Visual SPiM Tutorial

Background

Visual SPiM is an advanced programming language for designing and simulating computer models of biological processes. The tool is intended for use by researchers and students with an understanding of basic concepts in chemical kinetics and stochastic simulation.

Preliminaries: Installing Visual SPiM

1. Navigate to http://research.microsoft.com/spim and click on the “Prototype Web Simulator” link.
2. Visual SPiM requires Silverlight 4 to be installed on your machine. If this is not the case, you should receive a notification in your browser together with installation instructions. Silverlight 4 is available for Windows and Mac as a plugin for most major web browsers.
3. In the top-right corner, click on the “Install” button. This will install the software to your local machine for offline use. You will be presented with options to create shortcuts on the Start menu and the Desktop. If you have already installed the tool, the “Install” button will read “Update” and will check if a newer version of the tool is available to download.

I: Chemical systems

1. Load the “Decay1” example from the “Examples:” drop-down menu. The program code will appear in the “Code” tab on the left. The code describes the decay of a molecular species “X()” at a constant rate “c”. Use the “Simulate” button to run a stochastic simulation of the model. This generates a time series plots of the simulation in the “Plot” tab on the right. Observe how the population of X species decays over time in the “Plot” tab. Inspect the corresponding reactions that were enabled during the course of the simulation by selecting the “Reactions” tab on the right. In this example only a single decay reaction is produced. Modify the initial population of the X() species in the “Code” tab on the left by replacing the existing number 1000 with 5000, navigate back to the “Plot” tab, re-run the simulation and observe the simulation results. Modify the rate of decay “c” by replacing the existing rate 0.5 with rate 1.0 and re-run the simulation.
2. Load the “Lotka1” example from the “Examples:” drop-down menu. This example models a predator-prey ecosystem, in which a prey species Y1 feeds on an inexhaustible food source X to reproduce, a predator species Y2 feeds on Y1 to reproduce and the predator species Y2 can die of natural causes. The inexhaustible food source X is modelled as a process X(), which can be eaten by performing an input on channel c1, written ?c1, and then remain as X(). The prey Y1 is modelled as a process Y1(), which can eat by performing an output on c1, written !c1, and then reproduce as two Y1() processes in parallel, or be killed by performing an input on c2 and then disappear. The predator Y2 is modeled as a process Y2(), which can eat by performing an output on c2 and then reproduce as two Y2() processes, or die of natural causes by performing a stochastic delay at rate c3 and then disappear.

3. Click the “Text-to-graph” button to generate a graphical representation of the model. Select the “Graph” tab on the left to display the graphical representation. The “Directives” sub-tab displays the duration of the simulation (Duration), together with the number of sample points to be plotted (Sample). It also displays the species to be plotted (Plots), the rate declarations (Substitutions) the channel declarations (Channels) and the initial conditions (Runs). Select the “Graph” sub-tab inside the “Graph” tab to display a graphical representation of the processes. Enlarge the tab by dragging the separator to the right of the tab. Click the “Fit” button to resize the graph to fit the tab. Click “Simulate” to run the model. This will simulate the program code in the “Code” tab and produce predator-prey oscillations. View the corresponding reactions in the “Reactions” tab on the right. The “List” sub-tab shows these reactions in textual form. Compare the list of reactions with the graphical SPiM model displayed in the “Graph” sub-tab inside the “Graph” tab on the left. Which reactions correspond to which channel interactions in the graphical SPiM model? Hint: inputs (prefixed with “?”) and outputs (prefixed with “!”) on the same channel interact with each other to produce a reaction.

II: Gene regulatory networks

1. Load the “Repressilator” example program and switch to the “Code” tab on the left. This example describes a module Gene(a,b), which is activated by receiving on channel a and produces a Protein(b) that sends on channel b. The module is instantiated three times, which gives rise to a network of three genes that mutually repress each other. Simulate the model and observe the simulation results in the “Plot” tab on the right. The network of genes gives rise to oscillatory behaviour, where each of the three proteins dominates in turn. Display the graphical representation of the model by clicking “Text-to-graph” and switching to the “Graph” tab on the left, and compare this with the list of reactions in the “Reactions” tab on the right. How many chemical reactions were enabled during the simulation?
2. Modify the code in the “Code” tab on the left to program a bistable switch instead of an oscillator, by modifying the last line so that it reads “run (Gene(a,b) | Gene(b,a))” instead. Simulate the model multiple times and observe the simulation results in the “Plot” tab. Explain why sometimes only Protein(a) is expressed and other times only Protein(b) is expressed.

III: Modular programming

1. Load the external file “combinatorial1.spi” by clicking on the Open icon in the “Code” tab and navigating to the SPiM tutorial directory in “\\cscience\internal\ToolsEvaluation\SPiM”. Alternatively, you may wish to copy the file to a local folder on your computer first. This example describes a collection of proteins X1p, X2p, X3p which can each activate the proteins Y1,Y2, Y3 by interacting on channel “a”. The reverse interaction is also possible, via channel “d”. Simulate the model and compare the Reactions list with the graphical representation of the model in the “Graph” sub-tab of the “Graph” tab. Compare the number of reactions with the number of inputs (prefixed by “?”) and outputs (prefixed by “!”) in the model, and explain why the number of reactions is much larger.

2. In the “Graph” sub-tab on the left, right-click on an area of white space and select “Insert Choice”. An oval node is inserted in the graph, and a properties dialogue appears on the left. Type in the name X4p in the “Name:” field and click OK. Insert a “Choice” node called X4 in a similar fashion. Select “Edge” radio button at the top of the Graph sub-tab to enable edge insertion. Now click on the X4p node and drag the mouse to the X4 node and release. This should insert an edge from node X4p to X4 and a properties dialogue box should appear on the left. Select “Output” on the “Action:” drop-down menu and type “a” in the channel field, then click OK. This should insert an output on channel a. Now insert an edge from X4 to X4p following the steps above, but this time choose “Input” in the “Action:” drop-down menu and type “d” in the channel field, then click OK. Click on the “Layout” button to regenerate the layout. Now select the “Directives” sub-tab on the left and insert a new row in the “Runs:” table by clicking on the “Add” button below this table. Double click in the first column in the newly inserted row and type 100. Double click in the second column and type X4p. Click Graph-to-text to validate the changes. Be careful not to click Text-to-graph, otherwise this will revert all the changes! Now hit the “Simulate” button. How many new reactions have been generated in the Reactions list, and why? How is the simulation affected?
3. Now load the example combinatorial2.spi from the folder “\\cscience\internal\ToolsEvaluation\SPiM” using the “Code” tab on the left and simulate the model. How does the code differ from that combinatorial1.spi? How do the simulation results differ? You may wish to re-load both files to compare the code and simulation results.
4. Modify the code of combinatorial2.spi in the “Code” tab on the left by adding the following line. “run 100 of Xp(4)” and re-run the simulation. What effect does adding this line have?

Answers:

Part I, number 3: The first reaction corresponds to an interaction on channel c1 between Food X and Prey Y1. The third reaction to an interaction on channel c2 between Prey Y1 and Predator Y2. The second reaction is a delay at rate c3.

Part II, number 1: 12 reactions should be enabled.

Part II, number 2: The bistable switch consists of two genes that mutually repress each other. Since the system is stochastic, both genes have an equal chance of producing proteins. Depending on which gene produces proteins first, it will block the other gene, which remains blocked indefinitely.

Part III, number 1: There are 18 reactions, 9 of which correspond to each of the three Xp proteins activating the three Y proteins, and 9 reactions for the each of the three X proteins deactivating the three Yp proteins. For the SPiM model there are 6 inputs and 6 outputs actions. If N is the number of Xp and Y species, the SPiM model contains N inputs and N outputs, but generates N squared reactions.

Part III, number 2: The new active species X4p can activate the three species Y1, Y2 and Y3, introducing 3 new reactions. Furthermore, the new inactive species X4 can be activated by species Y1p, Y2p and Y3p, introducing a further 3 reactions, resulting in 6 new reactions in total.

Part III, number 3: This code is equivalent to the previous example combinatoria1.spi, except that the code is much more compact. The simulation results are also equivalent.

Part III, number 4: Adding this line has the same effect as the modifications to the previous example combinatorial1.spi.