Counting Time in Computing with Cells

Oscar H. Ibarra[1] and Andrei Păun[2]

[1] Department of Computer Science,
University of California - Santa Barbara, Santa Barbara, CA 93106
ibarra@cs.ucsb.edu
[2] Department of Computer Science/IfM, Louisiana Tech University,
P.O. Box 10348, Ruston, LA 71272
apaun@latech.edu

Abstract. We consider models of P systems using time either as the output of a computation or as a means of synchronizing the hugely complex processes that take place in a cell. In the first part of the paper, we introduce and study the properties of "timed symport/antiport systems". In the second part we introduce several new features for P systems: the association/deassociation of molecules (modeling for example the protein-protein interactions), ion channel rules and gene activation rules. We show that such timed systems are universal. We also prove several properties concerning these systems.

1 Introduction

We continue the work on symport/antiport P systems [7], [8], [15] using a new paradigm: time as the output of a computation. In recent years, several approaches have been undertaken considering time as part of a biological system's way of "computing". We can mention here the work of W. Maass: he considered a new way to compute with spiking neurons [12]. His model was based on the observation that if considering only the frequency of the neuron's firing signal as a computational framework for the brain, then the brain itself would be very slow computing using only 2-3 spikes per neuron in 150 ms. It is clear that other information is transmitted through these spikes of the neurons. Maass considered a new idea (that seems to be supported experimentally): that also the temporal pattern of the spikes emitted by a neuron is important for the actual message sent. Another feature studied by Maass is that during the "computation", the actual state of a neuron depends on the previous states that the neuron has passed through; maybe even from the birth of the organism. In the current paper we will define significant "configurations" for the system, using similar ideas as in [12] in the sense that for a cell it is important whether it passes through a few "important" configurations.

We also note that in the last year two papers considered the properties of systems in which the rules can take different amounts of time to be completed. Following this idea, in [5] and [14], the authors study the case when the time required for "executing" rules in a system may change sometimes even unpredictably. In such a setting it is an interesting question whether the cell can behave in a similar fashion for different times associated with rules; then several such systems with timed rules are studied. In other words, the authors considered the "time-free" devices; such systems will perform the same steps irrespective of the different time lengths associated with each rule in the system. We will follow a similar idea of timing each rule in such a system and propose a new model of P systems in Section 3, but we devise the new model to be closer to cell biology and useful from a cell-simulation point of view: our definition will consider the case when each rule has a specified duration for the reaction they model (which can be determined experimentally).

In this paper, we consider another way of outputting the result of the computation of a P system. The idea originates in [18] as Problem W; the novelty is that instead of the "standard" way to output, like the multiplicities of objects found at the end at the computation in a distinguished membrane as it was defined in the model from [15], it seems more "natural" to consider certain *events* (i.e., configurations) that may occur during a computation and to relate the output of such a computation with the time interval between such distinguished configurations. Our system will compute a set of numbers like in the case of "normal" symport/antiport [15], but the benefit of the current setting is that the computation and the observance of the output are now close to the biology and to the tools used for cell biology. The model of the "timed" P system that we investigate here is the symport/antiport P system. This has been a popular model that has been accepted by the research community immediately after its introduction. Symport/antiport systems are now a very successful model for P systems due to their simplicity and the fact that they observe the basic physical law of matter conservation (the system computes by communication: the objects are not created nor destroyed, but rather only moved in the system). Here, we are studying another way of viewing the output of such a system; the motivation comes from the fact that cells can become fluorescent if, for example, some types of proteins with fluorescence properties are present in the cells. Such a fluorescent "configuration" of a cell will be the configuration that starts the clock used for the output. Even more interesting (making our definition a very natural way of viewing the output of a system) is the fact that there are tools currently used by researchers in cell biology that can detect the fluorescence of each cell individually. The procedure is performed by a device which can check one cell at a time for fluorescence and can automatically decide to put the cell in either the test-tube containing the fluorescent cells or in the test-tube containing the non-fluorescent cells. The procedure does not destroy the cells, meaning that the same process can be performed repeatedly for a given cell computation. Such an automated technique for viewing the output of a computation using cells is highly desirable since it holds the promise of fast readouts of the computations (in contrast with manual "readouts" that could take several days, see for example the well-known Adleman's experiment, [1]).

The main idea of the new definition is that one has a colony of cells; each cell in the colony having the same (nondeterministic) program that performs a computation. There is a configuration of the system that gives the cell a fluorescence, property that can be detected by devices such as *fluorescence activated cell sorters*, in short FACS. Such a device takes the input test-tube (that contains our colony of cells) and splits it into cells that are not fluorescent yet and cells that are fluorescent. The cells that become fluorescent at some time t (i.e. are detected to be fluorescent for the first time) will have a timer associated to their test-tube that starts "ticking", and will be continuously checked whether they are still in the fluorescent state or not. We will consider the moment that a cell is no longer fluorescent as the moment when we receive the "stop clock" signal, and the system outputs the value computed by the cell to be the time interval when the cell was fluorescent and the instant when it is no longer fluorescent. In this way, by using a FACS one can obtain the output of the computation of such a P system automatically (we consider that it is a easy task to design a system which feeds back the fluorescent test-tube(s) into the FACS incrementing a counter/timer at each feedback, and writing on some medium the content of the counter if a cell was detected to be no longer fluorescent). In other words, we will "output" the duration in the number of "clock cycles" during which the cell was fluorescent. Such a system could output the computation of an entire colony of cells, not only the computation of a single cell. This gives another order of parallelism to our setting which is another strongly desirable feature.

2 Timed Symport/Antiport Systems

We will use a modified definition than the one in [15]; instead of specifying the output region where the result of the computation will be "stored" in a halting computation, we specify two relations C_{start} and C_{stop} (which are computable by multicounter machines) that need to be satisfied by the multisets of objects in the membrane structure at two different times during the computation.

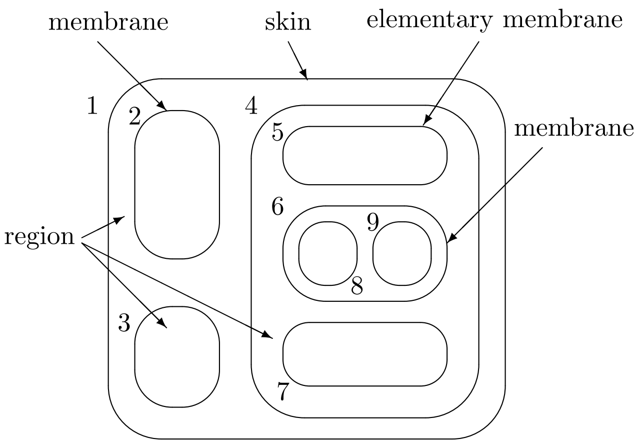

Fig. 1. A membrane structure

An important observation is the fact that we will not require the cell to "stop working" when reaching the result; i.e. we will not require the strong restriction that the system reach a halting configuration for a computation to have a result.

Before progressing further we give some basic notions used in the remainder of the paper; the language theory notions used but not described are standard, and can be found in any of the many monographs available, for instance, in [19].

A membrane structure is pictorially represented by a Venn diagram (like the one in Figure 1), and it will be represented here by a string of matching parentheses. For instance, the membrane structure from Figure 1 can be represented by $[_1[_2\ ]_2[_3\ ]_3[_4[_5\ ]_5[_6[_8\ ]_8[_9\ ]_9]_6[_7\ ]_7]_4]_1$.

A multiset over a set X is a mapping $M : X \longrightarrow \mathbf{N}$. Here we always use multisets over finite sets X (that is, X will be an alphabet). A multiset with a finite support can be represented by a string over X; the number of occurrences of a symbol $a \in X$ in a string $x \in X^*$ represents the multiplicity of a in the multiset represented by x. Clearly, all permutations of a string represent the same multiset, and the empty multiset is represented by the empty string, λ.

We will use symport/antiport rules[1]; mathematically, we can capture the idea of symport by considering rules of the form (ab, in) and (ab, out) associated with a membrane, and stating that the objects a, b can enter, respectively, exit the membrane together. For antiport we consider rules of the form $(a, out; b, in)$, stating that a exits and at the same time b enters the membrane.

Based on rules of this types, we modify the definition from [15] to introduce the model of a *timed symport/antiport system* as a construct,

$$\Pi = (V, \mu, w_1, \ldots, w_m, E, R_1, \ldots, R_m, C_{start}, C_{stop}),\ where:$$

- $V = \{a_1, ..., a_k\}$ is an alphabet (its elements are called *objects*);
- μ is a membrane structure consisting of m membranes, with the membranes (and hence the regions) bijectively labeled with $1, 2, \ldots, m$; m is called the *degree* of Π;
- $w_i, 1 \leq i \leq m$, are strings over V representing multisets of objects associated with the regions $1, 2, \ldots, m$ of μ, present in the system at the beginning of a computation;
- $E \subseteq V$ is the set of objects that are continuously available in the environment in arbitrarily many copies;
- $R_1, \ldots, R_m$ are finite sets of symport and antiport rules over the alphabet V associated with the membranes $1, 2, \ldots, m$ of μ;
- At any time during the computation, a configuration of Π can be represented by a tuple α in N^{mk}, where the (i, j) component corresponds to the multiplicity of symbol a_j in membrane i.
- C_{start} and C_{stop} are recursive subsets of N^{mk} (i.e., they are Turing machine computable or, equivalently, computable by multicounter machines).

[1] The definitions have their roots in the biological observation that many times two chemicals pass at the same time through a membrane, with the help of each other, either in the same direction, or in opposite directions; in the first case we say that we have a *symport*, in the second case we have an *antiport* (we refer to [2] for details).

For a symport rule (x, in) or (x, out), we say that $|x|$ is the *weight* of the rule. The *weight* of an antiport rule $(x, out; y, in)$ is $\max\{|x|, |y|\}$. The rules from a set R_i are used with respect to membrane i as explained above. In the case of (x, in), the multiset of objects x enters the region defined by the membrane, from the surrounding region, which is the environment when the rule is associated with the skin membrane. In the case of (x, out), the objects specified by x are sent out of membrane i, into the surrounding region; in the case of the skin membrane, this is the environment. The use of a rule $(x, out; y, in)$ means expelling the objects specified by x from membrane i at the same time with bringing the objects specified by y into membrane i. The objects from E (in the environment) are supposed to appear in arbitrarily many copies since we only move objects from a membrane to another membrane and do not create new objects in the system, we need a supply of objects in order to compute with arbitrarily large multisets. The rules are used in the non-deterministic maximally parallel manner specific to P systems with symbol objects: in each step, a maximally parallel multiset of rules is used.

In this way, we obtain transitions between the configurations of the system. A configuration is described by the m-tuple of multisets of objects present in the m regions of the system, as well as the multiset of objects from $V - E$ which were sent out of the system during the computation. It is important to note that such objects appear only in a finite number of copies in the initial configuration and can enter the system again (knowing the initial configuration and the current configuration in the membrane system, one can know precisely what "extra" objects are present in the environment). On the other hand, it is not necessary to take care of the objects from E which leave the system because they appear in arbitrarily many copies in the environment as defined before (the environment is supposed to be inexhaustible, irrespective how many copies of an object from E are introduced into the system, still arbitrarily many remain in the environment). The initial configuration is $\alpha_0 = (w_1, \ldots, w_m)$. Note that $(w_1, ..., w_m) = (s_{11}, s_{12}, ..., s_{1k}, ...s_{m1}, s_{m2}, ...s_{mk})$, i.e., a tuple in N^{mk}. A sequence of transitions is called a computation.

Let us now describe the way this systems "outputs" the result of its computation: when the system enters some configuration α satisfying C_{start}, we start a counter t that is incremented each time the simport/antiport rules are applied in the nondeterministic parallel manner. At some point, when the system enters some configuration β satisfying C_{stop}, we stop incrementing t, and the value of t represents the output of the computation[2]. If the system never reaches a configuration in C_{start} or in C_{stop}, then we consider the computation unsuccessful, no output is associated with the computation of the system in that case. The set of all such t's (computed as described) is denoted by $N(\Pi)$. The family of all sets $N(\Pi)$ computed by systems Π of degree at most $m \geq 1$, using symport rules of weight at most p and antiport rules of weight at most q, is denoted by $NTP_m(sym_p, anti_q)$ (we use here similar notations with the definitions from [15]).

[2] By convention, in the case when a configuration α is reached that satisfies both C_{start} and C_{stop}, then we consider that the system has computed the value 0.

We emphasize the implicit fact in the definition of Π, that we assume that C_{start} and C_{stop} are recursive. Some interesting cases are when C_{start} and C_{stop}:

– are exactly N^{mk} (i.e., they are *trivial*, as they consist of all the tuples);
– are computable by deterministic multicounter machines (i.e., recursive);
– are Presburger relations.

Details about P systems with symport/antiport rules can be found in [15]; a complete formalization of the syntax and the semantics of these systems is provided in [17] where reachability of symport/antiport configurations was discussed.

After defining the new way of considering the time of a computation performed by such a system it is a natural question to ask whether the new definition is powerful enough so that it can carry out universal computation. First we mention a recent result for a very special case where C_{start} consists only of the initial configuration of the system, and C_{stop} consists of all halting configurations. For this case, it was shown in [6] that the set of all times (i.e., intervals between the initial configuration and halting configurations) is recursive making such systems not universal. In light of this result, the proof of universality for our system becomes an interesting contrast. We note that the result in [6] is also in surprising contrast to various results in the literature, where most of the P systems defined so far have been proven to be powerful enough to be universal. We expected to obtain a universality result using similar techniques used for proving the universality of "regular" symport/antiport P systems, and indeed, as the following theorem shows, we are able to prove the universality of timed symport/antiport systems with 3 membranes and symport/antiport rules of minimal weight, as well as universality of one membrane and antiport of weight 2.

Theorem 1. *Using minimal restrictions*[3] *on the multiplicities of the objects for the* C_{start}*,* C_{stop} *rules we have* $NRE = NTP_m(sym_r, anti_t)$*, for* $(m, r, t) \in \{(1, 0, 2), (3, 1, 1)\}$.

Proof. We use a modification of the construction described in [20] and used for proving $NRE' = NOP_3(sym_1, anti_1)$. It is worth noting that the result from [20] is computing $NRE' = NRE - \{0, 1, 2, 3, 4\}$, thus the best result for regular symport/antiport is still considered to be the one from [9] where it was shown that $NRE = NOP_4(sym_1, anti_1)$. With the help of the timed symport/antiport we will prove that three membranes and minimal symport/antiport are universal by computing exactly NRE.

We will give in the following our construction following the ideas from [20] and [8]. We consider a counter automaton [13] a construct $M = (Q, C, R, q_0, f)$

[3] By minimal restrictions on multiplicities we mean the fact that for each object and each membrane, the C_{start}, C_{stop} rules will impose either a fixed multiplicity (for example 5) or not impose any restrictions for the object. For example C_{start} containing $s_{2,3} = 5$ means that the third object has to appear in exactly 5 copies in membrane 2. On the other hand, by $s_{6,2} = i,\ i \geq 0$ we mean that the second object can have any multiplicity in the membrane 6.

where C is the set of $n+1$ counters denoted by $c_0, \ldots, c_n$ and c_0 is the output counter of the automaton. We will construct a timed symport/antiport system Π that generates the set of numbers $L(M)$ as follows:

$\Pi = (V, [_1[_2[_3\ ]_3]_2]_1, w_1, w_2, w_3, E, R_1, R_2, R_3, C_{start}, C_{stop})$, where:

$$\begin{aligned}
V &= \{I_1, \overline{I_1}, I_2, I_3, I_4, \infty_1, \infty_2, \infty_3, \infty_4, P, t, t', t'', \overline{t}, \overline{t}', \overline{t}''\} \cup \{c_i, 0_i \mid \\
&\quad 0 \le i \le n\} \cup \{qr, qr' \mid (q \to r, X) \in R \text{ for } X \in \{i+, i-, i=0, \lambda\}\},\\
w_1 &= I_1\overline{I_1}, I_2I_3\infty_1\infty_2\infty_4^2, \qquad w_2 = \infty_1\infty_2\infty_3^2\overline{t}P0_00_1\ldots0_n, \qquad w_3 = t',\\
E &= \{qr, qr' \mid (q \to r, X) \in R \text{ for } X \in \{i+, i-, =0, \lambda\}\} \cup \{I_4, \overline{t}', \overline{t}''\}\\
&\cup \{c_i \mid 0 \le i \le n\},\\
R_i &= R_i^{ini} \cup R_i^{sim} \cup R_i^{timer}, \text{ for } 1 \le i \le 3 \text{ where } R_i^{\tau},\\
&\quad \text{where } \tau \in \{ini,\ sim,\ timer\} \text{ are defined as in the following:}\\
R_1^{ini} &= \{(qr', in; I_1, out) \mid (q \to r, X) \in R,\ X \in \{i+, i-, =0, \lambda\}\} \cup \{(I_1, in)\}\\
&\cup \{(c_j, in; \overline{I_1}, out) \mid 0 \le j \le n\} \cup \{(I_4, in; I_1, out), (\overline{I_1}, in), (I_3, out)\},\\
R_2^{ini} &= \{((qr', in; I_2, out) \mid (q \to r, X) \in R, \text{ for an operation } X\} \cup \{(I_2, in),\\
&\quad (I_3, in; \infty_2, out), (I_4, in; I_3, out), (I_1, in; I_4, out), (\infty_4, in; \infty_4, out),\\
&\quad (\overline{I_1}, in; I_5, out), (\infty_4, in; I_4, out), (\infty_1, in; \infty_1, out), (\infty_2, in; \infty_2, out)\},\\
R_3^{ini} &= \{(qr', in; I_3, out) \mid (q \to r, X) \in R, \text{ for some } X\} \cup \{(I_3, in),\\
&\quad (\infty_3, in; I_3, out), (I_1, in; I_5, out), (I_2, in; I_1, out), (\infty_3, in; \infty_3, out)\},\\
R_1^{sim} &= \{(q_0q, in; I_5, out), (rs, in; qr', out) \mid q_0, q, r, s \in Q \text{ and } (q_0 \to q, X),\\
&\quad (q \to r, Y), (r \to s, Z) \in R \text{ for some counter operations } X, Y, Z\},\\
R_2^{sim} &= \{(rs, in), (c_i, in; rs', out) \mid \text{for } (r \to s, i+) \in R\} \cup \{(rs, in; c_i, out),\\
&\quad (rs', out) \mid (r \to s, i-) \in R\} \cup \{(rs, in), (rs', out) \mid (r \to s, \lambda) \in R\}\\
&\cup \{(rs, in; 0_i, out), (0_i, in; c_i, out), (0_i, in; rs', out) \mid (r \to s, i=0) \in R\},\\
R_3^{sim} &= \{(rs, in; rs', out) \mid (r \to s, X) \in R, \text{ for a counter operation } X\},\\
R_1^{timer} &= \{(t, in; qf', out) \mid \text{for } f \text{ the final state}\}\\
&\cup \{(\overline{t}', in; \overline{t}, out), (t'', in; t', out), (\overline{t}'', in; t'', out)\},\\
R_2^{timer} &= \{(t, in; \overline{t}, out), (\overline{t}', in; t', out), (\overline{t}'', in)\},\\
R_3^{timer} &= \{(t, in; t', out), (\overline{t}', in), (\overline{t}'', in), (P, in; \overline{t}', out), (c_0, in; \overline{t}', out),\\
&\quad (c_0, in, \overline{t}'', out)\}.
\end{aligned}$$

We will give the rules in C_{start} and C_{stop} in the following format: for each membrane i we will give a rule r_i^{α}, $\alpha \in \{start,\ stop\}$ associated with that membrane as a sequence of letters each having their multiplicity expressed as the exponent; they will give the exact multiplicities of the objects needed to reach the respective configuration. If an object is not "mentioned" for such a rule associated with a membrane, then we assume that there is no restriction on the multiplicity of that object in that membrane to satisfy the rule C_{α}. If an object has no exponent, then we assume it has to appear in the configuration exactly one time (i.e. it is considered to have exponent 1). We are now ready to give the *start* and *stop* rules for configurations:

$$C_{start} : r_1^{start} = \infty_1 \infty_2^2 \infty_4^2;\ r_2^{start} = \infty_1 \infty_3^2 \bar{t}'';\ r_3^{start} = c_0^0 t P \bar{t}'.$$

$$C_{stop} : r_1^{stop} = r_3^{stop} = \lambda;\ r_2^{stop} = c_0^0.$$

We will explain briefly the work of the system: from the initial configuration Π will go through three phases of the computation; in the first phase (*Initialization*) the system will bring in from the environment an arbitrary number of objects qr'[4] and c_j that will be used later in the simulation phase. Most of the rules are defined for the initialization phase; such a big number of rules was needed to ensure that only if the system is following a "correct" path in the computation a result is produced. The next phase is the actual *simulation* of the counter automaton (with the use of the objects brought in the system in the previous phase); the two major phases mentioned before are similar to the proofs from [3], [11], [4], [9], [20], the reader can see a detailed explanation of the usage of the rules from R_i^{ini} and R_i^{sim} in [20]. We now pass to describing the final stage of the simulation, the actual output of the contents for the counter c_0 using the time between a configuration α satisfying the rule C_{start} and another configuration β satisfying the rules from C_{stop}.

It is easy to see (for more details we refer to [20]) that, only in the case of a successful simulation, the repartition of the objects in the system will satisfy the rule $C_{end\_sim} : r1 = \infty_1 \infty_2^2 \infty_4^2 F,\ r_2 = \infty_1 \infty_3^2 \bar{t} P,\ r_3 = t'$ where $F \in \{qf' \mid q \in Q\}$, configuration that has also the property that in membrane 2 there are i copies of the object c_0 where i is the actual result of the computation.

It is clear that we need to go from C_{start} to C_{stop} in exactly i steps. To do this, we perform a few steps of "pre-work" by bringing in the second membrane $\bar{t}'$ and then also $\bar{t}''$, that will move the objects c_0 into membrane 3 one copy for each maximally parallel application of the rules.

Let us describe the "flow" of time in the system starting the end of the simulation phase (the object qf' reaches membrane 1). In that moment qf' is replaced with t in the membrane 1 by the rule $(t, in; qf', out) \in R_1$ and then t enters membrane 2 and sends out in membrane 1 $\bar{t}$: $(t, in; \bar{t}, out) \in R_2$. At the next step two rules can be applied: $(\bar{t}', in; \bar{t}, out)$ in membrane 1 and $(t, in; t', out)$ in membrane 3. Next $\bar{t}'$ moves into membrane 2 while t' reaches membrane 1: $(\bar{t}', in; t', out) \in R_2$ so that at the next step $\bar{t}'$ finally arrives in membrane 3 by $(\bar{t}', in) \in R_3$. During this last step t' is replaced by the rule t'': $(t'', in; t', out) \in R_1$ applicable in membrane 1. Since $\bar{t}'$ is used to move the c_0 objects from membrane 2 into membrane 3, it can start doing this by using the rule $(c_0, in; \bar{t}', out) \in R_3$, but this would mean that the system would never reach a configuration satisfying C_{start} since at least one copy of c_0 would be present in membrane 3 and would never be removed. Instead, we can use the rule $(P, in; \bar{t}', out) \in R_3$ and bring in membrane 3 the symbol P[5] so that $\bar{t}''$ can

[4] We note that qr' is a single object "keeping track" of two different states $q, r \in Q$.

[5] The symbol P is used as a padding symbol.

finally come in membrane 2 and start the timer (the rule C_{start} is satisfied by the current configuration). One can note that both $\bar{t}'$ and $\bar{t}''$ can move exactly one copy of c_0 from membrane 2 into membrane 3, and then re-enter membrane 3 so that they can perform this work once more at the next step. Since one copy is moved for each step, the number of steps from C_{start} and C_{stop} is exactly the multiplicity of the symbol c_0 in membrane 2. This in fact means that we proved that $NRE = NTP_3(sym_1, anti_1)$. In the following we prove the second part of the theorem, that one membrane and antiport of size two and no symport are enough for universality:

To prove that $NRE = NTP_1(sym_0, anti_2)$, we follow the constructions from [8] or [7] where a similar result for "regular" simport/antiport P systems was obtained. The unique membrane will start with the start state as its only object in the initial configuration, and the work of the counter automaton can be simulated using the antiport rules in the following way:

For a rule $(p \to q, \lambda) \in R$ we will have in our timed P system the rule $(q, in; p, out)$; for an increment instruction on the counter c_i: $(p \to q, i+)$ we will add the following antiport rule to R_1: $(qc_i, in; p, out)$. The decrement instruction can only be applied if the counter is non zero, $(p \to q, i-)$ is simulated by $(q, in; pc_i, out)$. Finally, $(p \to q, i = 0)$ is simulated by the rules $(q'\bar{i}, in; p, out)$; $(\infty, in; \bar{i}c_i, out)$, $(q'', in; q', out)$, and $(q, in; q''\bar{i}, out)$ in three steps: first we replace p by q' and $\bar{i}$, then $\bar{i}$ checks whether the register i is empty or not; if nonempty, the special marker ∞ is brought in and the computation cannot continue; but in the case when the register was empty the computation can continue by expelling the two symbols q'' and $\bar{i}$ together to bring in the next state q.

It is clear now that the register machine is simulated in this way only by using antiport rules of weight 2[6]. When the final state appears as the current state of the simulation it is time to start "counting" the result; the rule C_{start} can be defined as $r_1^{start} = \infty^0 f$. The rule $(f, in; fc_0, out)$ will expel one symbol c_0 at a time, thus if we define the rule C_{stop} ro be $r_1^{stop} = fc_0^0$ we will have exactly i steps between C_{start} and C_{stop}, where i is the multiplicity of the symbol c_0 (i.e. the contents of the output register) in the system. Following the previous discussion the equality $NRE = NTP_1(sym_0, anti_2)$ was shown, which completes the proof. □

We will consider next other properties of the newly defined model of P systems with time, such as the possibility of simulating timed systems by using normal symport/antiport systems. We will also consider the case when the start/stop configurations do not have any constrains on the object multiplicities, etc.

2.1 Other Results for Timed Simport/Antiport Systems

The following result shows that a timed system can be simulated by a "time-less" system of the same type.

[6] The result can be strengthened in the following way: the construction works even if we only use antiport rules of dimensions $(1, 2)$ or $(2, 1)$ by adding to the only two rules of dimension $(1, 1)$ some padding symbols. For example the rule $(q'', in; q', out)$ can be padded with the extra symbol P in this way $(q''P, in; q', out)$.

Theorem 2. *For every timed symport/antiport system Π we can effectively construct a (regular) symport/antiport system Π' which computes $N(\Pi)$.*

Proof. We start with a timed symport/antiport system Π and proceed to show how to compute $N(\Pi)$. Assume that Π has m membranes and an alphabet V with k symbols. Let α_0 be the initial configuration of Π. Let M_{start} and M_{stop} be deterministic multicounter machines which compute the relations C_{start} and C_{stop}, respectively.

Our procedure will be implemented on a nondeterministic multicounter (or register) machine M. M will have a set C of mk counters. There is a special counter, called T, which will be the timer. We also need counters to simulate M_{start} and M_{stop}. In addition, there are other counters, called D counters, whose use will be explained later. Initially, all the counters are zero.

1. M starts by storing the initial configuration α_0 in the set of counters C. (Since the initial configuration is fixed, this can be incorporated in the finite-state control of M).
2. M nondeterministically selects a maximally parallel multiset of rules applicable to the configuration represented in C, collectively storing the "changes" in the multiplicities of the symbols in the different membranes resulting from the application of the rules in the auxiliary set of counters D. We will explain the details of how this is done later.
3. Using D, M updates C, and resets the counters in D to zeros.
4. M nondeterministically guesses to either go back to step 2 or proceed to the next step.
5. (When M enters this step, it is guessing that the configuration α represented in the C counters is in C_{start}). M simulates M_{start} and checks that α is indeed in C_{start}. If so, M proceeds to the next step; otherwise, M halts in a non-accepting state.
6. As in step 2, M nondeterministically selects a maximally parallel multiset of rules applicable to the configuration represented in C, storing the changes in the multiplicities of the symbols in the different membranes resulting from the step in in the auxiliary set of counters D.
7. Using D, M updates C, resets the counters in D to zeros, and increments the counter T by 1.
8. M nondeterministically guesses to either go back to step 6 or proceed to the next step.
9. M simulates M_{stop} to check that the configuration β represented in the C counters is in C_{stop}. If so, M halts in an accepting state; else it halts in a non-accepting state. (Clearly, when M halts in an accepting state, the value of counter T is the number of steps Π took to reach configuration β from α).

Since steps 6-9 are similar to steps 2-5, we just describe the details of how M carries out step 2.

For every membrane i and the (unique) membrane j enclosing it, define two sets of counters: The first set consists of counters $d_{(i,j,a_1)}, ..., d_{(i,j,a_k)}$, and they will keep track of the multiplicities of the objects that are moved from membrane

i to membrane j as a result of the application of the rules in membrane i (as described below). The other set of counters $d_{(j,i,a_1)}, ..., d_{(j,i,a_k)}$ will keep track of the multiplicities of objects that are moved from membrane j to membrane i. These sets of counters will be called D counters. At the start of step 2, the D counters are reset to zero. Let Q be the set of all rules in the membrane structure.

(a) M nondeterministically picks a rule r in Q. Note that r belongs to a unique membrane, say i. First assume that $i \neq 1$ (i.e., not the skin membrane).
(b) Clearly, r is of the form $(u, out; v, in)$, where u or v, but not both, can be λ (the null string).
(c) Let j be the membrane directly enclosing membrane i. M checks if r is applicable by examining the contents of the counters in C corresponding to the symbols in membrane i and the contents of the counters corresponding to the symbols in membrane j, decrementing these counters appropriately, and then updating the D counters for the pair (i, j) as a result of the application of rule r. M then goes to step (a).
If r is not applicable, then M deletes r from Q. If Q is not empty, M goes to the step (a); otherwise, M has applied a maximal set of rules, and the counters in D can now be used to update the values of the counters in C.

For the case $i = 1$, the enclosing membrane is the environment, which has an abundance of each symbol and, hence, M does not have to keep track of the multiplicities of the symbols in the environment. Note also that multiplicity of each symbol in $V - E$ is bounded and its distribution in the membranes and the environment (although is changing during the computation) can be recorded in the finite-state control of M.

From the discussion above and the fact that a multicounter machine can effectively be simulated by a symport/antiport system, the theorem follows. □

For the trivial case when there are no constraints on the multiplicities of the objects in the membranes, we have:

Theorem 3. *For every timed symport/antiport system Π, with $C_{start} = C_{stop} = N^{mk}$, $N(\Pi)$ is recursive.*

Proof. The idea is the following. Given n, to determine if n is in $N(\Pi)$, simulate all computation paths of Π starting from its initial configuration (note that, in general, there may be several paths because the system is nondeterministic). Use a separate counter for each path to count the number of maximally parallel steps in each path. If there is a path with n steps, then n is in $N(\Pi)$. If each path leads to a halting configuration before n steps, then n is not in $N(\Pi)$. □

Now, from Theorem 1, $N(\Pi)$ is recursive for a timed symport/antiport system even when C_{start} and C_{stop} are very simple cases of Presburger relations. However, from Theorem 3, if C_{start} and C_{stop} are the trivial relations, $N(\Pi)$ is recursive. It follows that the only cases when $N(\Pi)$ would be recursive is when Π and C_{start} and C_{stop} are restricted. An interesting case is when Π is a timed symport/antiport system which operates in such a way that *no* symbol is exported into the environment (thus there are no rules of the form (u, out) and $(u, out; v, in)$ in the skin membrane). Call this system a restricted symport/antiport system. This type of (un-timed) system was studied in [17]. We can show the following (due to the space restrictions we leave the proof to the reader).

Theorem 4. *Let Π be a restricted timed symport/antiport system and C_{start} and C_{stop} be Presburger relations. Then $N(\Pi)$ can be accepted by a deterministic polynomial-time multicounter machine. (This means that the multicounter machine, when given n, can decide whether or not n is in $N(\Pi)$ in time polynomial in n).*

3 A P System Model of Timed Rules and Combinatorial Gene Expression

The goal of this section is to define a P system model that is close to the biology of the cell and, at the same time, keep some of the features of the P systems so that it can be studied with the now widely used mathematical tools for P systems. With a similar goal in mind we have recently defined a successful model in [15] which has been adopted as one of the natural/biological models of P systems by the research community; the model has become one of the major paradigms in the field.

We now extend the model proposed in [15] with several new ideas: different reactions can take different amounts of time; objects in the system can bind/dissociate according to their physical properties (3D shape, polarities); the cell contains its genetic material, enabling it to produce new objects according to the blueprints provided in genes.

Another interesting modeling effort in the direction of defining more realistic, i.e. more bio-compatible P systems was reported recently in the membrane computing annual conference, where two papers [5], [14] were suggesting approaches to make the P systems "time independent". Both the authors take in consideration rules in the P system that can take various amounts of time, in contrast with the other definitions of P systems that are modeling the rules as taking each 1 clock-cycle.

Our proposed model is different from the model in [5] or [14] by the fact that we will have in the system the idea of binding two molecules together. We are also interested in models that "behave" as close as possible to reality, in contrast with the aforementioned papers that were focussing on finding systems "time independent"; i.e. systems that have the same output even though the time associated with a rule is changing.

Another novelty of our proposed model is the introduction of the genetic material: one of the regions of the P system will be labeled *nucleus*, and will contain, among other things, the genes of that cell. As far as we know, there are no other

P system models that are describing the interactions between various molecules in the system and genes; especially gene activation process, gene activators, gene repressors, etc.

The genes will be denoted by G_1, G_2, G_3, $\ldots$, G_n, and they will be either activated or de-activated. Since it is still an ongoing debate in the biology community about how exactly is the mRNA built from an activated gene, we chose to model the process in the following way: genes are by default deactivated, they become activated only when some specific activator molecules bind to a gene. In that moment the gene is activated, and the mRNA is produced and sent to cytoplasm to be translated into several copies of the protein. After all this process takes place, we consider that the gene becomes deactivated and some of its activator molecules (or all) have left the nucleus. If more activator molecules are/were present in the nucleus, then at the next clock-cycle they can start binding to the gene, and the gene is activated once more.

The model will contain the activator rules as well as repressor rules for each gene plus, when activated, the gene will be able to produce new objects in the system.

Definition 1. *A genetic P system is a construct* $\Pi = (V, \mu, G, w_{cyt}, w_{ER}, w_{nucleus}, R_{cyt}, R_{ER}, R_{nucleus})$, *where*

- V *is the alphabet of* Π *representing the set of all possible molecules that can appear in the system.*
- μ *gives the membrane structure of the system; the plasma membrane (labeled* `cyt`*) contains two different sub-regions labeled with* `ER` *(for endoplasmic reticulum) and* `nucleus`*. In standard membrane systems notation* μ *is written as* $\mu = [_{\mathtt{cyt}}[_{\mathtt{ER}}\ ]_{\mathtt{ER}}\ [_{\mathtt{nucleus}}\ ]_{\mathtt{nucleus}}]_{\mathtt{cyt}}$.
- G *is the set of genes for the cell.*
- w_{cyt}, w_{ER}, $w_{nucleus}$ *are words over* V *that represent the initial multiplicities of the objects in their corresponding regions in the system. Please note that in the initial configuration we assume all the objects in the system to be "unbound" with any other object. We will call from now on as objects/molecules elements from* V *and also complexes of bound together elements from* V. *They will be written in the form* $\langle XYZ \rangle$ *when* X, Y, $Z \in V$; *thus* $\langle XYZ \rangle$ *is a single object in the system.*
- R_{cyt}, R_{ER}, $R_{nucleus}$ *are finite sets of rules associated with each of the three regions defined by the P system. We will describe in the following the types of rules that can be found in each of the three sets of rules.*

The rules are of four different categories: general rules (g1, g2, g3), cytoplasm rules (c1, c2), endoplasmic reticulum rules (e1, e2, e3, e4) and nucleus rules (n1, n2, n3).

The general rules are specifying the types of rules that can appear in any region of the cell; they model the binding/unbinding of molecules (g1, g2) and the catalytic reactions from the cell (g3).

The cytoplasm rules can only be applied in cytoplasm; they are modeling the creation of new proteins from mRNA by the ribosomes (c1) and the destruction of proteins by the proteases (c2).

In the endoplasmic reticulum (ER) we have rules that model the movement of objects between CYT and ER. We model the work of ion channels (e1), uniport (e2), symport/antiport (e3/e4).

The last type of rules are the ones that can only appear in nucleus, they model the work of activators/repressors binding to genes (n1, n2) and the transcription of a gene followed by the expel of the mRNA into the cytoplasm (n3) so that the protein-building mechanism (c1) can start.

The general rules in the cytoplasm, endoplasmic reticulum and nucleus will have the forms:

g1. association (binding) rules: $\langle X\rangle + \langle Y\rangle \rightarrow_t \langle XY\rangle$ where $X, Y \in V^+$ and t specifies the number of clock-cycles it takes for the binding to take place for the specified molecules. It must be stressed the fact that the product $\langle XY\rangle$ is the same with the product $\langle YX\rangle$; we will write the products in the lexicographic order.
g2. dissociation (unbinding) rules: $\langle XY\rangle \rightarrow_t \langle X\rangle + \langle Y\rangle$ where $X, Y \in V^+$ and t specifies the number of clock-cycles required for the unbinding operation.
g3. catalysis rules: $\langle X\rangle + \langle Y\rangle \rightarrow_t \langle X\rangle + \langle Z\rangle$ where $X, Y, Z \in V^+$ and t specifies the number of clock-cycles required for the enzyme $\langle X\rangle$ to perform the catalysis.

We will apply the previous rules in a nondeterministically parallel manner with the only remark that the binding rules have higher priority than other rules such as the catalatic rules, ion channel rules, etc.; in this way the model accounts (among other things) for the allosteric changes of enzymes.

The following types of rules will be associated only with the cytoplasm:

c1. creation of proteins by the ribosome: $\langle A^n\rangle \rightarrow_t \langle A^{n-1}\rangle + \langle A\rangle_l$, where $l \in \{$*here, in-ER, in-nucleus*$\}$ for all $1 \leq n$, and $A^1 = A$.
c2. destruction of objects: $\langle PY_u\rangle \rightarrow_t \langle P\rangle$, where $P, Y_u \in V^+$ and P is a protease and Y_u is a protein marked for destruction by ubiquitin.

The following types of rules will be associated only with the endoplasmic reticulum:

e1. ion channels: $\langle Ion\rangle \rightarrow_{t_1} \langle \overline{Ion}\rangle$, $\langle \overline{Ion}\rangle \rightarrow_{t_2} \langle Ion\rangle$, $\langle Ion\rangle + \langle X\rangle \rightarrow_{t_3} \langle IonX\rangle$, $\langle IonX\rangle \rightarrow_{t_4} \langle Ion\rangle + \langle X\rangle_{in/out}$ where $Ion, \overline{Ion}, X \in V$. The rules defined for the ion channels take in consideration the fact that the channels have a periodical transition between the on and off configurations.
e2. uniport: $\langle Uni\rangle + \langle X\rangle \rightarrow_t \langle Uni\rangle + \langle X\rangle_{out}$, or $\langle Uni\rangle + \langle X\rangle_{cyt} \rightarrow_t \langle Uni\rangle + \langle X\rangle_{in}$, where $Uni, X \in V$ and $\langle X\rangle_{cyt}$ means that the object X is in that moment in the cytoplasm.
e3. symport: $\langle Sim\rangle + \langle XY\rangle \rightarrow_t \langle Sim\rangle + \langle X\rangle_{out} + \langle Y\rangle_{out}$, or $\langle Sim\rangle + \langle XY\rangle_{cyt} \rightarrow_t \langle Sim\rangle + \langle X\rangle_{in} + \langle Y\rangle_{in}$, where $Sim, X, Y \in V$ and $\langle XY\rangle_{cyt}$ means that the objects are in the cytoplasm.
e4. antiport: $\langle Anti\rangle + \langle X\rangle + \langle Y\rangle_{cyt} \rightarrow_t \langle Anti\rangle + \langle X\rangle_{out} + \langle Y\rangle_{in}$ where we have that $Anti, X, Y \in V$ and the subscript *cyt* specifies that the respective molecule is in the cytoplasm.

The following types of rules will be associated only with the nucleus:

n1. activator/repressor binding to a gene: $\langle A\rangle + \langle G_i\rangle \rightarrow_t \langle AG_i\rangle$ for $A \in V^+$ and $G_i \in G$.
n2. more activators binding to the gene G_i: $\langle A\rangle + \langle BG_i\rangle \rightarrow_t \langle ABG_i\rangle$ for $A, B \in V^+$ and $G_i \in G$.
n3. gene activation and mRNA move to cytoplasm: $\langle XG_i\rangle \rightarrow_t \langle YG_i\rangle + \langle A^k\rangle_{out}$, where $X, Y \in V^*$, $A \in V$, $Y \subseteq X$ and k is the number of copies of the protein A (codified in the mRNA) that will be produced by the ribosomes in the cytoplasm.

One can note that the system is defined flexibly enough so that much more biological processes can be expressed using the given rules. For example, the phosphorylation reaction can be expressed with several objects in V using the catalytic reaction of the type g3: $\langle X\rangle + \langle Y\rangle \rightarrow_t \langle X\rangle + \langle Y_p\rangle$ and can be continued for several steps: $\langle X\rangle + \langle Y_p\rangle \rightarrow_{t'} \langle X\rangle + \langle Y_{pp}\rangle$ by using other type g3 rules.

Remark. We assume that a complex of objects (several objects are bound together after a repeated use of rules of type g1) is working as a whole; thus if we have the complex object $\langle ABC\rangle$, then a rule defined only for $\langle AB\rangle$ cannot be applied to $\langle ABC\rangle$. This is due to the fact that the 3D shape of the complex $\langle AB\rangle$ can be changed dramatically by the binding with $\langle C\rangle$.

We note that the previous remark helps the rules n1, n2 simulate also the work of the gene regulation repressors, since the gene cannot be activated if the repressor is bound to it.

The rule allows for modeling the enzymatic regulation that takes place in cells: if the enzyme $\langle A\rangle$ is catalyzing the reaction from $\langle X\rangle$ into $\langle Y\rangle$ at some rate of 3 clock-cycles, we can write it as a rule of type g3: $\langle A\rangle + \langle X\rangle \rightarrow_3 \langle A\rangle + \langle Y\rangle$. Now, let us assume that the cell decides to down-regulate the enzyme to a ten times slower speed by using the molecule $\langle B\rangle$. In this case we would model the reactions using a binding rule $\langle A\rangle + \langle B\rangle \rightarrow_2 \langle AB\rangle$, and a catalytic rule $\langle AB\rangle + \langle X\rangle \rightarrow_{30} \langle AB\rangle + \langle Y\rangle$.

We now briefly discuss the different types of output for the new model. The first type of output of the system could be associated with the number of times each rule of the type n3 is applied for each of the genes contained in a predefined string $w \in G^*$ in a given amount of time. In other words, we are interested in a specific number of genes, and we are looking at how many times these genes have been activated. The system in this case can compute a vector of integer values; the number of components and the order of the components being given by the word w.

Another type of "termination" for the computation for such a system could be viewed as a combination of activated genes at a given moment as well as minimal multiplicities for several molecules in each component of the system. Such a configuration could be viewed as a "final state" of the machinery. In that moment one could use the first idea of the output of the system; i.e. counting the number of activations for particular genes.

Yet another idea is to consider as we have done for timed symport/antiport systems in Section 2 the time that passed between two distinguished configurations as the the result of the computation. This method of considering the computation of the system seems quite flexible and elegant. It does not require a cell to "accumulate" a large quantity of a specific molecule that would encode the output.

4 Final Remarks

In this paper, we considered (as in the previous papers that have been mentioned) time as an "active" participant in a computation. We also reviewed two known models – one using spiking neurons and the other, the time independent P systems. We then defined and obtained several results concerning some new models – the timed P system, P system with timed rules, and gene expression models. For the newly introduced timed P systems we improved or matched two of the best known results for "regular" symport/antiport P systems. We are currently working on proving the remaining two results ($NTP_3(sim_2, anti_0)$ and $NTP_2(sim_3, anti_0)$). It is worth noting that the new feature of outputting the result using time is more flexible than the previously considered methods, thus the previous results could be even improved by using completely different techniques that take advantage of the flexibility of the time as a framework of outputting the result of a computation.

Acknowledgements

We would like to acknowledge the fruitful discussions with B. Tănasă (MIT) and the insightful suggestions received from the anonymous referees. O. H. Ibarra gratefully acknowledges the support in part by NSF Grants CCR-0208595, CCF-0430945 and CCF-0524136; A. Păun gratefully acknowledges the support in part by LA BoR RSC grant LEQSF (2004-07)-RD-A-23 and NSF Grants IMR-0414903 and CCF-0523572.

References

1. L.M. Adleman, Molecular Computation of Solutions to Combinatorial Problems, Science 266 (1994), 1021–1024.
2. B. Alberts, *Essential Cell Biology. An Introduction to the Molecular Biology of the Cell*, Garland Publ. Inc., New York, London, 1998.
3. F. Bernardini, M. Gheorghe, On the Power of Minimal Symport/Antiport, Workshop on Membrane Computing, A. Alhazov et al. (eds.), WMC-2003, Tarragona, July 17-22, 2003, Technical Report N. 28/03, Research Group on Mathematical Linguistics, Universitat Rovira i Virgili, Tarragona (2003), 72–83.
4. F. Bernardini, A. Păun, Universality of Minimal Symport/Antiport: Five Membranes Suffice, WMC03 revised papers in *Lecture Notes in Computer Science* 2933, Springer (2004), 43–54.

5. M. Cavaliere, Towards Asynchronous P Systems, *Pre-proceedings of the Fifth Workshop on Membrane Computing (WMC5)*, Milano (Italy), June 14-16, 2004, 161–173.
6. M. Cavaliere, R. Freund, Gh. Păun, Event–Related Outputs of Computations in P Systems, personal communication, (manuscript).
7. R. Freund, A. Păun, Membrane Systems with Symport/Antiport: Universality Results, in *Membrane Computing. Intern. Workshop WMC-CdeA2002, Revised Papers* (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds.), *Lecture Notes in Computer Science*, 2597, Springer-Verlag, Berlin, 2003, 270–287.
8. P. Frisco, J.H. Hogeboom, Simulating Counter Automata by P Systems with Symport/Antiport, in *Membrane Computing. Intern. Workshop WMC-CdeA2002*, revised papers (Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds.), *Lecture Notes in Computer Science*, 2597, Springer-Verlag, Berlin, 2003, 288–301.
9. P. Friso, About P Systems with Symport/Antiport, Second Brainstorming Week in Membrane Computing, Sevilla, February 2004, Technical Report 01/2004 of the Research Group in Natural Computing, University of Sevilla, Spain, 2004, 224–236.
10. J. Hopcroft, J. Ulmann, *Introduction to Automata Theory, Languages, and Computation*, Addison-Wesley, 1979.
11. L. Kari, C. Martin-Vide, A. Păun, On the Universality of P Systems with Minimal Symport/Antiport Rules, *Lecture Notes in Computer Science* 2950, Berlin, (2004), 254–265.
12. W. Maas: Computing with Spikes. *Spec. Iss. on Found. of Inf. Processing of TELEMATIK*, 8, 1 (2002), 32–36.
13. M.L. Minsky, Recursive Unsolvability of Post's Problem of "Tag" and Other Topics in Theory of Turing Machines, *Annals of Mathematics*, 74 (1961), 437–455.
14. D. Sburlan, Clock-free P Systems, *Pre-proceedings of the Fifth Workshop on Membrane Computing (WMC5)*, Milano (Italy), June 14-16, 2004, 372–383.
15. A. Păun, Gh. Păun, The Power of Communication: P Systems with Symport/Antiport, *New Generation Computing*, 20, 3 (2002) 295–306.
16. Gh. Păun: *Membrane Computing – An Introduction.* Springer-Verlag, Berlin, 2002.
17. Gh. Păun, M. Perez-Jimenez, F. Sancho-Caparrini, On the Reachability Problem for P Systems with Symport/Antiport, *Proc. Automata and Formal Languages Conf.*, Debrecen, Hungary, 2002.
18. Gh. Păun, Further Twenty-six Open Problems in Membrane Computing, the Third Brainstorming Meeting on Membrane Computing, Sevilla, Spain, February 2005.
19. G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages*, 3 volumes, Springer-Verlag, Berlin, 1997.
20. G. Vazil, On the Size of P Systems with Minimal Symport/Antiport, Preproceedings of International Workshop WMC04, Milan, June 2004, 422–431.