A Model of the Quorum Sensing System In Genetically Engineered E.Coli Using Membrane Computing

Afshin Esmaeili, Iman Yazdanbod, Christian Jacob

The University of Calgary

Abstract

Quorum sensing is the way bacteria communicate with each other; they release signaling molecules to their environment and other bacteria receive and recognize the signals. Many species of bacteria use the information obtained to coordinate their gene expression in response to the size of their population, which is known as Quorum Sensing. In this article, we present a novel model of a synthetic Autoinducer-2 signaling system in genetically engineered Escherichia coli (E.coli) bacteria using the recently proposed Membrane Computing (MC) framework. Membrane computing is a branch of natural computing that is inspired by biological membranes structures and functions and is used for modeling features of cells in biological systems. This model allows us to observe the behavior of each individual cell as well as the emergent properties of the whole population. It also enables us to manipulate factors involved in the simulation to understand their effects in individual as well as colony behaviors.

Having defined our model in terms of compartments and interactions rules, any biological system that can be explained in terms of compartments and their corresponding rules can be simulated with this platform. In other words, we have developed a biological language for modelling biological systems using MC framework.

Key words: Quorum Sensing, Membrane Computing, Biological Modeling

1. Quorum Sensing Signaling System in Genetically Engineered E.coli Bacteria

Bacteria communicate with each other using chemical signaling molecules. The process of producing, releasing, detecting, and responding to the signaling molecules in bacteria is referred to as quorum sensing. Quorum sensing enables bacteria to obtain information from their environment and alter their behavior in a population size that is scaled in response to changes the number of bacteria present in their community. Thus, this method of intercellular communication allows populations of bacteria to work together, in contrast to the traditional unicellular view of prokaryotes.

In general, quorum-sensing bacteria produce and release signaling molecules into their environments which are called autoinducers. The concentrations of autoinducers increase in the bacteria environment as the bacteria population density grows. The bacteria detect the accumulation of a minimal threshold concentration of the autoinducer and regulate their gene expression in a population-wide scale in response. As a result, using these signaling systems, all the bacteria in a community alter their behavior at the same time in response to an external signal, and therefore function similar to a multicellular organism. The first time, quorum sensing was observed

in the bioluminescent marine bacterium, Vibro fischeri [3]. Today, studying the signaling mechanisms involved in quorum sensing has led to a new area of research: the knowledge obtained may be used in various clinical and industrial applications.

Many different quorum-sensing systems have been discovered. Some of them are extremely specific for a particular species and some of them are common in all bacteria. In this work, we sought to establish a synthetic autoinducer-2 (AI-2) signaling system taken from its natural counterpart in Vibrio harveyi. AI-2 is produced and released by many species of Gram-negative and Gram-positive bacteria. LuxS synthase is responsible for the production of AI-2. AI-2 is bound to a periplasm protein called LuxP. LuxP is constitutively attached to another membrane-bound kinase protein, LuxQ. The LuxPQ complex interacts as a sensor for AI-2. At low cell density, in the absence of sufficient amounts of AI-2 in the environment, this sensor acts as a kinase and phosphorylates cytoplasmic protein LuxU. Phosphorylated LuxU then acts as a kinase itself and adds a phosphate to DNA-binding response regulator protein LuxO. Lastly, the phosphorylated LuxO would bind to a complex of transcription factor, namely Sigma 54 and RNA-4 promoter (Pqrr4), which signals the expression of GFP protein. GFP emits light, so the bacteria glow.

However, at high cell density, AI-2 accumulates in the environment, eventually enters the periplasmic space of bacteria, and is detected by the LuxPQ complex. Hence, this sensor switches from being kinase to phosphatase, resulting in LuxPQ removing the phosphate group from LuxU. Since LuxU can just act as kinase, it is not able to de-phosphorylate the LuxO. However, the housekeeping phosphotases slowly take away the phosphate off LuxO, which does not bind to Pqrr4 promoter, thus turning off the signal. Without the signal, the bacteria are no longer able to produce GFP and they will turn dark by the degradation of the existing GFP [9].

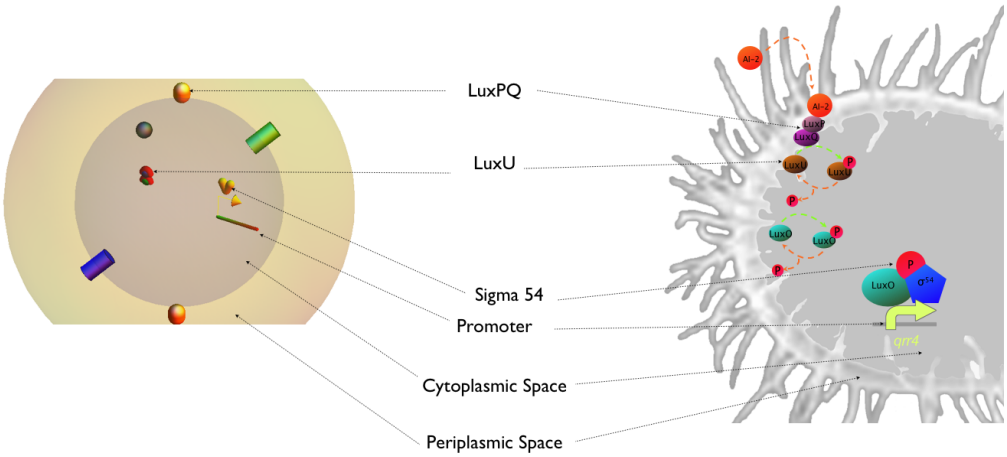

Figure 1: Artistic illustration versus animated visualization. This figure demonstrates the signaling system using two different illustration. AI-2 (the signaling molecule) binds to the LuxPQ complex and therefore this complex changes the phosphorylation cascades within the bacterium which leads to turning off the GFP promoter.

2. Why Modeling?

In this section, we will first explain why computational and mathematical models are useful and effective tools for understanding biology; then, we will narrow down this topic and argue why we believe that the MC approach is an ideal method for modeling the AI-2 signaling system engineered in E.coli bacteria.

During the past two decades, biology and computer science have been converging; many biologists use mathematical and computational models as powerful tools to gain a deeper understanding of biological systems [7]. Given that molecular biology experiments in vitro are very expensive and time consuming, building models of biological processes as a preliminary step helps to circumvent some of the drawbacks of performing hypothesis-testing in the wet lab. This is why we feel that computational modeling is important and useful. Particularly with the extent of synthetic biology, many of the biological systems that are being researched could not be found in nature because they are genetically engineered, so their behaviors are unknown and need to be characterized. For instance, in this project, a synthetic autoinducer-2 (AI-2) signaling system constructed in E.coli is taken from its natural counterpart in Vibrio harveyi, bypassing its small regulatory RNA networks. This engineered biological system shows new behaviors that are not observed in nature and need to be studied and characterized. Based on the reasons given above, using models could provide a faster and cheaper shortcut for biologists to gain a better understanding of the newly engineered system. However, it should be stressed that models, regardless of their accuracy, could not be used as a replacement for vitro experiments; however, they could be used as a preliminary step for characterizing the system and as a shortcut for biologists to gain a better understanding of the newly engineered system.

Emphasizing compartmentalization as a cornerstone feature of cells, membrane computing (MC) is a powerful approach for studying reactions in biological systems. The most important feature of this approach that we would like to emphasize is that it could create a common modeling language that is mathematical and precise and could be understood by biologists. MC allows the user not only to focus on interactions at the level of an individual cell, but also to observe the emergent properties of entire cell populations. The MC approach seems to be ideal for the construction of a quorum-sensing model since compartmentalization of the signal and the cascade proteins are critical features of this process.

3. Membrane Computing (MC)

Membrane computing is a branch of natural computing that was introduced by Gheorghe Paun in 1998 [5]. This new field of computation is rapidly growing and its formalism is used in many research areas. It should be noted that this approach is also used in sciences other than biology such as economics and statistics. For instance, membrane computing is used to solve Boolean satisfiability problems (SAT) and the traveling salesman problem (TSP) in economics [5].

In regard to biological models, MC could be thought of as a framework for devising compartmentalized models that are used to simulate cell functioning. In fact, MC draws inspiration from nature (biology) to provide a more realistic architecture of biological systems in computational simulations by introducing abstract computing devices that are called P systems. A P system is a hierarchal arrangement of membranes. A skin membrane separates the system from its environment. There are also two other kinds of membranes defined in P systems: elementary membranes and non-elementary membranes. Membranes that do not have any membrane inside

are called elementary membranes whereas those membranes that enclose others are referred to as non-elementary membranes. Each membrane defines a region that is indicated by the space that is enclosed by the membrane. For non-elementary membranes, regions are defined as the spaces between the non-elementary membranes and the membranes embedded in them [1, 5].

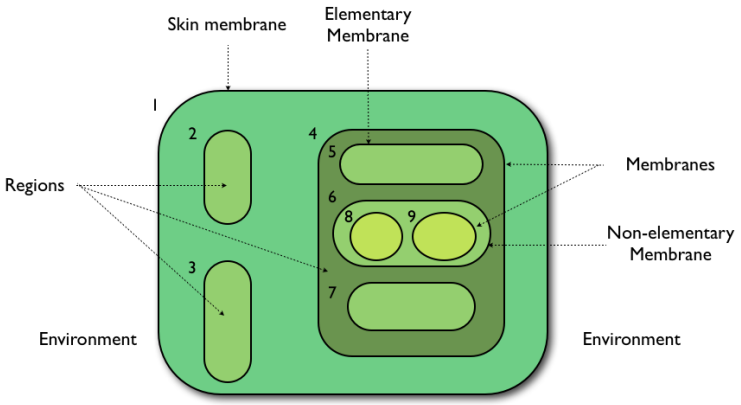

Figure 2: P systems. This figure shows a hierarchal arrangement of 9 membranes in a given P system where membrane 1 is the skin membrane and encloses other membranes from the environment. Membranes 2, 3, 5, 7, 8, and 9 are elementary membranes whereas membranes 4 and 6 are non-elementary due to enclosing other membranes inside.

In P systems, chemicals are modeled by symbols and known as objects. Each region is associated with a multiset of objects that are represented by strings. In addition, each region has a set of rules, which determines how objects are produced, consumed, and transported from one region to the other, and otherwise interact with one another. Labels are used to differentiate between membranes of a P system. Each membrane has its own label, which is a number [5].

3.1. Membrane Computing Formalism

Following is a brief summary of the concepts that have been discussed:

Membrane computing (MC) is a parallel computation approach that processes multi-sets of symbol objects in a localized manner. The membrane structure of a P system is a hierarchical arrangement of membranes embedded in a skin membrane, the one that plays the role of a boundary between the system and the environment. Each membrane encloses a region. Every region is associated with a multi-set of objects, a set of rules, and a label.

The chemicals are assigned to a membrane are referred to as objects. Objects can be represented by a specific multiplicity, which means, at a given time, a given membrane may contain many copies of a given object that is represented by a specific multiplicity. For instance, if B is an object and a given membrane has five copies of it (five Bs are available in the membrane), it is said that the multiplicity of object B is 5 and is indicated by 5B.

These multiplicities may change by the application of rules. The structures of rules are very similar to chemical equations' structures where objects involved in a rule are enclosed in a set of brackets ([]) and the bracket set has a subscript indicating where (which region) the rule's interaction takes place. For instance, the following rule demonstrates the production of object C

from the interaction of objects A and B in region 1 of cell 1:

$$[[A + B]_{region1}]_{cell1} \rightarrow [[C]_{region1}]_{cell1} \quad (1)$$

In this case, region 1 is a membrane that is enclosed by another membrane, cell 1. The reactant objects (A and B) are placed in region 1 and they react together to produce a new object, namely C, which then stays within the membrane in which the reaction took place in (region 1).

There are two important points about rules. First, that all rules are applicable only within the specific region to which they are assigned. In other words, rules have a local scope. For example, the rule in Eq.1 is applicable only to region 1 of cell 1. It means if we have objects A and B in regions other than region 1, this rule can not to be applied to produce object C. This point is shown in figure 3:

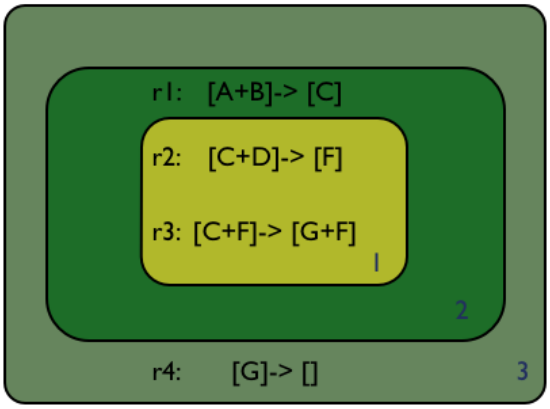

Figure 3: This figure indicates that rules have local scopes. For example, rule one (r1) is assigned to region 2 (indicated by blue label) and rules 2 and 3 (r2 and r3) are assigned to region 1 and lastly rule 4 (r4) is assigned to region 3. It implies that always rules are assigned to a specific region and can not be applied to any regions but the region specified.

Second, several approaches have been defined for determining the order of rules' applicability. These approaches are usually defined based on certain distributions of probabilities. We have used a modified version of well-known and proven algorithm, namely, Gillespie's algorithm. This version of the algorithm is modified to work in a compartmentalized manner. Various factors are involved to determine the order of rules' applicability in Gillespie's algorithm such as the multiplicity of reactant objects, the rule constant (a numerical constant that is placed at the top of any rule's arrow), and some random factors. Gillespie's algorithm and its modifications in this framework will be explained in detail in the following discussions.

4. The Quorum Sensing Signaling system in Genetically Engineered E.coli Bacteria Model in MC Framework:

In this section, we apply the membrane computing approach to the quorum sensing signaling system engineered in E.coli bacteria. To do so, we have to build the basic structure of the model based on the actual arrangement of membranes in these bacteria. An E.Coli bacterium has two membranes: the outer membrane that encloses the periplasmic region (space) and the

inner membrane that encloses the cytoplasm region (space)[9]. In this framework, each of these regions is indicated by a prefix and a set of brackets ([]). The prefix used for peripcasmic space is Pspace, and the prefix used for cytoplasm space is Cplasm (pSpace[] and cPlasm[]). As these two regions constitute the basic structure of our bacteria, each bacterium is presented by the following configuration:

$$eColi[pS\,pace[], cPlasm[]] \tag{2}$$

There are various objects (chemicals) that are available within these regions and need to be assigned to their particular regions based on their actual locations in the cell. When an object is assigned to a region, it is placed in the set of brackets of that region. For example, if object A is assigned to the pSpace region of a given bacterium and object B is assigned to the cPlasm region of the same bacterium, it is shown by:

$$eColi[pS\,pace[A], cPlasm[B]] \tag{3}$$

Table 1, following, summarizes these objects' names, their roles in the signaling system, and their location:

Table 1: Objects (chemicals) involved in the AI-2 signaling system engineered in E.coli and their locations

Name of The Objects	Description	Location
AI-2	Autoinducer (signaling molecule)	All
LuxPQ	The protein complex which acts as a sensor for AI-2	pSpace
LuxPQ.AI-2	AI-2 bound to the complex of LuxPQ	pSpace
LuxS	The protein which produces AI-2	cPlasm
LuxU	First protein acting in phosphorylation cascade	cPlasm
LuxO	Second protein acting in phosphorylation cascade	cPlasm
σ_{54}	Transcription factor	cPlasm
Pqrr4	RNA-4 promoter	cPlasm
mRNA	messenger RNA for production for GFP protein	cPlasm
p	phosphate group	cPlasm
LuxU.p	LuxU bound to a phosphate group	cPlasm
LuxO.p	LuxO bound to a phosphate group	cPlasm

Based on the actual locations of the objects presented in this signaling system, we are able to assign them to their particular regions as follow:

$$\begin{aligned} eColi[pS\,pace[LuxPQ, AI2, LuxPQ.AI2], cPlasm[LuxU, LuxO, \sigma_{54}, Pqrr4, mRNA, p, AI2, LuxU.p,\\ LuxO.p]] \end{aligned} \tag{4}$$

As seen in Eq.4 and Table 1, AI-2 could be transferred from one region to the others and therefore could be observed in both regions, whereas others have fixed locations.

The transportations, interactions, productions, and degradations of these objects occur by the applications of the rules. In general, the rules defined in this framework categorized into three main types. First, there are some rules that define interaction between two or more objects.

$$[A + B]_{Cell1} \rightarrow [C]_{Cell1} \tag{5}$$

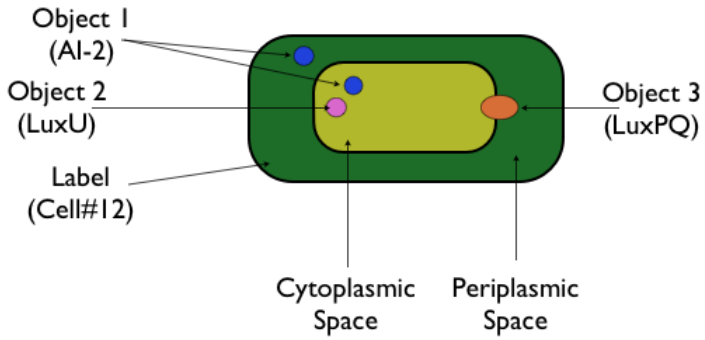

Figure 4: This figure demonstrates the location of some objects within an E.coli bacterium. AI-2 could be found within both regions whereas others (such as LuxPQ and LuxU) have fixed locations and can not be transferred from one region to the others and therefore could be found in one of the regions.

This rule expresses the interaction between objects A and B in a given membrane (Cell 1) that produces a new object, namely object C.

The second type of rules defines the chemical communication between different regions by transferring objects from one to the other. This type of rules is called "transport rules," which play an essential role in the computation power of P systems since they are responsible for establishing communications between various membranes[2].

$$[AI2]_{Cell1}[]_{Cell2} \rightarrow []_{Cell1}[AI2]_{Cell2} \tag{6}$$

This rule demonstrates that an AI-2 molecule moves from a given region (Cell 1) to another given region (Cell 2). Note that at the left side of the equation the empty set of brackets in front of cell 2 does not mean that this region does not contain any objects. Rather, it means that this region is empty in respect to the AI-2 molecule being transferred from cell 1. Using the same argument, the empty set of brackets in front of Cell 1 at the right side of the equation means that Cell 1 is empty in respect to the AI-2 molecule transferred to Cell 2 and it may contain other objects.

The last type of rules is called degradation rules, which eliminate an object from a region without transferring it to the other regions.

$$[A]_{Cell1} \rightarrow []_{Cell1} \tag{7}$$

This rule implies the degradation of a given object (A) in a given region (Cell 1).

In general, each interaction rule implies a micro transition within the simulated bacterium where the accumulation of these micro transitions lead to a macro transformation in its behavior. Because all the simulated bacteria are genetically identical, we assume an identical set of rules for them all. The set of rules available for each of the simulated bacteria consists of 23 interaction rules where each one of these rules takes care of one micro-step of the AI-2 signaling system. Now, we will briefly explain each of these rules.

The first rule (r_0) represents the production of AI-2 in cytoplasmic space by LuxS synthase:

$$r_1 = [[LuxS]_{cPlasm}]_{cell} \rightarrow [[LuxS + AI2]_{cPlasm}]_{cell} \tag{8}$$

LuxPQ complex in periplasmic space acts as kinase and add a phosphate group to the cytoplasmic protein, LuxU :

$$r_2 = [[LuxPQ]_{pS\,pace}[LuxU + p]_{cPlasm}]_{cell} \rightarrow [[LuxPQ]_{pS\,pace}[LuxU.p]_{cPlasm}]_{cell} \tag{9}$$

LuxU.p may be de-phospholated by housekeeping phosphotases in cytoplasmic space:

$$r_3 = [[LuxU.p]_{cPlasm}]_{cell} \rightarrow [[LuxU + p]_{cPlasm}]_{cell} \tag{10}$$

LuxU.p acts as a kinase and adds a phosphate group to the DNA-binding response regulator protein, LuxO:

$$r_4 = [[LuxU.p + LuxO + p]_{cPlasm}]_{cell} \rightarrow [[LuxU.p + LuxO.p]_{cPlasm}]_{cell} \tag{11}$$

Rule 5 indicates the de-phosohorylation of LuxO.p that occurs by housekeeping phosphotases in cytoplasmic space.

$$r_5 = [[LuxO.p]_{cPlasm}]_{cell} \rightarrow [[LuxO + p]_{cPlasm}]_{cell} \tag{12}$$

Next rule shows the bind between Pqrr4 promoter and Transcription factor σ_{54} in the cytoplasmic space of the bacteria.

$$r_6 = [[pqrr4_{promoter} + \sigma_{54}]_{cPlasm}]_{cell} \rightarrow [qrr4_{promoter}.\sigma_{54}]_{cPlasm}]_{cell} \tag{13}$$

The following rule shows the bind between Pqrr4 promoter and Transcription factor σ_{54} is degraded in the cytoplasmic space of the bacteria.

$$r_7 = [[\sigma_{54}.qrr4_{promoter}]_{cPlasm}]_{cell} \rightarrow [[pqrr4_{promoter} + \sigma_{54}]_{cPlasm}]_{cell} \tag{14}$$

Rule 8 demonstrates the bind between phospholated LuxO and the $Pqrr4_{promoter}$ and σ_{54} complex in cytoplasmic space:

$$r_8 = [[LuxO.p + pqrr4_{promoter}.\sigma_{54}]_{cPlasm}]_{cell} \rightarrow [LuxO.p.qrr4_{promoter}.\sigma_{54}]_{cPlasm}]_{cell} \tag{15}$$

Rule 9 is the reverse of rule 8:

$$r_9 = [[LuxO.p.pqrr4_{promoter}.\sigma_{54}]_{cPlasm}]_{cell} \rightarrow [LuxO.p + qrr4_{promoter}.\sigma_{54}]_{cPlasm}]_{cell} \tag{16}$$

The LuxO.p.σ_{54} complex binds to $Pqrr4_{promoter}$ which lead to production of the messenger RNA (mRNA) in cytoplasmic space:

$$r_{10} = [[LuxO.p.pqrr4_{promoter}.\sigma_{54}]_{cPlasm}]_{cell} \rightarrow [LuxO.p.qrr4_{promoter}.\sigma_{54} + mRNA]_{cPlasm}]_{cell} \tag{17}$$

The mRNA provides the information needed for production of GFP proteins for ribosomes in cytoplasmic space:

$$r_{11} = [[mRNA]_{cPlasm}]_{cell} \rightarrow [mRNA + GFP]_{cPlasm}]_{cell} \tag{18}$$

The mRNA might be used as the template for production of GFP proteins many times before degradation, but it will be degraded eventually:

$$r_{12} = [[mRNA]_{cPlasm}]_{cell} \rightarrow [mRNA + GFP]_{cPlasm}]_{cell} \tag{19}$$

The GFPs produced are degraded over the time:

$$r_{13} = [[GFP]_{cPlasm}]_{cell} \rightarrow []_{cPlasm}]_{cell} \tag{20}$$

Rule 14 demonstrates that AI-2 bounds to LuxPQ in periplasmic space:

$$r_{14} = [[AI2 + LuxPQ]_{pS\,pace}]_{cell} \rightarrow [[AI2.LuxPQ]_{pS\,pace}]_{cell} \tag{21}$$

The LuxPQ.AI2 complex may be degraded. This reaction takes place in periplasmic space:

$$r_{15} = [[AI2.LuxPQ]_{pS\,pace}]_{cell} \rightarrow [[AI2 + LuxPQ]_{pS\,pace}]_{cell} \tag{22}$$

The following rule describes the de-phosphorylation of LuxU.p, which occurs by LuxPQ.AI2 complex. This reaction takes place in cytoplasmic space:

$$r_{16} = [[AI2.LuxPQ]_{pS\,pace}[LuxU.p]_{cPlasm}]_{cell} \rightarrow [[LuxPQ.AI2]_{pS\,pace}[LuxU + p]_{cPlasm}]_{cell} \tag{23}$$

In the absence of LuxU.p complex, LuxO would not be phospholated anymore, and housekeeping phosphotases removes phosphate groups from existing LuxO.p complexes (rule 5).

There are also some rules that take care of transportations and degradation of AI2. As a review, AI2 is produced in cytoplasmic space and it is accumulated in the environment, therefore it is transported between different regions. There are 4 rules that demonstrate the transportation of AI2 within the membranes. Rule 17 takes care of the transportation of AI2 from cytoplasmic space to periplasmic space:

$$r_{17} = [[]_{pS\,pace}[AI2]_{cPlasm}]_{cell} \rightarrow [[AI2]_{pS\,pace}[]_{cPlasm}]_{cell} \tag{24}$$

AI2 is also able to move back to cytoplasmic space from periplasmic space:

$$r_{18} = [[AI2]_{pS\,pace}[]_{cPlasm}]_{cell} \rightarrow [[]_{pS\,pace}[AI2]_{cPlasm}]_{cell} \tag{25}$$

Rule 19 demonstartes the transportation of AI2 from periplasmic space to the environment:

$$r_{19} = [[[AI2]_{pS\,pace}]_{cell}]_{e} \rightarrow [AI2[[]_{pS\,pace}]_{cell}]_{e} \tag{26}$$

AI-2 may also move back to perplasmic space from the environment. The following rule usually occurs when a sufficient amount of AI-2 is accumulated in the environment:

$$r_{20} = [AI2[[]_{pS\,pace}]_{cell}]_{e} \rightarrow [[[AI2]_{pS\,pace}]_{cell}]_{e} \tag{27}$$

Finally, AI2 may be degraded in the cytoplasmic, periplasmic, and environment before getting any chance to bind to the LuxPQ complex. The last 3 rules show the degradation of AI-2 in these regions:

$$r_{21} = [[AI2]_{cPlasm}]_{cell} \rightarrow []_{cPlasm}]_{cell} \tag{28}$$

$$r_{22} = [[AI2]_{pS\,pace}]_{cell} \rightarrow []_{pS\,pace}]_{cell} \tag{29}$$

$$r_{23} = [AI2]_{e} \rightarrow []_{e} \tag{30}$$

5. Gillespie's Algorithm

In order to model (simulate) a biological systems evolution over time realistically, a precise mathematical model which takes into account stochasticity of involved processes is required. Traditional deterministic differential equation approach for modeling biological systems, while accurate for large systems, is often not sufficient for small systems where key species may be present in small numbers. Gillespie algorithm developed and published by Dan Gillespie in 1977 is a mathematical model which generates possible solutions of a stochastic equations. Gillespies algorithm is used to model chemical or biochemical systems precisely and sufficiency given limited computational power and has been proved over many years. Here we use an extended version of Gillespies algorithm called Multi-compartmental Gillespies Algorithm introduced by Perez-Jimenze and Romero-Capero, 2006. In the original Gillespies Algorithm only one volume is studied however in P systems, notion of multi-compartments dictates multiple regions. Besides the application of a rule inside a given membrane can also affect the content of another one. In order to realistically model interactions among different compartments (cells), Multi-compartmental Gillespies Algorithm is used [6].

Let $\Pi = env[elements[], cells[c_1, c_2, ..., c_n], rules[r_1, r_2], label[e]]$ be a P system.

1. For each cell in P system $\in \{c_1, c_2, ..., c_n\}$ calculate $a_0 = \sum p_{j_i}$ where p_{j_i} is the probability of a rule i contained in cell j to be applied in the next step of evolution; this probability is computed by multiplying a stochastic constant c_i by number of the object present on the left-side (reactant) of the rule i.

2. Generate two random numbers r_1 and r_2 uniformly distributed over the unit interval (0,1).

3. Calculate the waiting time for the next reaction as $w = \frac{1}{a_0} ln(\frac{1}{r_1})$

4. Select a rule to be applied next once a cell is selected such that $\sum_{k=1}^{j-1} p_k < r_2 a_0 \leq \sum_{k=1}^{j} p_k$

5. Return the triple (w, j, i).

The triple which is returned contains a waiting time for the cell, the label of the cell and index of which rule to be applied in respect to selected cell. It is worth noting that the higher the stochastic constant of a give rule is and higher the reactant concentration, the rule has a higher chance of being applied. Each time steps is computed by the algorithm so time steps between reactions are stochastically determined. These time steps are not totally random because their computed value depends also on the current configuration of the system. Using compartmentalized Gillespie's Algorithm we are simulating parallelism in our system. Because we do not have descrite time steps, we also consider the fact that different reactions take different amount of time to complete. Gillespie also allows us to monitor state of individual cells and reactions as well as the population of cells.

6. Division

Bacterial cell division is the process by which a bacterial cell creates two genetically identical daughter cells. It takes approximately 20 minutes for a newborn bacterial cell to reach to the next division stage in a proper environmental condition. Within this 20 minutes, parental genetic materials and its essential components and organelles are duplicated and distributed between the two daughter cells. However the chemicals available within the parent cell are not duplicated, whereas they are equally distributed between two daughter cells. One of the characteristics of bacterial populations is that they increase in size exponentially in optimum environment. However, toxic waste products and competition for food and nutrients reduce the speed of population growth and stabilize the number of bacteria in the population.

One of the unique aspects of this frame work is that division is introduced to the simulated bacterial population. This means that while a simulation is running the number of bacteria increases exponentially in proportion to the simulation time. For instance, if a simulation starts with one cell, the number of cells may increase to 2, 4, 8, 16, and so on (Figure 5).

As mentioned, all the membranes are synchronous using a global clock in the sense that a global clock is assumed which holds time for all the bacteria in the population [5]. This global time is counted by Gillespie's algorithm in which the algorithm assignsa specific time unit to each interaction rule and within each timestep there is only one rule applied to the system while other rules are waiting for the rule chosen to be applied. Division function is applied to the simulated population at specific time intervals. These intervals are calculated based on the Gillespie's global clock. Within each interval, simulated bacteria interact in a normal way. However at each division, genetic materials and essential components of each bacterium such as LuxPQ, LuxO, and Pqrr4 promoter are duplicated and distributed between daughter cells, whereas other chemicals like AI-2 molecules and phosphate groups are equally distributed between daughter cells without any duplications. Figure 6 demonstrates the distribution of AI-2 molecules from parent cells to daughter cells at two divisions.

As it could be observed, at each division AI-2 molecules are distributed equally between daughter cells and that is the way the number of AI-2 molecules changes logarithmically between divisions, and suddenly drops at each one.

As this model is used to qualitatively study the AI-2 signaling system within individual cells in bacterial populations, and one of the important characteristics of bacterial populations is division, it could be concluded that having the division function in the model provides closer inspiration of the emergent properties of the system. For instance, it was discussed in pervious sections that with high concentration of AI-2 in the environment, the bacterial cells signal each other to stop producing GFP proteins. When newborn cells come to existence, their LuxPQ complex should soon recognize the high concentration of AI-2 molecules in the environment and switch from being kinase to being phosphatase and hence the new cells should stop producing GFP proteins very fast. This interpretation of the system is depicted in Figure 7 where the first graph shows that the number of AI-2 molecules in the environment is continuously increasing over the 7000 steps. By the increase in the number of AI-2 molecules in the environment, the bacterial cell should recognize them easier and cancel the production of GFP proteins. The second graph shows the number of GFP proteins within one of the parent cells (indicated by"P") in the simulation. This cell keeps producing GFP proteins up to the division point. After that the newborn daughter cell (indicated by"D") continues the production of GFPs, however it does not last very long as this cell reaches to the point (indicated by red line) that recognizes the high concentration of AI-2 in the environment and cancel the production of GFPs. After this point, the degradations of GFPs occur which is the reason that the bacterial cells turn dark. It could be interpreted that the production of GFP proteins in the newborn cell is much less than the parent cell and the reason for that is, by the time that the newborn cell came to exist, the concentration of AI-2 in the environment was already high. Therefore it takes shorter for the LuxPQ complexes of the newborn bacterial cell torecognize the AI-2 molecules in the environment and hence switch ofthe production of GFPs. Graph 3 demonstrates the number of GFP proteins in one of the newborn cells in the simulation. As the division occurs at some point around 2500, the time step shown in this graph contains 4000 time step. This graph shows that the production ofGFP proteins occurs once (indicated by black line) in this cell. However after a short time the cell changes its biological cascade and hence the inherited GFP proteins and the produced one are degraded. Using the same reasoning, since the concentration of AI-2 molecules in the environment was high by

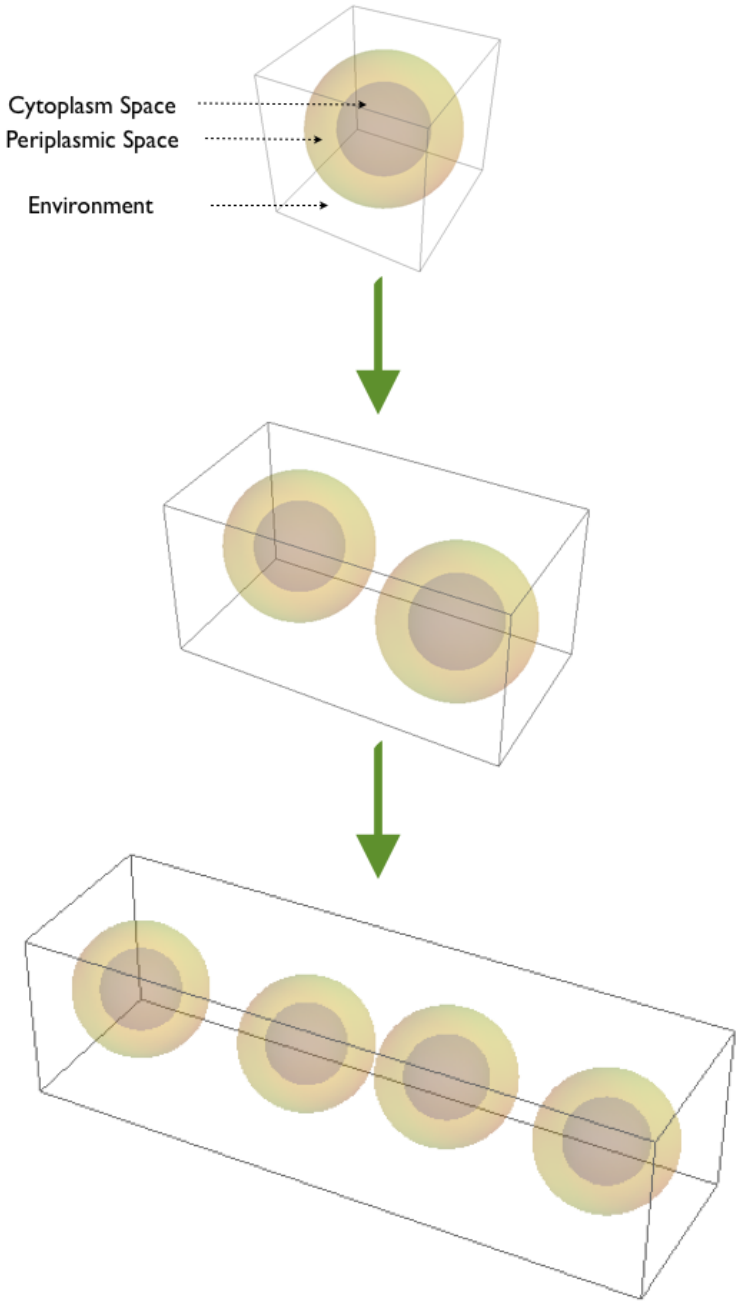

Figure 5: Division in Simulated Bacterial Population. This figures demonstrates that the simulation is started with one cell, and two divisions has been applied to the bacterial population. Therefore the simulation ended up with 4 bacteria. In this framework division as one of the most important characteristics of bacterial population has been taken into consideration. which enables users to gain a more realistic understanding of the emergent properties of the population.

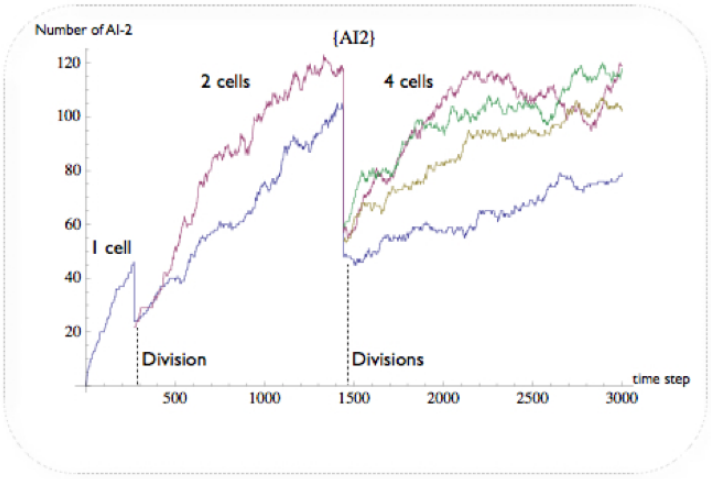

Figure 6: Distribution of AI-2 Molecules between newborn cells from parent cells. On X-axis, time steps are aligned, and on Y-axis number of AI-2 molecules are placed. This graph shows that the simulation is ran for 3000 time steps and started with one cell (blue line) and other cells are generated over time by divisions (red, green, and yellow lines). This graph demonstrates thatnumber of AI-2 molecules changes logarithmically between divisions, and suddenly drops at each division.

its birth time, it switches to the second cascade faster than their parent cells that kept producing GFPs for more than 2000 time steps.

It is now obvious that the newborn daughter cells behave differently than their parent cells in the population. This was an example of how division function covers some of the emergent properties of the simulated population. As it was mentioned, some factors such as wasted products and competition over nutrients slow down the growth of bacterial population over the time. The interesting point about the division function implemented in this framework is that as the number of cells increases, the division function is applied to the system in longer intervals. The reason is that as the number of bacterial cells grows, there are more interaction rules within the system that should be applied by Gillespie's algorithm, and therefore the division function's interval spans longer. For instance, Figure 8 shows the change of number of AI-2 in cells in a simulation, wherein the first division occurs some step before 200, and the second one takes place a few steps after 900, and then there is no division for more than 2000 steps.

7. Results

This frameworks provides interactive results that could be used to qualitatively study the biological system. We have utilized the use of various tools such as graphs, charts, and matrixes to represent the results. For example, concentration graphs for objects involved in the simulation demonstrate the qualitative trends in the system's behavior. It means that while the objects' concentrations might not be directly reflect the experiments in vitro, they present the biological trends underlying the dynamical behavior of the system. Figure 9 compares the change of concentration of four different objects namely AI-2, GFP, LuxU.p, and LuxO.p within one of the simulated cells to the change of concentration of AI-2 molecules in the environment. First

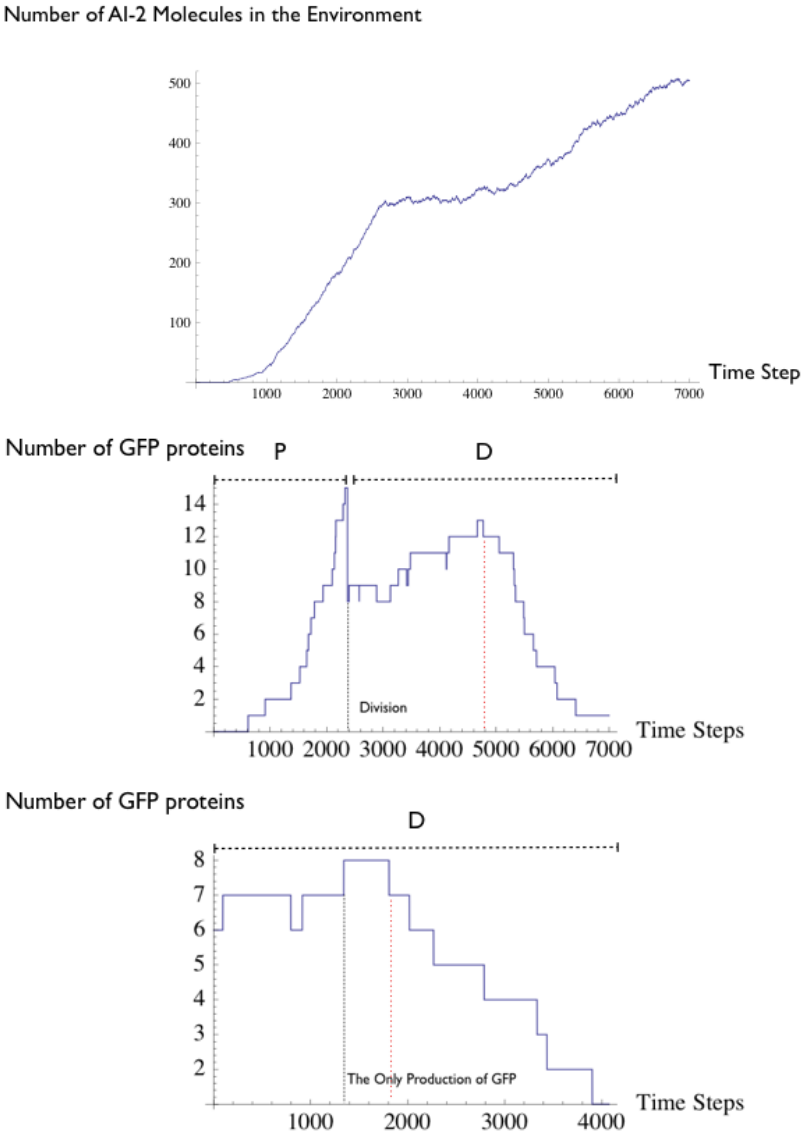

Figure 7: Comparison of production and degradation of GFP proteins in parent cells and newborn cells. The First graph shows that the number of AI-2 molecules in the environment is continuously increasing over the 7000 steps. The second graph shows the number of GFP proteins within one of the parent cells (indicated by ”P”) in the simulation. This cell keeps producing GFP proteins up to the division point. After that the newborn daughter cell (indicated by ”D”) continues the production of GFPs, however it does not last very long as this cell reaches to the point (indicated by red line) that recognizes the high concentration of AI-2 in the environment and cancel the production of GFPs. After this point, the degradations of GFPs occur. Graph 3 demonstrates the number of GFP proteins in one of the newborn cells in the simulation and implies that the production of GFP proteins occurs once (indicated by black line) in this cell. However after very soon the cell changes its biological cascade and hence the inherited GFP proteins and the produced one are degraded. The reason for observing this behavior is since the concentration of AI-2 molecules in the environment is high by the birth time of daughter cells, they switches to the second biological cascade faster than their parent cells that keep producing GFPs for more than 2000 time steps.

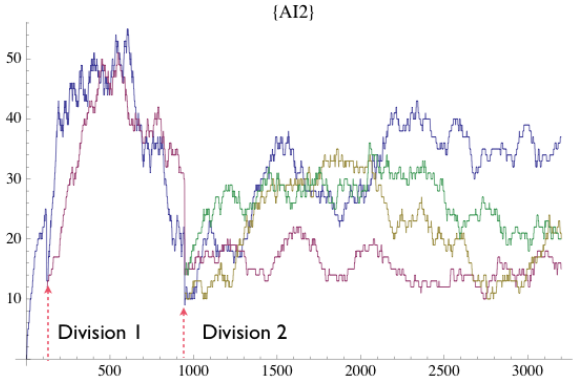

Figure 8: The division intervals span longer due to the competition over foods, and increasing amount of wasted products in bacterial populations. This figure indicates that the first division takes place before 200 time steps, whereas the second one is placed almost 800 steps after the first one. After the second division there is no more division for more than 2000 time steps.

graph demonstrates the change of concentration of AI-2 molecules within the cell. At the beginning, number of AI-2s is continuously increasing and these molecules are accumulated in the cell, without any transportation to the environment. However, after around 2500 steps the cell reaches to the point that it starts to transport AI-2 molecules to the environment (indicated by yellow line). As it could be seen in graph 5 in this figure, the number of AI-2 molecules are exponentially increasing in the environment after some steps between 2000 and 3000 (indicated by red arrow in the figure). After the massive increase in the concentration of AI-2s in the environment, the LuxPQ complex of the cell (which was adding phosphate groups to the cytoplasmic protein, LuxU) changes its behavior from being kinase to being phosphatase. Therefore, after this point, this complex starts removing phosphate groups from LuxU. This circumstance could be observed in graph 2, where the number of LuxU.p increases exponentially at the beginning of the simulation, and then this number drops suddenly at some step around 3000 (shown by green line), as the LuxPQ complex start de-phosphorylating these proteins. When LuxU.p is de-phosphorylated, the LuxO.p complex will be degraded by housekeeping phosphates as shown in the third graph. Without the LuxO.p complex, GFP proteins could not be produced anymore, and therefore their number decreases as they start to be degraded some step after 3000 (shown by black line in graph 5). These proteins will be completely degraded over the time, and that is the reason why the cell turn dark after a while.

There is another type of results that uses matrixes to represent the results.Figure 10 is an array plot matrix that demonstrates the trend of AI-2 binding to the LuxPQ complex. Twenty individual E.coli, indicated respectively by each column, are run through the simulation over 50 time steps, depicted by the horizontal rows. The simulation begins (i.e., t=0), a new row is added above the previous one and the degree to which AI-2 is bound to the LuxPQ complex is visualized for each of the 20 bacterial cells. This spectrum of LuxPQ bound AI-2 is represented by the color gradient of red to white to blue wherein an individual red cell represents an E.coli where the majority of LuxPQ complex is free, whereas a blue cell represents an E.coli where the

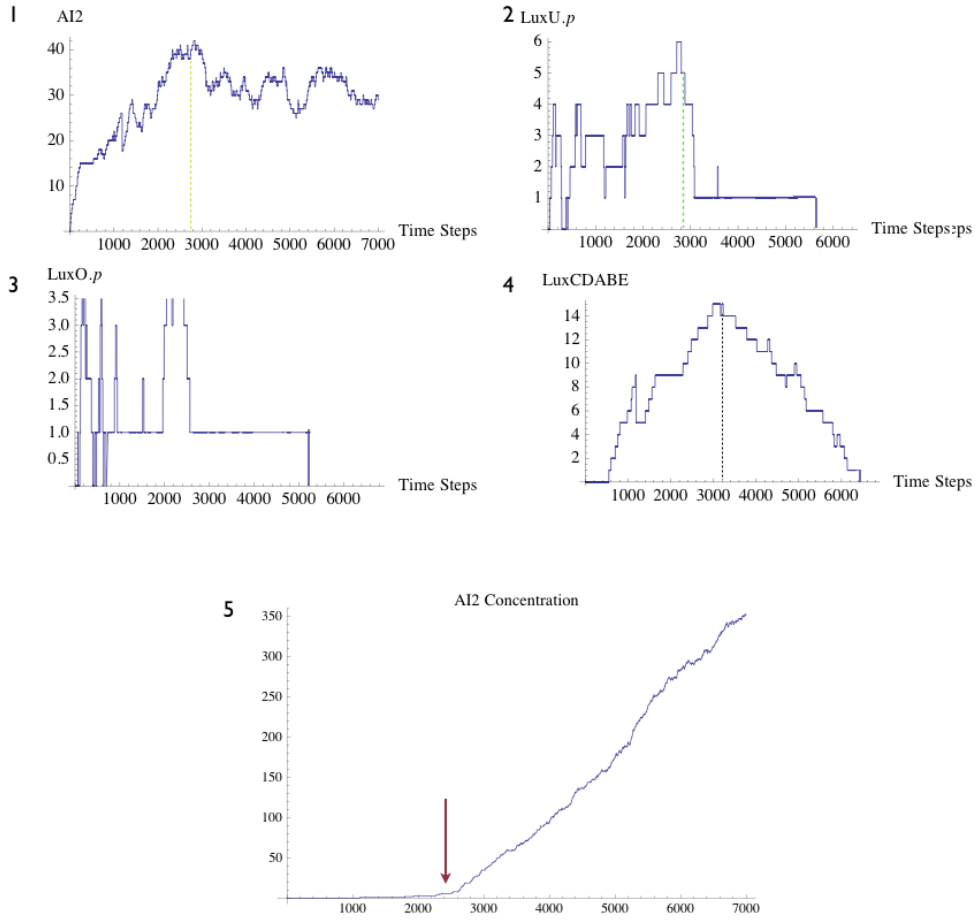

Figure 9: Comparison of change of concentrations in di?erent chemicals, namely AI-2 inside the cell, LuxCDABE (GFP), LuxO.p, LuxU.p, and AI-2 in the environment. This gure shows that as the number of AI-2 molecules increases in the environment, the degradation of LuxCDABE (GFP), LuxO.P, and LuxU.p occurs within the cell.

majority of the LuxPQ complex is AI-2 bound. As time progresses and more rules are applied, more AI-2 is produced by LuxS and becomes bound to the LuxPQ complex of all E.coli in the simulation. This is shown by the overall trend that the cells of each column converge to dark blue as time progresses.

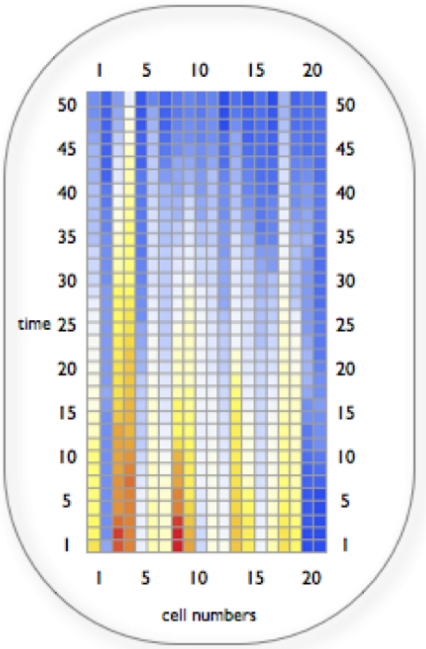

Figure 10: AI-2 Binding to the LuxP-LuxQ Protein Complex. Each column represents one of twenty E.coli bacteria. The state of the modelled bacteria over a period of 50 simulated time steps is depicted along the vertical axis. The color of each cell indicates the binding degree between AI-2 and the LuxP-Q complex. The color spectrum spans from red (no binding) over white to blue (complete binding). As time progresses an increasing amount of AI-2 is produced by LuxS and gets bound to the LuxPQ complex.

Another example of the results is a chart which indicates the distribution of interaction rules in the system. In this chart (Figure 11), the interaction rules are aligned on X-axis. Each rule is shown by r and a number (e.i., r0= rule 0). The number of applications of the rules are aligned on Y-axis. The heights of the columns on the chart demonstrate the numbers of times of each interaction rule's application. Various colors are assigned to the rules' columns for thepurposeof differentiation. This type of results provides information about theimportance of each interaction rule within the simulation and demonstrates how changes in the rates of reactions affect the rule distributions.

8. Features

Beside the fact that a model should be able to produce accurate and useful outputs, it should also be user-friendly and informative. In respect to this point, many features have been added

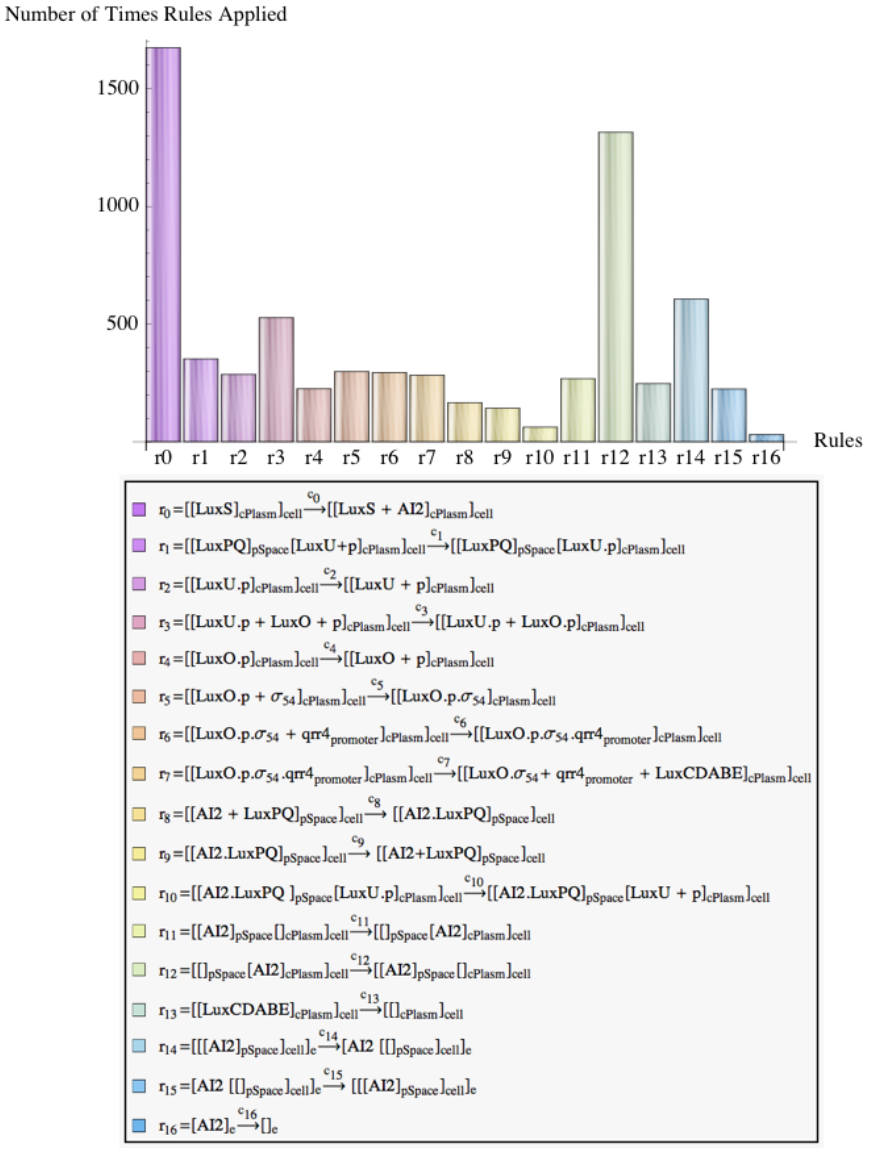

Figure 11: Distributions of Applied Rules. For each rule r0 to r16 the number of its application is charted. Charts like this help to understand which rules, i.e. which interactions, are more or less important within the simulated system or how changes in the rates of reactions affect the rule distributions.

to this model to make it as useful and instructive as possible for the users. These features are accessible from the interface designed for this model.

The basic idea underlying the interface of our model is that it enables the user to manipulate the parameters involved in the simulation without needing to know how the code works. Thus, users who are mostly biologists do not need to compile anything or type a single line of code while using this model. This is the beauty of Mathematica, the high level programming language that is used to implement this framework. Figure 12 demonstrates the interface of this model [10]. As it could be seen in the figure, there are three tabs at the top of the interface: "Input", "Visualization", and "About". Input tab includes 2 sub-tabs: "Simulation parameters" and "Rule Constants". Under simulation, the determining simulation parameters could be manipulated and modified by the users. These parameters are categorized in four main groups: Simulation, Cytoplasmic, Periplasmic, and Environment.

As the figures shows, there are three tabs at the top of the interface: "Input," "Visualization," and "About.". The input tab includes two sub-tabs: "Simulation parameters" and "Rule Constants. Under simulation, the determining simulation parameters could be manipulated and modified by the users. These parameters are categorized in four main groups: Simulation, Cytoplasmic, Periplasmic, and Environment.

In simulation, two general variablespopulation size and simulation stepscould be modified according to the users needs. Population size indicates the number of bacteria involved in the simulation, and simulation steps determine the number of steps the simulation will be running. Cytoplasm, Periplasmic, and environment also get their own list of parameters, which are the number of objects in each region of each bacterium. In addition, at the bottom of the interface is a check box that allows users to introduce division into the mix. When the division check box is not selected, division will not occur during the simulation and the simulation will end up with the same number of bacteria with which it started. However, when the division checkbox is selected, simulation might start with 20 cells and based on how many steps are specified by the user, it might end up with 40 or 80 cells. Next, below the checkbox, is a button for running the simulation; the status of simulation is shown below ("Ready!"). To give users an idea about the status of the simulation, we have added a progress bar, which appears only when the program is running, that demonstrates how long the simulation will take before completion. The following figure shows a screenshot of a simulation in progress:

Rule Constants is the second tab under the Input tab. Each rule in this framework is associated with a constant. As mentioned in section X, this constant is used for calculating the probability of the interaction rules. The system is loaded with default values for these constants. However, the user could modify each value before running the simulation. As it would be hard for users to remember which constant belongs to which rule, a tool-tip option is activated so that when a user moves the mouse pointer over each constant, its corresponding interaction rule is displayed (Figure 13).

"Visualization tab" is adjacent to Input tab and contains animated tutorials of the simulated biological system. As it was mentioned in section 1, the biological system consists of two cascades, where the first cascade occurs in absence of AI-2 molecules and leads to production of GFP proteins. These proteins emit light and therefore the bacteria glow. The second cascade takes place when AI-2 concentration is high in the environment. In this condition, these molecules enter the cells and switch off the production of GFP proteins and thus the bacteria turn dark by degradation of existing GFPs. The animations provided in this model demonstrate the interaction between the chemicals and their corresponding rules (Figure 14).

The last tab is called "About" and it is where users are informed about copy rights and

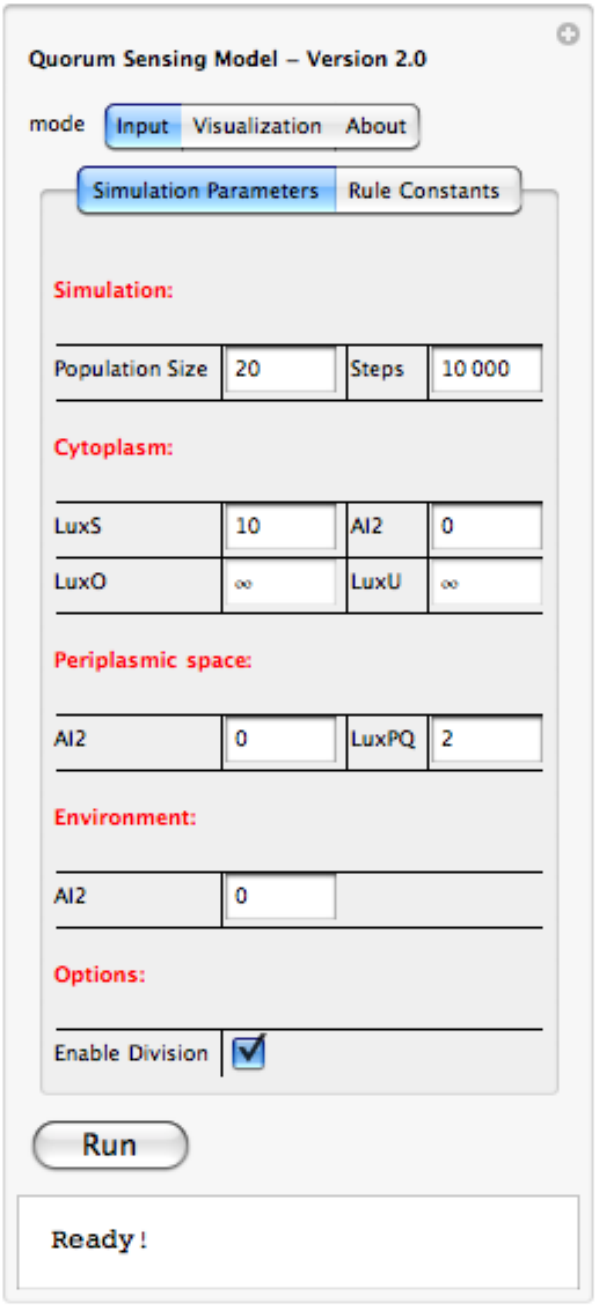

Figure 12: The interface of this model. This interface provides a very user-friendly environment for the users in a way that they do not need to know anything about neither computer science nor the way that this framework works.

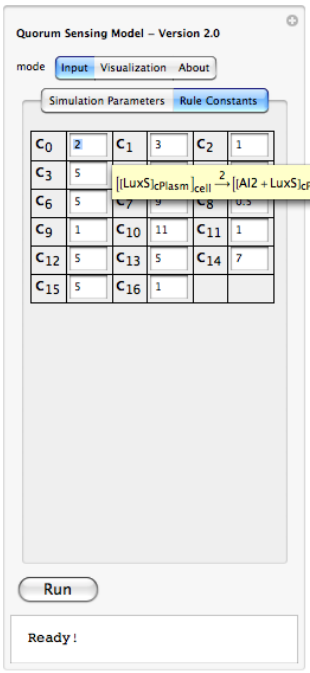

Figure 13: A screen shot from Rule Constant tab of the interface. In this section of the interface, users are able to manipulate rule constants and as it would be hard to remember the relation of each rule constant with its corresponding rule, a tool-tip option is activated, so that it the mouse pointer is moved over each constant, it shows the related rule.

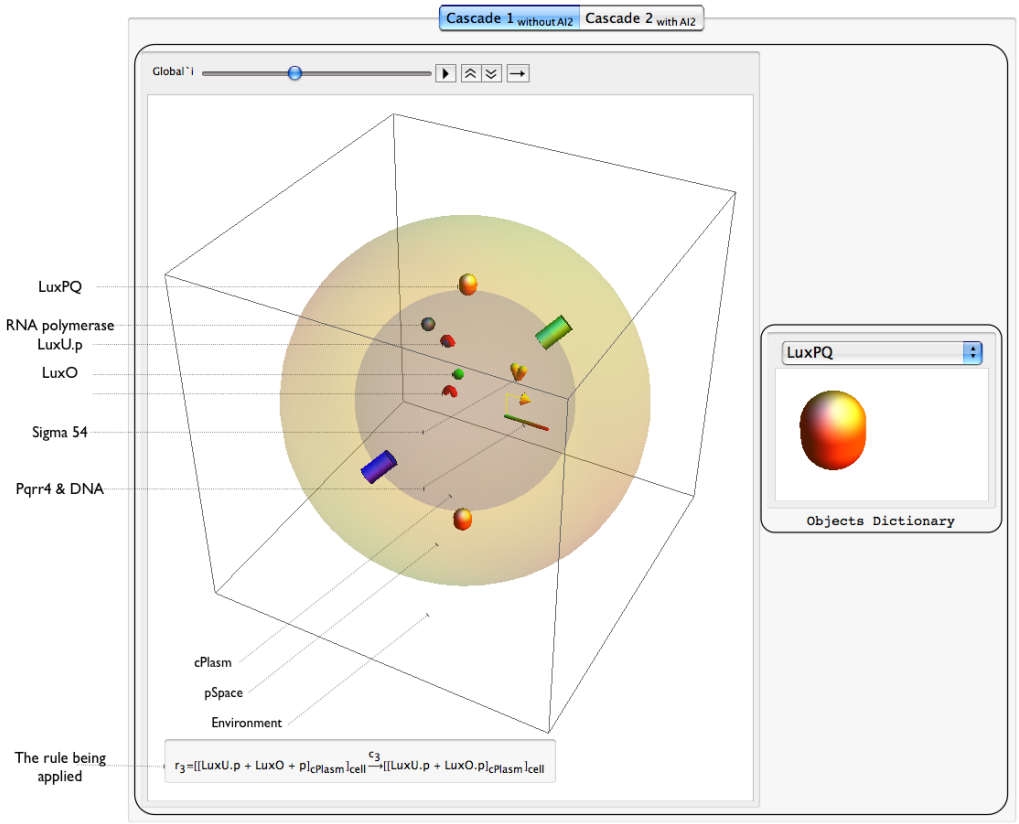

Figure 14: Animated Tutorials. In these animations, the interactions of each cascade is depicted. While an interaction is taking place, its corresponding rule is shown at the bottom of the panel. These animations enable the users to learn about the simulated biological system and to couple each interaction with its parallel rule.

developers informations.

9. Future Directions

In order to model our system even closer to biological system in vitro, we plan to take our model step further and take into account spacial dimensions using an agent based approach. At its current form, our special dimensions integrated in our model is in terms of compartments. Chemical substances in the model have special location with respect to each compartment. They are either inside or outside. There is no releative position to each membrane. Currently there is no notion of being closer or further from a cell. With introducing spacial dimension we take each cell from being an entity to an autonomous agent. Each cell will act on its own and will have its own interaction rules. Thus agents (cells) communicate and evolve over time. Due to abstractions integrated in our mode, this work once completed can be used as a tool for biologist to simulate complex biological systems. Integrating the ability to export our results in many different formats such as SBML (System's Biology Markup Language) as well as HTML (for presenting results online) allows users to import our model to other systems such as Cell Designer and also share important results (properly formatted) online.

10. Conclusion

The uniqueness of our approach compare to the traditional differential equations is that it provides various interactive results that could be used for qualitatively studying the biological system. The results of this framework highlight HOW and WHY the system behaves in certain ways, whereas other models such as differential equation based models are able to capture solid states of the systems' behaviour. In other words, if other models are described as pictures taken from different status of the biological system, this model would be described as a film taken from the transmission period of the system.

In this work, we have used quorum sensing as a complex biological system for evaluating our platform. In context of quorum sensing, this tools allow researchers to gain insight to aspects of the system never explored before. As an example, our model allows for looking at application of interaction rules and their frequency of application which allows for gaining new insights to stability and robustness of the system.

We have developed a biological language that is able to simulate any biological system that can be expressed in terms of membranes. Our work can be summarized simply as a tool for biologists to evaluate and monitor complex systems. In order to fill the gap between computer science and biology, we have paid special attention into developing a system that is intuitive and easy to use. With simple interface and pre-defined notations for rule implementation, this model provides a user friendly tool to biologists to develop complex models of different biological systems.

References

[1] Bernardini, F., Gheorghe, M. and Krasnogor, N. (2006). On P systems as a modelling tool for biological systems. *Lecture Notes in Computer Science*, **vol. 3850**: pp. 114 - 133

[2] Bernardini, F., Gheorghe, M. and Krasnogor, N. (2007). Quorum sensing P systems. *Theoretical Computer Science*, **vol. 371**: (1-2) pp. 20 - 33

[3] Miller, M. B. Bassler, B. L. (2001). *Annu. Rev. Microbiol.* **Vol. 55**, pp. 165-199

[4] Neiditch, MB., Federle, MJ., Miller, ST., Bassler, BL., Hughson, FM. (2005). Regulation of LuxPQ receptor activity by the quorum-sensing signal autoinducer-2. *Mol Cell*. **vol. 18 (5)**: pp. 507-518
[5] Päun, Gh. (2001). Membrane Computing: An Introduction. *Springer, Berlin*.
[6] Perez-Jimenez and Romero-Campero. (2006). P systems, a new computational modelling tool for systems biology. *Transactions on Computational System Biology*. **vol. 4220**: pp. 176-197
[7] Priami, C. (2009). Algorithmic Systems Biology. *Communication of the ACM*. **vol. 52 (5): pp. 81-89**
[8] Romero-Campero and Prez-Jimnez. (2008). A model of the quorum sensing system in Vibrio fischeri using P systems. *Artif Life*. **vol. 14 (1)** : pp. 95-109
[9] Waters C. and Bassler B. (2005). Quorum sensing: cell-to-cell communication in bacteria. *Annu Rev Cell Dev Biol*, **vol. 21**: pp. 319-46
[10] Wolfram Resarch Inc. `http://www.wolfram.com`
[11] Zwillinger, D. (1997). Handbook of Differential Equations (3rd edition). *Academic Press, Boston.*