Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 171 (2007) 11–23
www.elsevier.com/locate/entcs

Translating Mobile Ambients into P Systems

Aman Bogdan[1] and Gabriel Ciobanu[2]

Romanian Academy, Institute of Computer Science
Blvd. Carol I nr.8, 700505 Iaşi, Romania

Abstract

We present a translation of the mobile ambients without communication and replication into P systems with mobile membranes. We introduce a set of developmental rules over membranes, and describe the correspondence between the behaviour of an ambient and the evolution of its translated membrane system. We give an operational correspondence result between the mobile ambients and P systems.

Keywords: mobile ambients, membrane systems, operational correspondence.

1 Introduction

The mobile ambients and membrane systems (called also P systems) have similar structures and common concepts. Both have a hierarchical structure, and work mainly with a notion of location. Mobile ambients are suitable to represent migration of processes between certain boundaries; P systems are suitable to represent the evolution of a system composed by objects, rules and membranes. We consider these new computing models, and translate the mobile ambients into membrane systems. We present this translation by explaining carefully each step of the translation.

A successful formalism for expressing mobility is provided by ambient calculus [3], where ambients change their location by consuming certain capabilities. This formalism is well suited for expressing such issues of mobile computations as working environment, and access to information or resources [6]. Membrane systems represent a new abstract model inspired by cell compartments and molecular membranes [8]. Essentially, such a system is composed of various compartments, each compartment with a different task, and all of them working simultaneously to accomplish a more general task of the whole system. The P systems with mobile membranes [7] is a model which expresses mobility by the movement of membranes

[1] Email: baman@iit.tuiasi.ro
[2] Email: gabriel@iit.tuiasi.ro

doi:10.1016/j.entcs.2007.05.001

in such a system. The movement is given mainly by two operations: exocytosis and endocytosis. Both mobile ambients and P systems are used to model various aspects on the distributed systems. Distributed features of mobile ambients are described in [3], and distributed algorithms for membrane systems are presented in [4].

The structure of the paper is as follows. Section 2 presents the pure mobile ambients, whereas Section 3 presents the P systems with mobile membranes and local evolution rules. The core of the paper is represented by Section 4, where we present the translation of mobile ambients into P systems. We introduce a particular set of developmental rules, and simulate the behaviour of an ambient by applying the developmental rules in the corresponding P system. We establish an operational correspondence between the mobile ambients and membrane systems. Conclusion and references end the paper.

2 Mobile Ambients

We give a short description of pure mobile ambients; more information can be found in [3]. Given an infinite set of names $\mathcal{N}$ (ranged over by $m, n, \ldots$) we define the set $\mathcal{A}$ of MA-processes (denoted by $A, A', B, \ldots$) together with their capabilities (denoted by $C, C', \ldots$) as follows:

$$C \quad ::= \quad in\ n \quad | \quad out\ n \quad | \quad open\ n$$

$$A \quad ::= \quad \mathbf{0} \quad | \quad C.A \quad | \quad n[A] \quad | \quad A \mid B \quad | \quad (\nu n)A$$

The process $\mathbf{0}$ is an inactive process (it does nothing). The processes $C.A$ are called *actions*, and the processes $n[A]$ are called *ambients*. $A \mid B$ is a parallel composition of MA-processes A and B, and $(\nu n)A$ creates a new unique name n within the scope of A. The *structural congruence* $\equiv_{amb}$ on MA-processes is the least congruence satisfying the following requirements:

$(\mathcal{A}, |, \mathbf{0})$ is a commutative monoid;
$(\nu m)A \equiv_{amb} (\nu n)A\{n/m\}$ and $(\nu n)(A \mid B) \equiv_{amb} A \mid (\nu n)B$, where $n \notin fn(A)$;
if $n \neq m$ then $(\nu n)m[A] \equiv_{amb} m[(\nu n)A]$;
$(\nu n)\mathbf{0} \equiv_{amb} \mathbf{0}$, $(\nu n)(\nu m)A \equiv_{amb} (\nu m)(\nu n)A$.

The operational semantics of pure ambient calculus is defined in terms of a reduction relation $\Rightarrow_{amb}$ by the following axioms and rules.

Axioms:

$$\textit{(In)} \quad n[in\ m.A \mid A'] \mid m[B] \ \Rightarrow_{amb}\ m[n[A \mid A'] \mid B]\ ;$$

$$\textit{(Out)} \quad m[n[out\ m.A \mid A'] \mid B] \ \Rightarrow_{amb}\ n[A \mid A'] \mid m[B]\ ;$$

$$\textit{(Open)} \quad open\ n.A \mid n[B] \ \Rightarrow_{amb}\ A \mid B\ .$$

Rules:

$$\textit{(Res)} \quad \frac{A \ \Rightarrow_{amb}\ A'}{(\nu n)A \ \Rightarrow_{amb}\ (\nu n)A'}\ ; \qquad \textit{(Comp)} \quad \frac{A \ \Rightarrow_{amb}\ A'}{A \mid B \ \Rightarrow_{amb}\ A' \mid B}\ ;$$

$$\textit{(Amb)} \quad \frac{A \ \Rightarrow_{amb}\ A'}{n[A] \ \Rightarrow_{amb}\ n[A']}\ ; \qquad \textit{(Struc)} \quad \frac{A \equiv_{amb} A',\ A' \ \Rightarrow_{amb}\ B',\ B' \equiv_{amb} B}{A \ \Rightarrow_{amb}\ B}.$$

We define a set TA of top ambients, and a set TC of top capabilities:

Top ambients	Top capabilities
1. $TA(\mathbf{0}) = \emptyset$	1. $TC(\mathbf{0}) = \emptyset$
2. $TA(n[\,]) = n$	2. $TC(n[\,]) = \emptyset$
3. $TA(cap\ n.A) = \emptyset$	3. $TC(cap\ n.A) = cap\ n$
4. $TA(A \mid B) = TA(A) \cup TA(B)$	4. $TC(A \mid B) = TC(A) \cup TC(B)$
5. $TA((\nu n)A) = TA(A)$	5. $TC((\nu n)A) = TC(A)$

where $A, B \in \mathcal{A}$ and cap is in, out or $open$.

Besides the previous known notions in ambients calculus, we introduce here a notion of deadlock: an ambient A is a deadlock if there exists no B such that $A \Rightarrow_{amb} B$. The set $\mathcal{D}_{amb}$ of deadlocks in mobile ambients is defined by the following rules:

1. $$\frac{-}{\mathbf{0} \in \mathcal{D}_{amb}}$$

2. $$\frac{A \in \mathcal{A}}{cap\ n.A \in \mathcal{D}_{amb}}$$

3. $$\frac{D \in \mathcal{D}_{amb},\ TA(D) = \emptyset}{n[D] \in \mathcal{D}_{amb}}$$

4. $$\frac{\begin{array}{c} D \in \mathcal{D}_{amb};\ TA(D) \neq \emptyset;\ for\ all\ j \in TA(D) \\ open\ j | D \Rightarrow_{amb} D'_j,\ out\ n \notin TC(D'), \end{array}}{n[D] \in \mathcal{D}_{amb}}$$

5. $$\frac{A \in \mathcal{D}_{amb}}{(\nu n)A \in \mathcal{D}_{amb}}$$

6. $$\frac{\begin{array}{c} D_1, D_2 \in \mathcal{D}_{amb};\ for\ all\ k \in TA(D_i),\ for\ all\ m \in TA(D_j),\ i \neq j \\ open\ m \notin TC(D_i),\ open\ k \mid D_i \Rightarrow_{amb} D'^i_k,\ in\ m \notin TC(D'^i_k), \end{array}}{D_1 \mid D_2 \in \mathcal{D}_{amb}}$$

We explain the rules 3, 4 and 6.

3. If $D \in \mathcal{D}_{amb}$ does not contain any top ambients ($TA(D) = \emptyset$), then $D = D_1 \mid \ldots \mid D_k$ where each $D_j = cap\ j.D'_j,\ j = \overline{1,k}$. It follows that $n[D] \in \mathcal{D}_{amb}$.

4. On the other hand, if $D \in \mathcal{D}_{amb}$ contains top ambients ($TA(D) \neq \emptyset$), then $D = D_1 \mid \ldots \mid D_k$ where $D_j = [_j D'_j]_j$ or $D_j = cap\ j.D'_j,\ j = \overline{1,k}$. Then $n[D] \in \mathcal{D}_{amb}$ whenever $out\ n \notin TC(D'_j)$ for each $D_j = [_j D'_j]_j$.

6. If $D_1, D_2 \in \mathcal{D}_{amb}$, then $D_1 = D^1_1 \mid \ldots \mid D^1_t$ with $D^1_k = [_k D'^1_k]_k$ or $D^1_k = cap\ k.D'^1_k,\ k = \overline{1,t}$, and $D_2 = D^2_1 \mid \ldots \mid D^2_s$ with $D^2_k = [_k D'^2_k]_k$ or $D^2_k = cap\ k.D'^2_k,\ k = \overline{1,s}$. Then we have $D_1 \mid D_2 \in \mathcal{D}_{amb}$ if for each $D^i_k = [_k D'^i_k]_k$, D'^i_k does not contain a capability which allows the ambient k to enter into an ambient from the other D_j, and for each $D^i_k = cap\ k.D'^i_k$, $cap\ k$ does not open an ambient from the other D_j, where $i = \overline{1,2}$, $j = \overline{1,2}$ and $i \neq j$.

3 P Systems

A detailed description of the P systems can be found in [8]. A *membrane system* consists of a hierarchy of membranes which do not intersect, with a distinguishable membrane called *skin* surrounding all of them. The space outside the skin membrane is called the environment. A membrane contains multisets of *objects*, *evolution*

rules, and possibly other membranes. The multisets of objects from a membrane correspond to the "chemicals swimming in the solution in the cell compartment", while the rules correspond to the "chemical reactions possible in the same compartment". The rules must contain target indications, specifying the membrane where the new objects obtained after applying the rule are sent. The new objects either remain in the same membrane whenever they have a *here* target, or they pass through membranes in two directions: they can be sent *out* of the membrane, or can be sent *in* one of the membranes included in the current membrane which is precisely identified by its label. In one step, the objects can pass only through one membrane.

There are many variants and classes of P systems; many of them are introduced in [8]. We give here a short description of the P systems with mobile membranes having local evolution rules [7]. We use these particular P systems to build our translation.

Definition 3.1 A *P system with mobile membranes having local evolution rules* is a construct $\prod = (V, H, \mu, w_1, \ldots, w_n, R)$, where:

(i) $n \geq 1$ (the initial *degree* of the system);

(ii) V is an alphabet (its elements are called *objects*);

(iii) H is a finite set of *labels* for membranes;

(iv) μ is a membrane structure, consisting of n membranes, labelled (not necessarily in a one-to-one manner) with elements of H;

(v) $w_1, w_2, \ldots, w_n$ are strings over V , describing the *multisets of objects* placed in the n regions of μ;

(vi) R is a finite set of *developmental rules* of the following forms:

(a) $[_h[_m a \to v]_m]_h$ for $h, m \in H, a \in V, v \in V^*$; local evolution rules

These rules are called *local* because the evolution of an object a of m is possible only when m is inside h; if this restriction is not imposed, that is, the evolution of a in m is allowed irrespective of where m is placed, then we say that we have a global evolution rule, and write it simply as $[_m a \to v]_m$.

(b) $[_h a]_h[_m]_m \to [_m[_h b]_h]_m$ for $h, m \in H, a, b \in V$; endocytosis

An elementary membrane labelled h enters the adjacent membrane labelled m, under the control of object a. The labels h and m remain unchanged during this process; however the object a may be modified to b during the operation. Membrane m is not necessarily elementary.

(c) $[_m[_h a]_h]_m \to [_h b]_h[_m]_m$ for $h, m \in H, a, b \in V$; exocytosis

An elementary membrane labelled h is sent out of a membrane labelled m under the control of object a. The labels of the two membranes remain unchanged; the object a of membrane h may be modified to b during this operation. Membrane m is not necessarily elementary.

(d) $[_h a]_h \to [_h b]_h[_h c]_h$, $h \in H, a, b, c \in V$; elementary division rules

In reaction with an object a, the membrane labelled h is divided into two membranes labelled h, with the object a replaced in the two new

membranes by possibly new objects b and c.

We do not use rules of type (d) in the translation presented in this paper.

4 Relating Mobile Ambients to P Systems

In this section we describe a relationship between pure mobile ambients (without communication and replication) and P systems. This relationship is mainly provided by a translation of the ambients into P systems. Other encoding of the mobile ambients into new computational models is also presented in [5]. In order to translate the pure mobile ambient into a specific class of P systems we use the following translation steps:

* every ambient n is translated into a membrane having the same label n;
* every capability *cap n* from an ambient is translated both into an object "*cap n*" in the corresponding membrane, and into a membrane labelled "*cap n*", both placed in the same membrane; every path of capabilities from an ambient is translated into a nested structure of membranes; (for example *in m.out n* is translated into $in\ m[_{in\ m} out\ n[_{out\ n}]_{out\ n}]_{in\ m}$);
* an object *dlock* is placed near the membrane structure after all the translation is done. The additional object *dlock* is used to simulate the deadlocks from mobile ambients in membrane systems. It prevents the consumption of capability objects in a membrane system which corresponds to a deadlock structure in mobile ambients.

A feature of pure mobile ambients is that they have a spatial tree-like structure. The nodes in this structure are represented by ambients and capabilities. When translating a pure mobile ambient into P systems, we obtain the same tree structure of the membrane system, where every node is a membrane corresponding to an ambient or a capability. When translate the ambient $n[in\ m.0 \mid t[]] \mid m[]$ into a P system, we obtain $dlock\ [_n in\ m\ [_{in\ m}]_{in\ m}[_t]_t]_n[_m]_m$.

Whenever we translate a path of capabilities, we should preserve the order in which they are consumed. This order is preserved by the translation described above, even it requires a lot of resources. Another solution is to translate every capability only into an object, and to preserve the order of the objects by adding extra objects into the system. This should be done by introducing objects able to enchain a certain sequence of rules: for instance, if we have *in n.in m* ..., then in the corresponding P system we get the rules:

$$in\ n \to in\ n\ x, \quad in\ n\ x \to in\ m\ y, \quad \ldots$$

Cardelli and Gordon use in [3] the following structure

$$p[succ[open\ op]] \mid open\ q.open\ p.P \mid op[in\ succ.in\ p.in\ succ.$$
$$(q[out\ succ.out\ succ.out\ p] \mid open\ op)]$$

Starting from such a structure, we can understand why we use the translation steps given above. For every consumed capability in mobile ambients, we simulate the change of the ambient structure by the help of some special rules in P systems. Following [7], we introduce some developmental rules. In these rules, by star objects we mean cap^* and cap_*.

(a) $[_m[_n out\ m\ dlock\ single]_n]_m \rightarrow [_m[_n out^*m\ out_*m\ dlock]_n]_m$ – whenever we have a configuration which permits us to extract membrane n from membrane m, then the object $out\ m$ from membrane n is destroyed, and the objects out^*m, out_*m replace it; the object out^*m is created to announce membrane n that it can exit membrane m whenever it is an elementary membrane, and the object out_*m is created to announce that membrane $out\ m$ can be dissolved;

(b) $out_*m[_{out\ m}]_{out\ m} \rightarrow [_{out\ m}\delta]_{out\ m}$ – once the object $out\ m$ is consumed, this rule allows us to dissolve the corresponding membrane;

(c) $[_m[_n out^*m]_n]_m \rightarrow [_n]_n[_m]_m$ – if membrane n is elementary, and it contains an object out^*m created by the previous rule, then the membrane labelled by n exits the membrane labelled by m, and the object out^*m is consumed;

(d) $[_n in\ m\ dlock\ single]_n[_m]_m \rightarrow [_n in^*m\ in_*m\ dlock]_n[_m]_m$– whenever we have a configuration which permits us to introduce membrane n into membrane m, then an object $in\ m$ from membrane n is destroyed, and the objects in^*m, in_*m replace it; the object in^*m is created to announce membrane n that it can enter membrane m whenever it is an elementary membrane, and the object in_*m is created to announce that the membrane $in\ m$ can be dissolved;

(e) $in_*m[_{in\ m}]_{in\ m} \rightarrow [_{in\ m}\delta]_{in\ m}$ – once the object $in\ m$ is consumed, this rule allows to dissolve the corresponding membrane;

(f) $[_n in^*m]_n[_m]_m \rightarrow [_m[_n]_n]_m\ |_{[_m \neg cap^*\ \neg cap_*]_m}$ – if the membrane n is elementary and contains an object in^*m created by the previous rule, and the membrane m does not contain any star objects (this is denoted in membrane systems by $|_{[_m \neg cap^*\ \neg cap_*]_m}$), then the membrane labelled n enters the membrane labelled m, and the object in^*m is consumed;

(g) $[_m]_m open\ m\ dlock\ single \rightarrow [_m\delta]_m open_*m\ dlock$ – in the presence of an object $open\ m$, the membrane m is dissolved, and the object $open_*m$ is created to announce that membrane $open\ m$ can be dissolved;

(h) $open_*m[_{open\ m}]_{open\ m} \rightarrow [_{open\ m}\delta]_{open\ m}$ – once the object $open\ m$ is consumed, this rule allows us to dissolve the corresponding membrane;

(i) $[_n in^*m\ U^*\ [_t]_t]_n \rightarrow [_n in^*m\ U^*\ [_t in^*m\ out^*n\ in^*n\ U^*]_t]_n$, where by U* we denote the set of star objects placed in the membrane n, except the object in^*m – if membrane n contains other nested membranes and an object in^*m, then it begins the process of recursively extracting all the nested membranes in order to become elementary; after this process is finished, it is allowed to enter into membrane m; the created objects from the right part of the rule are used to obtain the initial configuration of nested membranes in n. The new created objects help the membranes to exit n, then to enter m, and finally to enter again n. This rule makes a copy of all-star objects of a membrane (namely U^* and in^*m), and sends them in the nested membrane; it also makes two new objects which have the signature of the membrane which creates them (for example the membrane n creates the objects out^*n, in^*n, and then sends them to all nested membranes);

(j) $[_n out^*m\ U^*\ [_t]_t]_n \rightarrow [_n out^*m\ U^*\ [_t out^*m\ out^*n\ in^*n\ U^*]_t]_n$, where by U* we

denote the set of star objects placed in the membrane n, except the object out^*m – if membrane n contains other nested membranes and an object out^*m, then it begins the process of recursively extracting all the nested membrane in order to become elementary; after this it is allowed to exit membrane m, and then rebuild the initial configuration of the nested membranes (in all the nested membranes in n are created objects which help the membranes to exit n, then to exit m, and finally to enter again in n to restore the initial configuration). This rule works similarly to the previous one;

(k) $dlock[_n]_n \rightarrow dlock[_n dlock]_n \; |_{\neg n[_n \neg dlock]_n}$ – if the object $dlock$ is outside a membrane n which does not already contain an object $dlock$, and there is no object n in the same membrane with the object $dlock$, then a new object $dlock$ is placed inside the membrane n; the rule specifies the fact that the object $dlock$ can only pass through membranes corresponding to translated ambients, this making impossible the consumption of capability objects from the translated deadlock ambients;

(l) $[_n dlock]_n \rightarrow [_n]_n$ – if the objects $dlock$ is placed inside a membrane, then it is removed; in this way the object $dlock$ outside the membrane system is preserved;

(m) $[_n dlock]_n \rightarrow [_n dlock\ single]_n$ – an object $single$ is created in an arbitrary membrane containing an object $dlock$; the object $single$ is used to ensure that only rule of type a), d), g) is applied at a given moment;

(n) $single \rightarrow [\delta]$ – an object $single$ is consumed; the last two rules assure that at most one object $single$ is in the membrane system at any moment.

Whenever we get the membrane structure

$$dlock\ [_n in\ m[_{in\ m}]_{in\ m}[_t out\ n[_{out\ n}]_{out\ n}]_t]_n[_m]_m$$

after applying in a maximally parallel manner the rules defined above, we could obtain either the configuration $dlock\ [_m[_n]_n[_t]_t]_m$ or $dlock\ [_t]_t[_m[_n]_n]_m$. The order in which the non-star objects ($cap\ n$) are consumed in the translated membrane should be the same as the order in which the capabilities are consumed in the translated ambient. However in the example above this order cannot be established; the non-star objects can be consumed by two rules applied in parallel. Related to this remark, we have the following priorities:

$$\text{b), e), h)} > \text{c), f), i), j)} > \text{a), d), g), k)} > \text{n)} > \text{l), m)}$$

According to these priorities, the membrane structure

$$dlock\ [_n in\ m[_{in\ m}]_{in\ m}[_t out\ n[_{out\ n}]_{out\ n}]_t]_n[_m]_m$$

evolves to the configuration $dlock\ [_m[_n]_n[_t]_t]_m$ whenever the first object consumed is $in\ m$. The applied rules are:

* $r_1 : dlock\ [_n]_n \rightarrow dlock\ [_n dlock]_n$ – a copy of the object $dlock$ is created inside the membrane n, which does not contain a $dlock$ object;

* $r_2 : dlock\ [_m]_m \rightarrow dlock\ [_m dlock]_m$ – a copy of the object $dlock$ is created inside the membrane m, which does not contain a $dlock$ object;

* $r_3 : dlock\ [_t]_t \rightarrow dlock\ [_t dlock]_t$ – a copy of the object $dlock$ is created inside the membrane t, which does not contain a $dlock$ object;

* $r_4 : [_n dlock]_n \to [_n dlock\ single]_n$ – an object *single* is created in the membrane n which contains the object *dlock*;

* $r_5 : [_n in\ m\ dlock\ single]_n[_m]_m \to [_n in^*m\ in_*m\ dlock]_n[_m]_m$ – membrane n can enter in membrane m under the control of the objects *in* m, *dlock* and *single*, and so the objects *in* m and *single* are consumed and other two objects (in_*m and in^*m) are created in order to control the process: in_*m dissolves the corresponding membrane labelled *in* m, and in^*m allows membrane n to enter in membrane m (if n is an elementary one);

* $r_6 : in_*m[_{in\ m}]_{in\ m} \to [_{in\ m}\delta]_{in\ m}$ – in the presence of an object in_*m, the membrane labelled *in* m is dissolved; the object in_*m signals the fact that the object *in* m has been consumed;

* $r_7 : [_n in^*m\ [_t]_t]_n \to [_n in^*m\ [_t in^*m\ out^*n\ in^*n]_t]_n$ – in the presence of star objects in membrane n (which is not an elementary one), all the star objects from this membrane are copied in the nested membrane, and another two objects out^*n, in^*n are created and sent in the nested membrane; these objects allow the nested membrane t to be extracted and then reintroduced in n; after n enters m with the help of this objects, n restores the initial nested structure;

* $r_8 : [_t in^*m\ out^*n\ in^*n\ [_{out\ n}]_{out\ n}]_t \to [_t in^*m\ out^*n\ in^*n\ [_{out\ n} in^*m\ out^*n\ in^*n\ in^*t\ out^*t]_{out\ n}]_t$ – in the presence of star objects in membrane t (which is not elementary), all the star objects from this membrane are copied in the nested membrane, and another two objects out^*t, in^*t are created and sent in the nested one, allowing the nested membrane *out* n to be extracted and then reintroduced in t;

* $r_9 : [_t[_{out\ n} out^*t]_{out\ n}]_t \to [_t]_t[_{out\ n}]_{out\ n}$ – the membrane *out* n, being elementary and containing the object out^*t, is extracted from the membrane t;

* $r_{10} : [_n[_{out\ n} out^*n]_{out\ n}]_n \to [_n]_n[_{out\ n}]_{out\ n}$ – the membrane *out* n, being elementary and containing the object out^*n, is extracted from membrane n;

* $r_{11} : [_n[_t out^*n]_t]_n \to [_n]_n[_t]_t$ – the membrane t, being elementary and containing the object out^*n, is extracted from the membrane n;

* $r_{12} : [_n in^*m]_n[_m]_m \to [_m[_n]_n]_m$ – the membrane n, being elementary and containing the object in^*m, is introduced in the membrane m which does not contain any star objects;

* $r_{13} : [_t in^*m]_t[_m]_m \to [_m[_t]_t]_m$ – the membrane t, being elementary and containing the object in^*m, is introduced in the membrane m which does not contain any star objects;

* $r_{14} : [_{out\ n} in^*m]_{out\ n}[_m]_m \to [_m[_{out\ n}]_{out\ n}]_m$ – the membrane *out* n, being elementary and containing the object in^*m, is introduced in the membrane m which does not contain any star objects;

* $r_{15} : [_t in^*n]_t[_n]_n \to [_n[_t]_t]_n$ – the membrane t, being elementary and containing the object in^*n, is introduced in the membrane n which does not contain any star objects;

* $r_{16} : [_{out\ n} in^*n]_{out\ n}[_n]_n \to [_n[_{out\ n}]_{out\ n}]_n$ – the membrane *out* n, being elementary and containing the object in^*n, is introduced in the membrane n which does not contain any star objects;

* $r_{17} : [_{out\ n} in^*t]_{out\ n}[_t]_t \to [_t[_{out\ n}]_{out\ n}]_t$ – the membrane *out* n, being elementary

and containing the object in^*t, is introduced in the membrane t which does not contain any star objects;

* $r_{18} : [_t dlock]_t \to [_t dlock\ single]_t$ – an object *single* is created in the membrane t which contains the object *dlock*;

* $r_{19} : [_n[_t out\ n\ dlock\ single]_t]_n \to [_n[_t out^*n\ out_*n\ dlock]_t]_n$ – membrane t can be extracted from membrane n under the control of an object *out* n; this object is consumed and other two objects (out_*n and out^*n) are created in order to control the remaining actions: out_*n dissolves the corresponding membrane labelled *out* n, and out^*n allows membrane t to be extracted from membrane n (if t is an elementary one);

* $r_{20} : out_*n[_{out\ n}]_{out\ n} \to [_{out\ n}\delta]_{out\ n}$ – in the presence of an object out_*n, the membrane labelled *out* n is dissolved; out_*n signals the fact that the object *out* n is consumed;

* $r_{21} : [_n[_t out^*n]_t]_n \to [_t]_t[_n]_n$ – the membrane t, being elementary and containing an object out^*n, is extracted from the membrane n.

As a final step, all the possible rules of type l) are applied in order to eliminate all the internal *dlock* objects. The rules are applied in the order we have presented them, with the additional remark that the rules of the tuples (r_1, r_2), (r_{10}, r_{11}), (r_{12}, r_{13}, r_{14}), (r_{15}, r_{16}) can be applied in any order. The computation stops when all the star objects from the membranes are consumed, and after introducing all the objects *dlock* by applying rules of the form k), none of the rules of the form a), d) or g) can be applied.

It is worth to note that the membrane structure

$$dlock\ [_n in\ m[_{in\ m}]_{in\ m}[_t out\ n[_{out\ n}]_{out\ n}]_t]_n[_m]_m$$

evolves to the configuration $dlock\ [_t]_t[_m[_n]_n]_m$ whenever the first step consumes *out* n.

We denote by $A, B, \ldots$ the mobile ambients, and by $M, N \ldots$ the P systems. We denote by $M \xrightarrow{r} N$ the fact that by applying a developmental rule r to a membrane system M we get a new membrane system N. Considering a membrane system M and the rules $r_1, \ldots, r_i$ such that $M \xrightarrow{r_1} \ldots \xrightarrow{r_i} N$, we say that such a computation is successful if N does not contain any star objects and contains only one object *dlock*.

We denote with 0 the object associated with the ambient **0**; 0 represents an object which does not appear in any reaction. To simplify the membrane systems, we use the following abbreviation: $[_n 0]_n = [_n]_n$.

Definition 4.1 The set $\mathcal{M}$ of membrane configurations M is defined by

$$M ::= 0 \mid O_1, O_2 \mid M_1, M_2 \mid O, M \mid [_n M]_n \mid (\nu n)M$$

where by O we denote a finite multiset of objects.

The restriction operator $(\nu n)M$ creates a fresh membrane around the configuration M. As in [3], the restriction operator can float outward to extend the scope of the membrane, and can float inward to restrict the scope of the membrane. The restriction construction is transparent with respect to reduction; this is expressed

by the following rule: if $M \xrightarrow{r} N$ then $(\nu n)M \xrightarrow{r} (\nu n)N$.

We can write O_1, O_2 or O, M omitting the surrounding membrane because all this structures are at least inside the skin membrane.

Definition 4.2 The structural congruence $\equiv_{mem}$ over $\mathcal{M}$ is the smallest congruence relation satisfying

$M, 0 \equiv_{mem} M, \quad M, N \equiv_{mem} N, M \quad M, (N, M') \equiv_{mem} (M, N), M'$

$(\nu m)M \equiv_{mem} (\nu n)M\{n/m\}$ and $(\nu n)(N, M) \equiv_{mem} M, (\nu n)N$

where n is not a membrane in M;

if $n \neq m$, then $(\nu n)[_m M]_m \equiv_{mem} [_m(\nu n)M]_m$;

$(\nu n)0 \equiv_{mem} 0$, $(\nu n)(\nu m)M \equiv_{mem} (\nu m)(\nu n)M$.

We deal with multisets of objects, and multisets of sibling membranes. For example, we have $[_n]_n[_m]_m \equiv_{mem} [_m]_m[_n]_n$, $in\ m[_n]_n \equiv_{mem} [_n]_n in\ m$ and $in^*n\ out^*m \equiv_{mem} out^*m\ in^*n$. The structural congruence has the following properties:

$M \equiv_{mem} M, \quad M \equiv_{mem} N$ implies $N \equiv_{mem} M$,

$M \equiv_{mem} N$ and $N \equiv_{mem} M'$ implies $M \equiv_{mem} M'$,

$M \equiv_{mem} N$ implies $M, M' \equiv_{mem} N, M'$,

$M \equiv_{mem} N$ implies $M', M \equiv_{mem} M', N$,

$M \equiv_{mem} N$ implies $M'[M] \equiv_{mem} M'[N]$.

The *operational semantics* of P systems is defined in terms of a relation $\xrightarrow{r}$ by the following rules:

$$\textit{(Res)} \quad \frac{M \xrightarrow{r} M'}{(\nu n)M \xrightarrow{r} (\nu n)M'}; \quad \textit{(Comp)} \quad \frac{M \xrightarrow{r} M'}{M, N \xrightarrow{r} M', N};$$

$$\textit{(Amb)} \quad \frac{M \xrightarrow{r} M'}{[_n M]_n \xrightarrow{r} [_n M']_n}; \quad \textit{(Struc)} \quad \frac{M \equiv_{mem} M',\ M' \xrightarrow{r} N',\ N' \equiv_{mem} N}{M \xrightarrow{r} N}.$$

Definition 4.3 We define a translation function $\mathcal{T} : \mathcal{A} \to \mathcal{M}$, $\mathcal{T}(A) = dlock\ \mathcal{T}_1(A)$ where:

$$\mathcal{T}_1(A) = \begin{cases} 0 & \text{if } A = \mathbf{0} \\ cap\ n[_{cap\ n}\mathcal{T}_1(A_1)]_{cap\ n} & \text{if } A = cap\ n.A_1 \\ [_n\mathcal{T}_1(A_1)]_n & \text{if } A = n[A_1] \\ (\nu n)\mathcal{T}_1(A_1) & \text{if } A = (\nu n)A_1 \\ \mathcal{T}_1(A_1) \mid \mathcal{T}_1(A_2) & \text{if } A = A_1 \mid A_2 \end{cases}$$

Proposition 4.4 *If A and B are two mobile ambients and M is a membrane system such that $A \Rightarrow_a B$ and $M = \mathcal{T}(A)$, then there exists a chain of transitions $M \xrightarrow{r_1} \ldots \xrightarrow{r_i} N$ such that $r_1, \ldots, r_i$ are developmental rules, and $N = \mathcal{T}(B)$.*

Proof. *(Sketch)* Since $A \Rightarrow_{amb} B$, then one of the requirements In, Out or $Open$ is fulfilled for subambients A' and B' which are included in A and B, respectively. We consider the case when $A' = n[in\ m] \mid m[\,]$ and $B' = m[n[\,]]$. Then according to the definition of the translation, M contains the submembrane structure $[_n in\ m[_{in\ m}]_{in\ m}]_n[_m]_m$, which after applying a number of rules of the form k) we obtain the following structure $[_n dlock\ in\ m[_{in\ m}]_{in\ m}]_n[_m dlock]_m$. Using the rules

$r_1 : [_n dlock]_n \to [_n dlock\ single]_n$

$r_2 : [_n in\ m\ dlock\ single]_n[_m]_m \to [_n in^*m\ in_*m\ dlock]_n[_m]_m$

$r_3 : in_* m[_{in\ m}]_{in\ m} \to [_{in\ m}\delta]_{in\ m}$

$r_4 : [_n in^* m]_n[_m]_m \to [_m[_n]_n]_m$,

and some rules of the form l) there exists the following sequence of transitions $M \overset{k)}{\longrightarrow}{}^* M_1' \xrightarrow{r_1} M_1 \xrightarrow{r_2} M_2 \xrightarrow{r_3} M_3 \xrightarrow{r_4} M_3' \overset{l)}{\longrightarrow}{}^* N$, where M_1', M_1, M_2', M_2 and M_3 are intermediate membrane configurations, and the membrane structure N contains the submembrane structure $[_m[_n]_n]_m$. Once the objects *dlock* and *single* are created near the object $in\, m$, these transitions are the only deterministic steps which can be performed. We can notice that $[_m[_n]_n]_m = \mathcal{T}_1(B')$. Hence, according to the definition of translation function $\mathcal{T}$ and transition relation $\xrightarrow{r}$, we reach the conclusion that the membrane structure M admits the required sequence of transitions leading to the membrane structure N which does not contain star objects, and $N = \mathcal{T}(B)$.

The case of a nested ambient where $A' = n[in\ m\ t[\,]] \mid m[\,]$ and $B' = m[n[t[\,]]]$ has already been presented. The other cases (when A' and B' satisfy the requirements *Out*, or *Open*) can be treated analogously. □

Proposition 4.5 *Let M and N be two membrane systems with no star objects and only one dlock object, and a mobile ambient A such that $M = \mathcal{T}(A)$. If there is a sequence of transitions $M \xrightarrow{r_1} \ldots \xrightarrow{r_i} N$, then there exists a mobile ambient B such that $N = \mathcal{T}(B)$. If only one non-star object is consumed, then we have $A \Rightarrow_{amb} B$.*

Proof. *(Sketch)* We proceed by structural induction. Since M does not contain any star object, the first rule which consumes a translated capability has one of the following forms:

$[_n in\ m\ dlock\ single]_n[_m]_m \to [_n in^* m\ in_* m\ dlock]_n[_m]_m$,

$[_m[_n out\ m\ dlock\ single]_n]_m \to [_m[_n out^* m\ out_* m\ dlock]_n]_m$,

$[_m]_m open\ m\ dlock\ single \to [_m\delta]_m open_* m\ dlock$.

We present only the first case, the others being treated similarly. If we consider that the first rule is applicable in M, then M contains the membrane structure $[_n in\ m[_{in\ m}]_{in\ m}]_n[_m]_m$. According to the definition of $\mathcal{T}$, M can be written as M_1, M' or $M_1[M']$, where $M' = [_n in\ m[_{in\ m}]_{in\ m}]_n[_m]_m$. We study only the first case, the other one being treated similarly. If A is a mobile ambient encoded by M, then according to the definition of $\mathcal{T}$ it contains two subambients $A' = n[in\ m] \mid m[\,]$ and A_1 such that $A = A_1 \mid A'$, $M' = \mathcal{T}_1(A')$, and $M_1 = \mathcal{T}_1(A_1)$. The application of the rule $[_n in\ m\ dlock\ single]_n[_m]_m \to [_n in^* m\ in_* m\ dlock]_n[_m]_m$ to the membrane system M changes only the membrane system M'. In M' we have two new objects, $in^* m$ and $in_* m$, which signal that membrane n can enter into the membrane m. These objects are used by some other rules, and control the moving of membrane n into the membrane m. After the application of these rules, M' is evolving to $N' = [_m[_n]_n]_m$. The induction hypothesis expresses that N' encodes an ambient B'. After obtaining N', N has the structure $N = M_1, N'$ and it encodes the mobile ambient $B = A_1 \mid B'$. The transition from M' to N' represents also the transition from M to N.

It should be notice that by consuming the capability $in\ m$ we have $A' \Rightarrow_{amb} B'$.

So the transition from M to N with the consumption of only one non-star object is simulated by the transition of A to B. □

Remark 4.6 If $M \xrightarrow{r_1} \ldots \xrightarrow{r_i} N$, and both M and N do not contain star objects and contain only one *dlock* object, then the number of steps between mobile ambients A and B is the number of non-star objects consumed during the computation in the membrane evolution. The order in which the rules are applied in ambients is the order in which the non-star objects are consumed in the P systems M and N.

If we have two membrane systems M and N with no star objects and only an object *dlock*, we say that $M \Rightarrow_{mem} N$ if there is a sequence of rules $r_1, \ldots, r_i$ such that $M \xrightarrow{r_1} \ldots \xrightarrow{r_i} N$, where only one rule consumes only one non-star object. Considering together the previous two propositions, we have an operational correspondence result.

Theorem 4.7 ***(operational correspondence)***

(i) *If $A \Rightarrow_{amb} B$, then $\mathcal{T}(A) \Rightarrow_{mem} \mathcal{T}(B)$.*

(ii) *If $\mathcal{T}(A) \Rightarrow_{mem} M$, then exists B such that $A \Rightarrow_{amb} B$ and $M = \mathcal{T}(B)$.*

5 Conclusion

In this paper we translate the mobile ambients into P systems, giving also an operational correspondence between these two formalisms. The translation of the mobile ambients into P systems can be presented as a relationship between $\Rightarrow_{amb}$ and $\Rightarrow_{mem}$. An example of using the translation is given in [1] by considering the cab protocol in ambient calculus presented in [6].

Previous attempts to relate one formalism to another had been published before in [9] and [2]. In [9] the authors show a simple way of encoding a P system into a mobile ambient having the same structure. In order to do this, the authors introduced a new primitive in the ambient calculus (namely the prioritized choice operator), importing an idea from P systems. The authors also provides ideas how other features from P systems can be simulated in mobile ambients:

i) P systems with cell division can be easily simulated by creating new ambients having the same name,

ii) P systems where both the objects and the membranes have no names but they are provided with electrical charge can be modelled by an ambient for which all the subambients are named either *pos* or *neg*, depending on the electrical charge.

In [2] the authors discuss about a relationship between the mobile ambients and P systems. They model the Ethernet network technology by using P systems, starting from the idea that ambient calculus are used for describing the mobile computation over the World Wide Web. The authors do not actually translate the mobile ambients into P systems. However they identify some problems which appear during such a translation:

i) the difficulty of expressing mobility of the ambients in P systems because there

is no explicit mobility of membranes in P systems; therefore mobility must be expressed in some indirect way: destroy the membrane, representing a moving ambient with its content at its initial location, and then create a new membrane with the same content at the destination location;

ii) the difficulty of having the right order of execution for primitive operations and capabilities of the ambient calculus.

In order to express the mobility of membranes, we introduce in P systems new developmental rules inspired by biology (endocytosis, exocytosis). Similar rules are presented in brane calculi.

References

[1] B. Aman, G. Ciobanu. On the Relationship Between P Systems and Mobile Ambients FML-06-02 Technical Report, ISSN 1842 - 1490. http://iit.iit.tuiasi.ro/TR/, 2006.

[2] E. Boian, V. Rogozhin. Simulation of Mobile Ambients by P Systems. *Proc WMC 2003*, Lecture Notes in Computer Science vol.2933, Springer, 304-319, 2004.

[3] L. Cardelli, A. Gordon. Mobile Ambients, *Proc. Foundations of Software Science and Computation Structures*, Lecture Notes in Computer Science vol.1378, Springer, 140-155, 1998.

[4] G. Ciobanu. Distributed Algorithms over Communicating Membrane Systems. *Biosystems* vol.70, Elsevier, 123-133, 2003.

[5] G. Ciobanu, V.A. Zakharov Encoding Mobile Ambients into the π-calculus. *Proc. Perspectives of System Informatics*, Lecture Notes in Computer Science, Springer, 2006.

[6] D. Hirschkoff, D. Teller, P. Zimmer. Using Ambients to Control Resources *Proc. CONCUR'02*, Lecture Notes in Computer Science vol.2421, Springer, 288-303, 2002.

[7] S.N. Krishna. On the Efficiency of a Variant of P Systems with Mobile Membranes *Cellular Computing; complexity aspects*, ESF PESC Exploratory Workshop, Fenix Editora, Sevilla, 237-246, 2005.

[8] Gh. Păun. *Membrane Computing. An Introduction.* Springer, 2002.

[9] I. Petre, L. Petre. Mobile Ambients and P Systems, *TUCS Technical Report* 293, 2001.