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
{barbuti,milazzo,maggiolo,troina}@di.unipi.it

Abstract. 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.

1 Introduction

In the past 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 [9], Hybrid Systems [1], and the π-calculus [12, 7]. Moreover, some new formalisms have been proposed to describe biomolecular and membrane interactions [2, 4–6, 8, 11].

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 closing 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 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], and justifies the introduction of a new calculus.

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 rewriting rules to be applied to terms. The definition of the rewriting rules depends on the system and the system evolution one wants to represent.

In this paper we are interested in modelling interactions among bacteria and bacteriophage viruses as well as bacteria sporulation. We represent bacteria and viruses as terms, and give a set of rewriting rules for describing how bacteria produces spores and spores germinate, and how bacteriophages parasitize bacteria.

Properties of the represented systems may be proved in two ways. The reachability of particular configurations may be decided for the systems of rewrite rules we consider. Properties, expressed in a suitable logic, may be proved on models obtained from the evolutions of the systems produced by applying rewrite rules. Abstraction may be necessary to have finite models. As examples, some properties referring to the interaction among bacteria and bacteriophage viruses are discussed.

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 $a, b, c, \ldots$.

Definition 1 (Terms). *A* Term T *of CLS is given by the following grammar:*

$$\begin{array}{rcl} T & ::= & Seq \;\mid\; (T)^L \;\mid\; T \rfloor T \;\mid\; T \mid T \\ Seq & ::= & a \;\mid\; T \cdot T \end{array}$$

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

A sequence Seq may be either an element in $\mathcal{E}$ or a concatenation of terms $T_1 \cdot T_2$. An example of sequence is $a \cdot b \cdot c$.

A term T may be a sequence Seq, a looping $(T)^L$ or a combination of terms by means of the containment operator $\rfloor$ and the parallel operator $\mid$. 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 $T_1 \mid T_2$ represents the adjacency of terms T_1 and 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 $\mid$ 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 $\mid$ operator. Therefore $T_1 \rfloor T_2 \mid T$ stands for $(T_1 \rfloor T_2) \mid T$.

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 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 $\big(\big(\big(c \cdot d \cdot e\big)^L \rfloor h\big) \cdot a \cdot b\big)^L \rfloor f \cdot g$.

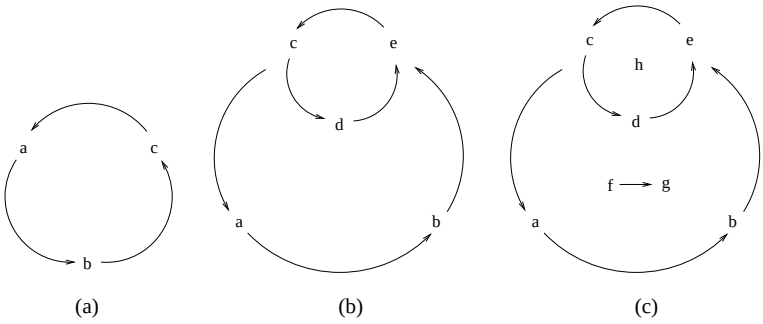

Fig. 1. Examples of CLS terms

Definition 2 (Structural Congruence). *The structural congruence $\equiv$ is the least congruence relation on terms satisfying associativity of $\_ \mid \_$ and $\_ \cdot \_$, right–associativity of $\_ \rfloor \_$ and the following axioms:*

$$
\begin{array}{ll}
A1. & (T_1 \mid T) \cdot T_2 \equiv (T_1 \cdot T_2) \mid T \equiv T_1 \cdot (T_2 \mid T) \\
A2. & (T_1 \mid T_2) \rfloor T \equiv (T_1 \rfloor T) \mid T_2 \\
A3. & \big(T \mid T_1\big)^L \equiv \big(T\big)^L \mid T_1 \\
A4. & T \mid T_1 \mid T_2 \equiv T \mid T_2 \mid T_1 \\
A5. & (T_1 \rfloor T_2) \rfloor T_3 \equiv T_1 \rfloor (T_2 \mid T_3) \\
A6. & \big(T_1 \cdot T_2\big)^L \equiv \big(T_2 \cdot T_1\big)^L \\
A7. & Seq \rfloor T \equiv Seq \mid T
\end{array}
$$

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 a special role and it cannot be commuted in a series of parallel composition. This is expressed by axiom A4.

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.

Finally, axiom A7 says 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.

Axioms A1, A2 and A3 are justified by the need of modeling a real situation in which a membrane that is part of another membrane breaks. 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. We assume that the content of the smaller membrane is freed outside the

resulting membrane. This is shown in Figure 2.b. Formally, the initial situation in Figure 2.a corresponds to the following term (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): $(((b\cdot\ldots\cdot b)^L \rfloor a)\cdot c\cdot\ldots\cdot c)^L$. Breaking the smaller membrane means removing the looping operator of the sequence of b elements, thus obtaining the term $((b\cdot\ldots\cdot b \rfloor a)\cdot c\cdot\ldots\cdot c)^L$. Applying congruence A7, 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.

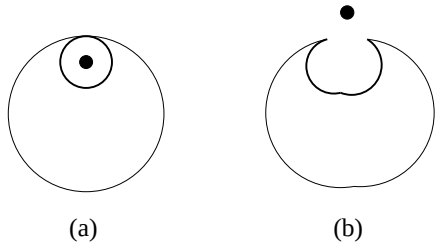

Fig. 2. A real situation

In general, we can prove the following proposition.

Proposition 1. *Consider a term consisting of a nesting of looping terms of the form:*

$$T = \left(\left(\left(\ldots\left(a_1\cdot\ldots\cdot a_n\right)^L \rfloor T_i\ldots\right)^L \rfloor T_2\ldots\right)^L \rfloor T_1\ldots\right)^L \rfloor T_0$$

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

$$\left(\left(\left(\ldots(a_1\cdot\ldots\cdot a_n) \rfloor T_i\ldots\right)^L \rfloor T_2\ldots\right)^L \rfloor T_1\ldots\right)^L \rfloor T_0$$

is structurally congruent to:

$$\left(\left(\left(\left(\ldots a_1\cdot\ldots\cdot a_n\ldots\right)^L \rfloor T_2\ldots\right)^L \rfloor T_1\ldots\right)^L \rfloor T_0\right) \mid T_i$$

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

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 function $\sigma : V \to \mathcal{T}$. With $\mathcal{T}_\mathcal{V}$ we denote the set of CLS terms which may also

contain variables in V and, given $T \in \mathcal{T}_{\mathcal{V}}$, 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 $T \in \mathcal{T}_{\mathcal{V}}$, 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 $T \in \mathcal{T}_{\mathcal{V}}$, 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 $occ : \mathcal{T} \times \mathcal{T} \to \mathbb{N}$ such that $occ(T', T)$ returns the number of the terms T' syntactically occurring in the term T.

Definition 3 (Rewrite Rule). *A rewrite rule is a triple (T, T', Σ') such that $T, T' \in \mathcal{T}_{\mathcal{V}}$, $Var(T') \subseteq Var(T)$, $\Sigma' \subseteq \Sigma$ 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 rules and under the application of the operators.

Definition 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 a term $T \in \mathcal{T}$, we say that a term $T' \in \mathcal{T}$ is *reachable* from T iff there exist $T_1, \ldots, T_n \in \mathcal{T}$ such that $T \rightarrow T_1 \rightarrow \ldots \rightarrow T_n \rightarrow T'$.

Now we introduce a particular case of rewrite rules.

Definition 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.

Proposition 2. *Given a 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 the formalism introduced in Section 2 can be used for describing 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, sporulation 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).

The entire process is depicted in Figure 3.

For the sake of clarity, before giving the rules for the process, let us introduce some denotations for terms which occur very often:

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

$$SPORE_1 ::= \left(\underbrace{c \cdot \ldots \cdot c}_{\frac{n}{2}}\right)^L \rfloor PRESPORE$$

$$SPORE_2 ::= \left(\underbrace{d \cdot \ldots \cdot d}_{\frac{n}{2}}\right)^L \rfloor PRESPORE$$

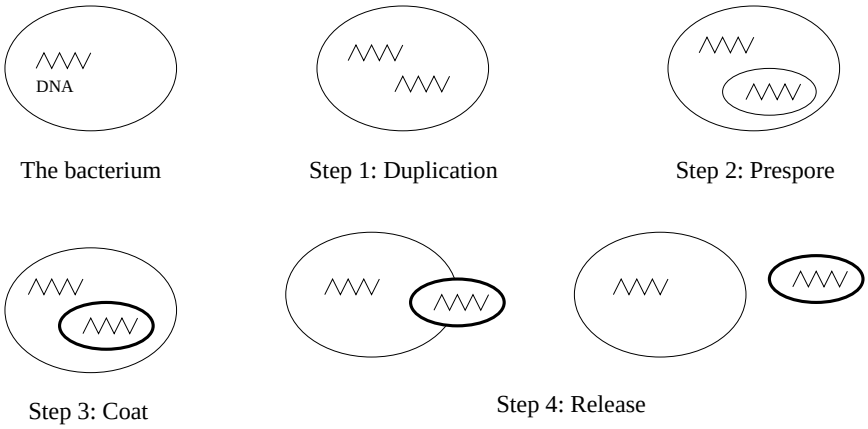

Fig. 3. The Sporulation Process

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

$S1.\quad \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (DNA_b \mid X) \longrightarrow$

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

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

$$\big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (DNA_b \mid PRESPORE \mid X)$$

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

$$\big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor (X \mid SPORE_1 \mid Y)$$

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

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

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

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; 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 $\frac{n}{2}$ (where n is the number of the bacterium membrane elements).

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) are unable to reproduce themselves autonomously, therefore they exploit the enzymes of the bacteria for duplicating their DNA. In particular, the viruses we consider behave according to the following pattern:

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 exploiting 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 membrane reaches a certain number, the membrane breaks and the new phages become free (release).

The entire process is depicted in Figure 4.

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

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

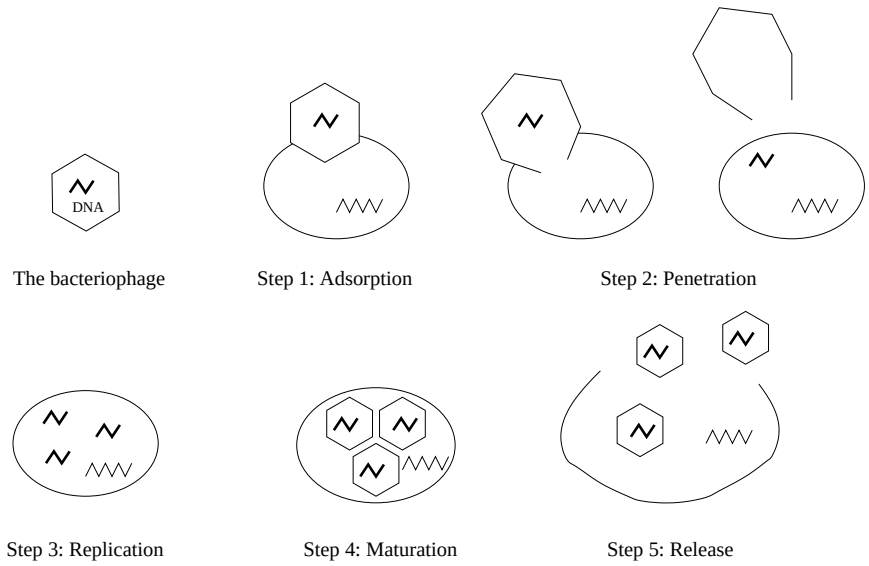

Fig. 4. The Bacteriophage Replication Process

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

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

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

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

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

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

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; 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 assume a symmetric rule for V1, where the first element of the parallel composition is the bacterium and the virus is in second place.

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, given an initial configuration of the system we can prove the reachability of a particular state.

Example 2. 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:

$$BACTERIUM \mid VIRUS \mid VIRUS \mid VIRUS$$

where

$$BACTERIUM \ ::= \ \big(\underbrace{m \cdot \ldots \cdot m}_{n}\big)^L \rfloor \, DNA_b.$$

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

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

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).

More general properties of the microbiological system we are considering, can be proved by using model checking. Properties one may verify are shown in the following examples.

Example 3. Suppose that the duplication of the bacteria DNA is inhibited, namely that rule S1 is removed. Then, starting from a configuration in which bacteria and phages are present, eventually, in every possible evolution, we reach a state in which no bacteria are present.

Example 4. Suppose that a spore is inhibited from crossing the bacterium membrane, namely that rule S4 is removed. Assume that we start from a configuration with a number of bacteria equal to N and a number of phages greater than 0. Then, in every reachable configuration, the number of bacteria is less than or equal to N. The reason of this behaviour is that a spore can exit the bacterium membrane only if a phage infection breaks it. But, in this case, the parent bacterium dies freeing the spore. In this example we cannot exclude that the number of viruses can grow indefinitely, thus producing an infinite model. However, for the sake of proving the property, one may assume a finite number of phages. Analogous situations may be coped with by using suitable abstractions.

4 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.

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. [10].

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. 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.
10. G. Păun. “Membrane Computing. An Introduction”. Springer, 2002.
11. A. Regev, E.M. Panina, W. Silverman, L. Cardelli and E. Shapiro. “BioAmbients: An Abstraction fo Biological Compartments”. Theoretical Computer Science, volume 325, number 1, pages 141–167, 2004.
12. 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.