A Case Study in (Mem)Brane Computation: Generating Squares of Natural Numbers

Nadia Busi[1] and Miguel A. Gutiérrez-Naranjo[2]

[1] Dipartimento di Scienze dell'Informazione - Università di Bologna
Mura Anteo Zamboni 7, I-40127 Bologna, Italy
busi@cs.unibo.it
[2] Dpto. de Ciencias de la Computación e Inteligencia Artificial
Universidad de Sevilla
Avda. Reina Mercedes s/n, 41012, Sevilla, Spain
magutier@us.es

Abstract. The aim of this paper is to start an investigation and a comparison of the expressiveness of the two most relevant formalisms inspired by membranes interactions, namely, P systems and Brane Calculi. We compare the two formalisms with respect to their ability to act as generator devices. In particular, we show different ways of generating the set $\mathcal{L} = \{n^2 \mid n \geq 1\}$ in P systems and in Brane Calculi.

1 Introduction

Natural Computing studies new computational paradigms inspired from various well known natural phenomena in physics, chemistry, and biology. It abstracts the way in which nature computes, conceiving new computing models. There are several fields in Natural Computing that are now well established. Among them, we mention *Genetic algorithms* introduced by J. Holland [7] that is inspired by natural evolution and selection in order to find a good solution in a large set of feasible candidate solutions, *Neural Networks* introduced by W.S. McCulloch and W. Pitts [8] which is based on the interconnections of neurons in the brain, and *DNA-based* molecular computing, that was born when L. Adleman [1] published a solution to an instance of the Hamiltonian path problem by manipulating DNA strands in a lab.

This paper is devoted to a new field in Natural Computing. Starting from the structure and functioning of cells as living organisms able to process and generate information, two different branches of Natural Computing were recently initiated: *Membrane Computing* and *Brane Calculi*.

Membrane Computing was introduced by Gh. Păun in [9]; a comprehensive presentation[1] can be found at [11]. The devices of this model are called *P systems*. Roughly speaking, a P system consists of a membrane structure, in the compartments of which one places multisets of objects which evolve according to given rules in a synchronous non-deterministic maximally parallel manner.

[1] A layman-oriented introduction can be found in [10] and further bibliography at [14].

H.J. Hoogeboom et al. (Eds.): WMC 7, LNCS 4361, pp. 233–249, 2006.

Brane Calculi were introduced by L. Cardelli in [4] on the assumption that in living cells membranes are not merely containers, they are highly dynamic and participate actively in the cell life. In this way, computation happens on the membrane, not inside of it.

The first attempt of bridging the two research areas was made in [6] by the *fathers* of the two disciplines, L. Cardelli and Gh. Păun. As they point out, Membrane Computing and Brane Calculi *have different objectives and develop in different directions. While Membrane Computing tries to abstract computing models, in the Turing sense, from the structure and the functioning of the cell (...), Brane Calculi pay more attention to the fidelity to the biological reality (...).*

In that paper [6], four basic operations from Brane Calculi, namely, *pino*, *exo*, *mate* and *drip*, are expressed in terms on the Membrane Computing formalism and the Turing completeness of systems which use the *mate*, *drip* operations is shown. The Turing universality of Brane Calculi (in fact, by using only *phago* and *exo* operations) was proved in [3]. Recently, it has been proved that P systems with *mate* and *drip* operations and using at most five membranes during any step of a computation are universal (see [2]). This result improves a similar one from [6] were eleven membranes are used.

In some sense, in this paper we cross the bridge in the other way. Instead of expressing Brane Calculi operations in terms of the Membrane Computing formalism, we take a problem from computability, the generation of a set of numbers, and we show how it can be handled both in Membrane Computing and in Brane Calculi.

The paper is organized as follows: first the case study, i.e., the set $\mathcal{L} = \{n^2 \mid n \geq 1\}$ and some considerations with respect to the codifications are fixed in the next section. In Section 3, two different Membrane Computing devices that generate $\mathcal{L}$ are shown. Inspired on the second Membrane Computing design, two Brane Calculi devices that generate $\mathcal{L}$ are presented in Section 4. Some final remarks are presented in the last section.

2 The Case Study

Computational devices can be designed in order to perform different tasks. Among such tasks, they can be designed to solve decision problems (I_X, θ_X) where I_X is a language over a finite alphabet (whose elements are called *instances*) and θ_X is a total boolean function over I_X. In a more general case, the function is not boolean and the problem consists on the computation of a function f from I_X onto a general set S.

Another type of tasks is the generation of various sets (of numbers, vectors, strings, etc.). Due to the nondeterminism, several different computations are obtained and some piece of information is considered as the *output*. Collecting all (acceptable) outputs, we get a set (of numbers, vectors, strings, etc.) generated by the computation device.

In order to fix ideas, let us consider the case study used in this paper. We will consider the set $\{n^2 \mid n \geq 1\}$. For its generation, we will design appropriate devices in the computational models *Membrane Computing* and *Brane Calculi*. Such devices are non-deterministic and several computations can be performed from the starting point. In each device, a piece of information will be considered the *output* of the system. In the case of Membrane Computing, the *output* is codified as the number of objects inside a fixed membrane in a halting configuration. In the Brane Calculi device, the *output* is codified as the number of membranes of a specific kind that are present in the system in a halting computation. The set of all possible outputs of the device is exactly $\mathcal{L} = \{n^2 \mid n \geq 1\}$. In this way, $\mathcal{L}$ is the set generated by the device.

3 Membrane Computing

In Membrane Computing, many different types of rules and different semantics have been presented. The choice of these rules and semantics lead us to different models of P systems. In this section we present two P systems constructed in two different models that generate the set $\{n^2 \mid n \geq 1\}$.

In these examples several types of rules are used (O is the alphabet of *objects*, H is a finite set of *labels*, and λ is the empty string):

- *Object evolution rules* $[a \to v]_h$ where $h \in H$, $a \in O$, and v is a string over O describing a multiset of objects. They are associated with membranes and depending only on the label of the membrane. Using such a rule means that an object a evolves to the multiset v inside the membrane with label h.
- *Cooperation rules:* $[v \to w]_h$ where $h \in H$ and v, w are string over O describing a multisets of objects. This rule is similar to the previous one, but in this type, the rule is triggered by a multiset of objects whereas in an *object evolution rule* only one object is necessary for triggering it.
- *Dissolution rules:* $[a]_h \to b$ where $h \in H$, $a \in O$, $b \in O \cup \{\lambda\}$. The object a inside the membrane labeled with h produces the dissolution of the membrane and it is transformed into the object b. This object b together with the remaining objects in the membrane h are placed inside the surrounding membrane.
- *Send-in communication rules:* $a[\ ]_h \to [b]_h$ where $h \in H$, $a, b \in O$. An object a out of the membrane labeled with h is sent into the membrane and transformed into b.
- *Send-out communication rules:* $[a]_h \to [\ ]_h\, b$ where $h \in H$, $a, b \in O$. This is dual to the previous case. An object a inside the membrane labeled with h is sent out of the membrane and transformed into b.

Rules are applied according to the following principles:

- Rules are used as usual in the framework of Membrane Computing, that is, in a maximally parallel way. In one step, each object in a membrane can only be used for one rule (non-deterministically chosen when there are

several possibilities), but any object which can evolve by a rule of any form must do it (with the restrictions indicated below).
- If a membrane is dissolved, its content (multiset and internal membranes) becomes part of the immediately external one. The skin membrane is never dissolved.
- All the elements which are not involved in any of the operations to be applied remain unchanged.
- Several rules can be applied to different objects in the same membrane simultaneously. The exception are the division rules since a membrane can be dissolved only once.

In order to generate a set, an output membrane is fixed and the number of objects in it is counted when the system halts. The number of objects can vary from one computation to other due to the nondeterminism of the system. In the next examples, the set of numbers obtained in the output membrane, i.e., the generated set, is $\{n^2 \mid n \geq 1\}$.

3.1 Cooperation and Priorities

The first P system that we show is taken from [11] (p. 75) and it uses two of the most powerful features in P systems. The first one is the use of rules with *cooperation* between objects as described above. This type of rules are not triggered by the occurrence of only one object, but two or more objects are necessary in order to trigger the rule. The second feature is the priority among rules. In the general framework of Membrane Computing, if two rules can be applied, one of them is chosen in a non-deterministic way. If a priority between rules is added, we decrease the non-determinism, since we have a precedence between them.

With the notation fixed above, the P system is $\Pi = (O, H, \mu, w_1, w_2, w_3, 1, R)$ where $O = \{a, b, d, e, f\}$ is the set of objects, $H = \{1, 2, 3\}$ is the set of labels, $\mu = [\,[\,[\,]_3\,]_2\,]_1$ is the membrane structure, $w_1 = \emptyset$, $w_2 = \emptyset$, $w_3 = af$ are the multisets placed in the membranes at the starting point, 1 is the label of the *output membrane*, and R is the set of rules:

Rule 1: $[a \to ab]_3$ **Rule 5:** $[b \to d]_2$
Rule 2: $[a]_3 \to b$ **Rule 6:** $[d \to de]_2$
Rule 3: $[f \to ff]_3$ **Rule 7:** $[ff \to f]_2$
Rule 4: $[d]_1 \to d\,[\,]_1$ **Rule 8:** $[f]_2 \to \lambda$

with the priority

$$(\textbf{Rule 7: } [ff \to f]_2) > (\textbf{Rule 8: } [f]_2 \to \lambda)$$

Rules **1**, **3**, **5** and **6** are *object evolution rules*. Rule **7** is a cooperation rule: we need *two* objects f in order to trigger the rule. Rules **2** and **8** are *dissolution rules*. Finally, rule **4** is a *send-out communication rule*.

The computation is performed as follows. In the initial configuration we only have objects af in the membrane labeled with 3.

$$C_0 = [\,[\,[\,af\,]_3\,]_2\,]_1$$

Due to rule **3**, the object f deterministically evolves to ff. For the object a we have two possibilities: By application of rule **1**, the object a evolves to ab or by applying rule **2**, membrane 3 dissolves. If we iterate the use of rules **1** and **3**, after n steps, $n \geq 0$, we get n occurrences of b, one copy of a, and 2^n occurrences of f in membrane 3.

$$\begin{aligned} C_1 &= [\,[\,[\,abf^2\,]_3\,]_2\,]_1 \\ C_2 &= [\,[\,[\,ab^2f^4\,]_3\,]_2\,]_1 \\ &\dots \\ C_n &= [\,[\,[\,ab^nf^{2^n}\,]_3\,]_2\,]_1 \end{aligned}$$

If then rule **2** is chosen, the membrane labeled with 3 is dissolved *after* the evolution of f. With the dissolution, the 2^{n+1} copies of object f and the $n+1$ copies of b become occurrences of objects of membrane 2.

$$C_{n+1} = [\,[\,b^{n+1}f^{2^{n+1}}\,]_2\,]_1$$

In one step, the $n+1$ copies of b are transformed into $n+1$ copies of d by rule **5**, while the number of occurrences of f is halved.

$$C_{n+2} = [\,[\,d^{n+1}f^{2^n}\,]_2\,]_1$$

In the next step each occurrence of d introduces one occurrence of e and the number of occurrences of f is halved again.

$$C_{n+3} = [\,[\,d^{n+1}e^{n+1}f^{2^{n-1}}\,]_2\,]_1$$

After n applications of rule **7**, $[ff \to f]_2$, only one copy of object f is present in membrane labeled with 2. In the meantime, rule **6** is applied $n+1$ times in each step.

$$\begin{aligned} C_{n+4} &= [\,[\,d^{n+1}e^{2(n+1)}f^{2^n-2}\,]_2\,]_1 \\ C_{n+5} &= [\,[\,d^{n+1}e^{3(n+1)}f^{2^n-3}\,]_2\,]_1 \\ &\dots \end{aligned}$$

Following the priority relation, rule **7** $[ff \to f]_2$ is used as much as possible; when only one object f remains, rule **8** is used.

$$\begin{aligned} C_{2n+2} &= [\,[\,d^{n+1}e^{n(n+1)}f\,]_2\,]_1 \\ C_{2n+3} &= [\,d^{n+1}e^{(n+1)^2}\,]_1 \end{aligned}$$

With the dissolution of membrane 2, all the objects d become objects of membrane 1. In the next step, rule **4** is applied $n+1$ times and all copies of d are sent out to the environment.

$$C_{2n+4} = [\,e^{(n+1)^2}\,]_1\,d^{n+1}$$

No further step is possible and the computation stops. In the membrane labeled with 1 we have $(n+1)(n+1)$ copies of object e for some $n \geq 0$, hence the set generated is $\{n^2 \mid n \geq 1\}$.

3.2 A Simplified Solution

Now we present a *new* solution to the same problem. We do not use cooperation or priorities. Only send-in communication, dissolution and object evolution rules are applied. The design is based on the well-known property of natural numbers

$$\sum_{k=0}^{n}(2k+1) = (n+1)^2 \quad \text{for all } n \geq 0$$

The P system is the following: $\Pi = (O, H, \mu, w_e, w_r, w_s, r, R)$ with the set of objects $O = \{a, b, c, z\}$, the set of labels $H = \{e, r, s\}$, the membrane structure $\mu = [\,[\,]_e\,[\,]_r\,]_s$. The initial multisets are $w_e = a^2bz$, $w_r = \emptyset$ and $w_s = \emptyset$, i.e., the membranes s and r are empty and there exist two copies of a and one copy of b and z in the membrane e. The *output membrane* is labeled with r and the set of rules R is the following:

Rule 1: $[\,a \to ab\,]_e$ **Rule 5:** $[\,a \to \lambda\,]_s$
Rule 2: $[\,b \to bc\,]_e$ **Rule 6:** $[\,b \to \lambda\,]_s$
Rule 3: $[\,z \to z\,]_e$ **Rule 7:** $c\,[\,]_r \to [\,c\,]_r$
Rule 4: $[\,z\,]_e \to \lambda$

Note that the only non-determinism in this example is produced by the object z. This object can trigger two rules. The first one is $[\,z \to z\,]_e$ which represents that the object z inside the membrane e does not change. The second one is $[\,z\,]_e \to \lambda$ which means that the object z dissolves the membrane e. The collateral effect of the application of this rule is that the remaining objects in e are sent to s.

The initial configuration is $C_0 = [\,[\,a^2bz\,]_e\,[\,]_r\,]_s$. In the first step the two objects a evolve according to the rule **1**, $[\,a \to ab\,]_e$, and the object b evolves following the rule **2**, $[\,b \to bc\,]_e$. These evolutions are deterministic. For the object z we have two options, rules **3** and **4**. Let us suppose that z remains unchanged following rule **3**, $[\,z \to z\,]_e$. We obtain the configuration $C_1 = [\,[\,a^2b^3cz\,]_e\,[\,]_r\,]_s$. Let us suppose that in the next steps the object z does not dissolve the membrane e. We obtain $C_2 = [\,[\,a^2b^5c^4z\,]_e\,[\,]_r\,]_s$, $C_3 = [\,[\,a^2b^7c^9z\,]_e\,[\,]_r\,]_s$,... and in general, if the element z does not dissolves the membrane e, in the n-th ($n \geq 1$) step we reach the configuration

$$C_n = [\,[\,a^2b^{2n+1}c^{n^2}z\,]_e\,[\,]_r\,]_s$$

Let us now suppose that in the n-th step the object z dissolves the membrane e by using rule **4**. Since the dissolution is considered *after* the evolution of objects a and b, we reach the configuration

$$C_{n+1} = [\,a^2b^{2(n+1)+1}c^{(n+1)^2}z\,[\,]_r\,]_s \quad n \geq 0$$

One of the effects of the dissolution is that the objects a, b, and c are now in the membrane s. On one hand the rules $[\, a \to \lambda \,]_s$ and $[\, b \to \lambda \,]_s$ are triggered in the next step, so objects a and b disappear. On the other hand, objects c are in the region surrounding the membrane r, so the communication rule $c\,[\,]_r \to [\, c\,]_r$ are applied and all the elements c go into membrane r. In this way, the next configuration is $C_{n+2} = [\, [c^{(n+1)^2}\,]_r\,]_s$ with $n \geq 0$.

No more rules can be applied, so this is a halting configuration and we have computed the number n^2 with $n \geq 1$ (encoded by the elements c) in the output membrane.

4 Brane Calculi

In this section we tackle the problem of generating the set $\{n^2 \mid n \geq 1\}$ in Brane Calculi.

Brane Calculi [4] are a family of process calculi proposed for modeling the behavior of biological membranes. In a process algebraic setting, Brane Calculi represent an evolution of BioAmbients [12], a variant of Mobile Ambients [5] based on a set of biologically inspired primitives of interaction. The main novelty of Brane calculi consists in the fact that the active entities reside on membranes, and not inside membranes.

In this paper we are interested in the membrane operations of two basic instances of Brane calculi proposed in [4]: the Phago/Exo/Pino (PEP) and the Mate/Bud/Drip (MBD) calculi.

The interaction primitives of PEP are inspired by *endocytosis* (the process of incorporating external material into a cell by engulfing it with the cell membrane) and *exocytosis* (the reverse process). A relevant feature of such primitives is *bitonality*, a property ensuring that there will never be a mixing of what is inside a membrane with what is outside, although external entities can be brought inside if safely wrapped by another membrane. As endocytosis can engulf an arbitrary number of membranes, it turns out to be a rather uncontrollable process. Hence, it is replaced by two simpler operations: *phagocytosis*, that is engulfing of just one external membrane, and *pinocytosis*, that is engulfing zero external membranes.

The primitives of MBD are inspired by membrane fusion (mate) and fission (mito). Because membrane fission is an uncontrollable process that can split a membrane at an arbitrary place, it is replaced by two simpler operations: *budding*, that is splitting off one internal membrane, and *dripping*, that consists in splitting off zero internal membranes. An encoding of the MBD primitives in PEP is provided in [4].

4.1 Basic Brane Calculi: Syntax and Semantics

In this section we recall the syntax and the semantics of Brane Calculi [4]. A system consists of nested membranes, and a process is associated to each membrane.

Definition 1. *The set of systems is defined by the following grammar:*

$$P, Q \quad ::= \quad \diamond \mid P \circ Q \mid !P \mid \sigma(\!| P |\!)$$

The set of membrane processes is defined by the following grammar:

$$\sigma, \tau \quad ::= \quad 0 \mid \sigma|\tau \mid !\sigma \mid a.\sigma$$

Variables a, b range over actions that will be detailed later.

The term $\diamond$ represents the empty system; the parallel composition operator on systems is $\circ$. The replication operator ! denotes the parallel composition of an unbounded number of instances of a system. The term $\sigma(\!| P |\!)$ denotes the membrane that performs process σ and contains system P.

The term 0 denotes the empty process, whereas | is the parallel composition of processes; with $!\sigma$ we denote the parallel composition of an unbounded number of instances of process σ. Term $a.\sigma$ is a guarded process: after performing the action a, the process behaves as σ.

We adopt the following abbreviations: with a we denote $a.0$, with $(\!| P |\!)$ we denote $0(\!| P |\!)$, and with $\sigma(\!|\,|\!)$ we denote $\sigma(\!| \diamond |\!)$.

The structural congruence relation on systems and processes is defined as follows:[2]

Definition 2. *The structural congruence $\equiv$ is the least congruence relation satisfying the following axioms:*

$$\begin{array}{ll}
P \circ Q \equiv Q \circ P & \sigma \mid \tau \equiv \tau \mid \sigma \\
P \circ (Q \circ R) \equiv (P \circ Q) \circ R & \sigma \mid (\tau \mid \rho) \equiv (\sigma \mid \tau) \mid \rho \\
P \circ \diamond \equiv P & \sigma \mid 0 \equiv \sigma \\
 & \\
!\diamond \equiv \diamond & !0 \equiv 0 \\
!(P \circ Q) \equiv !P \circ !Q & !(\sigma \mid \tau) \equiv !\sigma \mid !\tau \\
!!P \equiv !P & !!\sigma \equiv !\sigma \\
P \circ !P \equiv !P & \sigma \mid !\sigma \equiv !\sigma \\
 & \\
0(\!| \diamond |\!) \equiv \diamond &
\end{array}$$

Definition 3. *The basic reaction rules are the following:*

$$(\texttt{par}) \quad \frac{P \;\rightarrow\; Q}{P \circ R \;\rightarrow\; Q \circ R} \qquad (\texttt{brane}) \quad \frac{P \;\rightarrow\; Q}{\sigma(\!| P |\!) \;\rightarrow\; \sigma(\!| Q |\!)}$$

$$(\texttt{strucong}) \quad \frac{P' \equiv P \qquad P \;\rightarrow\; Q \qquad Q \equiv Q'}{P' \;\rightarrow\; Q'}$$

[2] With abuse of notation we use $\equiv$ to denote both structural congruence on systems and structural congruence on processes.

Rules (par) and (brane) are the contextual rules that permit to a system to execute also if it is in parallel with another process or if it is inside a membrane, respectively. Rule (strucong) ensures that two structurally congruent systems have the same reactions.

With $\rightarrow^*$ we denote the reflexive and transitive closure of a relation $\rightarrow$.

We say that a system P is *deterministic* iff for all P', P'': if $P \rightarrow P'$ and $P \rightarrow P''$ then $P' \equiv P''$. We say that P has a *halting computation* (or a deadlock) if there exists Q such that $P \rightarrow^* Q$ and $Q \not\rightarrow$.

The system P' is a *derivative* of the system P if $P \rightarrow^* P'$; the set of *derivatives* of a system P is denoted by $Deriv(P)$.

The Phago/Exo/Pino Calculus (PEP). The PEP calculus is inspired by endocytosis/exocytosis. Endocytosis is the process of incorporating external material into a cell by "engulfing" it with the cell membrane, while exocytosis is the reverse process. As endocytosis can engulf an arbitrary amount of material, giving rise to an uncontrollable process, in [4] two more basic operations are used: *phagocytosis*, engulfing just one external membrane, and *pinocytosis*, engulfing zero external membranes.

Definition 4. *Let Name be a denumerable set of ambient names, ranged over by $n, m, \ldots$. The set of actions of PEP is defined by the following grammar:*

$$a \quad ::= \quad \curlyvee_n \mid \curlyvee_n^{\perp}(\sigma) \mid \curlywedge_n \mid \curlywedge_n^{\perp} \mid \circledcirc(\sigma)$$

Action $\curlyvee_n$ denotes phagocytosis; the co-action $\curlywedge_n^{\perp}$ is meant to synchronize with $\curlywedge_n$; names n are used to pair-up related actions and co-actions. The co-phago action is equipped with a process σ, this process will be associated to the new membrane that engulfs the external membrane. Action $\curlywedge_n$ denotes exocytosis, and synchronizes with the co-action $\curlywedge_n^{\perp}$. Exocytosis causes an irreversible mixing of membranes. Action $\circledcirc$ denotes pinocytosis. The pino action is equipped with a process σ: this process will be associated to the new membrane, that is created inside the membrane performing the pino action.

Definition 5. *The reaction relation for PEP is the least relation containing the following axioms, and satisfying the rules in Definition 3:*

(phago) $\quad \curlyvee_n.\sigma|\sigma_0\langle\!\langle P \rangle\!\rangle \ \circ \ \curlywedge_n^{\perp}(\rho).\tau|\tau_0\langle\!\langle Q \rangle\!\rangle \rightarrow \tau|\tau_0\langle\!\langle \rho\langle\!\langle \sigma|\sigma_0\langle\!\langle P \rangle\!\rangle \rangle\!\rangle \circ Q \rangle\!\rangle$

(exo) $\quad \curlywedge_n^{\perp}.\tau|\tau_0\langle\!\langle \curlywedge_n.\sigma|\sigma_0\langle\!\langle P \rangle\!\rangle \circ Q \rangle\!\rangle \rightarrow P \ \circ \ \sigma|\sigma_0|\tau|\tau_0\langle\!\langle Q \rangle\!\rangle$

(pino) $\quad \circledcirc(\rho).\sigma|\sigma_0\langle\!\langle P \rangle\!\rangle \rightarrow \sigma|\sigma_0\langle\!\langle \rho\langle\!\langle\, \rangle\!\rangle \circ P \rangle\!\rangle$

The Mate/Bud/Drip Calculus (MBD). The MBD calculus is inspired by membrane fusion and splitting. To make membrane splitting more controllable, in [4] two more basic operations are used: *budding*, consisting in splitting off one internal membrane, and *dripping*, consisting in splitting off zero internal membranes. Membrane fusion, or merging, is called *mating*.

Definition 6. *The set of actions of MBD is defined by the following grammar:*

$$a \quad ::= \quad mate_n \mid mate_n^{\perp} \mid bud_n \mid bud_n^{\perp}(\sigma) \mid drip(\sigma)$$

Actions $mate_n$ and $mate_n^{\perp}$ will synchronize to obtain membrane fusion. Action bud_n permits to split one internal membrane, and synchronizes with the co-action $bud_n^{\perp}$. Action *drip* permits to split off zero internal membranes. Actions $bud^{\perp}$ and *drip* are equipped with a process σ, that will be associated to the new membrane created by the membrane performing the action.

Definition 7. *The reaction relation for MBD is the least relation containing the following axioms, and satisfying the rules in Definition 3:*

(mate) $\quad mate_n.\sigma|\sigma_0\langle\!|\, P \,|\!\rangle \;\circ\; mate_n^{\perp}.\tau|\tau_0\langle\!|\, Q \,|\!\rangle \rightarrow \sigma|\sigma_0|\tau|\tau_0\langle\!|\, P \circ Q \,|\!\rangle$

(bud) $\quad bud_n^{\perp}(\rho).\tau|\tau_0\langle\!|\, bud_n.\sigma|\sigma_0\langle\!|\, P \,|\!\rangle \circ Q \,|\!\rangle \rightarrow \rho\langle\!|\, \sigma|\sigma_0\langle\!|\, P \,|\!\rangle\,|\!\rangle \;\circ\; \tau|\tau_0\langle\!|\, Q \,|\!\rangle$

(drip) $\quad drip(\rho).\sigma|\sigma_0\langle\!|\, P \,|\!\rangle \rightarrow \rho\langle\!|\,|\!\rangle \;\circ\; \sigma|\sigma_0\langle\!|\, P \,|\!\rangle$

In [4] it is shown that the operations of mating, budding and dripping can be encoded in PEP.

For the sake of simplicity, in the present paper we consider a basic calculus containing the membrane interaction primitives of both the PEP and the MBD calculi. As the primitives of MBD can be encoded in PEP, we conjecture that the system described in the following part of the paper can be encoded in an equivalent system that makes use of the PEP primitives only.

4.2 Computing $\{n^2 \mid n \geq 1\}$ in Brane Calculi

Now we show how to model our case study in Brane Calculi. Our solution is inspired by the simplified solution in Subsection 3.2. When moving from P systems to Brane Calculi, two main problems arise.

The first problem is concerned with the fact that in Basic Brane Calculi there are no objects/proteins floating inside the membranes. Hence, we need an alternative representation of the output of our system. In the solution based on P systems presented in Subsection 3.2, the natural number n is represented as n occurrences of object c inside membrane r. Here the idea is to represent the output as a family of membranes with a particular process C on them, such that process C can be distinguished by other processes residing on other auxiliary membranes.

A second major problem is concerned with the interleaving semantics of Brane Calculi. We note that the maximal parallelism semantics of P systems is a very powerful synchronization mechanism. This ensures that – at each computational step – for each occurrence of object b a new object c is created and for each occurrence of object a a new object b is created. If we simply encode each object a (resp. b, c) with a membrane $A\langle\!|\,|\!\rangle$ (resp. $B\langle\!|\,|\!\rangle$, $C\langle\!|\,|\!\rangle$), thus obtaining a flat multiset of membranes, then for mimicking a computational step of the corresponding P

system we need to perform a synchronization among an unbounded number of membranes, and this seems to be a very difficult task in Brane Calculi. On the other hand, it is quite easy to synchronize an a priori fixed number of membranes. To solve this problem, we decided to move from the flat structure of membranes proposed above (and consisting in a multiset of membranes $A(\!|\ |\!)$, $B(\!|\ |\!)$, and $C(\!|\ |\!)$ contained in the same surrounding membrane) to a hierarchical structure.

We start presenting a simplified version of the solution, where the output of the system is represented by the number of occurrences of C appearing in the whole structure of the system, and not inside a specific membrane. Then, we present a more elegant solution where the output of the system is represented by the number of occurrences of C contained in a specific membrane.

Solution with output scattered in the whole system. The initial system consists of an external membrane, containing two instances of membranes representing an encoding of object a and one brane representing an encoding of object b, as depicted in Figure 1 (the need for the auxiliary membranes decorated with processes X, Ta and Tb will be clarified in the following).

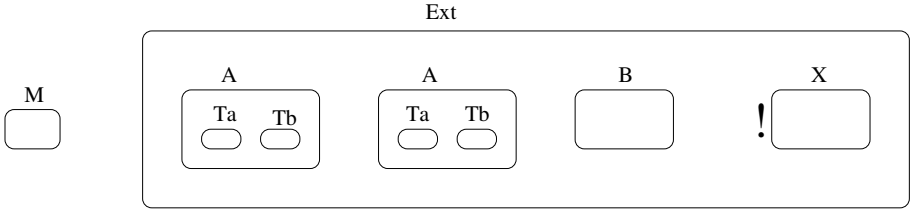

Fig. 1. The initial membrane system (with $M = mate_n^{\perp}$)

We mimic a single maximal parallelism computational step of the P system in Subsection 3.2 by the following sequence of steps: each membrane encoding object b creates – by dripping – a new membrane representing an encoding of c; each membrane encoding object a is surrounded by a newly created membrane representing a and containing a new instance of a membrane representing b.

An evolution of the representation of an object a as a nested family of membranes is reported in Figure 2.

The representations of objects a and b are arranged in a hierarchical structure: there exists a membrane with process A (and representing object a) surrounding both a membrane with process B (representing object b) and another membrane with process A' (surrounded by another membrane with process E – such a membrane is created during the phagocytosis to preserve bitonality and cannot be avoided). The membrane with object A' contains a membrane decorated with B and another membrane E containing a membrane A', and so on. The most internal instance of membrane decorated with A' contains the two terminal membranes Ta and Tb.

A maximal parallelism computational step of the P system in Subsection 3.2 is mimicked in the following way: the external membrane – with process Ext –

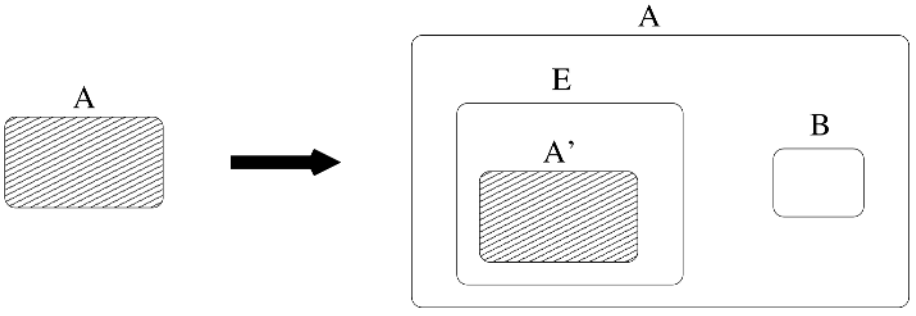

Fig. 2. The evolution of the system encoding object a

sends one (asynchronous) signal to each of its children. The child membrane with process B reacts to the signal by spawning a new child membrane with process C, and sends a signal to the external brane to communicate that it has finished its task. Each child membrane with process A reacts in the following way:

- first of all, the A membrane sends two signals to its children – decorated with B and E – that will be used to wake up the instances of membranes decorated with B inside the hierarchical structure (each of such B membranes will spawn a new C membrane);
- then it waits for two signals from its children, to acknowledge the end of the creation of new copies of C by the B membranes in the hierarchical structure;
- now, a new membrane is created, and the A membrane enters this new membrane by phagocytosis and spawns a new membrane with process B;
- finally, the A membrane sends a signal to the external membrane to acknowledge the end of its task, and evolves to a membrane with process A'.

Before presenting the definition of the system, we show how to obtain asynchronous communication between a father and a child membrane. If the father membrane wants to send a signal to one of its children, it produces by pinocytosis a bubble with process $mate_x^{\perp}$; the child accepts this signal by performing an action $mate_x$. On the other hand, if a child wants to send a signal to its father, it produces by dripping a bubble with process $\circlearrowleft_x$; the father receives this signal by performing an action $\circlearrowleft_x^{\perp}$.

Formally, the system is defined as follows:

$$\begin{aligned} mate_n^{\perp}(\!|\ \ |\!) \circ Ext(\!| & A(\!| Ta(\!|\ \ |\!) \circ Tb(\!|\ \ |\!) |\!) \circ \\ & A(\!| Ta(\!|\ \ |\!) \circ Tb(\!|\ \ |\!) |\!) \circ \\ & B(\!|\ |\!) \circ \\ & !(X(\!|\ |\!)) |\!) \end{aligned}$$

So, we have a big membrane containing two copies of A and one copy of B, plus the membrane $mate_n^{\perp}(\!|\ \ |\!)$. The membrane $mate_n^{\perp}(\!|\ \ |\!)$ is a trigger that fuses with the big membrane: if the fusion is performed by the first $mate_n$ action of

Ext, then some new copies of C are produced; otherwise, the system ends. As we already said before, the output of the system is represented by the number of occurrences of C appearing in the whole structure of the system, and not inside a specific membrane.

The process Ext is the following:

$$\begin{aligned} Ext = {} & !mate_n.\, \circledcirc(mate^{\perp}_{a_s}).\, \circledcirc(mate^{\perp}_{a_s}).\, \circledcirc(mate^{\perp}_{b_s}).\circlearrowleft^{\perp}_{a_f}.\circlearrowleft^{\perp}_{a_f}. \\ & \circlearrowleft^{\perp}_{b_f}.drip(mate^{\perp}_{n}) \mid \\ & mate_n.0 \end{aligned}$$

The program Ext triggers the two copies of A and B by producing three bubbles by pinocytosis that can fuse with the two instances of A and with B. The membrane B simply produces a child bubble labeled with C then signals the termination of this task to the external membrane. In this simplified version of the solution, C may be any process that can be distinguished from the others.

The evolution of membrane A is depicted in Figure 2; here we give a more detailed description of the behavior of such a kind of membrane.

First of all, the membrane A sends a signal to its children: at the beginning, this membrane has two dummy children (represented by systems Ta and Tb) that simply send back the signal; however, during the computation the last created membrane A has to send a signal to its children to permit to its descendants of kind B to produce new copies of C. Thus, membrane A sends a signal with label a_s to its child with process E and a signal with label b_s to its child with process B to trigger the starting of the execution of a computational step by the two children. Then, the membrane A waits for two signals: a signal with label a_f from its child E (meaning that all the B descendants have spawn a new copy of C) and a signal with label b_f from its child B (meaning that B has spawn a new copy of C). After the membrane A has received these two signals from its children, membrane A creates a new sibling bubble decorated with D, then A enters the D bubble (note that phagocytosis creates a new membrane surrounding A inside D; this causes the necessity to propagate signals across this membrane, that has process E). After A enters D, D creates a child with process B by pinocytosis, and then signals that it has finished its task to its father, and then, by fusing with a copy of an X membrane, it becomes a membrane with program A.

The definitions of the remaining systems and processes are as follows:

$$\begin{aligned} A &= mate_{a_s}.\, \circledcirc(mate^{\perp}_{a_s}).\, \circledcirc(mate^{\perp}_{b_s}).\circlearrowleft^{\perp}_{a_f}.\circlearrowleft^{\perp}_{b_f}.drip(D).\circlearrowleft_{d}.A' \\ A' &= !mate_{a_s}.\, \circledcirc(mate^{\perp}_{a_s}).\, \circledcirc(mate^{\perp}_{b_s}).\circlearrowleft^{\perp}_{a_f}.\circlearrowleft^{\perp}_{b_f}.drip(\circlearrowleft_{a_f}) \\ D &= \circlearrowleft^{\perp}_{d}(E).\, \circledcirc(B).drip(\circlearrowleft_{a_f}).mate^{\perp}_{x} \\ X &= mate_x.A \\ E &= !mate_{a_s}.\, \circledcirc(mate^{\perp}_{a_s}).\circlearrowleft^{\perp}_{a_f}.drip(\circlearrowleft_{a_f}) \\ B &= !mate_{b_s}.\, \circledcirc(C).drip(\circlearrowleft_{b_f}) \\ Ta &= (!mate_{a_s}.drip(\circlearrowleft_{a_f})) \\ Tb &= (!mate_{b_s}.drip(\circlearrowleft_{b_f})) \end{aligned}$$

Solution with output contained in a specific membrane. Now we show how to put the encoding of the output of the system inside a single membrane, with process Res. First of all, we surround the system by two membranes: the external membrane is decorated with process $Ext1$ and the internal membrane is decorated with process $Ext2$. The initial state of the system is reported in Figure 3.

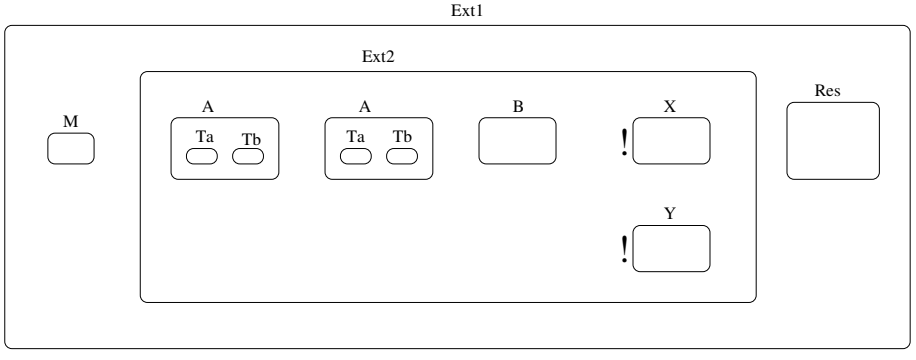

Fig. 3. The initial configuration of the system with output in the Res membrane (with $M = mate_n^{\perp}$)

The system behaves as the system presented in the previous subsection as far as the generation of new copies of C is concerned. On the other hand, when we decide to terminate (by choosing the second $mate_n$ action) then, instead of blocking the system, the continuation of process $Ext2$ (together with system $!Y(\!|\ |\!)$) permits to the nested membranes A,A' and B to perform an exocytosis. In this way, all the C membranes (as well as the terminating Ta and Tb membranes) are put in the region of the external membrane. The $Ext2$ membrane, as well as the E membranes, disappear by performing an exocytosis with the external membrane, whereas each C membrane produces a child decorated with C' by pinocytosis, and then fuses with the Res membrane.

When the computation stops, the result is represented by the number of C' membranes contained inside the Res membrane, and the structure of the system is depicted in Figure 4.

Formally, the system is defined as follows:

$$
\begin{aligned}
Ext1(\!|mate_n^{\perp}(\!|\ |\!) \circ\ Ext2(\!|\ & A(\!|\ Ta(\!|\ |\!) \circ Tb(\!|\ |\!)\ |\!) \circ \\
& A(\!|\ Ta(\!|\ |\!) \circ Tb(\!|\ |\!)\ |\!) \circ \\
& B(\!|\ |\!) \circ \\
& !(X(\!|\ |\!)) \circ \\
& !(Y(\!|\ |\!))\ |\!) \circ \\
Res(\!|\ & |\!)\ |\!)
\end{aligned}
$$

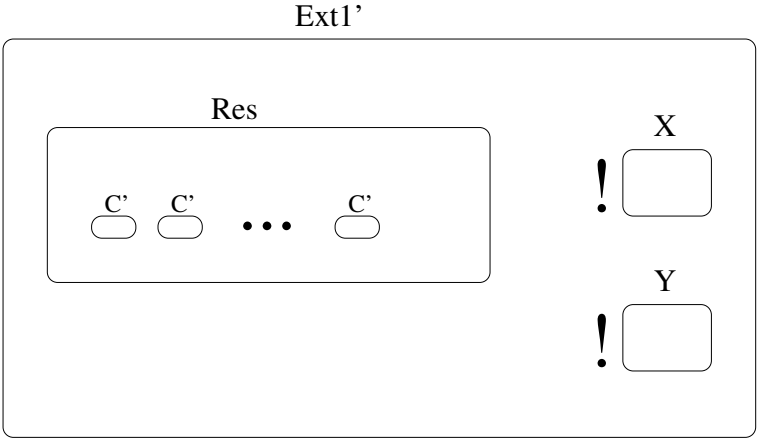

Fig. 4. The final configuration of the system with output in the *Res* membrane

The processes $Ext1$ and $Ext2$ are defined as follows:

$$Ext1 = !\circlearrowleft^{\perp}_{out}$$

$$\begin{aligned} Ext2 = {} & !mate_n.\ \circledcirc(mate^{\perp}_{a_s}).\ \circledcirc(mate^{\perp}_{a_s}).pino(mate^{\perp}_{b_s}).\circlearrowleft^{\perp}_{a_f}.\circlearrowleft^{\perp}_{a_f}. \\ & \circlearrowleft^{\perp}_{b_f}.drip(mate^{\perp}_{n}) \mid \\ & mate_n.\circlearrowleft^{\perp}_{a_e}.\circlearrowleft^{\perp}_{a_e}.\circlearrowleft^{\perp}_{b_e}.\circlearrowleft_{out} \end{aligned}$$

The definitions of the remaining systems and processes are as follows:

$$\begin{aligned} A &= mate_{a_s}.\ \circledcirc(mate^{\perp}_{a_s}).\ \circledcirc(mate^{\perp}_{b_s}).\circlearrowleft^{\perp}_{a_f}.\circlearrowleft^{\perp}_{b_f}.drip(D).\circlearrowleft_{d}.A' \mid \circlearrowleft_{a_e} \\ A' &= !mate_{a_s}.\ \circledcirc(mate^{\perp}_{a_s}).\ \circledcirc(mate^{\perp}_{b_s}).\circlearrowleft^{\perp}_{a_f}.\circlearrowleft^{\perp}_{b_f}.drip(\circlearrowleft_{a_f}) \mid \circlearrowleft_{a_e} \\ D &= \circlearrowleft^{\perp}_{d}(E).\ \circledcirc(B).drip(\circlearrowleft_{a_f}).mate^{\perp}_{x} \\ X &= mate_x.A \\ Y &= mate_y.\circlearrowleft^{\perp}_{a_e}.\circlearrowleft^{\perp}_{b_e}.\circlearrowleft^{\perp}_{out} \\ E &= !mate_{a_s}.\ \circledcirc(mate^{\perp}_{a_s}).\circlearrowleft^{\perp}_{a_f}.drip(\circlearrowleft_{a_f}) \mid mate^{\perp}_{y} \\ B &= !mate_{b_s}.\ \circledcirc(C).drip(\circlearrowleft_{b_f}) \mid \circlearrowleft_{b_e} \\ Ta &= (!mate_{a_s}.drip(\circlearrowleft_{a_f})) \mid \circlearrowleft_{out} \\ Tb &= (!mate_{b_s}.drip(\circlearrowleft_{b_f})) \mid \circlearrowleft_{out} \\ Res &= !mate^{\perp}_{res} \\ C &= \circledcirc(C').mate_{res} \end{aligned}$$

5 Final Remarks

In the last years, two branches of Natural Computing, *Membrane Computing* and *Brane Calculi* have been developed at the crossroads of Cell Biology and Computation. Both branches start from the idea of cells are capable to process and to generate information. Nonetheless, they have followed different paths.

Membrane Computing are more interested in the study of computational devices, by taking the cell as inspiration whereas Brane Calculi try to stay as close to the Biology as possible.

In a certain sense, Brane Calculi are dual to Membrane Computing, since they work with object placed *on* membranes, not with object placed in the regions surrounded by membranes. This is a key difference. In Membrane Computing, the objects represent chemicals swimming in an aqueous solution inside the membranes and membranes separate the compartments where local rules are applied. In Brane Calculi, objects are placed on membranes and they correspond to proteins embedded in the real membranes. The computation is made by membrane operations controlled by these objects.

Another notable difference between Brane Calculi and P systems is concerned with the semantics of the two formalism: whereas Brane Calculi are usually equipped with an interleaving, sequential semantics (each computational step consists of the execution of a single instruction), the usual semantics in membrane computing is based on maximal parallelism (a computational step is composed of a maximal set of independent interactions).

In this paper we started a joint investigation of both formalisms inspired by the behavior of biological membranes. In particular, we investigate their computational power w.r.t. their ability to generate sets of numbers, and we take as a case study the set $\mathcal{L} = \{n^2 \mid n \geq 1\}$.

First we recalled the P systems presented in [11] which generates $\mathcal{L}$, then we provided a new, simplified solution. Then we move to Brane Calculi, and we tackle the problem of presenting a solution to the case study based on the simplified solution we propose for P systems. After discussing the problems which arise when moving from P systems to Brane Calculi, we present two solutions of the problem in Brane Calculi. The most relevant problem is due to the shift from the maximal parallelism semantics of P systems to the interleaving semantics of Brane Calculi: while maximal parallelism turns out to be a very powerful synchronization tool, permitting to synchronize an unbounded number of components, it seems that this form of synchronization turns out to be problematic in Brane Calculi. We solve this problem by moving from a "flat" representation of the system to a hierarchical representation, that can be easily obtained by making use of an unbounded number of membranes.

We think that the present paper could represent a first step in the comparison of the two aforementioned formalisms. As future work, we plan to investigate the possibility to compute **NP**-complete problems in polynomial time with Brane Calculi, by taking as a starting point the encouraging results on this topic obtained for P systems (see, for example, [13] and references therein).

Acknowledgement

The second author acknowledges the support by Project TIN2005-09345-C03-01 of the Ministry of Education and Science of Spain, cofinanced by FEDER funds, and by the Project of Excellence TIC-581 of the Junta de Andalucía.

References

1. L.M. Adleman. Molecular computations of solutions to combinatorial problems. *Science*, 226 (1994), 1021–1024.
2. D. Besozzi, N. Busi, G. Franco, R. Freund, Gh. Păun. Two universality results for (mem)brane systems. In *Proceedings of the Fourth Brainstorming Week on Membrane Computing, Vol. I* (M.A. Gutiérrez Naranjo, Gh. Păun, A. Riscos-Núñez, F.J. Romero-Campero, eds.), Fénix Editora, 2006, 49–62.
3. N. Busi, R. Gorrieri. On the computation power of brane calculi. Third Workshop on Computational Methods in Systems Biology, Edinburgh, 2005.
4. L. Cardelli. Brane calculi. In *Computational Methods in Systems Biology 2004* (V. Danos, V. Schachter, eds.), LNBI **3082**, Springer-Verlag, Berlin, 2005, 257–278.
5. L. Cardelli and A.D. Gordon. Mobile ambients. *Theoretical Computer Science*, 240, 1 (2000), 177–213.
6. L. Cardelli, Gh.Păun. An universality result for a (mem)brane calculus based on mate/drip operations. *Intern. J. Found. Computer Sci.*, 17, 1 (2006), 49–68.
7. J.H. Holland. *Adaptation in Natural and Artificial Systems*. Ann Arbor, MI: University of Michigan Press, 1975.
8. W.S. McCulloch, W. Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5 (1943) 115–133.
9. Gh. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143.
10. Gh. Păun, M.J. Pérez-Jiménez. Recent computing models inspired from biology: DNA and membrane computing. *Theoria*, 18 (2003), 72–84.
11. Gh. Păun. *Membrane Computing – An Introduction* Springer-Verlag, Berlin, 2002.
12. A. Regev, E. M. Panina, W. Silverman, L. Cardelli, E. Shapiro. BioAmbients: An abstraction for biological compartments. *Theoretical Computer Science*, 325, 1 (2004), 141–167.
13. A. Riscos-Núñez. *Cellular Programming: Efficient Resolution of Numerical NP-Complete Problems*. Ph.D. Thesis. University of Seville, 2004.
14. P systems web page `http://psystems.disco.unimib.it/`