Fundamenta Informaticae 72 (2006) 21–35
IOS Press

A Calculus of Looping Sequences for Modelling Microbiological Systems

Roberto Barbuti, Andrea Maggiolo–Schettini, Paolo Milazzo* **and Angelo Troina**
Dipartimento di Informatica, Università di Pisa
Largo B. Pontecorvo 3, 56127 - Pisa, Italy
E-mail: {barbuti,maggiolo,milazzo,troina}@di.unipi.it

Abstract. The paper presents the Calculus of Looping Sequences (CLS) suitable to describe microbiological systems and their evolution. The terms of the calculus are constructed by basic constituent elements and operators of sequencing, looping, containment and parallel composition. The looping operator allows tying up the ends of a sequence, thus creating a circular sequence which can represent a membrane. We show that a membrane calculus recently proposed can be encoded into CLS. We use our calculus to model interactions among bacteria and bacteriophage viruses, and to reason on their properties.

1. Introduction

In the last few years a notable research effort has been devoted to formally describe biological processes by using means originally developed by computer scientists to model systems of interacting components. This permits simulation of system behaviour and verification of properties. Among the many formalisms that have been applied to biology there are Petri Nets [11], Hybrid Systems [1], and the π-calculus [14, 7]. Moreover, some new formalisms have been proposed to describe biomolecular and membrane interactions [2, 4, 5, 6, 8, 13].

In this paper we present a new calculus suitable to describe microbiological systems and their evolution. The terms of our calculus are constructed by starting from basic constituent elements and composing them by means of operators of sequencing, looping, containment and parallel composition. Looping allows tying up the ends of a sequence, thus creating a circular sequence of the constituent elements. We assume that the elements of a circular sequence can rotate, and this motivates the terminology of

*Address for correspondence: Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 3, 56127 - Pisa, Italy

looping sequence. A looping sequence can represent a membrane and the containment operator allows representing that some element is inside the membrane.

Viewing membranes as sequences of elements allows representing interactions of these elements, and thus describing real biological phenomena. This cannot be described by calculi that consider membranes as atomic objects, as in [5].

A set of congruence rules allows considering as equivalent terms that are intended to represent the same biological system. The evolution of a system is described by a set of rewrite rules to be applied to terms. We show that reachability of a term is decidable for monotonic rewrite rules.

As an application, we model interactions among bacteria and bacteriophage viruses, and bacteria sporulation. We represent bacteria and viruses as terms, and give a set of rewrite rules for describing how bacteriophages parasitize bacteria, and how bacteria produces spores and spores germinate.

Finally, we encode Cardelli's phago/exo/pino calculus (PEP Calculus, [5]) into CLS and prove the correctness of the encoding.

2. Calculus of Looping Sequences

In this section we introduce our Calculus of Looping Sequences (CLS). We assume a set $\mathcal{E}$ of elementary constituents $a, b, c, \ldots$, and a neutral term ϵ.

Definition 2.1. (Terms)
A *Term* T of CLS is given by the following grammar:

$$T \ ::= \ a \quad | \quad \epsilon \quad | \quad T \cdot T \quad | \quad \big(T\big)^L \quad | \quad T \rfloor T \quad | \quad T \,|\, T$$

where a is a generic element of $\mathcal{E}$. We denote with $\mathcal{T}$ the (infinite) set of terms.

A term T may be either an element in $\mathcal{E}$ or a concatenation of terms $T_1 \cdot T_2$ (a sequence), or a looping $(T)^L$ or a combination of terms by means of the containment operator $\rfloor$ and the parallel operator $|$. A term $(T)^L$ is a closed circular sequence of the elements constituting term T. Term $T_1 \rfloor T_2$ represents the containment of term T_2 in the term T_1, while term $T_1 \,|\, T_2$ represents term T_1 in parallel with term T_2. If we have the $\rfloor$ operator together with a looping, as in $(T_1)^L \rfloor T_2$, we have that the term T_2 is really inside the closed circular sequence $(T_1)^L$ represents, otherwise the $\rfloor$ operator reduces to the $|$ operator for non-looping (open) terms.

Brackets can be used to indicate the order of application of the operators in a term. We assume the $\cdot$ operator to have the highest precedence and the $\rfloor$ operator to have the precedence over the $|$ operator. Therefore $T_1 \rfloor T_2 \,|\, T$ stands for $(T_1 \rfloor T_2) \,|\, T$. Moreover, we assume $\rfloor$ to be right–associative, therefore with $T_1 \rfloor T_2 \rfloor T$ we denote the term $T_1 \rfloor (T_2 \rfloor T)$.

Example 2.1. In Figure 1 we give a visual representation of some examples of CLS terms. In Figure 1.a we represent the term $\big(a \cdot b \cdot c\big)^L$, in Figure 1.b we represent the term $\big(\big(c \cdot d \cdot e\big)^L \cdot a \cdot b\big)^L$, in Figure 1.c we represent the term $\big(\big(\big(c \cdot d \cdot e\big)^L \rfloor h\big) \cdot a \cdot b\big)^L \rfloor f \cdot g$.

Concatenation can represent a physical link between elements, while parallel composition represents juxtaposition of separated elements. When we have the parallel composition of two or more terms in a sequence, only one may actually be linked to elements of the sequence. We assume that this element

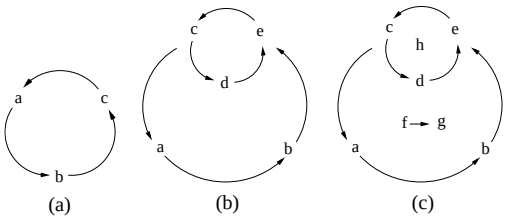

Figure 1. Examples of CLS terms

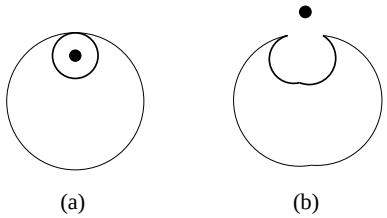

Figure 2. A real situation

is the first in the parallel composition, while the others are considered close to it. For example, the term $a \cdot (b \mid d) \cdot c$ represents a sequence $a \cdot b \cdot c$ with the element d near the sequence in proximity of b. Analogously, when we have that a term is contained in the parallel composition of two terms, the term may be contained only in one of the two, and again we assume that this is the first term. Finally, a looping of a parallel composition of terms is intended as the looping of one, the first term, in parallel with the other terms. These assumptions allow us to define a structural congruence relation on terms of the calculus as follows.

Definition 2.2. (Structural Congruence)
The structural congruence $\equiv$ is the least congruence relation on terms satisfying associativity of $\_ \mid \_$ and $\_ \cdot \_$, and the following axioms:

$$A1.\ (T_1 \mid T) \cdot T_2 \equiv (T_1 \cdot T_2) \mid T \equiv T_1 \cdot (T_2 \mid T)$$

$$A2.\ (T_1 \mid T_2) \rfloor T \equiv (T_1 \rfloor T) \mid T_2 \qquad A3.\ \big(T \mid T_1\big)^L \equiv \big(T\big)^L \mid T_1$$

$$A4.\ T \mid T_1 \mid T_2 \equiv T \mid T_2 \mid T_1 \qquad A5.\ (T_1 \rfloor T_2) \rfloor T_3 \equiv T_1 \rfloor (T_2 \mid T_3)$$

$$A6.\ \big(T_1 \cdot T_2\big)^L \equiv \big(T_2 \cdot T_1\big)^L \qquad A7.\ a \rfloor T \equiv a \mid T \qquad A8.\ (T_1 \cdot T_2) \rfloor T \equiv (T_1 \cdot T_2) \mid T$$

$$A9.\ T \mid \epsilon \equiv T \rfloor \epsilon \equiv T \qquad A10.\ T \cdot \epsilon \equiv \epsilon \cdot T \equiv T \qquad A11.\ \big(\epsilon\big)^L \equiv \epsilon$$

Axioms A1, A2 and A3 state that if we apply either sequential composition, containment or looping to a parallel composition of terms, these operators act upon the first term of the parallel composition. This means that the first element of the parallel composition plays the special role discussed before and it cannot be commuted in a series of parallel compositions. This is said by axiom A4.

We remark that assigning a special role to an element of a parallel composition is not unusual. For instance, in [9, 10] the last element in a series of parallel compositions has the special role of giving the result of the computation of the whole series. Thus, it cannot be commuted.

Axiom A5 says that putting a term inside a term which already contains a term, results in the term containing the parallel composition of the terms inside.

Axiom A6 says that terms in a looping can rotate and axioms A7 and A8 say that a term cannot stay inside a term which is not a looping term, and, in this case, the containment operator is equivalent to the parallel composition.

Finally, axioms A9, A10 and A11 describe the neutral role of ϵ and $\big(\epsilon\big)^L$ with respect to the operators of the calculus. We remark that in axiom A9 the neutral term ϵ is placed on the right hand side of the $\mid$ operator otherwise ϵ could be inserted at the left hand of a series of parallel compositions and its first term would lose its privileged role.

Remark 2.1. We have $T \rfloor (T_1 \mid T_2) \equiv T \rfloor (T_2 \mid T_1)$.

Proof:
The equivalence of the two terms can be derived as follows: $T \rfloor (T_1 \mid T_2) \overset{(A9)}{\equiv} (T \rfloor \epsilon) \rfloor (T_1 \mid T_2) \overset{(A5)}{\equiv} T \rfloor (\epsilon \mid T_1 \mid T_2) \overset{(A4)}{\equiv} T \rfloor (\epsilon \mid T_2 \mid T_1) \overset{(A5)}{\equiv} (T \rfloor \epsilon) \rfloor (T_2 \mid T_1) \overset{(A9)}{\equiv} T \rfloor (T_2 \mid T_1)$. □

The remark shows that the first element of a series of parallel compositions can be commuted when the whole series is contained inside another term. As a consequence, if one wants to have unrestricted commutativity of a parallel composition at the top level of a term, he can insert the term into the term $(\epsilon)^L$ by using the containment operator. In this way we forbid the first element of a series of parallel compositions to commute only when the whole series is an element of a sequence. Standard commutativity holds otherwise.

Consider the model of a real situation in which a membrane that is part of another membrane breaks, and its content is released in the environment. This situation is depicted in Figure 2.a. The smaller membrane (depicted with a thick line) is part of the larger one (depicted with a lighter line), namely, it is part of the sequence representing that membrane (in the picture it is shown to adhere to it). If the smaller membrane breaks and opens, we have as a result of the process one only membrane, and the content of the smaller one (the black circle) is freed. The definition of axioms A1, A2 and A3 is such that the content of the smaller membrane is freed outside the resulting membrane (as is shown in Figure 2.b). Formally, the initial situation in Figure 2.a corresponds to the term $(((b \cdot \ldots \cdot b)^L \rfloor a) \cdot c \cdot \ldots \cdot c)^L$ in which the large membrane is represented by a looping of c and the smaller one by a looping of b and the content by a. Breaking the smaller membrane means removing the looping operator of the sequence of b elements, thus obtaining the term $((b \cdot \ldots \cdot b \rfloor a) \cdot c \cdot \ldots \cdot c)^L$. By applying congruence A8, we obtain the term $((b \cdot \ldots \cdot b \mid a) \cdot c \cdot \ldots \cdot c)^L$, and then, applying congruence A1, we obtain the term $(b \cdot \ldots \cdot b \cdot c \cdot \ldots \cdot c \mid a)^L$. Finally, applying congruence A3, we obtain the term $(b \cdot \ldots \cdot b \cdot c \cdot \ldots \cdot c)^L \mid a$, which represents the final situation in Figure 2.b. Situations in which the content of the smaller membrane is released inside the bigger one should be explicitly described by using rewrite rules (defined in the following).

The situation described above is generalized by the next proposition to terms representing n membranes where the i-th membrane is element of the $(i-1)$-th membrane.

Proposition 2.1. Consider a term consisting of a nesting of looping terms:

$$T = \big(\big(\big(\big(\big(\ldots\big(\big(\big(a_1 \cdot \ldots \cdot a_n\big)^L \rfloor T_i\big) \cdot b_{i1} \cdot \ldots \cdot b_{in_i}\big)^L \ldots\big)^L \rfloor T_2\big) \cdot b_{21} \cdot \ldots \cdot b_{2n_2}\big)^L \rfloor T_1\big) \cdot b_{11} \cdot \ldots \cdot b_{1n_1}\big)^L \rfloor T_0$$

where T_j may be any term and each $\rfloor T_j$ can also be absent. The term obtained by removing from T the looping operator of the sequence at the i-th level

$$\big(\big(\big(\big(\ldots\big(\big((a_1 \cdot \ldots \cdot a_n) \rfloor T_i\big) \cdot b_{i1} \cdot \ldots \cdot b_{in_i}\big)^L \ldots\big)^L \rfloor T_2\big) \cdot b_{21} \cdot \ldots \cdot b_{2n_2}\big)^L \rfloor T_1\big) \cdot b_{11} \cdot \ldots \cdot b_{1n_1}\big)^L \rfloor T_0$$

is structurally congruent to:

$$\big(\big(\big(\big(\big(\ldots\big(a_1 \cdot \ldots \cdot a_n \cdot b_{i1} \cdot \ldots \cdot b_{in_i}\big)^L \ldots\big)^L \rfloor T_2\big) \cdot b_{21} \cdot \ldots \cdot b_{2n_2}\big)^L \rfloor T_1\big) \cdot b_{11} \cdot \ldots \cdot b_{1n_1}\big)^L \rfloor T_0\big) \mid T_i$$

Proof:
By structural induction and by applying the structural congruence $\equiv$. □

Now, we define rewrite rules, which can be used to describe the evolution of terms, and we give a transition relation as a semantics for rule applications.

We assume a set V of term variables $X, Y, Z, \ldots$. An *instantiation* is a partial function $\sigma : V \to \mathcal{T}$. With $\mathcal{T}_V$ we denote the set of CLS terms which may also contain variables in V and, given $T \in \mathcal{T}_V$, with $T\sigma$ we denote the term obtained by replacing each occurrence of each variable $X \in V$ appearing in T with the corresponding term $\sigma(X)$. With Σ we denote the set of all the possible instantiations. Finally, given $T \in \mathcal{T}_V$, with $Var(T)$ we denote the set of variables in T and with $Var_M(T)$ we denote the multiset of variables in T. For example, if $T = a \cdot X \mid (Y)^L \rfloor X$, we have that $Var(T) = \{X, Y\}$ and $Var_M(T) = \{X, X, Y\}$.

Given a term $T \in \mathcal{T}_V$, we denote with $size(T)$ the number of constituent elements syntactically occurring in T. For example, if $T = (a \cdot b)^L \mid c$ we have $size(T) = 3$, and if $T = a \cdot a \mid X$ we have $size(T) = 2$. Moreover, we define a function $occ : \mathcal{T} \times \mathcal{T} \to \mathbf{N}$ such that $occ(T', T)$ returns the number of the terms T' syntactically occurring in the term T.

Definition 2.3. (Rewrite Rules)
A rewrite rule is a triple (T, T', Σ') such that $T, T' \in \mathcal{T}_V$, $Var(T') \subseteq Var(T)$, $\Sigma' \subseteq \Sigma$ and, for all $\sigma \in \Sigma'$, $Var(T) \subseteq Dom(\sigma)$. We denote with $\Re$ the infinite set of all the possible rewrite rules.

A rewrite rule (T, T', Σ') states that a ground term $T\sigma$, obtained by instantiating variables in T by an instantiation function $\sigma \in \Sigma'$, can be transformed into the ground term $T'\sigma$ (note that we assume $Var(T') \subseteq Var(T)$). A rule can be applied to all the terms which can be obtained by instantiating the variables in T with any of the instantiations in Σ'. For instance, if $\Sigma' = \{\sigma \in \Sigma | occ(a, \sigma(X)) = 0\}$, then a rule $(b \cdot X \cdot b, c \cdot X \cdot c, \Sigma')$ can be applied to $b \cdot c \cdot b$ (obtaining $c \cdot c \cdot c$) and to $b \cdot c \cdot c \cdot b$ (obtaining $c \cdot c \cdot c \cdot c$), but not to $b \cdot a \cdot b$.

In what follows, we shall often write a rewrite rule as $T \longrightarrow T' \ [\mathcal{C}]$ instead of $(T, T', \Sigma' = \{\sigma \in \Sigma \mid \mathcal{C}\sigma\})$, where $\mathcal{C}$ is a condition, and we shall omit Σ' when $\Sigma' = \Sigma$ and write $T \longrightarrow T'$. For instance, with $b \cdot X \cdot b \longrightarrow c \cdot X \cdot c \ \ [occ(a, X) = 0]$ we denote $(b \cdot X \cdot b, c \cdot X \cdot c, \Sigma' = \{\sigma \in \Sigma | occ(a, \sigma(X)) = 0\})$.

We define the transition relation between terms, based on the application of rewrite rules. We assume that the relation is closed under structural congruence and under the application of the operators.

Definition 2.4. (Reaction Semantics)
Given a set of rewrite rules $\mathcal{R} \subseteq \Re$, the *reaction semantics* of the CLS is the transition system given by the least relation $\rightarrow$ on terms closed under $\equiv$, $\_ \mid \_$, $\_ \rfloor \_$, $\_ \cdot \_$, $(\_)^L$ and satisfying the following inference rule:

$$\frac{(T, T', \Sigma') \in \mathcal{R} \qquad \sigma \in \Sigma'}{T\sigma \rightarrow T'\sigma}$$

Given a set of rewrite rules $\mathcal{R}$ and two term $T, T' \in \mathcal{T}$, we say that T' is *reachable* from T (denoted $T \rightarrow^* T'$) iff there exist $T_1, \ldots, T_n \in \mathcal{T}$ s.t. $T \rightarrow T_1 \rightarrow \ldots \rightarrow T_n \rightarrow T'$.

Now we introduce a particular class of rewrite rules.

Definition 2.5. (Monotonic Rewrite Rules)
A rewrite rule $(T, T', \Sigma') \in \Re$ is *monotonic* iff $Var_M(T) = Var_M(T')$ and $size(T') \geq size(T)$.

Intuitively, a rewrite rule is monotonic if the number of constituent elements of the term we obtain applying the rule is greater than or equal the number of constituent elements of the initial term.

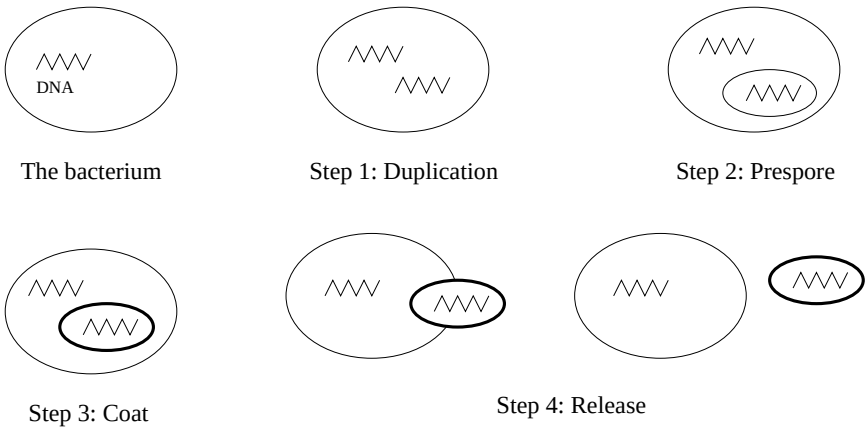

Figure 3. The Sporulation Process

Proposition 2.2. Given a finite set of monotonic rewrite rules $\mathcal{R}$ and a CLS term $T \in \mathcal{T}$, it is decidable whether a term $T' \in \mathcal{T}$ is reachable from T.

Proof:
Based on the fact that terms obtained by applying a monotonic rewrite rule or a structural congruence axiom have a size greater than or equal to the size of the source terms. □

3. An Application

In this section we show how CLS can be used to describe some aspects of the reproduction of bacteria and of bacteriophage viruses. For the sake of our study we can assume that a bacterium consists of a cellular membrane containing its DNA. In particular, as regards bacteria reproduction, we consider the sporulation mechanism, which allows producing inactive and very resistant forms, called spores. A spore can germinate and then produce a new bacterium.

Schematically, the sporulation process (shown in Fig. 3) proceeds as follows:

1. the DNA inside the bacterium is duplicated (duplication);
2. inside the bacterium a new membrane is formed containing the copy of the DNA (prespore);
3. around the prespore a second layer is formed (coat);
4. eventually, the spore passes through the bacterium membrane and becomes a free spore (release).

For the sake of clarity, before giving the rules for the process, let us introduce some denotations for

terms which occur very often:

$$PRESPORE \ ::= \ \big(\underbrace{m \cdot \ldots \cdot m}_{\frac{n}{2}}\big)^L \rfloor DNA_b$$

$$SPORE_1 \ ::= \ \big(\underbrace{c \cdot \ldots \cdot c}_{\frac{n}{2}}\big)^L \rfloor PRESPORE \qquad SPORE_2 \ ::= \ \big(\underbrace{d \cdot \ldots \cdot d}_{\frac{n}{2}}\big)^L \rfloor PRESPORE$$

Now, the rewrite rules for describing the steps of the process are the following:

$$S1. \quad \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (DNA_b \,|\, X) \ \longrightarrow \ \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (DNA_b \,|\, DNA_b \,|\, X) \qquad [occ(DNA_b, X) = 0]$$

$$S2. \quad \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (DNA_b \,|\, DNA_b \,|\, X) \ \longrightarrow \ \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (DNA_b \,|\, PRESPORE \,|\, X)$$

$$S3. \quad \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (X \,|\, PRESPORE \,|\, Y) \ \longrightarrow \ \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (X \,|\, SPORE_1 \,|\, Y)$$

$$S4. \quad \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (X \,|\, SPORE_1 \,|\, Y) \ \longrightarrow \ \big(SPORE_1 \cdot \underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (X \,|\, Y)$$

$$S5. \quad \big(SPORE_1 \cdot \underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor X \ \longrightarrow \ \big(\big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor X\big) \,|\, SPORE_2$$

$$S6. \quad SPORE_2 \ \longrightarrow \ \underbrace{d \cdot \ldots \cdot d}_{\frac{n}{2}} \,|\, \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor DNA_b$$

Rule S1 describes DNA duplication inside a bacterium (step 1 of the process). The bacterium membrane is represented by a looping of n membrane elements m (with n fixed); DNA_b represents the bacterium DNA and the term variable X represents any other element inside the bacterium membrane. The condition that DNA_b does not appear in the term X means that a sporulation process must terminate before starting a second one (no more than one copy of DNA inside the bacterium at one time).

Rule S2 models the forming of a prespore (step 2). Conventionally, we assume that the number of membrane elements of a prespore is $\frac{n}{2}$.

Rule S3 models the forming of the spore coat (step 3), where c represents the elements of the outer coat. The double layer of the spore is represented by two looping terms, one inside the other, by the term:

$$\big(\underbrace{c \ldots c}_{\frac{n}{2}}\big)^L \rfloor \big(\big(\underbrace{m \ldots m}_{\frac{n}{2}}\big)^L \rfloor DNA_b\big).$$

Rules S4 and S5 model the exiting of the spore from the bacterium (step 4). In a first phase (rule S4) the spore adheres to the bacterium membrane, becoming one element of the looping representing it. Note that the spore is represented in the rule as first element of the looping, but it can be shifted to any position by using the congruence rules. In a second phase (rule S5) the spore becomes free. In this phase, in order to distinguish a free spore from a spore inside the bacterium, the outer coat of the spore changes its elements from c to d.

A free spore may germinate by loosing its coat, which becomes an open membrane, and by growing to a normal size of n membrane elements (rule S6).

Bacteriophage viruses (or phages) exploit the enzymes of the bacteria for duplicating their DNA. In particular, they behave according to the following pattern (depicted in Figure 4):

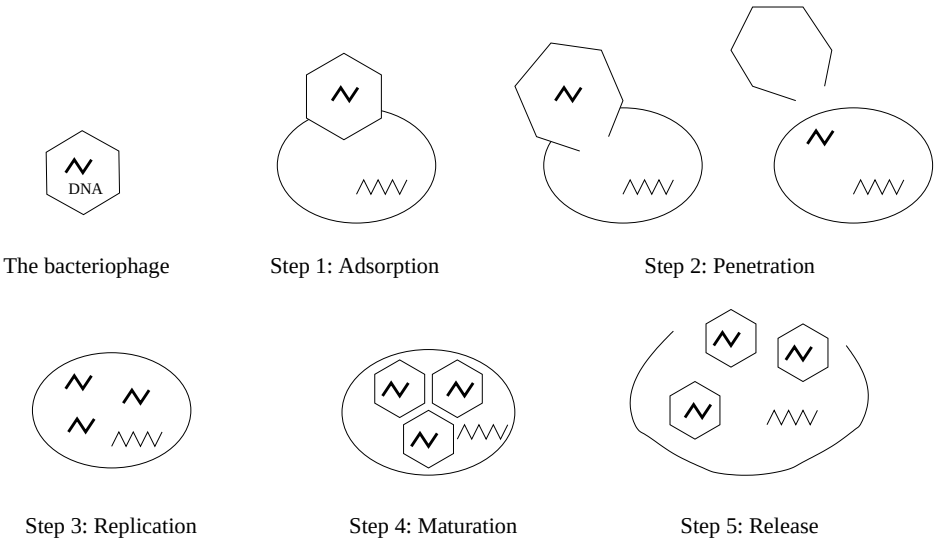

Figure 4. The Bacteriophage Replication Process

1. the phage joins with the bacterium membrane (adsorption);
2. the phage releases its DNA inside the bacterium (penetration);
3. the DNA of the phage replicates itself using bacterium enzymes (replication);
4. each copy of the phage DNA forms a new phage inside the bacterium membrane (maturation);
5. when the number of new phages inside the bacterium reaches a certain number, the membrane breaks and the new phages become free (release).

As before, we introduce a denotation for a term which occurs quite often:

$$VIRUS \ ::= \ \big(\underbrace{v \cdot \ldots \cdot v}_{k}\big)^L \rfloor DNA_v$$

The rewrite rules for describing the steps of the process are the following:

$$V1. \quad VIRUS \,|\, \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor X \ \longrightarrow \ \big(VIRUS \cdot \underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor X$$

$$V2. \quad \big(VIRUS \cdot \underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor X \ \longrightarrow \ \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (X \,|\, DNA_v) \,|\, \underbrace{v \cdot \ldots \cdot v}_{k}$$

$$V3. \quad \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (X \,|\, DNA_v) \ \longrightarrow \ \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (X \,|\, DNA_v \,|\, DNA_v) \quad [occ(DNA_v, X) < max]$$

$$V4. \quad \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (X \,|\, DNA_v) \ \longrightarrow \ \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (X \,|\, VIRUS) \qquad [occ(DNA_v, X) > max - s]$$

$$V5. \quad \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor X \ \longrightarrow \ \underbrace{m \cdot \ldots \cdot m}_{n} \,|\, X \qquad [occ(VIRUS, X) > max - s]$$

Rule V1 describes the joining of phage with the bacterium membrane (step 1 of the process). The phage membrane is represented by a looping of k membrane elements v (with k fixed); DNA_v represents the phage DNA. The application of the rule causes the phage to become part of the bacterium membrane. Namely, the looping representing the phage becomes an element of the looping representing the bacterium membrane.

We remark that the situation described, in which the phage joins the membrane without entering it, cannot be described by membrane calculi as [5, 13].

Rule V2 models the releasing of phage DNA inside the bacterium. The phage membrane becomes a free open membrane (step 2).

Rule V3 describes the replication of phage DNA inside the bacterium (step 3). We assume that the replication happens only if the occurrences of DNA_v inside the bacterium are less than a number max.

Rule V4 describes the formation of a membrane around a phage DNA inside the bacterium (step 4).

Rule V5 models the breaking of the bacterium membrane when the number of phages inside it reaches a value close enough to max (the distance is less then a value $s > 0$). The bacterium membrane becomes a free open membrane, and everything contained in it (variable Y) is released (step 5).

Note that we have assumed that bacteria and phages cannot die a natural death. In particular, bacteria can die only if parasitized by viruses, and viruses die only when inoculating their DNA inside the bacterium.

We remark that congruence rules have the same number of constituent elements in the left- and in the right-hand side, and that the rewrite rules are monotonic. Hence, by Proposition 2.2, given an initial configuration of the system we can prove the reachability of a particular state. More general properties of the microbiological system we are considering, can be proved by model checking.

Example 3.1. Assume $max = 2$ and $s = 0$, namely that no replication of DNA_v can occur in a bacterium already containing two or more copies of DNA_v, and that the bacterium membrane can break when at least two viruses are inside. Consider the initial configuration in which there is one bacterium and three phages. This is represented by the term:

$$((\underbrace{m \cdot \ldots \cdot m}_{n})^L \rfloor DNA_b) \,|\, VIRUS \,|\, VIRUS \,|\, VIRUS.$$

We can prove that, in a possible evolution, we can reach the configuration:

$$(\underbrace{m \cdot \ldots \cdot m}_{n})^L \rfloor (DNA_b|DNA_v|DNA_v|DNA_v|DNA_v)|\underbrace{v \cdot \ldots \cdot v}_{k}|\underbrace{v \cdot \ldots \cdot v}_{k} \mid \underbrace{v \cdot \ldots \cdot v}_{k}.$$

The configuration represents a situation in which the bacterium contains a number of copies of virus DNA greater than max.

Actually, the steps to reach the configuration are the following: one virus infects the bacterium and its DNA is replicated inside the bacterium membrane (by application of rules V1, V2 and V3, in the order). Then the other two phages infect the bacterium (rule V1) and inoculate their DNA in it (rule V2).

4. Encoding Brane Calculi

In the previous section we have remarked that CLS can model situations which cannot be described by other membrane calculi. In this section, we recall the definition of the phago/exo/pino (PEP) calculus,

Syntax

$$P,Q,R\ldots ::= \diamond \mid P\circ P \mid !P \mid \sigma(\!|P|\!) \quad \text{Systems}$$
$$\sigma,\tau,\rho,\ldots ::= \mathbf{0} \mid \sigma|\sigma \mid !\sigma \mid a.\sigma \quad \text{Branes}$$
$$a,b,c,\ldots ::= \phi_n \mid \phi_n^{\perp}(\sigma) \mid \varepsilon_n \mid \varepsilon_n^{\perp} \mid \circledcirc\,(\sigma) \quad \text{Actions}$$

Structural Congruence

The least congruence relation $\equiv$ satisfying the following axioms

$$P\circ Q\equiv Q\circ P \qquad P\circ(Q\circ R)\equiv(P\circ Q)\circ R \qquad P\circ\diamond\equiv P$$
$$!\diamond\equiv\diamond \qquad !!P\equiv !P \qquad !P\equiv P\circ !P \qquad \mathbf{0}(\!|\diamond|\!)\equiv\diamond$$
$$\sigma|\tau\equiv\tau|\sigma \qquad \sigma|(\tau|\rho)\equiv(\sigma|\tau)|\rho \qquad \sigma|\mathbf{0}\equiv\sigma$$
$$!\mathbf{0}\equiv\mathbf{0} \qquad !!\sigma\equiv !\sigma \qquad !\sigma\equiv\sigma|!\sigma$$

Reaction Semantics

The least relation containing the following axioms, closed wrt $\_\circ P$, $\sigma(\!|\_|\!)$ and $\equiv$

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

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

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

Figure 5. The phago/exo/pino (PEP) calculus: syntax and semantics

which is the simplest of Brane Calculi [5], and we give a sound and complete encoding of it into CLS.

4.1. The PEP Calculus

The syntax and the semantics of the PEP calculus is summarized in Figure 5. Terms are systems. Systems consist of composition of systems, $\circ$, with unit $\diamond$. Replication ! is used to model the notion of "multitude" of systems. Systems can be membrane containing systems, $\sigma(\!|P|\!)$. Membranes can be a parallel compositions $\sigma|\sigma'$ with unit $\mathbf{0}$, or replication of membranes, or action prefixing.

Actions are: *phagocytosis*, denoted ϕ_n, incorporates one external membrane into another by "engulfing" it; *exocytosis*, denoted by ε_n, is the reverse process; *pinocytosis*, denoted by $\circledcirc$, engulfs zero external membranes. Phagocytosis and exocytosis have co-actions that are intended to interact with, indicated by the symbol $^{\perp}$. Pinocytosis does not have a co-action. Figure 6 gives a pictorial representation of the three actions.

We consider a structural congruence relation $\equiv$ that describes associativity, commutativity, replica-

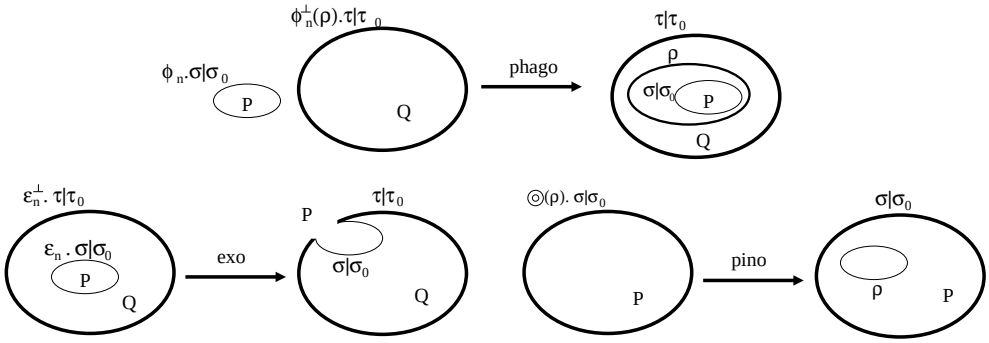

Figure 6. Pictorial representation of phagocytosis, exocytosis and pinocytosis

tion and unit elements of operators on systems and membranes. We denote with PEP the infinite set of Systems, and with $Branes$ the infinite set of membranes in the PEP calculus. Moreover, we denote with $\mathcal{N}$ the (possibly infinite) set of names n used as subscripts of Actions.

4.2. Encoding of the PEP Calculus

We define an encoding of a system of the PEP calculus into a CLS term. The encoding of a system results in a pair of a CLS sequence and a set of elementary constituents.

Operators and actions of the encoded system are translated into elements of the sequence. The encodings of the operands and of the action parameters follow the encodings of the corresponding operators and actions in the sequence, and are delimited by elements acting as separators. The set of elementary constituents given by the encoding contains all these separators.

The elementary constituent act is used in the sequence as a program counter: during the evolution of the term it preceeds every element which encodes a currently active action.

Definition 4.1. (Encoding)
The encoding of a system P of the PEP calculus into the CLS is the term $T \in \mathcal{T}$ such that, for some (finite) $E \subset \mathcal{E}$, it holds $\{\!|P|\!\} = (T, E)$, where $\{\!|\cdot|\!\} : PEP \to \mathcal{T} \times \mathcal{P}(\mathcal{E})$ is given by the following recursive definition:

$$\{\!|\diamond|\!\} = \big(act \cdot \mathbf{0}, \varnothing\big)$$

$$\{\!|P_1 \circ P_2|\!\} = \big(act \cdot circ \cdot a \cdot P_1'\{^{\epsilon}/_{act}\} \cdot a \cdot P_2'\{^{\epsilon}/_{act}\}, \{a\} \cup E_1 \cup E_2\big)$$
$$\text{where } \{\!|P_i|\!\} = (P_i', E_i), E_1 \cap E_2 = \varnothing \text{ and } a \in \mathcal{E} \setminus (E_1 \cup E_2)$$

$$\{\!|!P|\!\} = \big(act \cdot bangS \cdot P'\{^{\epsilon}/_{act}\}, E\big) \qquad \text{where } \{\!|P|\!\} = (P', E)$$

$$\{\!|\sigma(\!|P|\!)|\!\} = \big(act \cdot brane \cdot a \cdot \sigma'\{^{\epsilon}/_{act}\} \cdot a \cdot P'\{^{\epsilon}/_{act}\}, \{a\} \cup E_P \cup E_\sigma\big)$$
$$\text{where } \{\!|P|\!\} = (P', E_P), [\![\sigma]\!] = (\sigma', E_\sigma),$$
$$a \in \mathcal{E} \setminus (E_P \cap E_\sigma) \text{ and } E_P \cap E_\sigma = \varnothing$$

where $[\![\cdot]\!] : Branes \to \mathcal{T} \times \mathcal{P}(\mathcal{E})$ is given by the following recursive definition:

$$
\begin{aligned}
[\![\mathbf{0}]\!] &= \big(act \cdot \mathbf{0}, \varnothing\big) \\
[\![\sigma_1|\sigma_2]\!] &= \big(act \cdot par \cdot a \cdot \sigma_1'\{^{\epsilon}/_{act}\} \cdot a \cdot \sigma_2'\{^{\epsilon}/_{act}\} \cdot a, E_1 \cup E_2 \cup \{a\}\big) \\
&\quad \text{where } [\![\sigma_i]\!] = (\sigma_i', E_i), E_1 \cap E_2 = \varnothing \text{ and } a \in \mathcal{E} \setminus (E_1 \cup E_2) \\
[\![!\sigma]\!] &= \big(act \cdot bangB \cdot a \cdot \sigma'\{^{\epsilon}/_{act}\} \cdot a, E \cup \{a\}\big) \\
&\quad \text{where } [\![\sigma]\!] = (\sigma', E) \text{ and } a \in \mathcal{E} \setminus E \\
[\![\phi_n.\sigma]\!] &= \big(act \cdot \phi \cdot n \cdot a \cdot \sigma'\{^{\epsilon}/_{act}\} \cdot a, E \cup \{a\}\big) \\
&\quad \text{where } [\![\sigma]\!] = (\sigma', E) \text{ and } a \in \mathcal{E} \setminus E \\
[\![\phi_n^{\perp}(\rho).\sigma]\!] &= \big(act \cdot \phi^{\perp} \cdot n \cdot a \cdot \rho'\{^{\epsilon}/_{act}\} \cdot a \cdot \sigma'\{^{\epsilon}/_{act}\} \cdot a, E_\rho \cup E_\sigma \cup \{a\}\big) \\
&\quad \text{where } [\![\rho]\!] = (\rho', E_\rho), [\![\sigma]\!] = (\sigma', E_\sigma) \text{ and } a \in \mathcal{E} \setminus (E_\rho \cup E_\sigma) \\
[\![\varepsilon_n.\sigma]\!] &= \big(act \cdot \varepsilon \cdot n \cdot a \cdot \sigma'\{^{\epsilon}/_{act}\} \cdot a, E \cup \{a\}\big) \\
&\quad \text{where } [\![\sigma]\!] = (\sigma', E) \text{ and } a \in \mathcal{E} \setminus E \\
[\![\varepsilon_n^{\perp}.\sigma]\!] &= \big(act \cdot \varepsilon^{\perp} \cdot n \cdot a \cdot \sigma'\{^{\epsilon}/_{act}\} \cdot a, E \cup \{a\}\big) \\
&\quad \text{where } [\![\sigma]\!] = (\sigma', E) \text{ and } a \in \mathcal{E} \setminus E \\
[\![\circledcirc(\rho).\sigma]\!] &= \big(act \cdot \circledcirc \cdot a \cdot \rho'\{^{\epsilon}/_{act}\} \cdot a \cdot \sigma'\{^{\epsilon}/_{act}\} \cdot a, E_\rho \cup E_\sigma \cup \{a\}\big) \\
&\quad \text{where } [\![\rho]\!] = (\rho', E_\rho), [\![\sigma]\!] = (\sigma', E_\sigma) \text{ and } a \in \mathcal{E} \setminus (E_\rho \cup E_\sigma)
\end{aligned}
$$

In Figure 7 we give the rewrite rules which are applicable to encoded systems and membranes. We denote with $x, y, z, \ldots$ variables which can be instantiated only to single elementary constituens, and with $\widetilde{x}, \widetilde{y}, \widetilde{z}, \ldots$ variables which can be instantiated to (possibly empty) sequences of elementary constituents.

Rules are conceptually of two kinds. Rules from rule (par) to rule (sc6) rearrange elementary CLS sequences encoding PEP systems and membranes, into CLS terms (containing all CLS operators) and simplifying them accordingly to structural congruence on PEP terms. We denote with $\mathcal{R}_{\langle\rangle}$ this set of rules. Rules from rule (phago) to rule (bangB) correspond to PEP semantics. In particular, rules (phago), (exo) and (pino) correspond to phagocytosis, exocytosis and pinocytosis, respectively, and rules (bangS) and (bangB) correspond to structural congruence for the replication operator.

We remark that by applying rules in $\mathcal{R}_{\langle\rangle}$ to the encoding of a PEP system P we obtain a term T in which each membrane system $(\!|P'|\!)$ in P is represented by a looping sequence in T, and each occurrence of $\circ$ in P is represented by an occurrence of $\mid$ in T.

Example 4.1. Let us consider the PEP system $!\big(P\big)$ where $P = \phi_n(\!|\diamond|\!) \circ \phi_n^{\perp}(\mathbf{0})(\!|\diamond|\!)$. According to the semantics of the calculus the system may evolve as follows:

$$!(P) \;\equiv\; !(P) \circ \phi_n(\!|\diamond|\!) \circ \phi_n^{\perp}(\mathbf{0})(\!|\diamond|\!) \;\to\; !(P) \circ \mathbf{0}(\!|\mathbf{0}(\!|\mathbf{0}(\!|\diamond|\!)|\!) \circ \diamond|\!) \;\equiv\; !(P)$$

By applying the encoding to the system we obtain the following term T:

$$act \cdot bangS \cdot circ \cdot e \cdot brane \cdot b \cdot \phi \cdot n \cdot a \cdot \mathbf{0} \cdot a \cdot b \cdot \mathbf{0} \cdot e \cdot brane \cdot d \cdot \phi^{\perp} \cdot n \cdot c \cdot \mathbf{0} \cdot c \cdot \mathbf{0} \cdot c \cdot d \cdot \mathbf{0}$$

$$\left(act \cdot par \cdot x \cdot \widetilde{y} \cdot x \cdot \widetilde{z} \cdot x \cdot \widetilde{w}\right)^L \rfloor X \;\mapsto\; \left(act \cdot \widetilde{y} \cdot act \cdot \widetilde{z} \cdot \widetilde{w}\right)^L \rfloor X \qquad \text{(par)}$$

$$act \cdot circ \cdot x \cdot \widetilde{y} \cdot x \cdot \widetilde{z} \;\mapsto\; act \cdot \widetilde{y} \mid act \cdot \widetilde{z} \qquad \text{(circ)}$$

$$act \cdot brane \cdot x \cdot \widetilde{y} \cdot x \cdot \widetilde{z} \;\mapsto\; \left(act \cdot \widetilde{y}\right)^L \rfloor act \cdot \widetilde{z} \qquad \text{(brane)}$$

$$x \cdot \widetilde{w} \mid act \cdot \mathbf{0} \;\mapsto\; x \cdot \widetilde{w} \qquad \left(x \cdot \widetilde{w}\right)^L \rfloor act \cdot \mathbf{0} \;\mapsto\; \left(x \cdot \widetilde{w}\right)^L \qquad \text{(sc1,2)}$$

$$act \cdot bangS \cdot \mathbf{0} \;\mapsto\; act \cdot \mathbf{0} \qquad \left(act \cdot \mathbf{0}\right)^L \;\mapsto\; act \cdot \mathbf{0} \qquad \text{(sc3,4)}$$

$$\left(act \cdot \mathbf{0} \cdot x \cdot \widetilde{w}\right)^L \rfloor X \;\mapsto\; \left(x \cdot \widetilde{w}\right)^L \rfloor X \qquad \text{(sc5)}$$

$$\left(act \cdot bangB \cdot \mathbf{0} \cdot \widetilde{w}\right)^L \rfloor X \;\mapsto\; \left(act \cdot \mathbf{0} \cdot \widetilde{w}\right)^L \rfloor X \qquad \text{(sc6)}$$

$$\left(act \cdot \phi^{\perp} \cdot x_n \cdot x \cdot \widetilde{y} \cdot x \cdot \widetilde{z} \cdot x \cdot \widetilde{w}\right)^L \rfloor X \mid \left(act \cdot \phi \cdot x_n \cdot x' \cdot \widetilde{y'} \cdot x' \cdot \widetilde{z'}\right)^L \rfloor Y$$
$$\mapsto \left(act \cdot \widetilde{z} \cdot \widetilde{w}\right)^L \rfloor \left(X \mid \left(act \cdot \widetilde{y}\right)^L \rfloor \left(act \cdot \widetilde{y'} \cdot \widetilde{z'}\right)^L \rfloor Y\right) \qquad \text{(phago)}$$

$$\left(act \cdot \varepsilon^{\perp} \cdot x_n \cdot x \cdot \widetilde{y} \cdot x \cdot \widetilde{z}\right)^L \rfloor \left(X \mid \left(act \cdot \varepsilon \cdot x_n \cdot x' \cdot \widetilde{y'} \cdot x' \cdot \widetilde{z'}\right)^L \rfloor Y\right)$$
$$\mapsto\; Y \mid \left(act \cdot \widetilde{y} \cdot \widetilde{z} \cdot act \cdot \widetilde{y'} \cdot \widetilde{z'}\right)^L \rfloor X \qquad \text{(exo)}$$

$$\left(act \cdot \circledcirc \cdot x \cdot \widetilde{y} \cdot x \cdot \widetilde{z} \cdot x \cdot \widetilde{w}\right)^L \rfloor X \;\mapsto\; \left(act \cdot \widetilde{z} \cdot \widetilde{w}\right)^L \rfloor \left(X \mid \left(act \cdot \widetilde{y}\right)^L\right) \qquad \text{(pino)}$$

$$act \cdot bangS \cdot \widetilde{x} \;\mapsto\; act \cdot bangS \cdot \widetilde{x} \mid act \cdot \widetilde{x} \qquad \text{(bangs)}$$

$$\left(act \cdot bangB \cdot x \cdot \widetilde{y} \cdot x \cdot \widetilde{w}\right)^L \rfloor X \;\mapsto\; \left(act \cdot bangB \cdot x \cdot \widetilde{y} \cdot x \cdot act \cdot \widetilde{y} \cdot \widetilde{w}\right)^L \rfloor X \qquad \text{(bangb)}$$

Figure 7. Rewrite rules associated with the encoding of the PEP calculus

which may evolve as follows:

$$\begin{aligned}
T \;&\to\; T \mid act \cdot circ \cdot e \cdot brane \cdot b \cdot \phi \cdot n \cdot a \cdot \mathbf{0} \cdot a \cdot b \cdot \mathbf{0} \cdot e \\
&\quad \cdot brane \cdot d \cdot \phi^{\perp} \cdot n \cdot c \cdot \mathbf{0} \cdot c \cdot \mathbf{0} \cdot c \cdot d \cdot \mathbf{0} && (bangS) \\
&\to\; T \mid act \cdot brane \cdot b \cdot \phi \cdot n \cdot a \cdot \mathbf{0} \cdot a \cdot b \cdot \mathbf{0} \\
&\quad \mid act \cdot brane \cdot d \cdot \phi^{\perp} \cdot n \cdot c \cdot \mathbf{0} \cdot c \cdot \mathbf{0} \cdot c \cdot d && (circ) \\
&\Rightarrow\; T \mid \left(act \cdot \phi \cdot n \cdot a \cdot \mathbf{0} \cdot a\right)^L \rfloor act \cdot \mathbf{0} \\
&\quad \left(act \cdot \phi^{\perp} \cdot n \cdot c \cdot \mathbf{0} \cdot c \cdot \mathbf{0} \cdot c\right)^L \rfloor act \cdot \mathbf{0} && 2 \times (brane) \\
&\to\; T \mid \left(act \cdot \mathbf{0}\right)^L \rfloor \left(act \cdot \mathbf{0} \mid \left(act \cdot \mathbf{0}\right)^L \rfloor \left(act \cdot \mathbf{0}\right)^L \rfloor act \cdot \mathbf{0}\right) && (phago) \\
&\Rightarrow\; T
\end{aligned}$$

Now we introduce a normal form for CLS terms which will be used to prove the correctness of the encoding. This normal form can be obtained by applying rules in $\mathcal{R}_{\langle\rangle}$ as long as possible.

Proposition 4.1. (Normal Form)
Assume $\mathcal{R}_{\langle\rangle}$ as the set of rules that can be applied to terms. Given a CLS term T, there exists a unique CLS term (modulo structural congruence), denoted $\langle T\rangle$, such that $T \rightarrow^* \langle T\rangle$ and $\langle T\rangle \not\rightarrow$.

Proof:
The term $\langle T\rangle$ is reachable after a finite number of rule applications as all rules in $\mathcal{R}_{\langle\rangle}$ reduce the number of elementary constituents in the term. Moreover, it is easy to see that, by definition of the rules in $\mathcal{R}_{\langle\rangle}$, $\langle T\rangle$ is unique. □

We prove now the correctness of the encoding in terms of soundness and completeness. For the sake of simplicity, let us denote with $\{\![P]\!\}$ and $[\![\sigma]\!]$ the terms obtained by the application of the encoding to system P and to membrane σ, respectively. Moreover, we denote with $\rightarrow^*$ the reflexive and transitive closure of $\rightarrow$ for both CLS and PEP semantics.

Theorem 4.1. (Soundness)
Given a system P of the PEP calculus:

$$P \rightarrow P' \implies \exists T.\exists P''.\ \text{s.t. } \{\![P]\!\} \rightarrow^* T, \langle T\rangle \equiv \langle \{\![P'']\!\}\rangle \text{ and } P'' \equiv P' .$$

Proof:
We show first that structurally congruent PEP systems without replication are encoded into CLS terms whose normal forms are structurally congruent. As regards replication, we show that given two congruent PEP systems the encoding of one can be transformed into the encoding of the other by applications of a rule in $\mathcal{R}_{\langle\rangle}$. Therefore, the prove can be done by induction on the structure of P without considering structural congruence. □

Theorem 4.2. (Completeness)
Given a system P of the PEP calculus:

$$\{\![P]\!\} \rightarrow^* T \implies \exists P' \text{ s.t. } \langle T\rangle \equiv \langle \{\![P']\!\}\rangle \text{ and either } P \equiv P' \text{ or } P \rightarrow^* P' .$$

Proof:
By induction on the number of steps in $\{\![P]\!\} \rightarrow T$. □

5. Conclusions

The paper presents a new calculus suitable to describe microbiological systems and their evolution. We use the calculus to model interactions among bacteria and bacteriophage viruses, and to reason on their properties, and we give an encoding of one of Cardelli's Brane Calculi into ours.

We remark that systems composed by parts delimited by membranes and whose evolution may be described by rewrite rules have been considered by authors interested in proposing new models of computing inspired by biological systems (natural computing). See e.g. [12].

We believe that a further step of our work should be the introduction in the calculus of concepts of time and probability, which have been considered in other formalisms and have been shown to be crucial for describing biological systems [2, 3].

References

[1] R. Alur, C. Belta, F. Ivancic, V. Kumar, M. Mintz, G.J. Pappas, H. Rubin and J. Schug. “Hybrid modeling and simulation of biomolecular networks”. In Hybrid Systems: Computation and Control, LNCS 2034, pages 19–32, Springer, 2001.

[2] R. Barbuti, S. Cataudella, A. Maggiolo-Schettini, P. Milazzo and A. Troina. “A probabilistic model for molecular systems”. Fundamenta Informaticae, volume 67, pages 13–27, 2005.

[3] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo and A. Troina. “An alternative to Gillespie’s algorithm for simulating chemical reactions”. Computational Methods in Systems Biology (CMSB’05), Edinburgh, 2005.

[4] L. Cardelli. “Bioware languages”. In Computer Systems. Theory, Technology and Applications. Papers for Roger Needham, Springer, pages 59–66, 2003.

[5] L. Cardelli. “Brane calculi. interactions of biological membranes”. Computational Methods in Systems Biology (CMSB’04), LNCS 3082, pages 257–280, Springer, 2005.

[6] N. Chabrier-Rivier, M. Chiaverini, V. Danos, F. Fages and V. Schachter. “Modeling and querying biomolecular interaction networks”. Theoretical Computer Science, volume 325, number 1, pages 25-44, 2004.

[7] M. Curti, P. Degano, C. Priami and C.T. Baldari. “Modelling biochemical pathways through enhanced pi-calculus”. Theoretical Computer Science, volume 325, number 1, pages 111–140, 2004.

[8] V. Danos and C. Laneve. “Formal molecular biology”. Theoretical Computer Science, volume 325, number 1, pages 69–110, 2004.

[9] C. Flanagan and M. Abadi. “Object types against races”. CONCUR’99, LNCS 1664, pages 288-303, Springer, 1999.

[10] A. Gordon and P. Hankin. “A concurrent object calculus: reduction and typing”. High-Level Concurrent Languages (HLCL’98), Elsevier ENTCS, volume 16, number 3, 1998.

[11] H. Matsuno, A. Doi, M. Nagasaki and S. Miyano.“Hybrid Petri net representation of gene regulatory network”. In Pacific Symposium on Biocomputing, World Scientific Press, pages 341–352, 2000.

[12] G. Păun. “Membrane computing. an introduction”. Springer, 2002.

[13] A. Regev, E.M. Panina, W. Silverman, L. Cardelli and E. Shapiro. “BioAmbients: an abstraction for biological compartments”. Theoretical Computer Science, volume 325, number 1, pages 141–167, 2004.

[14] A. Regev, W. Silverman and E.Y. Shapiro. “Representation and simulation of biochemical processes using the pi-calculus process algebra”. Pacific Symposium on Biocomputing, World Scientific Press, pages 459–470, 2001.