# Ingeneue switch tutorial: Building a simple network

This second tutorial will guide you through building a model, using Ingeneue, of the simplest genetic network possible: a two-state switch consisting of an mRNA and the protein it encodes. The mRNA is translated to the protein, while the protein is a transcription factor that activates transcription of its own mRNA (the gene auto-activates itself). Both the mRNA and the protein also degrade over time. Depending on the rates of degradation versus production, and the initial concentrations of each element, this switch may end up stably holding either an "on" state (both mRNA and protein at high concentrations) or an "off" state (nothing expressed). Numerous developmental regulators promote their own production, either directly (e.g. *achaete*, *scute*, *fushi tarazu*) or indirectly (e.g. *wingless*), and thus potentially exhibit intrinsic switch-like behavior. However, as you will see below (or could prove with a paper and pencil, in this simple case), whether or not even such a simple device exhibits this interesting dynamical property depends critically on the parameters governing the interactions in the network.

Currently all input to Ingeneue is done through text files. An Ingeneue network file has the following format:

```
&Model

      &width       3
      &height      3
      &numsides    6

      &Network     network_name

      &Genes
      ...
      &endGenes

      &Interactions
      ...
      &endInteractions

      &ParameterValues
      ...
      &endParameterValues

      &InitLevels
      ...
      &endInitLevels

&endNetwork
```

The tutorial takes you through each of the above pieces of the network file. The manual describes each piece in much more detail, and you may want to refer to it as you go through this tutorial. You may also want to refer to the program's html documentation as it contains descriptions of each of the Affectors and how to use them.

## Beginning a new network

1) Find the file "Template.net", make a copy, and call the new file something like "Switch.net"

2) Open "Switch.net" in a text editor.

*Note: You can use any text editor you want, as long as you save the resulting text as pure "Text". Unix text editors will normally save straight text. On Windows and Macintosh you may have to export as "Text". Although your regular word-processor will work, we highly recommend using a programmer's editor (on the Macintosh, try BBEdit Light, available for free over the web). A programmer's editor will format what you type in a more readable way – for instance, it won't wrap lines, it will automatically indent each line to the same level as the previous line, and so on.*

Upon opening "Switch.net" you will find it contains exactly the text shown in the introduction above. The first few lines are straightforward. The first line should read "&Model". This is how Ingeneue knows that this file defines a network model. Other file types that Ingeneue understands have a different initial tag.

3) The next two lines specify the width and height of the field of cells to be used in the model. The simple switch that you'll make is completely internal to a single cell, so there is no point in having more than one cell in the model. Change the grid of cells in the model to be 1 x 1 by modifying the &Width and &Height lines to read:

```
&width      1
&height     1
```

"&numsides" is the number of sides that each cell has. Currently, Ingeneue can deal with 2 sided cells (for a linear chain of cells), 4 sided cells (square cells), and 6 sided cells (hexagonal cells). Ingeneue is designed for modeling things that happen in (as yet 2–D) fields of embryonic cells, and most cells in embryonic epithelia are roughly hexagonal. The 2– and 4–sided options are relics of early work when we needed to test basic stuff about Ingeneue's guts and needed to simplify. We have only extensively tested hexagonal cells, and all our work has been done with hexagons, so to be honest we can't guarantee that the other numbers still work, so leave it at 6 sides.

4) "&Network" gives a name to the network defined in this file. Ingeneue uses this name to identify output files, although you can override it. Give your network an appropriate name by changing the token that comes after the &Network tag. Your name should not contain spaces or other odd characters.

## Defining the Nodes

The first substantial section in the network file defines the Nodes for the network. Recall that a Node is an element in the network: a mRNA, protein, protein complex, and so on; in other words, some molecular species whose dynamics figure in the model. You need to define one Node for any species whose concentration will change over time as the network runs (and sometimes even some that don't, but we won't worry about that here). In defining a Node, you give it a name, and also give it a number of characteristics.

Table 1 lists the most important characteristics of a Node. There are more, but the defaults for those will work fine for this tutorial.

| Tag | Description |
|---|---|
| &Location | Whether the molecular species represented by this Node is located on the cell's membrane or in the cytoplasm of the cell. Legal values are "membrane" or "cytoplasmic". Nuclear-localized molecules are considered cytoplasmic. There may be more compartments in the future. |
| &Type | The type of molecule this Node represents. Current options are "rna", "protein", "complex" (i.e. of two or more proteins) or "input" (an as-yet-poorly-defined input to the network). These aren't so important now but may become so. |
| &Color | The color to draw the Node in Cell View. Many common colors (e.g. "blue", "green", "pink") will work. |
| &Show | Whether to show the Node in the Cell View window ("on") or to hide it ("off"). |

5) To make the mRNA node for the gene "onoff", make a new line between the &Genes and &endGenes tags and type in:

```
<<&Genes>>
      &onoff
      // characteristics will go here in next step
      &endonoff
<<&endGenes>>
```

In the above text, the parts marked off with << ... >> are already in the file, so don't type them – they are included in the snippet just to orient you. You can also skip typing comments (comments are anything that comes after a "//").

6) Give the mRNA a location, type, color, and show it in the Cell View by adding the following lines inside of the &onoff / &endonoff pair.

```
<<&onoff>>
      &Location         cytoplasmic
      &Type             rna
      &Color            pink
      &Show             on
<<&endonoff>>
```

7) Define the protein in a similar way by adding these lines following the &endonoff line

```
<<&endonoff>>

&ONOFF
      &Location         cytoplasmic
      &Type             protein
      &Color            pink
      &Show             on
&endONOFF

<<&endGenes>>
```

Ingeneue is case-sensitive, so "onoff" is something different from "ONOFF". Our habit is to use all-lowercase names for mRNA's and all-uppercase names for proteins, which helps avoid confusion. Each block of the text file ends with an "&end" plus the name of the thing that is ending, in this case the name of the gene.

*Note: for programmers who might be curious about such things, the structure of Ingeneue's input files corresponds roughly to how software objects load themselves. Ingeneue uses a generic token-recognizer module, and many Ingeneue classes know how to interpret a suite of tags ("&xxx") and the tokens that follow on the same line. When the stream contains a declaration for a new object of a certain type, whichever object is currently managing the input creates an object of the right type, and hands it the input stream to get its setup information from. Each object knows it is done with its setup information when it comes to an "&end..." tag with its own name, and hands the input stream back to whichever object it got it from.*

You have now defined the two Nodes in this network. Next you must define the interactions between them.

## Defining the Affectors

This switch is as simple as it gets. The mRNA gets translated into protein, and the rate of translation is proportional to the amount of mRNA. The protein, in turn, activates production of the mRNA according to a sigmoid (S-shaped) dose-response function. The S-shaped function has three parameters: a maximum rate of transcription; a concentration of regulator at which transcription proceeds at half its maximal rate; and a non-linearity parameter (analogous to a Hill coefficient, if you are familiar with biochemical kinetics) that gives the steepness of the curve. Both mRNA and protein also undergo non-specific degradation.[1]

Each of these interactions makes a piece of the differential equations that determine how quickly the mRNA and protein concentrations change over time. The mRNA has two parts to its equation – the sigmoidal activation term and degradation. The protein also has two parts to its equation – the translation term and degradation. We call these equation pieces "Affectors", and so far we have a collection of some 50-odd Affectors that represent a broad spectrum of possible interactions between elements of genetic networks. Most of the Affectors have documentation online that explains what they do and what their parameters are.

8) Start by defining the Affectors for the mRNA. Make a new line between the "&Interactions" and "&endInteractions" tags and type in:

```
<<&Interactions>>
      &onoff
      // Affectors for onoff will go here
      &endonoff
<<&endInteractions>>
```

9) Add the first-order decay Affector between this second set of &onoff - &endonoff tags by typing:

[1] Because of some scaling we have done (a standard mathematical sleight-of-hand called "non-dimensionalization" of the equations), both the maximal rate of translation and the maximal rate of transcription disappear, but the equations now need to know the half-lives instead. The Supplementary Information for von Dassow *et al*. (00), available at our website http://www.ingeneue.org/, explains the mathematical foundations of Ingeneue models.

```
<<&onoff>>
      &DecayAff  onoff  H_onoff
<<&endonoff>>
```

The names of all Affectors end in "Aff". The first-order decay Affector ("DecayAff") needs to know two things: which Node it affects, and the name of the parameter which gives the half-life for that Node (i.e. the inverse rate of decay). Although you are welcome to call your model's parameters anything you want, we recommend our practice of using certain letters for certain types of parameters, and following the identifying letter with the name(s) of the Node(s) that the parameter "goes with". Thus "H" is our letter for half-lives, and the half-life for "onoff" is given by "H_onoff".

10) If you have the html documentation files for Ingeneue, search for the "affectors" package, and inside of this search for the "DecayAff" class. Read the comments at the top of this class which give the function of that Affector and a description of its parameters.

*Note: we are still in the process of adding or improving these comments for many of the Affector classes. If you don't have the html documentation files, you can simply open the DecayAff.java source code file in your text editor and read the comment at the top.*

11) Now add the sigmoidal transcriptional activation Affector "Txn1Aff" to the list of Affectors for the onoff mRNA:

```
<<&onoff>>
      <<&DecayAff  onoff  H_onoff>>
      &Txn1Aff  ONOFF  K_ONOFFonoff  nu_ONOFFonoff  H_onoff
<<&endonoff>>
```

We use "K_..." for half-maximal activity coefficients, and "nu..." for non-linearity (or "cooperativity") coefficients. Thus K_ONOFFonoff is the concentration at which ONOFF half-maximally activates onoff.

12) Look up Txn1Aff in the documentation pages or the source code as in step 10.

13) Define the Affectors for the ONOFF protein analogously to the onoff Affectors by typing in the following:

```
<<&endonoff>>
      &ONOFF
            &DecayAff  ONOFF  H_ONOFF
            &TlnAff  onoff  H_ONOFF
      &endONOFF
<<&endInteractions>>
```

The Affectors in the network you've defined have four parameters. There are the two half-lives, H_onoff and H_ONOFF. There are also the two parameters associated with the sigmoid activation curve, K_ONOFFonoff and nu_ONOFFonoff. In the next part of the file, you give these parameters initial values and define the range over which they can vary.

## Setting Parameter Values and Ranges

Parameter values and ranges are specified on one line each in the "ParameterValues" block, with the following format:

```
&parameter_name  set_value  min_value  max_value  how_to_pick
```

14) Set both half-lives to 20 min. by making a new line in between the &ParameterValues and &endParameterValues lines and typing in the following:

```
<<&ParameterValues>>
      &H_onoff     20     1      100    Linear
      &H_ONOFF     20     1      100    Linear
<<&endParameterValues>>
```

This not only sets the parameter's value to 20 minutes, it also says that the half-lives can range between 1 and 100 minutes. These ranges will not be used when you simply run the model, but they will be used if you have Ingeneue automatically change parameter values, for instance to search for parameter sets that confer a particular behavior on the model. The last argument tells Ingeneue whether to vary the parameters linearly or logarithmically when picking new values.

*Note: as a rough guideline, if a particular parameter ranges over several orders of magnitude, you should specify that it varies logarithmically, or else you will bias the selection of parameter values towards the high end. For instance, we have the half-maximal activation parameter, K_ONOFFonoff, vary between 0.001 and 1.0. 90% of this space is between 0.1 and 1.0 on a linear scale, so if you randomly sampled from a linear distribution, 90% of your samples would have values >0.1. By taking the logarithm before picking parameter values, you will get an equal number of samples in the intervals [0.001,0.01], [0.01,0.1], and [0.1,1.0]. If you have a small range, you might rather choose a linear range. We typically use a linear range for half-lives and cooperativity coefficients. You are welcome to do whatever you want in this regard, but if you have even the slightest desire to make any analysis of parameter space, especially comparing between two different models, you* ***must*** *be consistent about the ranges.*

15) Set the initial half-maximal activation parameter to 0.1 and the initial non-linearity to 2 by adding the following lines:

```
      <<&H_ONOFF         20     1      100    Linear>>
      &K_ONOFFonoff      0.1    0.001  1.0    Logarithmic
      &nu_ONOFFonoff     2      1      10     Linear

<<&endParameterValues>>
```

## Setting Initial Conditions

The last bit of the network file sets the initial concentrations of each Node in each Cell. We have made a number of different modules for setting initial concentrations, each with its own name. One module sets concentrations in columns of Cells, another sets a middle Cell differently than the cells surrounding it, and so on. For this network, use the simplest initial condition, "CellIC" which just sets a particular Cell to a particular value for a particular Node. Of course, one could make an entire pattern in even a very large field of cells using just the CellIC object, but that would be extremely cumbersome and therefore error-prone. Our goal, which we have not yet achieved, is to have a general-purpose toolkit of initial condition modules, just as we have for Affectors. Indeed, that is part of the overall design of Ingeneue: using standardized, stereotyped modules for writing equations, for specifying initial conditions, and even for pattern recognition and other functions.

16) Set the initial concentration of onoff mRNA to 0.5 in cell (0,0) using the following lines:

```
<<&InitLevels>>
      &CellIC
            &Node       onoff
            &Value      0.5
            &XPos       0
            &YPos       0
      &endIC
<<&endInitLevels>>
```

Note that unlike the other "&end..." statements, each initial condition finishes with the same "&endIC" tag. The ending tag doesn't change based on the initial condition.

17) Find the CellIC documentation pages in the "initialconditions" package, or else open the CellIC.java file and read the comments at the top.

18) Set the initial concentration of the ONOFF protein in an analogous way:

```
      <<&endIC>>

      &CellIC
            &Node       ONOFF
            &Value      0.5
            &XPos       0
            &YPos       0
      &endCellIC
<<&endInitLevels>>
```

That's the whole network file. You should now be able to run this file in Ingeneue.

## Running the Network

19) Run Ingeneue and select the 'Load' command from the 'File' menu to load your new network file.

If you get error messages, often they say something meaningful from which you can figure out where the problem is. If not, you can cheat by looking in the SwitchComplete.net file, which should be identical to the file you just typed out.

20) Run the switch network by selecting 'Run Model' from the 'Run' menu.

When running this model, the action may go by too quick for you to see it unless you use the 'Step Model' command rather than the 'Run Model' command (you will want to reset the model first by choosing 'Reset model' from the 'Run' menu).

21) See if you can get the network to switch stably "off" rather than "on" by changing the initial concentrations of protein and mRNA. You can do this either from within the program (see the first tutorial) or by changing the network file and reloading it.

That completes this tutorial. From here, you might want to try adding more complexities to this simple model. You will want to look through the different Affectors we have included with the program to see the types of interactions that you can easily (without writing code) add to your network. Some of these Affectors are rather complicated and it may help you to understand the math before using them. The Supplement to von Dassow *et al*. (00) discusses the mathematical framework of Ingeneue in the context of our segment polarity network

model.  The documentation on both Affectors and Iterators will continue to improve over time, but right now it is still rather sparse in places.  If you are a programmer you can also look at the code for the Affectors themselves to see what equation each one encodes.