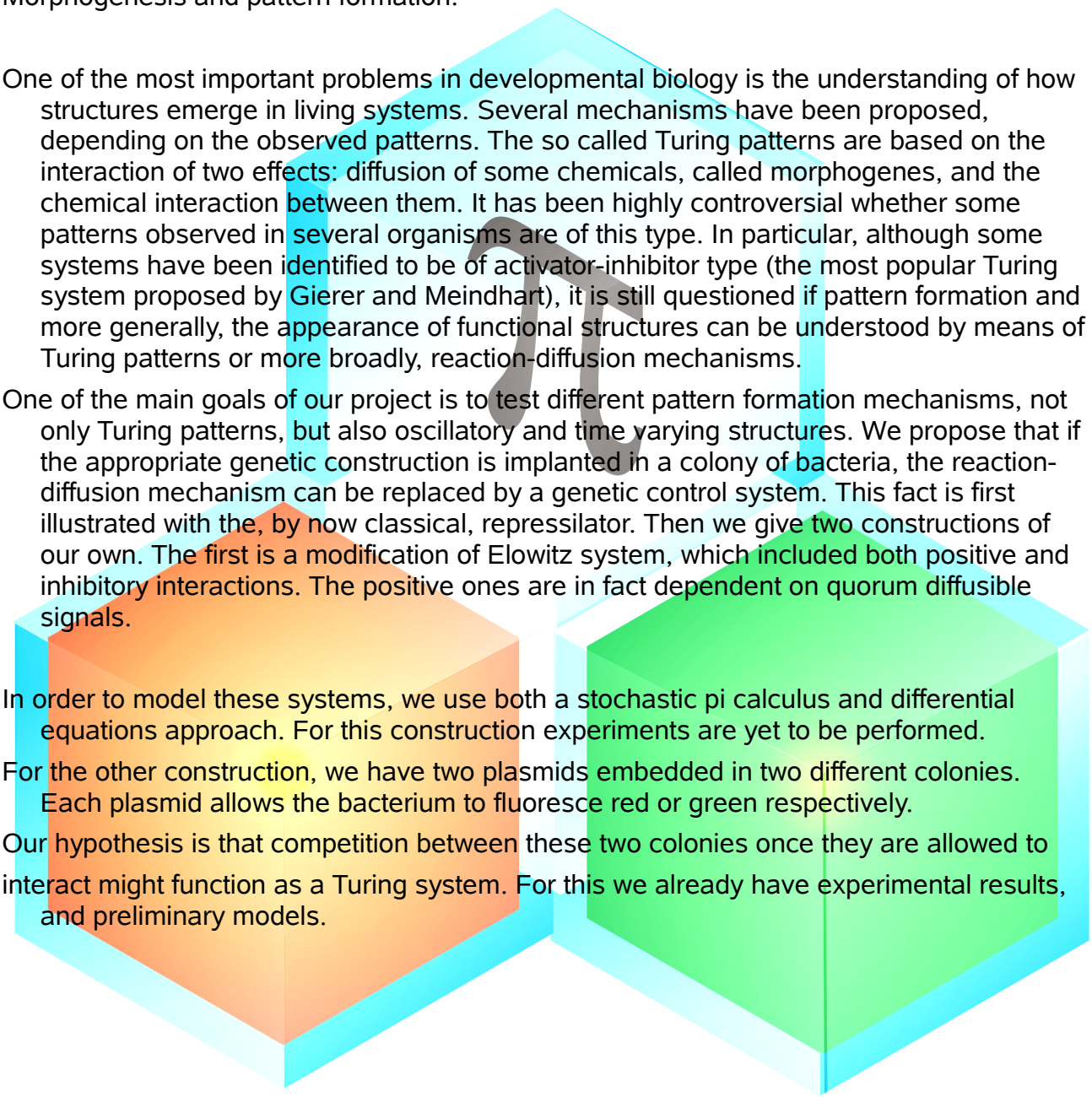
iGEM México Simulation on Stochastic π-Calculus

Introduction

Morphogenesis and pattern formation.

One of the most important problems in developmental biology is the understanding of how structures emerge in living systems. Several mechanisms have been proposed, depending on the observed patterns. The so called Turing patterns are based on the interaction of two effects: diffusion of some chemicals, called morphogenes, and the chemical interaction between them. It has been highly controversial whether some patterns observed in several organisms are of this type. In particular, although some systems have been identified to be of activator-inhibitor type (the most popular Turing system proposed by Gierer and Meindhart), it is still questioned if pattern formation and more generally, the appearance of functional structures can be understood by means of Turing patterns or more broadly, reaction-diffusion mechanisms.

One of the main goals of our project is to test different pattern formation mechanisms, not only Turing patterns, but also oscillatory and time varying structures. We propose that if the appropriate genetic construction is implanted in a colony of bacteria, the reaction-diffusion mechanism can be replaced by a genetic control system. This fact is first illustrated with the, by now classical, repressilator. Then we give two constructions of our own. The first is a modification of Elowitz system, which included both positive and inhibitory interactions. The positive ones are in fact dependent on quorum diffusible signals.



In order to model these systems, we use both a stochastic pi calculus and differential equations approach. For this construction experiments are yet to be performed.

For the other construction, we have two plasmids embedded in two different colonies. Each plasmid allows the bacterium to fluoresce red or green respectively.

Our hypothesis is that competition between these two colonies once they are allowed to interact might function as a Turing system. For this we already have experimental results, and preliminary models.

General description of the Stochastic π-Calculus:

Systems Biology can be broadly divided into two complementary approaches. On the one hand, scientists are doing experiments in the lab and studying the results, in order to infer general properties of biological systems. On the other hand, scientists are trying to build detailed models of systems on a computer and then testing these models in the lab. This is the approach that we are focussing on at this part of iGEM México. Such models can be a powerful tool, since they allow a biologist to simulate a range of experiments on a computer, before testing the most promising ones in the lab, saving considerable time and money. They also clarify how a biological system functions.

In order to build such models, we need tools that are suitable for modelling complex, parallel biological systems. We also need tools that can scale up to very large systems. This points towards the need for a biological programming language.

Over the years, there has been considerable research on designing programming languages that are suitable for developing complex parallel computer systems. Interestingly, some of this research is also applicable to biological systems, which are typically highly complex and massively parallel. One example of such a language is the stochastic pi-calculus...

The pi-calculus is also a mathematical programming language. There has been decades of research on analysis techniques for pi-calculus. Many of these techniques are directly applicable to biological modelling, and could help provide insight into fundamental properties of biological systems.

The stochastic pi-calculus is essentially a calculus of actions and processes. Specifically developed for modelling concurrent systems in Computer Science, but can be used to model Biological systems in a nice way. Each process can be used to describe the behavior of a molecule in the system, such as a gene or protein. The actions describe what the molecules can do. In particular, a molecule can do an input, an output or just a delay.

Delay represents an internal reaction like radioactive decay, or change of shape. Delay is associated with a rate, which determines the average duration of the reaction. Like the rate of radiactive decay, which is used to determine the half-life.

Input and output represent interactions between two molecules, which interact by performing a complementary input and output on the same channel. Can represent two proteins with complementary shapes, or two chemicals that are known to react with each other. Can also send values, such as an electron or a phosphate. Note that 3-way reactions are extremely rare: very low probability of three molecules reacting at exactly the same time. Usually two molecules interact, and then a third. Pi-calculus fits this model nicely.

Often a given molecule can do more than one thing. i.e. can either react with another molecule or decay. Represented as a choice of actions.

We can of course have multiple molecules in parallel. Represented as parallel composition. Each parallel process represents a separate molecule.

We also want to define different types of molecules, using parameterized “macros” or “procedures”. These are defined in an environment. Finally, want to have private bonds, which are used to represent the formation of complexes.

Simulation with Stochastic π-Calculus(SPiM[1]):

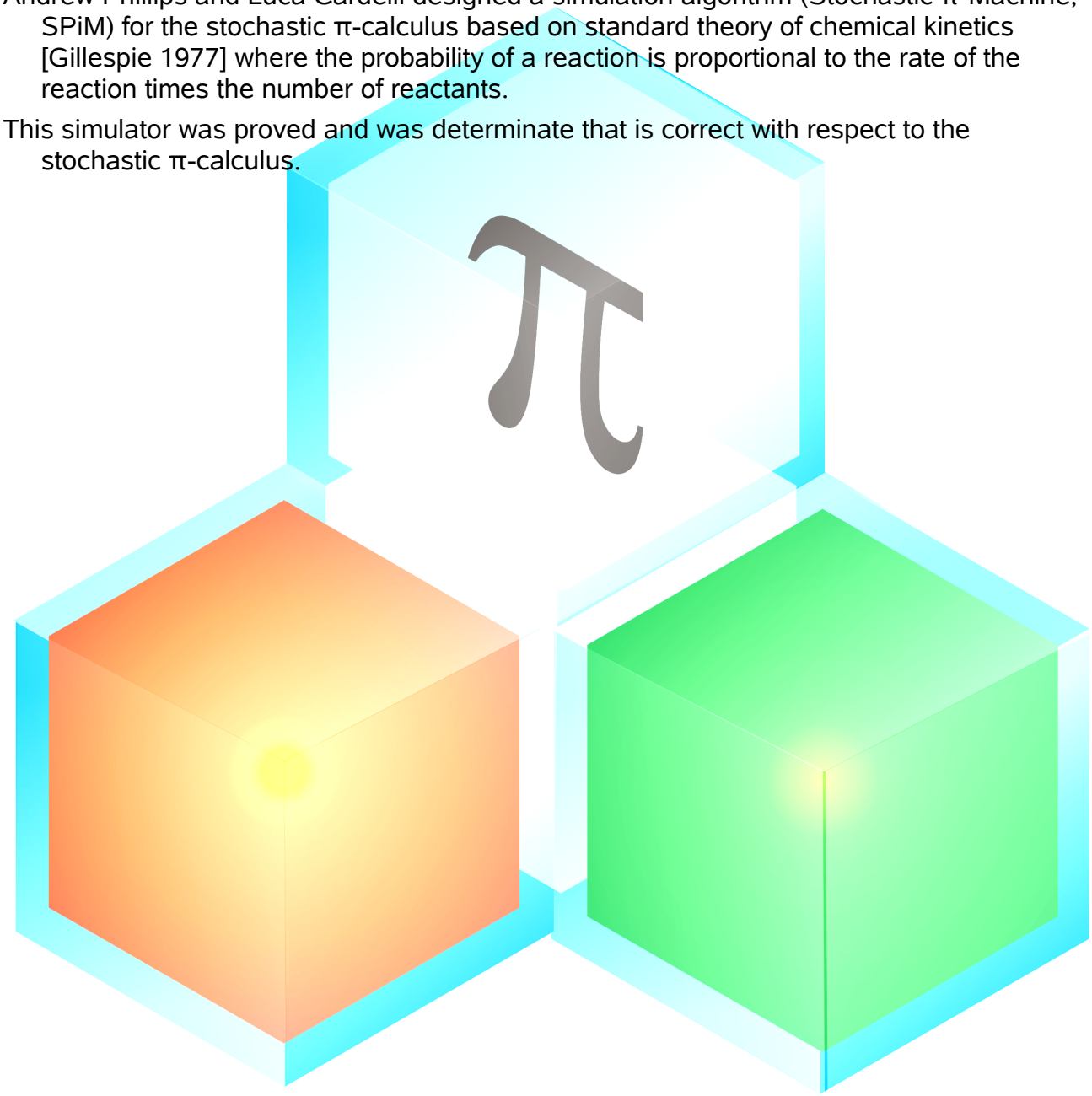
1©Andrew Phillips 2007. The Stochastic Pi Machine (SpiM). Version 1.12

The Stochastic π-Calculus has been used to model and simulate a range of biological systems [1,2]. One of the main benefits of the calculus is its ability to model large systems incrementally, by composing simpler models of subsystems in a intuitive way. Various stochastic simulators have been developed for the calculus , in order to perform virtual experiments on biological system models. Such *in silicio* experiments can be used to formulate testable hypothesis on the behavior of biological systems, **as a guide to future experiments *in vivo*.**

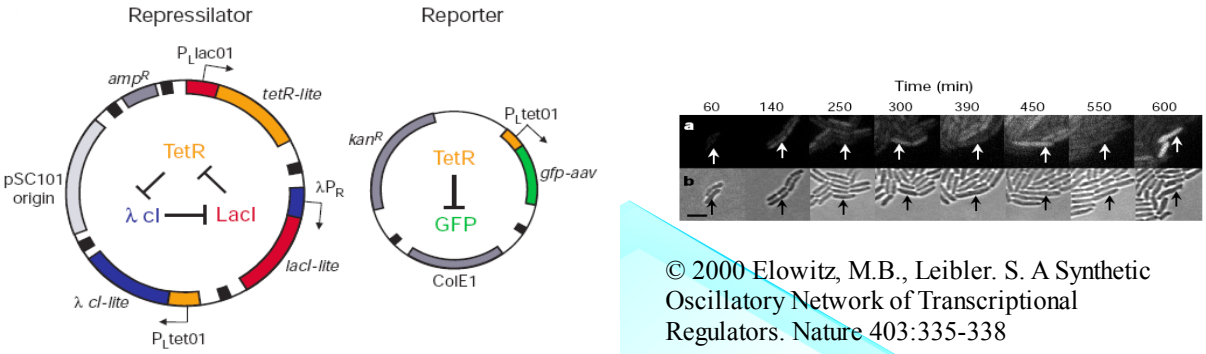
Currently available simulators for the stochastic π-calculus are implemented based on standard theory of chemical kinetics, using an adaptation of the Gillespie algorithm[3]. This algorithm has the distinct advantage of being mathematical exact, enabling accurate simulation of biological models.

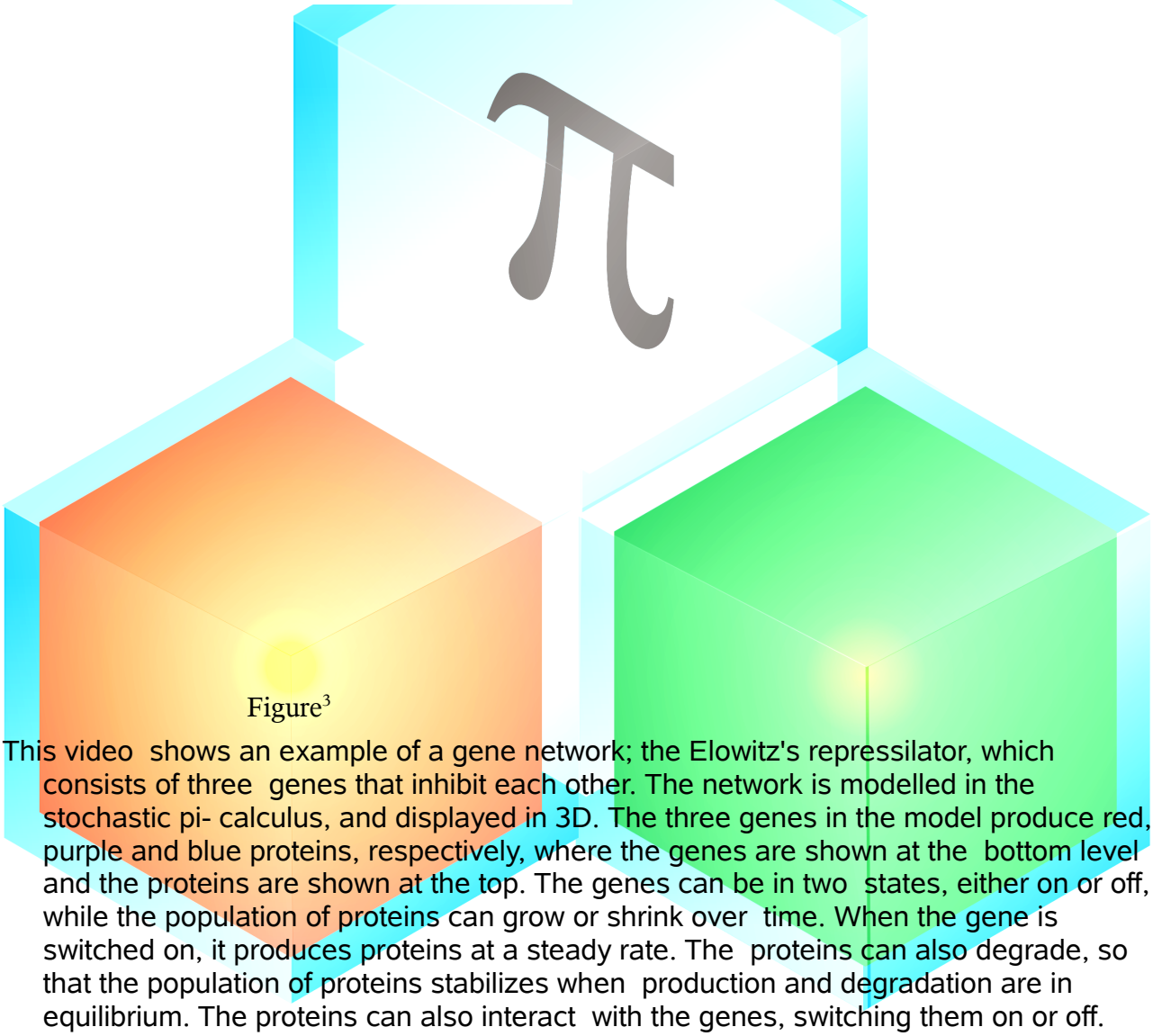
Andrew Phillips and Luca Cardelli designed a simulation algorithm (Stochastic π-Machine, SPiM) for the stochastic π-calculus based on standard theory of chemical kinetics [Gillespie 1977] where the probability of a reaction is proportional to the rate of the reaction times the number of reactants.

This simulator was proved and was determinate that is correct with respect to the stochastic π-calculus.



Stochastic π-Calculus in the Repressilator of Elowitz[2]





Figure[3]

This video shows an example of a gene network; the Elowitz's repressilator, which consists of three genes that inhibit each other. The network is modelled in the stochastic pi- calculus, and displayed in 3D. The three genes in the model produce red, purple and blue proteins, respectively, where the genes are shown at the bottom level and the proteins are shown at the top. The genes can be in two states, either on or off, while the population of proteins can grow or shrink over time. When the gene is switched on, it produces proteins at a steady rate. The proteins can also degrade, so that the population of proteins stabilizes when production and degradation are in equilibrium. The proteins can also interact with the genes, switching them on or off. The red gene produces proteins that switch off the purple gene, which produces proteins that switch off the blue gene, which produces proteins that switch off the red gene, completing the cycle.

The video shows how the gene network can produce oscillations in protein levels. Initially there is a large population of red proteins, which switch off the purple gene (1). The

2© 2000 Elowitz, M.B., Leibler. S. A Synthetic Oscillatory Network of Transcriptional Regulators. Nature 403:335-338

3 http://research.microsoft.com/~aphillip/talks/repressilator.wmv

blue gene then starts producing proteins, which switch off the red gene (2). Since no more red proteins are produced, the population of red proteins slowly decreases over time, until all of the red proteins are degraded (3). The purple gene then starts producing proteins, which switch off the blue gene (4). Since no more blue proteins are produced, the population of blue proteins slowly decreases over time, until all of the blue proteins are degraded (5). The red gene then starts producing proteins, which switch off the purple gene, completing the cycle.

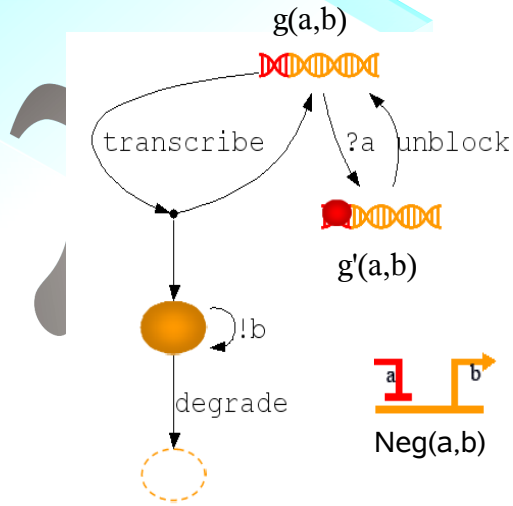
We can model the behavior of the repressilator in Stochastic Pi-Calculus language; here we have one gene of the three of the repressilator, modeled as follows:

Stochastic Pi-Calculus model:

$g(a,b) = ?a \,.\, g'.(a,b) + \tau_t \,.\, (P(b)|g(a,b)$

$g'(a.b) = \tau_u \,.\, g(a,b)$

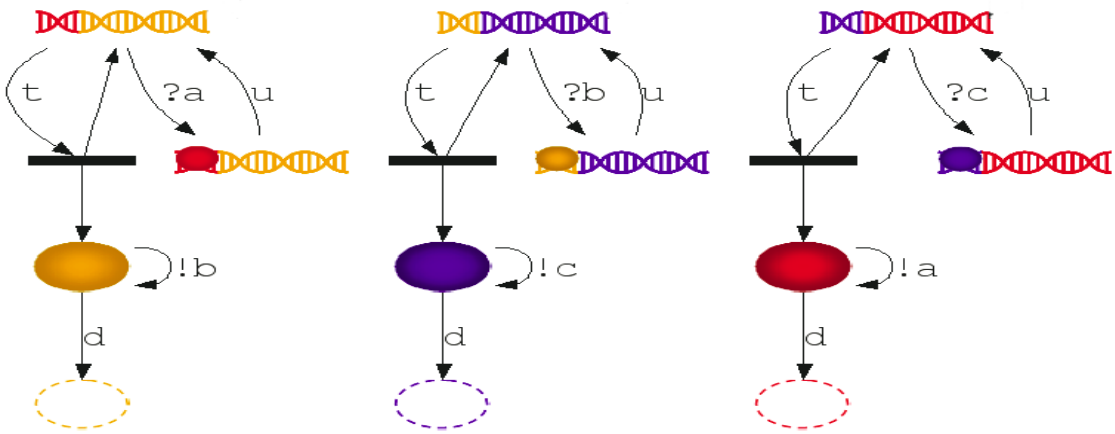
$P(b) = !b \,.\, P(b) + \tau_d \,.\, 0$



And the program on the SPiM for the repressilator:

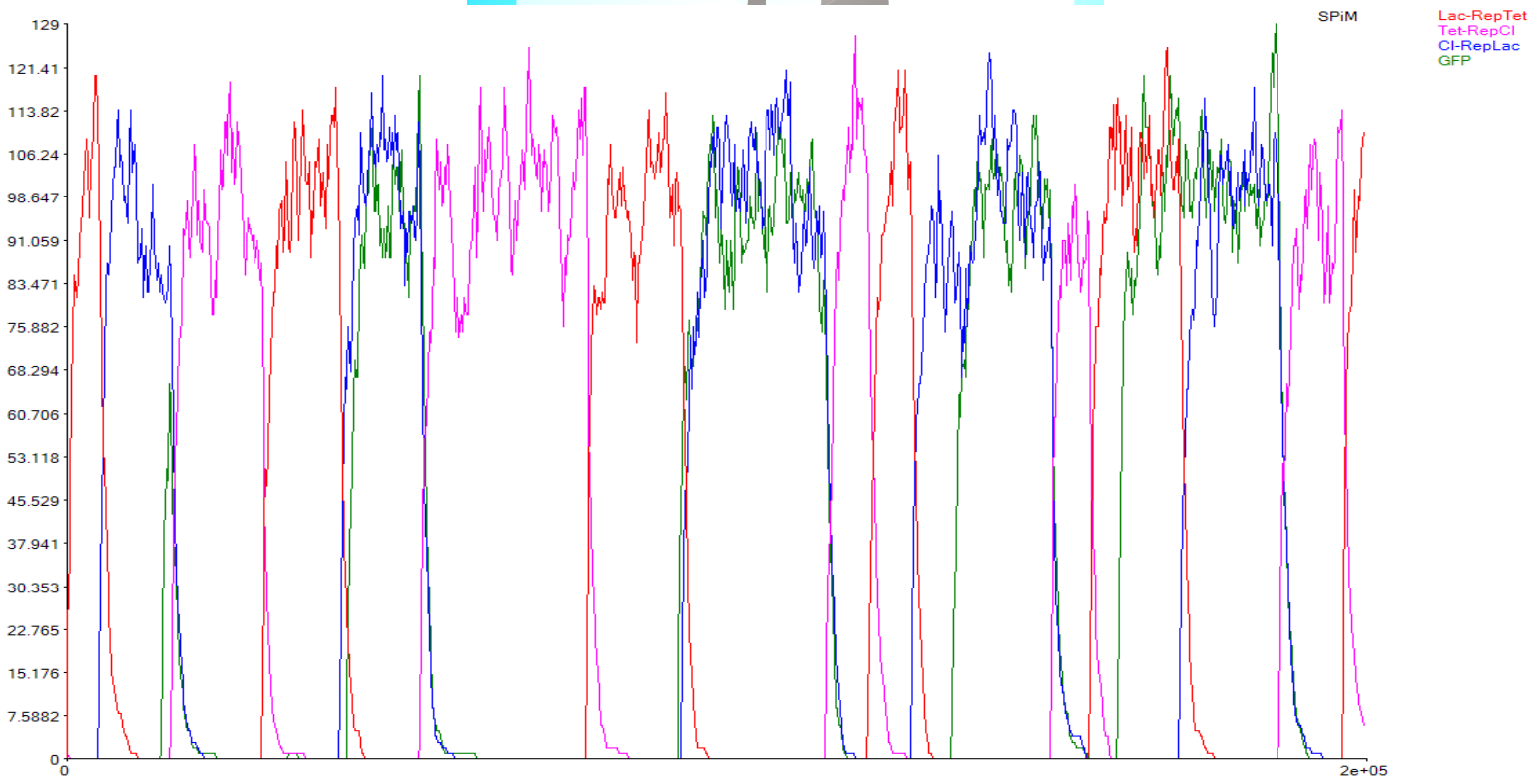
```
val transcribe = 0.1     val degrade = 0.001
val unblock = 0.0001   val bind=1,0

new a@bind:chan       new b@bind:chan
new c@bind:chan

let Neg(a:chan,b:chan) =
  do delay@transcribe;
     (Protein(b) | Neg(a,b))
  or ?a; Blocked(a,b)

and Blocked(a:chan,b:chan) =
  delay@unblock; Neg(a,b)

and Protein(b:chan) =
  do !b; Protein(b)
  or delay@degrade


run Neg(a,b) | Neg(b,c) | Neg(c,a))
```

We have to highlight that elowitz's repressilator has negative logical gates at each gene.



We can see above the repressilator with the three genes represented in the graphical Stochastic pi-calculus diagram.

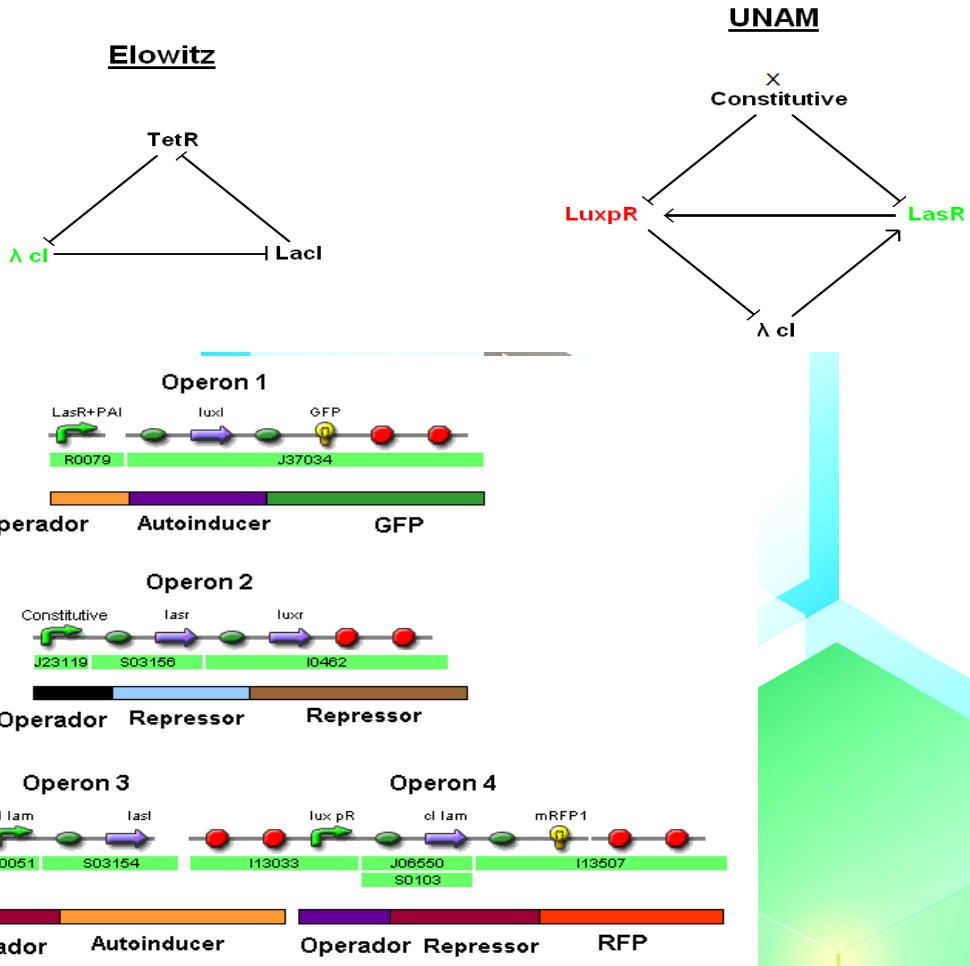
Network construction gives a high probability that the genes oscillate in a particular sequence.



Here we have the repressilator implemented in the SPiM of Andrew Phillips. With color red is represented the Lac Gene, with color Blue lambda CI Gene, with Magenta the Tet Gene; and with Green GFP Gene

GFP will be expressed only when Tet is blocked; then we can see that the most of the times it happens when lambda CI is switch on.

IGEM México Construction:



Repressilator vs. UNAM's Construction

Basically in the repressilator we can found three logical gates with negative control. Our construction consists of two negative gates and two positive gates.

The characteristics of the construction are as follows:

1. We have an X Constitutive gene that inhibits LasR and LuxpR genes.
2. The lambda cI gene is then opened and begin producing the autoinductor protein that switch promotes LuxpR
3. When LasR is unblock, it starts parallel production of two proteins, GFP and the protein that activates LuxpR.
4. LuxpR gets unblock and produce at the same time RFP and the inhibitor of the lambda cI gene.
5. When lambda cI get blocked it won't produce anymore the protein that activates LasR.
6. As the gene LasR is blocked, GFP stops and the production of the protein that unblock LuxpR stops.
7. Then LuxpR won't produce RFP and the inhibitor of lambda cI stops producing.
8. When the inhibitor of lamda cI degrades the cycle is finished and we returned to the original state

We present now the model in Stochastic Pi-Calculus of our construction:

$g(a,b) = ?a \,.\, g'.(a,b) + \tau_t \,.\, (P(b)|g(a,b)$

$g'(a.b) = \tau_u \,.\, g(a,b)$

$P(b) = !b \,.\, P(b) + \tau_d \,.\, 0$

$g(b,c) = ?b \,.\, g'(b,c) + ?e.\tau_t \,.\, (P(b)|GFP()|g(a,b))$

$g'(b,c) = \tau_u \,.\, (P(c)|g(b,c)|)$

$P(c) = !c \,.\, P(c) + \tau_d \,.\, 0$

$GFP() = \tau_d \,.\, 0$

$g(c,a) = ?c \,.\, g'(c,a) + ?f.\tau_t \,.\, (P(a)|RFP()|g(c,a)$

$g'(c,a) = \tau_u \,.\, (P(a)|RFP()|g(c,a))$

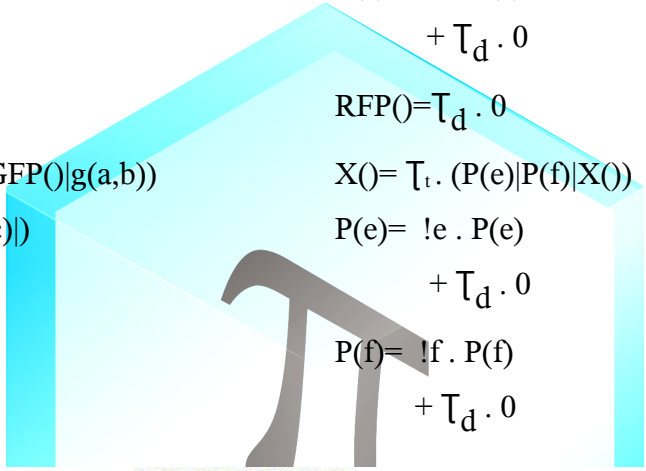
$P(a) = !a \,.\, P(a) + \tau_d \,.\, 0$

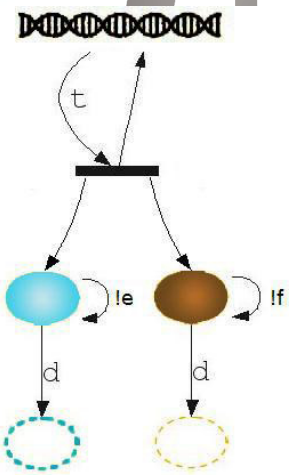
$RFP() = \tau_d \,.\, 0$

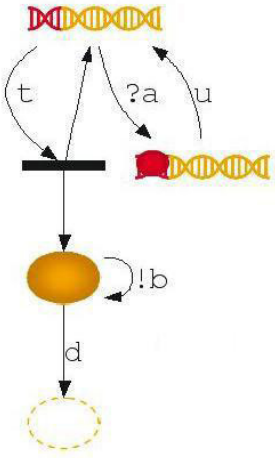
$X() = \tau_t \,.\, (P(e)|P(f)|X())$

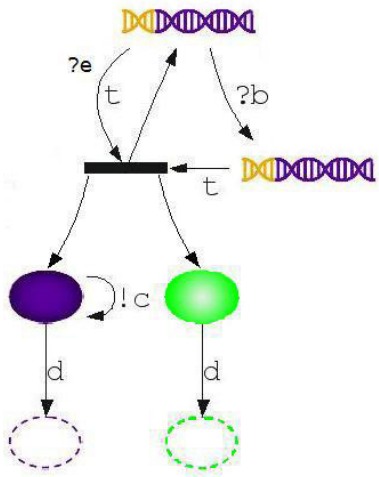
$P(e) = !e \,.\, P(e) + \tau_d \,.\, 0$

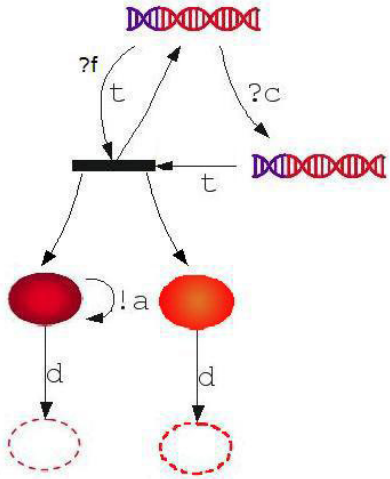
$P(f) = !f \,.\, P(f) + \tau_d \,.\, 0$











Graphical Representation of the model of our construction in Stochastic Pi-Calculus

Construction's Program Code for SPiM:

```
val t = 0.1          (*Decay Rate*)
val d = 0.001        (*Inhibition Rate*)
val u = 0.0001       (*Constitutive Rate*)
val bind = 1.0       (*Protein binding rate*)

let Cl(a:chan,b:chan)=
  do delay@t;(ALas(b) | Cl(a,b))
  or ?a;delay@u;Cl(a,b)
and ALas(b:chan)=
  do !b;ALas(b)
  or delay@d
new a@bind:chan
new b@bind:chan
new c@bind:chan
new e@bind:chan
new f@bind:chan

val t = 0.1      val d = 0.001
val u = 0.0001   val bind = 1.0

let Las(b:chan,c:chan)=
  do ?e;delay@t;(ALux(c)|GFP()|Las(b,c))
  or ?b; delay@t; (ALux(c)|GFP()|Las(b,c))
and ALux(c:chan)=
  do !c;ALux(c)
  or delay@d
and GFP()=
  delay@d
new a@bind:chan
new b@bind:chan
new c@bind:chan
new e@bind:chan
new f@bind:chan

val t = 0.1      val d = 0.001
val u = 0.0001   val bind = 1.0

let Lux(c:chan,a:chan)=
  do ?f;delay@t;(RCl(a)|RFP()| Lux(c,a))
  or ?c; delay@t; (RCl(a)|RFP()|Lux(c,a))
and RCl(a:chan)=
  do !a;RCl(a)
  or delay@d
and RFP()=
  delay@d
new a@bind:chan
new b@bind:chan
new c@bind:chan
new e@bind:chan
new f@bind:chan

val t = 0.1      val d = 0.001
val u = 0.0001   val bind = 1.0

let X()=
  delay@t;(RLas(e)|RLux(f)|X())
and RLas(e:chan)=
  do !e;RLas(e)
  or delay@d
and RLux(f:chan)=
  do !f;RLux(f)
  or delay@d
new a@bind:chan
new b@bind:chan
new c@bind:chan
new e@bind:chan
new f@bind:chan


run(Cl(a,b))
run(Las(b,c))
run(Lux(c,a))
run(X())
```

Figure 1

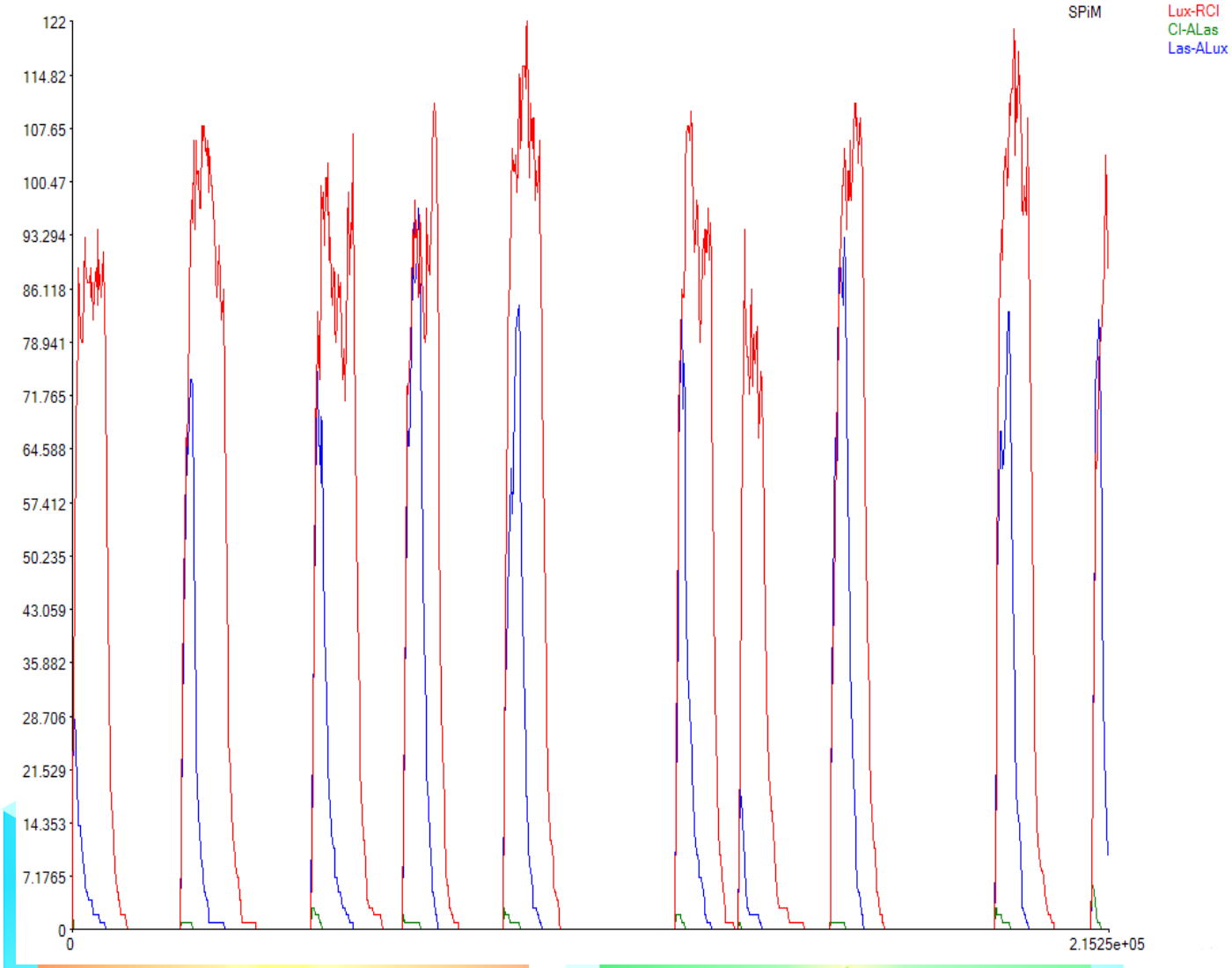


Figure 2

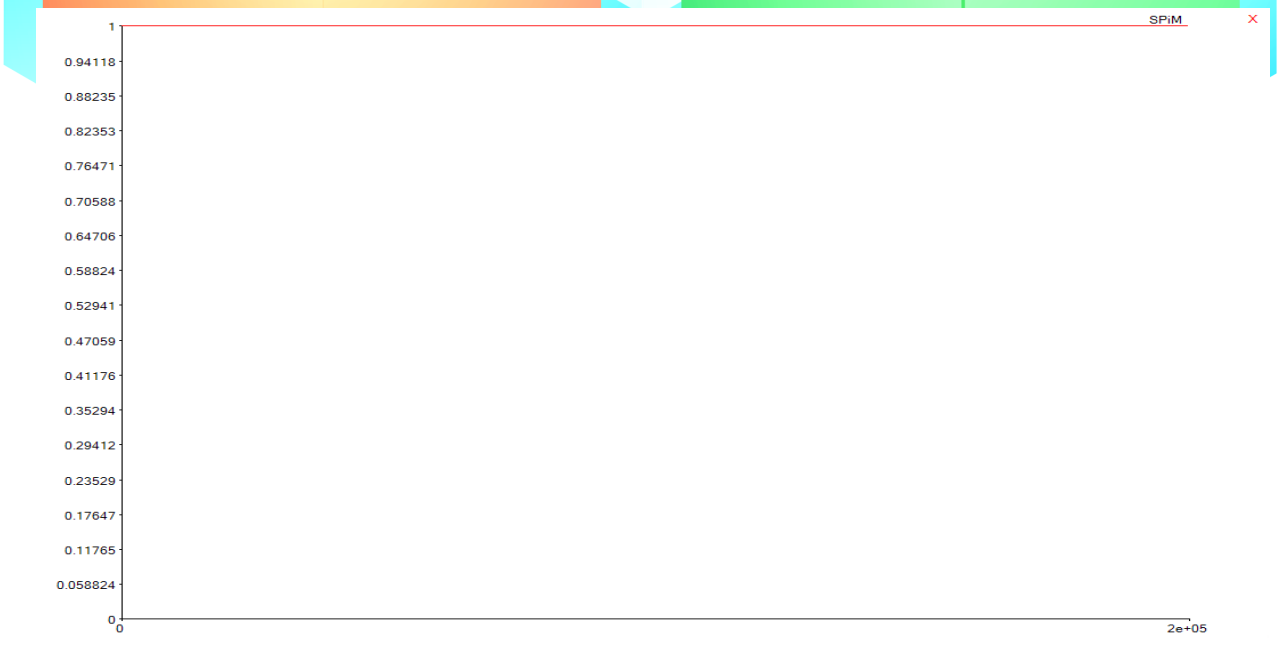
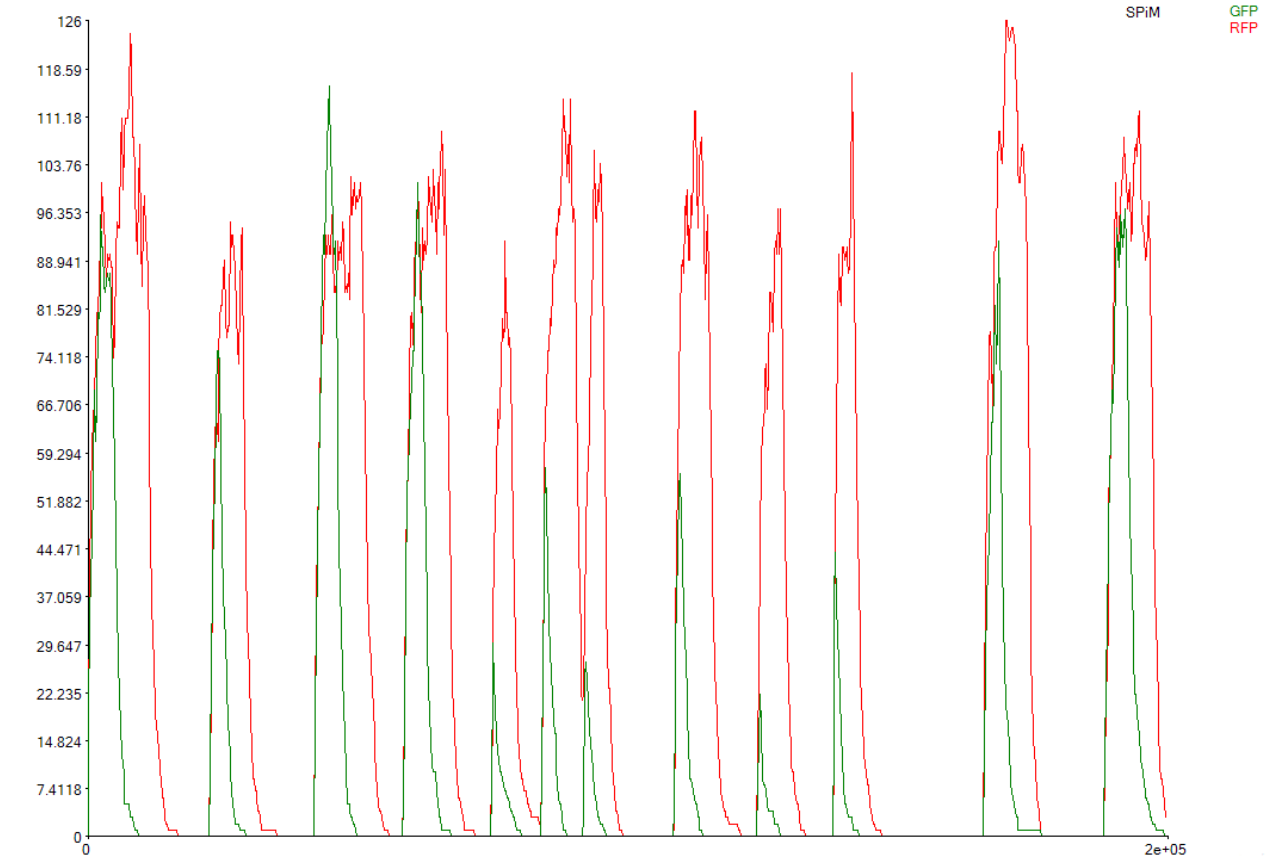


Figure 3



In Figure 1 we simulates the expressions of the genes LasR, LuxpR and lambda cl.

As results of these simulation we can say that the gene lambda cl is blocked most of the time; gene Las is expressed more than Gene lambda Cl. Finally gene Lux is the gene that is expressed the most.

Similarly, GFP and RFP in Figure 2 are expressed; but like Lux is highly expressed, RFP will predominate on the top of the oscillations.

Figure 3 shows that the X gene will be constant during the simulation time, then we have a functional circuit.

Conclusion:

As a conclusion of these simulations we have that our construction might be indeed functional, since the model predicts particular oscillations at protein levels. We pretend to corroborate these results at the laboratory; and see if the colony of engineered cells with these construction gets synchronised to obtain Turing Patterns or at least another interesting pattern.

References

1,[Priami et. al.,2003]Paola Lecca and Corrado Priami. Cell cycle control in eukaryotes: a biospimodel.In BioConcur'03. ENTCS, 2003.

2,[Priami et al., 2001] Priami, C., Regev, A., Shapiro, E., and Silverman, W. (2001). Application of a stochastic name-passing calculus to representation and simulation of molecular processes. Information Processing Letters, 80:25–31.

3. [Gillespie, 1977] Gillespie, D. T. (1977). Exact stochastic simulation of coupled chemical reactions. J. Phys. Chem., 81(25):2340–2361.

[Blossey et al., 2006] Blossey, R., Cardelli, L., and Phillips, A. (2006). A compositional approach to the stochastic dynamics of gene networks. Transactions in Computational Systems Biology, 3939:99–122.

[Guet et al., 2002] Guet, C.C., Elowitz, M.B., Hsing, W. & Leibler, S. (2002) Combinatorial synthesis of genetic networks. Science 296 1466-1470.

[Phillips, 2006] Phillips, A. (2006). The Stochastic Pi-Machine: A Simulator for the Stochastic Pi-calculus.

[Phillips and Cardelli, 2004] Phillips, A. and Cardelli, L. (2004). A correct abstract machine for the stochastic pi-calculus.

Phillips and Cardelli, 2005] Phillips, A. and Cardelli, L. (2005). A graphical representation for the stochastic Pi-Calculus.

[Phillips et al., 2006] Phillips, A., Cardelli, L., and Castagna, G. (2006). A graphical representation for biological processes in the stochastic pi-calculus. Transactions in Computational Systems Biology, 4230:123–152.