$. The corresponding families of sets of non-negative integers and the sets of vectors of non-negative integers generated/accepted by tPAMSs are denoted by N_δ(tPAMS) and Ps_δ(tPAMS), respectively.

5 Results

As a first result, we observe that rules changing membrane label, i.e., $[\,]_h \to [\,]_{h'}$, and elementary membrane deletion rules, i.e., $[\,]_h \to \lambda$, 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.,* $[\,]_h \to [\,]_{h'}$*, and elementary membrane deletion rules, i.e.,* $[\,]_h \to \lambda$*, can be simulated by elementary membrane division and membrane / anti-membrane annihilation rules.*

Proof. A rule changing the membrane label, i.e., $[\,]_h \to [\,]_{h'}$, can be simulated by the rules $[\,]_h \to [\,]_{h'}[\,]_{h''}$, $[\,]_{h''} \to [\,]_g[\,]_{g^-}$, and $[\,]_g[\,]_{g^-} \to \lambda$, where h'', g, g^- are new labels (separately for each label h).

An elementary membrane deletion rule, i.e., $[\,]_h \to \lambda$, can be simulated by the rules $[\,]_h \to [\,]_g[\,]_{g^-}$ and $[\,]_g[\,]_{g^-} \to \lambda$, where g, g^- are new labels (separately for each label h). □

A similar result obviously also holds for tPAMS: rules changing a cell label, i.e., $\bigcirc_h \to \bigcirc_{h'}$, and cell deletion rules, i.e., $\bigcirc_h \to \lambda$, are not needed and can be replaced by using only cell division and suitable cell / anti-cell annihilation rules. The corresponding proof verbatim follows the proof of Lemma 1, just replacing the notation $[\,]_h$ by $\bigcirc_h$.

Corollary 1. *Rules changing cell label, i.e.,* $\bigcirc_h \to \bigcirc_{h'}$*, and cell deletion rules, i.e.,* $\bigcirc_h \to \lambda$*, can be simulated by cell division and cell / anti-cell annihilation rules.*

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 Lemma 1 we obtain the following normal form theorem:

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

A similar normal form result obviously also holds for tPAMS as an immediate consequence of Corollary 1:

Corollary 2. *For every tPAMS Π we can construct a tPAMS Π' in normal form such that $N_\delta(\Pi) = N_\delta(\Pi')$ and $Ps_\delta(\Pi) = Ps_\delta(\Pi')$, with $\delta \in \{gen, acc\}$.*

5.1 Computational Completeness

We now show that PAMSs characterize the families NRE and $PsRE$, respectively. The main proof idea – 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 $\delta \in \{gen, acc\}$,*

$$Y_\delta(PAMS) = YRE.$$

Proof. Let $M = (m, B, l_0, l_h, P)$ be a register machine. We now construct a PAMS Π which simulates (the computations of) M:

- $\Pi = (H \cup \{0\}, [\,]_0, w_0, R)$;
- $H = \{r, r^- \mid 1 \leq r \leq m\} \cup \{l, l' \mid l \in B\} \cup \{\#, \#^-\}$ 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 $[\,]_r$ representing the contents of register r; the labels r^- are for the corresponding anti-membranes;
- in the generating case, initially the skin membrane contains only the elementary membrane $[\,]_{l_0}$; 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 $[\,]_r$, $1 \leq r \leq m$, and for each membrane $[\,]_r$ we also consider the corresponding anti-membrane $[\,]_{r^-}$.

The instructions of M are simulated by the following rules in R_1:

- $l_1 : (ADD(j), l_2, l_3)$, with $l_1 \in B \setminus \{l_h\}$, $l_2, l_3 \in B$, $1 \leq j \leq m$.
  Simulated by the rules
  $$[\,]_{l_1} \rightarrow [\,]_r [\,]_{l_2} \text{ and } [\,]_{l_1} \rightarrow [\,]_r [\,]_{l_3}.$$

- $l_1 : (SUB(r), l_2, l_3)$, with $l_1 \in B \setminus \{l_h\}$, $l_2, l_3 \in B$, $1 \leq r \leq m$.
  As rules common for the simulations of all SUB-instructions, we have
  $$[\,]_{r^-} \rightarrow [\,]_{\#^-}, \; 1 \leq r \leq m,$$
  and the annihilation rules
  $$[\,]_r [\,]_{r^-} \rightarrow \lambda, \; 1 \leq r \leq m, \text{ and } [\,]_\# [\,]_{\#^-} \rightarrow \lambda$$

as well as the trap rules

$$[\,]_{\#^-} \to [\,]_{\#}[\,]_{\#} \text{ and } [\,]_{\#} \to [\,]_{\#}[\,]_{\#};$$

these last two rules lead the system into an infinite computation whenever a membrane with one of the trap symbols # or $\#^-$ is left without being annihilated.

The *zero test* for instruction l_1 is simulated by the rules

$$[\,]_{l_1} \to [\,]_{l_1'}\,[\,]_{r^-} \text{ and } [\,]_{l_1'} \to [\,]_{\#}[\,]_{l_3}.$$

The membrane labeled by #, generated by the second rule $[\,]_{l_1'} \to [\,]_{\#}[\,]_{l_3}$ can only be eliminated if the anti-membrane $[\,]_{r^-}$ generated by the first rule $[\,]_{l_1} \to [\,]_{l_1'}\,[\,]_{r^-}$ is not annihilated by $[\,]_r$, i.e., only if register r is empty, which allows for applying the rule $[\,]_{r^-} \to [\,]_{\#^-}$ and for using the annihilation rule $[\,]_{\#}[\,]_{\#^-} \to \lambda$ afterwards in the next derivation step.

The *decrement case* for instruction l_1 is simulated by the rule

$$[\,]_{l_1} \to [\,]_{l_2}[\,]_{r^-}.$$

The anti-membrane $[\,]_{r^-}$ either correctly annihilates one copy of membrane $[\,]_r$, thus decrementing the register r, or else traps an incorrect guess by forcing the anti-membrane $[\,]_{r^-}$ to evolve to $[\,]_{\#^-}$ and then to $[\,]_{\#}[\,]_{\#}$ in the next two steps in case register r is empty.

We finally observe that these two remaining derivation steps for trapping the decrement case as well as the remaining derivation step for correctly completing the decrement case or the zero test case do not influence the correct simulation of another SUB-instruction, even on the same register r, as the involved symbols have evolved at least one step before they could influence the symbols being generated by the new simulation sequence.

- $l_h : HALT$. Simulated by $[\,]_{l_h} \to \lambda$.

When the computation in M halts, the membrane $[\,]_{l_h}$ is removed, and no further rules can be applied provided the simulation has been carried out correctly, i.e., if no membranes labeled by trap symbols # are present in this situation. The remaining membranes in the system represent the result computed by M. □

For $\delta \in \{gen, acc\}$, let us denote the families of sets of non-negative integers and the sets of vectors of non-negative integers generated/accepted by PAMSs in normal form by N_δ(NFPAMS) and Ps_δ(NFPAMS), respectively.

Then, by combining Lemma 1 and Theorem 2, we obtain the following result:

Theorem 3. *For any $Y \in \{N, Ps\}$ and $\delta \in \{gen, acc\}$,*

$$Y_\delta(NFPAMS) = YRE.$$

Similar results obviously also hold for tissue P systems with anti-cells; the corresponding proofs again verbatim follow the proofs of Theorems 2 and 3, just replacing the notation $[\ ]_h$ by $\bigcirc_h$.

Corollary 3. *For any* $Y \in \{N, Ps\}$ *and* $\delta \in \{gen, acc\}$,

$$Y_\delta(PAMS) = YRE.$$

Corollary 4. *For any* $Y \in \{N, Ps\}$ *and* $\delta \in \{gen, acc\}$,

$$Y_\delta(NFPAMS) = YRE.$$

5.2 Derivation Modes

So far, we only have considered the maximally parallel derivation mode. Yet a thorough investigation of the proofs given so far in this section shows that in a successful derivation each rule need only be applied at most once, which means that instead of the maximally parallel derivation mode we can use any of the set derivation modes, where each rule can only be applied once, defined as follows:

$setmax$ take a non-extendable set of rules
$setmax_{rules}$ take a non-extendable set of rules with the maximal number of rules possible
$setmax_{objects}$ take a non-extendable set of rules affecting the maximal number of objects

The concept of using the maximal number of rules or objects can also be taken over for the maximally parallel derivation mode:

max take a non-extendable multiset of rules
max_{rules} take a non-extendable multiset of rules with the maximal number of rules possible
$max_{objects}$ take a non-extendable multiset of rules affecting the maximal number of objects

Let us now specify the derivation mode

$$\gamma \in \{max, max_{rules}, max_{objects}, setmax, setmax_{rules}, setmax_{objects}, \}$$

as additional subscript to δ, $\delta \in \{gen, acc\}$, for denoting the set of natural numbers and the set of vectors of natural numbers obtained by PAMS and tPAMS, i.e., we now write $N_{\gamma,\delta}$ and $Ps_{\gamma,\delta}$, respectively.

Moreover, we use the bracket notation [t]PAMS to indicate that we mean both PAMS and tPAMS, respectively, and in a similar way for PAMS and tPAMS in normal form.

With any of these derivation modes, using sets or multisets of rules, we now get the same normal form and computational completeness results as for the maximally parallel derivation mode max as established so far:

For the [t]PAMS to be transformed into normal form we observe that in the construction of the normal form given in the proof of Lemma 1, for each membrane label we used new additional labels and thus the corresponding new rules are independent from other such rules needed for simulating the change of a membrane label or the deletion of a membrane; hence, if in one of the set modes, one such rule is replaced to get the normal form, all the simulating rules are also needed only once, too, during the simulation sequences.

Hence, we can summarize the results obtained in this paper in the following form, for any of the derivation modes defined above.

We first state our normal form theorem:

Theorem 4. *For every [t]PAMS Π we can construct a [t]PAMS Π' in normal form such that*

$$Y_{\gamma,\delta}(\Pi) = Y_{\gamma,\delta}(\Pi')$$

for any $Y \in \{N, Ps\}$ and any $\delta \in \{gen, acc\}$ as well as any

$$\gamma \in \{max, max_{rules}, max_{objects}, setmax, setmax_{rules}, setmax_{objects}\}.$$

As our main result, we have shown computational completeness for PAMS and tPAMS, even in normal form, with all the derivation modes as defined above: we again emphasize that in the proofs given so far in this section, in a successful derivation each rule need only be applied at most once, hence, the simulations of the instructions of a register machine work for the set modes as well. On the other hand, whether the trap rule $[\ ]_{\#} \rightarrow [\ ]_{\#}[\ ]_{\#}$ is applied only once or as often as possible makes no difference for the desired effect to keep the system trapped in an infinite loop.

Theorem 5. *For any $Y \in \{N, Ps\}$ and $\delta \in \{gen, acc\}$,*

$$Y_{\gamma,\delta}([NF][t]PAMS) = YRE$$

for any

$$\gamma \in \{max, max_{rules}, max_{objects}, setmax, setmax_{rules}, setmax_{objects}\}.$$

Finally we mention that computational completeness can also be extended from the generating and accepting case to the computing case, i.e., PAMS and tPAMS, even in normal form, can also compute any partial recursive function or relation.

6 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 assume 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.

Moreover, we have established similar results for tissue P systems with cell division rules and cell / anti-cell annihilation rules. In both cases, as derivation modes we may also take the standard maximally parallel derivation mode(s) as well as any of the maximally parallel set derivation modes (non-extendable (multi)sets of rules, (multi)sets with maximal number of rules, (multi)sets of rules affecting the maximal number of objects) to obtain computational completeness.

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 $[\ ]_h \rightarrow [\ [\ ]_{h'}]_{h''}$. 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 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

1. Alhazov, A., Aman, B., Freund, R.: P systems with anti-matter. 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. 66–85. Springer (2014). https://doi.org/10.1007/978-3-319-14370-5_5
2. Alhazov, A., Aman, B., Freund, R., Păun, Gh.: Matter and anti-matter in membrane systems. In: Macías-Ramos, L.F., Martínez-del-Amor, M.A., Păun, Gh., Riscos-Núñez, A., Valencia-Cabrera, L. (eds.) Proceedings of the Twelfth Brainstorming Week on Membrane Computing. pp. 1–26 (2014), http://www.gcn.us.es/files/12bwmc/001_bwmc2014AntiMatter.pdf

3. Alhazov, A., Sburlan, D.: Static sorting P systems. In: Ciobanu, G., Pérez-Jiménez, M.J., Păun, Gh. (eds.) Applications of Membrane Computing, pp. 215–252. Natural Computing Series, Springer (2006). https://doi.org/10.1007/3-540-29937-8_8
4. Dassow, J., Păun, Gh.: Regulated Rewriting in Formal Language Theory. Springer (1989), https://www.springer.com/de/book/9783642749346
5. Díaz-Pernil, D., Peña-Cantillana, F., Alhazov, A., Freund, R., Gutiérrez-Naranjo, M.A.: Antimatter as a frontier of tractability in membrane computing. Fundamenta Informaticae **134**(1-2), 83–96 (2014). https://doi.org/10.3233/FI-2014-1092
6. Freund, R.: Purely catalytic P systems: Two catalysts can be sufficient for computational completeness. In: Alhazov, A., Cojocaru, S., Gheorghe, M., Rogozhin, Yu. (eds.) CMC14 Proceedings – The 14th International Conference on Membrane Computing, Chişinău, August 20–23, 2013. pp. 153–166. Institute of Mathematics and Computer Science, Academy of Sciences of Moldova (2013), http://www.math.md/cmc14/CMC14_Proceedings.pdf
7. Freund, R., Kari, L., Oswald, M., Sosík, P.: Computationally universal P systems without priorities: two catalysts are sufficient. Theoretical Computer Science **330**(2), 251–266 (2005). https://doi.org/10.1016/j.tcs.2004.06.029
8. Freund, R., Oswald, M.: Catalytic and purely catalytic P automata: control mechanisms for obtaining computational completeness. In: Bensch, S., Drewes, F., Freund, R., Otto, F. (eds.) Fifth Workshop on Non-Classical Models for Automata and Applications - NCMA 2013, Umeå, Sweden, August 13 - August 14, 2013, Proceedings. books@ocg.at, vol. 294, pp. 133–150. Österreichische Computer Gesellschaft (2013)
9. Freund, R., Păun, Gh.: How to obtain computational completeness in P systems with one catalyst. In: Neary, T., Cook, M. (eds.) Proceedings Machines, Computations and Universality 2013, MCU 2013, Zürich, Switzerland, September 9-11, 2013. EPTCS, vol. 128, pp. 47–61 (2013). https://doi.org/10.4204/EPTCS.128.13
10. Minsky, M.L.: Computation. Finite and Infinite Machines. Prentice Hall, Englewood Cliffs, NJ (1967)
11. Pan, L., Păun, Gh.: Spiking neural P systems with anti-matter. International Journal of Computers, Communications & Control **4**(3), 273–282 (2009). https://doi.org/10.15837/ijccc.2009.3.2435, http://univagora.ro/jour/index.php/ijccc/article/download/2435/901
12. Păun, Gh.: Computing with membranes. Journal of Computer and System Sciences **61**(1), 108–143 (2000). https://doi.org/10.1006/jcss.1999.1693
13. Păun, Gh.: Membrane Computing: An Introduction. Springer (2002). https://doi.org/10.1007/978-3-642-56196-2
14. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press (2010)
15. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages. Springer (1997). https://doi.org/10.1007/978-3-642-59136-5
16. The P Systems Website. http://ppage.psystems.eu/