

# NetBuilder'

As per 30/08/2006

## General

NetBuilder' is a modelling and simulation environment that is specifically aimed at the study of genetic regulatory networks. Therefore its mathematical modelling capabilities (definition of the kinetic laws) are limited in comparison with those in tools like Gepasi/COPASI, Jarnac, etc. The differences will be discussed in more detail below. Its simulation capabilities are the same.

NetBuilder' users (humans as well as automata) will follow the following scenario:

1. Create a Petri-net model (NBLayer) of a genetic or other network, and set its initial state, using the routines available in NBNetBuilder
2. Convert the Petri-Net model to an instance of NBMathModel, using NBConverter. The NBMathModel instance contain all structural and mathematical information required for the simulation of the model
3. Create an NBSimulation session; supply the session with an NBSimulator instance (the simulation tool), the NBMathModel object created above, a start and end time and time step for the simulation
4. Perform the simulation; extract the trajectories of specified places from the NBSimulation session.

## Model creation

Petri-net models consist of *places* (states, species), *transitions* (reactions), and directed *arcs* (interactions) that connect places and transitions. Places and transitions are located within a Petri-net *layer*, which is a logical or organizational, rather than a physical structure. Layers themselves may be contained in other layers, but at present it is not possible for layers to partly overlap. Thus, apart from the outer layer which has none, all Petri-net nodes (places, transitions, and layers) have exactly one direct parent layer.

### ***Petri-net components and their attributes***

Petri-net components have the following attributes:

#### **Layer**

- *name* - must be unique among all named components (places, transitions, and layers) within its parent layer

## Place

- *name* – must be unique within layer
- *value* - real number that must be zero (default) or greater

## Transition

- *name* – must be unique within layer
- *parameter map* - the parameter map of a transition has two entries:
  - “*rateConstant*” - the rate constant for the unmodified reaction, must be zero (default) or greater
  - “*combinationID*” - identifies the type of operation that is used to combine the groups of modifiers; allowed operations are:
    - “+” – Addition of individual contributions
    - “\*” – Multiplication of contributions
    - “|” – Disjunction of contributions (logical or; set union)
    - “&” – Conjunction of contributions (logical and; set cross section)
- *modifier group* – consists of:
  - *arcs* – list of references to arcs taking part in the combination; may be empty. One arc may function in at most one group.
  - *operationID* - allowed operation identifiers are:
    - “+” - Addition
    - “x” – Multiplication
    - “&” – Conjunction (logical and; set cross section)
    - “c” - Competitive combination
    - “h”: Highly cooperative combination
    - “p”: Preformed complex combination

(It is expected that the operations “&”, “h”, and “p” will give similar qualitative behaviour, and only using one of them to combine modifiers is probably sufficient. However, “c” has quite different qualities. “+” and “\*” are suitable for combining arcs whose mode is linear)
- *contribution* – contribution of group to total flux

## Arc

- *type*
  - “*input*” indicates that the place associated with the arc provides a reactant to the reaction represented by the associated transition. Firing of the transition decreases the value of the reactant place by a stoichiometric amount (see below).
  - “*output*” indicates that the place collects a product of the reaction. Firing of the transition increases the value of the product place by a stoichiometric amount.
  - “*activator*” or “*inhibitor*” indicates that associated place is a rate modifier, and that the rate at which the transition fires increases (activation) or decreases (inhibition) when the value of the place increases. Firing of the transition does not change the value of the associated place.
- reference to the connected *place*
- reference to the connected *transition*
- *parameter map* - The set of parameters associated with an arc is dependent on the type of arc:
  - “*weight*” – (input and output arcs, default 1.0) stoichiometry; the value that is subtracted (input) or added (output) upon the firing of the transition
  - “*multiplier*” - (input, activator and inhibitor arcs; default 1.0), multiplier used in flux calculation
  - “*exponent*” (input, activator and inhibitor arcs; default 1.0), exponent used in flux calculation
  - “*mode*” - (input, activator and inhibitor arcs), options:
    - “*saturating*” – (default for activator and inhibitor arcs) uses  $x = \frac{(multiplier * value)^{exponent}}{1 + (multiplier * value)^{exponent}}$  in flux calculation
    - “*linear*” – (default for input arcs) uses  $x = (multiplier * value)^{exponent}$  in flux calculation

## **NBMathModel**

The Petri-net model constructed with NBNBuilder is converted into an NBMathModel instance by feeding NBConverter with the root layer of the Petri-net. To create the NBMathModel, the Petri-net is flattened, and the attribute values of all places, transitions, and arcs are used to create three separate attribute vectors: placeVector, transitionVector, and arcVector. Each place, transition, and arc is given a unique number, which corresponds to the index of their entry in the attribute vectors.

## Attributes

- *placeVector*
  - $placeVector[i][0]$  = name (string)
  - $placeVector[i][1]$  = value (double, default 0.0)
  - $placeVector[i][2]$  = function reference (object, default None), takes a function that takes  $placeVector[i][1]$  and another parameter (e.g. time) and returns the outcome of the specified function (e.g.  $function(val, time) = val * \sin(2 * \pi * time / period)$ ). Can be used to modulate input.
- *transitionVector*
  - $transitionVector[i][0]$  = name (string)
  - $transitionVector[i][1]$  = parameters (dictionary of key:value pairs)
    - "rateConstant": double (default = 0)
    - "combinationID": "|" (default), "&", "+", or "\*\*"
  - $transitionVector[i][2]$  = modifier groups
    - $transitionVector[i][2][0]$  = arcs (list of arc numbers, may be empty)
    - $transitionVector[i][2][1]$  = operationID ("+", "\*\*", "|", "&", "+", "\*\*", "p", "h", "c")
    - $transitionVector[i][2][2]$  = contribution (double)
- *arcVector*
  - $arcVector[i][0]$  = type ("input", "output", "activator", "inhibitor")
  - $arcVector[i][1]$  = place (integer, refers to place number)
  - $arcVector[i][2]$  = transition (integer, refers to transition number)
  - $arcVector[i][3]$  = parameters (dictionary of key:value pairs)
    - "weight": double (default = 1.0)
    - "multiplier": double (default 1.0)
    - "exponent": double (default 1.0)
    - "mode": string ("linear" or "saturating")

## Functions

- *flux* – returns the flux through a given transition, calculated on the basis of the current system state (values of all places) and time

- *derivatives* – returns the derivative values for each place (in the order of placeVector), constructed on the basis of the model parameters, the current system state and time

## Simulation

NetBuilder' has simulators to carry out stochastic or deterministic numerical integration on the basis of a given initial state and the expressions for the flux that are constructed (by hidden functions in NBMathModel) on the basis of the model parameters. There are two types of deterministic simulators: one (NBContinuousSimulator) uses LSODA (via scipy) for continuous integration, the other (NBDifferenceEquationSimulator) uses a simple forward Euler with fixed step size (and deals, therefore, with difference equations). The stochastic simulator (NBStochasticSimulator) implements a priority queue method inspired by Gibson & Bruck).

The times of the next independent events (stochastic) and the derivative values (deterministic) both calculate the flux through the appropriate transitions for each simulation step. The way the flux calculation is done is explained below.

### Flux calculation

In all simulations, stochastic or deterministic, the flux through each transition – the rate at which it is working - in the model is calculated before the next step in the simulation is taken. Fluxes indicate *rates* (how fast reactions are progressing), *not* quantities (how much material is removed from the reactant, or deposited in the product places). In relatively simple systems such as biochemical reaction networks, fluxes are only dependent on the current state of the system, i.e. the values of the places in the model. The current flux through a particular transition is determined by the values of its reactant and modifier places. For NetBuilder' (in its current incarnation) fluxes ( $J$ ) are calculated using the simple equation:

$$J = r \cdot mass \cdot mod \quad (1)$$

$$mass = \tilde{O}[X_j] \quad (2)$$

$$mod = f^{combi} [c_k \times f^{group}_k [[X_j]]] \quad (3)$$

$$f^{group}_k [[X_j]] = num^{group}[a[X_i]] / denom^{group}[X_i] \quad (4)$$

$$X = (F \cdot val)^P \quad (5)$$

Here  $r$  is the rate constant parameter of the transition,  $mass$  is the factor contributed by the combined reactant places, and  $mod$  the factor contributed by the modifiers (activators and inhibitors). The indices  $i$ ,  $j$ , and  $k$  enumerate input arcs, modifier arcs, and modifier groups.

According to equation 5, in which  $F$  and  $P$  are the values of the *multiplier* and *exponent* parameters associated with the arc that connects the place with the transition, and  $val$  is the current value of the place,  $X$  is the modified current value of the input or modifier places. Thus, the value of  $mass$  is simply the product of the modified values  $X_i$  for each input arc  $i$ ,

whereas  $mod$  is the combination  $f^{combi}$  (specified by the *combination* operator for the transition, can be cross section, union, product, or sum) of each contributions of modifier group  $k$ . Here  $c_k$  is the value of the *contribution* parameter,  $f^{group}_k$  is the operation specified by the *operatorID* associated with the modifier group.  $f^{group}_k[[X_j]]$  is the quotient of a numerator, whose dependence on  $X_i$  ( $a[X_i]$  depends on the modifier arc type (activator or inhibitor).

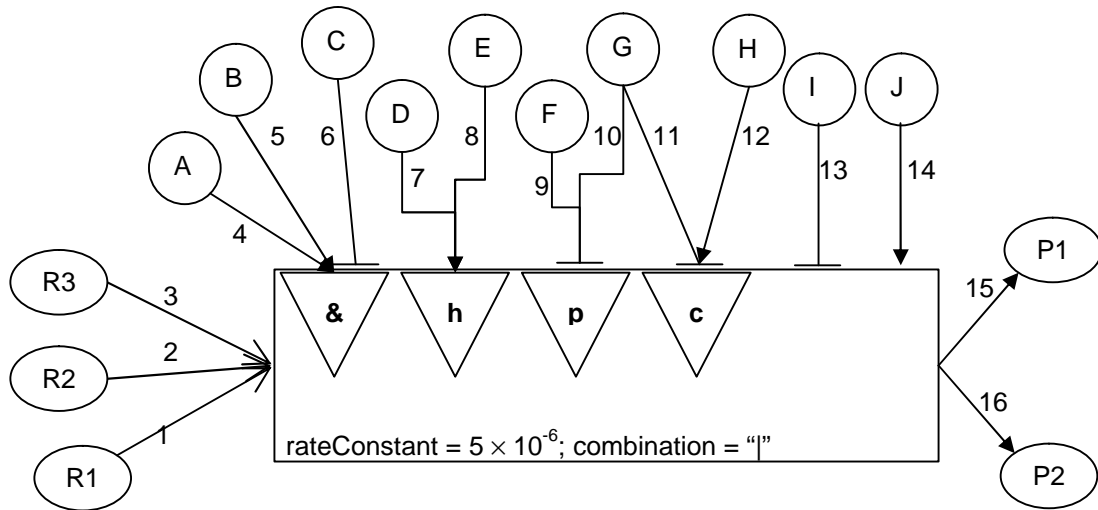
The following table indicates how  $f^{group}_k[X_i]$  is calculated for the various operators.

<b>opID</b>	<b>Description</b>	<b>a[X<sub>i</sub>] (activator)</b>	<b>a[X<sub>i</sub>] (inhibitor)</b>	<b>Num</b>	<b>Denom</b>
	Single (saturating)	$(F_{val})^P$	1	$a [ X_i ]$	$1.0 + X_i$
	Single (linear)	$(F_{val})^P$	$(F_{val})^{-P}$	$a [ X_i ]$	1.0
<b>+</b>	Sum (linear)	$(F_{val})^P$	$-(F_{val})^P$	$\Sigma[ a [ X_i ] ]$	1.0
<b>*</b>	Product (linear)	$(F_{val})^P$	$(F_{val})^{-P}$	$\Pi[ [ X_i ] ]$	1.0
<b> </b>	Disjunction (saturating)	$(F_{val})^P$	1	$\cup[ a [ X_i ] ]$	$\Pi[1.0 + X_i]$
<b>&amp;</b>	Conjunction (saturating)	$(F_{val})^P$	1	$\Pi[ a [ X_i ] ]$	$\Pi[1.0 + X_i]$
<b>h</b>	Highly cooperative	$(F_{val})^P$	1	$\Pi[ a [ X_i ] ]$	$1 + \Pi[ X_i ]$
<b>p</b>	Preformed complex	$(F_{val})^P$	1	$\min[ a [ X_i ] ]$	$1 + \min[ X_i ]$
<b>c</b>	Competitive	$(F_{val})^P$	0	$S[ a [ X_i ] ]$	$1 + S[ X_i ]$

The disjunctive operation ( $\cup$ ) works out as recursive  $a = c+b(1-c)$  recursive (where  $b$  or  $c$  can be substituted with  $e+d(1-e)$ , etc).

## Example

(somewhat over the top):



arc nr	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
type <sup>1</sup>	in	in	in	act	act	inh	act	act	inh	inh	inh	act	inh	act	out	out
weight	1.	2.	2.												1	3
multiplier	1.	1.	1.	10.	5.	0.1	0.2	0.2	1.	1.	10.	15.	0.4	10.		
exponent	2.5	2.	0.	2.	1.	1.5	3.	3.	2.	2.	1.	1.	3.	1.5		

<sup>1</sup>in = input, out = output, act = activator, inh = inhibitor

Here, 1, 2, and 3 are input arcs connecting reactants (R1 - R3), 14 and 15 output arcs connecting products (P1, P2), and the others are modifier arcs, with 7, 9, 10, and 11 inhibitor and the rest activator arcs, connecting rate modifying agents that decrease (inhibitors, C, F, G, note that G functions in two interactions) or increase (activators, A, B, D, E, H, J) the rate as their value increases.

Suppose the current state (values of the places) is as in the table below:

Place	R1	R2	R3	A	B	C	D	E	F	G	H	J	P1	P2
value	10.	20.	3.	1.	0.	10.	14.	0	4.	8.	20.	10.	0.	3.

Then the values for X and a[X] for each arc are:

arc	4	5	6	7	8	9	10	11	12	13	14
X	$(1/10)^2$	$(0/5)^1$	1000.	$3.4e5$	0.0	16.	100.	0.8	1.3	$1.6e4$	1.0
a[X]	0.01	0.0	1.0	$3.4e5$	0.0	1.0	1.0	0.0	1.3	1.0	1.0

Thus the numerator and denominator values for the fractional activity  $frA$  according to the above equations and values are:

Modifier groups, operators, and contribution values (specified), and values for the nominator, denominator, and  $f^{group}[X_i]$  (calculated) are given in the next table:

arcs	OpID	Contribution	num	denom	$f^{group}[X_i]$
(4, 5, 6)	&	0.1	0.0	1011.	0.0
(7, 8)	h	1.0	0.0	1.0	0.0
(9, 10)	p	0.5	1.0	17.	0.03
(11, 12)	c	0.9	1.3	2.1	0.56
(13)	& <sup>1</sup>	0.2	1.0	1.6e4	6e-5
(14)	& <sup>1</sup>	0.8	1.0	2.0	0.5
( )	& <sup>1</sup>	0.1			0.1

<sup>1</sup>Note that the operator for ungrouped arcs and empty groups is & - in fact the operator is immaterial.

The values of  $f_rA$  are then disjunctively combined, which yields a value of 0.808 for the total modification ( $mod$ ).

The value of  $mass$  is calculated as  $10^{2.5} \times 20^2 \times 3 \approx 3.8 \times 10^5$  (namely the place values to the power of the kinetic order parameter – which only for arc 2 is equal to the stoichiometry).

The total flux is the calculated as  $J = r \times mass \times mod$ , where the rate constant  $r$  is  $5 \times 10^{-6}$  (as indicated in figure), and is, therefore 1.53.