BioModelAnalyzer

http://biomodelanalyzer.org/

To use computers to understand cell behaviour we need to build a **model** - a mathematical representation that describes how proteins and genes talk and regulate one another and change over time. Here we have used a web-based tool called the BioModelAnalyzer to build such models. This allows you to draw the proteins, genes, metabolites and cells on a blank canvas and then give each a function to describe how it changes over time.

Once you have a model, you can then analyse it using the buttons on the right hand-side of the screen.

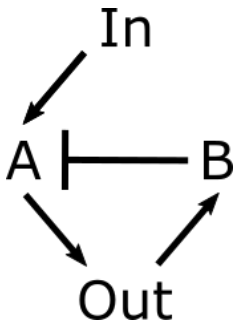
In this set of exercises, we introduce you to the interface of the BioModelAnalyzer and then show you three different ways that it is used to understand biological models:

- How to tell which network correctly describes the biological phenomena
- How metabolism mutations can change cancer metabolism
- How a model of leukaemia can be used to look for new drug combination treatments

In this word document we have embedded models for you to use in the BioModelAnalyzer. To open them in the tool, left click to select the json icon, copy and paste the model into a folder on your computer. This model can then be loaded in the tool by opening the model tab (on the left), clicking import, and selecting the file.

Using the BioModelAnalyzer Interface

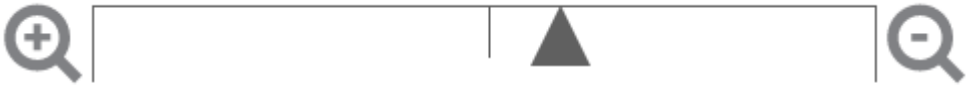
Here will we try make a small model of a negative feedback loop (below) in the interface, and test how it behaves. This type of loop leads to oscillations (i.e. cycles) under some conditions.



Building models and moving around the canvas

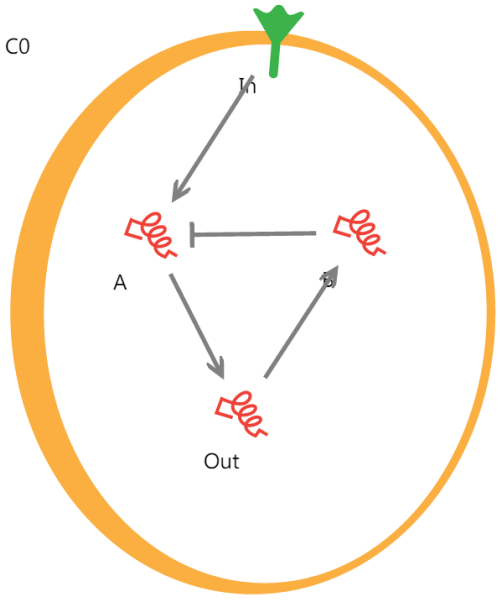
At the top of the canvas are the model building tools. Drag **cells** (orange) and **extracellular proteins** (grey) to the canvas to add them. Drag and drop **intracellular proteins** (red) into the cells, and **membrane proteins** (green) to the edge of the cell. Finally, click the **activation** or **inhibition** icons, and draw links between proteins in your model.

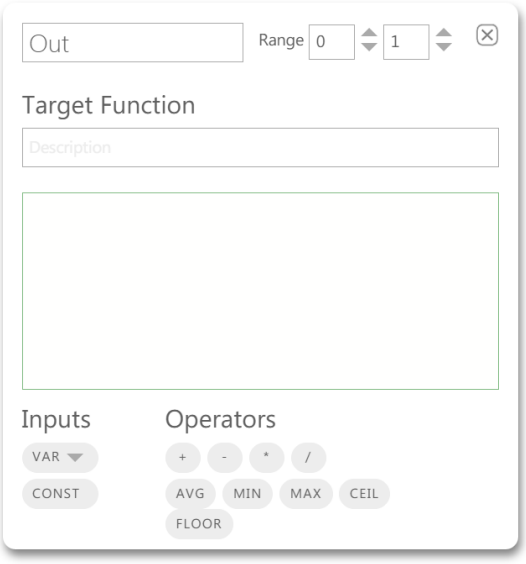
To move around the canvas, click the **selection** tool and drag on empty space. **Undo** or **redo** any mistakes you make!



Zoom into parts of the model with the **zoom bar**.

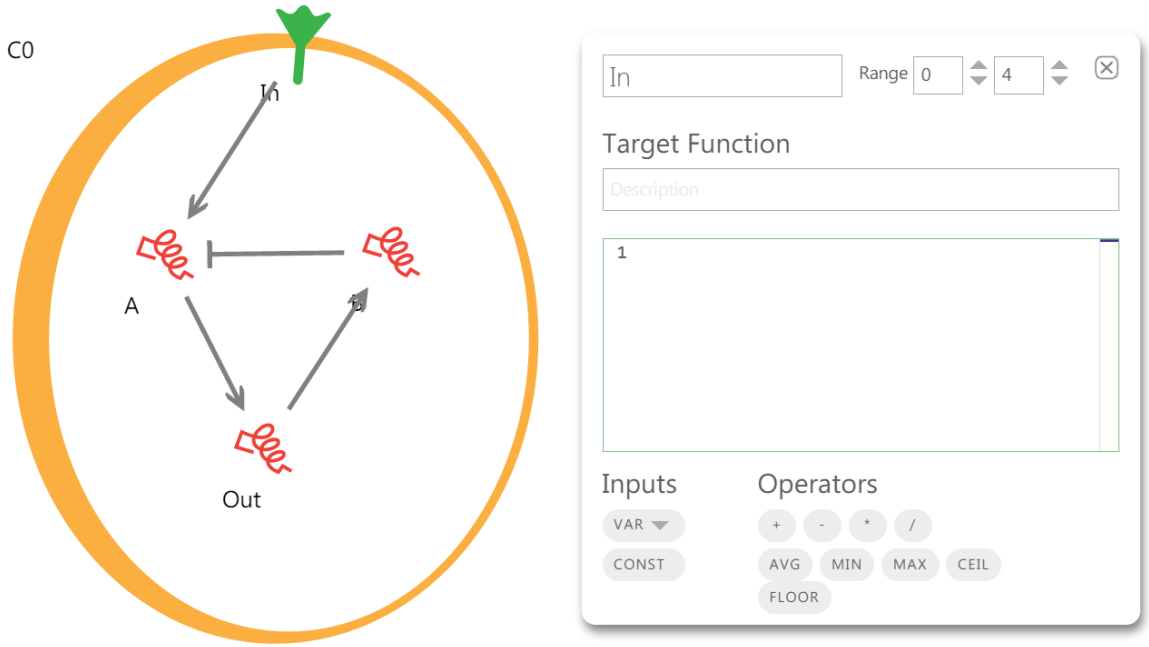
To build the negative feedback model add a cell to the canvas and add 3 intracellular proteins and a receptor. Connect them with the appropriate arrows and name the variables by right clicking and entering the protein name in the dialogue box. Leave the box labelled “Target Function” blank for now. By default, the function is the average of the positive inputs minus the average of the negative inputs.



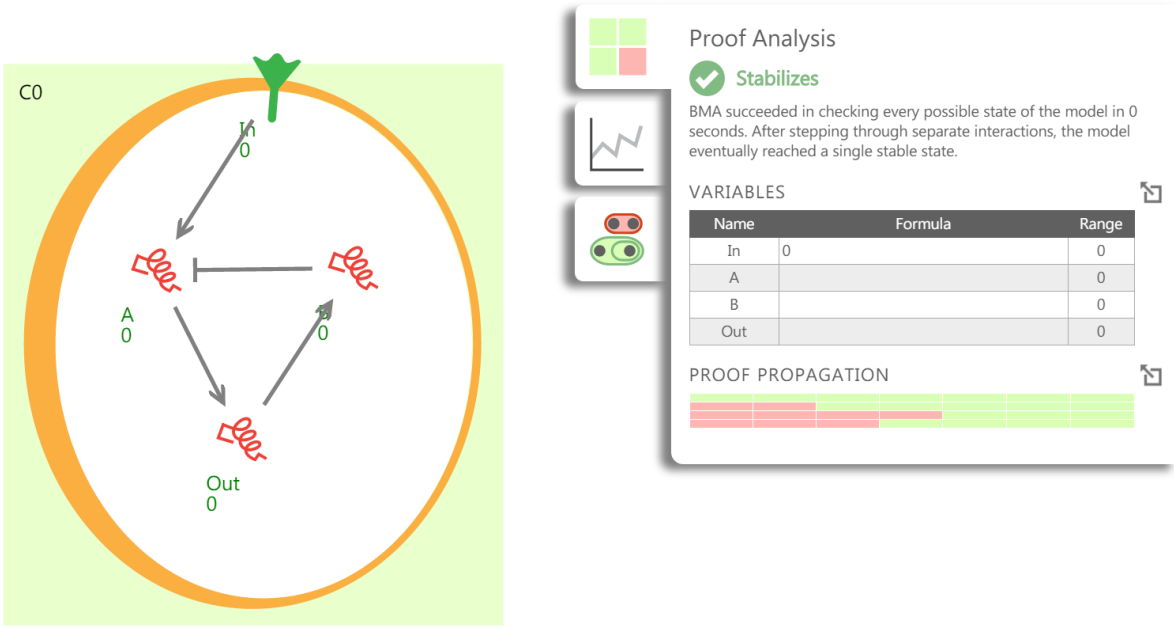


Testing your model

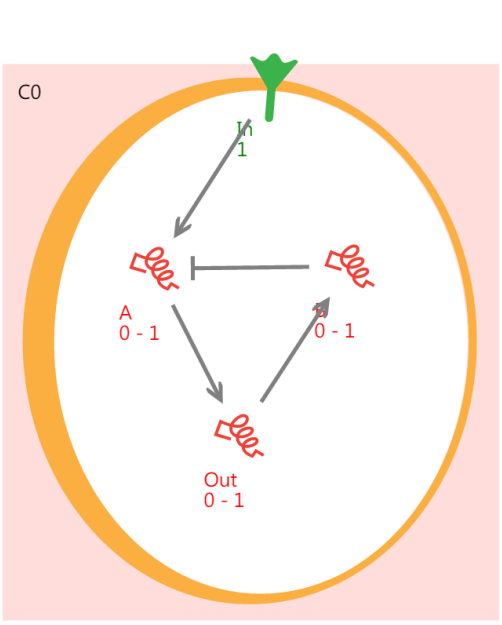
To test your model, use the buttons on the right-hand side of the screen. The button above will apply **stability analysis** to see how the model behaves. Stability analysis tells you how **all** simulations end- if you can prove your model is stable then it means that all simulations end with one set of values. When you click the button, it runs a fast analysis to prove stability. If this fast approach is not able to prove stability, clicking on further testing at the bottom of the pull-out will do a more complex set of tests to either prove stability or return examples of bifurcations (simulations can end in more than one way) or oscillations (the system can enter a loop). We will use this a lot in the examples below to analyse our models behave correctly.

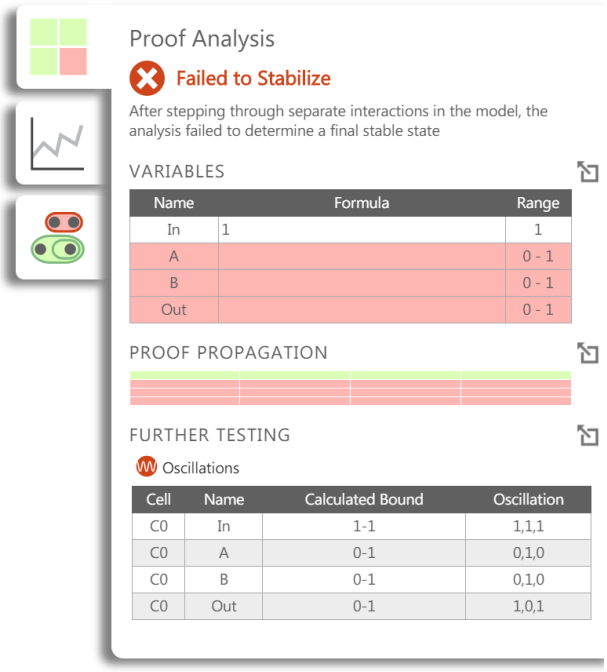


If we test this model we can find that it is stable- that is it always returns to a single state and does not oscillate.



To create oscillations, we need an input to the system. Open the “In” variable and set the target function to 1, and retest. Now click further testing to find the oscillation.





This button will open the **simulation** pull-out. In a simulation you pick a starting state and see how it changes over time. You can change the starting state by clicking on the maximise button next to the variable table (), and you can examine graphs of the simulation.

If you run the simulation now it will start from a state where all variables are equal to zero. This shows the same oscillation found in stability testing. To look at the graph in more detail, click the maximise button ().

Finally, this button opens the **LTL Analysis** pull-out. This is a powerful way to search for simulations with specific behaviours. For example, if you want to find a simulation where one protein started off inactive, became active briefly, then shut off again later. This is useful if you know what you want to find but want to consider all possible starting states. As you need to describe the simulation you need to define two parts of your search; *states*, which consist of assignments for multiple variables, and *queries*, that link them together in time.

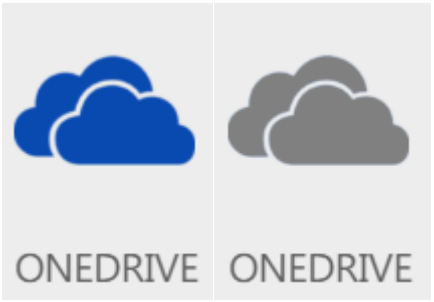
Here, create a query as below. Open the tab, click on add temporal properties and create from the eventually operand and the self-loop default state.

This will tell you if the model can lead to an endpoint that is not an oscillation; it turns red to indicate that no simulations do. This means that the only behaviour the model has is oscillation, when the input is set to one.

Saving your work

Click the name of the model to change it. Click the arrow to open the **model menu.**

Here you can work with multiple models. **Models** brings up a list of your saved models. By default, all files are saved in your web browser, but if you log into OneDrive they will all be saved securely on the cloud. If the OneDrive icon has a line through it, you are logged out. Click to log in.

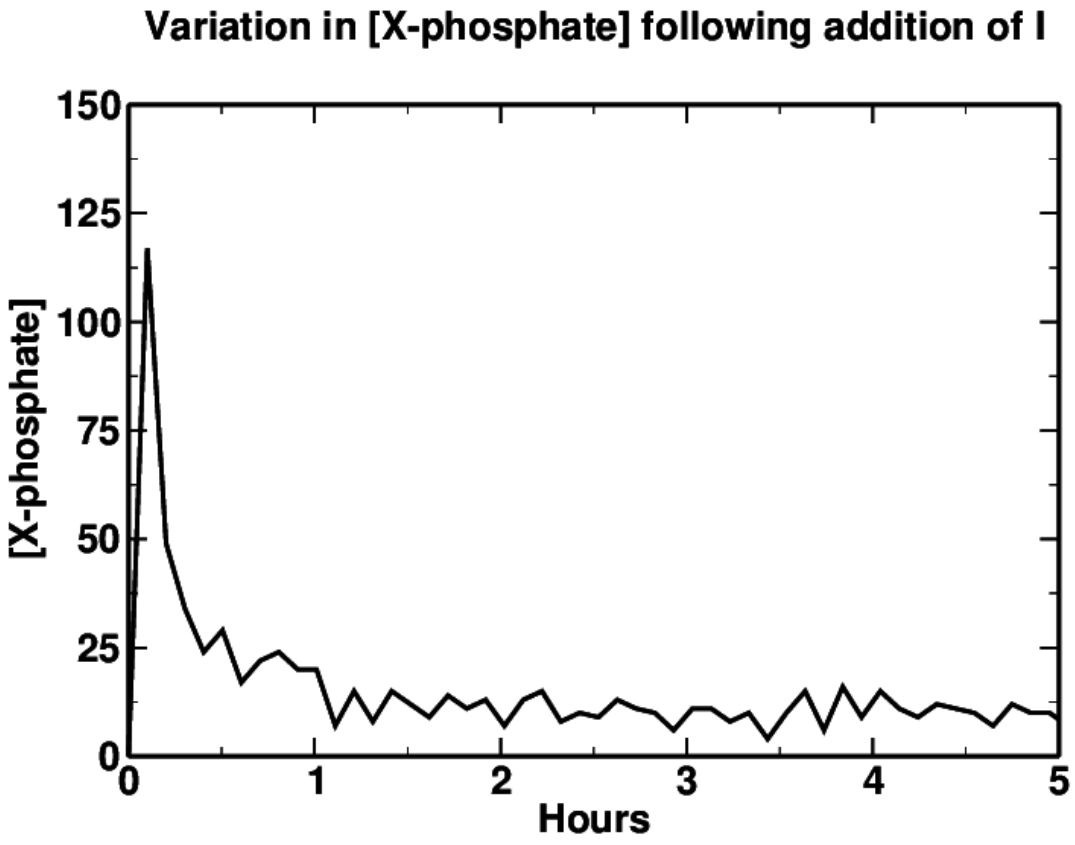


Once you have logged in the icon turns blue, and the list of models now reflects the models on OneDrive. If you click it again, you can see the models stored in the browser. Right click the icon to log out again.

Network selection

Kinases are proteins that modify other proteins. Specifically, they add a phosphate group to them. This addition can change a proteins behaviour, such as by activating an enzyme to catalyse a reaction. A phosphorylation cascade is a series of kinases that activate one another by adding a phosphate group. For example, in the chain below, when A is activated, it phosphorylates B, which in turn becomes active and phosphorylates C. Dephosphorylases are enzymes that remove phosphate groups from proteins, reversing the action of kinases.

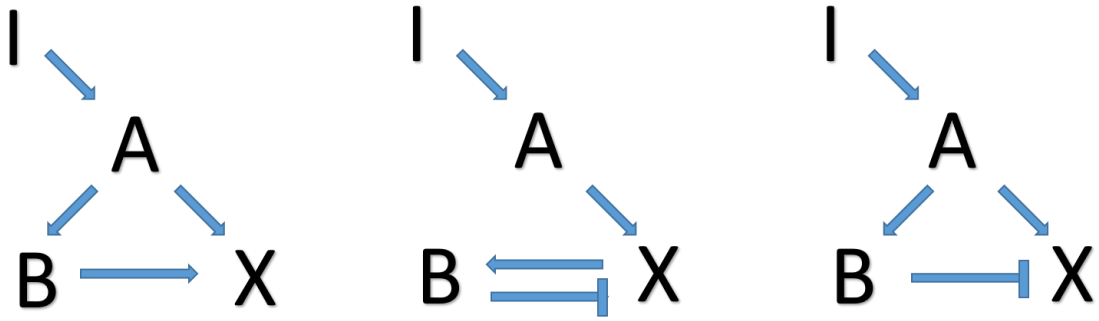
Phosphorylation cascades



We have been studying a system where there are 3 proteins in a phosphate cascade, A, B, and X. If there is no input signal to the system, the level of X phosphorylation is constantly low. If a large amount of input signal (I), the graph above is the result- X-phosphate levels go up initially, but rapidly go down again and stay low. We can split these observations down into three tests that the model must pass before we consider it to be correct:

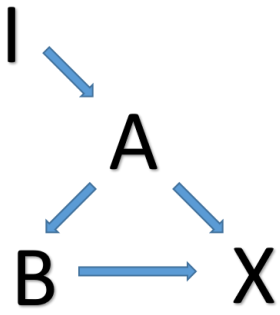
1) When I is low, X eventually becomes low and stays low
2) When I is high, X eventually becomes low and stays low
3) When I rises, X becomes high before becoming low

From reading the literature, different people believe that the following three networks could explain the observed behaviour.



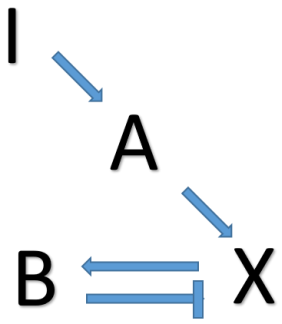
We want to build each of these models separately to test which one matches the experiment. Here, set all the variables in the BMA to have a range from 0-1. This is also the default.

Build the first network in BMA. By default, I will be set to zero as it has no incoming arrows to activate it. The first experimental observation was that X is low when I is low. We can test this in BMA with stability testing.

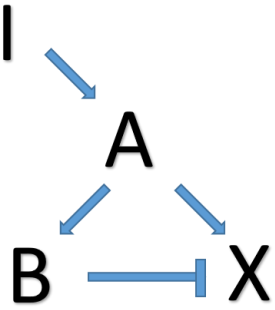


When we test the first network above, we find that the model is stable (all simulations lead to the same final state) and in the stable state, X is zero). So, we can say that the model matches the first of the three experimental observations, and go on to check the others.

If we set the target function of I to 1 (right click I and write “1” into the target function box, without quotation marks), we test the second experimental observation. We find that the model is stable, but the level of X is one- so it fails the second test. What features of the network lead to it failing? What do you think is missing?



We can repeat this process with the second network. If I is zero, we can see that again the model is stable and it passes the first test. However, when we get to the second test (I is one) we cannot prove stability. If you click on further testing, we can run a slower analysis that can prove stability or will return an example of why the model is not stable. This could be a bifurcation (where there are two or more stable states), or an oscillation (where a series of states repeat). Is the second model stable when I is one? Why do you think that the model behaves like this?



Finally, let's build the last model. Does this pass the first two tests? Why do you think that you see these results?

To test the final observation, the transient rise in X when I goes up, we need to run a simulation. We want to see how the model changes from one environment to another, so make a note of the stable state when I is zero. Then, set I to one and open the simulation tab. Click on the maximise button next to the variable table ( ) and edit the initial states to match the stable state of the network when I is zero. Next click run and examine the table and graphs. Does this model match the final experimental observation?

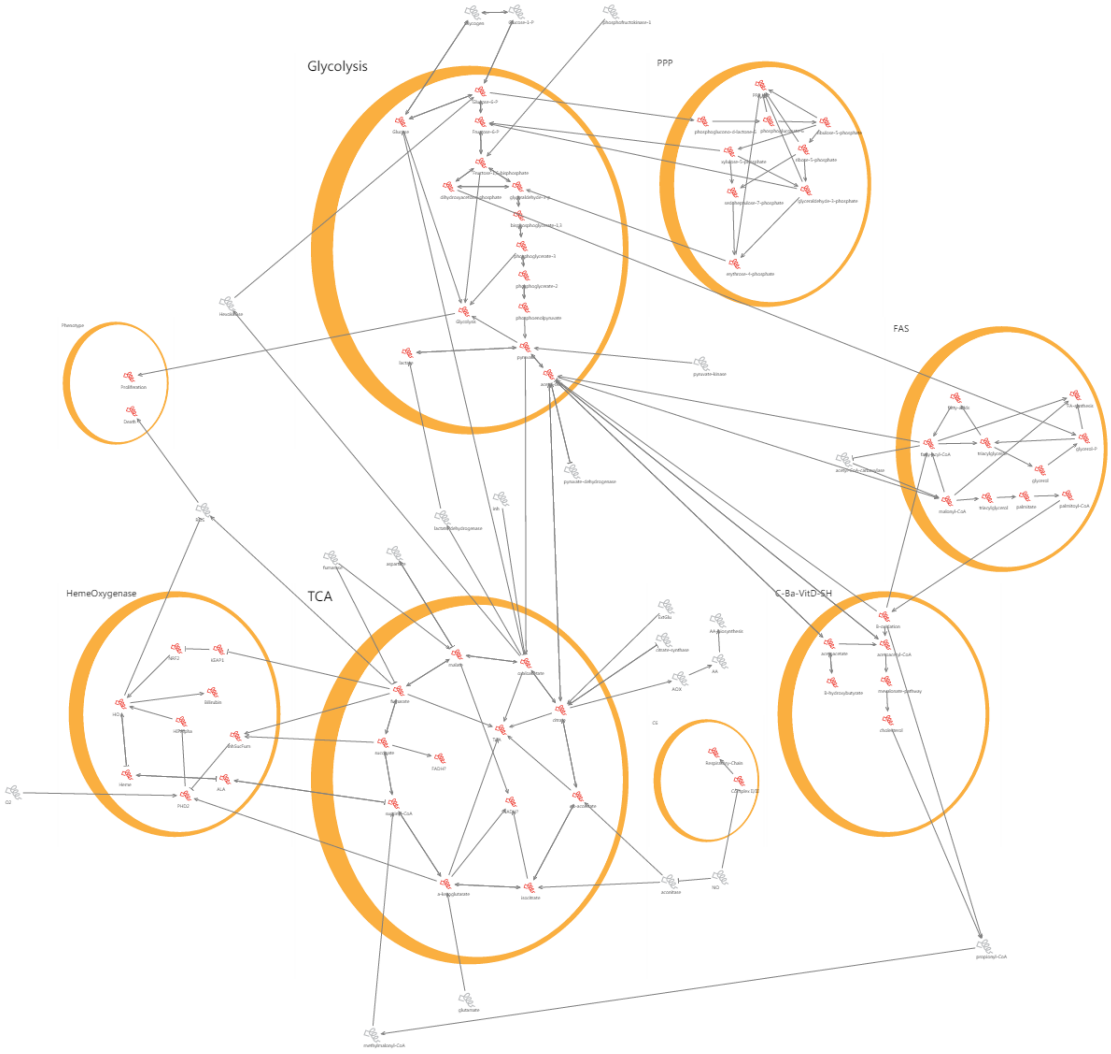
Cancer metabolism

To open this model in the tool, left click to select the json icon, copy and paste the model into a folder on your computer. This model can then be loaded in the tool by opening the model tab (on the left), clicking import, and selecting the file.

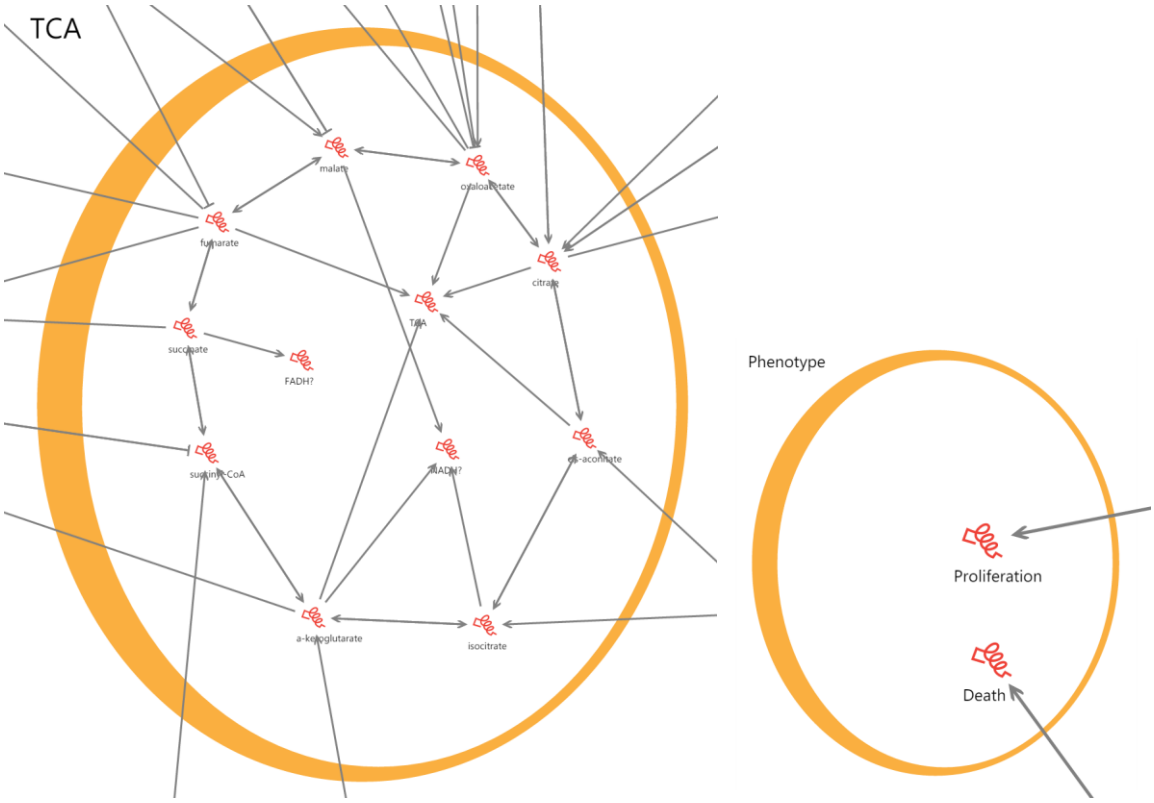
Metabolism demo.json

Cancer changes many key components of the cell behaviour. This includes changes to how the cell grows and communicates with its environment, but also alters how the cell works on a molecular level. Included in these changes is alterations to the cell's **metabolism**- that is, how the cell digests sugars and other nutrients and converts them to energy. Today we are going to look at how you can use computers to simulate the effects of mutations to metabolism on the cell behaviour. Computational biologists use these techniques to look for proteins that might be good drug targets in the future, and to suggest where we are missing information (and where to do the next experiments!).

Metabolism Model

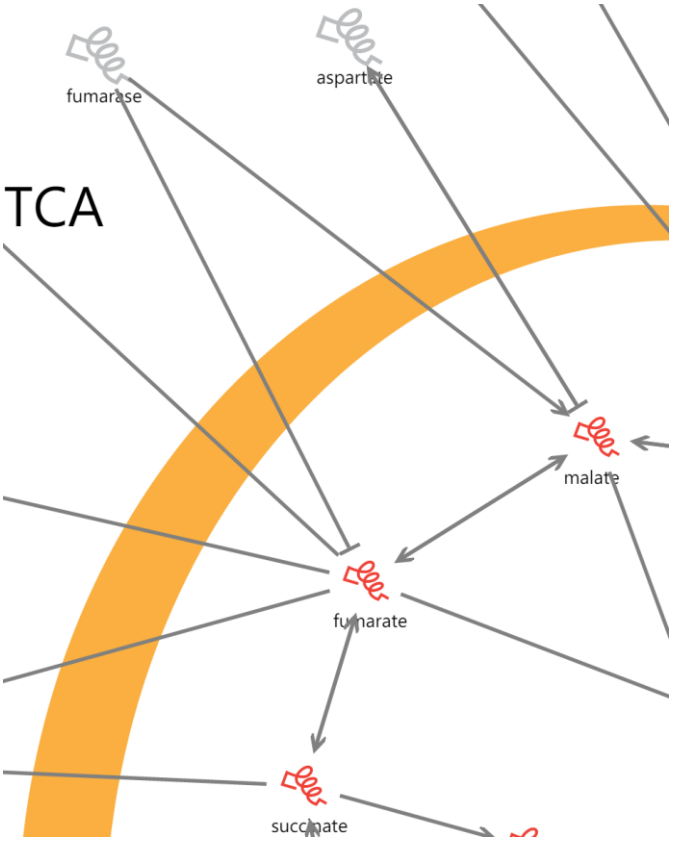


In metabolism sugar (glucose) is converted by a number of different processes into energy. In this model we have separated them with different cells. For example, **glycolysis**, the first stage of sugar metabolism, is represented by a large cell at the top of the model. Inside the glycolysis "cell" are the metabolites and proteins that carry out that conversion.

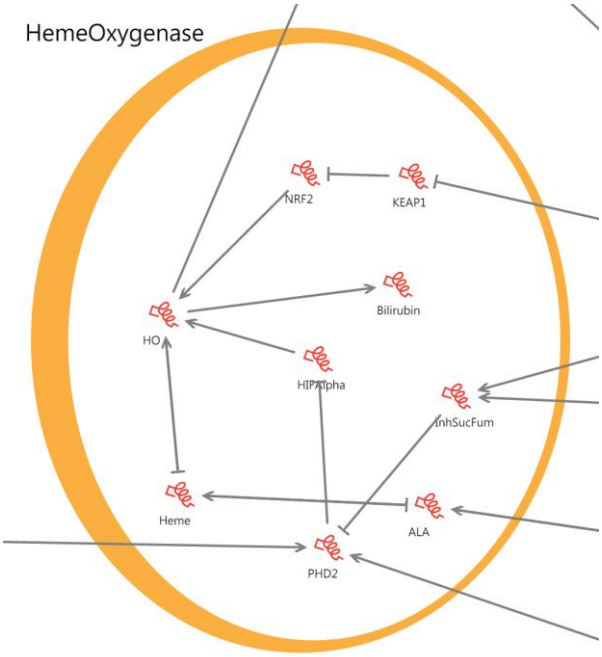


The two processes we will look at are **TCA** and **Phenotype**. The first is an important stage that follows glycolysis, and in which there are proteins that are mutated in cancer. The second describes how the whole cell behaves in response to any mutations; for example, **proliferation** indicates how much we expect the cell to grow and **death** indicates that the cell is likely to die. Most metabolites are represented by numbers from 0 to 4. These represent distinct ranges of concentrations that can be understood as No activity, low, medium, medium-high, and high activities.

If we perform a stability analysis we can see what the normal state of the cell is like. It shows that the model is stable, and that most proteins and metabolites have a value in the middle of their ranges. Looking specifically at the metabolites **succinate**, **fumarate**, and **malate** (in **TCA**) we can see that it normally has a value of 2 (medium activity), and the protein **HO** (in **HemeOxygenase**) has a level of 1.



The protein **fumarase** converts **fumarate** to **malate** and is lost in some cancers. We can simulate this by right clicking fumarase, selecting edit, and setting its target function from 1 to 0 (i.e. no activity). When we perform the stability analysis again, we can see how the cell has changed; there is no longer any malate in the model (shown by its value of 0), whilst fumarate has accumulated to high levels (its value is 4). **Succinate** has also accumulated, to a lesser extent. This causes other changes in the system- can you identify what has changed and has not? Why do you think they have changed?



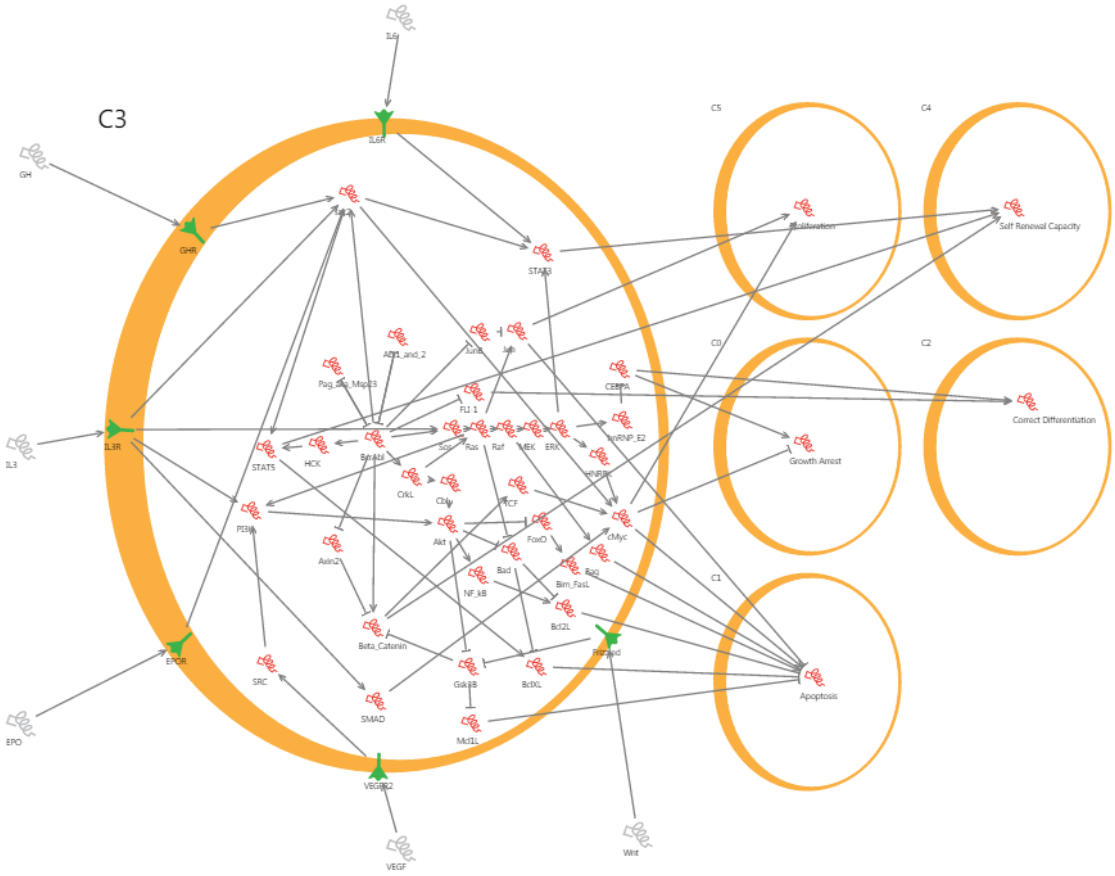
The Frezza grouped discovered that a second mutation in **HO** causes cells to die. What happens when you mutate HO (change its value to 0)? What happens if you have a HO mutation but **fumarase** is active (set its value back to 1)?

Understanding leukaemia

To open this model in the tool, left click to select the json icon, copy and paste the model into a folder on your computer. This model can then be loaded in the tool by opening the model tab (on the left), clicking import, and selecting the file.

Leukaemia.json

Here we are going to study a model of a single type of leukaemia, chronic myelogenous leukaemia. The progression of this disease from a manageable state (chronic phase) to a dangerous state (blast crisis) is characterised by a mutation that creates a single protein **Bcr-Abl**. In the clinic the disease is treated with a drug called **imatinib** that turns off Bcr-Abl and makes the disease manageable. In this tutorial we will look at the changes the disease makes to the cell, and how the drug reverts some of these changes.

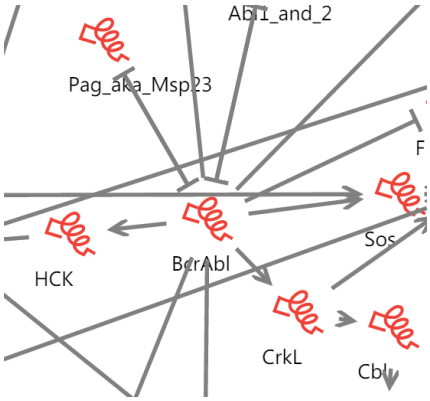


The model initially represents the cell in its **chronic phase** before Bcr-Abl is active. The large cell contains the proteins and genes that dictate cell behaviour, whilst the smaller cells on the right describe the **phenotype** or behaviour of the cell. Specifically, these represent **hallmarks of cancer** - types of behaviour changed in cancer.

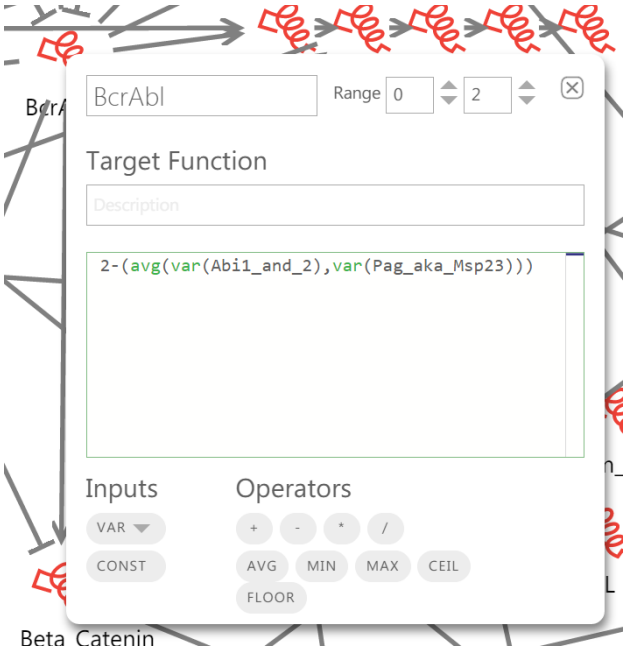
- **Proliferation** and **self-renewal capacity** describe the cells ability to grow and heal and are increased in cancer

- **Growth arrest** and **apoptosis** describe the ability of the cell to stop growth or, in extreme situations, commit suicide in response to mutation, and are reduced in cancer
- **Correct differentiation** describes the ability of the cell to perform its normal behaviours correctly, and is reduced in cancer

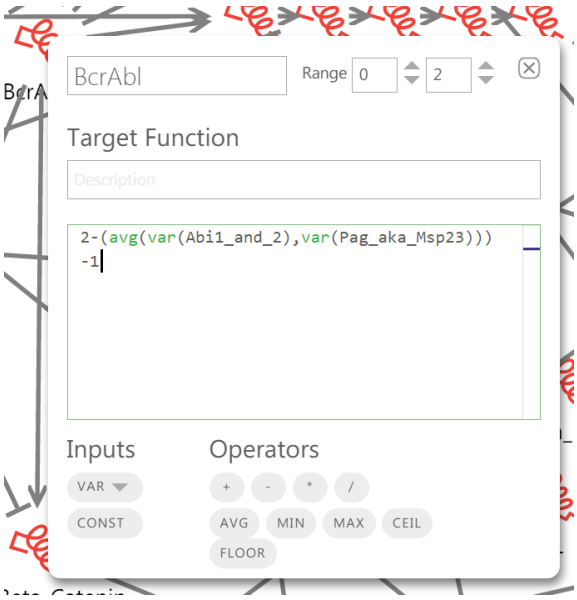
Stability analysis shows how the cell behaves in this state. Before the Bcr-Abl mutation occurs, the cells have medium levels of most of these activities. This is controlled by the large network of proteins interacting with one another.



We can study the cell at its most dangerous phase (**blast crisis**) by adding the Bcr-Abl mutation. Right click on Bcr-Abl (near the centre of the cell) and change the upper limit of its range from zero to two.



Reapply the stability test, and you can see how the cell has changed. How are the phenotypes different?



Finally, we will add the effect of the drug to the system.

To do this, right click the Bcr-Abl gene and edit the target function. This describes how the gene activity behaves in relation to the genes it depends on. To mimic the effect of the drug we can take 1 from the function and retest.

How has the model changed? There are times when drug resistance arises; can you find other proteins or pairs of proteins to drug using the model?