Annales Mathematicae et Informaticae
44 (2015) pp. 197–210
http://ami.ektf.hu

Boundaries of membrane in P systems relying on multiset approximation spaces in language R

Péter Takács[a], Zoltán Ernő Csajbók[a], Tamás Mihálydeák[b]

[a]Department of Health Informatics, Faculty of Health, University of Debrecen
{takacs.peter,csajbok.zoltan}@foh.unideb.hu

[b]Department of Computer Science, Faculty of Informatics, University of Debrecen
mihalydeak.tamas@inf.unideb.hu

Submitted September 11, 2014 — Accepted May 13, 2015

Abstract

Membrane computing is an area within computer science which aims to develop a new computational model through the study of the characteristics of biological cells. It is a distributed and parallel computing model. Communication between regions through membranes, as well as membrane system and its environment, plays an important role in the process. Combination of P system with multiset approximation space leads to the abstract concept of 'to be close enough to a membrane'. The designated goal is to perform calculations in this two-fold system by the help of language R. Some packages can perform calculations with multisets in R (such as 'sets' package), but they are more closely linked to fuzzy systems. In this paper a new program library in language R is initiated which had been created to encourage some fundamental calculations in membrane systems combined with multiset approximation spaces. Data structures and functions are illustrated by examples.

Keywords: multiset approximation spaces, membrane computing, R language

MSC: 68U20

1. Introduction

The classical set theory does not enable multiply occurrences of the same objects. However, the multiset theory provides the opportunity to do so [2, 14]. Membrane computing also works with multisets [8, 9, 10, 11]. In [4, 5], the authors developed an abstract concept of the 'to be close enough to a membrane'. They used a generalization of classical Pawlakian rough set theory for multisets which is called the Pawlakian multiset approximation space (MAS). One part of the work is to carry out related calculations in MAS quickly and accurately.

There are several packages in language R (e.g., package 'sets' [15]), which allows multiset calculations. However, by the use of these applications, it is difficult to perform MAS calculations. The goal of this paper is to present R functions which facilitate in a quick and easy manner to perform all the most important calculations in membrane system combined with MAS. In Section 2, some initial relations and operations for multisets with illustrative R examples are described. Then, in Section 3 and 4, R functions for Pawlakian multiset approximation spaces are introduced in order to apply them to membrane computing.[1]

2. Multiset functions in R

2.1. Multisets

Let U be a finite nonempty set called the universe. A *multiset* (or *mset*) M over U is a mapping $M : U \to \mathbb{N} \cup \{\infty\}$ ($\mathbb{N}$ is the set of natural numbers). For instance, if $a \in U$ and M is a multiset with three occurrences of a, then $M(a) = 3$. This fact is often referred to as a^3. In general, if more than one element are repeated in a multiset, it is usually expressed in power form. For example, $M = a^3b^2$ is a multiset with three a's and two b's. M is *empty multiset*, denoted by $\emptyset$, if $M(a) = 0$ $(a \in U)$.

Let $\mathcal{MS}(U)$ denote the set of all multisets over U. $M \in \mathcal{MS}(U)$ is *finite*, if $M(a) < \infty$ $(a \in U)$. A *macroset* $\mathcal{M}$ is a set of finite multisets [3]. In the framework the following two fundamental macrosets are used:

- $\mathcal{MS}^n(U)$ $(n \in \mathbb{N})$ is the set of all multisets M such that $M(a) \leq n$ $(a \in U)$;
- $\mathcal{MS}^{<\infty}(U) = \bigcup_{n=0}^{\infty} \mathcal{MS}^n(U)$.

Let us note that $\mathcal{MS}^0(U) = \emptyset$ and $\mathcal{MS}^n(U) \subsetneqq \mathcal{MS}^{n+1}(U)$ $(n = 0, 1, 2, \dots)$.

Some basic R functions over $\mathcal{MS}^{<\infty}(U)$ have been developed to calculate the multiset relations and operations. Let M, M_1, $M_2 \in \mathcal{MS}^{<\infty}(U)$.

Set-theoretical relations for multisets implemented in R are the following:

- *Multiplicity relation*: $a \in M$ $(a \in U)$ if $M(a) \geq 1$.

[1] There is no room here to describe R functions in code level. We send it to everyone who is interested in.

- *Equality relation* is: $M_1 = M_2$ if $M_1(a) = M_2(a)$ $(a \in U)$.
- *Inclusion relation*: $M_1 \sqsubseteq M_2$ if $M_1(a) \leq M_2(a)$ $(a \in U)$.

Set-theoretical operations for multisets implemented in R are the following:

- *Set-type union*: $(M_1 \sqcup M_2)(a) = \mathsf{max}\{M_1(a), M_2(a)\}$ $(a \in U)$.
- *Intersection*: $(M_1 \sqcap M_2)(a) = \mathsf{min}\{M_1(a), M_2(a)\}$ $(a \in U)$.
- *Multiset addition*: $(M_1 \oplus M_2)(a) = M_1(a) + M_2(a)$ $(a \in U)$.
- *n-times addition* $(n \in \mathbb{N})$: it is given by the following inductive definition:
  1. $\oplus_0 M = \emptyset$
  2. $\oplus_1 M = M$
  3. $\oplus_n M = \oplus_{n-1} M \oplus M$ $(n > 1)$.
- *Multiset subtraction*: $(M_1 \ominus M_2)(a) = \mathsf{max}\{M_1(a) - M_2(a), 0\}$ $(a \in U)$.

By the help of n-times addition, a new multiset relation can be defined:

- *n-times inclusion relation* $(n \in \mathbb{N})$: Let $M_1 \neq \emptyset$. $M_1 \sqsubseteq^n M_2$ if $\oplus_n M_1 \sqsubseteq M_2$ but $\oplus_{n+1} M_1 \not\sqsubseteq M_2$.

2.2. Implementation of multiset relations

Throughout our implementation, it is assumed that the universe U is finite, fixed and its elements are totally ordered.

Implemented R functions are demonstrated by the help of a running example. To this end, first, let $U = \{a, b, c, d, e\}$ with the natural English alphabet ordering. It will be given as the fixed universe with the following command:

```
> U <- c("a","b","c","d","e").
```

Remark 2.1. Here and later on, '>' denotes the R prompt. `c()` is the concatenation function in R. Therefore, the universe U in R can be viewed as the string "*abcde*".

Each multiset is described in Parikh vector representation form. This means that the elements which are not actually included in the multiset are indicated by zero exponent. For instance, let us take the multiset $M = ce^5$. Its R representation in Parikh vector form is $a^0b^0c^1d^0e^5$, whereas its R realization is:

```
> M <- c(0,0,1,0,5).
```

Remark 2.2. M in R can be viewed as the string "00105".

Turning to the implementation of multiset relations, the first function is a technical one. It is a verification function which is called for every relation and operation.

mcheck(mS, SU)
Parameters: multiset **mS**; universe **SU**.

Description: This function checks the number of elements of multiset **mS**. If the cardinality of distinct elements in **mS** is equal to the cardinality of **SU**, the function returns 1, otherwise it returns 0.

The next three functions realize the set-theoretical relations for multisets.

min(mS, o, SU) – Multiplicity relation
Parameters: multiset **mS**, object $\mathbf{o} \in \mathbf{SU}$; universe **SU**.
Description: This function checks the membership of **o** in **mS**. It returns 1 if $\mathbf{mS}(\mathbf{o}) \geq 1$, otherwise it returns 0.

Example 2.3. Multiplicity relation

`> M <- c(1,2,3,0,0)`	$M = ab^2c^3$
`> min(M,"a",U)`	
`[1] 1`	$a \in M$
`> min(M,"e",U)`	
`[1] 0`	$e \notin M$

Remark 2.4. Result of R command, if any, is located in its underlying row beginning with '[1]' sign. '|' is a selector line which separates the mathematical formulae (the second column) from their implementations in R (the first column).

mequal(mS1, mS2, SU) – Equality relation
Parameters: multisets **mS1**, **mS2**; universe **SU**.
Description: This function checks the equality relation of two multisets **mS1**, **mS2**. It returns 1 if **mS1** and **mS2** are equal, otherwise it returns 0.

Example 2.5. Equality relation

`> M1 <- c(0,0,1,0,5)`	$M_1 = ce^5$
`> M2 <- c(3,2,0,1,0)`	$M_2 = a^3b^2d$
`> mequal(M1,M1,U)`	
`[1] 1`	$M_1 = ce^5 = ce^5 = M_1$
`> mequal(M1,M2,U)`	
`[1] 0`	$M_1 = ce^5 \neq a^3b^2d = M_2$

mpartof(mS1, mS2, SU) – Inclusion relation
Parameters: multisets **mS1**, **mS2**; universe **SU**.
Description: This function checks whether **mS1** is included in **mS2** or not. It returns 1 if the multiset **mS1** is part of the multiset **mS2**, otherwise it returns 0.

Example 2.6. Inclusion relation

`> M3 <- c(0,0,2,1,5)`	$M_3 = c^2de^5$
`> mpartof(M1,M3,U)`	
`[1] 1`	$M_1 = ce^5 \sqsubseteq c^2de^5 = M_3$
`> mpartof(M1,M2,U)`	
`[1] 0`	$M_1 = ce^5 \not\sqsubseteq a^3b^2d = M_2$

2.3. Implementation of basic multiset operations

In demonstration examples, these multisets will be used in the following:

```
> W1 <- c(0,0,1,0,5)
> W2 <- c(3,2,0,1,0)
> W3 <- c(1,2,3,4,0)
> W4 <- c(3,1,0,0,0)
> W5 <- c(1,1,2,3,0)
```

$W_1 = ce^5$
$W_2 = a^3b^2d$
$W_3 = ab^2c^3d^4$
$W_4 = a^3b$
$W_5 = abc^2d^3$

munion(mS1, mS2, SU) – Set-type union
Parameters: multisets **mS1**, **mS2**; universe **SU**.
Description: This function computes the set-type union of multisets **mS1**, **mS2**.

Example 2.7. Set-type union

```
> munion(W4,W5,U)
[1] 3 1 2 3 0
```

$W_4 \sqcup W_5 = a^3bc^2d^3$

mintersec(mS1, mS2, SU) – Intersection
Parameters: multisets **mS1**, **mS2**; universe **SU**.
Description: This function computes the intersection of multisets **mS1**, **mS2**.

Example 2.8. Intersection

```
> mintersec(W2,W3,U)
[1] 1 2 0 1 0
```

$W_2 \sqcap W_3 = ab^2d$

madd(mS1, mS2, SU) – Multiset addition
Parameters: multisets **mS1**, **mS2**; universe **SU**.
Description: This function computes the multiset addition of multisets **mS1**, **mS2**.

Example 2.9. Multiset addition

```
> madd(W4,W5,U)
[1] 4 2 2 3 0
```

$W_4 \oplus W_5 = a^4b^2c^2d^3$

mnadd(mS, n, SU) – n-times addition
Parameters: multiset **mS**, $\mathbf{n} \in \mathbb{N}$; universe **SU**.
Description: This function computes n-times addition of multiset **mS**.

Example 2.10. n-times addition

```
> mnadd(W1,0,U)
[1] 0 0 0 0 0
> mnadd(W1,1,U)
[1] 0 0 1 0 5
> mnadd(W1,3,U)
[1] 0 0 3 0 15
```

$\oplus_0 W_1 = \emptyset$
$\oplus_1 W_1 = ce^5$
$\oplus_3 W_1 = c^3e^{15}$

mdiff(mS1, mS2, SU) – Multiset subtraction
Parameters: multisets **mS1**, **mS2**; universe **SU**.
Description: This function computes the multiset subtraction of multisets **mS1**, **mS2**.

Example 2.11. Multiset subtraction

```
> mdiff(W3,W2,U)
[1] 0 0 3 3 0
```

$W_3 \ominus W_2 = c^3d^3$

mnpartof(mS1, mS2, SU) n-times inclusion relation
Parameters: multisets **mS1**$(\neq \emptyset)$, **mS2**; universe **SU**.
Description: This function determines how many times **mS1** is included in **mS2**. It returns $n(\in \mathbb{N})$ if $\oplus_n$**mS1** $\sqsubseteq$ **mS2** but $\oplus_{n+1}$**mS1** $\not\sqsubseteq$ **mS2**.

Example 2.12. n-times inclusion relation

```
> M  <- c(0,0,0,0,0)
> M1 <- c(0,0,1,3,0)
> M2 <- c(0,0,1,3,1)
> M3 <- c(1,0,3,11,1)
> mnpartof(M3,M,U)
[1] 0
> mnpartof(M1,M2,U)
[1] 1
> mnpartof(M1,M3,U)
[1] 3
```

$M = \emptyset$
$M_1 = cd^3$
$M_2 = cd^3e$
$M_3 = ac^3d^{11}e$

$M_3 \sqsubseteq^0 M = \emptyset$

$M_1 \sqsubseteq^1 M_2$ (i.e., $M_1 \sqsubseteq M_2$)

$M_1 \sqsubseteq^3 M_3$

3. Calculation in Pawlakian multiset approximation spaces

3.1. Pawlakian multiset approximation spaces

To define the abstract notion of boundaries in membrane systems, rough set theory (RST) should be a plausible opportunity [12, 13]. However, RST works within the traditional set theory, while regions in membrane systems are represented by multisets. Thus, to be able to apply the notions of RST, first, we have to generalize them for multisets.

Such a generalized multiset approximation space has four basic components:

- *Domain*: a set of multisets whose members are approximated.
- *Base system*: a set of some distinguished multisets (called *base multisets*) of the domain as the basis of approximations. Members of the base system are primary tools of the approximation process. Definable sets describe how they are combined, whereas approximation primitives give an account of how they are utilized in this process.

- *Definable multisets*: a set of multisets. They are
  - derived from base multisets (all base multisets are definable);
  - candidates for possible approximations and boundaries of the members of the domain.
- *Approximation primitives*: they determine lower/upper approximations and boundaries of the domain members using definable multisets.

Let U be a nonempty set. The 6-tuple $\mathsf{MAS}(U) = \langle \mathcal{MS}^{<\infty}(U), \mathfrak{B}, \mathfrak{D}_{\mathfrak{B}}, \mathsf{l}, \mathsf{b}, \mathsf{u} \rangle$ is a *multiset approximation space* if

- (*domain*) $\mathcal{MS}^{<\infty}(U) \subseteq \mathcal{MS}(U)$;
- (*base system*) $\mathfrak{B}(\neq \emptyset) \subseteq \mathcal{MS}^{<\infty}(U)$ and if $B \in \mathfrak{B}$, then $B \neq \emptyset$;
- (*definable multisets*) $\mathfrak{B} \subseteq \mathfrak{D}_{\mathfrak{B}}$; $\emptyset \in \mathfrak{D}_{\mathfrak{B}}$; if $B \in \mathfrak{B}$, $\oplus_n B \in \mathfrak{D}_{\mathfrak{B}}$ $(n = 1, 2, \dots)$;
- (*approximation primitives*) functions $\mathsf{l}, \mathsf{b}, \mathsf{u} : \mathcal{MS}^{<\infty}(U) \to \mathcal{MS}^{<\infty}(U)$ meet the following requirements:
  - (i) $\mathsf{l}(\mathcal{MS}^{<\infty}(U)), \mathsf{b}(\mathcal{MS}^{<\infty}(U)), \mathsf{u}(\mathcal{MS}^{<\infty}(U)) \subseteq \mathfrak{D}_{\mathfrak{B}}$ (*definability*);
  - (ii) the functions l and u are monotone (*monotonicity*);
  - (iii) $\mathsf{u}(\emptyset) = \emptyset$ (*normality* of u);
  - (iv) if $M \in \mathcal{MS}^{<\infty}(U)$, then $\mathsf{l}(M) \sqsubseteq \mathsf{u}(M)$ (*weak approximation property*);
  - (v) $\mathsf{b}(M) \sqcap M \neq \emptyset$ but $\mathsf{b}(M) \not\sqsubseteq M$ and $\mathsf{b}(M) \ominus M \neq \emptyset$, provided that $\mathsf{b}(M) \neq \emptyset$ $(M \in \mathcal{MS}^{<\infty}(U))$ (*Janus-faced nature of boundary*).

By historical reasons, lower and upper approximations together is called the approximation pair and denoted by $\langle \mathsf{l}, \mathsf{u} \rangle$. With the above properties, it is said that $\langle \mathsf{l}, \mathsf{u} \rangle$ is a *weak approximation pair*.

A number of important and interesting variations of $\mathsf{MAS}(U)$ can be formed. For our aim, the most interesting case is when $\mathsf{MAS}(U)$ is Pawlakian type.

Let $\mathfrak{B}^{\oplus} = \{\oplus_n B \mid B \in \mathfrak{B},\ n = 1, 2, \dots\}$. $\mathsf{MAS}(U)$ is a *strictly set-union type* multiset approximation space if $\mathfrak{D}_{\mathfrak{B}}$ is given by the following inductive definition:

1. $\emptyset \in \mathfrak{D}_{\mathfrak{B}}$, $\mathfrak{B}^{\oplus} \subseteq \mathfrak{D}_{\mathfrak{B}}$, and
2. if $\mathfrak{B}' \subseteq \mathfrak{B}^{\oplus}$, then $\bigsqcup \mathfrak{B}' \in \mathfrak{D}_{\mathfrak{B}}$.

Let $\mathsf{MAS}(U)$ be a strictly set-union type multiset approximation space. Then, $\mathsf{l}, \mathsf{u}, \mathsf{b} : \mathcal{MS}^{<\infty}(U) \to \mathcal{MS}^{<\infty}(U)$ form a *Pawlakian multiset approximation pair* $\langle \mathsf{l}, \mathsf{u} \rangle$ and a *Pawlakian boundary* b if for any multiset $M \in \mathcal{MS}^{<\infty}(U)$

1. $\mathsf{l}(M) = \bigsqcup\{\oplus_n B \mid n \in \mathbb{N}^+, B \in \mathfrak{B} \text{ and } B \sqsubseteq^n M\}$,
2. $\mathsf{b}(M) = \bigsqcup\{\oplus_n B \mid B \in \mathfrak{B}, B \not\sqsubseteq M,\ B \sqcap M \neq \emptyset \text{ and } B \sqcap M \sqsubseteq^n M\}$,
3. $\mathsf{u}(M) = \mathsf{l}(M) \sqcup \mathsf{b}(M)$.

In this case, $\mathsf{MAS}(U)$ is called a *Pawlakian multiset approximation space*.

3.2. R functions in Pawlakian multiset approximation spaces

Let $\mathsf{MAS}(U)$ be a Pawlakian multiset approximation space. Any member of the domain $\mathcal{MS}^{<\infty}(U)$ can be represented in Parikh vector form by the concatenation function `c()` as usual.

Of course, it is assumed that the number of base multisets is finite. Base system is represented in matrix form. It can be formed in three steps with the help of R functions `c()` and `matrix()`:

1. defining base multisets by `c()` (it is assumed that the number of base multisets is n, where $n(>0) \in \mathbb{N}$);

2. forming a base vector from base multisets by `c()`;

3. building the base matrix from the base vector by `matrix()`.

Let $U = \{a, b, c, d, e\}$ as above. The previous process is illustrated by the following example:

1. Defining base multisets:

`> B1 <- c(2,0,0,0,0)`	$B_1 = a^2$
`> B2 <- c(1,1,0,0,0)`	$B_2 = ab$
`> B3 <- c(0,1,0,0,0)`	$B_3 = b$
`> B4 <- c(0,0,1,1,1)`	$B_4 = cde$
`> n <- 4`	$n = 4$

2. Forming the base vector:

```
> Base_vect <- c(B1,B2,B3,B4)
```

3. Building the base systems in matrix form from the base vector:

```
> B <- matrix(Base_vect, nrow=n, ncol=length(U), byrow=T) .
```

That is, the matrix B represents the base system as follows: B has 4 rows (the number of base multisets) and 5 columns (the cardinality of U). The ith row contains the components of the Parikh vector representation of the ith base multiset.

plow(mS, BASE, SU) – Lower approximation
Parameters: multiset **mS** ; base system **BASE**; the universe **SU**.
Description: This function computes the lower approximation of the multiset **mS** over the base system **BASE**.

Example 3.1. Lower approximation

`> plow(W1,B,U)`	
`[1] 0 0 0 0 0`	$\mathsf{l}(W_1) = \emptyset$
`> plow(W2,B,U)`	
`[1] 2 2 0 0 0`	$\mathsf{l}(W_2) = a^2b^2$
`> plow(W3,B,U)`	
`[1] 1 2 0 0 0`	$\mathsf{l}(W_3) = ab^2$
`> plow(W4,B,U)`	
`[1] 2 1 0 0 0`	$\mathsf{l}(W_4) = a^2b$
`> plow(W5,B,U)`	
`[1] 1 1 0 0 0`	$\mathsf{l}(W_5) = ab$

pbound(mS, BASE, SU) – Boundary
Parameters: multiset **mS**, base system **BASE**; the universe **SU**.
Description: This function computes the boundary of the multiset **mS** over the base system **BASE**.

Example 3.2. Boundary

`> pbound(W1,B,U)`	
`[1] 0 0 1 1 1`	$\mathsf{b}(W_1) = cde$
`> pbound(W2,B,U)`	
`[1] 0 0 1 1 1`	$\mathsf{b}(W_2) = cde$
`> pbound(W3,B,U)`	
`[1] 2 0 3 3 3`	$\mathsf{b}(W_3) = a^2c^3d^3e^3$
`> pbound(W4,B,U)`	
`[1] 0 0 0 0 0`	$\mathsf{b}(W_4) = \emptyset$
`> pbound(W5,B,U)`	
`[1] 2 0 2 2 2`	$\mathsf{b}(W_5) = a^2c^2d^2e^2$

pupp(mS, BASE, SU) – Upper approximation
Parameters: multiset **mS**, base system **BASE**; the universe **SU**.
Description: This function computes the upper approximation of the multiset **mS** over the base system **BASE**.

Example 3.3. Upper approximation

`> pupp(W1,B,U)`	
`[1] 0 0 1 1 1`	$\mathsf{u}(W_1) = cde$
`> pupp(W2,B,U)`	
`[1] 2 2 1 1 1`	$\mathsf{u}(W_2) = a^2b^2cde$
`> pupp(W3,B,U)`	
`[1] 2 2 3 3 3`	$\mathsf{u}(W_3) = a^2b^2c^3d^3e^3$
`> pupp(W4,B,U)`	
`[1] 2 1 0 0 0`	$\mathsf{u}(W_4) = a^2b$
`> pupp(W5,B,U)`	
`[1] 2 1 2 2 2`	$\mathsf{u}(W_5) = a^2bc^2d^2e^2$

4. Calculations of boundaries in membrane systems

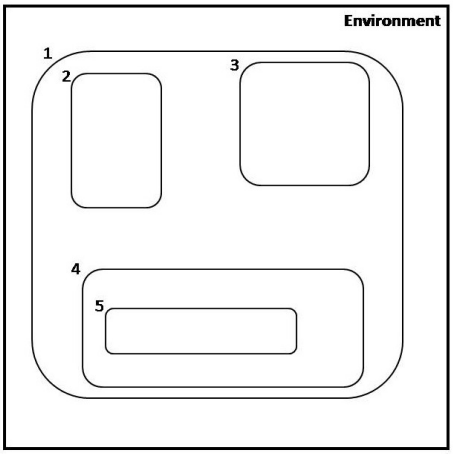
In this section the relationship between multiset approximation spaces and membrane systems is presented.

4.1. Membrane systems

Membrane system, or P system for short, was invented by Gheorghe Păun about 2000 [8, 9]. It was inspired by the architecture and functioning of living cells in order to formulate a model of computation.

Formally, a P system of degree $m(\geq 1)$ is a tuple

$$\Pi = \langle U, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m \rangle.$$



Membranes delimit *regions* $w_1, \ldots, w_m$ separating "inside" from "outside".

Regions are arranged in a hierarchical structure μ.

Each region is
- represented by multisets over a finite set of objects U;
- endowed with two sets of rules.

Evolutions rules regulate the events taking place in the regions.

Communication rules regulate movements of objects through membranes.

Figure 1: A P system represented as a set of nested membranes ($m = 5$)

4.2. Membrane boundaries

Let the P system $\Pi = \langle U, \mu, w_1, w_2, \ldots, w_m, R_1, R_2, \ldots, R_m \rangle$ be given. Further, let $\mathsf{MAS}(\Pi) = \langle \mathcal{MS}^{<\infty}(U), \mathfrak{B}, \mathfrak{D}_{\mathfrak{B}}, \mathsf{l}, \mathsf{b}, \mathsf{u} \rangle$ be a Pawlakian mset approximation space. Then, $\mathsf{MAS}(\Pi)$ is called a *joint (multiset) approximation space* of Π. It should be noted that both the P system Π and the joint approximation space $\mathsf{MAS}(\Pi)$ are defined over the same universe.

Regions in P system Π are represented by multisets $w_1, w_2, \ldots, w_m$. Therefore, putting $w_1, w_2, \ldots, w_m$ into the joint approximation space of Π, they can be approximated, i.e., their Pawlakian lower/upper approximations and boundaries can be determined.

Pawlakian lower approximations of all regions follow the membrane structure. Furthermore, Pawlakian upper approximation and the boundary of the skin membrane completely lie within the environment. However, the upper approximations and boundaries of not skin membranes do not obey the membrane structure in general. Thus, these Pawlakian boundaries have to be adjusted to the membrane structure. This adjustment can be carried out as follows.

Let Π be a P system and $\mathsf{MAS}(\Pi)$ be its joint approximation space. First, let us determine the following quantities. If $B \in \mathfrak{B}$ and $i = 1, 2, \ldots, m$, let

$$N(B,i) = \begin{cases} 0, \text{if } B \sqsubseteq w_i \text{ or } B \sqcap w_i = \emptyset; \\ n, \text{if } i = 1 \text{ and } B \sqcap w_1 \sqsubseteq^n w_1; \\ \mathsf{min}\{k, n \mid B \sqcap w_i \sqsubseteq^k w_i,\ B \ominus w_i \sqsubseteq^n w_{\mathsf{parent}(i)}\}, \quad \text{otherwise.} \end{cases}$$

Then, the functions *membrane boundaries*, *outside* and *inside membrane boundaries* are defined as follows ($i = 1, \ldots, m$):

$$\mathsf{bnd}(w_i) = \bigsqcup\{\oplus_{N(B,i)} B \mid B \in \mathfrak{B}\};$$

$$\mathsf{bnd}^{\mathsf{out}}(w_i) = \mathsf{bnd}(w_i) \ominus w_i;$$

$$\mathsf{bnd}^{\mathsf{in}}(w_i) = \mathsf{bnd}(w_i) \ominus \mathsf{bnd}^{\mathsf{out}}(w_i).$$

4.3. Calculations of membrane boundaries

First, let us give the regions in matrix form. The membrane structure μ is given in vector form in which the ith element defines the parent of the ith region.

Let us illustrate this process by the following example (it is assumed that the multisets W_1, W_2, W_3, W_4, W_5 represent regions, and the multisets B_1, B_2, B_3, B_4 which were given earlier form the base system):

1. Giving regions:
```
> Region <- c(W1,W2,W3,W4,W5)
> m <- 5
> R <- matrix(Region, nrow=m, ncol=length(U), byrow=T)
```
2. Giving the membrane structure:
```
> MU <- c(0,1,1,1,4)
```

`MU` follows the membrane structure which is depicted in Figure 1: W_1 is the skin membrane; W_2, W_3, W_4 are nested in W_1, and W_5 is nested in W_4.

The first function is an auxiliary one in order to be able to calculate the quantities $N(B,i)$'s. It will be called the `NBi()` function.

NB(REGION, i, BASE, j, SU, SMU) – Calculating $N(B,i)$ for fixed base multiset and region
Parameters: regions in matrix form **REGION**, **i**th region, base system **BASE**, **j**th base multiset, the universe **SU**, membrane structure **SMU**.
Description: This function calculates the quantity $N(B,i)$ for the ith region and the jth base multiset.

Example 4.1. Calculating $N(B_1, 3)$

```
> NB(R,3,B,1,U,MU)
[1] 0
```

$N(B_1, 3) = 0$

NBi(i) – Calculating $N(B, i)$'s for the ith region and all base multisets
Parameters: region **i**.
Description: This function calculates the quantities $N(B, i)$'s for the ith region and all base multisets. It calls the **NB**() function.

Example 4.2. Calculating all $N(B, i)$'s

```
> NBi(1)
[1] 0 0 0 1
```

$N(B_1, 1) = 0, N(B_2, 1) = 0, N(B_3, 1) = 0, N(B_4, 1) = 1$

```
> NBi(2)
[1] 0 0 0 1
```

$N(B_1, 2) = 0, N(B_2, 2) = 0, N(B_3, 2) = 0, N(B_4, 2) = 1$

```
> NBi(3)
[1] 0 0 0 3
```

$N(B_1, 3) = 0, N(B_2, 3) = 0, N(B_3, 3) = 0, N(B_4, 3) = 3$

```
> NBi(4)
[1] 0 0 0 0
```

$N(B_1, 4) = 0, N(B_2, 4) = 0, N(B_3, 4) = 0, N(B_4, 4) = 0$

```
> NBi(5)
[1] 1 0 0 0
```

$N(B_1, 5) = 1, N(B_2, 5) = 0, N(B_3, 5) = 0, N(B_4, 5) = 0$

Having obtained the quantities $N(B, i)$'s, the membrane boundaries can be calculated.

bnd(REGION, i, BASE, SU) – Calculating membrane boundary
Parameters: regions in matrix form **REGION**, **i**th region, base system **BASE**; the universe **SU**.
Description: This function calculates the boundary of the ith region, i.e., the ith membrane boundary.

Example 4.3. Calculating all membrane boundaries

```
> bnd(R,1,B,U)
[1] 0 0 1 1 1
```

$\mathsf{bnd}(W_1) = cde$

```
> bnd(R,2,B,U)
[1] 0 0 1 1 1
```

$\mathsf{bnd}(W_2) = cde$

```
> bnd(R,3,B,U)
[1] 0 0 3 3 3
```

$\mathsf{bnd}(W_1) = c^3d^3e^3$

```
> bnd(R,4,B,U)
[1] 0 0 0 0 0
```

$\mathsf{bnd}(W_1) = \emptyset$

```
> bnd(R,5,B,U)
[1] 2 0 0 0 0
```

$\mathsf{bnd}(W_1) = a^2$

Last, the outside/inside membrane boundaries are calculated.

bndout(REGION, i, BASE, SU) – Calculating outside membrane boundary
Parameters: regions in matrix form **REGION**, **i**th region, base system **BASE**;

the universe **SU**.
Description: This function calculates the outside boundary of the ith region, i.e., the ith outside membrane boundary.

Example 4.4. Calculating all outside membrane boundaries

```
> bndout(R,1,B,U)
[1] 0 0 0 1 0
```
$\mathsf{bnd}^{\mathsf{out}}(W_1) = d$

```
> bndout(R,2,B,U)
[1] 0 0 1 0 1
```
$\mathsf{bnd}^{\mathsf{out}}(W_2) = ce$

```
> bndout(R,3,B,U)
[1] 0 0 0 0 3
```
$\mathsf{bnd}^{\mathsf{out}}(W_3) = e^3$

```
> bndout(R,4,B,U)
[1] 0 0 0 0 0
```
$\mathsf{bnd}^{\mathsf{out}}(W_4) = \emptyset$

```
> bndout(R,5,B,U)
[1] 1 0 0 0 0
```
$\mathsf{bnd}^{\mathsf{out}}(W_5) = a$

bndin(REGION, i, BASE, SU) – Calculating inside membrane boundary
Parameters: regions in matrix form **REGION**, **i**th region, base system **BASE**; the universe **SU**.
Description: This function calculates the inside boundary of the ith region, i.e., the ith inside membrane boundary.

Example 4.5. Calculating all inside membrane boundaries

```
> bndin(R,1,B,U)
[1] 0 0 1 0 1
```
$\mathsf{bnd}^{\mathsf{in}}(W_1) = ce$

```
> bndin(R,2,B,U)
[1] 0 0 0 1 0
```
$\mathsf{bnd}^{\mathsf{in}}(W_2) = d$

```
> bndin(R,3,B,U)
[1] 0 0 3 3 0
```
$\mathsf{bnd}^{\mathsf{in}}(W_3) = c^3d^3$

```
> bndin(R,4,B,U)
[1] 0 0 0 0 0
```
$\mathsf{bnd}^{\mathsf{in}}(W_4) = \emptyset$

```
> bndin(R,5,B,U)
[1] 1 0 0 0 0
```
$\mathsf{bnd}^{\mathsf{in}}(W_5) = a$

5. Summary

In this paper such R functions have been presented which allow us to carry out calculations in membrane systems combined with multiset approximation spaces. In this framework membrane boundaries (even inside and outside) can be determined. The calculations are illustrated with examples mainly coming from [6, 7]. The results presented in this paper also prove the usability of R language in membrane computing. Further research direction may be the implementation of membrane communication rules in R in order to show how maximal parallelism can actually be controlled with the help of generated membrane boundaries [1, 6].

References

[1] Csajbók, Z.E., Mihálydeák, T., Maximal parallelism in membrane systems with generated membrane boundaries. In: Beckmann, A., Csuhaj-Varjú, E., Meer, K. (eds.) Language, Life, Limits. 10th Conference on Computability in Europe, CiE 2014, Budapest, Hungary, June 23-27, 2014. Proceedings. LNCS, vol. 8493 (2014), Springer International Publishing, Switzerland, 103–112.

[2] Girish, K.P., John, S.J., Relations and functions in multiset context. Information Sciences **179**(6) (2009), 758–768.

[3] Kudlek, M., Martín-Vide, C., Păun, Gh., Toward a formal macroset theory. In Calude, C., Păun, Gh., Rozenberg, G., Salomaa, A., eds.: WMP. LNCS, vol. 2235 (2001), Berlin Heidelberg, Springer-Verlag, 123–134.

[4] Mihálydeák, T., Csajbók, Z.E., Membranes with boundaries. In: Csuhaj-Varjú, E., Gheorghe, M., Rozenberg, G., Salomaa, A., Vaszil, Gy. (eds.) Membrane Computing. CMC 2012, Budapest, Hungary, August 28-31, 2012, Revised Selected Papers. LNCS, vol. 7762 (2013), Springer-Verlag, Berlin Heidelberg, 277–294.

[5] Mihálydeák, T., Csajbók, Z.E., Partial approximation of multisets and its applications in membrane computing. In: Lingras, P., Wolski, M., Cornelis, C., Mitra, S., Wasilewski, P. (eds.) Rough Sets and Knowledge Technology, 8th International Conference, RSKT 2013, Halifax, NS, Canada, October 11-14, 2013, Proceedings. LNCS-LNAI, vol. 8171 (2013), Springer-Verlag, Berlin, Heidelberg, 99–108.

[6] Mihálydeák, T., Csajbók, Z.E., Takács, P., On the Membrane Computations in the Presence of Membrane Boundaries. Journal of Automata, Languages and Combinatorics 19(1-4) (2014), 227–238.

[7] Mihálydeák, T., Csajbók, Z.E., Takács, P., Communication rules controlled by generated membrane boundaries. In: Alhazov, A., Cojocaru, S., Gheorghe, M., Rogozhin, Y., Salomaa, A. (eds.) Membrane Computing, 14th International Conference, CMC 2013, Chişinău, Republic of Moldova, August 20-23, 2013, Revised Selected Papers. LNCS, vol. 8340 (2014), Springer, Berlin Heidelberg, 265–279.

[8] Păun, Gh., Computing with membranes. Journal of Computer and System Sciences 61(1) (2000), 108–143

[9] Păun, Gh., Membrane Computing. An Introduction. Springer-Verlag, Berlin, (2002)

[10] Păun, Gh., Rozenberg, G., An introduction to and an overview of membrane computing. In: Păun et al. [11], 1–27.

[11] Păun, Gh., Rozenberg, G., Salomaa, A., The Oxford Handbook of Membrane Computing. Oxford University Press, Inc., New York, NY, USA (2010)

[12] Pawlak, Z., Rough sets. International Journal of Computer and Information Sciences 11(5) (1982), 341–356.

[13] Pawlak, Z., Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)

[14] Syropoulos, A., Mathematics of Multisets, *Multiset Processing, Mathematical, Computer Science, and Molecular Computing Points of View,* Workshop on Multiset Processing. Curtea de Arges, Romania, August 21-25, 2000. (2000), 347–358.

[15] Meyer, D., Hornik, K., Generalized and Customizable Sets in R. *Journal of Statistical Software*, Vol. 31 (2009), 1–27.