Membrane Systems and Multiset Approximation: The Cases of Inner and Boundary Rule Application

Péter Battyányi and György Vaszil(✉)

Department of Computer Science, Faculty of Informatics, University of Debrecen, Kassai út 26, Debrecen 4028, Hungary
{attyanyi.peter,vaszil.gyorgy}@inf.unideb.hu

Abstract. We continue the study of generalized P systems with dynamically changing structure based on an associated multiset approximation framework. We consider membrane systems where the applicability of the multiset transformation rules is determined by the approximating multisets of the membrane regions. We consider two cases: First, we study systems with inner rules where we allow only rule applications such that the multisets involved in the rules are part of the lower approximation of the respective regions, then we consider systems with boundary rules where rule application is defined on the boundaries, that is, rules can only manipulate the elements outside of the lower approximation. We show that the second variant benefits from the underlying approximation framework by demonstrating an increase in its computational strength. On the other hand, by presenting an appropriate simulating Petri net, we show that the computational power of systems with inner rule application remains weaker than that of Turing machines (as long as the unsynchronized version is considered).

1 Introduction

Membrane systems, introduced in [15], are biologically inspired models of computation: their operation imitates in a sense the functioning of living cells. The computation proceeds in distinct regions, called membranes or compartments. The compartments allow computation with multisets: they accomplish transformations of their contained multisets by various evolution (multiset rewriting) rules. Several variants of P systems have been introduced and studied, see the monograph [16] for a thorough introduction, or the handbook [17] for a summary of notions and results of the area.

In the original symbol object model, the compartments are organized in a tree like structure. Each membrane except for the outermost one, the skin membrane, have a unique parent membrane, the parent-child relationship depicts the connection when one membrane (the parent) contains the other membrane (the child). The rules account for the distributed computational processes in the compartments. In this basic model, the lefthand side of a rule is a multiset of

T. Mihálydeák et al. (Eds.): IJCRS 2019, LNAI 11499, pp. 239–252, 2019.
https://doi.org/10.1007/978-3-030-22815-6_19

objects inside one of the regions and the righthand side of a rule is a multiset of objects labelled with target indications *here*, *out* and in_j indicating the positions the elements should be placed before the next computational step begins. Usually, computation in a region takes place in a maximally parallel manner, this means that a computational step in a region is understood as the simultaneous application of a multiset of rules which is maximal, that is, it cannot be augmented by any applicable rule. The membrane system waits for each of its compartments to finish its maximally parallel computational process, then the objects labelled with the target indications are moved to their correct places and a new computational step of the P system can begin. Objects with target indication *here* remain in the region, objects with *out* move to the parent region, objects with in_j enter the jth child region of the given compartment. The computation proceeds until no rule can be applied in any of the regions. The result is usually formed by the objects of a designated region, the output region, after the computation having come to a halt.

The structure of a membrane system can be represented in various ways, cell-like membrane systems have a membrane structure which can be described by a tree. Systems with graph-like membrane structures called tissue-like P systems were also considered, where the connection between the membranes are established by edges forming the communication routes. Here we study variants of tissue-like systems called generalized P systems (see [3]).

The question of how to define dynamically changing membrane structures using topological spaces, and how the underlying topologies influence the behaviour of P systems was already examined in [4,5]. Multiset approximation spaces were defined in [7,8], which made it possible to talk about lower and upper approximations of the contents of membranes of a P system. This led to various notions of membrane borders, and notions of closeness of membranes. Restricting the interaction to membranes that are close to each other, or permitting only rules that manipulate multisets which are on the boundaries of the membranes can affect the computational strength of the membrane system. The study of this area was initiated in [9], where also an intention to model chemical stability played an important role. The results in [9] were formulated for the so-called symport/antiport P systems, but the investigations were also continued for so called generalized P systems in [2]. In the present paper we also study generalized P systems, but we do not rely on any notion of closeness of membranes. Instead, we focus on the notion of clear observability. We consider lower approximations and boundaries of compartments, and restrict the applicability of the rules accordingly. It will turn out that the use of boundary rules, that is, rules which can only manipulate objects on the boundaries of compartments, results in an increase of the computational power of certain variants of generalized P systems to the level of the power of Turing machines. On the other hand, if we restrict rule applications only to rules that manipulate multisets which lie in the inner approximations of the membranes (inner rules), this restriction is not enough to provide Turing completeness.

The main contributions of the paper are of two kinds: the introduction of the above described variants of generalized P systems with associated multiset approximation spaces, and the presented results about their computational power.

In the following, we first recall the necessary definitions, then take up the examination of the two variants of generalized P systems with dynamically changing communication structure based on multiset approximation spaces. As maximal parallel rule application makes already the basic model of generalized P systems computationally complete, we study the weaker, unsynchronized variants. We first show that generalized P systems with inner rules can be simulated by simple place-transition Petri nets, thus, their computational power is less than that of Turing machines. Then we consider systems with boundary rules and show that they are able to simulate so called register machines, which demonstrates that their computational power is the same as the power of Turing machines. Finally, the paper ends with a few concluding remarks.

2 Preliminaries

Let $\mathbb{N}$ and $\mathbb{N}_{>0}$ be the set of non-negative integers and the set of positive integers, respectively, and let O be a finite nonempty set (the set of object). A *multiset* M over O is a pair $M = (O, f)$, where $f : O \to \mathbb{N}$ is a mapping which gives the *multiplicity* of each object $a \in O$. The set $\mathrm{supp}(M) = \{a \in O \mid f(a) > 0\}$ is called the *support* of M. If $\mathrm{supp}(M) = \emptyset$, then M is the empty multiset. If $a \in \mathrm{supp}(M)$, then $a \in M$, and $a \in^n M$ if $f(a) = n$.

Let $M_1 = (O, f_1), M_2 = (O, f_2)$. Then $(M_1 \sqcap M_2) = (O, f)$ where $f(a) = \min\{f_1(a), f_2(a)\}$; $(M_1 \sqcup M_2) = (O, f')$, where $f'(a) = \max\{f_1(a), f_2(a)\}$; $(M_1 \oplus M_2) = (O, f'')$, where $f''(a) = f_1(a) + f_2(a)$; $(M_1 \ominus M_2) = (O, f''')$ where $f'''(a) = \max\{f_1(a) - f_2(a), 0\}$; and $M_1 \sqsubseteq M_2$, if $f_1(a) \leq f_2(a)$ for all $a \in O$.

For any $n \in \mathbb{N}$, n-times addition of M, denoted by $\oplus_n M$, is given by the following inductive definition:

- $\oplus_0 M = \emptyset$;
- $\oplus_1 M = M$;
- $\oplus_{n+1} M = (\oplus_n M) \oplus M$.

Let $M_1 \neq \emptyset, M_2$ be two multisets. For any $n \in \mathbb{N}$, $M_1 \sqsubseteq^n M_2$, if $\oplus_n M_1 \sqsubseteq M_2$ but $\oplus_{n+1} M_1 \not\sqsubseteq M_2$.

The number of copies of objects in a finite multiset $M = (O, f)$ is its cardinality: $\mathrm{card}(M) = \Sigma_{a \in \mathrm{supp}(M)} f(a)$. Such an M can be represented by any string w over O for which $|w| = \mathrm{card}(M)$, and $|w|_a = f(a)$ where $|w|$ denotes the length of the string w, and $|w|_a$ denotes the number of occurrences of symbol a in w.

We define the $\mathcal{MS}^n(O)$, $n \in \mathbb{N}$, to be the set of all multisets $M = (O, f)$ over O such that $f(a) \leq n$ for all $a \in O$, and we let $\mathcal{MS}^{<\infty}(O) = \bigcup_{n \geq 0} \mathcal{MS}^n(O)$.

2.1 Generalized P Systems

Now we present the notion of generalized P systems, variants of tissue P systems introduced in [3].

An $n+3$-tuple $\Pi = (O, w_1, w_2, \ldots, w_n, R, i_o)$ is a *generalized P system* of degree $n \geq 1$, where

- O is a finite set of objects;
- $w_i \in \mathcal{MS}^{<\infty}(O)$, $1 \leq i \leq n$, is a finite multiset of objects, the initial contents of the ith region of Π;
- R is a finite set of transformation rules of the form $(x_1, \alpha_1) \ldots (x_k, \alpha_k) \rightarrow (y_1, \beta_1) \ldots (y_l, \beta_l)$, where $x_i, y_j \in \mathcal{MS}^{<\infty}(O)$, and $1 \leq \alpha_i, \beta_j \leq n$ indicate labels of the regions of the system for all $1 \leq i, j \leq n$;
- $1 \leq i_o \leq n$ is the label of output compartment.

The rules of a generalized P system can be considered to model interactions of objects simultaneously affecting several regions of the membrane system. Thus, the links between participating compartments are defined dynamically, through the applicability of the rules by the functioning of the system.

Given a generalized P system Π as above, a *configuration* of Π is an n-tuple $c = (u_1, u_2, \ldots, u_n)$ with $u_i \in MS^{<\infty}(O)$, $1 \leq i \leq n$, and $c_0 = (w_1, w_2, \ldots, w_n)$ is called its *initial configuration*. The multisets $u_1, u_2, \ldots, u_n$ are the *contents* of the corresponding compartments $1, 2, \ldots, n$, in configuration c.

A generalized P system changes its configurations by applying its rules. In the basic setting, a rule $r \in R$, is *applicable* to a configuration c, if and only if x_i is a submultiset of u_{α_i} for all $1 \leq i \leq k$. As a result of applying r to c, each multiset x_i is removed from the region u_{α_i}, $1 \leq i \leq k$, and each multiset y_j is added to the region u_{β_j}, $1 \leq j \leq l$.

The configuration $c' = (v_1, \ldots, v_n)$ of Π is obtained directly from the configuration $c = (u_1, \ldots, u_n)$ by applying the rules in the *unsynchronized* manner, if there is a multiset R' of rules from R, such that all of them are simultaneously applicable to different copies of objects in configuration c, and the configuration c' is the result of the application of the rules in R'. The configuration c' is obtained from c by applying the rules in the *maximally parallel* manner, if we add the additional requirement that the set R' is maximal, that is, for any $r \in R$, the rules in the rule multiset $\{r\} \oplus R'$ are not simultaneously applicable to c.

A sequence of configurations $c_0, c_1, \ldots$ of Π is called a *computation* if each configuration in the sequence is obtained directly from the previous one, starting from the initial configuration. Computations halt if no rule can be applied, the result of a *halting computation* is the number of objects that are present in the output compartment (compartment i_o) in the halting configuration.

2.2 Multiset Approximation Spaces

There are different ways of set approximations originating in rough set theory proposed in the early 1980's, [11,12]. The theory and its different generalizations uses different kinds of indiscernibility relations to provide lower and upper

approximations of sets. An indiscernibility relation on a given set of objects is given by a set of base sets by which lower and upper approximations can be constructed for any set. This way of set approximation was generalized to partial set approximation in [7], giving the possibility to embed available knowledge into an approximation space. The lower and upper approximations also rely on base sets which can be thought of as representants of the available knowledge. Having the concepts of lower and upper approximations, we can also introduce the concept of boundary as the difference between these two.

As membrane systems can be represented by multisets, in order to use the above described concepts in membrane systems theory, we need to generalize the set approximation framework for multisets. With the membrane structure as a background, an underlying multiset approximation space can be defined. The nature of this space is basically determined by its constituents, to a certain extent, independently of the membrane structure. The notion of multiset approximation spaces has been introduced in [7] (see also [8] for more details). Multiset approximations also rely on a set of base multisets given beforehand. By creating the lower and upper approximations using the usual approximation technique, the boundaries of multisets (boundaries of membrane regions) can also be defined, and we will make use of this feature in subsequent parts of the paper.

A multiset approximation space over a finite alphabet O consists of the following:

- A *domain*: in our case it is $\mathcal{MS}^{<\infty}(O)$, the set of finite multisets over some finite set O. The elements of the domain are approximated using the approximation space.
- A *base system*: $\mathfrak{B} \subseteq \mathcal{MS}^{<\infty}(O)$, a nonempty set of finite *base multisets* providing the basis for the approximation process.
- The *approximation functions*: $\mathsf{l}, \mathsf{u}, \mathsf{b} : \mathcal{MS}^{<\infty}(O) \to \mathcal{MS}^{<\infty}(O)$ determining the lower and upper approximations (and the boundaries) of multisets of the domain.

A *multiset approximation space* is a quintuple $(O, \mathfrak{B}, \mathsf{l}, \mathsf{u}, \mathsf{b})$ where O is a finite set, $\mathfrak{B} \subseteq \mathcal{MS}^{<\infty}(O)$ is a base system (a set of base multisets), and $\mathsf{b}, \mathsf{u}, \mathsf{l} : \mathcal{MS}^{<\infty}(O) \to \mathcal{MS}^{<\infty}(O)$ are the approximation functions generated by $\mathfrak{B}$.

For any multiset $M = (O, f) \in \mathcal{MS}^{<\infty}(O)$, we define the *lower approximation function*:

$$\mathsf{l}(M) = \bigsqcup\{\oplus_n B \mid B \in \mathfrak{B},\ B \sqsubseteq M, \text{and } B \sqsubseteq^n M\},$$

the *boundary function*:

$$\mathsf{b}(M) = \bigsqcup\{\oplus_n B \mid B \in \mathfrak{B},\ \text{and } B \sqcap (M \ominus \mathsf{l}(M)) \sqsubseteq^n M \ominus \mathsf{l}(M)\},$$

and the *upper approximation function*:

$$\mathsf{u}(M) = \mathsf{l}(M) \sqcup \mathsf{b}(M).$$

In addition, we also define $\mathsf{b}^{\mathsf{e}}(M) = \mathsf{b}(M) \ominus M$ as the *external part* of the boundary of M, and $\mathsf{b}^{\mathsf{i}}(M) = \mathsf{b}(M) \sqcap M$, the *internal part* of the boundary of M.

Intuitively, we can think of the lower approximation of the multiset M as the collection of elements that can be covered by the base multisets in such a way that the covering is inside M completely. If we also cover those elements of M that are left out of the lower approximation, then the union of the covering base sets contains M, thus, it can be thought of as the upper approximation of M, while the difference between the upper and the lower approximations of M is the boundary.

3 Regulating Rule Application in the Multiset Approximation Framework

In [2] we considered P systems with dynamical structure where the dynamic character of the membrane system was encoded in the reformulation of the region structure regarding a closeness property defined among the membranes based on the actual configuration of the system. Here we examine questions that arise when we require that in order for a rule to be applicable, the multisets on its lefthand side must conform to certain properties defined in the multiset approximation framework associated to the system. We discuss the following two approaches: first we require that a rule to be applied should only work with the lower approximations of the compartments' contents. The second approach demands that the multisets on the lefthand sides of the rules should come from the boundaries of the respective compartments.

Conforming the requirement of clear observability when dealing with rough sets, first we stipulate in the following definition that a rule should be applicable in a P system only if the multisets on its lefthand side come from the inner approximations of the containing regions, this means that we are absolutely sure that the rule application affects elements of the corresponding regions. The second requirement, on the other hand, corresponds to a system where rule application can only alter those elements about which our knowledge is vague, so the configuration changes of these systems might be thought of as steps in the direction of reducing vagueness, obtaining more and more determinate knowledge about the objects distributed in the membranes.

We formalize these notions in the following definition.

Definition 1. Let $\Pi = (O, \mathfrak{B}, w_1, w_2, \ldots, w_n, R, i_o)$ where $\mathfrak{B} \subseteq \mathcal{MS}^{<\infty}(O)$ is a base system and $(O, w_1, w_2, \ldots, w_n, R, i_o)$ is a generalized P system.

We call Π a *generalized P system with an associated multiset approximation space and inner rules*, if the applicability of a rule $r = (x_1, \alpha_1) \ldots (x_k, \alpha_k) \to (y_1, \beta_1) \ldots (y_l, \beta_l) \in R$ in a configuration $c = (u_1, \ldots, u_n)$ is defined by the requirement that x_i is a submultiset of $\mathsf{l}(u_{\alpha_i})$, the inner approximation of the respective region, $1 \leq i \leq k$. If $r \in R$ is applicable to c in this sense, then we call r an *inner rule* (with respect to c).

We call Π a *generalized P system with an associated multiset approximation space and boundary rules*, if the applicability of a rule $r = (x_1, \alpha_1) \dots (x_k, \alpha_k) \to (y_1, \beta_1) \dots (y_l, \beta_l) \in R$ in a configuration $c = (u_1, \dots, u_n)$ is defined by the requirement that x_i is a submultiset of $\mathsf{b}^{\mathsf{i}}(u_{\alpha_i})$, the internal part of the boundary of the respective region, $1 \leq i \leq k$. If $r \in R$ is applicable to c in this sense, then we call r a *boundary rule* (with respect to c).

Example 1. Assume that $C = (w_1, w_2)$ is a configuration of Π, a generalized P system with an associated multiset approximation space for $w_1 = a^3b^3c^2$ and base sets $B_1 = a^2$, $B_2 = bc$. Further, let $r_1 = (ab^2, 1) \to (c, 1)(d^3, 2)$ and $r_2 = (ab, 1) \to (e^2, 1)$ be to rules of Π.

If Π is a system with inner rules, then both rules are applicable in C, as $B_1 \sqcup \oplus_2 B_2 = a^2b^2c^2$ is the lower approximation of w_1.

If Π is a system with boundary rules, then only the rule r_2 is applicable in C, as a^2bc is the boundary of w_1 with inner part ab.

We claim that the use of inner rules do not add much to the computational strength of the P system in the sense that in the non-synchronized mode a generalized P system with an associated multiset approximation space and inner rules is not Turing complete. To show this, we construct a simple place-transition Petri net that simulates the P system in question. This is sufficient, because Petri nets in this simple setting are strictly weaker in computational power than Turing machines, see for example [13,14]. The idea of the proof is similar to that of Theorem 2 in [2], the construction of the Petri net, however, is different.

A *place-transition Petri net* [13] is a quintuple $U = (P, T, F, V, m_0)$ such that P, T are finite sets with $P \cap T = \emptyset$, $P \cup T \neq \emptyset$, the sets of *places* and *transitions*, respectively. The set $F \subseteq (P \times T) \cup (T \times P)$, is a set of "arcs" connecting places and transitions, the *flow relation* of U. The function $V : F \to \mathbb{N}_{>0}$ determines the multiplicity (the *weight*) of the arcs, and $m_0 : P \to \mathbb{N}$ is a function called the initial marking. In general, a *marking* is a function $m : P \to \mathbb{N}$ associating nonnegative integers (the number of *tokens*) to the places of the net. Moreover, for every transition $t \in T$, there is a place $p \in P$ such that $f = (p, t) \in F$ and $V(f) \neq 0$.

Let $x \in P \cup T$. The *pre- and postsets* of x, denoted by ${}^\bullet x$ and $x^\bullet$, respectively, are defined as ${}^\bullet x = \{y \mid (y, x) \in F\}$ and $x^\bullet = \{y \mid (x, y) \in F\}$.

For each transition $t \in T$, we define two markings, $t^-, t^+ : P \to \mathbb{N}$ as follows:

$$t^-(p) = \begin{cases} V(p,t), & \text{if } (p,t) \in F, \\ 0 & \text{otherwise,} \end{cases}$$

$$t^+(p) = \begin{cases} V(t,p), & \text{if } (t,p) \in F, \\ 0 & \text{otherwise.} \end{cases}$$

A transition $t \in T$ is said to be *enabled* if $t^-(p) \leq m(p)$ for all $p \in {}^\bullet t$. Let $\triangle\, t(p) = t^+(p) - t^-(p)$ for $p \in P$, and let us define the *firing of a transition* as follows. A transition $t \in T$ can fire in m (notation: $m \longrightarrow^t$) if t is enabled in m. After the firing of t, the Petri net obtains a new marking $m' : P \to \mathbb{N}$, where $m'(p) = m(p) + \triangle\, t(p)$ for all $p \in P$ (notation: $m \longrightarrow^t m'$).

Petri nets can be considered as computing devices: Starting with the initial marking, going through a series of configuration changes by the firing of a series of transitions, we might obtain a marking where no transitions are enabled. This final marking is the result of the Petri net computation.

Theorem 1. *For any generalized P system with an associated multiset approximation space and inner rules, Π, there is a place-transition Petri net N, such that N generates the same set of numbers as Π in the unsynchronized manner of rule application.*

Proof. Let $\Pi = (O, \mathfrak{B}, w_1^0, w_2^0, \ldots, w_n^0, R, i_o)$ be a generalized P system with an associated multiset approximation space and inner rules, let the underlying set of base sets be $\mathfrak{B} = \{B_i \mid 1 \leq i \leq m\}$, and let $x \in \mathcal{MS}^{<\infty}(O)$ be arbitrary. Then there exists an $h_x \in \mathbb{N}$ such that, for any subset $\{B_1, \ldots, B_s\} \subseteq \mathfrak{B}$, either $x \sqsubseteq \oplus_{h_x} B_1 \sqcup \ldots \sqcup \oplus_{h_x} B_s$ or x cannot be covered by the union of sums of $\{B_1, \ldots, B_s\}$ at all. In fact, if $x = a_1^{j_1} a_2^{j_2} \ldots a_t^{j_t}$, then it is enough to choose $h_x = max\{j_1, \ldots, j_t\}$.

Assume that $r = (x_1, \alpha_1) \ldots (x_k, \alpha_k) \to (y_1, \beta_1) \ldots (y_l, \beta_l) \in R$, and let $h_r = \max\{h_{x_1}, \ldots, h_{x_k}\}$ (which is a positive integer number). Let us denote with $\mathfrak{H}(r)$ the set of all tuples $H = (H_1, \ldots, H_k)$, such that $H_j = \oplus_{h_1^j} B_1^j \sqcup \ldots \sqcup \oplus_{h_{n_j}^j} B_{n_j}^j$ with $h_t^j \leq h_r$ and $x_j \sqsubseteq H_j$ $(1 \leq j \leq k)$. Since $x_i \sqsubseteq \mathsf{I}(u_{\alpha_i})$ if and only if there exists a $H_i \sqsubseteq u_{\alpha_i}$ such that $x_i \sqsubseteq H_i$, in order to check the applicability of r in a configuration $(u_{\alpha_1}, \ldots, u_{\alpha_n})$, it is enough to check whether there exists an $H \in \mathfrak{H}(r)$, such that $H_i \sqsubseteq u_{\alpha_i}$ for every element of H, $1 \leq i \leq k$. We construct the Petri net which makes sure that $x_i \sqsubseteq \mathsf{I}(u_{\alpha_i})$ and simulates the rule application at the same time.

Let us define the Petri net $N = (P, T, F, V, m_0)$ with $P = O \times \{1, \ldots, n\} \cup \{p_{ini}\}$. A place $(a, j) \in P$ represents the number of objects $a \in O$ inside the jth membrane at every step of the computational sequence, so let us set $m_0(p) = w_j^0(a)$ for every place $p = (a, j) \in O \times \{1, \ldots, n\}$, and let also $m_0(p_{ini}) = 1$.

The net N consists of subnets for each pair $(r, H) \in RH = \{(r, H) \mid r \in R, H \in \mathfrak{H}(r)\}$. These subnets are responsible for the simulation of the effect of r together with checking the condition that r is an inner rule. The place p_{ini} makes sure that only one of the subnets can operate at a time, hence the simulation of the rule executions are mutually exclusive.

Let $T = \{t_\delta \mid \delta \in RH\}$ with $\delta = (r, H_1, \ldots, H_k) \in RH$, and let r be denoted as $r = (x_1, \alpha_1) \ldots (x_k, \alpha_k) \to (y_1, \beta_1) \ldots (y_l, \beta_l)$. Then, for $1 \leq j \leq k$,

$$p = (a, \alpha_j) \in {}^\bullet t_\delta \cap t_\delta^\bullet \text{ if and only if } a \in H_j.$$

For $p = (a, \beta_q)$, $1 \leq q \leq l$, we have

$$p = (a, \beta_q) \in {}^\bullet t_\delta \cap t_\delta^\bullet \text{ if and only if } \beta_q = \alpha_j$$

for some $1 \leq j \leq k$ and $a \in H_j$. Otherwise, $p = (a, \beta_q) \in t_\delta^\bullet$. In addition, $p_{ini} \in {}^\bullet t_\delta \cap t_\delta^\bullet$.

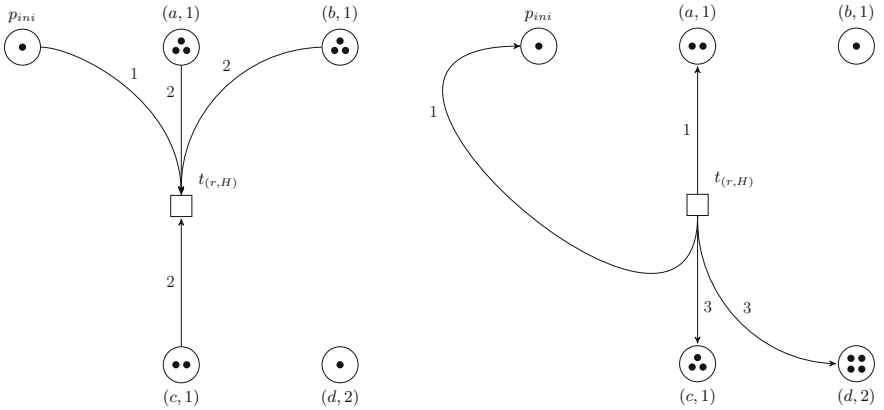

Fig. 1. Assume that $w_1^0 = a^3b^3c^2$, $w_2^0 = d$, $r = (ab^2, 1) \to (c, 1)(d^3, 2)$ and let $B_1 = a^2$, $B_2 = bc$ be base sets. Then, for $H = (H_1) \in \mathfrak{H}(r)$, $H_1 = B_1 \sqcup \oplus_2 B_2 = a^2b^2c^2$ are appropriate. The figure on the left shows the arcs pointing to transition $t_{(r,H)}$ together with their weights, the figure on the right shows the arcs going out from transition $t_{(r,H)}$ together with their weights. (The arcs with zero weight are not indicated explicitly.)

Let $p = (a, \alpha_j)$ with $1 \leq j \leq k$, and let $t_\delta \in T$, $\delta \in RH$, then the weights of the arcs are computed as $V(p, t_\delta) = H_j(a)$, that is, we check whether $H_j(a) \leq u_{\alpha_j}(a)$. Additionally, if $\alpha_j \neq \beta_q$ ($q \in \{1, \ldots, l\}$), then we have $V(t_\delta, p) = H_j(a) - x_{\alpha_j}(a)$, so that the necessary amount of tokens (those which correspond to the objects in $H_j \ominus x_j$) are returned to $p = (a, \alpha_j)$. This way the Petri net transition decreases the number of tokens in p only by $x_j(a)$. When $\alpha_j = \beta_q$ for some $q \in \{1, \ldots, l\}$, then $V(t_\delta, p) = H_j(a) - x_j(a) + y_q(a)$, that is, the righthand side of the rule returns further tokens to u_{α_j}.

For $p = (a, \beta_q)$, $1 \leq q \leq l$, if $\beta_q = \alpha_j$ for some $1 \leq j \leq k$, then the situation is as above. Otherwise, if $\beta_q \neq \alpha_j$ for any $1 \leq j \leq k$, then $V(p, t_\delta) = 0$ and $V(t_\delta, p) = y_{\beta_q}(a)$. Furthermore, $V(p_{ini}, t_\delta) = V(t_\delta, p_{ini}) = 1$.

To summarize the idea of the construction above, the places of the Petri net represent the objects in the different compartments of the P system. For every r, we are able to identify the union of the finite sums of the base sets that must be examined in order to decide whether the multisets $x_1, \ldots, x_k$ appearing on the lefthand side of the rule are in the inner approximation of u_{α_i}, that is, for every x_i we confine ourselves to $(B_1^i, \ldots, B_{k_i}^i) \in \mathfrak{H}(x_i)$, $1 \leq i \leq k$, such that $x_i \sqsubseteq \oplus_{h_1^i} B_1^i \sqcup \ldots \sqcup \oplus_{h_{k_i}^i} B_{k_i}^i$ and $h_t^j \leq h_r$, where $h_r \in \mathbb{N}$ is determined by r. Let $H = (H_1, \ldots, H_k) \in \mathfrak{H}(r)$ be a tuple of such multiset unions. To render r applicable and inner, we have to check whether $H_i \sqsubseteq u_{\alpha_i}$. For each pair $\delta = (r, H)$, where $r = (x_1, \alpha_1) \ldots (x_k, \alpha_k) \to (y_1, \beta_1) \ldots (y_l, \beta_l) \in R$ and $H = (H_1, \ldots, H_k) \in \mathfrak{H}(r)$ is a tuple of elements of $\mathfrak{B}$ with $x_i \sqsubseteq H_i$, we define a subnet consisting of all the places of N and a transition t_δ together with the corresponding arcs. This subnet simulates an application of r while the conditions on H ensure that r is an inner rule. The whole process is controlled

by the place p_{ini}. Each of the subnets is connected with p_{ini} in such a manner that only one subnet is able to operate at a time. Figure 1 illustrates the structure of one such a subnet that constitutes the whole Petri net N.

The Petri net halts if and only if the membrane system halts, and the number of objects in the output membrane are indicated by the number of tokens in the corresponding places.

As we have already mentioned, the expressive power of place-transition Petri nets are less than that of Turing machines, so we obtain the following corollary.

Corollary 1. *Generalized membrane systems with multiset approximation spaces and inner rules using the unsynchronized manner of rule application are strictly weaker in computational power than Turing machines, that is, they are not computationally complete.*

Now we continue with the investigation of the case of boundary rules. We show that generalized P systems with boundary rules generate any recursively enumerable set of numbers. We do this by demonstrating how these systems simulate the computations of register machines, a computational model equivalent in power to Turing machines.

A *register machine* is a construct $W = (m, H, l_0, l_h, \text{Inst})$, where m is the number of registers, H is the set of instruction labels, l_0 is the start label, l_h is the halting label, and Inst is the set of instructions. Each label from H labels only one instruction from Inst. There are several types of instructions which can be used. For $l_i, l_j, l_k \in H$ and $r \in \{1, \ldots, m\}$ we have:

- $l_i : (\texttt{nADD}(r), l_j, l_k)$ - *nondeterministic add*: Add 1 to register r and then go to one of the instructions with labels l_j or l_k, nondeterministically chosen.
- $l_i : (\texttt{ADD}(r), l_j)$ - *deterministic add*: Add 1 to register r and then go to the instruction with label l_j.
- $l_i : (\texttt{CHECKSUB}(r), l_j, l_k)$ - *zero check and subtract*: If register r is empty, then go to the instruction with label l_j, if r is non-empty, then subtract one from it and go to the instruction with label l_k.
- $l_h : \texttt{HALT}$ - *halt*: Stop the machine.

Register machines compute sets of numbers by starting their computation with empty registers and proceeding by applying instructions in the order indicated by the labels, beginning with the instruction l_0. If the machine reaches the halt instruction $l_h : \texttt{HALT}$, then its work is finished, and the number stored in the first register is said to be the result of the computation. Note that the computed sets of numbers can be infinite, due to the nondeterminism in choosing the continuation of the computation in the case of nondeterministic add instructions, $l_i : (\texttt{nADD}(r), l_j, l_k)$.

We would like to add here, that register machines can also be defined as deterministic computing devices (without the nondeterministic add instructions). In this case they compute functions of input values placed initially in (some of) the registers. They are able to compute all functions which are Turing computable

(see, for example [10]), if they have at least two registers. By providing the machine with the nondeterministic add instruction, as above, we obtain a device which generates sets of numbers starting from a unique initial configuration. Since any recursively enumerable set can be obtained as the range of a Turing computable function on the set of non-negative integers, register machines defined this way are able to generate any recursively enumerable set of numbers.

Theorem 2. *Generalized P systems with associated multiset approximation spaces and boundary rules generate any recursively enumerable set of numbers, even in the unsynchronized manner of rule application.*

Proof. Let L be a recursively enumerable set of numbers, and consider the register machine $W = (m, H, l_0, l_h, \text{Inst})$ generating L. We construct a generalized P system with an associated multiset approximation space and boundary rules, such that it also generates L in the sense that the generated numbers correspond to the multiplicity of a certain object in the output region when the computation halts. Let $\Pi = (O, \mathfrak{B}, w_1, w_2, w_3, R, 2)$ with

$$\begin{aligned} O &= \{l, l' \mid l \in H\} \cup \{a_r, a'_r \mid 1 \le r \le m\} \cup \{b_1, b_2, c\}, \\ \mathfrak{B} &= \{a_r a'_r \mid 1 \le r \le m\} \cup \{l_h b_1 \mid l_h : \texttt{HALT}\} \cup \{b_2 c, lc \mid l \in H\}, \\ w_1 &= l_0 b_1,\ w_2 = \emptyset,\ w_3 = \emptyset, \\ R &= R_{Add} \cup R_{CheckSub} \cup R_{Ex}, \end{aligned}$$

where

$$\begin{aligned} R_{Add} = \{&(l_i, 1) \to (l_j, 1)(a_r, 2), (l_i, 1) \to (l_k, 1)(a_r, 2) \mid \text{for all} \\ &l_i : (\texttt{nADD}(r), l_j, l_k) \in \text{Inst}\} \cup \\ &\{(l_i, 1) \to (l_j, 1)(a_r, 2) \mid \text{for all } l_i : (\texttt{ADD}(r), l_j) \in \text{Inst}\}, \end{aligned}$$

$$\begin{aligned} R_{CheckSub} = \{&(l_i, 1) \to (l'_j, 1)(a'_r, 2), (l'_j, 1)(a'_r, 2) \to (l_j, 1), \\ &(l_i, 1)(a_r, 2) \to (l_k, 1) \mid \text{for all } l_i : (\texttt{CHECKSUB}(r), l_j, l_k) \in \text{Inst}\}, \end{aligned}$$

$$R_{Ex} = \{(b_1, 1) \to (b_2, 3), (b_2, 3) \to (b_1, 1)\}.$$

To see how Π simulates the computations of W, consider its initial configuration $(l_0 b_1, \emptyset, \emptyset)$: it corresponds to the initial configuration of W, as the first region contains l_0, the label of the instruction that has to be executed next, and the number of occurrences of a_r, $1 \le r \le m$, in the second region are 0, corresponding to the fact that all registers are initially empty.

Notice that as long as l_h is not present, it is possible to exchange b_1 in the first region with b_2 in the third (and back), since both symbols are in the boundary of the respective regions, so one of the rules of R_{Ex} is always applicable. When l_h

appears in the first region, then after b_1 also appears there, they are "removed" from the boundary of the region (as $l_h b_1 \in \mathfrak{B}$ is a base multiset of the multiset approximation space), and after this happens, no rule of R is applicable. From these considerations we can see that Π reaches a halting configuration only if the label of the halting instruction, l_h appears.

Let us consider the case when the generalized P system Π is in the configuration $(l_i\delta_1 w_{1,in}, ww_{2,in}, \delta_2)$ with $w(a_r) = k_r,\ 1 \leq r \leq m$, corresponding to a situation when W is going to execute instruction l_i, and the contents of register r is $k_r \geq 0,\ 1 \leq r \leq m$. The symbols δ_1, δ_2 are used to denote either b_1 or b_2, their exact meaning is not important, as they do not interfere with the simulation process until l_h appears. The submultisets $w_{1,in}$ and $w_{2,in}$ denote those elements of the first two regions that are not on the region boundary. By looking at the rules of Π, we might notice that as long as the object l_h is not present, all elements of the first region are on the boundary, thus, we might omit the submultiset $w_{1,in}$ from the above notation, having the configuration $(l_i\delta_1, ww_{2,in}, \delta_2)$. Note also, that $w_{2,in} = (a_r a'_r)^i$ for some $i \geq 0$.

If l_i is the label of an add, or nondeterministic add instruction, then the rule simulating the instruction $l_i : (\texttt{nADD}(r), l_j, l_k)$ is applicable, yielding the configuration $(l'\delta_1, wa_r w_{2,in}, \delta_2)$ with $l' \in \{l_j, l_k\}$ (or the configuration $(l_j\delta_1, wa_r, \delta_2)$ if $l_i : (\texttt{ADD}(r), l_j)$ is simulated). In any of these cases, we get a configuration $(l\delta_1, wa_r, \delta_2)$ where l corresponds to the instruction that has to be executed next, while the second region contains one more object a_r, that is, the number stored in register r was incremented, as required by the simulated add instructions.

Suppose now, that Π is in a configuration $(l_i\delta_1, ww_{2,in}, \delta_2)$ and the instruction to be executed is $l_i : (\texttt{CHECKSUB}(r), l_j, l_k)$. By applying the rules in $R_{CheckSub}$ we might obtain $(l'_j\delta_1, wa'_r w_{2,in}, \delta_2)$. If $w(a_r) = 0$, we get $(l_j\delta_1, ww_{2,in}, \delta_2)$ after the next step, or if $w(a_r) > 0$, then as $a_r a'_r \in \mathfrak{B}$ is a base multiset, one copy of a_r and a'_r is removed from the boundary, so we have $(l'_j\delta_1, w'w'_{2,in}, \delta_2)$ where $a_r w' = w$ and $w'_{2,in} = a_r a'_r w_{2,in}$. In this case l'_j cannot be changed any more, and due to the rules in R_{Ex}, the computation can never reach a halting configuration. On the other hand, if $w(a_r) > 0$, then applying the rule $(l_i, 1)(a_r, 2) \to (l_k, 1)$, we get $(l_k\delta_1, w'w_{2,in}, \delta_2)$ with $w'a_r = w$, thus the checking and subtracting instruction of W is correctly simulated by the system Π.

The simulation is finished when the object l_h appears in the first region. The only rules that are applicable are the rules of R_{Ex}, but when b_1 also appears in the first region, the computation halts, because $l_h b_1$ is a base multiset, so all these objects disappear from the region boundary.

After halting, the result of the computation is the number of a_1 objects in the second region, as they correspond to the contents of the first register (the output register) of the register machine W.

4 Concluding Remarks

We have used multiset approximation spaces to restrict the applicability of multiset evolution rules of generalized P systems. This way we incorporated some additional "dynamics" into the system, as not only the presence or absence of elements, but also the underlying approximation spaces have a role in determining the applicability of the rules.

It turned out that restricting the operation of the rules to the boundaries of compartments increases the computational power of generalized P systems, as they are able to generate any recursively enumerable sets of numbers even in the unsynchronized manner of rule application. We have shown this by demonstrating that they are able to simulate register machines, a computational model equivalent in power to the model of Turing machines. On the other hand, a similar restriction allowing the rules to manipulate only elements of the lower approximation of the compartments of the system does not result in a similar increase of the computational power, as the resulting systems can be simulated by simple place-transition Petri nets, a model which is known to be weaker in computational power than the model of Turing machines.

As a final remark, we would like to add some thoughts on a related model called P systems with anti-matter [1,6]. In P systems with anti-matter, objects have complementary "anti objects", and when they are both present, they annihilate (disappear). In this paper we considered boundary rules which cannot be applied to objects that are not on the boundary: when all the elements of a base multiset are present in a region, they "disappear" from the scope of boundary rules. This effect is similar to the effect of annihilation rules, although not exactly the same. The difference can be seen from a simple example: let two base multisets be $ab, ac \in \mathfrak{B}$. The fact that they form base multisets is not directly modeled by the annihilation rules $ab \to \varepsilon, bc \to \varepsilon$ (as used in the case of P systems with anti-matter), because of the following. If a region contains ab, then these are "invisible" for the boundary rules, but they are not annihilated, as can be seen when an object c enters the region. As bc is also a base multiset, c immediately "disappears" by becoming part of the inner, lower approximation part of the region contents. As we see, the relationship of boundary rules and anti-matter is not as simple as it might look, but it definitely seems to be an interesting topic for further investigations.

Acknowledgments. G. Vaszil was supported by grant K 120558 of the National Research, Development and Innovation Office of Hungary (NKFIH), financed under the K 16 funding scheme. The work is also supported by the EFOP-3.6.1-16-2016-00022 project, co-financed by the European Union and the European Social Fund.

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.) CMC 2014. LNCS, vol. 8961, pp. 66–85. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-14370-5_5

2. Battyányi, P., Mihálydeák, T., Vaszil, G.: Generalized membrane systems with dynamical structure, Petri nets, and multiset approximation spaces. In: 18th International Conference on Unconventional Computation and Natural Computation. UCNC (2019, Accepted)
3. Bernardini, F., Gheorgue, M., Margenstern, M., Verlan, S.: Networks of cells and Petri nets. In: Díaz-Pernil, D., Graciani, C., Gutiérrez-Naranjo, M.A., Păun, G., Pérez-Hurtado, I., Riscos-Núñez, A. (eds.) Proceedings of the Fifth Brainstorming Week on Membrane Computing, pp. 33–62. Fénix Editora, Sevilla (2007)
4. Csuhaj-Varjú, E., Gheorghe, M., Stannett, M.: P systems controlled by general topologies. In: Durand-Lose, J., Jonoska, N. (eds.) UCNC 2012. LNCS, vol. 7445, pp. 70–81. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32894-7_8
5. Csuhaj-Varjú, E., Gheorghe, M., Stannett, M., Vaszil, G.: Spatially localised membrane systems. Fundamenta Informaticae **138**(1–2), 193–205 (2015)
6. Leporati, A., Manzoni, L., Mauri, G., Porreca, A.E., Zandron, C.: The counting power of P systems with antimatter. Theoret. Comput. Sci. **701**, 161–173 (2017)
7. Mihálydeák, T., Csajbók, Z.E.: Membranes with boundaries. In: Csuhaj-Varjú, E., Gheorghe, M., Rozenberg, G., Salomaa, A., Vaszil, G. (eds.) CMC 2012. LNCS, vol. 7762, pp. 277–294. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36751-9_19
8. Mihálydeák, T., Csajbók, Z.E.: On the membrane computations in the presence of membrane boundaries. J. Automata Lang. Comb. **19**(1), 227–238 (2014)
9. Mihálydeák, T., Vaszil, G.: Regulating rule application with membrane boundaries in P systems. In: Rozenberg, G., Salomaa, A., Sempere, J.M., Zandron, C. (eds.) CMC 2015. LNCS, vol. 9504, pp. 304–320. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-28475-0_21
10. Minsky, M.L.: Computation: Finite and Infinite Machines. Prentice-Hall Inc., Upper Saddle River (1967)
11. Pawlak, Z.: Rough sets. Int. J. Comput. Inf. Sci. **11**(5), 341–356 (1982)
12. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
13. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. Prentice Hall PTR, Upper Saddle River (1981)
14. Popova-Zeugmann, L.: Time and Petri Nets. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-41115-1
15. Păun, G.: Computing with membranes. J. Comput. Syst. Sci. **61**(1), 108–143 (2000)
16. Păun, G.: Membrane Computing: An Introduction. Natural Computing Series, 1st edn. Springer, Heidelberg (2002). https://doi.org/10.1007/978-3-642-56196-2
17. Păun, G., Rozenberg, G., Salomaa, A.: The Oxford Handbook of Membrane Computing. Oxford University Press Inc., New York (2010)