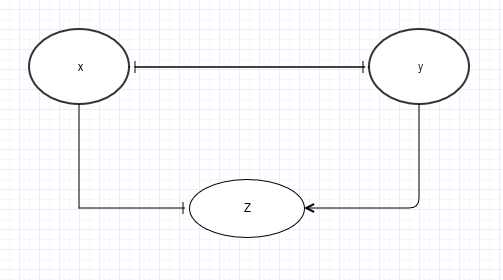
**Homework 2** (chapters 3-6 of U. Alon “An introduction to Systems Biology”).

Date: 2014-05-05

**Part 1**

*Consider two genes which repress each other, and , and which both regulate a third gene as and . Determine the steady states. Describe the responses to a change of the state variables (X and Y, one at a time), starting from a steady state.*



**Steady states:**

In our case we have a double-negative feedback loop, which is made of two repression interactions. An output gene Z is regulated as shown in the figure. We have two steady states, for which either gene is ON in the double-negative loop. The following table sums it up.

|  |  |  |  |
| --- | --- | --- | --- |
|  | **X** | **Y** | **Z** |
| **Steady-state 1** | ON | OFF | OFF |
| **Steady-state 2** | OFF | ON | ON |

In one stable state X is ON and Y is OFF, so that protein X represses Y expression and prevents it from being produced. The other stable state is the reverse: X is OFF and Y is ON. Thus, the double-negative loop expresses either X or Y. This is useful when genes regulated by X belong to different cell fates than the genes regulated by Y.

**Responses:**

The basic equation describing the evolution of a protein product X is:

Therefore, to determine the steady states for, let’s say X, we have to solve (1) for .

Now we have and we get:

Explaining the result: The higher the production rate , the higher the protein concentration reached, . The higher the rate of decay , the lower is .

Starting from the Steady-state 1, if we change the state variable X so that it’s signal, let’s say , disappears and its production stops, we will get an exponential decay of X concentration. This is better understood if we solve equation (1) for X with .

Thus,

A good measure for the speed at which X levels change is the **response time** i.e. the time to reach halfway between the initial and final levels in a dynamic process. We solve **(ε)** for and we get:

Now that Y is no longer repressed, given a signal , it will begin to accumulate. The concentration of Y rises from zero and gradually converges on the steady-state . According to the book, this convergence is shown in the following formula (which is derived from equation (1) ):

Moreover, the response time is once again.

In the opposite case, something similar happens. If a signal suddenly appears, then X will begin to accumulate, in a similar manner to (ζ) and it will begin to repress Y, which in turn will see an exponential concentration decay similar to (ε).

*(****NOTE:*** *See Appendix in the end of the document for comments on the input functions)*

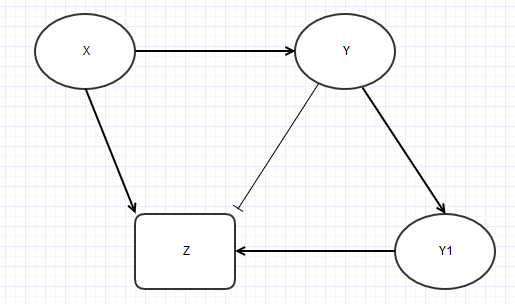
**Part 2**

*Consider four genes X, Y, and Z that interact as two interconnected feed-forward loops: and such that (X,Y,Z) is an incoherent FFL of type I1 in Alon’s classification (Fig 4.3 page 47); and such that (Y,Z) is an incoherent FFL of type I3.*

*Consider as in the general set-up used by Alon that the three genes X, Y and are driven by three input signals and . When the input signals are off (value 0) the corresponding protein is in its inactive form and cannot act as a transcription factor, even if present in high enough concentration. When on the other hand the signals are on (value 1) the corresponding protein is in its active form and can act as a transcription factor, if in high enough concentration.*

*The output node (Z) is now driven by three inputs: X, Y and . Choose one way of combining the inputs which makes Z a non-canalizing Boolean function of the inputs. We repeat from the lectures that a non-canalizing Boolean function is one which fully depends on all its arguments.*

*Describe precisely which Boolean function you chose and show that it is non-canalizing. Show how the expression of Z changes in response to steps in signals and . Comment.*

****

|  |  |  |  |
| --- | --- | --- | --- |
| X |  | Y |  |
| 0 | 0 | 0 | **1** |
| 1 | 0 | 0 | **1** |
| 0 | 1 | 0 | **1** |
| 0 | 0 | 1 | **0** |
| 1 | 1 | 0 | **1** |
| 1 | 0 | 1 | **0** |
| 0 | 1 | 1 | **0** |
| 1 | 1 | 1 | **1** |

I claim that the logical expression **,** which is a combination of the three inputs in our graph,makes Z a non-canalizing Boolean function of the inputs. It is non-canalizing since, none of the inputs can affect the outcome of the output individually, namely, our function is not canalizing towards any of the inputs. I show the truth table beside, as a simulation of the network. The value ‘1’ should translate as “Protein’s corresponding signal is present, thus the protein is in its active form” and the value ‘0’ should denote the opposite.

**Part 3**

*In this part it is sufficient to use logic input functions of the response of a downstream element to its net input. The net input should however be taken as a weighted sum of the upstream elements.*

*Consider a model of a three-level signal transduction pathway with two input signals (e.g. ligands) and , two “MAKKK” kinases and , two “MAPKK” kinases and , two “MAPK”-level elements and , and then an output element W, e.g. a transcription factor.*

*The net input to is and the net input to is the net input to is , the net input to is , the net input to is , the net input to is , and the net input to W is .*

*For the kinase , the two weights and must all be non-negative, and similar for all the other kinases, while for a phosphatase all weights must be non-positive. The thresholds for activation/repression are for etc. and the base-line dephosphorylation rates are etc..*

*Suppose that the two “MAPK”-level elements and are kinases also.*

*Can you choose weights such that this three-layer perceptron computes the logical function XOR and EQ? If you cannot do so, can you give a reason why this is not possible?*

*Is the answer different if either or both of and act as a phosphatase?*

*Define what you consider as in- and outputs for these biochemical logical functions (e.g. what corresponds to TRUE and FALSE in your system) and try to use as few free parameters as possible.*

|  |
| --- |
| ***Figure 1. XOR decision boundary example*** |

**Answer:**

The XOR and EQ logical functions, both are complex functions which are not linearly separated by the perceptron in two dimensions. Therefore, they need to be approached by a multi-layer perceptron which is proved to create convex (more complex) decision surfaces.

The decision surface for XOR can be seen in Figure 1 (for EQ it is similar, perpendicular to XOR’s) as the grey area between the two decision boundaries. In our case, we can see that there is no way to choose weights, for and acting as kinases, so that this three-layer perceptron computes the logical functions XOR or EQ. We need some inhibition, namely negative weights in our multi-layer perceptron, otherwise the weights will “push” our decision boundaries continuously and they will cover all our plane (the weights are the ones that position the decision boundaries in certain slopes/angles).

Just like the book on page 114, figure 6.13, we need a grey/active area for our output **W**, when either or are highly active, but not both, when talking about the XOR function. As far as the EQ function is concerned, we need a grey/active area for our output, when both and are active, but not when either of them is. Note that, “TRUE” value denotes the fact that the weighted sum of the two input nodes is greater than a threshold value (which could be approximately 1) of the next layer’s node to which they are inputs. “FALSE” means that the weighted sum did not overcome the threshold. In our case, by inputs we mean input kinase activities as well as the threshold, which is also discussed in terms of concentration of a phosphorylated kinase.

**APPENDIX**

Since we are talking about a repressor (X and Y form a negative feedback loop), we have the following models for our input functions:

|  |  |  |
| --- | --- | --- |
| **Logic input function** | **Michaelis-Menten function** | **Cooperative function of Hill coefficient 3** |
|  |  |  |

The first model (Logic input), is often useful to show the essential behavior of the input functions and retain mathematical clarity, whereas the next two models are used to describe our input functions in more graded detail. The hill coefficient (n=3 for the third model) defines the steepness of the S-shaped function with high *n* resulting in more steep curves. If n=1, then we get the Michaelis-Menten function. Here is an example graph from the book:

|  |
| --- |
| http://openwetware.org/images/thumb/b/be/Repressed_Hill_Function.png/300px-Repressed_Hill_Function.png |

The Hill function, in the limit when , becomes a step function, which is what the Logic Input function looks like graphically. Therefore, we could say that the logic input function is an approximation of graded functions like Hill functions.. The graded input functions show expression as soon as X\* appears, whereas the logic input function shows expression only when X\* crosses the threshold K. So, we conclude that we basically use the same function with an ‘n-coefficient’ (the dynamics are quite similar), which varies as to the level of detail in which we want to describe our system.