Name: ______________________________

BIOINF 525 Module 3
Lab #3
4/6/2017

Please complete the exercises below. Throughout the lab sessions for this module, we will use the following notation:

Plain text indicates actions that should be taken
Italicized text indicates explanatory material
Bold text indicates a point where a written response is required

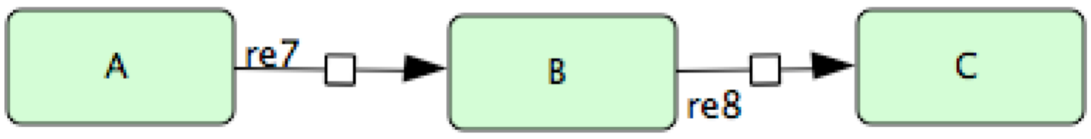
We will pause at each point with a long horizontal line for discussion. If you reach those pause points but the rest of the class is still working, I encourage you to try out some variations on whatever step you were currently working on.

Exercise 1: Kinetic modeling of regulatory networks

Go to File->New and generate a new model (you can call it something like simple_example)

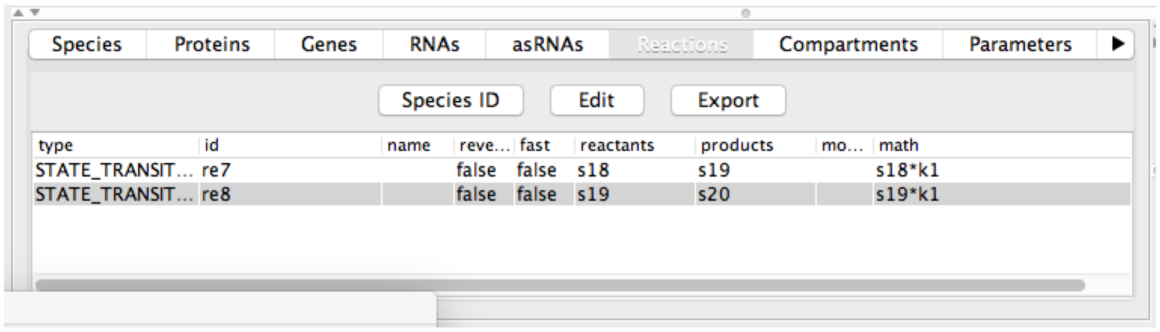
As a simple example of a generic pathway, we will just create three proteins, A, B, and C, where A is converted to B at a constant rate, and B is converted to C at a constant rate.

Draw three protein nodes, labeled A, B, and C. Then define state transitions between them, such that A -> B and B -> C. To make a particular type of reaction, first select the arrow type that you want in the top menu, then click on the center of the source node, then the destination node. Your network should look like the image below (the reaction numbers may be different).



At this point we have the network layout, but no reactions defined yet. Right click on the arrow connecting A->B, and go to Edit KineticLaw. Set it to use Mass Action Kinetics and set the constant to 0.2. Then click update, and close the menus. Do the same for the B->C reaction, but set the constant to 0.1. This means that A will be converted to B at a rate of 0.2 times the current concentration of A, and B will be converted to C at a rate of 0.1 times the current concentration of B.

At this point you should see a display like the following in the reactions tab of the bottom menu:



type	id	name	reve…	fast	reactants	products	mo…	math
STATE_TRANSIT…	re7		false	false	s18	s19		s18*k1
STATE_TRANSIT…	re8		false	false	s19	s20		s19*k1

Write the differential equations that govern the behavior of the network here as it is currently set up:

A' = ________________ **B' =** ______________________ **C' =** ____________________

Now we can simulate the behavior of our network, and observe how it changes as we change either initial conditions or parameters.

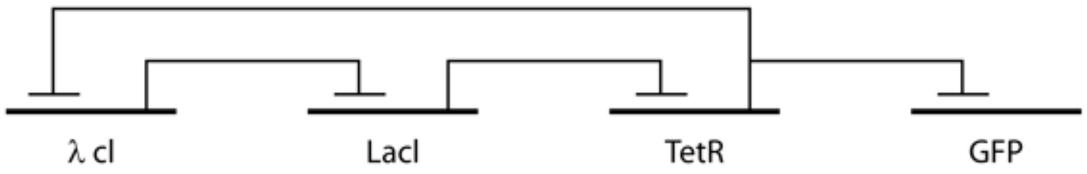
Go to Simulation->Control Panel and bring up the simulation menu. Set the initial quantity of A to be 1.0, and leave the others at 0. Click "Execute" to simulate the model. **Record the (approximate) peak concentration of B, and the time that it takes for 90% of the mass of the system to be converted to C.**

Now we will change the initial conditions of the system and see how the dynamics change. Set the initial quantity of A to 2.0, and rerun the simulation. **Record the same quantities that you did before, and compare them to the results from the first run. Are you surprised by what you find?**

Keeping the same initial conditions, change, the rate constant for the B->C conversion from 0.1 to 0.2. **What do you think this will do to the observables that we had recorded relative to the previous simulations?**

Now run the simulation. **Record the resulting values for the peak B concentration and time to reach 90% maximal C. How do the results compare with your expectations?**

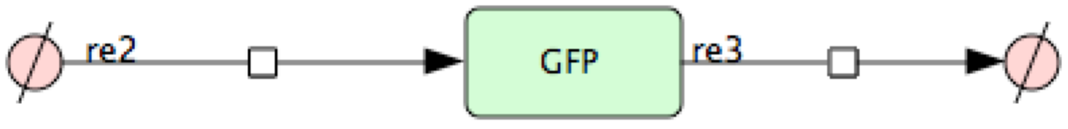
We will now design a toy example of a network called a repressilator. This represents one of the first efforts to make a completely synthetic regulatory network for a specific purpose, in this case, to have long term oscillatory behavior in the expression of a GFP reporter. The design of the network is



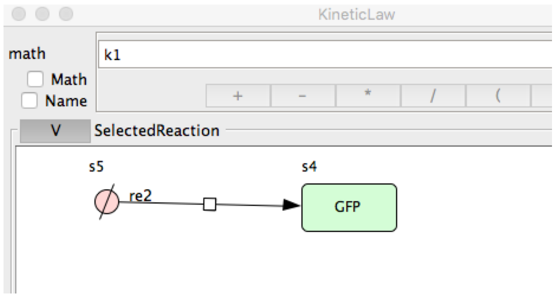
(this was originally described in detail in Nature 403(6767):335-8, 2000)

Close the simulation control panel and go to File->New to create a new network for your repressilator. Note that for the purposes of this example, we're going to ignore transcription/translation and just represent each entity as an abstract object that can repress synthesis of its target.

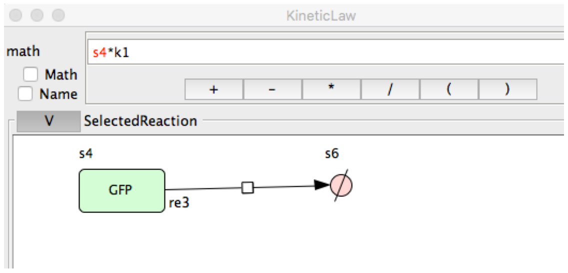
First let's make a network that just shows constitutive expression of GFP. Make a "GFP" protein node, and a 'degraded" note (which effectively represents nothing. Then generate a state transition from the 'degraded' node to GFP – this represents translation of GFP. Likewise, generate a state transition from GFP to another degraded node – this represents degredation of the GFP.



Now use the "edit kineticlaw" interface to set up rates for both reactions. For the synthesis, use a NonPredefinedFunction and set the rate to k1. For the degredation, use Mass_Action_Kinetics. Use a rate constant of 1.0 for the synthesis reaction and 0.1 for the degredation reaction.



math: k1



math: s4*k1

What is the differential equation for GFP expression being implemented by this approach?

d[GFP]/dt =

What steady state value will GFP attain?

Now open the simulation Control Panel and simulate the network's behavior, starting with zero concentrations of GFP. Note that you will probably want to uncheck the boxes on the right for the source and sink nodes, which are not informative to look at.

What is the actual steady-state GFP concentration that you observe?

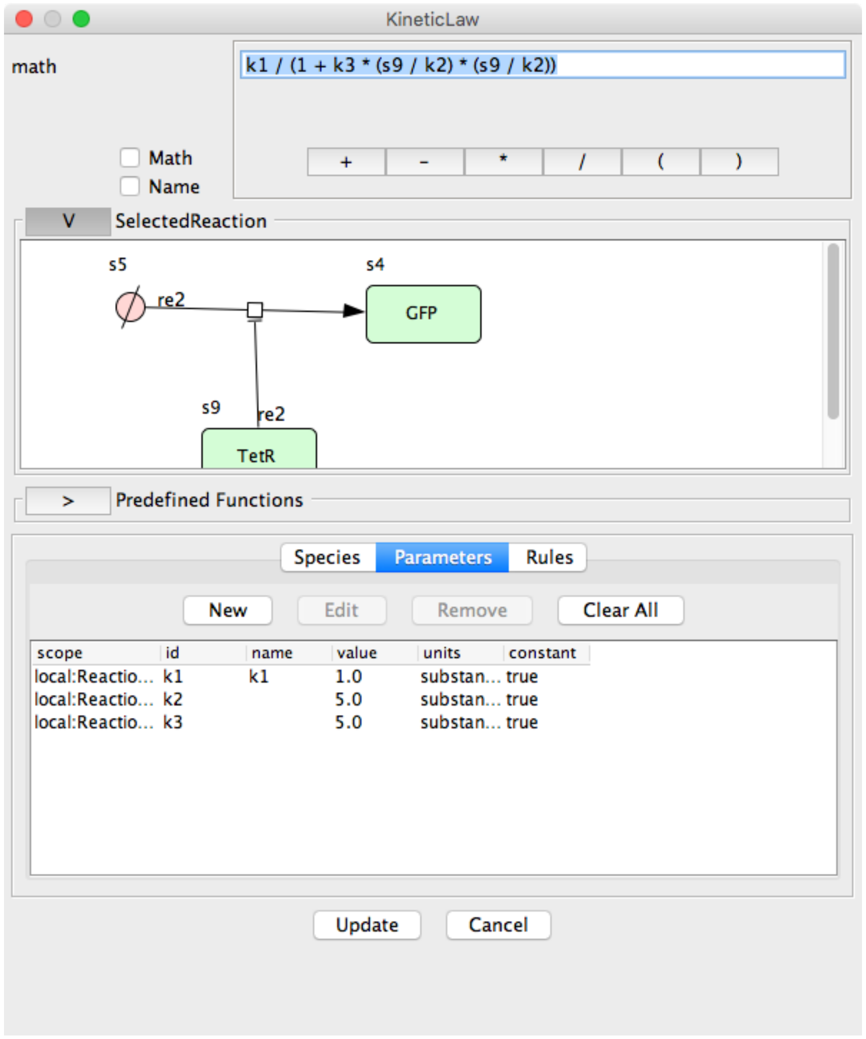
Now, to build up the repressilator, we need to apply some regulation to the GFP synthesis. Define another network node called TetR, and set up its synthesis and degredation to be identical to GFP. Finally, draw an "Inhibition" arrow between TetR and the GFP synthesis reaction.

To actually add repression, you need to edit the kinetic law for GFP production to be inhibited by TetR. Here we will use the form

k1 / (1 + k3 * (s9 / k2) * (s9 / k2))

Here k1, k2, and k3 are constants, and s9 is the identifier for TetR (this may differ in your setup – look at the "Species" tab at the bottom of the KineticLaw menu to see the variable name for each concentration). Set k1=1, k2=5, k3=5.

Your kinetic law should now look something like:

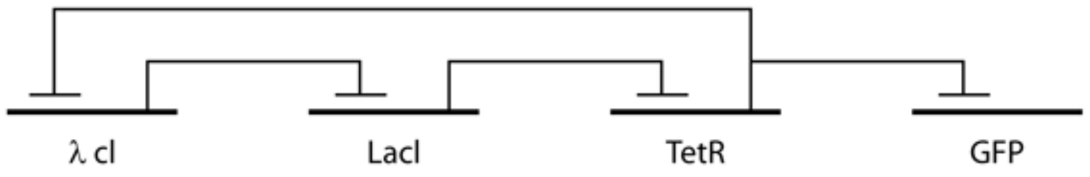


k1 / (1 + k3 * (s9 / k2) * (s9 / k2))

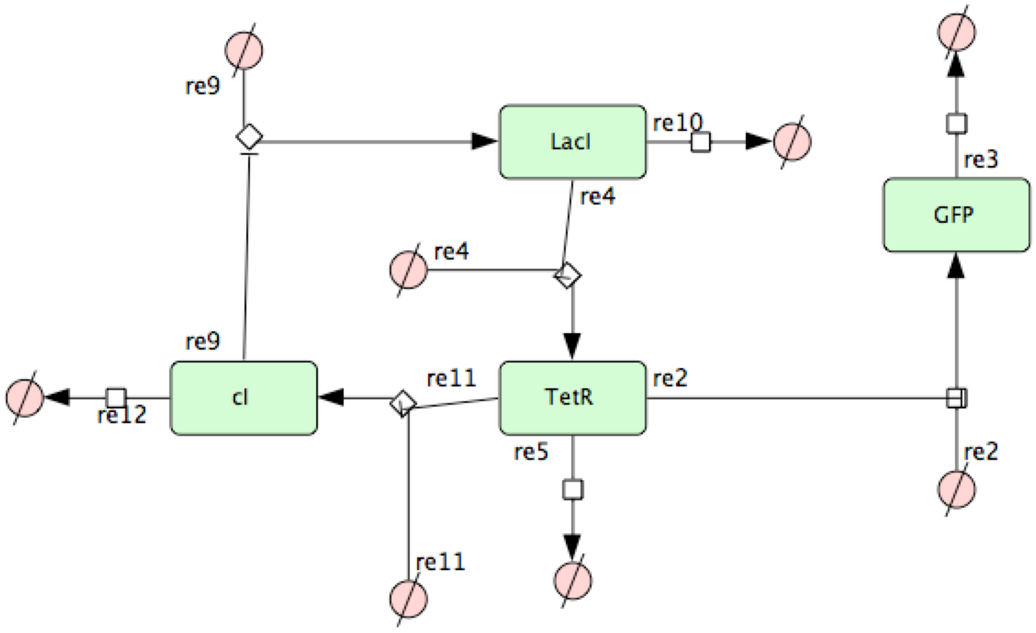
scope	id	name	value	units	constant
local:Reactio...	k1	k1	1.0	substan...	true
local:Reactio...	k2		5.0	substan...	true
local:Reactio...	k3		5.0	substan...	true

Now simulate the network's behavior using the ControlPanel as before, starting with all concentrations at zero. **What is the maximum concentration of GFP that is achieved?**

Using the same protein-only setup for the other components, assemble a complete repressilator, following the diagram below:



Set all of the synthesis and degradation terms to have the same logic and parameters that you used for the synthesis of GFP (but driven by the appropriate repressor). Then, to let the GFP part be more visible, raise the k1 constant to 3 instead of 1 in the GFP synthesis reaction. Your model should look something like:



In your model network, what is the differential equation for the rate of change of LacI concentration?

D[LacI]/dt =

Go to the simulation control panel, set the "End Time" and "Num of points" both to 250, set the initial concentration of TetR to 1, and then run the simulation. As usual, remove the source/sink nodes that you're not interested in and then re-zoom.

Does your network actually undergo any kind of oscillation? Does it eventually reach a steady state? If so, what does that steady state look like?

Try changing the initial concentrations of the three repressors to a variety of different values. **How stable are the dynamics of the system to different initial concentrations? What about its endpoint? Can you find initial conditions where no oscillation occurs?**

Now, try changing the *parameters* of the network, first by increasing the rate constant for the degradation of TetR from 0.1 to 0.5. **What do you think this will do to the steady state? (What will the rank ordering be of repressor abundances)**

Now run the simulation with the altered parameters. **What actually happens?**

Next, try changing all of the degradation constants for the repressors to 0.01, and then simulate the system again. (You will probably want to increase the total number of steps to 2000 or so). **How is the behavior changed by making the repressors more stable?**

CellDesigner also allows you to download a variety of models from databases such as the ones that we discussed in class. Go to database->Import model from biomodels.net, search for repressilator, and load the Elowitz2000 model. Simulate it for at least 1000 steps. **How does its behavior compare with that of your model network?**

Exercise 2: Flux balance analysis

We will use the Sybil R package to perform several variations of flux balance analysis on the core metabolic network of *E. coli*. The examples shown here are based on the Sybil reference paper: BMC Sys. Biol. 7:125, 2013.

Open an R session, load the Sybil library and glpkAPI package, and then load a model of the E. coli central carbon metabolism network with the command

```
data(Ec_core)
```

You can then fit an FBA model to determine the fluxes through this network with the command

```
fba <- optimizeProb(Ec_core, algorithm = "fba")
```

We are first interested in looking at fluxes into and out of the cells. We can find a list of all of the exchange reactions with

```
ex <- findExchReact(Ec_core)
ex
```

The resulting listing gives boundaries on the flux of several metabolites into (negative) and out of (positive) the cells.

In the initial formulation of the model, what carbon sources are available to the cells?

Now, we can get the distribution of fluxes from the fitted model with the command

```
fd <- getFluxDist(fba, ex)
```

The fluxes can be observed in a simpler form using the command

```
getNetFlux(fd)
```

and we can observe the value of the objective function (in this case, biomass production) with the command

```
mod_obj(fbap)
```

Simply by looking at the fluxes from this model, what factors aside from nutrient depletion might halt the growth of E. coli cultures?

Now we will use FBA to predict how the growth of E. coli will change if we alter the carbon source that they are using.

We will use the Ec_core model as the basis for a new model where we have replaced the main carbon source with pyruvate:

```
Ec_core_pyr = changeBounds(Ec_core, ex[c("EX_glc(e)", "EX_pyr(e)")], lb = c(0, -10))
ex.p<- findExchReact(Ec_core_pyr)
ex.p
```

Fit a new FBA model for the cells growing on pyruvate as a sole carbon source. How does the growth rate compare to that observed for growth on glucose?

Has there been any notable change in the net import or export of resources for cells growing in pyruvate? (Aside from the change in primary carbon source being imported)

Now we can look at the behavior of the cells in the face of a genetic perturbation; we will do so separately using glucose and pyruvate as a carbon source.

A knockout can be simulated by adding a "gene" parameter to optimizeProb; genes can be looked up directly in the model files or on a database like ecocyc.org (but don't do that just yet).

Fit models for knockouts of the gene b4025 as follows (note that this gene naming is the standard systematic notation for E. coli):

```
optimizeProb(Ec_core, gene = "b4025", lb = 0, ub = 0)
optimizeProb(Ec_core_pyr, gene = "b4025", lb = 0, ub = 0)
```

(for each fitted model, we want to look at the value of the objective function, which gives the growth rate)

Gene b4025 is phosphoglucose isomerase, responsible for the second step in glycolysis. **Are you surprised by your results?**

We can use FBA to perform a more systematic analysis of lethal genes and how they change under different conditions. The central function for this approach is the oneGeneDel function, which will fit the model with individual simulated deletions of each gene. For the original, glucose-using model, this can be done with:

```
ref.glu = optimizeProb(Ec_core)
opt.glu = oneGeneDel(Ec_core)
essential.genes = lethal(opt.glu,wt=mod_obj(ref.glu))
allGenes(Ec_core)[essential.genes]
```

A list of essential genes will then be printed.

Identify the essential genes for growth on glucose and on pyruvate as carbon sources. Record any that differ between the two carbon sources: