

# Universality with RBC-like Objects

Shankara Narayanan KRISHNA

Department of Computer Science and Engineering  
Indian Institute of Technology, Bombay  
Powai, Mumbai, India - 400 076  
E-mail: krishna@cfavs.iitb.ac.in

## Abstract

Red Blood Corpuscles (RBCs) are the basic elements of all kinds of cells and they are present in all the cells of mammals in huge numbers. They get replaced periodically. They do not evolve or divide like usual cells; they are just carriers of oxygen and hence are communicating agents in a cell. This being the case, symport/antiport rules are the most suitable control structures to model their activity. We consider a class of P systems where the objects represent RBCs (multisets of objects) and symport/antiport rules are used for communication. We prove a universality result using three membranes where the symport/antiport rules have weight one.

## 1 Introduction

In recent years, observations of cells and their biochemical processes have been of inspiration for the creation of theoretical computational devices. One of the most recent attempts in this direction has been to look at the structure of cells as a set of nested compartments delimited by membranes. Each of the membranes are composed of chemicals and interact with the chemicals swimming in the aqueous solution from the compartments. These objects (chemicals) in the compartments can evolve, and interact with other chemicals and pass to other membranes. Membrane systems (P systems) were introduced in [10].

One of the most elegant variants of P systems was introduced in [9] under the name of membrane systems with symport/antiport. This variant models the synchronized movement of chemicals present in a cell: specific groups of objects may pass together through a membrane either in the same or in opposite direction. The former case is referred to as symport and the latter, antiport. There is no modification to any of the objects, communication is the only driving power of these systems.

Various variants of symport/antiport have been considered in [1], [2], [4], [5] and [6]. In [1] the authors show that passing at most one object per time can generate any recursively enumerable set of numbers using nine membranes; this was improved in [4] to six membranes, which was further improved to five membranes in [2]. The most recent improved result was by P. Frisco wherein universality is obtained in four membranes.

In this paper, we consider P systems having symport/antiport rules motivated by the movement and regeneration of red blood corpuscles (RBC) in the membranes of mammals. We prove universality of such systems in three membranes using symport/antiport rules of weight one. In the next section, we give a brief overview of RBCs and their functions and the motivation for considering them. In Section 3, we illustrate the functioning of the new class of P systems with two examples. In Section 4, we recall the concepts of register machines, which we use for proving universality in Section 5.

## 2 Red Blood Corpuscles (RBC)

Red Blood Corpuscles (RBC) are the basic elements of all kinds of cells. They are composed of a colorless stroma filled in with semifluid haemoglobin and other matters. In most mammals the red corpuscles are circular, but in the camels, birds, reptiles, and the lower vertebrates generally, they are oval, and sometimes more or less spherical in form. In Amphioxus, and most invertebrates, the blood corpuscles are all white or colorless. The RBC (also called erythrocytes) make up 44% of the volume of the blood. The rest of the blood are the white cells (1%) and the plasma. The RBC are the small (7/1000 mm diameter) cells in the blood that contain the red pigment, haemoglobin and carry oxygen around the body. They are round biconcave cells (flattened in the middle) and have no nucleus. Hence they cannot be considered to be individual cells. The total area of all the RBC in the body is about 3,800 square meters.

The RBC are made in the marrow of the flat bones and the blood contains 5 million of them in each cc of blood. The RBC flow through the arteries, veins and capillaries. The RBC live for about 100 to 120 days and then they are broken up. The broken up RBCs are taken by the tissues of the body. About  $25 \times 10^{10}$  corpuscles are replaced daily, a turnover rate of 2.5 million per second. Both damaged and normal but “worn-out” erythrocytes are removed from the vascular system by macrophages, which are found primarily in the liver, spleen, and bone marrow. Breakdown products of hemoglobin are used in the formation of bile (bilirubin), and iron is conserved and used in new red cell production. The production rate of RBCs depend on environmental conditions; at high altitudes, shortage of oxygen stimulates the body to produce more erythropoietin (EPO) that stimulates the production of more RBC. We refer to the period after which new RBCs are produced (100-120 days) as ‘period of production’. An organism dies when it is not capable of producing new RBCs within the period of production.

The main functions of RBC include:

- Transport oxygen in the blood from the lungs to all the cells and tissues of the body.
- Assist with the transport of carbon (IV) oxide from the tissues to the lungs.
- Regulating the acid-base balance of the blood, preventing large changes in pH.
- Assist when a blood clot is formed.

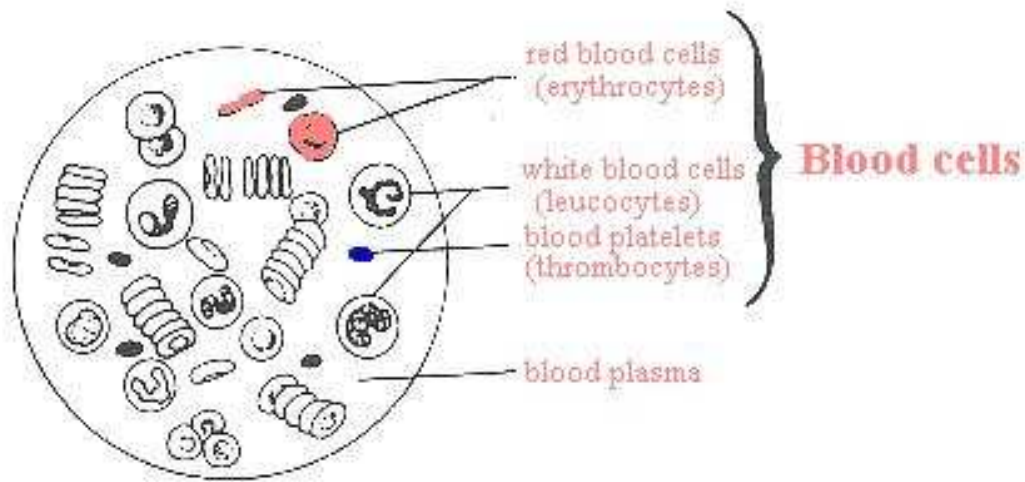


Figure 1: Composition of Blood

## 2.1 Motivation

We are mainly motivated by the following facts about RBC:

- (a) there are enormously many RBCs in the membranes of a mammal at any point of time;
- (b) old RBCs are not used (broken up and absorbed by the system) when new ones are produced;
- (c) the production of new RBCs occur periodically;
- (d) an organism dies when it is not capable of producing new RBCs.

In the following, we consider P systems with multisets of objects. In view of (a), we consider systems having enumerably many objects in its membranes.

Modeling (b) :

The entry of a new object into the system from the environment represents the production of new RBCs in the system. This can be considered as a trigger which results in the use of new RBCs in all the membranes. All the objects which were used for intracellular communication till the step when the new object enters the system are not used for communication from then on. The entry of a new object into the system marks the start of the usage of new RBCs. Since production of new RBCs is a process that is carried out entirely within the system and the old RBCs are 'recycled' into new ones, we represent this by considering enumerably many objects in all membranes, such that every time a new object enters the system, the objects used for communication are those which were never used before. The objects which were used for communication previously are 'ignored'.

Modeling (c) and (d):

We assign a real number  $p$  representing the ‘period of production’. This means that a new object must enter the system after the execution of every  $p$  (or at most  $p$ ) steps. In case new objects do not enter the system for more than  $p$  steps, then all the existing unused objects in all the membranes are ignored. (The unused objects correspond to new RBCs. Every time a new object enters the system, it is due to the communication caused by a set of objects which were used only for at most previous  $p$  steps. Hence, if no new object enters the system for more than  $p$  steps, it is because the current used symbols are unable to bring in any more by way of their communication. This also implies that the remaining unused objects have no more function to play, and so can be ignored. The objects responsible for sustaining the life of the system are the used objects. It will be these used objects which will be representing the output of the system after its death.)

The system halts when no new object enters the system from the environment for more than  $p$  steps and rules to all the current communicating objects are exhausted. The output of the system consists of all the used objects present in a designated output membrane at the end of a halting configuration.

The rules for communication are the usual symport/antiport rules of weight one. An RBC may enter or exit a region (symport); likewise the entry of an RBC into a region can be coupled with the exit of another RBC (antiport). Since an RBC is considered an indivisible single unit till it is broken up, this means we are considering rules of the form  $(x, in)$ ,  $(y, out)$  or  $((x, in); (y, out))$ ,  $|x| = |y| = 1$ .

We do not give a formal definition of this class of P systems since the concept of applying symport/antiport rules to all possible objects (multisets of objects) is well known [8]. We refer to this class of systems as ‘P systems with symport/antiport having RBC like objects’. The set of (vectors of) natural numbers computed by these systems having at most  $m$  membranes is denoted by  $NRBCP_m(sym_a, anti_b)$ ,  $a \leq 1, b \leq 1$ . Now, we give some examples.

### 3 Examples

In this example, the number of objects in each membrane is finite.

**Example 3.1** Consider the system

$$\Pi = (V, \mu, w_1, w_2, w_3, E, R_1, R_2, R_3, p, 3),$$

where

- $V = \{s, a, b, c, d, e, f, g, h, l, m, n, o, r\}$  is the alphabet of objects;
- $\mu = [1 [2 [3 ]3 ]2 ]1$  is the membrane structure;
- $w_1 = \{s, g, h\}$ ,  $w_2 = \{a, b\}$ ,  $w_3 = \{e, f, m, n, o, r\}$   
are the initial multisets of objects in membranes 1, 2 and 3;
- $E = \{c, d, l\}$  is the set of objects present in the environment;
- $R_1 = \{((s, out); (c, in)), ((a, out); (d, in)), ((b, out); (l, in))\}$ ,

- $R_2 = \{((c, in); (a, out)); ((d, in); (b, out)), ((g, in); (e, out)), ((h, in); (f, out))\}$ ,
- $R_3 = \{((c, in); (e, out)); ((d, in); (f, out))\}$ ,
- $p$  the period of production. From the rules, it can be seen that a new object enters the system at the end of every third step.

We illustrate the working of the system in the following steps:

1.  $[_1 s g h [{}_2 a b [{}_3 e f m n o r]_3]_2]_1$
2.  $[_1 c g h [{}_2 a b [{}_3 e f m n o r]_3]_2]_1$   $c$  : new object
3.  $[_1 a g h [{}_2 c b [{}_3 e f m n o r]_3]_2]_1$
4.  $[_1 d g h [{}_2 e b [{}_3 c f m n o r]_3]_2]_1$   $d$  : new object
5.  $[_1 b g h [{}_2 e d [{}_3 c f m n o r]_3]_2]_1$   
(The rule  $((g, in); (e, out))$  of  $R_2$  cannot be applied anymore here since  $e$  is a communicating object of the previous cycle)
6.  $[_1 l g h [{}_2 e f [{}_3 c d m n o r]_3]_2]_1$   $l$  : new object
7.  $[_1 l g h [{}_2 e f [{}_3 c d m n o r]_3]_2]_1$  no new object
8.  $[_1 l g h [{}_2 e f [{}_3 c d m n o r]_3]_2]_1$  no new object
9. If  $p = 3$ , then the system halts as no more new symbols come in for 3 steps and no more applicable rules to objects that were unused before. The output of the system is collected in membrane 3 and consists of the objects  $c, d$  (the other objects in membrane 3 are not considered as part of the output as they are unused).

Note that even if  $p$  is not considered to be 3 but to be some finite number  $k$ , the system halts with the same output at the end of  $k + 6$  steps.

**Example 3.2** Consider the system

$$\Pi = (V, \mu, w_1, w_2, w_3, E, R_1, R_2, R_3, p, 3),$$

where:

- $V = \{a^{(i)}, b^{(i)}, c^{(i)}, e^{(i)}, k^{(i)}, l^{(i)}, p^{(i)}, f^{(i)}, s^{(0)} \mid \forall i \in \mathbf{N}\}$ ,
- $\mu = [{}_1 [{}_2 [{}_3]_3]_2]_1$ ,
- $w_1 = \{s^{(0)}, b^{(i)}, c^{(i)} \mid \forall i \in \mathbf{N}\}$ ,
- $w_2 = \{e^{(i)} \mid \forall i \in \mathbf{N}\}$ ,
- $w_3 = \{k^{(i)}, p^{(i)}, f^{(i)} \mid \forall i \in \mathbf{N}\}$ ,
- $E = \{a^{(i)}, l^{(i)} \mid \forall i \in \mathbf{N}\}$ ,

- $R_1 = \{((s^{(0)}, out); (a^{(1)}, in))\}$   
 $\cup \{((k^{(i)}, out); (a^{(i+1)}, in)), ((k^{(i)}, out); (l^{(i+1)}, in)) \mid \forall i \in \mathbf{N}\},$
- $R_2 = \{((e^{(i)}, out); (a^{(i)}, in)) \mid \forall i \in \mathbf{N}\}$   
 $\cup \{((f^{(i)}, out); (b^{(i)}, in)), ((k^{(i)}, out); (c^{(i)}, in)) \mid \forall i \in \mathbf{N}\},$
- $R_3 = \{((f^{(i)}, out); (a^{(i)}, in)), ((k^{(i)}, out); (b^{(i)}, in)) \mid \forall i \in \mathbf{N}\}$   
 $\cup \{((p^{(i)}, out); (c^{(i)}, in)) \mid \forall i \in \mathbf{N}\},$
- $p = 7.$

We will observe the computation for a few steps:

1.  $[_1 s^{(0)} b^{(i)} c^{(i)}]_2 e^{(i)} [{}_3 k^{(i)} p^{(i)} f^{(i)}]_3 ]_2 ]_1, \forall i \in \mathbf{N}$
2.  $[_1 a^{(1)} b^{(i)} c^{(i)}]_2 e^{(i)} [{}_3 k^{(i)} p^{(i)} f^{(i)}]_3 ]_2 ]_1, \forall i \in \mathbf{N}$   
 $(a^{(1)} : \text{new object})$
3.  $[_1 e^{(1)} b^{(i)} c^{(i)}]_2 a^{(1)} e^{(i+1)} [{}_3 k^{(i)} p^{(i)} f^{(i)}]_3 ]_2 ]_1, \forall i \in \mathbf{N}$
4.  $[_1 e^{(1)} b^{(i)} c^{(i)}]_2 f^{(1)} e^{(i+1)} [{}_3 k^{(i)} p^{(i)} f^{(i+1)} a^{(1)}]_3 ]_2 ]_1, \forall i \in \mathbf{N}$
5.  $[_1 e^{(1)} f^{(1)} b^{(i+1)} c^{(i)}]_2 b^{(1)} e^{(i+1)} [{}_3 k^{(i)} p^{(i)} f^{(i+1)} a^{(1)}]_3 ]_2 ]_1, \forall i \in \mathbf{N}$
6.  $[_1 e^{(1)} f^{(1)} b^{(i+1)} c^{(i)}]_2 k^{(1)} e^{(i+1)} [{}_3 b^{(1)} k^{(i+1)} p^{(i)} f^{(i+1)} a^{(1)}]_3 ]_2 ]_1, \forall i \in \mathbf{N}$
7.  $[_1 e^{(1)} f^{(1)} b^{(i+1)} c^{(i+1)} k^{(1)}]_2 c^{(1)} e^{(i+1)} [{}_3 b^{(1)} k^{(i+1)} p^{(i)} f^{(i+1)} a^{(1)}]_3 ]_2 ]_1,$
8.  $[_1 e^{(1)} f^{(1)} b^{(i+1)} c^{(i+1)} l^{(2)}]_2 p^{(1)} e^{(i+1)} [{}_3 b^{(1)} k^{(i+1)} p^{(i+1)} c^{(1)} f^{(i+1)} a^{(1)}]_3 ]_2 ]_1,$   
 $\forall i \in \mathbf{N} \quad (l^{(2)} : \text{new object})$
9. *In the previous step, a new object  $l^{(2)}$  has entered the system. There are no more rules applicable and hence, at the end of 7 steps, the system halts. The output would be the symbols  $a^{(1)}, b^{(1)}, c^{(1)}$  in membrane three, since the symbols  $\{k^{i+1}, p^{i+1}, f^{i+1} \mid i \geq 1\}$  are all unused and hence are ignored.*

In general, the output of this system consists of equal numbers of occurrences of  $a^{(i)}, b^{(i)}$  and  $c^{(i)}$  in membrane three.

In both the examples above, we have used only antiport rules.

## 4 Register Machines

In this section, we briefly recall the concept of Minsky's register machine. Minsky [3] showed that the universal computational power can be reached by such an abstract machine using a finite number of registers for storing arbitrarily large non-negative integers. The machine runs a program consisting of numbered instructions of several simple types. Several variants of the machine with different number of registers and different instruction sets were shown to be computationally universal. The basic instruction types we use here are (these instructions include those used by Minsky, also see [7]):

- $a^+(l)$  – add 1 to the contents of register  $a$  and continue with the  $l$ th instruction,
- $a^-(l, k)$  – if the contents of register  $a$  is non-zero, then subtract 1 from it and continue with the  $l$ th instruction, otherwise continue with the  $k$ th instruction,
- *Halt* – halt the machine.

We can assume that *Halt* is used as the last instruction of each program. Based on the results established by Minsky in [7, 3], it was shown that any partial recursive function  $f$  can be computed by a register machine with only 3 registers using instructions as those defined above, starting with a value  $n$  and halting with the value  $f(n)$  in the first register, if  $f(n)$  is defined; if  $f(n)$  is undefined for some  $n \geq 0$ , then the register machine never halts with input  $n$ .

## 5 Generative Power of RBC P Systems

We prove the main result of this paper, viz., universality can be obtained using *RBCP* systems of degree 3 having symport/antiport rules of weight one.

**Theorem 5.1** *For each recursively enumerable function  $f : \mathbf{N} \rightarrow \mathbf{N}$ , there is an RBCP system  $\Pi$  with 3 membranes using symport/antiport rules of weight 1 such that the following condition is satisfied: For any arbitrary  $x \in \mathbf{N}$ , the system  $\Pi$  halts if and only if  $f(x)$  is defined, and if so,  $NRBCP_3(sym_1, anti_1) = \{f(x)\}$ .*

*Proof.* Consider a register machine with  $m$  registers, the last one being a special output register which is never decremented. Let there be a program  $P$  which computes  $f$ . The input value  $x$  is expected to be in  $\mathbf{r}_1$  and the output value in  $\mathbf{r}_m$ . Without loss of generality, assume that all registers other than  $\mathbf{r}_1$  contain zero at the beginning of a computation.

In view of [3], we consider a register machine with 3 registers. Let the last instruction *Halt* correspond to an instruction number  $n$  in the program  $P$ . Consider an RBCP system

$$\Pi = (V, \mu, w_1, w_2, w_3, E, R_1, R_2, R_3, p, 3),$$

where:

$$\begin{aligned} V &= \{P_l^{(i)}, P_{lk}^{(i)}, P_{lfgk}^{(i)}, L_n^{(i)} \mid \forall i \in \mathbf{N}, 1 \leq l, f, g \leq n, 1 \leq k \leq 3\} \\ &\cup \{o_1^{(i)}, o_2^{(i)}, o_{l3}^{(i)} \mid \forall i \in \mathbf{N}, 1 \leq l \leq n\} \cup \{s\} \\ &\cup \{r_{21}^{(i)}, r_{22}^{(i)}, r_{l3}^{(i)} \mid \forall i \in \mathbf{N}\} \\ &\cup \{t_2^{(i)}, Q_{lfk}^{(i)}, A_{l3}^{(i)}, U_{l3}^{(i)}, V_{l3}^{(i)}, S_{l3}^{(i)} \mid \forall i \in \mathbf{N}, 1 \leq l, f \leq n, 1 \leq k \leq 3\} \cup \{s\}, \\ \mu &= [1 \ [2 \ [3 \ ]_3 \ ]_2 \ ]_1, \\ w_1 &= \{s, o_2^{(i)}, o_1^{(x+i)} \mid \forall i \in \mathbf{N}\}, \\ w_2 &= \{Q_{lfk}^{(i)}, r_{21}^{(i)}, r_{22}^{(i)}, t_2^{(i)}, o_1^{(j)} \mid \forall i \in \mathbf{N}, 1 \leq l, f \leq n, 1 \leq j \leq x, k = 1, 2\}, \\ w_3 &= \{Q_{l3}^{(i)}, A_{l3}^{(i)}, U_{l3}^{(i)}, V_{l3}^{(i)}, S_{l3}^{(i)}, r_{l3}^{(i)} \mid \forall i \in \mathbf{N}, 1 \leq l, f \leq n\}, \end{aligned}$$

$$E = \{P_l^{(i)}, P_{lk}^{(i)}, P_{l_f/gk}^{(i)}, L_n^{(i)}, o_{l3}^{(i)} \mid \forall i \in \mathbf{N}, 1 \leq l, f, g \leq n, 1 \leq k \leq 3\},$$

$$p = 7.$$

Before going into the rules, we will explain the terminology used above. The terms  $P_{lk}^{(i)}$  represent the instruction-register pair  $(l, k)$ . It essentially represents that the register machine is at instruction  $l$  and that register  $k$  is incremented. Similarly,  $P_{l_f/gk}^{(i)}$  represents that the register machine at instruction  $l$  tries to decrement register  $k$ , and if successful goes to instruction  $f$ , otherwise continues with instruction  $g$ . The symbols  $o_1^{(i)}, o_2^{(i)}, o_{l3}^{(i)}$  represent the contents of registers 1, 2 and 3. The symbols  $r_{21}^{(i)}, r_{22}^{(i)}$  and  $r_{l3}^{(i)}$  are used to carry out addition to registers 1, 2 and 3 respectively. Likewise, the symbols  $A_{l3}^{(i)}, U_{l3}^{(i)}$  and  $V_{l3}^{(i)}$  are used to add to registers 1, 2 and 3 respectively. The symbols  $Q_{l_g3}^{(i)}, Q_{l_fk}^{(i)}, S_{l_f3}^{(i)}, 1 \leq k \leq 2$  facilitate subtraction of registers 1 and 2. The symbol  $L_n^{(i)}$  indicates that the last instruction has been executed and halts the system. The symbol  $s$  in membrane 1 in the initial configuration triggers of the computation.

Now we will elaborate on the rules. The contents of  $\mathbf{r}_1$  ( $o_1^{(j)}, j \leq x$ ) is present in membrane two at the initial configuration. The computation starts with the the rule  $((s, out); (P_{1k}^{(1)}, in))$  or  $((s, out); (P_{1_f/gk}^{(1)}, in))$  in  $R_1$ . This represents the fact that we are at the first instruction of the program and an addition may be carried out to any of the registers  $\mathbf{r}_k, k \leq 3$  or a subtraction to  $\mathbf{r}_k, k \leq 2$ . A symbol  $P_{lk}^{(i)}$  or  $P_{l_f/gk}^{(i)}$  in membrane one indicates that the program is at instruction  $l$  and an operation to register  $k$  is going to be carried out.

Addition to register 3:

For this to happen, the new symbol  $P_{l3}^{(j)}, j \in \mathbf{N}$ , must be in membrane 1. The rules applied in order are as follows:

1.  $R_2 : ((P_{l3}^{(j)}, in))$
2.  $R_3 : ((P_{l3}^{(j)}, in); (r_{l3}^{(j)}, out))$
3.  $R_2 : ((r_{l3}^{(j)}, out))$
4.  $R_1 : ((r_{l3}^{(j)}, out); (o_{l3}^{(j+1)}, in))$  *new object*
5.  $R_2 : ((o_{l3}^{(j+1)}, in); (t_2^{(j+1)}, out))$
6.  $R_3 : ((o_{l3}^{(j+1)}, in); (V_{l3}^{(j+1)}, out))$
7.  $R_2 : ((V_{l3}^{(j+1)}, out))$
8.  $R_1 : ((V_{l3}^{(j+1)}, out); (P_m^{(j+2)}, in))$ , if  $i_l = 3^+(m)$ , *new object*
9.  $R_1 : ((P_m^{(j+2)}, out); (P_{mk}^{(j+2)}, in))$   $i_m = k^+(q)$  *new object*
10.  $R_1 : ((P_m^{(j+2)}, out); (P_{m_f/gk}^{(j+2)}, in))$   $i_m = k^-(f, g)$  *new object*

Rule 1 takes the symbol  $P_{l_3}^{(i)}$  in to membrane 3, where the symbols  $o_{l_3}^{(i)}$  representing the contents of register 3 have to be kept;  $P_{l_3}^{(i)}$  enters membrane 3 by pulling out  $r_{l_3}^{(i)}$ , which travels to the skin membrane where from it is ejected out, simultaneously taking in a new symbol  $o_{l_3}^{(i+1)}$ . This new symbol travels to its destination membrane 3, by using an intermediate symbol  $t_2^{(i+1)}$  in membrane 2. Finally, it enters membrane 3 by pushing out  $V_{l_3}^{(i+1)}$ , which then brings into membrane one the next symbol for proceeding the computation.

Addition to registers 1,2:

Let the symbol  $P_{l_a}^{(j)}$ ,  $j \in \mathbf{N}$ ,  $a = 1$  or  $2$ , be in membrane one. Then the following rules are applied in order:

1.  $R_2 : ((P_{l_a}^{(j)}, in); (r_{2a}^{(j)}, out))$
2.  $R_1 : ((r_{2a}^{(j)}, out))$
3.  $R_3 : ((P_{l_1}^{(j)}, in); (A_{l_3}^{(j)}, out))$  and  $((P_{l_2}^{(j)}, in); (U_{l_3}^{(j)}, out))$
4.  $R_2 : ((A_{l_3}^{(j)}, out); (o_1^{(j+1)}, in))$  and  $((U_{l_3}^{(j)}, out); (o_2^{(j+1)}, in))$
5.  $R_1 : ((A_{l_3}^{(j)}, out); (P_m^{(j+1)}, in))$ , if  $i_l = 1^+(m)$  new object
6.  $R_1 : ((U_{l_3}^{(j)}, out); (P_m^{(j+1)}, in))$ , if  $i_l = 2^+(m)$  new object
7.  $R_1 : ((P_m^{(j+1)}, out); (P_{mk}^{(j+1)}, in))$   $i_m = k^+(q)$  new object
8.  $R_1 : ((P_m^{(j+1)}, out); (P_{m_{f/g}k}^{(j+1)}, in))$   $i_m = k^-(f, g)$  new object

The first rule pulls the symbol  $P_{l_a}^{(j)}$  to membrane 2, simultaneously pushing out  $r_{2a}^{(j)}$ . The second and third rules happen in parallel: The object  $r_{2a}^{(j)}$  leaves membrane one; at the same time,  $P_{l_a}^{(j)}$  enters membrane 3 pushing out  $A_{l_3}^{(j)}$  or  $U_{l_3}^{(j)}$  depending on whether  $a$  is one or two. These symbols ( $A_{l_3}^{(j)}$  or  $U_{l_3}^{(j)}$ ) then bring in from membrane one the symbols  $o_1^{(j+1)}$ ,  $o_2^{(j+1)}$  and from the skin the next symbol to proceed with the computation.

Subtraction of registers 1, 2:

We have the symbol  $P_{l_{f/g}k}^{(j)}$ ,  $j \in \mathbf{N}$ ,  $k = 1, 2$ , in membrane one. The following rules are applied in order:

1.  $R_2 : ((P_{l_{f/g}k}^{(j)}, in); (Q_{l_{fk}}^{(j)}, out))$ ,  $k = 1, 2$
  2.  $R_2 : ((Q_{l_{fk}}^{(j)}, in); (o_k^{(j)}, out))$ ,  $k = 1, 2$ ,  $j < i$
  3.  $R_3 : ((P_{l_{f/g}k}^{(j)}, in); (Q_{l_{g3}}^{(j)}, out))$
- If no  $o_k^{(j)}$  :
4.  $R_2 : ((Q_{l_{g3}}^{(j)}, out); (Q_{l_{fk}}^{(j)}, in))$ ,  $k = 1, 2$

5.  $R_1 : ((Q_{l_g 3}^{(j)}, out); (P_g^{(j+1)}, in))$  new object
6.  $R_1 : ((P_g^{(j+1)}, out); (P_{gk}^{(j+1)}, in))$ , if  $i_g = k^+(m)$  new object
7.  $R_1 : ((P_g^{(j+1)}, out); (P_{g_m/q k}^{(j+1)}, in))$ , if  $i_g = k^-(m, q)$  new object
- If  $o_k^{(j)}$  :
8.  $R_3 : ((Q_{l_f k}^{(j)}, in); (S_{l_f 3}^{(j)}, out))$
9.  $R_2 : ((S_{l_f 3}^{(j)}, out))$
10.  $R_1 : ((S_{l_f 3}^{(j)}, out); (P_f^{(j+1)}, in))$  new object
11.  $R_1 : ((P_f^{(j+1)}, out); (P_{fk}^{(j+1)}, in))$ , if  $i_f = k^+(m)$  new object
12.  $R_1 : ((P_f^{(j+1)}, out); (P_{f_m/q k}^{(j+1)}, in))$ , if  $i_f = k^-(m, q)$  new object

For a subtraction to take place, the symbol  $P_{l_f/g k}^{(j)}$ ,  $k = 1, 2$ , must be in membrane one. The first rule guesses that subtraction is possible and pushes the symbol  $Q_{l_f k}^{(j)}$  out of membrane two when simultaneously taking in  $P_{l_f/g k}^{(j)}$ . This is followed by rule 2 wherein the symbol  $Q_{l_f k}^{(j)}$  re-enters membrane two, with an  $o_k^{(j)}$  coming out. The third rule is applied in parallel to the second rule;  $P_{l_f/g k}^{(j)}$  enters membrane three pushing out  $Q_{l_g 3}^{(j)}$ .

Case 1: If contents of  $\mathbf{r}_k$  is zero.

In this case, rule 2 would not have been applicable and so we would be having  $Q_{fk}^{(j)}$  in membrane one and  $Q_{l_g 3}^{(j)}$  in membrane two. Hence rule 4 is applicable and  $Q_{l_g 3}^{(j)}$  enters membrane one. From membrane one, it goes out and gets the new symbol to continue the rest of the computation.

Case 2: If contents of  $\mathbf{r}_k$  is non-zero.

In this case, we would have both  $Q_{l_g 3}^{(j)}$  and  $Q_{l_f k}^{(j)}$  in membrane two. Rule 4 cannot be applied; but rule 8 can be applied. This takes  $Q_{l_f k}^{(j)}$  to membrane three pushing out  $S_{l_f 3}^{(j)}$ . This symbol then exits the skin getting in the next new symbol to continue the computation.

Note here that even if  $\mathbf{r}_k$  was zero, then we would have ended up having  $Q_{l_f k}^{(j)}$  in membrane two (by rule 4). In the next step, rule 5 brings in a new symbol which forbids the application of rules 9 and 10.

Termination:

If the last instruction  $n$  of the program can be reached successfully, then we would have  $P_n^{(j)}$  in membrane one. There are no more instructions to be executed, so the system needs to halt and the contents of register 3 should be read from membrane

3. For this, we have the following rules:

1.  $R_1 : ((P_n^{(j)}, out); (L_n^{(j+1)}, in))$  *new object*
2.  $R_2 : ((L_n^{(j+1)}, in)$
3.  $R_3 : ((L_n^{(j+1)}, in); (a, out)),$   
 $a \in \{P_{lk}^{(i)}, P_{l_f/gk}^{(i)} \mid 1 \leq k \leq 3, \forall i \in \mathbf{N}, 1 \leq l, f, g \leq n\}$
4.  $R_3 : ((L_n^{(j+1)}, out))$

When the last instruction is reached, the special symbol  $L_n^{(j)}$  enters the system. This symbol then brings out all used symbols  $P_{lk}^{(i)}, P_{l_f/gk}^{(i)}$  from membrane 3 and remains in membrane 2 at the end. From the rules it can be seen that  $7 \leq p \leq k$ . So, once all the used symbols  $P_{lk}^{(i)}, P_{l_f/gk}^{(i)}$  lodged in membrane 3 are out, or if seven steps are over after  $L_n^{(j+1)}$  is in, whichever happens later, the system halts. The number of used symbols in membrane 3 at the end is the output which is precisely the set of symbols  $o_{t_3}^{(i)}$ . ■

## 6 Concluding Remarks

In this paper, we have considered a class of P systems having symport/antiport rules, inspired from the behaviour of RBCs from the mammalian cells, hence having a direct biological motivation. Modeling the behaviour of RBCs has yielded a significant power: universality is obtained with three membranes using symport/antiport rules of weight one.

## References

- [1] F. Bernadini, M. Gheorghe, On the power of minimal symport/antiport, *Pre-proceedings of Workshop on Membrane Computing, WMC-2003*, Tarragona, July 17-22, 2003.
- [2] F. Bernadini, A. Păun, Symport/antiport : Five membranes suffices, In A. Alhazov, C. Martin-Vide and G. Păun editors, *Workshop on Membrane Computing, WMC 2003*, Tarragona, 72–83, 2003.
- [3] R. Freund and M. Oswald, GP systems with forbidding context, *Fundamenta Informaticae*, 49, 1-3, 81–102, 2002
- [4] L. Kari, C. Martin-Vide, A. Păun, *Aspects of Molecular Computing: Essays dedicated to Tom Head on the Occasion of his 70th birthday*, vol. 2950 of LNCS, chapter on Universality of P systems with minimal symport/antiport rules, 254–265, Springer-Verlag, Berlin, Heidelberg, NewYork, 2004.

- [5] C. Martin-Vide, A. Păun, G. Păun, On the power of P systems with symport rules, *Journal of Universal Computer Science*, 8:317–331, 2002.
- [6] C. Martin-Vide, A. Păun, G. Păun, G. Rozenberg, Membrane systems with coupled transport: universality and normal forms, *Fundamenta Informaticae*, 49:1–15, 2002.
- [7] M.L. Minsky, *Finite and Infinite Machines*, Prentice Hall, Englewood Cliffs, New Jersey, 1967.
- [8] A. Păun, Unconventional Models of Computation: DNA and Membrane Computing, Ph.D thesis, Chapter 14, 2003.
- [9] A. Păun, G. Păun, The power of communication: P systems with symport/antiport, *New Generation Computing*, 20(3), 295–306, 2002.
- [10] G. Păun, Computing with Membranes, *Journal of Computer and System Sciences*, 1(61), 108–143, 200.