On the computational power of Brane Calculi *

Nadia Busi and Roberto Gorrieri

Dipartimento di Scienze dell'Informazione, Università di Bologna,
Mura A. Zamboni 7, I-40127 Bologna, Italy.
E-mail: {busi,gorrieri}@cs.unibo.it

Abstract. Brane calculi are a family of biologically inspired process calculi proposed in [3] for modeling the interactions of dynamically nested membranes.
In [3] a basic calculus for membranes interactions – called Phago/Exo/Pino – is proposed, whose primitives are inspired by endocytosis and exocytosis. An alternative basic calculus – called Mate/Bud/Drip and inspired by membrane fusion and fission – is also outlined and shown to be encodable in Phago/Exo/Pino in [3].
In this paper we investigate and compare the expressiveness of such two calculi w.r.t. their ability to act as computational devices.
We show that (a fragment of) the Phago/Exo/Pino calculus is Turing powerful, by providing an encoding of Random Access Machines.
On the other hand, we show the impossibility to define a "faithful" encoding of Random Access Machines in the Mate/Bud/Drip calculus, by providing a proof of the decidability of the existence of a divergent computation in Mate/Bud/Drip.

1 Introduction

Brane calculi [3] are a family of process calculi proposed for modeling the behavior of biological membranes. The formal investigation of biological membranes has been initiated by G. Păun [13], in the field of automata and formal language theory, with the definition of P systems. In a process algebraic setting, the notions of membranes and compartments are explicitly represented in BioAmbients [16], a variant of Mobile Ambients [5] based on a set of biologically inspired primitives of interaction. Brane calculi represent an evolution of BioAmbients: the main difference w.r.t. previous approaches consists in the fact that the active entities reside on membranes, and not inside membranes. In this paper we are interested in two basic instances of brane calculi proposed in [3]: 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

* Revised and full version of the extended abstract in Proc. Workshop on Computational Methods in Systems Biology, Edinburgh, April 2005.

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 [3]. Cardelli also observed that the reverse encoding does not exist, if the encoding must preserve the nesting structure of membranes. The reason is that in MBD the maximum nesting level of membranes cannot grow during the computation, while this property does not hold for PEP.

The aim of this work is to investigate the expressiveness of PEP and MBD as computational devices. We show that a fragment of PEP, namely, the calculus comprising only the phago and exo primitives, is Turing powerful. The proof is carried out by showing how to encode a Random Access Machine [17], a well known Turing powerful formalism. As a consequence, universal termination turns out to be undecidable on PEP. As far as MBD is concerned, we show that universal termination is a decidable property. The proof of the decidability of universal termination is based on the theory of well-structured transition systems [8]. The decidability of universal termination for MBD provides an expressiveness gap between MBD and PEP, as Random Access Machines can be encoded in the second calculus, but not in the first calculus. As a corollary, we get the impossibility to provide an encoding of PEP in MBD that preserves universal termination of systems.

The paper is organized as follows: in Section 2 we present the syntax and the semantics of the two calculi; Section 3 contains the encoding of Random Access Machines in PEP, while the decidability of universal termination for MBD is presented in Section 4. Section 5 reports some conclusive remarks.

2 Brane Calculi: Syntax and Semantics

In this section we recall the syntax and the semantics of Brane Calculi [3]. 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 brane 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 brane that performs process σ and contains system P.

The term 0 denotes the empty process, whereas $\mid$ 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:[1]

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:*

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

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

Rules `(par)` and `(brane)` are the contextual rules that respectively 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 $\rightarrow$. Given a reduction relation $\rightarrow$, we say that a system P has a *divergent computation* (or infinite computation) if there exists an infinite sequence of systems $P_0, P_1, \ldots, P_i, \ldots$ such that $P = P_0$ and $\forall i \geq 0 : P_i \rightarrow P_{i+1}$. We say that a system P *universally terminates* if it has no divergent computations. We say that

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

P is *deterministic* iff for all P', P'': if $P \to P'$ and $P \to P''$ then $P' \equiv P''$. We say that P has a *terminating computation* (or a deadlock) if there exists Q such that $P \to^* Q$ and $Q \not\to$. A system P satisfies the universal termination property if P has no divergent computations. A system P satisfies the existential termination property if P has a deadlock. Note that the existential termination and the universal termination properties are equivalent on deterministic systems.

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

We use $\prod$ (resp. $\bigcirc$) to denote the parallel composition of a set of processes (resp. systems), i.e., $\prod_{i\in\{1,\ldots,n\}} \sigma_i = \sigma_1 \mid \ldots \mid \sigma_n$ and $\bigcirc_{i\in\{1,\ldots,n\}} P_i = P_1 \circ \ldots \circ P_n$. Moreover, $\prod_{i\in\emptyset} \sigma_i = 0$ and $\bigcirc_{i\in\emptyset} P_i = \diamond$.

2.1 The Phago/Exo/Pino Calculus (PEP)

The first calculus we investigate is proposed in [3], and it 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 [3] 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:*

$$a \quad ::= \quad \lhd_n \mid \lhd_n^\perp(\sigma) \mid \rhd_n \mid \rhd_n^\perp \mid \circledcirc(\sigma)$$

Action $\lhd_n$ denotes phagocytosis; the co-action $\lhd_n^\perp$ is meant to synchronize with $\lhd_n$; 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 $\rhd_n$ denotes exocytosis, and synchronizes with the co-action $\rhd_n^\perp$. 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 brane 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) $\quad \lhd_n.\sigma|\sigma_0\langle\!|\, P \,|\!\rangle \circ \lhd_n^\perp(\rho).\tau|\tau_0\langle\!|\, Q \,|\!\rangle \to \tau|\tau_0\langle\!|\, \rho\langle\!|\, \sigma|\sigma_0\langle\!|\, P \,|\!\rangle \,|\!\rangle \circ Q \,|\!\rangle$

(exo) $\quad \rhd_n^\perp.\tau|\tau_0\langle\!|\, \rhd_n.\sigma|\sigma_0\langle\!|\, P \,|\!\rangle \circ Q \,|\!\rangle \to P \circ \sigma|\sigma_0|\tau|\tau_0\langle\!|\, Q \,|\!\rangle$

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

2.2 The Mate/Bud/Drip Calculus (MBD)

The second calculus, also proposed in [3], is inspired by membrane fusion and splitting. To make membrane splitting more controllable, in [3] 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 brane 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 [3] it is shown that the operations of mating, budding and dripping can be encoded in PEP.

3 PEP is Turing powerful

In this section we show that a fragment of PEP, namely, the calculus without the pino action, is Turing powerful. The result is proved by showing how to model Random Access Machines (RAMs) [17], a well known Turing powerful formalism. A direct consequence of this result is the undecidability of universal termination for PEP. We start by recalling what RAMs are.

3.1 Random Access Machines

RAMs are a computational model based on finite programs acting on a finite set of registers. More precisely, a RAM R is composed of the registers $r_1, \ldots, r_n$, that can hold arbitrary large natural numbers, and by a sequence of indexed instructions $(1 : I_1), \ldots, (m : I_m)$. In [12] it is shown that the following two instructions are sufficient to model every recursive function:

- $(i : Succ(r_j))$: adds 1 to the contents of register r_j and goes to the next instruction;

– $(i : DecJump(r_j, s))$: if the contents of the register r_j is not zero, then decreases it by 1 and goes to the next instruction, otherwise jumps to the instruction s.

The computation starts from the first instruction and it continues by executing the other instructions in sequence, unless a jump instruction is encountered. The execution stops when an instruction number higher than the length of the program is reached.

A state of a RAM is modelled by $(i, c_1, \ldots, c_n)$, where i is the program counter indicating the next instruction to be executed, and $c_1, \ldots, c_n$ are the current contents of the registers $r_1, \ldots, r_n$, respectively. We use the notation $(i, c_1, \ldots, c_n) \rightarrow_R (i', c'_1, \ldots, c'_n)$ to denote that the state of the RAM R changes from $(i, c_1, \ldots, c_n)$ to $(i', c'_1, \ldots, c'_n)$, as a consequence of the execution of the i-th instruction.

A state $(i, c_1, \ldots, c_n)$ is *terminated* if the program counter i is strictly greater than the number of instructions m. We say that a RAM R *terminates* if its computation reaches a terminated state.

3.2 Modelling RAMs in PEP

In this section we show how to model RAMs in PEP. The modelling of RAMs is based on an encoding function, which transforms instructions and registers independently.

The basic idea for modelling the natural numbers contained in the registers is the following: the natural number n is represented by the nesting of $2n + 1$ branes. The increment is performed by producing a new membrane that performs a phago on the representation of n, while a decrement is performed by executing an exo of the membrane representing $n - 1$, that lies inside the membrane representing n.

Consider a RAM R with instructions $(1 : I_1)$, ..., $(m : I_m)$ and registers r_1, ..., r_n; the encoding of an initial state $(1, c_1, \ldots, c_n)$ is defined as follows:

$$[\![(1, c_1, \ldots, c_n)]\!] = \mho_{p_1}(\!|\,|\!)\circ !\,[\![(1 : I_1)]\!] \circ \ldots \circ !\,[\![(m : I_m)]\!] \circ [\![r_1 = c_1]\!] \circ \ldots \circ [\![r_n = c_n]\!] \circ GARB(\!|\,|\!)$$

where

$$GARB = !\,\mho^{\perp}_{exitpc} \mid !\,\mho^{\perp}_{garbage}(\mho^{\perp}_{exitpc})$$

is the process on the garbage collector membrane.

The encoding of an initial state of the RAM is composed by the following parts: $\mho_{p_1}(\!|\,|\!)$, representing the program counter, (an unbounded number of occurrences of) the encodings of each instruction, the encodings of the initial contents of registers, and a garbage membrane used to collect no longer used membranes and to inhibit their actions.

The presence of a "program counter" system $\mho_{p_i}(\!|\,|\!)$ denotes the fact that the next instruction to be executed is I_i.

The encoding of register r_j with content zero is

$$[\![r_j = 0]\!] = Z_j(\!|\,|\!)$$

where

$$Z_j = !\circlearrowleft_{inc_j} \mid !\circlearrowleft^{\perp}_{decreq_j}(\circlearrowleft^{\perp}_{z}).\circlearrowleft^{\perp}_{exitpc}$$

If an increment operation on r_j is executed, then – by action $\circlearrowleft_{inc_j}$ – the system $[\![r_j = 0]\!]$ is engulfed in a new membrane, thus obtaining a representation of $[\![r_j = 1]\!]$.

If the membrane of system $[\![r_j = 0]\!]$ is entered by a request for decrement or jump – action $\circlearrowleft^{\perp}_{decreq_j}$ – the choice corresponding to the zero case is selected, and the program counter corresponding to the jump is expelled by $\circlearrowleft^{\perp}_{exitpc}$.

The encoding of register r_j with content $n + 1$ is

$$[\![r_j = n + 1]\!] = S_j(\!|\circlearrowleft_{exitreg}(\!|\,[\![r_j = n]\!]\,|\!)|\!)$$

where

$$S_j = !\circlearrowleft_{inc_j} \mid \circlearrowleft^{\perp}_{decreq_j}(\circlearrowleft^{\perp}_{nz}).\circlearrowleft^{\perp}_{exitreg}.\circlearrowleft_{garbage}.\circlearrowleft^{\perp}_{exitpc}$$

The case of an increment operation is treated in the same way as in the encoding of $r_j = 0$.

If the membrane of system $r_j = n + 1$ is entered by a request for decrement or jump, then the choice corresponding to the non-zero case is selected by $\circlearrowleft^{\perp}_{nz}$, and the membrane representing $r_j = n$ is expelled by $\circlearrowleft^{\perp}_{exitreg}$. At this point, the no longer used external membrane of the system $[\![r_j = n + 1]\!]$ is engulfed by the garbage collector membrane, by $\circlearrowleft_{garbage}$. Finally, the program counter corresponding to the next instruction is expelled by $\circlearrowleft^{\perp}_{exitpc}$.

Note that it is necessary to engulf the external membrane of the system $[\![r_j = n + 1]\!]$ in the garbage collector, to inhibit the possibility for the process $!\circlearrowleft_{inc_j}$ to capture a subsequent request for increment of register r_j.

The encoding for the instruction $(i : I_i)$ is as follows:

$$\begin{aligned}
[\![(i : Succ(r_j))]\!] &= \circlearrowleft^{\perp}_{p_i}(0).\circlearrowleft^{\perp}_{inc_j}(\circlearrowleft_{exitreg}).\circlearrowleft^{\perp}_{exitpc}.S_j(\!|\circlearrowleft_{exitpc}(\!|\circlearrowleft_{p_{i+1}}(\!|\;\;|\!)|\!)|\!)\\
[\![(i : DecJump(r_j, s))]\!] &= \circlearrowleft^{\perp}_{p_i}(0).\circlearrowleft_{decreq_j}.J_{i+1,s}
\end{aligned}$$

where

$$J_{h,k} = \circlearrowleft^{\perp}_{znz}(\!|\,\circlearrowleft_{znz}(\!|\,\circlearrowleft_{nz}\;(\!|\circlearrowleft_{exitpc}(\!|\circlearrowleft_{exitpc}(\!|\circlearrowleft_{exitpc}(\!|\circlearrowleft_{p_h}(\!|\;|\!)|\!)|\!)|\!)|\!)|\!)\;\circ \\ \circlearrowleft_{z}\;\;(\!|\circlearrowleft_{exitpc}(\!|\circlearrowleft_{p_k}(\!|\,|\!)|\!)|\!)\;\;|\!)\,|\!)$$

The encoding of each instruction consists in a membrane, and the encoding of a RAM contains an unbounded number of copies of the encoding of each instruction.

When a program counter system $\circlearrowleft_{p_i}(\!|\,|\!)$ appears at top-level, an (occurrence of) instruction $(i : I_i)$ is activated by engulfing the program counter.

If the i-th instruction is an increment of register r_j, then the external membrane of the encoding of such an instruction will become the new external membrane of the encoding of the contents of register r_j. To this aim, the external

membrane of $[\![(i : Succ(r_j))]\!]$ engulfs the system $[\![r_j = k]\!]$ – where k is the current contents of r_j – by $\mho^{\perp}_{inc_j}$; $[\![r_j = k]\!]$ is wrapped with a membrane decorated with process $\mho_{exitreg}$, that will permit to $[\![r_j = k]\!]$ to be expelled when a decrement operation will be performed on system $[\![r_j = k + 1]\!]$. At this point, the internal program counter $\mho_{p_{i+1}}(\!|\,|\!)$ is expelled to the top-level, by action $\mho^{\perp}_{exitpc}$, and the process associated to the external brane will behave as S_j, i.e., the process on the external brane of $[\![r_j = k + 1]\!]$.

Suppose that the i-th instruction is a decrement of register r_j, or jump to instruction s if the contents of r_j is zero.

If the contents of r_j is not zero, e.g., $r_j = k + 1$, then it is necessary to remove the two outer membranes of $[\![r_j = k + 1]\!]$, or, in other words, to permit to $[\![r_j = k]\!]$ to be expelled from the system $[\![r_j = k + 1]\!]$. Moreover, the program counter $\mho_{p_{i+1}}(\!|\,|\!)$ should be produced at top-level.

On the other hand, if the contents of r_j is zero, the program counter $\mho_{p_s}(\!|\,|\!)$ should be produced.

In both cases, the membrane representing the i-th instruction is engulfed by the encoding of register r_j, by performing action $\mho_{decreq_j}$. The system engulfed by the encoding of r_j is $J_{i+1,s}$, and essentially permits to perform a choice between the two program counters $\mho_{p_{i+1}}(\!|\,|\!)$ and $\mho_{p_s}(\!|\,|\!)$. After entering the encoding of r_j, $J_{i+1,s}$ evolves to $\mho_{nz}(\!|\mho_{exitpc}(\!|\mho_{exitpc}(\!|\mho_{exitpc}(\!|\mho_{p_{i+1}}(\!|\,|\!)|\!)|\!)|\!)|\!) \circ \mho_z(\!|\mho_{exitpc}(\!|\mho_{p_s}(\!|\,|\!)|\!)|\!)$. If $r_j = 0$, and hence the encoding of the i-th instruction has been engulfed by $[\![r_j = 0]\!]$, then $J_{i+1,s}$ has been engulfed by an internal membrane with associated process $\mho^{\perp}_z$, and only the program counter $\mho_{p_s}(\!|\,|\!)$ will be expelled. (The other program counter will remain forever inside the system $[\![r_j = 0]\!]$, surrounded by a membrane with an associated empty process.)

On the other hand, if $r_j > 0$, then $J_{i+1,s}$ has been engulfed by an internal membrane with associated process $\mho^{\perp}_{nz}$, and only the the program counter $\mho_{p_{i+1}}(\!|\,|\!)$ will be expelled.

3.3 Correctness of the encoding

In this section we show that our encoding of RAMs preserves universal termination. In the previous section, we noted that during the computation some innocuous garbage is created inside the garbage collector membrane and inside the membrane of the system $[\![r_j = 0]\!]$.

To show the correctness of the encoding we need to keep this additional garbage systems into account; hence, we define the encoding $[\![\,]\!]$, mapping a state of the RAM on a set of processes.

The set $[\![\![r_j = n]\!]\!]$ contains all the processes that are equal to the encoding of registers, but for the presence of some innocuous garbage inside the inner membrane.

Definition 8. *Let R be a system.*

We say that $R \in [\![\![r_j = 0]\!]\!]$ iff there exists I such that $R \equiv Z_j(\!|GZ_I|\!)$ where $GZ_I = \bigcirc_{i \in I} 0(\!|\mho_{nz}(\!|\mho_{exitpc}(\!|\mho_{exitpc}(\!|\mho_{exitpc}(\!|\mho_{p_{h_i}}(\!|\,|\!)|\!)|\!)|\!)|\!)|\!)$ and $Z_j = !\mho_{inc_j} \mid !\mho^{\perp}_{decreq_j}(\mho^{\perp}_z).\mho^{\perp}_{exitpc}$.

We say that $R \in [\![r_j = n+1]\!]$ *iff there exists* $R' \in [\![r_j = n]\!]$ *such that*
$R \equiv S_j(\!|\, \Im_{exitreg}(\!|\, R' \,|\!)\,|\!)$
and $S_j = !\Im_{inc_j} \mid \Im^{\perp}_{decreq_j}(\Im^{\perp}_{nz}).\Im^{\perp}_{exitreg}.\Im_{garbage}.\Im^{\perp}_{exitpc}.$

Given a state $(i, c_1, \ldots, c_n)$, with $[\![(i, c_1, \ldots, c_n)]\!]$ we denote the set of processes that are equal to process $[\![(i, c_1, \ldots, c_n)]\!]$ but for the presence of some innocuous garbage.

Definition 9. *Let* P *be a system.*
We say that $P \in [\![(i, c_1, \ldots, c_n)]\!]$ *iff there exist* $J, R_1, \ldots, R_n$ *such that*
$R_i \in [\![r_i = c_i]\!]$ *for* $i = 1, \ldots, n$ *and*
$P \equiv \Im_{p_i}(\!|\,|\!) \circ ! [\![(1 : I_1)]\!] \circ \ldots \circ ! [\![(m : I_m)]\!] \circ$
$\quad R_1 \circ \ldots \circ R_n \circ GARB(\!|\, GG_J \,|\!)$
where $GG_J = \bigcirc_{j \in J} 0(\!|!\Im_{inc_j}(\!|\Im_z(\!|\Im_{exitpc}(\!|\Im_{p_{k_j}}(\!|\,|\!)|\!)|\!)|\!)|\!)$.

As the encoding of the initial state of a RAM defined in the previous section does not contain any garbage, it is easy to see that it belongs to the set $[\![\,]\!]$ of encodings corresponding to the initial state:

Proposition 1. *Let* R *be a RAM with program* $(1 : I_1), \ldots, (m : I_m)$ *and initial state* $(1, c_1, \ldots, c_n)$. *Then* $[\![(1, c_1, \ldots, c_n)]\!] \in [\![(1, c_1, \ldots, c_n)]\!]$.

We show that a system belonging to the set of encodings of a RAM state is able to mimic a computational step of the RAM by a (nonempty) sequence of steps.

Lemma 1. *Let* R *be a RAM with program* $(1 : I_1), \ldots, (m : I_m)$ *and initial state* $(1, c_1, \ldots, c_n)$. *Let* $(i, c_1, \ldots, c_n)$ *be a state of* R. *If* $(i, c_1, \ldots, c_n) \rightarrow_R (i', c'_1, \ldots, c'_n)$ *then for all systems* $P \in [\![(i, c_1, \ldots, c_n)]\!]$ *there exists* $Q \in [\![(i', c'_1, \ldots, c'_n)]\!]$ *such that* $P \rightarrow^+ Q$.

Proof. The proof is by case analysis. Four cases can happen:

1. the ith instruction is an increment on register r_j and $c_j = 0$;
2. the ith instruction is an increment on register r_j and $c_j > 0$;
3. the ith instruction is a decrement on register r_j and $c_j = 0$;
4. the ith instruction is a decrement on register r_j and $c_j > 0$.

We report only the first case.
Suppose that the ith instruction is an increment on r_j and $c_j = 0$.
Hence, $i' = i + 1$, $c'_j = 1$ and $c'_i = c_i$ for $i = 1, \ldots, n$ and $i \neq j$.
As $P \in [\![(i, c_1, \ldots, c_n)]\!]$, we have that exist $J, R_1, \ldots, R_n$ such that
$R_i \in [\![r_i = c_i]\!]$ for $i = 1, \ldots, n$ and
$P \equiv \Im_{p_i}(\!|\,|\!) \circ ! [\![(1 : I_1)]\!] \circ \ldots \circ ! [\![(m : I_m)]\!] \circ$
$\quad R_1 \circ \ldots \circ R_n \circ GARB(\!|\, GG_J \,|\!)$
with $GG_J = \bigcirc_{j \in J} 0(\!|!\Im_{inc_j}(\!|\Im_z(\!|\Im_{exitpc}(\!|\Im_{p_{k_j}}(\!|\,|\!)|\!)|\!)|\!)|\!)$.
As the ith instruction is an increment on register r_j, we have that
$[\![(i : I_i)]\!] = \Im^{\perp}_{p_i}(0).\Im^{\perp}_{inc_j}(\Im_{exitreg}).\Im^{\perp}_{exitpc}.S_j(\!|\Im_{exitpc}(\!|\Im_{p_{i+1}}|\!)|\!)$.

Hence the program counter system $\mathfrak{O}_{p_i}(\!|\,|\!)$ is engulfed by $[\![(i : I_i)]\!]$. Formally, $P \to P_1$ where

$P_1 = \mathfrak{O}^{\perp}_{inc_j}(\mathfrak{O}_{exitreg}).\mathfrak{O}^{\perp}_{exitpc}.S_j(\!|\mathfrak{O}_{exitpc}(\!|\mathfrak{O}_{p_{i+1}}(\!|\,|\!)|\!)|\!)\circ$
$\quad !\,[\![(1 : I_1)]\!] \circ \ldots \circ !\,[\![(m : I_m)]\!] \circ$
$\quad R_1 \circ \ldots \circ R_n \circ GARB(\!|\,GG_J\,|\!)$.

As $R_j \in [\![r_j = c_j]\!]$, we have that there exists I such that $R_j \equiv Z_j(\!|GZ_I|\!)$, with $GZ_I = \bigcirc_{i\in I} 0(\!|\mathfrak{O}_{nz}(\!|\mathfrak{O}_{exitpc}(\!|\mathfrak{O}_{exitpc}(\!|\mathfrak{O}_{exitpc}(\!|\mathfrak{O}_{p_{h_i}}(\!|\,|\!)|\!)|\!)|\!)|\!)|\!)$ and
$Z_j = !\,\mathfrak{O}_{inc_j} \mid !\,\mathfrak{O}^{\perp}_{decreq_j}(\mathfrak{O}^{\perp}_{z}).\mathfrak{O}^{\perp}_{exitpc}$.

By performing action $\mathfrak{O}^{\perp}_{inc_j}(\mathfrak{O}_{exitreg})$, the external membrane of the encoding of the ith instruction engulfs system R_j. Formally, $P_1 \to P_2$ with

$P_2 = \mathfrak{O}^{\perp}_{exitpc}.S_j(\!|\mathfrak{O}_{exitpc}(\!|\mathfrak{O}_{p_{i+1}}(\!|\,|\!)|\!) \circ \mathfrak{O}_{exitreg}(\!|R_j|\!)|\!)\circ$
$\quad !\,[\![(1 : I_1)]\!] \circ \ldots \circ !\,[\![(m : I_m)]\!] \circ$
$\quad R_1 \circ \ldots R_{j-1} \circ R_{j+1} \ldots \circ R_n \circ GARB(\!|\,GG_J\,|\!)$

By performing $\mathfrak{O}^{\perp}_{exitpc}$, the new program counter $\mathfrak{O}_{p_{i+1}}(\!|\,|\!)$ is expelled; hence $P_2 \to P_3$ with

$P_3 = \mathfrak{O}_{p_{i+1}}(\!|\,|\!) \circ S_j(\!|\mathfrak{O}_{exitreg}(\!|R_j|\!)\,|\!)\circ$
$\quad !\,[\![(1 : I_1)]\!] \circ \ldots \circ !\,[\![(m : I_m)]\!] \circ$
$\quad R_1 \circ \ldots R_{j-1} \circ R_{j+1} \ldots \circ R_n \circ GARB(\!|\,GG_J\,|\!)$

As $R_j \in [\![r_j = 0]\!]$, we have that $S_j(\!|\mathfrak{O}_{exitreg}(\!|R_j|\!)\,|\!) \in [\![r_j = 1]\!]$; hence, $P_3 \in [\![(i+1, c_1, \ldots, c_{j-1}, 1, c_{j+1}, \ldots, c_n)]\!]$.

Summing up, we have that
$(i, c_1, \ldots, c_{j-1}, 0, c_{j+1}, \ldots, c_n) \to_R (i+1, c_1, \ldots, c_{j-1}, 1, c_{j+1}, \ldots, c_n)$ and
$P \to^+ P_3$ with $P_3 \in [\![(i+1, c_1, \ldots, c_{j-1}, 1, c_{j+1}, \ldots, c_n)]\!]$.

We show that a sufficiently long sequence of computational steps of a weak encoding of a RAM state corresponds to one (or more) steps of the RAM.

Lemma 2. *Let R be a RAM with program $(1 : I_1), \ldots, (m : I_m)$ and initial state $(1, c_1, \ldots, c_n)$. Let $(i, c_1, \ldots, c_n)$ be a state of R and $P \in [\![(i, c_1, \ldots, c_n)]\!]$. If $P \to P_1 \to \ldots \to P_j \to \ldots$ then one of the following holds:*

- *the ith instruction is $(i : Succ(r_j))$ and $P_3 \in [\![(i+1, c_1, \ldots, c_j + 1, \ldots, c_n)]\!]$;*
- *the ith instruction is $(i : DecJump(r_j, s))$, $c_j = 0$ and $P_5 \in [\![(s, c_1, \ldots, c_n)]\!]$;*
- *the ith instruction is $(i : DecJump(r_j, s))$, $c_j > 0$ and $P_9 \in [\![(i+1, c_1, \ldots, c_j - 1, \ldots, c_n)]\!]$.*

Proof. (Sketch) The proof is by case analysis.

It is easy to see that the systems contained in the garbage collector membrane, as well as the systems in the membrane of the encoding of registers with content zero, are innocuous and can perform no move.

Concerning the first two cases, the first three (resp. five) steps of computation are univocally determined and lead to a system belonging to the set of encodings of the next state of the RAM.

In the last case, some nondeterminism is introduced by the interleaving of actions $\mathfrak{O}^{\perp}_{znz}$ and $\mathfrak{O}^{\perp}_{nz}$ – performed by the system encoding the DecJump instruction – with actions $\mathfrak{O}^{\perp}_{exitreg}$ and $\mathfrak{O}_{garbage}$ – performed by the encoding of register r_j. Whatever execution order is chosen, after the execution of the four actions mentioned above the reached system is the same.

We can now conclude with the Theorem which states that our modelling of RAMs preserves universal termination.

Theorem 1. *Let R be a RAM with program $(1 : I_1), \ldots, (m : I_m)$ and initial state $(1, c_1, \ldots, c_n)$. Then we have that the RAM R terminates if and only if all computations of the system $[\![(i, c_1, \ldots, c_n)]\!]$ terminate.*

An immediate consequence is the undecidability of universal termination:

Corollary 1. *The universal termination property is undecidable for PEP systems.*

The above theorem provides no information on the decidability of existential termination: if the RAM does not terminate, we only deduce that there exists at least one divergent computation starting from the encoding of the RAM.

A first possibility to prove the undecidability of existential termination consists in showing that the encoding presented in this section is *uniform w.r.t. termination*. A process P is uniform w.r.t. termination if the following property holds: if P has a terminating computation, then all computations starting from P terminate. An encoding satisfying this uniformity property provides a faithful modeling of the behaviour of the RAM: if the RAM terminates (resp. diverges) then all the computations of the encoding terminate (resp. diverge).

An alternative possibility consists in providing a deterministic encoding of RAMs; as for deterministic systems existential and universal termination are equivalent properties, the undecidability of existential termination directly follows from Corollary 1.

3.4 A deterministic encoding of RAMs

In this section we show how to obtain a deterministic encoding of RAMs by a slight modification of the encoding presented in section 3.2. In the previous encoding some nondeterminism is present in the execution of a decrement operation. After the membrane of the DecJump instruction entered the encoding of the register to decrement, the action $\heartsuit^{\perp}_{znz}$ (performed by the encoding of the instruction) can be executed in parallel with actions $\heartsuit^{\perp}_{exitreg}$ and $\heartsuit_{garb}$ (performed by the encoding of register). Here we introduce a synchronization membrane that forces a sequential execution of the two sequences of actions mentioned above.

Consider a RAM R with instructions $(1 : I_1)$, ..., $(m : I_m)$ and registers r_1, ..., r_n; the encoding of an initial state $(1, c_1, \ldots, c_n)$ is defined as follows:

$$\begin{aligned}\langle\!\langle (1, c_1, \ldots, c_n) \rangle\!\rangle = \heartsuit_{p_1}(\!|\,|\!)\circ ! \ \langle\!\langle (1 : I_1) \rangle\!\rangle \ \circ \ \ldots \ \circ \ ! \ \langle\!\langle (m : I_m) \rangle\!\rangle \circ \\ \langle\!\langle r_1 = c_1 \rangle\!\rangle \ \circ \ \ldots \ \circ \ \langle\!\langle r_n = c_n \rangle\!\rangle \ \circ \ GARB(\!|\,|\!)\end{aligned}$$

where $GARB = ! \heartsuit^{\perp}_{exitpc} \mid ! \heartsuit^{\perp}_{garbage}(\heartsuit^{\perp}_{exitpc})$ is the process on the garbage collector membrane.

The encoding of register r_j with content zero is $\langle\!\langle r_j = 0 \rangle\!\rangle = Z_j(\!|\,|\!)$, where $Z_j = ! \heartsuit_{inc_j} \mid ! \heartsuit^{\perp}_{decreq_j}(\heartsuit^{\perp}_{z}).\heartsuit^{\perp}_{exitpc}$.

The encoding of register r_j with content $n+1$ is

$$\langle\!\langle r_j = n+1 \rangle\!\rangle = S_j (\!| \mathcal{O}_{exitreg} (\!| \langle\!\langle r_j = n \rangle\!\rangle |\!) |\!)$$

where

$$S_j = !\mathcal{O}_{inc_j} \mid \mathcal{O}^{\perp}_{decreq_j}(\mathcal{O}^{\perp}_{nz}).\mathcal{O}^{\perp}_{sync}.\mathcal{O}^{\perp}_{exitreg}.\mathcal{O}_{garbage}.\mathcal{O}^{\perp}_{exitpc}$$

The encoding for the instruction $(i : I_i)$ is as follows:

$$\begin{array}{ll} \langle\!\langle (i : Succ(r_j)) \rangle\!\rangle & = \mathcal{O}^{\perp}_{p_i}(0).\mathcal{O}^{\perp}_{inc_j}(\mathcal{O}_{exitreg}).\mathcal{O}^{\perp}_{exitpc}.S_j (\!| \mathcal{O}_{exitpc} (\!| \mathcal{O}_{p_{i+1}} (\!|\ |\!) |\!) |\!) \\ \langle\!\langle (i : DecJump(r_j, s)) \rangle\!\rangle & = \mathcal{O}^{\perp}_{p_i}(0).\mathcal{O}_{decreq_j}.J_{i+1,s} \end{array}$$

where

$$\begin{array}{l} J_{h,k} = \mathcal{O}^{\perp}_{znz} (\!| \mathcal{O}_{znz} (\!| \mathcal{O}_{nz} (\!| \mathcal{O}_{sync} (\!|\ |\!) \circ \mathcal{O}_{exitpc} (\!| \mathcal{O}_{exitpc} (\!| \mathcal{O}_{exitpc} (\!| \mathcal{O}_{p_h} (\!|\ |\!) |\!) |\!) |\!) |\!) |\!) \ \circ \\ \qquad\qquad \mathcal{O}_z (\!| \mathcal{O}_{exitpc} (\!| \mathcal{O}_{p_k} (\!|\ |\!) |\!) |\!) \ |\!) \ |\!) \end{array}$$

It is easy to show that this encoding is deterministic, and that (a slight variation of) the results presented in the previous section hold also for this deterministic encoding.

An immediate consequence of Corollary 1 is the undecidability of existential termination:

Corollary 2. *The existential termination property is undecidable for PEP systems.*

4 Decidability of termination for MBD

In this section we show that the existence of a divergent computation is decidable for MBD.

The decidability proof exploits the techniques similar to the ones developed in [2] for (fragments of) Mobile Ambients [5], and is based on the theory of well-structured transition systems [8]: the existence of an infinite computation starting from a given state is decidable for finitely branching transition systems, provided that the set of states can be equipped with a well-quasi-ordering, i.e., a quasi-ordering relation which is compatible with the transition relation and such that each infinite sequence of states admits an increasing subsequence.

We start by providing some basic definitions and results on well-structured transition systems and on well-quasi-ordering on sequences of elements belonging to a well-quasi-ordered set, that will be used in the following parts of this Section.

Then, we prove the decidability of termination for MBD; to this aim, we first provide an alternative semantics that is equivalent w.r.t. termination to the one presented in Section 2, but which is based on a finitely branching transition system and permits to define a well-quasi-ordering on the derivatives of a given system (i.e., the set of systems reachable from a given initial system). Then, by exploiting the theory developed in [8], we show that termination is decidable for MBD systems.

4.1 Well-Structured Transition System

We start by recalling some basic definitions and results from [8], concerning well-structured transition systems, that will be used in the following.

A *quasi-ordering* (qo) is a reflexive and transitive relation.

Definition 10. *A* well-quasi-ordering *(wqo) is a quasi-ordering $\leq$ over a set X such that, for any infinite sequence $x_0, x_1, x_2, \ldots$ in X, there exist indexes $i < j$ such that $x_i \leq x_j$.*

Note that, if $\leq$ is a wqo, then any infinite sequence $x_0, x_1, x_2, \ldots$ contains an infinite increasing subsequence $x_{i_0}, x_{i_1}, x_{i_2}, \ldots$ (with $i_0 < i_1 < i_2 < \ldots$).

Transition systems can be formally defined as follows.

Definition 11. *A* transition system *is a structure $TS = (S, \rightarrow)$, where S is a set of* states *and $\rightarrow \subseteq S \times S$ is a set of* transitions.
We write $Succ(s)$ to denote the set $\{s' \in S \mid s \rightarrow s'\}$ of immediate successors of $s \in S$.
TS is finitely branching *if $\forall s \in S : Succ(s)$ is finite. We restrict to finitely branching transition systems.*

Well-structured transition systems, defined as follows, provide the key tool to decide properties of computations.

Definition 12. *A* well-structured transition system (with strong compatibility) *is a transition system $TS = (S, \rightarrow)$, equipped with a quasi-ordering $\leq$ on S, also written $TS = (S, \rightarrow, \leq)$, such that the two following conditions hold:*

1. **well-quasi-ordering***: $\leq$ is a well-quasi-ordering, and*
2. **strong compatibility***: $\leq$ is (upward) compatible with $\rightarrow$, i.e., for all $s_1 \leq t_1$ and all transitions $s_1 \rightarrow s_2$, there exists a state t_2 such that $t_1 \rightarrow t_2$ and $s_2 \leq t_2$.*

The following theorem (a special case of a result in [8]) will be used to obtain our decidability result.

Theorem 2. *Let $TS = (S, \rightarrow, \leq)$ be a finitely branching, well-structured transition system with decidable $\leq$ and computable Succ. The existence of an infinite computation starting from a state $s \in S$ is decidable.*

4.2 Higman's Lemma

To show that the quasi-ordering relation we will define on MBD systems is a well-quasi-ordering we need the following result, due to Higman [10] and stating that the set of the finite sequences over a set equipped with a wqo is well-quasi-ordered.

Given a set S, with S^* we denote the set of finite sequences of elements in S.

Definition 13. *Let S be a set and $\leq$ a wqo over S. The relation $\leq_*$ over S^* is defined as follows. Let $t, u \in S^*$, with $t = t_1 t_2 \ldots t_m$ and $u = u_1 u_2 \ldots u_n$. We have that $t \leq_* u$ iff there exists an injection f from $\{1, 2, \ldots, m\}$ to $\{1, 2, \ldots, n\}$ such that $t_i \leq u_{f(i)}$ and $i \leq f(i)$ for $i = 1, \ldots, m$.*

Note that relation $\leq_*$ is a quasi-ordering over S^*.

Lemma 3. *[Higman] Let S be a set and $\leq$ a wqo over S. Then, the relation $\leq_*$ is a wqo over S^*.*

Also the following propositions will be used to prove that the relation on systems is a well-quasi-ordering:

Proposition 2. *Let S be a finite set. Then the equality is a wqo over S.*

Proposition 3. *Let S, T be sets and $\leq_S$, $\leq_T$ be wqo over S and T, respectively. The relation $\leq$ over $S \times T$ is defined as follows: $(s_1, t_1) \leq (s_2, t_2)$ iff ($s_1 \leq_S s_2$ and $t_1 \leq_T t_2$). The relation $\leq$ is a wqo over $S \times T$.*

4.3 A finitely branching semantics for MBD systems

Because of the structural congruence rules, the reaction transition system for MBD is not finitely branching. To obtain a finitely branching transition system (with the same behavior w.r.t. termination), we take the transition system whose states are the equivalence classes of structural congruence.

Technically, it is possible to define a normal form for systems, up to the commutative and associative laws for the $\circ$ and $|$ operators.

In a system in normal form, the presence of a replicated version of a sequential process $!a.\sigma$ (resp. system $!(\sigma(\!| P |\!))$) forbids the presence of any occurrence of the nonreplicated version of the same process (resp. system), as well as of other occurrences of the replicated version of the process (resp. system). Moreover, replication is distributed over the components of parallel composition operators, and redundant replications and empty systems and terms are removed.

Definition 14. *Let $\stackrel{ca}{=}$ be the least congruence on systems satisfying the commutative and associative rules for $\circ$ and $|$.*

A brane process σ is in normal form if $\sigma \stackrel{ca}{=} \prod_{i \in I} a_i.\sigma_i \mid \prod_{j \in J} !a'_j.\sigma'_j$, where

- *σ_i and σ'_j are in normal form for $i \in I$ and $j \in J$;*
- *if $a_i = bud_n^{\perp}(\rho)$ or $a_i = drip(\rho)$ then ρ is in normal form;*
  if $a'_j = bud_n^{\perp}(\rho)$ or $a'_j = drip(\rho)$ then ρ is in normal form;
- *if $\sigma_i \stackrel{ca}{=} \sigma'_j$ then $a_i \stackrel{ca}{\neq} a'_j$;*
- *if $\sigma'_i \stackrel{ca}{=} \sigma'_j$ and $a'_i \stackrel{ca}{=} a'_j$ then $i = j$.*

A system P is in normal form if $P \stackrel{ca}{=} \bigcirc_{i \in I} \sigma_i(\!| P_i |\!) \circ \bigcirc_{j \in J} !(\sigma'_j(\!| P'_j |\!))$, where

- *σ_i, P_i, σ'_j and P'_j are in normal form for $i \in I$ and $j \in J$;*
- *if $P_i \stackrel{ca}{=} P'_j$ then $\sigma_i \stackrel{ca}{\neq} \sigma'_j$;*

– *if* $P'_i \stackrel{ca}{=} P'_j$ *and* $\sigma'_i \stackrel{ca}{=} \sigma'_j$ *then* $i = j$.

The function nf produces the normal form of a process or a system:

Definition 15. *The normal form of a process is defined inductively as follows:*

$$\begin{array}{ll} nf(0) & = 0 \\ nf(mate_n.\sigma) & = mate_n.nf(\sigma) \\ nf(mate_n^{\perp}, \sigma) & = mate_n^{\perp}.nf(\sigma) \\ nf(bud_n.\sigma) & = bud_n.nf(\sigma) \\ nf(bud_n^{\perp}(\rho).\sigma) & = bud_n^{\perp}(nf(\rho)).nf(\sigma) \\ nf(drip(\rho).\sigma) & = drip(nf(\rho)).nf(\sigma) \end{array}$$

Let $nf(\sigma) = \prod_{i\in I} a_i.\sigma_i \mid \prod_{j\in J} !a'_j.\sigma'_j$ *and* $nf(\tau) = \prod_{h\in H} b_h.\tau_h \mid \prod_{k\in K} !b'_k.\tau'_k$. *Then*

$$\begin{array}{rl} nf(\sigma \mid \tau) = & \prod\{a_i.\sigma_i \mid i \in I \wedge \forall k \in K : a_i.\sigma_i \stackrel{ca}{\neq} b'_k.\tau'_k\} \mid \\ & \prod\{b_h.\tau_h \mid h \in H \wedge \forall j \in J : b_h.\tau_h \stackrel{ca}{\neq} a'_j.\sigma'_j\} \mid \\ & \prod\{!a'_j.\sigma'_j \mid j \in J\} \mid \\ & \prod\{!b'_k.\tau'_k \mid k \in K \wedge \forall j \in J : b'_k.\tau'_k \stackrel{ca}{\neq} a'_j.\sigma'_j\} \end{array}$$

and

$$nf(!\sigma) = \prod\{!a_i.\sigma_i \mid i \in I\} \mid \prod\{!a'_j.\sigma'_j \mid j \in J\}$$

Definition 16. *The normal form of a system is defined inductively as follows:*

$$\begin{array}{ll} nf(\diamond) = & \diamond \\ nf(\sigma.P) = & nf(\sigma).nf(P) \end{array}$$

Let $nf(P) = \bigcirc_{i\in I} \sigma_i(\!| P_i |\!) \circ \bigcirc_{j\in J} !(\sigma'_j(\!| P'_j |\!))$ *and* $nf(Q) = \bigcirc_{h\in H} \tau_h(\!| Q_h, |\!) \circ \bigcirc_{k\in K} !(\tau'_k(\!| Q'_k |\!))$. *Then*

$$\begin{array}{rl} nf(P \circ Q) = & \bigcirc\{\sigma_i(\!| P_i |\!) \mid i \in I \wedge \forall k \in K : \sigma_i(\!| P_i |\!) \stackrel{ca}{\neq} \tau'_k(\!| Q'_k |\!)\} \circ \\ & \bigcirc\{\tau_h(\!| Q_h |\!) \mid h \in H \wedge \forall j \in J : \tau_h(\!| Q_h |\!) \stackrel{ca}{\neq} \sigma'_j(\!| P'_j |\!)\} \circ \\ & \bigcirc\{!\sigma'_j(\!| P'_j |\!) \mid j \in J\} \circ \\ & \bigcirc\{!\tau'_k(\!| Q'_k |\!) \mid k \in K \wedge \forall j \in J : \tau'_k(\!| Q'_k |\!) \stackrel{ca}{\neq} \sigma'_j(\!| P'_j |\!)\} \end{array}$$

$$nf(!P) = \bigcirc\{!\sigma_i(\!| P_i |\!) \mid i \in I\} \circ \bigcirc\{!\sigma'_j(\!| P'_j |\!) \mid j \in J\}$$

It is easy to see that function nf produces processes and systems in normal form.

Proposition 4. *Let* σ *be a process. Then* $nf(\sigma)$ *is a process in normal form. Let* P *be a system. Then* $nf(P)$ *is a system in normal form.*

Proof. By induction of the structure of σ (resp. P).

The normal form of two structurally congruent processes (resp. systems) is the same, up to associativity and commutativity of $\mid$ (resp. $\circ$) operator.

Proposition 5. *Let σ, τ be two processes. If $\sigma \equiv \tau$ then $nf(\sigma) \stackrel{ca}{=} nf(\tau)$. Let P, Q be two systems. If $P \equiv Q$ then $nf(P) \stackrel{ca}{=} nf(Q)$.*

Proof. By induction on the structure of the proof of $\sigma \equiv \tau$ (resp. $P \equiv Q$).

We define an alternative semantics for systems in normal form. This semantics turns out to be finitely branching, and equivalent to the reduction semantics of section 2.

Definition 17. *The reaction relation $\mapsto$ for systems in normal form is the least relation satisfying the axioms and rules in Tables 1, 2 and 3.*

The reduction relation $\mapsto$ is finitely branching over the set of systems in normal form:

Proposition 6. *Let P be a system in normal form. The set of immediate successors $Succ(P) = \{P' \mid P \mapsto P'\}$ is finite.*

Proof. By induction on the structure of P. Let $P = \bigcirc_{i \in I} \sigma_i(\!| P_i |\!) \circ \bigcirc_{j \in J} !(\sigma'_j(\!| P'_j |\!))$. By inductive hypothesis, the sets $Succ(P_i)$ and $Succ(P'_j)$ are finite for all $i \in I$ and $j \in J$.

Let $\#act(P)$ be the number of occurrences of actions in system P.

Consider a reduction $P \mapsto P'$. The last rule applied in the proof of this reduction can either be one of the axioms of Tables 1, 2 and 3 or one of the rules in Table 3.

Consider axiom (`mate1`): the number of reductions obtained by application of this axiom is bounded by the product of the number of actions *mate* occurring in P with the number of actions $mate^{\perp}$ occurring in P, which is smaller than $(\#act(P))^2$.

The same reasoning can be applied to each axiom, hence the number of immediate successors of P obtained by application of an axiom is bounded by $(10 + 9 + 3) \times (\#act(P))^2$.

Consider now the set of immediate successors obtained by application of rule (`brane1`). This set is bounded by $|I| \times max\{|Succ(P_i)| \mid i \in I\}$. By inductive hypothesis the sets of immediate successors of systems P_i, for $i \in I$, are finite. Hence also the set of immediate successors of P obtained by application of rule (`brane1`) is finite.

The same reasoning applies to rule (`brane2`), hence the set of immediate successors of P is finite.

We now show that a system P terminates (according to the reaction rule $\rightarrow$) iff $nf(P)$ terminates (according to the reaction rule $\mapsto$). To this aim, we need the following auxiliary results.

Lemma 4. *Let P be a system. If $P \rightarrow P'$ then $nf(P) \mapsto nf(P')$.*

Proof. By induction on the proof of $P \rightarrow P'$ (the difficult case is when the last rule in the proof tree is (`par`)).

Lemma 5. *Let P be a system. If $nf(P) \mapsto Q$ then $nf(P) \to Q$.*

Proof. By induction on the proof of $nf(P) \mapsto Q$.

Corollary 3. *Let P be a system. The system P terminates iff $nf(P)$ terminates.*

4.4 Decidability of termination for MBD systems

Let us consider a system P in normal form. In this section we provide a quasi-order on the derivatives of P (and a quasi-order on brane processes) that turns out to be a wqo compatible with $\mapsto$. Hence, exploiting the results in section 4.2, we obtain decidability of termination.

We note that each system (resp. process) in normal form is essentially a finite sequence of objects of kind $\sigma(\!|\, Q \,|\!)$ or $!(\sigma(\!|\, Q \,|\!))$ (resp. of objects of kind $a.\sigma$ or $!a.\sigma$). If we consider the nesting level of membranes, we note that each subsystem Q contained in a subterm $\sigma(\!|\, Q \,|\!)$ or $!(\sigma(\!|\, Q \,|\!))$ of a system R is simpler than R. More precisely, the maximum nesting level of membranes in Q is strictly smaller than the maximum nesting level of membranes in R. As already observed in [4], the reactions in MBD preserve the nesting level of membranes; hence, the nesting level of membranes in a system P provides an upper bound to the nesting level of membranes in the set of the (normal forms of the) derivatives of P.

Definition 18. *The nesting level of a system is defined inductively as follows:*

$$\begin{array}{ll} nl(\diamond) = & 0 \\ nl(\sigma(\!|P|\!)) = & nl(P) + 1 \\ nl(P \circ Q) = & max\{nl(P), nl(Q)\} \\ nl(!P) = & nl(P) \end{array}$$

Proposition 7. *Let P be a system in normal form. If $P \mapsto P'$ then $nl(P') \leq nl(P)$.*

Proof. By induction of the proof of $P \mapsto P'$.

Thanks to normal forms, we have that the set of processes of kind $a.\sigma$ or $!a.\sigma$ that occur as subterms in the derivatives (w.r.t. $\mapsto$) of a process in normal form is finite. This fact will be used to show that the quasi-orders on processes and on systems are wqo.

Definition 19. *Let P be a system in normal form. The set of derivatives of P w.r.t. $\mapsto$ is defined as follows: $nfDeriv(P) = \{P' \mid P \mapsto^* P'\}$.*

Definition 20. *The set of sequential and replicated sequential subprocesses of a process is defined inductively as follows:*

$$\begin{array}{ll} Subp(0) = & \emptyset \\ Subp(mate_n.\sigma) = & \{mate_n.\sigma\} \cup Subp(\sigma) \\ Subp(mate_n^{\perp}.\sigma) = & \{mate_n^{\perp}.\sigma\} \cup Subp(\sigma) \\ Subp(bud_n.\sigma) = & \{bud_n.\sigma\} \cup Subp(\sigma) \\ Subp(bud_n^{\perp}(\rho).\sigma) = & \{bud_n^{\perp}(\rho).\sigma\} \cup Subp(\rho) \cup Subp(\sigma) \\ Subp(drip(\rho).\sigma) = & \{drip(\rho).\sigma\} \cup Subp(\rho) \cup Subp(\sigma) \\ Subp(\sigma \mid \tau) = & Subp(\sigma) \cup Subp(\tau) \\ Subp(!\sigma) = & \{!\sigma' \mid \sigma' \in Subp(\sigma)\} \cup Subp(\sigma) \end{array}$$

The set of sequential and replicated sequential subprocesses of a system is defined inductively as follows:

$$\begin{array}{ll} Subp(\diamond) = & \emptyset \\ Subp(\sigma(\!|P|\!)) = & Subp(\sigma) \cup Subp(P) \\ Subp(P \circ Q) = & Subp(P) \cup Subp(Q) \\ Subp(!P) = & Subp(P) \end{array}$$

It is easy to see that the set of sequential and replicated sequential subprocesses of a process (resp. system) is finite:

Proposition 8. *Let σ be a process. The set $Subp(\sigma)$ is finite.*
Let P be a system. The set $Subp(P)$ is finite.

Proof. By induction on the structure of σ (resp. P).

The following proposition – stating that the transformation of a process (resp. system) in normal form does not increase its set of sequential and replicated sequential subprocesses – will be used to show that the set of sequential and replicated sequential subprocesses does not increase after execution of a reduction step.

Proposition 9. *Let σ be a process. Then $Subp(nf(\sigma)) \subseteq Subp(\sigma)$.*
Let P be a system. Then $Subp(nf(P)) \subseteq Subp(P)$.

Proof. By induction on the structure of σ (resp. P).

Proposition 10. *Let P be a process in normal form. If $P \mapsto P'$ then $Subp(P') \subseteq Subp(P)$.*

Proof. By induction on the proof of $P \mapsto P'$.

Definition 21. *Let P be a process in normal form. The set of subprocesses of the derivatives of P is defined as $SubDeriv(P) = \bigcup_{P' \in nfDeriv(P)} Subp(P')$.*

Proposition 11. *Let P be a process in normal form. Then the set $SubDeriv(P)$ is finite.*

Proof. Let $P' \in nfDeriv(P)$. Then $P \mapsto^* P'$. By induction on the length of the derivation $P \mapsto^* P'$ and by Proposition 10 we obtain $Subp(P') \subseteq Subp(P)$. By Proposition 8 we obtain that $SubDeriv(P)$ is finite.

We introduce a quasi-order $\preceq_{proc}$ on processes in normal form such that $\sigma \preceq_{proc} \tau$ if

- for each occurrence of a replicated guarded process at top-level in σ there is a corresponding occurrence of the same process at top-level in τ;
- for each occurrence of a guarded process at top-level in σ there is either a corresponding occurrence of the same process or an occurrence of the replicated version of the process at top-level in τ.

Definition 22. *Let σ and τ be two processes in normal form. Let $\sigma = \prod_{i\in I} a_i.\sigma_i \mid \prod_{j\in J} !a'_j.\sigma'_j$ and $\tau = \prod_{h\in H} b_h.\tau_h \mid \prod_{k\in K} !b'_k.\tau'_k$, and $H \cap K = \emptyset$. We say that $\sigma \preceq_{proc} \tau$ if there exists a pair of functions (f, g) such that:*

- $f : I \to H \cup K$ *and* $g : J \to K$
- $\forall i, i' \in I :$ *if* $f(i) = f(i')$ *and* $f(i) \in H$ *then* $i = i'$
- $\forall i \in I :$ *if* $f(i) \in H$ *then* $b_{f(i)}.\tau_{f(i)} \stackrel{ca}{=} a_i.\sigma_i$
- $\forall i \in I :$ *if* $f(i) \in K$ *then* $b'_{f(i)}.\tau'_{f(i)} \stackrel{ca}{=} a_i.\sigma_i$
- $\forall j \in J : b'_{g(j)}.\tau'_{g(j)} \stackrel{ca}{=} a'_j.\sigma'_j$

We define a quasi-order on systems such that $R \preceq_{sys} S$ if

- for each replicated membrane $!(\rho(\!| R_1 |\!))$ at top-level in R there is a corresponding replicated membrane $!(\sigma(\!| S_1 |\!))$ at top-level in S such that ρ is smaller than σ and R_1 is smaller than S_1;
- for each occurrence of a membrane $\rho(\!| R_1 |\!)$ at top-level in R there is
  - either a corresponding occurrence of a membrane $\sigma(\!| S_1 |\!)$ at top-level in S such that ρ is smaller than σ and R_1 is smaller than S_1
  - or an occurrence of a replicated membrane $!(\rho(\!| R_1 |\!))$ at top-level in S.

Definition 23. *Let P, Q be systems. Let $P = \bigcirc_{i\in I} \sigma_i(\!| P_i |\!) \circ \bigcirc_{j\in J} !(\sigma'_j(\!| P'_j |\!))$ and $Q = \bigcirc_{h\in H} \tau_h(\!| Q_h, |\!) \circ \bigcirc_{k\in K} !(\tau'_k(\!| Q'_k |\!))$ and $H \cap K = \emptyset$. We say that $P \preceq_{sys} Q$ if there exists a pair of functions (f, g) such that:*

- $f : I \to H \cup K$ *and* $g : J \to K$
- $\forall i, i' \in I :$ *if* $f(i) = f(i')$ *and* $f(i) \in H$ *then* $i = i'$
- $\forall i \in I :$ *if* $f(i) \in H$ *then* $\sigma_i \preceq_{proc} \tau_{f(i)}$ *and* $P_i \preceq_{sys} Q_{f(i)}$
- $\forall i \in I :$ *if* $f(i) \in K$ *then* $\sigma_i \preceq_{proc} \tau'_{f(i)}$ *and* $P_i \preceq_{sys} Q'_{f(i)}$
- $\forall j \in J : \sigma'_j \preceq_{proc} \tau'_{g(j)}$ *and* $P'_j \preceq_{sys} Q'_{g(j)}$

It is easy to see that $\preceq_{proc}$ and $\preceq_{sys}$ are quasi-orderings.

We start showing that the relation $\preceq_{sys}$ is strongly compatible with $\mapsto$. To this aim, we need the following auxiliary propositions:

Proposition 12. *Let $\sigma_1, \sigma_2, \tau_1, \tau_2$ be processes in normal form. If $\sigma_i \preceq_{proc} \tau_i$ for $i = 1, 2$ then $nf(\sigma_1 \mid \sigma_2) \preceq_{proc} nf(\tau_1 \mid \tau_2)$.*

Proposition 13. *Let P_1, P_2, Q_1, Q_2 be systems in normal form. If $P_i \preceq_{sys} Q_i$ for $i = 1, 2$ then $P_1 \circ P_2 \preceq_{sys} Q_1 \circ Q_2$.*

Theorem 3. *Let P, P', Q be systems in normal form. If $P \mapsto P'$ and $P \preceq_{sys} Q$ then there exists Q' in normal form such that $Q \mapsto Q'$ and $Q \preceq_{sys} Q'$.*

Proof. The proof is by induction on the structure of P, then by case analysis on the last axiom or rule applied to obtain the reduction $P \mapsto P'$.

The following proposition will be used to show that $\preceq_{proc}$ is a wqo over the set of subprocesses of the derivatives of a system P.

Proposition 14. *Let σ and τ be two processes in normal form.*
Let $\sigma = \prod_{i=1}^{n_1} a_i.\sigma_i \mid \prod_{j=n_1+1}^{n_1+n_2} !a'_j.\sigma'_j$ and $\tau = \prod_{h=1}^{m_1} b_h.\tau_h \mid \prod_{k=m_1+1}^{m_1+m_2} !b'_k.\tau'_k$.
If $a_1.\sigma_1 \ldots a_{n_1}.\sigma_{n_1} !a'_{n_1+1}.\sigma'_{n_1+1} \ldots !a'_{n_1+n_2}.\sigma'_{n_1+n_2} =_*$
$b_1.\tau_1 \ldots b_{m_1}.\tau_{m_1} !b'_{m_1+1}.\tau'_{m_1+1} \ldots !b'_{m1+m_2}.\tau'_{m1+m_2}$ then $\sigma \preceq_{proc} \tau$.

Proof. If $a_1.\sigma_1 \ldots a_{n_1}.\sigma_{n_1} !a'_{n_1+1}.\sigma'_{n_1+1} \ldots !a'_{n_1+n_2}.\sigma'_{n_1+n_2} =_*$
$b_1.\tau_1 \ldots b_{m_1}.\tau_{m_1} !b'_{m_1+1}.\tau'_{m_1+1} \ldots !b'_{m1+m_2}.\tau'_{m1+m_2}$ then there exists an injection $f : \{1, \ldots, n_1 + n_2\} \to \{1, \ldots, m_1 + m_2\}$ mapping the each element of the first sequence into a corresponding, equal element in the second sequence. The first n_1 (resp m_1) elements of the first (resp. second) sequence are sequential processes, whereas the last n_2 (resp. m_2) elements of the first (resp. second) sequence are replicated sequential processes. Hence the first n_1 (resp. last n_2) elements of the first sequence will be mapped on the first m_1 (resp. last m_2) elements of the second sequence.

Consider the pair of functions (g_1, g_2) such that

- $g_1(i) = f(i)$ for $i = 1, \ldots, n_1$
- $g_2(i) = f(i)$ for $i = n_1 + 1, \ldots, n_1 + n_2$

The pair of functions (g_1, g_2) satisfies the conditions of Definition 22, hence $\sigma \preceq_{proc} \tau$.

By Higman lemma and Proposition 2 it easy to prove that

Lemma 6. *Let P be a system in normal form. The relation $\preceq_{proc}$ is a wqo over the set of processes in $SubDeriv(P)$.*

Proof. Take an infinite sequence $\sigma_1, \sigma_2, \ldots, \sigma_h, \ldots$ of normal form processes in $SubDeriv(P)$. Let $\sigma_h = \prod_{i=1}^{n_h} a_{i,h}.\sigma_{i,h} \mid \prod_{j=1}^{m_h} !a'_{j,h}.\sigma'_{j,h}$. By definition of $SubDeriv$ and $Subp$, we have that $\forall h > 0 : \forall 1 \leq i \leq n_h : a_{i,h}.\sigma_{i,h} \in SubDeriv(P)$ and $\forall h > 0 : \forall 1 \leq i \leq m_h : !a'_{j,h}.\sigma_{j,h} \in SubDeriv(P)$. Hence, we have an infinite sequence of elements of $SubDeriv(P)^*$; as $SubDeriv(P)$ is finite (by Proposition 11), by Proposition 2 and Higman Lemma (Lemma 3) we have that $=_*$ is a wqo over $SubDeriv(P)$. Thus there exist i, j such that
$a_{1,i}.\sigma_{1,i} \ldots a_{n_i,i}.\sigma_{n_i,i} !a'_{1,i}.\sigma'_{1,i} \ldots !a'_{m_i,i}.\sigma'_{m_i,i} =_*$
$a_{1,j}.\sigma_{1,j} \ldots a_{n_j,j}.\sigma_{n_j,j} !a'_{1,j}.\sigma'_{1,j} \ldots !a'_{m_j,j}.\sigma'_{m_j,j}$.
By Proposition 14 we have $\sigma_i \preceq_{proc} \sigma_j$.

Now it is possible to prove that $\preceq_{sys}$ is a wqo.

The set of derivatives of a system P w.r.t. $\mapsto$ with nesting level not greater than n is defined as

Definition 24. *Let P be a system in normal form and $n \geq 0$. We define $nfDeriv_n(P) = \{Q \mid Q \in nfDeriv(P) \wedge nl(Q) \leq n\}$.*

Proposition 15. *Let P be a system in normal form. Then $nfDeriv(P) = nfDeriv_{nl(P)}(P)$.*

Proof. A consequence of Proposition 7.

Now we show that $\preceq_{sys}$ is a wqo over a subset of derivatives whose elements have a nesting level smaller than a given natural number. The proof proceeds by induction on the nesting level of membranes, and makes use of Higman's Lemma, of Lemma 6 and of Proposition 3.

Theorem 4. *Let P be a system in normal form and $n \geq 0$. The relation $\preceq_{sys}$ is a wqo over the set $nfDeriv_n(P)$.*

Proof. The proof is by induction on n.

The case $n = 0$ is trivial, as $nfDeriv_0(P) \subseteq \{\diamond\}$.

For the inductive step, take $n > 0$ and an infinite sequence $P_1, P_2, \ldots, P_h, \ldots$ with $P_h \in nfDeriv_n(P)$.

Let $P_h = \bigcirc_{i=1}^{n_h} \sigma_{i,h}(\!| P_{i,h} |\!) \circ \bigcirc_{j=n_h+1}^{m_h} !(\sigma'_{j,h}(\!| P'_{j,h} |\!))$.

As $P_h \in nfDeriv_n(P)$, we have that $\sigma_{i,h} \in Subp(P)$ for $i = 1, \ldots, n_h$ and $\sigma'_{j,h} \in Subp(P)$ for $j = n_h + 1, \ldots, m_h$.

Moreover, $nl(P_h) \leq n$; by definition of nesting level, we obtain $nl(P_{i,h}) \leq n - 1$ for $i = 1, \ldots, n_h$ and $nl(P'_{j,h}) \leq n - 1$ for $j = n_h + 1, \ldots, m_h$.

Hence, $P_{i,h} \in nfDeriv_{n-1}(P)$ for $i = 1, \ldots, n_h$ and $P'_{j,h} \in nfDeriv_{n-1}(P)$ for $j = n_h + 1, \ldots, m_h$.

By Lemma 6 we know that $\preceq_{proc}$ is a wqo over $Subp(P)$. By inductive hypothesis, we have that $\preceq_{sys}$ is a wqo over $nfDeriv_{n-1}(P)$.

By Proposition 3 we obtain that $\preceq = (\preceq_{proc}, \preceq_{sys})$ is a wqo over $Subp(P) \times nfDeriv_{n-1}(P)$.

By Higman Lemma we obtain that $\preceq_*$ is a wqo over $(Subp(P) \times nfDeriv_{n-1}(P))^*$.

By Proposition 3 we obtain that $\preceq' = (\preceq_*, \preceq_*)$ is a wqo over $(Subp(P) \times nfDeriv_{n-1}(P))^* \times (Subp(P) \times nfDeriv_{n-1}(P))^*$.

Consider the infinite sequence $s_1, \ldots, s_h, \ldots$ with

$$\begin{aligned} s_h = (&((\sigma_{1,h}, P_{1,h}), \ldots, (\sigma_{n_h,h}, P_{n_h,h})),\\ &((\sigma'_{n_h+1,h}, P'_{n_h+1,h}), \ldots, (\sigma'_{n_h+m_h,h}, P'_{n_h+m_h,h}))) \end{aligned}$$

We have that $s_h \in (Subp(P) \times nfDeriv_{n-1}(P))^* \times (Subp(P) \times nfDeriv_{n-1}(P))^*$ for $h > 0$. As $\preceq'$ is a wqo over such a set, there exist k, q such that $s_k \preceq' s_q$.

This means that

$$(\sigma_{1,k}, P_{1,k}), \ldots, (\sigma_{n_k,k}, P_{n_k,k}) \preceq_* (\sigma_{1,q}, P_{1,q}), \ldots, (\sigma_{n_q,q}, P_{n_q,q})$$

Thus there exists an injection $f : \{1, \ldots, n_k\} \to \{1, \ldots, n_q\}$ such that $\sigma_{i,k} \preceq_{proc} \sigma_{i,f(k)}$ and $P_{i,k} \preceq_{sys} P_{i,f(k)}$ for $i = 1, \ldots n_k$.

We also have that

$$\begin{aligned} &(\sigma'_{n_k+1,k}, P'_{n_k+1,k}), \ldots, (\sigma'_{n_k+m_k,k}, P'_{n_k+m_k,k}) \preceq_*\\ &(\sigma'_{1,q}, P'_{1,q}), \ldots, (\sigma'_{n_q+m_q,q}, P'_{n_q+m_q,q}) \end{aligned}$$

Thus there exists an injection $g : \{n_h + 1, \ldots, n_h + m_k\} \to \{n_q + 1, \ldots, n_q + m_q\}$ such that $\sigma'_{i,k} \preceq_{proc} \sigma'_{i,g(k)}$ and $P'_{i,k} \preceq_{sys} P'_{i,g(k)}$ for $i = n_k + 1, \ldots n_k + m_k$.

Consider the systems P_k and P_q and the pair of functions (f, g). We have that $P_k = \bigcirc_{i=1}^{n_k} \sigma_{i,k}(\!| P_{i,k} |\!) \circ \bigcirc_{j=1}^{m_k} !(\sigma'_{j,k}(\!| P'_{j,k} |\!))$ and $P_q = \bigcirc_{i=1}^{n_q} \sigma_{i,q}(\!| P_{i,q} |\!) \circ \bigcirc_{j=1}^{m_q} !(\sigma'_{j,q}(\!| P'_{j,q} |\!))$. Moreover, the pair of functions (f, g) satisfies the requirements of Definition 23. Hence, $P_k \preceq_{sys} P_q$.

Summing up, we have showed that $\preceq_{sys}$ is a wqo over $nfDeriv_n(P)$.

Theorem 5. *Let P be a system in normal form. The relation $\preceq_{sys}$ is a wqo over the set $nfDeriv(P)$.*

Proof. An easy consequence of Theorem 4 and of Proposition 15.

The following theorem ensures that the hypothesis of Theorem 2 are satisfied.

Theorem 6. *Let P be a system in normal form. Then the transition system $(nfDeriv(P), \mapsto, \preceq_{sys})$ is a well-structured transition system with decidable $\preceq_{sys}$ and computable Succ.*

Proof. Strong compatibility of $\preceq_{sys}$ with the transition relation $\mapsto$ has been proved in Theorem 3. By Theorem 5 we have that $\preceq_{sys}$ is a wqo over $nfDeriv(P)$. From Definition 23 it is possible to deduce an effective procedure to check $\preceq_{sys}$. From Definitions 16 and 17 it is possible to deduce an effective procedure to compute *Succ*.

By the above theorem and Theorem 2 we get the following

Corollary 4. *Let P be a MBD system. The termination of P is decidable.*

5 Conclusion

The aim of this paper is to investigate and compare the expressiveness of the two basic brane calculi PEP and MBD w.r.t. their ability to encode computable functions.
Regarding the PEP calculus we provided the following results:

(1) an encoding of RAMs that preserves the universal termination property, i.e., the existence of a divergent computation;
(2) an improved, deterministic encoding of RAMs that faithfully models the RAM behavior, in the following sense:
- If the RAM terminates then the (unique) computation of the RAM encoding terminates;
- if the RAM does not terminate then the (unique) computation of the RAM encoding diverges.

The impact of the above results on the decidability of termination properties in PEP is the following:

- as a consequence of the existence of the encoding (1), we obtain the undecidability of universal termination for PEP systems. Note that the existence of the encoding (1) does not imply the undecidability of existential termination on PEP systems, because PEP systems are in general nondeterministic, and may have both a terminating and a divergent computation.

– as a consequence of the existence of the deterministic encoding (2), and of the equivalence of existential and universal termination properties for deterministic systems, we also obtain the undecidability of existential termination for PEP systems.

Regarding the MBD calculus, we provided the following result:

(3) Universal termination is decidable on MBD systems.

As a consequence of the decidability of universal termination, we obtain the impossibility to provide an encoding of RAMs in MBD that preserves the universal termination property, i.e., an encoding that has a divergent computation if and only if the RAM diverges. Hence, it is also impossible to define a deterministic encoding of RAMs in MBD.

From the above results we deduce the following expressiveness gap between PEP and MBD, w.r.t. their ability to reproduce the behavior of a RAM: there exists a deterministic (hence, both universal and existential termination preserving) encoding of RAMs in PEP, but there exist neither a deterministic nor a universal termination preserving encoding of RAMs in MBD. Regarding the decidability of properties, we have that universal termination is decidable in MBD, whereas it turns out to be undecidable in PEP. Hence, there exist no encoding of PEP in MBD that preserves the existence of a divergent computation.

Regarding the MBD calculus, the decidability of universal termination (3) does not tell anything about the decidability of existential termination, and about the possibility to define a weaker, nondeterministic encoding of RAMs in MBD, that preserves existential termination (i.e., the encoding has a terminating computation if and only if the RAM terminates). This topic has been investigated in [1], where a nondeterministic encoding of RAMs in MBD, which preserves existential termination, is provided. As a consequence of this result, we obtain the undecidability of the existential termination property for MBD.

The comparison of a (nondeterministic) model with its deterministic fragment is an interesting topic in automata theory, that has recently attracted the interest of the research community working on membrane computing (see, e.g., [11] and the references therein). From the results presented in this paper and in [1], we deduce the following:

– The deterministic fragment of PEP is as powerful as the full (nondeterministic) PEP calculus w.r.t. the ability to encode RAMs;
– There exists a gap between the deterministic fragment and the full MBD calculus, as there exists a weak, existential termination preserving encoding of RAMs in MBD [1], but there exist no such encoding in the deterministic fragment of MBD (this is a consequence of the decidability of universal termination on MBD, and of the equivalence of universal and existential termination on deterministic systems).

The results presented in this paper hold for the interleaving semantics, according to which a single interaction is executed at each computational step. While interleaving semantics is the classical semantics for process calculi, the

usual semantics in membrane computing is based on maximal parallelism: at each computational step, a maximal set of independent interactions must be executed simultaneously. As for many variants of P systems the computational power decreases when moving from the maximal parallelism to the interleaving semantics (see [9]), it seems worthwhile to investigate if the decidability of universal termination for MBD with the interleaving semantics also holds when moving to the maximal parallelism semantics. Concerning MBD with the maximal parallelism semantics, in [1] we provided a deterministic encoding of RAMs. Hence, both the universal and existential termination properties turn out to be undecidable in MBD with the maximal parallelism semantics. From the decidability of universal termination in MBD with interleaving semantics (3), we obtain an expressiveness gap between MBD with the interleaving and with the maximal parallelism semantics, thus confirming the intuition emerging from [9] for P systems: in most cases the computational power increases when moving from interleaving to maximal parallelism. Note that the undecidability of universal termination for MBD with maximal parallelism semantics does not contradict the decidability of universal termination for MBD with interleaving semantics provided in the present paper, as the well-quasi-ordering on systems $\preceq_{sys}$ defined in Section 4 is not compatible with the maximal parallelism reduction semantics. Consider, e.g., the systems $P_1 = mate_n(\!|\,|\!) \circ mate_n^{\perp}(\!|\,|\!) \circ mate_m(\!|\,|\!)$ and $P_2 = P_1 \circ mate_m^{\perp}(\!|\,|\!)$. We have that $P_1 \preceq_{sys} P_2$; the only reduction step that P_1 can perform is the fusion of the two membranes $mate_n(\!|\,|\!)$ and $mate_n^{\perp}(\!|\,|\!)$; thus, according to the maximal parallelism semantics, $P_1 \Rightarrow mate_m(\!|\,|\!)$. On the other hand, two independent fusions can be performed by P_2; thus, according to the maximal parallelism semantics, both fusions must be performed, and the only computational step for P_2 is $P_2 \Rightarrow \diamond$. Thus, $P_1 \preceq_{sys} P_2$ and $P_1 \Rightarrow mate_m(\!|\,|\!)$, but there exist no system P_2' such that $P_2 \Rightarrow P_2'$ and $mate_m(\!|\,|\!) \preceq_{sys} P_2'$, thus preventing the compatibility of $\preceq_{sys}$ with $\Rightarrow$ to hold.

In [6] a variant of P systems, inspired by the interaction primitives of the Brane Calculi, has been proposed and investigated. Quite surprisingly, it is shown that the class of P systems containing only the *mate* and *drip* operations is Turing powerful. A deep comparison of Brane Calculi and the class of P systems proposed in [6] deserves further investigation. At a first sight, it seems that the interaction primitives used in P systems are more powerful than the MBD primitives. Some other, evident differences are the following: Brane Calculi are equipped with an interleaving semantics, whereas P systems have a maximal parallelism semantics; in [6] only a finite number of membranes is sufficient to achieve Turing completeness, whereas in the present paper (and in [1]) an unbounded number of membranes is required.

In the present paper we showed that universal termination is a decidable property for MBD. The technique employed to prove the decidability of universal termination is based on the theory of well-structured transition systems: besides universal termination, such a theory permits to analyse other interesting properties, such as, e.g., coverability, boundedness, and eventuality proper-

ties [8]. We plan to investigate the possibility to use this theory for the analysis of biologically relevant properties.

We also plan to extend our investigation to recent refinements of Brane Calculi, such as, e.g., the Projective Brane Calculus [7], as well as to quantitative variants of Brane Calculi, along the lines of, e.g., [15].

Acknowledgement: We thank the anonymous referees for their comments. We are indebted to Fabien Tarissan for his careful comments and precious suggestions.

References

1. N. Busi. On the computational power of the Mate/Bud/Drip Brane Calculus: interleaving vs. maximal parallelism. Proc 6th International Workshop on Membrane Computing (WMC6), LNCS 3850, Springer, 2006.
2. N. Busi and G. Zavattaro. On the expressive power of movement and restriction in pure mobile ambients. *Theoretical Computer Science*, 322:477–515, 2004.
3. L. Cardelli. Brane Calculi - Interactions of biological membranes. Proc. Computational Methods in System Biology 2004 (CMSB 2004), LNCS 3082, Springer, 2005.
4. L. Cardelli. Abstract Machines for System Biology. Draft, 2005.
5. L. Cardelli and A.D. Gordon. Mobile Ambients. *Theoretical Computer Science*, 240(1):177–213, 2000.
6. L. Cardelli and G. Păun. An universality result for a (Mem)Brane Calculus based on mate/drip operations. International Journal of Foundations of Computer Science, to appear.
7. V. Danos and S. Pradalier. Projective Brane Calculus. Proc. Computational Methods in System Biology 2004 (CMSB 2004), LNCS 3082, Springer, 2005.
8. A. Finkel and Ph. Schnoebelen. Well-Structured Transition Systems Everywhere! *Theoretical Computer Science*, 256:63–92, Elsevier, 2001.
9. R. Freund. Asynchronous P Systems and P Systems Working in the Sequential Mode Proc. 5th International Workshop on Membrane Computing (WMC5), LNCS 3365, Springer, 2005.
10. G. Higman. Ordering by divisibility in abstract algebras. In *Proc. London Math. Soc.*, vol. 2, pages 236–366, 1952.
11. O. H. Ibarra. Some Recent Results Concerning Deterministic P Systems. Proc 6th International Workshop on Membrane Computing (WMC6), LNCS 3850, Springer, 2006.
12. M.L. Minsky. *Computation: finite and infinite machines.* Prentice-Hall, 1967.
13. G. Păun. *Membrane Computing. An Introduction.* Springer, 2002.
14. G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.
15. C. Priami, A. Regev, W. Silverman, and E. Shapiro. Application of a stochastic passing-name calculus to representation and simulation of molecular processes. Information Processing Letter, 80:25-31, 2001.
16. A. Regev, E. M. Panina, W. Silverman, L. Cardelli, E. Shapiro. BioAmbients: An Abstraction for Biological Compartments. *Theoretical Computer Science*, 325(1):141–167, Elsevier, 2004.
17. J.C. Shepherdson and J.E. Sturgis. Computability of recursive functions. *Journal of the ACM*, 10:217–255, 1963.

(mate1) $mate_n.\sigma|\sigma_0(\!| P |\!) \circ mate_n^{\perp}.\tau|\tau_0(\!| Q |\!) \circ R \mapsto$
$nf(\sigma|\sigma_0|\tau|\tau_0(\!| P \circ Q |\!) \circ R)$

(mate2) $(!mate_n.\sigma)|\sigma_0(\!| P |\!) \circ mate_n^{\perp}.\tau|\tau_0(\!| Q |\!) \circ R \mapsto$
$nf(\sigma|(!mate_n.\sigma)|\sigma_0|\tau|\tau_0(\!| P \circ Q |\!) \circ R)$

(mate3) $!(mate_n.\sigma|\sigma_0(\!| P |\!)) \circ mate_n^{\perp}.\tau|\tau_0(\!| Q |\!) \circ R \mapsto$
$nf(\sigma|\sigma_0|\tau|\tau_0(\!| P \circ Q |\!) \circ !(mate_n.\sigma|\sigma_0(\!| P |\!)) \circ R)$

(mate4) $mate_n.\sigma|\sigma_0(\!| P |\!) \circ (!mate_n^{\perp}.\tau)|\tau_0(\!| Q |\!) \circ R \mapsto$
$nf(\sigma|\sigma_0|\tau|(!mate_n^{\perp}.\tau)|\tau_0(\!| P \circ Q |\!) \circ R)$

(mate5) $(!mate_n.\sigma)|\sigma_0(\!| P |\!) \circ (!mate_n^{\perp}.\tau)|\tau_0(\!| Q |\!) \circ R \mapsto$
$nf(\sigma|(!mate_n.\sigma)|\sigma_0|\tau|(!mate_n^{\perp}.\tau)|\tau_0(\!| P \circ Q |\!) \circ R)$

(mate6) $!(mate_n.\sigma|\sigma_0(\!| P |\!)) \circ !(mate_n^{\perp}.\tau)|\tau_0(\!| Q |\!) \circ R \mapsto$
$nf(\sigma|\sigma_0|\tau|!(mate_n^{\perp}.\tau)|\tau_0(\!| P \circ Q |\!) \circ !(mate_n.\sigma|\sigma_0(\!| P |\!)) \circ R)$

(mate7) $mate_n.\sigma|\sigma_0(\!| P |\!) \circ !(mate_n^{\perp}.\tau|\tau_0(\!| Q |\!)) \circ R \mapsto$
$nf(\sigma|\sigma_0|\tau|\tau_0(\!| P \circ Q |\!) \circ !(mate_n^{\perp}.\tau|\tau_0(\!| Q |\!)) \circ R)$

(mate8) $(!mate_n.\sigma)|\sigma_0(\!| P |\!) \circ !(mate_n^{\perp}.\tau|\tau_0(\!| Q |\!)) \circ R \mapsto$
$nf(\sigma|(!mate_n.\sigma)|\sigma_0|\tau|\tau_0(\!| P \circ Q |\!) \circ !(mate_n^{\perp}.\tau|\tau_0(\!| Q |\!)) \circ R)$

(mate9) $!(mate_n.\sigma|\sigma_0(\!| P |\!)) \circ !(mate_n^{\perp}.\tau|\tau_0(\!| Q |\!)) \circ R \mapsto$
$nf(\sigma|\sigma_0|\tau|\tau_0(\!| P \circ Q |\!) \circ !(mate_n.\sigma|\sigma_0(\!| P |\!)) \circ !(mate_n^{\perp}.\tau|\tau_0(\!| Q |\!)) \circ R)$

(mate10) $!(mate_n.\sigma|mate_n^{\perp}.\sigma'|\sigma_0(\!| P |\!)) \circ R \mapsto$
$nf(\sigma|\sigma'|\sigma_0|\sigma_0(\!| P \circ P |\!) \circ !(mate_n.\sigma|mate_n^{\perp}.\sigma'|\sigma_0(\!| P |\!)) \circ R)$

(mate11) $!((!mate_n.\sigma)|mate_n^{\perp}.\sigma'|\sigma_0(\!| P |\!)) \circ R \mapsto$
$nf(\sigma|(!mate_n.\sigma)|\sigma'|\sigma_0|\sigma_0(\!| P \circ P |\!) \circ$
$\quad !((!mate_n.\sigma)|mate_n^{\perp}.\sigma'|\sigma_0(\!| P |\!)) \circ R)$

(mate12) $!(mate_n.\sigma|(!mate_n^{\perp}.\sigma')|\sigma_0(\!| P |\!)) \circ R \mapsto$
$nf(\sigma|\sigma'|(!mate_n^{\perp}.\sigma')|\sigma_0|\sigma_0(\!| P \circ P |\!) \circ$
$\quad !(mate_n.\sigma|(!mate_n^{\perp}.\sigma')|\sigma_0(\!| P |\!)) \circ R)$

(mate13) $!((!mate_n.\sigma)|(!mate_n^{\perp}.\sigma')|\sigma_0(\!| P |\!)) \circ R \mapsto$
$nf(\sigma|(!mate_n.\sigma)|\sigma'|(!mate_n^{\perp}.\sigma')|\sigma_0|\sigma_0(\!| P \circ P |\!) \circ$
$\quad !((!mate_n.\sigma)|(!mate_n^{\perp}.\sigma')|\sigma_0(\!| P |\!)) \circ R)$

Table 1. The axioms for the reduction relation $\mapsto$ (mating).

(bud1) $bud_n^{\perp}(\rho).\tau|\tau_0(\!|\, bud_n.\sigma|\sigma_0(\!|\, P \,|\!) \circ Q \,|\!) \;\circ R \mapsto$
$nf(\rho(\!|\, \sigma|\sigma_0(\!|\, P \,|\!) \,|\!) \;\circ\; \tau|\tau_0(\!|\, Q \,|\!) \;\circ R)$

(bud2) $(!bud_n^{\perp}(\rho).\tau)|\tau_0(\!|\, bud_n.\sigma|\sigma_0(\!|\, P \,|\!) \circ Q \,|\!) \;\circ R \mapsto$
$nf(\rho(\!|\, \sigma|\sigma_0(\!|\, P \,|\!) \,|\!) \;\circ\; \tau|(!bud_n^{\perp}(\rho).\tau)|\tau_0(\!|\, Q \,|\!) \;\circ R)$

(bud3) $!(bud_n^{\perp}(\rho).\tau|\tau_0(\!|\, bud_n.\sigma|\sigma_0(\!|\, P \,|\!) \circ Q \,|\!)) \;\circ R \mapsto$
$nf(\rho(\!|\, \sigma|\sigma_0(\!|\, P \,|\!) \,|\!) \;\circ\; \tau|\tau_0(\!|\, Q \,|\!) \circ$
$!(bud_n^{\perp}(\rho).\tau|\tau_0(\!|\, bud_n.\sigma|\sigma_0(\!|\, P \,|\!) \circ Q \,|\!)) \;\circ R)$

(bud4) $bud_n^{\perp}(\rho).\tau|\tau_0(\!|\, !(bud_n.\sigma)|\sigma_0(\!|\, P \,|\!) \circ Q \,|\!) \;\circ R \mapsto$
$nf(\rho(\!|\, \sigma|!(bud_n.\sigma)|\sigma_0(\!|\, P \,|\!) \,|\!) \;\circ\; \tau|\tau_0(\!|\, Q \,|\!) \;\circ R)$

(bud5) $(!bud_n^{\perp}(\rho).\tau)|\tau_0(\!|\, !(bud_n.\sigma)|\sigma_0(\!|\, P \,|\!) \circ Q \,|\!) \;\circ R \mapsto$
$nf(\rho(\!|\, \sigma|!(bud_n.\sigma)|\sigma_0(\!|\, P \,|\!) \,|\!) \;\circ\; \tau|(!bud_n^{\perp}(\rho).\tau)|\tau_0(\!|\, Q \,|\!) \;\circ R)$

(bud6) $!(bud_n^{\perp}(\rho).\tau|\tau_0(\!|\, !(bud_n.\sigma)|\sigma_0(\!|\, P \,|\!) \circ Q \,|\!)) \;\circ R \mapsto$
$nf(\rho(\!|\, \sigma|!(bud_n.\sigma)|\sigma_0(\!|\, P \,|\!) \,|\!) \;\circ\; \tau|\tau_0(\!|\, Q \,|\!) \circ$
$!(bud_n^{\perp}(\rho).\tau|\tau_0(\!|\, !(bud_n.\sigma)|\sigma_0(\!|\, P \,|\!) \circ Q \,|\!)) \;\circ R)$

(bud7) $bud_n^{\perp}(\rho).\tau|\tau_0(\!|\, !(bud_n.\sigma|\sigma_0(\!|\, P \,|\!)) \circ Q \,|\!) \;\circ R \mapsto$
$nf(\rho(\!|\, \sigma|\sigma_0(\!|\, P \,|\!) \,|\!) \;\circ\; \tau|\tau_0(\!|\, !(bud_n.\sigma|\sigma_0(\!|\, P \,|\!)) \circ Q \,|\!) \;\circ R)$

(bud8) $(!bud_n^{\perp}(\rho).\tau)|\tau_0(\!|\, !(bud_n.\sigma|\sigma_0(\!|\, P \,|\!)) \circ Q \,|\!) \;\circ R \mapsto$
$nf(\rho(\!|\, \sigma|\sigma_0(\!|\, P \,|\!) \,|\!) \;\circ\; \tau|(!bud_n^{\perp}(\rho).\tau)|\tau_0(\!|\, !(bud_n.\sigma|\sigma_0(\!|\, P \,|\!)) \circ Q \,|\!) \;\circ R)$

(bud9) $!(bud_n^{\perp}(\rho).\tau|\tau_0(\!|\, !(bud_n.\sigma|\sigma_0(\!|\, P \,|\!)) \circ Q \,|\!)) \;\circ R \mapsto$
$nf(\rho(\!|\, \sigma|\sigma_0(\!|\, P \,|\!) \,|\!) \;\circ\; \tau|\tau_0(\!|\, !(bud_n.\sigma|\sigma_0(\!|\, P \,|\!)) \circ Q \,|\!) \circ$
$!(bud_n^{\perp}(\rho).\tau|\tau_0(\!|\, !(bud_n.\sigma|\sigma_0(\!|\, P \,|\!)) \circ Q \,|\!)) \;\circ R)$

Table 2. The axioms for the reduction relation $\mapsto$ (budding).

(drip1) $drip(\rho).\sigma|\sigma_0(\!|\, P \,|\!) \circ R \mapsto nf(\rho(\!|\,|\!) \circ \sigma|\sigma_0(\!|\, P \,|\!) \circ R)$

(drip2) $(!drip(\rho).\sigma)|\sigma_0(\!|\, P \,|\!) \circ R \mapsto nf(\rho(\!|\,|\!) \circ \sigma|(!drip(\rho).\sigma)|\sigma_0(\!|\, P \,|\!) \circ R)$

(drip3) $!(drip(\rho).\sigma|\sigma_0(\!|\, P \,|\!)) \circ R \mapsto nf(\rho(\!|\,|\!) \circ \sigma|\sigma_0(\!|\, P \,|\!) \circ !(drip(\rho).\sigma|\sigma_0(\!|\, P \,|\!)) \circ R)$

(brane1) $$\frac{P \rightarrow Q}{\sigma(\!|\, P \,|\!) \circ R \mapsto \sigma(\!|\, Q \,|\!) \circ R}$$

(brane2) $$\frac{P \rightarrow Q}{!(\sigma(\!|\, P \,|\!)) \circ R \mapsto \sigma(\!|\, Q \,|\!) \circ !(\sigma(\!|\, P \,|\!)) \circ R}$$

Table 3. The axioms and rules for the reduction relation $\mapsto$.