

MS-BioNET User's Manual

version: 0.99

Main author: Sara Montagna

Creation date: 20110705

Last Changes date: 20110705

DEIS, Università di Bologna, Italy

Chapter 1

Getting Started

1.1 MS-BioNET Distribution

MS-BioNET is a simulation framework for modelling and simulating large networks of compartments hosting a chemical solution and communicating through an enhanced model of chemical reaction addressing molecule transfer.

On top of the framework, a logic-oriented specification language is used to flexibly specify simulation scenarios: on the one side it tightly focusses on biochemistry, by providing constructs to directly express biochemical reactions, compartments, compartment link topology, and reactions involving selective transfer through membranes; on the other side it relies on logic-based goal resolution and unification, achieving the expressiveness needed to easily handle size and complexity of the biochemical network. Behind the hood, such a specification is turned into an intermediate language (a sort of bytecode) that feeds a simulation engine implemented by adapting the optimised version of Gillespie’s algorithm (described in [1]) to the computational model of biochemical cell networks.

MS-BioNET 0.99 distribution contains the source code and the binaries of the framework infrastructure, including: a front-end compiler for the surface language, and a back-end simulation engine producing output results as a text file.

The distribution includes the following folders:

- `frontend`

- `bio-compiler.pl` — a GNU Prolog program

The front-end receives a specification file as described in previous section, parses it, checks it for correctness, and generates an intermediate file containing a list of commands to initialise the simulation engine.

- **engine**

- **src** — this folder contains the source code of the simulator
- **makefile** — this is the file to compile the engine

The engine is a simulator (written in C++): it receives the intermediate file, initialises all the proper data structures, and the proceeds with the simulation process, which is based on the optimised Gillespie's algorithm described in [1].

- **examples**

- **diffuse.bio** — this file contains the model for creating a computational field
- **brusselator2D.bio** — this file contains the brusselator model with spatial extension

- **doc**

- **getting_started.pdf** — this document

1.2 Installing and Using MS-BioNET

A Unix, Linux, or Mac OS X system is required to run MS-BioNET .

The MS-BioNET front-end is implemented in GNU Prolog ¹. To run the compiler you will need to have GNU Prolog 1.4.0 or previous versions installed on your machine.

The installation process via command-line is as follows:

1. go to MS-BioNET directory

- **cd MS-BioNET**

2. to compile the front-end

- **cd frontend**

¹<http://www.gprolog.org/>

- `gplc bio-compiler.pl`

3. to execute the front-end

- `/bio-compiler -inputfile nameinputfile -outputfile nameoutputfile
-warningfile namewarningfile -args argumentlist`

all the parameters are optional. `argumentlist` is a list of atoms separated with commas and enclosed in square bracket.

4. to compile the engine

- `cd ..`
- `cd engine`
- `make`

5. to check the result of the local compilation use

- `cd ..`
- `./frontend/bio-compiler < ./examples/diffuse.bio | ./engine/simulator > out.txt`

the expected result is a file with a set of matrices showing the time evolution of a computational field into a 20×20 grid.

Chapter 2

How to Write a Model

The model, *i.e.*, the system structure and behaviour must be specified through the following terms:

- `const size(N,M)` to create a set of $N \times M$ locations. If $N = M$ it is sufficient to specify only one variable.
- `molecule M [where Condition]` to specify the set of molecules involved in the model
- `reaction N : Pre --> Post rate R [where Condition]` to specify a reaction. `Pre` and `Post` are list of reactants and products respectively. They can be empty or composed by a different number of molecules. Among the molecules listed in `Post` it is possible to define a `left(molecule)`
- `compartment C [where Condition]` to create compartments
- `link C >>> D molecule M rate R [where Condition]` to create a link among two compartments `C` and `D`
- `link C >>> D molecule M rate R gradient L factor F [where Condition]` to create a link among two compartments considering that molecules move following a gradient.
- `concentration N of M into C [where Condition]` to set the initial concentration `N` of a molecule `M` into a specific compartment `C`
- `place R into C [where Condition]` to enable the reaction `R` into compartment `C`.

- `final time T` to set the duration of the simulation at time T.
- `final steps S` to set the number of simulation steps to execute.
- `sample steps S` and `sample time T` to set the sampling step and time to print the simulation results.
- `out Term [where Condition]` to specify the format of the output. Term can be
 - `molecule(C,M)` to print the current concentration of molecule M into compartment C;
 - `molecules(C,LM)` to print the current sum of concentrations of the set of molecules LM into compartment C;
 - `time` to print the current time;
 - `step` to print the current step of simulation;
 - `space` to leave a space between values;
 - `tab` to create the tab character;
 - `end of line` to create the end of line character;
 - `string(S)` to print string S.

Almost all the terms have the possibility to use the `where Condition` to specify a set of conditions on the variables listed before. It is useful to create multiple things of the same type with only one sentence. For instance to create a set of molecules it is possible to write

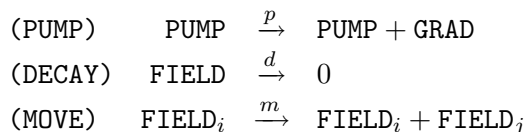
```
molecule M where (M in [mol1,mol2,mol3]).
```

Chapter 3

Examples

3.1 Example 1: Simple Computational Field

One of the simplest examples of a spatial model is the creation of a computational field, or gradient. A gradient is a field initiated (i.e. pumped) from a source node, and diffusing in the surrounding until each node of the network features a stable value. The chemical system that create a gradient is the following:



Initially, assume a **PUMP** molecule with concentration 1 that is inserted into the source node. Reaction **(PUMP)** starts spawning units of the **FIELD** molecule, such that **FIELD** concentration starts raising. On the other hand, **(DECAY)** rule makes any single **FIELD** unit disappear. Additionally, **(MOVE)** is used to move a **FIELD** unit into a neighbouring node. As a result, the computational gradient starts diffusing in the neighbourhood where it is also subject to decay.

The MS-BioNET model

This system is specified in Figure 3.1. The first line declares the grid size to be 70. Molecules **pump** and **field** are then declared. Notice that the internal interpreter invokes goal “**molecule M**” to inspect molecules’ type, which yields two solutions, binding **M** to **pump** and then to **field**—this mechanism is used in all other declarations, and is key in our language semantics.

```

constant size(70).

molecule M where (M in [pump,field]).
reaction r(pump)  : [pump] --> [pump,field] rate 10.0.
reaction r(diff)  : [field] --> [field,left(field)] rate 0.2.
reaction r(decay) : [field] --> [0] rate 0.1.

compartment c(X,Y) where (size(N), X in 1..N, Y in 1..N).
link c(X,Y) >> c(X,Y+1) rate 10000.0 molecule field.
link c(X,Y) >> c(X,Y-1) rate 10000.0 molecule field.
link c(X,Y) >> c(X-1,Y) rate 10000.0 molecule field.
link c(X,Y) >> c(X+1,Y) rate 10000.0 molecule field.
concentration 1 of pump in c(M,M) where (size(N), M is N/2).
place _ into _ .

final_steps 500000.
sample_steps 50000.
out [time,step,end_of_line].
out S where ( compartment c(X,Y), S1 = molecule(c(X,Y),field)),
              ((X=N,S=[S1,end_of_line]);S=S1)
).

```

Figure 3.1: Code for a 70x70 field diffusion scenario

Three chemical reactions are defined, one that pumps molecules of **field** if molecule **pump** is present, one that diffuses a copy of **field** in some neighbouring compartment, and one to decay **field** substance.

The **compartment** declaration defines the 70x70 grid: note that, due to Prolog resolution, they are ordered as follows:

```
c(1,1),...,c(1,70),c(2,1),...,c(70,70)
```

Then, links are declared for **field**—note that the interpreter automatically excludes links escaping the grid. First, the **concentration** declaration is used to place one molecule of **pump** in the centre of the grid, while all other concentrations are set to 0 by default. The **place** declaration is then used to place all defined reactions in all compartments. Finally, we define a total number of simulation steps, the number of steps between two observations, and printing commands: the last **out** declaration emits the value of **field**

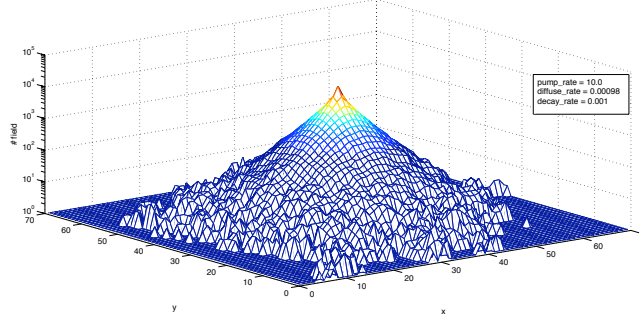


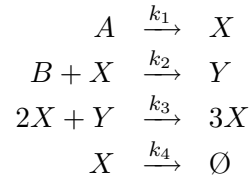
Figure 3.2: A computational field.

in each compartment in the right order, also producing an end-of-line at the end of each row—recall that in Prolog, “;” stands for logical disjunction.

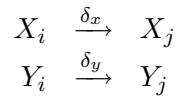
The whole specification is also reported in the file `field.bio` inside the `examples` folder. A result of the model execution (with a different set of parameters) is shown in Figure 3.2.

3.2 Example 2: 2D Spatial Brusselator

The reactions for the theoretical Brusselator model are



and the two reactions for modelling the interactions among Brusselator nodes through the exchange of X and Y molecules



where i, j are indices of compartments disposed in a 2D grid.

The MS-BioNET model

We could encode the spatial Brusselator model in MS-BioNET as follow. To define the molecules of the system we use:

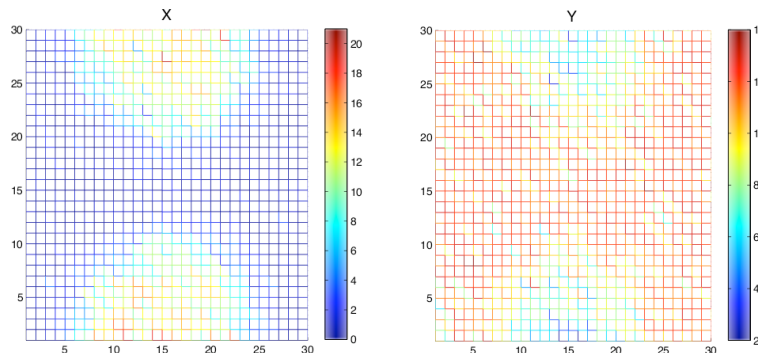


Figure 3.3: Spatial brusselator.

molecule M where (M in [a,b,x,y]).

The complete list of the model parameters is specified as:

```
parameters([k1,k2,k3,k4,deltax,deltay]).
```

The reactions of the theoretical unicellular model are:

```
reaction r(a) : [a] --> [a,x] rate R where (param(k1,R)).
reaction r(b) : [b,x] --> [b,y] rate R where (param(k2,R)).
reaction r(c) : [x,x,y] --> [x,x,x] rate R where (param(k3,R)).
reaction r(d) : [x] --> [] rate R where (param(k4,R)).
```

where A and B appears in the reactions in both Pre and Post list as soon as their concentration is assumed to not vary with time. The diffusion reactions are:

```
reaction r(diff1) : [x] --> [left(x)] rate R where (param(deltax,R)).
reaction r(diff2) : [y] --> [left(y)] rate R where (param(deltay,R)).
```

The complete MS-BioNET specification of the model can be found in `brusselator2d.bio` file into the `examples` folder. To execute the model instantiating the values of the parameters use

```
./frontend/bio-compiler -args [4.0,15.0,0.4,1.0,30.0,50.0]
< ./examples/brusselator2d.bio | ./engine/simulator > out.txt
```

A result of the model execution is shown in Figure 3.2.

Bibliography

- [1] M. A. Gibson and J. Bruck. Efficient exact stochastic simulation of chemical systems with many species and many channels. *The Journal of Physical Chemistry A*, 104(9):1876–1889, March 2000.