Quorum Sensing P Systems

Francesco Bernardini [a], Marian Gheorghe [a,1], Natalio Krasnogor [b]

[a] *Department of Computer Science, The University of Sheffield Regent Court, Portobello Street, Sheffield S1 4DP, UK*

[b] *Automated Scheduling, Optimisation and Planning Research Group School of Computer Science and Information Technology University of Nottingham, Jubilee Campus, Nottingham NG8 1BB, UK*

Abstract

This paper continues the investigation of population P systems model [4] by considering bacterium quorum sensing (QS) phenomena as basis of the new approach. A new computational model called QS P system is introduced. It is proved that QS P systems are able to simulate counter machines, and hence they are equivalent in power to Turing machines. An example of a QS P system modelling the behaviour of Vibrio fischeri bacteria colonies is also presented and the emergence of the QS mechanism is illustrated.

Key words: P systems, Turing machines, Quorum sensing, Vibrio fischeri

1 Introduction

The inter-play between computer science and biology is a long standing process that has led to research developments in each of these areas and also in new emerging disciplines like computational biology or natural computing. The former covers all aspects of computational modelling that refer to biological sciences as applied to biological processes, ranging from specific algorithms and data structures for genomic and proteomic support up to highly complex

[1] *Corresponding author: Marian Gheorghe Tel: +44 114 2221843 Fax: +44 114 2221810*

[2] *Email addresses: F.Bernardini@dcs.shef.ac.uk, M.Gheorghe@dcs.shef.ac.uk, Natalio.Krasnogor@nottingham.ac.uk*

systems biology modelling. The wide spectrum of models used in systems biology includes continuous and discrete approaches, deterministic and stochastic methods [32], and recent developments of integrative complex hybrid methods [23]. On the other hand, natural computing envisages the development of new computational models and paradigms inspired from the way nature behaves. Well-established fields of research have been developed: neural computing [2], genetic programming [21], evolutionary computing and programming [7] up to the new fields of DNA computing [28], gene assembly [11], membrane computing [26] etc.

Compartments play an important role in all the biological systems by organising their structure in a hierarchical way with well defined components [22]. At the cellular level, the membrane is the main structural component that delimits its internal compartments and the cell from its environment. Membranes are essentially involved in many reactions taking place inside different compartments of a cell, and they act as selective channels of communication between different regions as well as between the cell and its environment [1].

Although the range of modelling is very broad, there are very few models addressing the role of membranes and compartments in biological systems. Notable exceptions are hybrid Petri net model, where a stochastic Petri net abstraction is extended with an additional layer [25], and Statecharts where the compartmentalisation is an integral part of the model [19]. In [30], an extended version of the stochastic $\pi-$ calculus, called bioambients, is introduced in order to provide a better abstraction of compartmentalisation that allows the study of biological compartments of different granularities, movement of molecules between compartments, and dynamic rearrangements of compartments and chemical complexes. Another modelling perspective considering the cell structure with its compartments as a key concept has been introduced through the *P system* framework [26]. This model, also called *membrane computing,* is grounded in formal grammars. Membrane computing formalises essential features of living cells. However, according to its original motivation, it does not intend to provide a comprehensive and accurate model of the cell but, rather, to explore the computational nature of biological membranes and compartments that define a biological cell [26]. Recent developments have been dedicated to the use of the P system paradigm as a modelling tool for various biological systems [10].

The P system model that has been developed as an abstraction of the cell structure and behaviour is intended to be [27]: (i) minimalistic (as elegant as possible, containing as restricted ingredients as possible), but (ii) without losing the biological inspiration (hence remaining as "realistic" as possible), with (iii) good computability properties (as *powerful* as possible, compared with standard models of computing – Turing machine and its restrictions, and as *efficient* as possible – useful in solving computationally hard problems in

a feasible time). These are contradictory criteria and consequently different models proposed are focusing only on some of these aspects.

Briefly, the *P system* model consists of a membrane structure containing several membranes arranged in a hierarchical structure inside of a main membrane, called the *skin*, and delimiting *regions*. Each region is bounded by a membrane surrounding it and the immediately inner membranes, if any. The membranes which do not contain any other membrane inside are called *elementary* membranes. A graphical representation of a membrane structure is given in Figure 1.

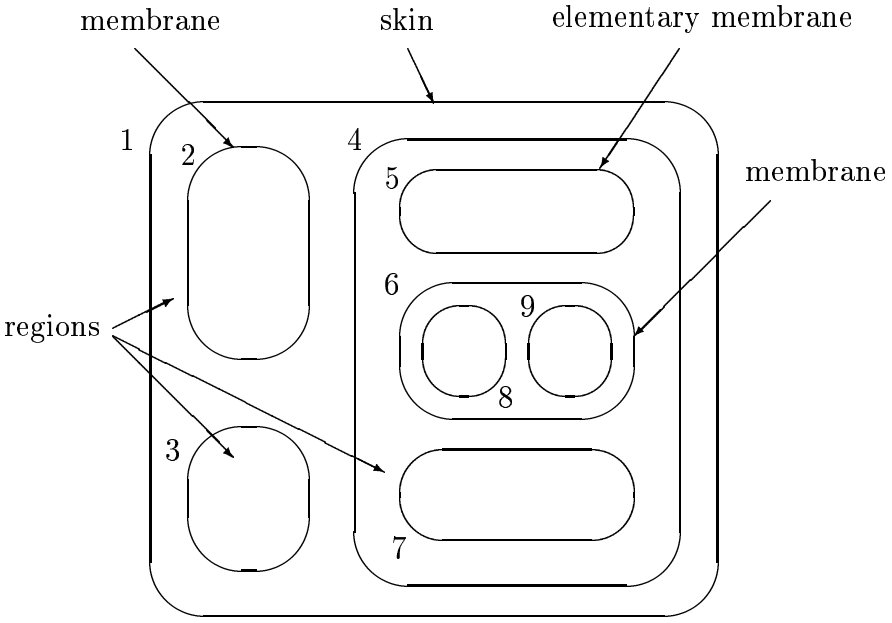


Fig. 1. A membrane structure containing 9 membranes and 9 corresponding regions; labels are used to uniquely identify each distinct membrane in the system.

A membrane structure can be represented as a tree with nodes representing the membranes; the root represents the skin membrane and for a given node the direct descendants correspond to the membranes that are immediately inside of the membrane associated to that node. The leaves of the tree correspond to elementary membranes.

Each region contains zero or more *objects*, each one appearing with a specific multiplicity. That is, each region, in general, hosts a *multiset of objects* rather than a set. As well as this, finite sets of *evolution rules* are assigned to regions, which are used to modify the objects associated with these regions. The objects can also be moved (*communicated*) between regions. The membranes can be created, divided, dissolved and their permeability modified. The rules have a *local scope*: those assigned to a specific region, inside the system, can be

applied only to the objects and the membranes that exist in that region. The rules are applied non-deterministically and in a maximally parallel manner (in one step, all the objects that can evolve and the membranes that can be transformed must do so). In this way, we get *transitions* from one *configuration* of the system to another one. A *computation* in a P system is a sequence of transitions. The transition process is supposed to be synchronous: a global clock is assumed, marking the time units common to all the regions. A computation leading to a configuration where no more rules can be applied to the objects occurring inside the membranes is called a *halting* one. Each halting computation in a P system produces a result, which is typically given by the number of objects contained inside a well-specified output membrane in the final configuration. A computation that does not halt, yields no result.

Since this model has been proposed for the first time in 1998 [26], many variants of P systems have been proposed that retain the basic features discussed above and essentially differ only by the kind of rules associated with the regions of the system and by the mechanisms used to control the way they are applied (priority relationships, contextual conditions, variable number of rules etc). Moreover, since many chemical or biological components occurring in various reactions may have a more complex structure, strings of symbols (*string-objects*) rather than simple objects have been considered. This approach recalls other biologically inspired models acting on strings, Lindenmayer systems [24] and splicing systems [28]. The study of P systems has been focusing on the *computational power* with respect to the notion of Turing computability, or considering their *efficiency* in algorithmically solving hard problems, like NP-complete problems, in a polynomial time. For further details, examples and other variants we refer to [26]. A comprehensive bibliography of P systems and further information about membrane computing can be found at the P systems web page [36].

Besides cell-like P systems, there have been also considered *tissue-like* and *neural-like* systems, inspired by the way cells cooperate in tissues and neural nets formation [26]. The structure of these systems is represented as a graph. Its nodes represent the membranes (or cells) and the edges are interpreted as abstract communication channels. A related notion is that of *population P systems* [4]. From a biological point of view, a population P systems model is an abstraction of the way populations of bio-units aggregate and function as a bio-entity that is more complex than its component parts. The biological systems are composed of many individual components that interact and cooperate with each other as well as with an environment. Most of the time these complex systems will have a dynamic structure, with new individuals joining the system, others leaving it, with dynamically established inter-relationships. The population P systems model addresses features which are common to many different biological systems from the molecular level to the case of colonies of more complex organisms. A similar modelling perspective has been expressed

by using agent based approaches. For instance, intracellular protein interactions are described in [12] as individual and autonomous agents capable of mediating the cell responses to external signalling through a complex network of interactions where information is processed in a distributed and parallel way. Agent-based approaches are also used in [34] to model tissues and in [17], [16] to model ant colonies. In fact, in formal language theory, the aforementioned idea of agents is not new as it has already been considered in grammar systems [8], [9] where languages are generated through the cooperation between different string-processing components. More recently, a concept of P colony has been introduced as a combination of the eco-grammar paradigm and the P systems approach [20], [13].

In a variant of population P systems that has been studied, the underlying graph is being continuously modified [4]. After each step of transformation-communication, according to a specific set of bond making rules, new edges are added to or old ones are removed from the graph defining the current structure of the system. A transformation-communication step involves rules modifying the objects and communicating them from one membrane (cell) to another. In particular, communication rules are inspired by the general mechanism of cell communication based on signals and receptors and they are presented with a formalism related to the notion of boundary rules [5]. As well as this, cell division rules for duplicating the existing cells in the system, cell differentiation rules for changing the type of the cells, and cell death rules for removing cells from the system, can also be applied in a transformation-communication step. Moreover, another important feature of the model is the notion of environment as a repository of objects that are sent out from the cells. The objects in the environment can subsequently re-enter the cells, and this provides a form of indirect communication between cells.

The power of population P systems has been investigated by considering different restrictions: when the structure of the underlying graph cannot be modified by any rule or when only the edges of the graph can be modified by using a finite set of bond making rules or when both the edges and the nodes in the graph can be modified by allowing cell division, cell differentiation, and cell death rules. In the former case, the main result obtained shows the universality for completely unstructured P systems where cells can communicate only indirectly by means of the environment without ever forming any bond. Next, when bond making rules are considered, the population P systems model is computationally complete even when simple communication rules that move objects from a cell to another one without any restriction are used. It is also proved that the population P systems model using only the operation of cell differentiation is again computationally complete. Another universality result is obtained for population P systems where cell division and bond making rules are allowed [4].

Recently, population P systems with boundary rules that are applied with a limited parallelism have been considered and a framework of a formal semantics similar to the operational style presented in [3] has been defined [6]. P systems with boundary rules and limited parallelism [29] and population P systems [33] have been considered in modelling quorum sensing (QS) in bacteria.

QS is a widely used strategy for behavioural coordination among bacteria. It is important to notice that the process of coordinated gene-expression (and hence phenotypic change) in bacteria is best understood by noticing that only by pooling together the activity of a quorum of cells can a colony be successfull. The QS mechanism is a communication strategy based on diffusible signals, which kick-in under high cellular density. In particular, it is generally observed that, when a QS process is activated, the concentration of the signal molecules reflects the number of cells in the colony, or at least the number of cells in a particular physiological state. Bacteria can then respond to variations of the concentration of signal molecules and, when this value exceeds a specific threshold limit that indicates the population is "quorated", they start to behave in a coordinated way (i.e., they "sense" that the quorum necessary to take a collective decision has been reached). This behavioural shift is possible because QS changes the gene expression of a bacterium's DNA by triggering a cascade of transcriptional activity through the activation of several genes.

The use of the P system approach in the context of QS has shown that although the variants considered seem to be quite appropriate as abstract models for cell-to-cell communication, there are aspects that are yet not captured. The fact that the environment has a limited capacity and the bacteria communicate locally, the signalling process has various intensities depending on the status of the colony, are some of the aspects that have not been addressed in the previous variants of the model and that constitute the focus of the current investigation.

In this paper we discuss a new class of population P systems inspired by QS mechanisms in bacterial colonies and investigate the power of it. We also briefly present a model of quorum sensing in Vibrio fischeri bacteria by using the current theoretic approach. We show that we can model at least the aspects provided in [29], but we are also able to approach problems regarding the finiteness of the environment (components) and local interactions between bacteria.

2 Preliminaries

We here recall some basic notions and notations commonly used in membrane computing and in formal language theory that we need in the rest of the paper. We refer to [26], [31] for further details.

An alphabet is a finite non-empty set of abstract symbols. Given an alphabet V, we denote by V^* the set of all possible strings over V, including the empty string λ. The length of a string $x \in V^*$ is denoted by $|x|$ and, for each $a \in V$, $|x|_a$ denotes the number of occurrences of the symbol a in x. A multiset over V is a mapping $M : V \longrightarrow N$ such that, $M(a)$ denotes the multiplicity of a in the multiset M (N denotes the set of natural numbers). Such a multiset can be represented by a string $a_1^{M(a_1)} a_2^{M(a_2)} \ldots a_n^{M(a_n)} \in V^*$ and by all its permutations with $a_j \in V$, $M(a_j) \neq 0$, $1 \leq j \leq n$. In other words, we can say that each string $x \in V^*$ identifies a finite multiset over V defined by $M_x = \{ (a, |x|_a) \mid a \in V \}$. Moreover, given two strings $x, y \in V^*$, we denote by xy their catenation, which corresponds to the union of the multiset represented by string x and the multiset represented by string y.

In the following proofs we will use the notion of a *counter machine* in the form considered in [14]. Informally, a counter machine is a finite state machine that has a finite number of counters able to store values represented by natural numbers; the machine runs a program consisting of instructions that can increase or decrease by one the contents of the registers or can test them for zero, changing in the same time the state of the machine; starting with each counter being empty, the machine performs a computation; if it reaches a terminal state, then a vector corresponding to the values stored in some specified counters is said to be generated during this computation.

Definition 1 *A counter machine is a construct*

$$M = (Q, F, p_0, A, O, I),$$

where:

(1) Q *is the set of states,*
(2) $F \subseteq Q$ *is the set of final states,*
(3) $p_0 \in Q$ *is the initial state,*
(4) A *is the set of counters,*
(5) $O = (c_1, \ldots, c_k)$ *with* $c_i \in A$, *for all* $1 \leq i \leq k$, *is an ordered tuple of output counters,*
(6) I *is a finite set of instructions of the following forms:*
  (a) $(p \rightarrow q, +c)$, *with* $p, q \in Q$, $c \in A$: *add 1 to the value of the counter* c *and move from state* p *into state* q;
  (b) $(p \rightarrow q, -c)$, *with* $p, q \in Q$, $c \in A$: *if the current value of the counter*

c is not zero, then subtract 1 from the value of the counter c and move from state p into state q; otherwise the computation is blocked in state p;

(c) $(p \rightarrow q, c = 0)$, *with* $p, q \in Q$, $c \in A$: *if the current value of the counter c is zero, then move from state p into state q; otherwise the computation is blocked in state p;*

(d) $(p \rightarrow q, \epsilon)$, *with* $p, q \in Q$: *move from state p into state q without changing the value of any counter.*

A transition step in such a counter machine consists in updating/checking the value of a counter according to an instruction of one of the types presented above and moving from a state to another one. Starting with the number zero stored in each counter, we say that the counter machine computes the vector $(n_1, \ldots, n_k)$ if and only if, $O = (c_1, \ldots, c_k)$ and, by starting from the initial state, the machine reaches a final state after a finite sequence of transitions by having produced the value n_i inside every counter c_i with $1 \leq i \leq k$. Without loss of generality, we may assume that in the end of the computation the machine makes zero all the counters but the output ones, and that there are no transitions that start from a final state. From a computational point of view, counter machines are proved to be equivalent to Turing machines [15], and we will make below an essential use of this result. Specifically, every family N^kRE of sets of vectors of size $k \geq 1$ can be generated by counter machines equipped with $k + 2$ counters. A counter machine is said to be *partially blind* if it does not use the operation of testing the register for zero. As reported in [14], partially blind counter machines are known to be strictly less powerful than counter machines. We denote by N^kPBC_n, for $n \geq k \geq 1$, the families of sets of vectors of size k generated by partially blind counter machines with at most n counters but exactly k output counters.

3 QS P systems

Here, we introduce a class of P systems, called *QS P Systems*, which are inspired by QS mechanisms occurring in bacterial colonies. In this respect, we abstract from a number of underlying biological mechanisms by focusing only on the process of consuming/producing signals, which can then be moved from a cell to another one by means of diffusion through an environment. More precisely, QS P systems are defined as a variant of the notion of population P systems introduced in [4], by considering a colony of bacteria as being a population of cells interacting with each other by means of a common shared environment partitioned into different sub-regions. Such a multi-environment model is motivated by an observation made in [35] about the existence of a "barrier" on the diffusion of signal molecules, which tend to remain confined to some particular micro-habitat. This feature allows QS to convey information

about the physiological state of spatially separated sub-populations. Moreover, a multi-environment model can capture the idea of an environment as a non-homogeneous medium of communication characterised by bio-chemical properties that vary from one sub-region to another one.

Thus QS P systems are defined by assuming the following behaviour of the cells composing the population and the local environments:

- a cell can store a *natural number* $s \geq 0$, which can become arbitrarily large and which represents the number of signal molecules currently present inside the cell; we are here only interested in the "concentration" of the signal molecules and how they diffuse between a cell and the environment; this is why we do not distinguish between different signal molecules but we assume all these signal molecules to be of the same sort and we just count them by means of the value s;
- a cell can store at most one symbol from a given alphabet Q, which represents the particular physiological state of the cell; we in fact call the alphabet Q the *set of states*;
- a cell, depending on the value s, can change its state and simultaneously consume some signal molecules in order to produce some new ones inside the cell and/or release more signal molecules into a given environment; this operation is meant to model the response of the cell to variations on the concentration of signal molecules inside, and it is modelled in terms of consumption/production/release of signal molecules;
- an environment can store a *natural number* $t \geq 1$, which cannot exceed a given capacity, and which represents the number of signal molecules currently present in that environment; environments are in fact considered as being passive repositories for signal molecules that act as finite buffers for communication between cells;
- finite amounts of signal molecules can diffuse from an environment to a certain cell depending on the values s and t.

Therefore, in QS P systems, cells can interact only through some finite environment components by exchanging numerical values representing finite amounts of signal molecules. Formally, a QS P system is introduced by the following definition.

Definition 2 *A QS P system is a construct*

$$\mathcal{P} = (Q, C_1, C_2, \ldots, C_n, E_1, E_2, \ldots, E_m, R, O)$$

where:

(1) Q *is a finite set of states;*
(2) $C_i = (q_i, s_i)$*, for each* $1 \leq i \leq n$*, is a cell with* $q_i \in Q$ *the initial state of the cell, and* $s_i \geq 0$ *the initial value stored by the cell;*

(3) $E_i = (c_i, t_i)$, *for each* $1 \leq i \leq$, *is an environment with* $c_i \geq 0$ *the capacity of the environment, and* $0 \leq t_i \leq c_i$ *the initial value stored by the environment;*

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

(a) $(q, s)_i \rightarrow (q', s')_i$, *with* $1 \leq i \leq n$, $q, q' \in Q$ *and* $s, s' \geq 0$ *(*transformation rules*),*

(b) $(q, s)_i[]_j \rightarrow (q', s')_i[s'']_j$, *with* $1 \leq i \leq n$, $1 \leq j \leq m$, $q, q' \in Q$, $s, s', s'' \geq 0$, *and* $s'' \leq c_j$ *(*releasing rules*),*

(c) $(s)_i[t]_j \rightarrow (s')_i[t']_j$, *with* $1 \leq i \leq n$, $1 \leq j \leq m$, $s, s' \geq 0$, $c_j \geq t, t' \geq 0$ *and* $s + t = s' + t'$ *(*diffusion rules*);*

(5) $O = (o_1, \ldots, o_k)$ *with* $o_i \in \{1, 2, \ldots, n\}$, *for all* $1 \leq i \leq k$, *is an ordered tuple of output cells.*

Therefore, a QS P system is defined as a collection of $n \geq 1$ cells and $m \geq 1$ environments. A cell C_i, with $1 \leq i \leq n$, is characterised by a state, initially q_i, (the configuration of chemical elements belonging to C_i) and by a natural number, initially s_i, (the number of signal molecules present inside C_i). An environment E_j, $1 \leq j \leq m$, can instead store only a natural number, initially equal to t_j, which can never exceed the capacity of the environment given by c_j. In the rest of the paper, we will refer to cell C_i and environment E_j by just using the expressions "cell i" and "environment j".

A finite set of rules R is then considered which can contain three different types of rules: transformation rules, releasing rules and diffusion rules. A transformation rule of the form $(q, s)_i \rightarrow (q', s')_i$, with $1 \leq i \leq n$, $q, q' \in Q$ and $s, s' \geq 0$, specifies that cell i can change its state from q to q' by consuming s occurrences of signal molecules and producing s'. This corresponds to subtracting s from the value currently stored in cell i and adding s' to the same value. Therefore, such a rule can be applied inside cell i if and only if this cell contains at least s signal molecules. A releasing rule of the form $(q, s)_i[]_j \rightarrow (q', s')_i[s'']_j$, with $1 \leq i \leq n$, $1 \leq j \leq m$, $q, q' \in Q$, $s, s', s'' \geq 0$ and $s'' \leq c_j$, specifies that a cell i can change its state from state q to state q' by consuming s occurrences of the signal molecule and producing s' inside cell i and s'' occurrences of the signal molecule in the environment j. Such a rule can be applied if and only if, cell i contains at least s signal molecules. A diffusion rule of the form $(s)_i[t]_j \rightarrow (s')_i[t']_j$, with $1 \leq i \leq n$, $1 \leq j \leq m$, $s, s' \geq 0$, $c_j \geq t, t' \geq 0$ and $s + t = s' + t'$, specifies that if $s \geq s'$ ($s \leq s'$) then $s - s'$ ($s' - s$) occurrences of the signal molecule can be moved from cell i to environment j (environment j to cell i) provided that cell i contains at least s signal molecules and environment j contains at least t signal molecules.

Notice that we adopt the convention of representing a cell as a pair of round parentheses and an environment as a pair of square brackets. The set of rules in R implicitly defines the structure of the system by determining the connections that exist between the cells and the environments in the system.

Specifically we say that an environment j, with $1 \leq j \leq m$, is *local* to (or contains) a cell i, with $1 \leq i \leq n$, if and only if there is a rule in R containing on its left-side the pair cell i, environment j, which is either a releasing rule or a diffusion rule; if that is the case then, we also say that cell i is *connected* to environment j. Thus, in our model, the same cell can be connected to many different environments, but there cannot be any direct connection neither between two different cells nor between two different environments. Specifically, each environment can be used as a buffer of communication to convey information between the cells connected to that environment, whereas each cell can be used as a buffer of communication to convey information between the environments which are local to that cell.

Let $\mathcal{P} = (Q, C_1, C_2, \ldots, C_n, E_1, E_2, \ldots, E_m, R, O)$ be a QS P system as specified in Definition 2; a configuration of $\mathcal{P}$, at moment τ is a tuple:

$$\Sigma^\tau = \langle C_1^\tau, C_2^\tau, \ldots, C_n^\tau, E_1^\tau, E_2^\tau, \ldots, E_m^\tau \rangle$$

where, for each $1 \leq i \leq n$, we have $C_i^\tau = (q_i^\tau, s_i^\tau)$, for some $q_i^\tau \in Q$ and $s_i^\tau \geq 0$, and, for each $1 \leq j \leq m$, $E_j^\tau = (c_j, t_j^\tau)$, for some $c_j \geq s_j^\tau \geq 0$. The initial configuration of $\mathcal{P}$ is

$$\Sigma^0 = \langle C_1^0, C_2^0, \ldots, C_n^0, E_1^0, E_2^0, \ldots, E_m^0 \rangle$$

where $C_i^0 = C_i$, $1 \leq i \leq n$ and $E_j^0 = E_j$, $1 \leq j \leq m$. A QS P system operates in a parallel mode. Specifically, given two configurations $\Sigma^\tau, \Sigma^{\tau+1}$ of the QS P system $\mathcal{P}$, we say that $\mathcal{P}$ goes, in one step, from configuration Σ^τ to configuration $\Sigma^{\tau+1}$ in a parallel way, and we write $\Sigma^\tau \Longrightarrow \Sigma^{\tau+1}$, if and only if, $\Sigma^{\tau+1}$ is obtained from Σ^τ by applying in parallel a rule, non deterministically chosen, per each cell in the system. That is:

- all the cells that can evolve by means of at least one rule must evolve in parallel at the same time within the same transition step;
- at most one rule per each cell can be used to modify the content of a cell within the same transition step;
- the content of an environment j, with $1 \leq j \leq h$, can be modified by different releasing and diffusion rules, which may be applied in parallel to different cells connected to the environment j; these rules must be selected such as to make sure that, after their use, the value stored by this environment will be at least zero and no larger than the capacity c_j.

Thus, according to this parallel mode, different cells can simultaneously operate on the same local environment within the same transition step, but each one of them can be involved in the application of at most one rule at a time, which can be either a transformation rule, or a releasing rule, or a diffusion rule.

A computation is then defined as being a sequence of transitions. Such a computation is said to be successful if it reaches a configuration Σ^f such that:

$$\Sigma^f = \langle (q_1^f, s_1^f)_1^f, \ldots, (q_n^f, s_n^f)_n^f, E_1^f, E_2^f, \ldots, E_m^f \rangle$$

and there are no more rules which can be applied to any cell i, with $1 \leq i \leq n$, or to any environment j, with $1 \leq j \leq m$. If $O = (o_1, \ldots, o_k)$, then the result of such a computation is the vector $(s_{o_1}^f, \ldots, s_{o_k}^f)$ where, for all $1 \leq i \leq k$, $s_{o_i}^f$ is the value stored inside the output cell o_i in the configuration Σ_f. The set of all vectors of size k generated by the QS P system $\mathcal{P}$ is denoted by $N^k(\mathcal{P})$.

4 Main Results

In this section, we investigate the computational power of QS P systems. To this aim, we introduce the families of vectors $N^kQSP_{n,m}(\delta, \sigma)$, with $n \geq k \geq 0$, $m \geq 0$,$\delta \geq 0$, $\sigma \geq 0$, generated by QS P systems with at most n cells, but exactly k output cells, and at most m environments, where the capacity of every environment is at most δ and the cardinality of the set of states is at most σ. As usual, these values are replaced by $*$ whenever they are not bounded.

The first obvious result concerns the power of QS P systems where there is one single cell and there is no environment to interact with.

Lemma 3 *$N^1QSP_{1,0}(0, *) = NREG$.*

It is in fact easy to see that QS P systems with one cell and no environments are no more than regular grammars over one-letter alphabets and, vice versa, that every regular grammar over one-letter alphabet can be simulated by a QS P systems with one cell.

Next, QS P systems with a greater number of cells are proved to be equivalent to counter machines. This is done by firstly showing that partially blind counter machines can be simulated by QS P systems consisting of a population of cells which interact with each other by means of a single environment. Specifically, for a partially blind counter machine with n counters it is shown that the number of cells in the population is $n + 2$ and the environment's capacity can be bounded by $3n + 2$.

Theorem 4 *$N^kPBC_n \subseteq N^kQSP_{n+2,1}(3n + 2, *)$, for all $n \geq k \geq 2$.*

PROOF. Let $M = (Q, F, p_0, A, O, I)$ be a partially blind counter machine as specified in Section 2 where $A = \{c_1, c_2, \ldots, c_n\}$ and $O = (c_{o_1} \ldots, c_{o_k})$ for

some $n \geq k \geq 1$. In order to simulate such a counter machine, we define a QS P system $\mathcal{P}$ such that

$$\mathcal{P} = (Q', C_1, C_2, \ldots, C_n, C_{n+1}, C_{n+2}, E_1, R, O)$$

where:

$$\begin{aligned}
Q' &= Q \cup \{ q_i \mid 3 \leq i \leq 2n+2 \} \cup \{\#, \$\}, \\
C_i &= (\$, 0), \text{ for all } 1 \leq i \leq n, \\
C_{n+1} &= (p_0, 0), \\
C_{n+2} &= (\$, 0), \\
E_1 &= (2n+2, 0), \\
R &= \{(p,0)_{n+1}[]_1 \to (q_{h_i^+}, 0)_{n+1}[h_i^+]_1 \mid (p \to q, +c_i) \in I, 1 \leq i \leq n \} \\
&\quad \cup \{(p,0)_{n+1}[]_1 \to (q_{h_i^-}, 0)_{n+1}[h_i^-]_1 \mid (p \to q, -c_i) \in I, 1 \leq i \leq n \} \\
&\quad \cup \{(p,0)_{n+1} \to (q,0)_{n+1} \mid (p \to q, \epsilon) \in I, 1 \leq i \leq n \} \\
&\quad \cup \{(0)_i[h_i^+]_1 \to (1)_i[h_i^+ - 1]_1 \mid 1 \leq i \leq n \} \\
&\quad \cup \{(1)_i[h_i^-]_1 \to (0)_i[h_i^- + 1]_1 \mid 1 \leq i \leq n \} \\
&\quad \cup \{(0)_{n+1}[h_i^+ - 1]_1 \to (h_i^+ - 1)_{n+1}[0]_1 \mid 1 \leq i \leq n \} \\
&\quad \cup \{(0)_{n+1}[h_i^- + 1]_1 \to (h_i^- + 1)_{n+1}[0]_1 \mid 1 \leq i \leq n \} \\
&\quad \cup \{(0)_{n+2}[1]_1 \to (1)_{n+2}[0]_1, (\$, 1)_{n+2} \to (\#, 0)_{n+2}\} \\
&\quad \cup \{(q_{h_i^+}, h_i^+ - 1)_{n+1} \to (q, 0)_{n+1} \mid q \in Q, 1 \leq i \leq n\} \\
&\quad \cup \{(q_{h_i^-}, h_i^- + 1)_{n+1} \to (q, 0)_{n+1} \mid q \in Q, 1 \leq i \leq n\} \\
&\quad \cup \{(q_{h_i^+}, s)_{n+1} \to (\#, 0)_{n+1} \mid q \in Q, 1 \leq i \leq n, 1 \leq s < h_i^+ - 1\} \\
&\quad \cup \{(q_{h_i^-}, s)_{n+1} \to (\#, 0)_{n+1} \mid q \in Q, 1 \leq i \leq n, 1 \leq s < h_i^- + 1\} \\
&\quad \cup \{(\#, 0)_i \to (\#, 0)_i \mid n+1 \leq i \leq n+2\}, \\
O &= (o_1, \ldots, o_k),
\end{aligned}$$

with $h_i^+ = 2i + 1$, for each $1 \leq i \leq n$ and $h_i^- = 2n + 1 + i$, for each $1 \leq i \leq n$; h_i^+ and h_i^- are built such that $h_1^+ < h_2^+ < \ldots h_n^+ < h_1^- < h_2^- < \ldots h_n^-$.

Thus, the QS P system $\mathcal{P}$ simulates the counter machine M by using cell $n+1$ to control the correct execution of a sequence of instructions from I. Cell $n+1$ contains in fact the state of the machine M and it produces in the environment either a value h_i^+ or a value h_i^- in order to either increase or decrease the value of register i, for some $1 \leq i \leq n$, which is kept inside cell i.

Specifically, let $(p \to q, +c_i)$ be an instruction from I and let p be the current state of the machine M. The state p is supposed to be contained inside cell $n+1$ where, in order to simulate the aforementioned instruction, we produce the value h_i^+, which is immediately sent to the environment, and we introduce the state $q_{h_i^+}$. Next, we can apply the rule $(0)_i[h_i^+]_1 \to (1)_i[h_i^+ - 1]_1$, which

increases by 1 the value stored inside cell i corresponding to the value of register c_i. Then, the value $h_i^+ - 1$ can diffuse back into cell $n+1$ where we can use the rule $(q_{h_i^+}, h_i^+ - 1)_{n+1} \rightarrow (q, 0)_{n+1}$ to complete the simulation of the current instruction. Let us consider that h_i^+, $h_i^+ - 1$ are taken by wrong targets. If h_i^+ is taken by some cells j, $j < i$ then the only way to finish the computation is to iteratively transform it into 0; otherwise cells $n+1$ and/or $n+2$ will generate an endless computation. We will show that this chain of changes transforming h_i^+ into 0 is not possible. Similarly when $h_i^+ - 1$ is left into the environment by cell i it may be also taken (shared) by some cells j, $j < i$ and we face the same problem of transforming it into 0. Let us denote by H_i either h_i^+ or $h_i^+ - 1$. If H_i is taken by cells j_k, $j_k < i$, $1 \leq k \leq t$, i.e. $H_i = h_{j_1}^+ + \ldots h_{j_t}^+$, then they return back into the environment $H_i - t$ which is then taken by other cells j until the value will decrease to 0. In this way we obtain a decreasing sequence of positive values

$$H_{i,p} > H_{i,p-1} > \ldots > H_{i,1} > 0$$

where $H_{i,p} = H_i$ and $H_{i,1}$ is the last value before 0. This means $H_{i,1}$ is taken by cells j_k, $1 \leq k \leq s$, i.e. $H_{i,1} = h_{j_1}^+ + \ldots h_{j_s}^+$ and

$$h_{j_1}^+ - 1 + \ldots h_{j_s}^+ - 1 = 0$$

and this means $h_{j_k}^+ = 1$, $1 \leq k \leq s$ which contradicts the definition of h_i^+, $1 \leq i \leq n$. So h_i^+, $h_i^+ - 1$ can not be exhausted by cells j, $j < i$.

Now, let $(p \rightarrow q, -c_i)$ be an instruction from I and let p be the current state of the machine M. The state p is supposed to be contained inside cell $n+1$ where, in order to simulate the aforementioned instruction, we produce the value h_i^-, which is immediately sent to the environment, and we introduce the state $q_{h_i^-}$. Next, if the current value stored inside cell i is greater than 0, we can use the rule $(1)_i[h_i^-]_1 \rightarrow (0)_i[h_i^- + 1]_1$ in order to decrease by 1 the value stored inside cell i, which corresponds to the current value of the register i. Then, the value $h_i^- + 1$ can diffuse back into cell $n+1$ where we can use the rule $(q_{h_i^-}, h_i^- + 1)_{n+1} \rightarrow (q, 0)_{n+1}$ to complete the simulation of the current instruction. If h_i^-, $h_i^- + 1$ are taken by wrong targets then either the computation will never stop or, similar to h_i^+, $h_i^+ - 1$, it may be shown that these values can not be consumed by cells j, $j < i$.

An instruction $(p \rightarrow q, \epsilon)$ is instead simulated by just applying the rule $(p, 0)_{n+1} \rightarrow (q, 0)_{n+1}$, which simply introduces the new state inside cell $n+1$.

Finally, when a final state is reached, no further rules can be applied inside cell $n+1$ and the computation halts by having correctly simulated a sequence of instructions from i. $\square$

This proof essentially shows how to simulate the operation of increasing and decreasing the registers of a counter machine by means of a population of cells interacting with each other by means of an environment with a bounded capacity. We do not know whether these systems can also simulate the operation of comparing to 0 or not. However, as shown by the next result, this latter operation can be simulated by adding some extra cells and some extra environments. In particular, as we know that the family N^kRE, for any $k \geq 1$, can be generated by machines with $k+2$ counters, we can provide limits on the number of cells and on the number of environments necessary to generate each family N^kRE by means of QS P systems.

Theorem 5 *$N^kQSP_{2k+6,k+3}(3k+8,*) = N^kRE$, for all $k \geq 1$.*

PROOF. Let $r = k+2$, and let $M = (Q, F, p_0, A, c_{out}, O)$ be a counter machine as specified in Section 2 where $A = \{c_1, \ldots, c_r\}$ and $O = (c_{o_1} \ldots, c_{o_k})$. From Theorem 4, we know how to simulate all the instructions from I but those comparing the value of a register to 0. In the case of a counter machine with r registers, this can be done by constructing a QS P system $\mathcal{P}$ such that:

$$\mathcal{P} = (Q', C_1, \ldots, C_r, C_{r+1}, C_{r+2}, E_1, R, O)$$

with Q', $C_1, \ldots, C_r, C_{r+1}, C_{r+2}$, R, O as in the proof of Theorem 4, and $E_1 = (3r+2, 0)$. Here we embed the structure of the QS P system $\mathcal{P}$ in another QS P system $\mathcal{P}'$, which contains all the rules necessary for the simulation of the operation of comparing the value of a register with 0. Specifically, we consider the following P system:

$$\mathcal{P}' = (Q'', C_1, \ldots, C_r, C_{r+1}, C_{r+2}, C_{r+3}, \ldots, C_{2r+2}, E_1, E_2, \ldots, E_{r+1}, R', O)$$

where:

$$\begin{aligned}
Q'' &= Q' \cup \{\, q^{(i)} \mid q \in Q, 1 \leq i \leq 4 \,\}, \\
C_{i+r+2} &= (\$, 0), \text{ for all } 1 \leq i \leq r, \\
E_{i+1} &= (2, 0), \text{ for all } 1 \leq i \leq r, \\
R' &= R \\
&\quad \cup \{(p,0)_{r+1}[]_{i+1} \to (q^{(1)},0)_{r+1}[1]_{i+1} \mid (p \to q, c_i = 0) \in I, 1 \leq i \leq r\,\} \\
&\quad \cup \{(q^{(1)},0)_{r+1} \to (q^{(2)},0)_{r+1}, (q^{(2)},0)_{r+1} \to (q^{(3)},0)_{r+1} \mid q \in Q\,\} \\
&\quad \cup \{(q^{(3)},0)_{r+1} \to (q^{(4)},1)_{r+1}, (q^{(4)},2)_{r+1} \to (q,0)_{r+1} \mid q \in Q\} \\
&\quad \cup \{(1)_i[1]_{i+1} \to (0)_i[2]_{i+1} \mid 1 \leq i \leq r\,\} \\
&\quad \cup \{(0)_{i+r+2}[2]_{i+1} \to (2)_{i+r+2}[0]_{i+1} \mid 1 \leq i \leq r\,\} \\
&\quad \cup \{(1)_{r+1}[1]_{i+1} \to (2)_{r+1}[0]_{i+1} \mid 1 \leq i \leq r\,\} \\
&\quad \cup \{(\$,2)_{i+r+2} \to (\#,0)_{i+r+2}, (\#,0)_{i+r+2} \to (\#,0)_{i+r+2} \mid 1 \leq i \leq r\,\}.
\end{aligned}$$

The QS P system $\mathcal{P}'$ simulates the instructions $(p \to q, +c_i)$, $(p \to q, -c_i)$, $(p \to q, \epsilon)$ from I in the same way as $\mathcal{P}$ by using the rules in R. In order to simulate an instruction $(p \to q, c_i = 0)$, we have instead to apply a rule $(p, 0)_{r+1}[]_{i+1} \to (q^{(1)}, 0)_{r+1}[1]_{i+1}$, which introduces the value 1 in the environment $i+1$, $1 \leq i \leq r$. Now, if the value stored in cell i is equal to 0 then, no rule can be applied to the value 1 stored in the environment $i+1$ until the state $q^{(4)}$ is produced inside cell $r+1$ together with the value 1. At that point, we can diffuse back into cell $r+1$ the value 1 contained in the environment $i+1$. Eventually the new state q is produced by using the rule $(q^{(4)}, 2)_{r+1} \to (q, 0)_{r+1}$. Otherwise, if the value stored by cell i is greater than 0 then, we are forced to apply the rule $(1)_i[1]_{i+1} \to (0)_i[2]_{i+1}$, which adds 1 to the current content of the environment $i+1$. Next, since the capacity of the environment $i+1$ is 2, the only applicable rule is $(0)_{i+r+2}[2]_{i+1} \to (2)_{i+r+2}[0]_{i+1}$ because the state $q^{(4)}$ has not yet been produced inside cell 4. Thus, an infinite computation is generated in the presence of the rules $(\$, 2)_{i+r+2} \to (\#, 0)_{i+r+2}$, $(\#, 0)_{i+r+2} \to (\#, 0)_{i+r+2}$, $1 \leq i \leq r$. A computation in $\mathcal{P}'$ halts whenever a final state is produced inside cell $r+1$; this corresponds to having correctly simulated a sequence of instructions by the machine M.

Thus, since $r = k+2$, the QS P system $\mathcal{P}'$ has $2r+2 = 2k+6$ cells and $r+1 = k+3$ environments; the capacity of every environment is bounded by the value $3r+2 = 3k+8$. □

The complexity parameters considered in the above proof may be improved by finding a better codification of the values h_i^+, h_i^-. The overall capacity of the environment can be improved if instead of using one environment for all h_i^+, h_i^- we use two different environments, one for h_i^+ values and another one for h_i^- values. In the particular case of $k = 1$, which is commonly investigated in the area of membrane computing by considering P systems as systems generating sets of natural numbers (e.g., see [26]), we obtain the following values for the number of cells, number of environments, and maximal capacity of an environment.

Corollary 6 $NQSP_{8,4}(11, *) = NRE$.

The proofs above show how to simulate register machines by using QS P systems where the cells coordinate each other by exchanging finite amount of signal molecules. In particular, the "complexity" of such cell-to-cell communication can be bounded by the maximal capacity of the environments. Moreover, the proofs show how different cells can become active at different times and how they communicate with each other. In this respect, it is interesting to point out that such cells become active only when the concentration of signal molecules inside exceeds a certain value thus mimicking the natural quorum sensing mechanism occurring in bacterial colonies.

5 Vibrio Fischeri QS model

The QS process is a very general and widely spread communication mechanism in bacteria colonies. In this section we will investigate it for the marine bacterium Vibrio fischeri.

The QS mechanism in Vibrio fischeri relies on the synthesis, accumulation and subsequent sensing of a signal molecule, 3-oxo-C6-HSL that will be denoted as *OHHL*. This signal molecule, *OHHL*, is synthesised by the protein *LuxI* and sensed by the protein *LuxR*. When a small number of bacteria is present in a given environment these proteins and the signal molecule are produced at a basal rate (low density). *OHHL* diffuses out of the bacterial cells and spreads into the surrounding environment where it may accumulate. When the number of cells increases, the signal accumulates in the environment at a high density and can also diffuse into bacterial cells. The signal is able to interact with the *LuxR* protein to form the complex *LuxR-OHHL*. This complex binds to a region of DNA called the *lux* box causing the transcription of some genes. Consequently more *OHHL* is produced. Bacteria sense their cell density by measuring the amount of signal present.

5.1 Modelling QS in Vibrio Fischeri

Our model of QS in Vibrio fischeri is developed using QS P systems and considers, apart from the above mentioned characteristics identified by a previous model based on population P systems [29], a direct codification of the chemical concentration of signal molecules and multi-environments. These are important features of the QS mechanisms occurring in bacterial colonies as the "shape and degree of enclosure of the culture" is potentially a factor influencing the production of autoinducer concentration [18].

The role of the multi-environment is significant in modelling different colonies working simultaneously in different states and on various parts of a host or in simulating processes of chemical degradation.

The rules (abstraction of the chemical reactions) have the following forms:

- *evolution rules* $(u, m)_i \rightarrow (v, n)_i$ where u, v, are multisets of objects and m, n are chemical concentrations in a bacterium cell i;
- *sending into an environment compartment* $(m)_i \rightarrow (m-p)_i[p]_j$ where m is an initial chemical concentration in a bacterium cell i and p is a chemical concentration released into the environment component j and $m-p$ the concentration left into i;
- *bringing from an environment component* $(m)_i[p]_j \rightarrow (m+s)_i[p-s]_j$ where

m, p are initial chemical concentrations in a bacterium cell i and in the component environment j, respectively, and s a chemical concentration brought from the component j into the cell i.

In what follows we specify the evolution rules in the bacteria and the communication rules between bacteria and environment components.

Initially, a bacterium produces the signal *OHHL* at a basal rate. The number of signal molecules inside is less than a threshold we will denote by T. The rule will be

$(LuxBox,\, m)_b \rightarrow (LuxBox,\, m+1)_b$

where $m < T$; this shows an increase in signal molecules.

The protein *LuxR* acts as a receptor and *OHHL* as its ligand. They can form a complex, denoted *LuxR-OHHL* which in turn can dissociate back into its components.

$(LuxR\ OHHL,\, m)_b \rightarrow (LuxR\text{-}OHHL,\, m)_b,$
$(LuxR\text{-}OHHL,\, m)_b \rightarrow (LuxR\ OHHL,\, m)_b.$

The complex *LuxR-OHHL* acts as a transcription factor or as a promotor binding to a region of the bacterium *DNA* called *LuxBox* and starting the transcription of different proteins. The process is also reversible.

$(LuxBox\ LuxR\text{-}OHHL,\, m)_b \rightarrow (LuxBox\text{-}LuxR\text{-}OHHL,\, m)_b,$
$(LuxBox\text{-}LuxR\text{-}OHHL,\, m)_b \rightarrow (LuxBox\ LuxR\text{-}OHHL,\, m)_b.$

The binding of the complex *LuxR-OHHL* to the *LuxBox* produces a massive increase of the production of the signal *OHHL*.

$(LuxBox\text{-}LuxR\text{-}OHHL,\, T)_b \rightarrow (LuxBox\ LuxR\text{-}OHHL,\, T+p)_b, p > 1.$

OHHL is a small molecule that diffuses outside the bacterium and so it can accumulate in the environment.

$(m)_b \rightarrow (m-p)_b[p]_e,$

where $p = 1$ when $m < T$ and $p > 1$ otherwise.

The signal molecule *OHHL* undergoes a process of degradation in the bacterium.

$(m)_b \rightarrow (m-1)_b.$

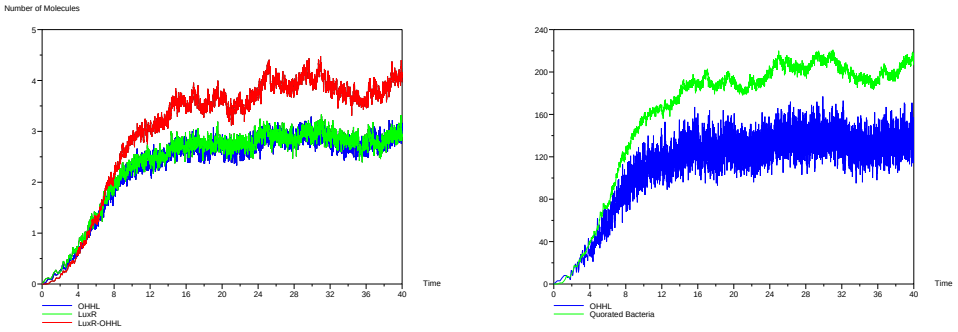
When the signal molecule accumulates in the environment it can diffuse inside

the bacteria.

$(m)_b[p]_e \rightarrow (m+1)_b[p-1]_e$.

The model may be simulated in different ways; either in a nondeterministic way as most of the P systems do or by associating some kinetic rates to the rules involved that are proportional to the number of molecules. In the latter case we will get the model given in [29].

In the current case study we have considered, similar to [29], the behaviour of a system with a population of 300 bacteria. The signal molecules *OHHL* are traced down both in the environment Fig. 2 (b) and inside the colony Fig. 2 (a) (an average concentration across the colony has been considered) [29].



(a) (b)

Fig. 2. (a) Evolution of bacteria. (b) Evolution of the environment

It may be observed that the signal, *OHHL*, accumulates in the environment until saturation and then, when the threshold is reached, bacteria are able to detect that the size of the population is big enough. Initially, a few bacteria are quorated and then the process of recruitment accelerates and the colony behaves in a coordinated way. It may be noted that there is a correlation between the number of signals in the environment and the number of quorated bacteria as well as between the concentration of signal molecules in the environment and inside the bacteria (as it shows the average concentration across the colony). There is also a strong correlation between the signal *OHHL*, the protein *LuxR* and the complex *LuxR-OHHL*.

These results are obtained by using both the P systems approach in [29] and the current approach based on QS P systems. In the current framework we can also model local interactions, by considering local environments connected to a part of the colony, movement of the bacterium cells, as well as considering different properties for the environment components like permeability, degradation rate etc.

6 Conclusions

The model investigated in this paper, called QS P systems, draws its inspiration from the natural quorum sensing mechanism in bacterial colonies. It is proved that the model is powerful enough to simulate counter machines.

The model is not "another variant" of a computational paradigm - in this case P systems, as it draws its roots from a very well-studied biological mechanism and it abstracts in some of the proofs the very specific behaviour of the natural quorum sensing mechanism. More than this, we return back to biology and show that this model may be used to specify the behaviour of bacterial colonies. A tool has been built that helps us to simulate such models [29].

A number of questions remain open after this initial attempt to introduce the model. It is almost certain that the complexity values associated to the parameters introduced in our proofs - number of cells and environments, and capacities, are not optimal. So, what about the optimal results? In our definitions we have made use of some "contexts" - number of signal molecules existing simultaneously in some cells and environments. To what extent is it possible to get rid of them? All the connections established between cells and environments are fixed. It will be of interest to consider models with a dynamic structure of the system in the sense of [4].

More challenging and interesting remain the questions related to the model's potential in specifying biological mechanisms. How suitable is it to thoroughly model natural quorum sensing? How to make it available to biologists, as it claims that it uses inspiration from nature? What else do we have to add to it to make it useful and powerful as a modelling paradigm?

Acknowledgements. The authors are indebted to Grzegorz Rozenberg and Gheorghe Păun for the enlightening discussions we have had in Leiden where the multi-environment concept associated with (population) P systems has been initially identified and to the anonymous referees for their comments that allowed us to better reformulate Theorems 3 and 4. The research of Francesco Bernardini and Marian Gheorghe has been supported by the Engineering and Physical Science Research Council (EPSRC) of United Kingdom, Grant GR/R84221/01. Natalio Krasnogor acknowledges EPSRC for the project EP/D021847/1.

References

[1] B. Alberts, A. Johnson, J. Lewis, M. Raff, K. Roberts, P. Walter, The Molecular Biology of the Cell, 4th edition, Garland Publ. Inc., London, 2002.

[2] J.A. Anderson, An Introduction to Neural Networks, The MIT Press, Cambridge, MA, 1996.

[3] O. Andrei, G. Ciobanu, D. Lucanu, Structural Operational Semantics of P Systems, in: Proceedings of WMC 2005, Vienna, Austria, 1 – 23.

[4] F. Bernardini, M. Gheorghe, Population P systems, Journal of Universal Computer Science 10 (5) (2004) 509–539.

[5] F. Bernardini, V. Manca, P Systems with Boundary Rules, in: Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron (Eds.), Lecture Notes in Computer Science 2597, 2003, 107 – 118.

[6] F. Bernardini, F.J. Romero-Campero, M. Gheorghe, M.J. Pérez-Jiménez, M. Margenstern, S. Verlan, N. Krasnogor, On P Systems with Bounded Parallelism, Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, IEEE Computer Society, Timisoara, 25-29 September 2005, 399 – 406.

[7] Th. Bäck, D. B. Fogel, Z. Michalewicz (Eds.), Handbook of Evolutionary Computation, Oxford University Press, New York, and Institute of Physics Publishing, Bristol, 1997.

[8] E. Csuhaj-Varju, J. Dassow, J. Kelemen, Gh. Păun, Grammar Systems. A Grammatical Approach to Distribution and Cooperation, Gordon and Breach, London, 1994.

[9] E. Csuhaj-Varju, J. Kelemen, A. Kelemenova, Gh. Păun, Eco-Grammar Systems: A Grammatical Framework for Studying Life-Like Interactions, Artificial Life 3 (1997) 1 – 28.

[10] G. Ciobanu, Gh. Păun, M.J. Pérez-Jiménez (Eds.), Applications of Membrane Computing, Springer–Verlag, Berlin, 2004.

[11] A. Ehrenfeucht, T. Harju, I. Petre, D.M. Prescott, G. Rozenberg, Computations in Living Cells, Springer–Verlag, Berlin, 2004.

[12] M. Fischer, R. Paton, K. Matsuno, Intracellular Signalling Proteins as Smart Agents in Parallel Distributed Processes, BioSystems 50 (3) (1999) 159 – 171.

[13] R. Freund, M. Oswald, P Colonies Working in the Maximally Parallel and in the Sequential Mode, from [36].

[14] P., Frisco, J.H., Hoogeboom, Simulating Counter Automata by P Systems with Symport/Antiport, in: Gh. Păun, G. Rozenberg, A. Salomaa, (Eds.), in: Membrane Computing, International Workshop. WMC-CdeA 2002, Curtea de Arges, Romania, August 2002. Revised Papers in Lecture Notes in Computer Science 2597, 2003, 288 – 301.

[15] J. Hopcroft, J. Ulmann, Introduction to Automata Theory Languages, and Computation, Addison-Wesley, 1979.

[16] D. Jackson, M. Gheorghe, M. Holcombe, F. Bernardini, An Agent-Based Behavioural Model of Monomorium Pharaonis Colonies, in: C. Martín-Vide, G. Mauri, Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Membrane Computing, International Workshop, WMC 2003, Tarragona, Spain. Revised Papers in Lecture Notes in Computer Science 2933, 2004, 232 – 240.

[17] D. Jackson, M. Holcombe, F. Ratnieks, Coupled Computational Simulation and Empirical Research into the Foraging System of Pharao's ant (Monomorium Pharaonis), BioSystems 76 (1–3) (2004) 101 – 112.

[18] S. James, P. Nilsson, G. James, S. Kjelleberg, T. Fagerström, Luminescence Control in the Marine Bacterium Vibrio fischeri: An Analysis of the Dynamics of lux Regulation, J. Mol. Biol. 296 (2000) 1127 – 1137.

[19] N. Kam, I.R. Cohen, D. Harel, The Immune System as a Reactive System: Modelling T Cell Activation with Statecharts, in: Proc. Visual Languages and Formal Methods (VLFM01), Trento, Italy, 2001, 15 – 22.

[20] J. Kelemen, A. Kelemenova, Gh. Păun, On the Power of Biochemically Inspired Simple Computing Model: P Colonies, from [36].

[21] J.H. Koza, J.P. Rice, Genetic Algorithms: The Movie, MIT Press, Cambridge, Mass., 1992.

[22] H. Lodish, A. Berk, S.L. Zipursky, P. Matusudaira, D. Baltimore, J.E. Darnell, Molecular Cell Biology, 4th Edition, W.H. Freeman, New York, 2000.

[23] H. Kitano, Computational Systems Biology, Nature 420 (2002) 206 – 210.

[24] A. Lindenmayer, Mathematical Models for Cellular Interaction in Development, Parts I and II, Journal of Theoretical Biology, 18 (1968) 280 – 315.

[25] H. Matsuno, R. Murakami, R. Yamane, N. Yamasaki, S. Fujita, H. Yoshimori, S. Miyano, Boundary Formation by Notch Signaling in Drosophila Multicellular Systems: Experimental Observations and Gene Network Modelling by Genomic Object Net, in: R.B. Altman, A.K. Dunker, L. Hunter, T.E. Klein (Eds.), Pacific Symp. on Biocomputing, World Scientific Press, Singapore, 2003, 152 – 163.

[26] Gh. Păun, Membrane Computing. An Introduction, Springer–Verlag, Berlin, 2002.

[27] Gh. Păun, M.J. Pérez-Jiménez, Membrane Computing: Brief Introduction. Recent Results and Applications, submitted.

[28] Gh. Păun, G. Rozenberg, A. Salomaa, DNA Computing. New Computing Paradigms, Springer–Verlag, Berlin, 1998.

[29] M.J. Pérez-Jiménez, F.J. Romero-Campero, Modelling Vibrio Fischer's behaviour using P systems, VIIIth European Conference on Artificial Life, Systems Biology Workshop, 5 – 9 september, Kent, United Kingdom.

[30] A. Regev, E.M. Panina, W. Silverman, L. Cardelli, E. Shapiro, Bio-Ambients: An Abstraction for Biological Compartments, Theoretical Computer Science 325 (2004) 141 – 167.

[31] G. Rozenberg, A. Salomaa (Eds.), Handbook of Formal Languages (3 volumes), Springer–Verlag, Berlin, 1997.

[32] D.B. Searls, The language of genes, Nature, 420 (2002) 211 – 217.

[33] G. Terrazas, N. Krasnogor, M. Gheorghe, F. Bernardini, S. Diggle, M. Camara, An Environment Aware P System Model of Quorum Sensing, in: S. Barry Cooper, B. Löwe, L. Torenvliet (Eds.), New Computational Paradigms. First Conf. on Computability in Europe, CiE2005, Amsterdam, Lecture Notes in Computer Science 3536, 2005, 479 – 485.

[34] D. Walker, M. Holcombe, J. Southgate, S. McNeil, R. Smallwood, The Epitheliome: Agent-Based Modelling of the Social Behaviour of Cells, BioSystems 76 (1–3) (2004) 89 – 100.

[35] K. Winzer, K. R. Hardie, P. Williams, Bacterial Cell-to-Cell Communication: Sorry , Can't Talk Now - Gone To Lunch, Current Opinion in Microbiology 5 (2002), 216 – 222.

[36] The P systems webpage http://psystems.disco.inimib.it