Introduction

This work has been supported by the project CARESSE (MISTIC-UJF pole).

A dynamical system is a fixed rule that describes what future states follow from the current state. Its evolution is described by a fixed phase space and it is supported by a static graph. But in many applications, including communication networks, embedded systems or biological behaviors (such as collaboration or communication between ants, behavior of a set of cells which share local information, etc.) graphs are subject to discrete changes, such as additions or deletions of agents or links. In the last decade there has been a growing interest in such dynamically changing graphs. The difficulty of such structures is to deal, simultaneously, with the evolution of a number of agents (or nodes), the number of links between agents, the states of agents, and eventually, the states of links.
For biological problems, specific particular models have been developed such as L-systems, adaptive dynamics, TreeGCS (hierarchical Growing Cell Structures) and DS2 (Dynamical System with a Dynamic Structure) . On the other hand, for robotic problems, graph grammar theory has been used.
In this software, we propose a framework called Dynamical system based on dynamic graphs such that agents and links between agents corespond, respectivly, to nodes and edges of a graph.

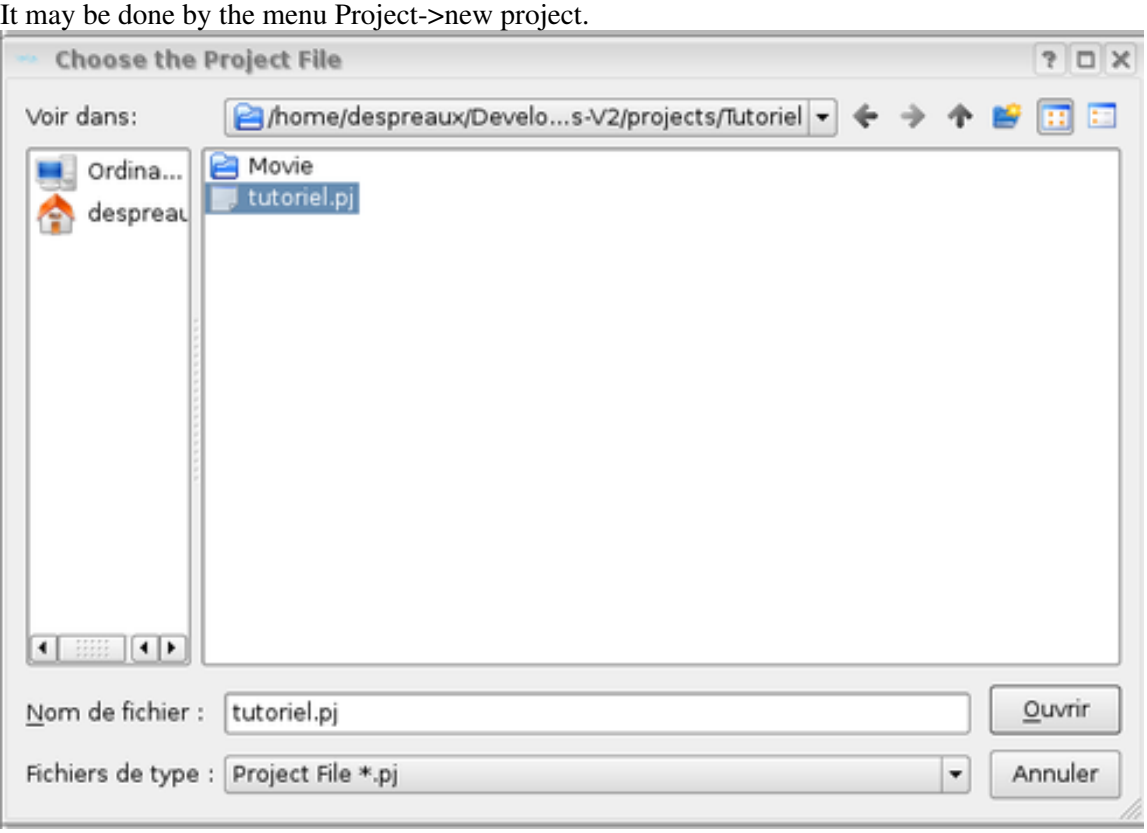
- Each node of the graph obeys a dynamical system,
- Each edge of the graph obeys a dynamical system,
- The evolution of nodes and edges is determined thanks to the state of there neighborhood.
- Under specific conditions, the graph evolves thanks to transformation rules.

Nodes and edges can be added to or removed from the graph. The graph evolution is determined thanks to local transformation rules. Our aim is to model agent reactions due to local informations. Consequently, transformation rules are function of a node and its neighborhood, or, of an edge and its neighborhood. We have implemented the framework of Dynamical system based on dynamic graphs in a program called DynSys. In the following parts, we will present this software:

- the first part is devoted to the configuration of Dynsys.
- the second part explains how to create a dynamical system
- the third part presents some examples
- the last part explains how to install the software and how to install it on various platforms (linux,windows)

Project Managment

Dynsys uses a project approach. A project deals with components of a dynamical system: states equations, adjacency matrix and graph visualization. To create a project, it is necessary to create a path for the project and in this path, save the project file.
It may be done by the menu Project->new project.

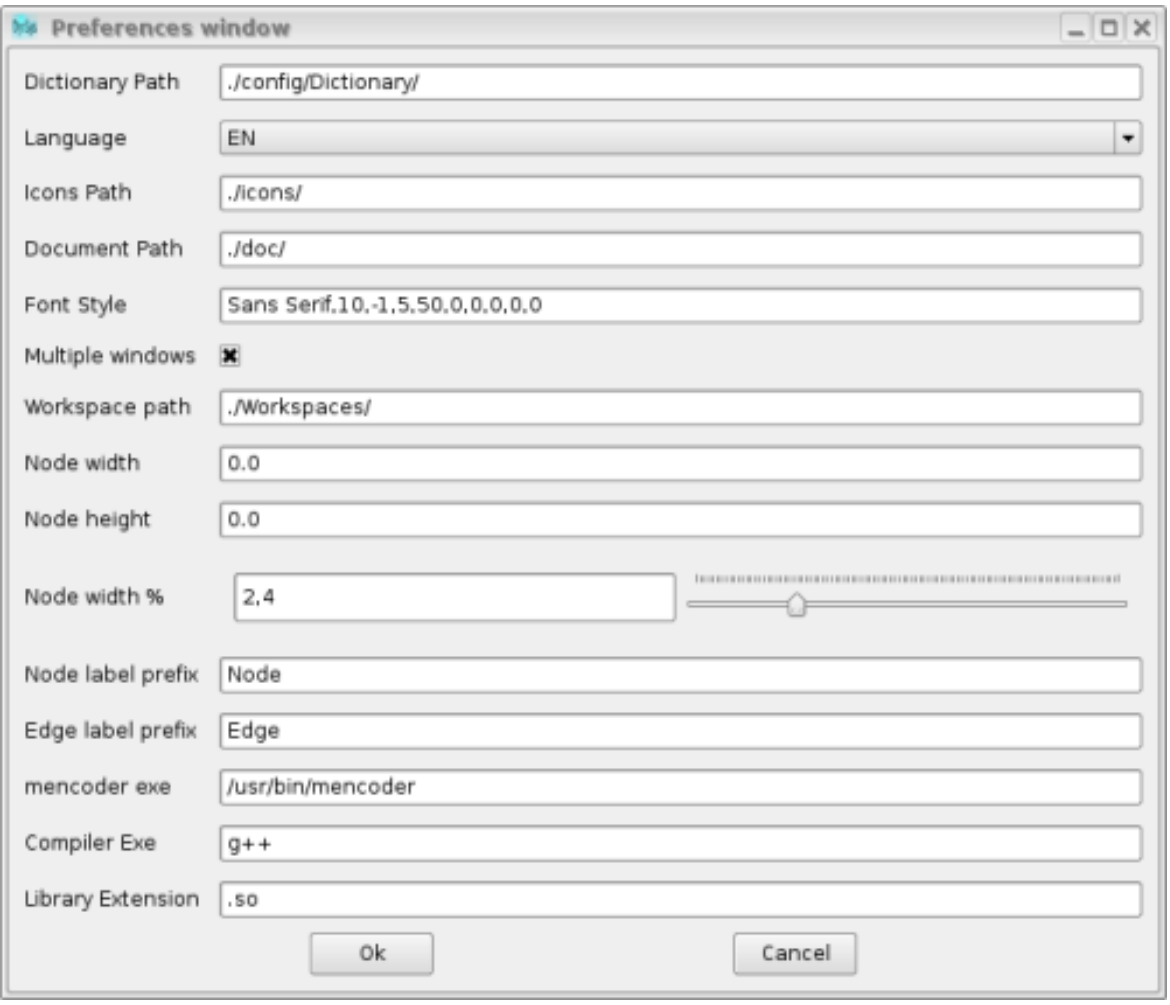


In the path of the project are created a path Movie used to store the dynamic graph iterations used to display graph.
The project management is as usual: it is possible to save, to copy, to load or create a project.

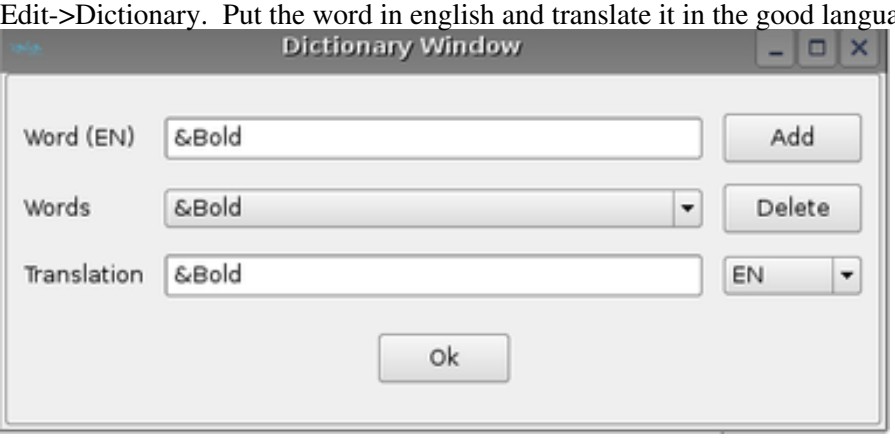
Preferences Managment

The configuration of DynSys is done thanks to the preferences menu Edit->Preferences

2



DynSys is multi-langage, the first field indicates the path of the dictionary files. By double clicking on the line edit, it is possible to change this path. The language supported are the french and the english. It is possible to change the dictionary thants to the menu Edit->Dictionary. Put the word in english and translate it in the good language.

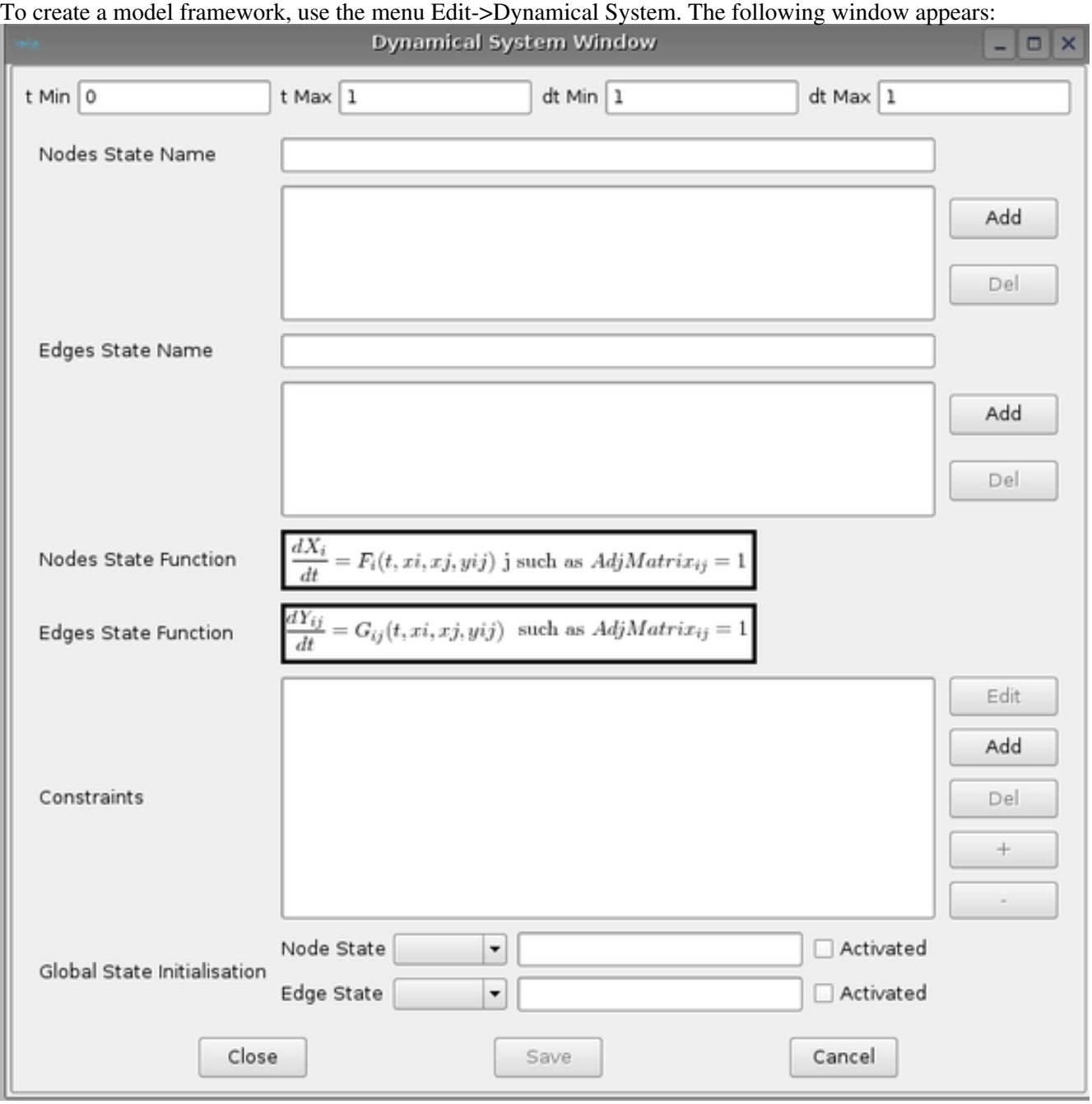


The icons of the software are stored in the path mentioned in the field *icons path* the software
The document path is the path where is the documentation.
The style of the software may be changed by double-clicking on the font style line edit.
Multiple windows allows to specify if all the windows are free or contained in one main window.
Workspace path is the path where windows configuration models are stored.
Node with, node height and node width % are the default attributes for node geometry. Node width % is the max size percent value of screen size of the node.
Node label prefix is the default label of the node.If no name is set, nodes have no label.
Compiler exe is the executable to compile the state functions and library extension is the extension of the compiled library. Those 2 fields depend on the platform.

Workspace Managment
A workspace is an image of the position and size of the windows at a time. it is possible to save, restore and delete workspaces accesible by the menu Project->Load/Save/Delete workspace.

states equations

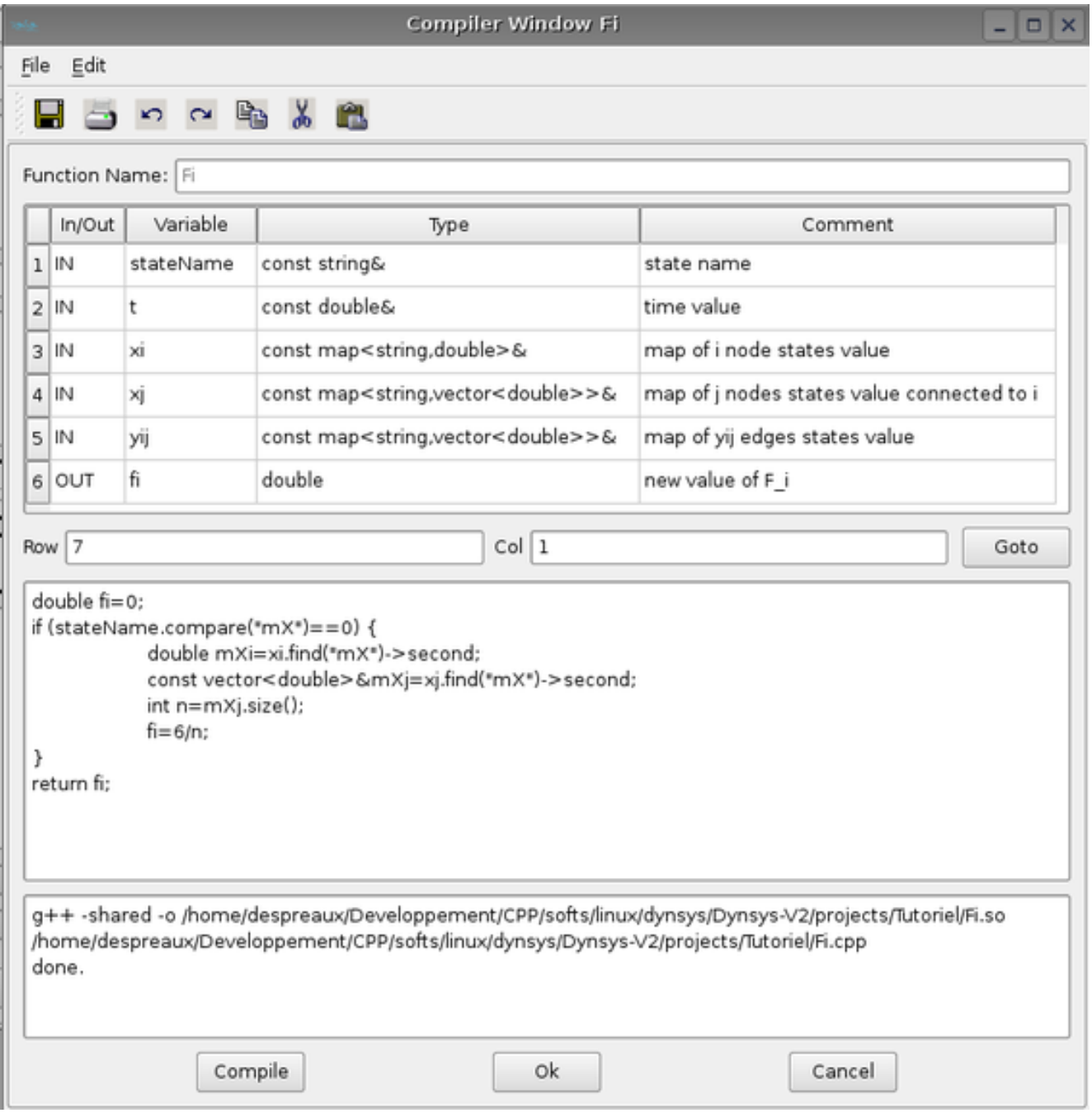
To create a model framework, use the menu Edit->Dynamical System. The following window appears:



The 2 first fields reprensent the bounds of th interval of simulation and the 2 last fields are the min and max time step to make computation.

The user can define the state name of the nodes and edges by filling in the name in the line edit Nodes State Name or Edges State Name and clicking on the button Add. To delete a state name select it in the list box and click on the corresponding Del button. To define the state functions, click on the corresponding button. The following window appears:

4

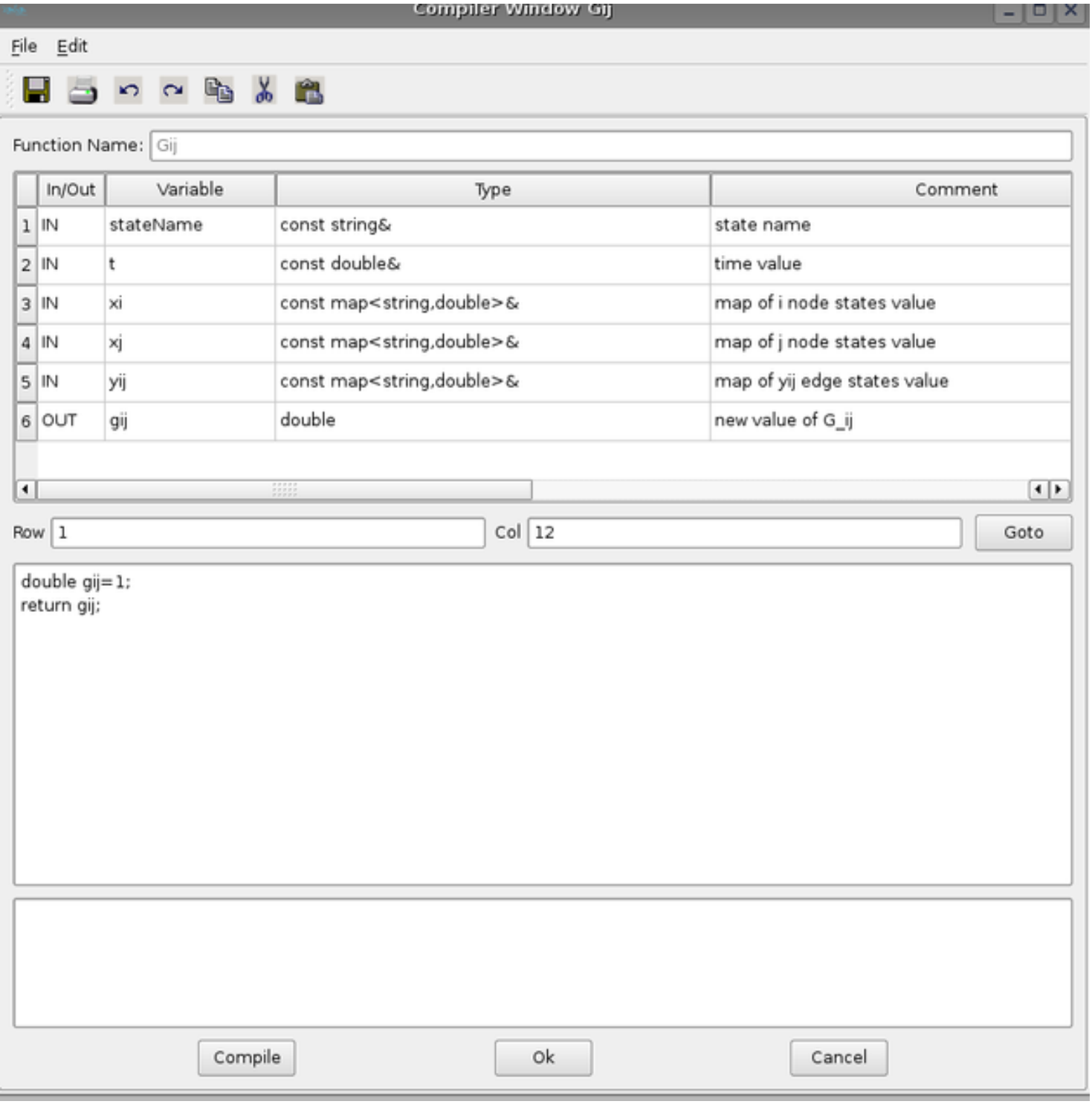


Available variables are listed in the panel:

- the stateName variable is the name of the state to compute.
- t variable is the time value of computation
- xi variable is the state value of the node i in which the state name are computed. User has to specify the state variable in that way *double mXi=xi.find("mX")*
- xj variable is a vector of state values corresponding to nodes j connected to node i. Its size is the number of connections of the node i.
- yij variable is a vector of state values of the edges connected to node i and j. Its size is exactly the same of the size of xj.
- fi is the computed state value stateName variable of node i.
-

A variable can only be read if IN is written in the IN/OUT column. Its value can be changed if OUT is written.
The body of the function must be written in C++ language. To compile it, click on the Compile button. It use the compile command provided in preferences window. To close the window click on the Ok button, to cancel the function click on the button.

To provide the edge state function click on the corresponding button and the new window (on the same model of nodes state function) appears:



Available variables are listed in the panel in order to specify the dynamic of the edge [i,j]:

- the stateName is the name of the state to compute.
- t is the time value of computation
- xi is the state value of the node i
- xj is the state value of the node j
- yij is the state value of the edge [i,j]

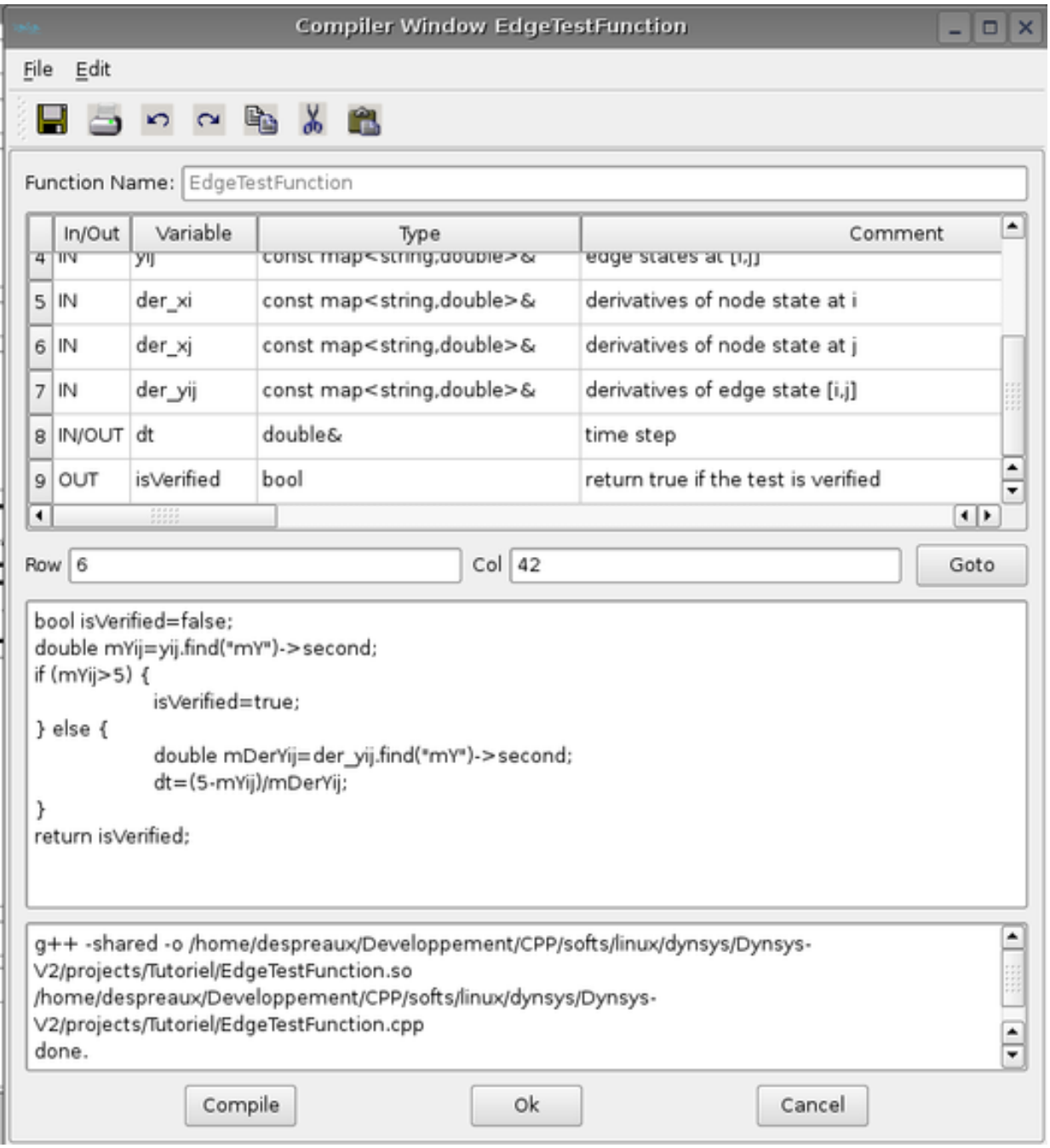
  i and j. Its size is exactly the same of the size of xj.
- gij is the computed state value stateName variable of edge [i,j]
-

The body of the function must be written in C++ language. To compile it, click on the Compile button. It use the compile command provided in preferences window. To close the window click on the Ok button, to cancel the function click on the button.

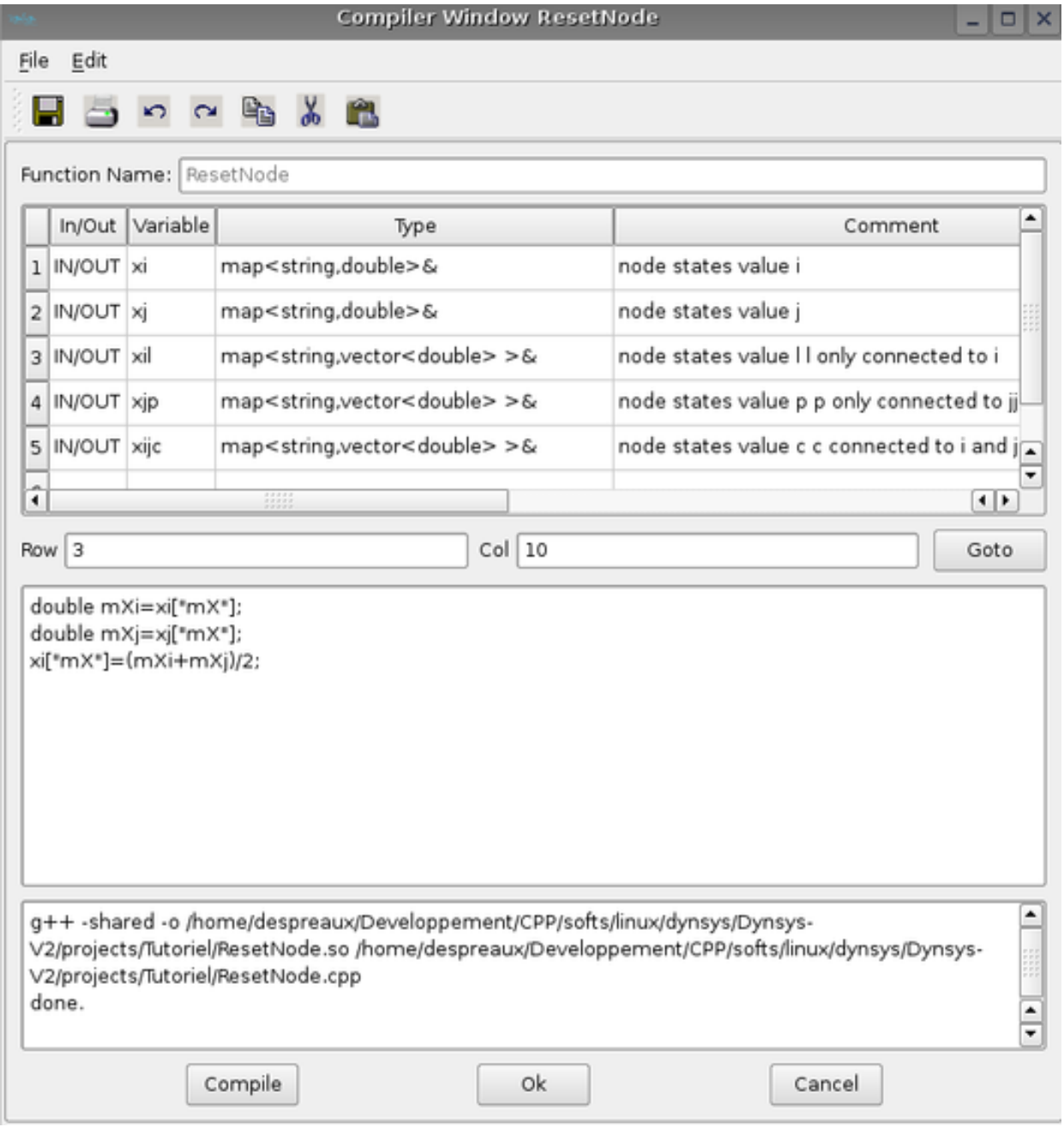
Once the states equations are entered, the user has to define the graph transformation. The different rules are:

For each rules, a constraint, reset node and edge must be defined in agreement with the rule.
For example for the constraint of an edge fusionning, we have 9 variables:
t is the time value of the iteration
xi is the i node states of the edge [i,j]
xj is the j node states of the edge [i,j]
yij is the edge states of the edge [i,j]
der_xi, der_xj,der_yij are the derivatives values of the states
dt is the new computed time step for the prediction time where the constraint test seems to be verified (linear approach)
isVerified is the return value which has to be true if the test is verified, false otherwise.
The aim of this prediction is to capture all the graph transformation.

7



For this rule, a node reset function has to be provided:
This function has 5 variables:
xi the state values of node i of edge [i,j]
xj the state values of node j of edge [i,j] (the node j will disappear, so xj can not be modified)
xil the state values of node l where l is connected to i but not to j
xjp the state values of node p where p is connected to j but not to i
xijc the state values of node c where c is connected to i and j



For this rule an edge reset function has to be provided:
This function has 6 variables:
yij the state values of edge [i,j] (ReadOnly - the edge will disappear)
yil the state values of edge [i,l] where l is only connected to i
yjp the state values of edge [j,p] where p is only connected to j (ReadOnly - the edge will disappear)
yip the state values of edge [i,p] where p is only connected to j (Edge will appear)
yic the state values of edge [i,c] where c is connected to i and j
yjc the state values of edge [i,c] where c is connected to i and j ReadOnly - the edge will disappear)

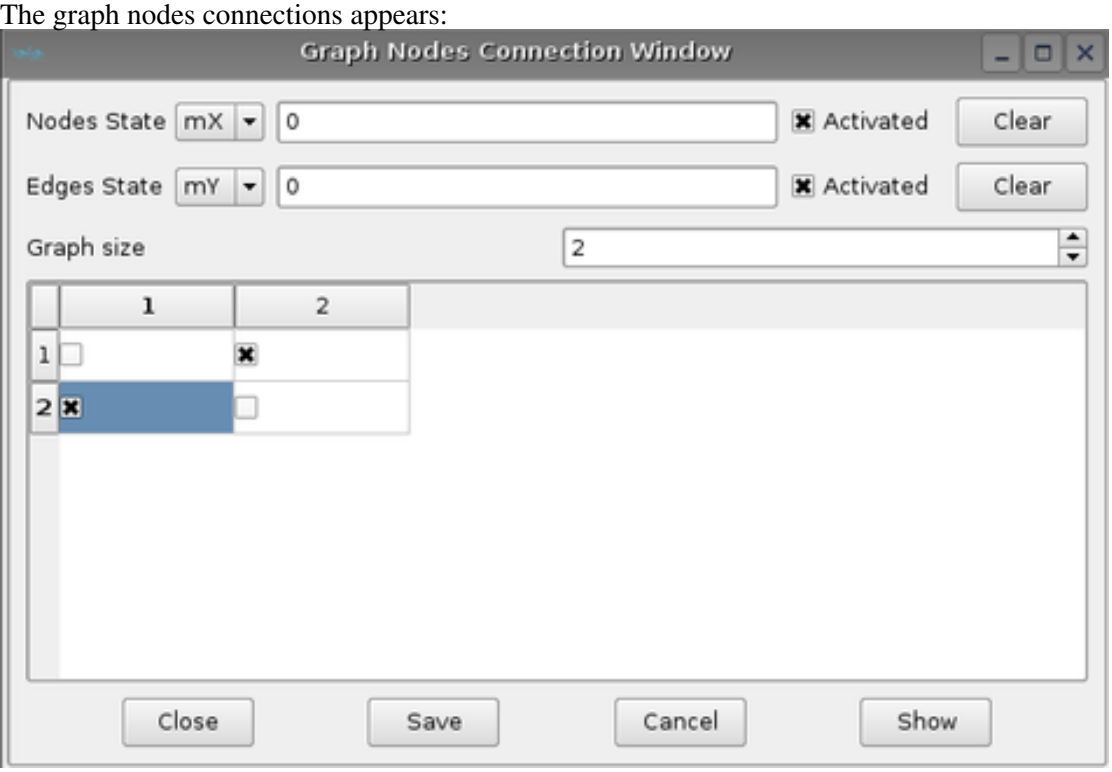
Then to Add the rule, click on the Add button.
To edit / Delete the constraint, click on the Edit/ Del button of the dynamical system window.
When many constraints are added, the order of the constraints are primordial. That's why it is possible to modify the priority of the constraint by clicking on +/- buttons. The position of the rule in the list determine its order of priority.

It is possible to use the global node or edge states initialisation instead of local initiatialisation by selecting the components of .

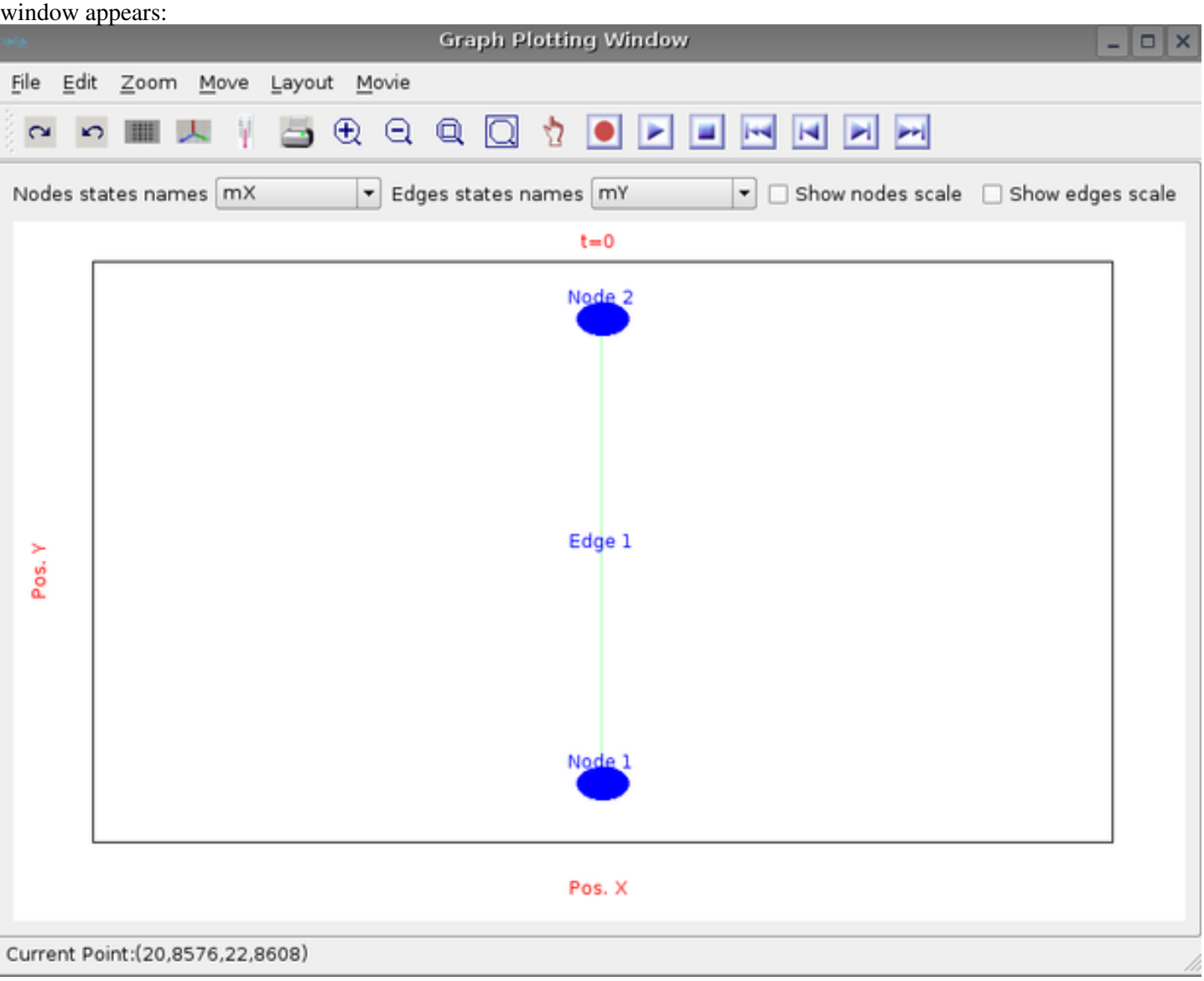
Don't forget to save the dynamical system in order not to lose your job !

Adjacency Matrix

The initial graph of the dynamical system is set thanks to the menu Edit->Graph Nodes Connection.
The graph nodes connections appears:

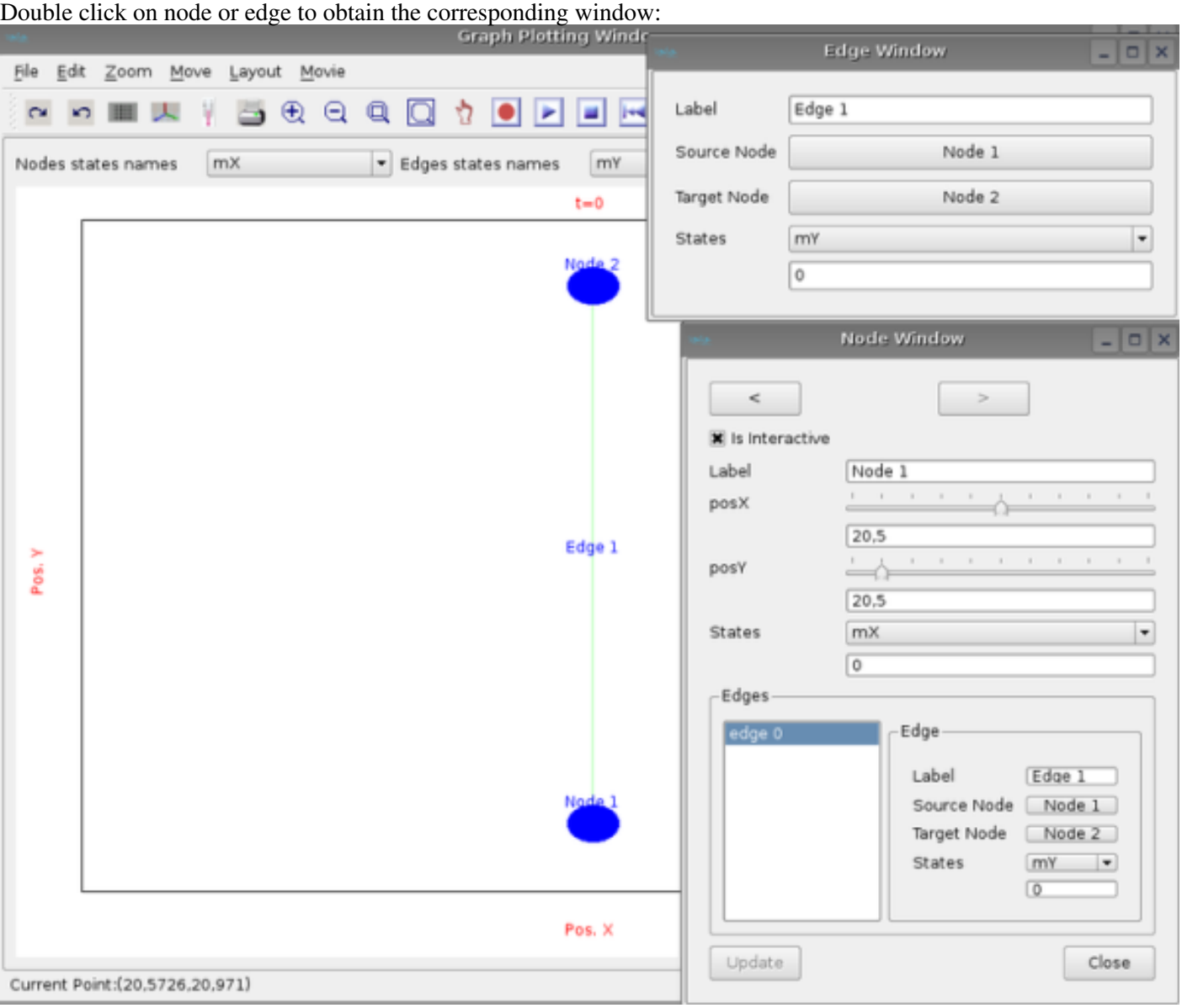


To set the dimension of the initial graph, put the right value in the spin box Graph size; To add a connection between node i and j click on the matrix check box. As the edge are unoriented, the matrix is symetric.
To init all the nodes or edges states values , click on activated check box. If not see the graph by clicking on show and the graph window appears:



10

Double click on node or edge to obtain the corresponding window:



All the properties of the node or edge can be changed thanks to those windows.

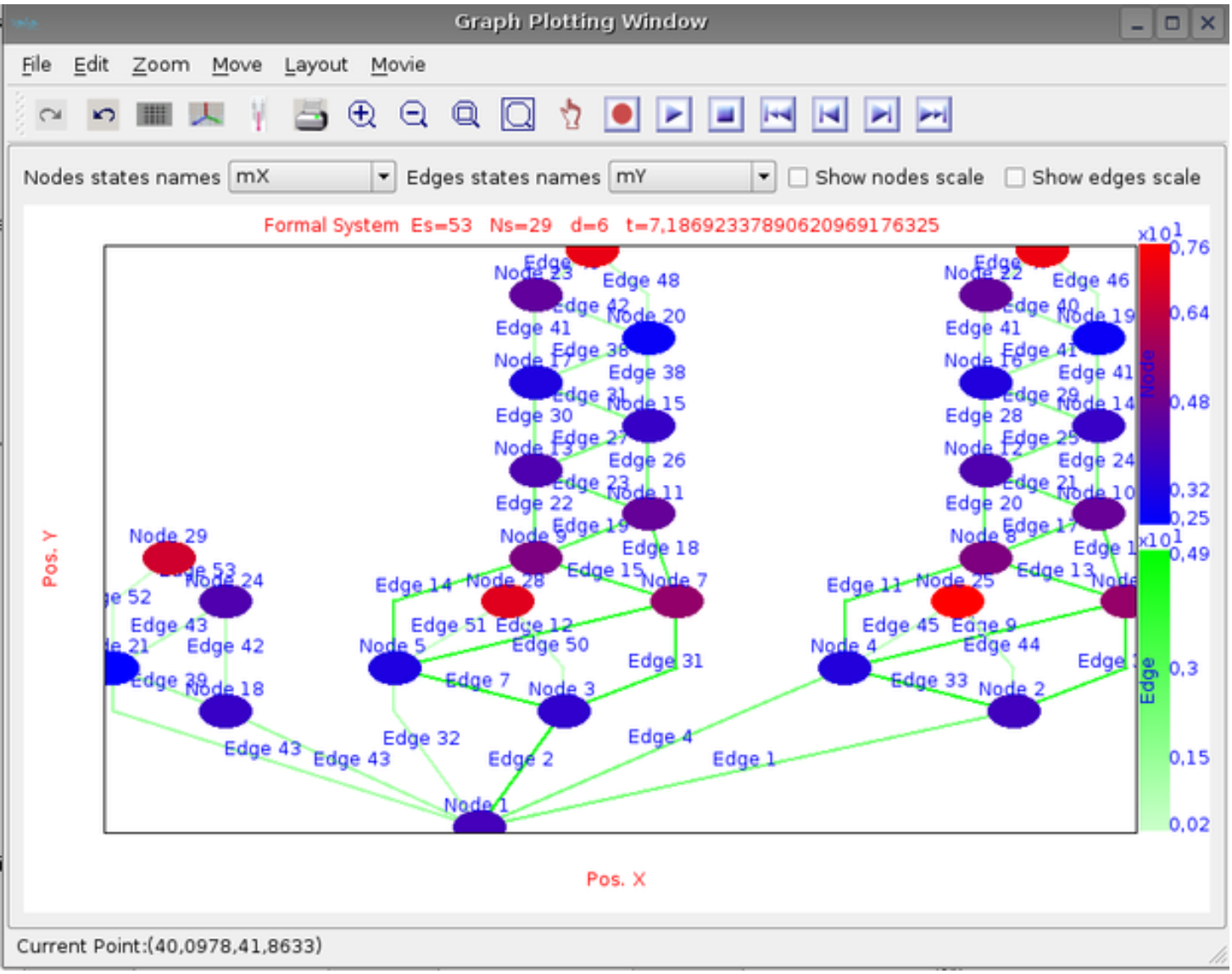
Notice that at the beginning the position of the nodes in the graph are not computed. There is 2 ways to compute the position of nodes:

- either use the layout of graph thanks to menu layout in the graph window. 4 layouts are available an empty, an hierarchic, an orthogonal layout and an energy based layout. Empty do nothing place the nodes as youuu want thanks to the node position cursor in node window, the hierarchy layout try to display the nodes in a hierarchic way, the same thing is done concerning the orthogonal layout. the energy based layout tried to minimized a criterium function.Other layouts will be implemented in a next future.
- or use 2 particular states: mPosX and mPosY and define the dynamic equations of those 2 states.

graph visualization

To simulate the evolution of the dynamical system, click on the menu Simulation->Run. The system will interactivly evoluate on the screen. It is possible to stop the simulation by clicking on the Cancel button of the progress window.
The graph evolution appears on the window; It is possible to browse the files thanks to the movie button on the right corner of the tool bar of the graph plotting window:

11



To zoom in click on the icon +, to zoom out, on the button - , to move on the window click on the drawing area and move the mouse. to select a region , unselect the hand icon and select the zoom square icon then select an are by a red rectangle. Many options are available.

It is possible to record all the images in one movie file by clicking on the record button (red point) or Movie->record. In order to work, you have to have mencoder installed on your computer and to put its path in the corresponding line edit in preferences panel. To play it, use the command on linux: "mplayer -vo x11 Movie/movie.avi".

Analysis

Some tools are available to make analysis on graph. Eigen values of the graph may be computed. Other tools will be implemented in a next future.

12

We want to simulate a dynamical system with only one state on nodes call mX.

The equations of the state on the node i is:
dmX/dt=2-n/2 where n is the number of connection of node i.

The Fi function code is:

```
int n=xj.find("mX")->second.size();

double fi=2-(((double)n)/2);

cout << "fi:"<<fi<<"\n";

return fi;
```

There is only one constraint which is *adding a node j to the node i* if the state value of node i is greater than 10.

If the constraint is not verified, a maximum time step is computed to forseen when the next constraint at the node i will appear.
The test function of the constraint is:

```
bool v=false;

double x_value=xi.find("mX")->second;

if (x_value>=10) {

v=true;

cout << "constraint verified ! \n";

} else {

int n=xj.find("mX")->second.size();

double fi=2-(((double)n)/2);

double dx=10-x_value;

if (fi>0) dt=dx/fi;

cout << "constraint unverified !\n";

}

return v;
```

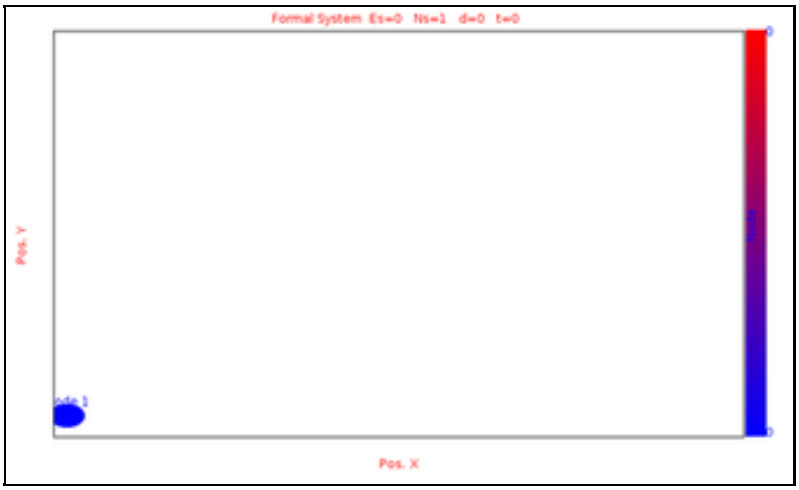
When the constraint is verified, a local reset node function has to be defined. In that case, we set the i state node to 8 and the new node has its state initialized to 2. We set this code to the node reset function of the constraint:

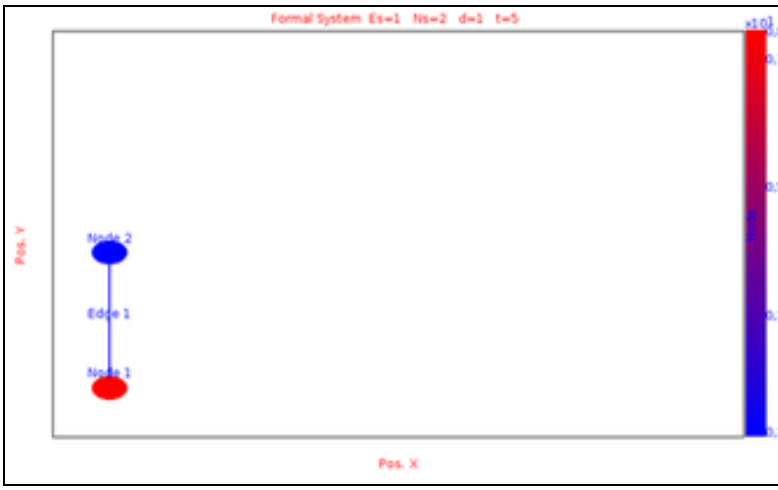
```
xi=8;
```

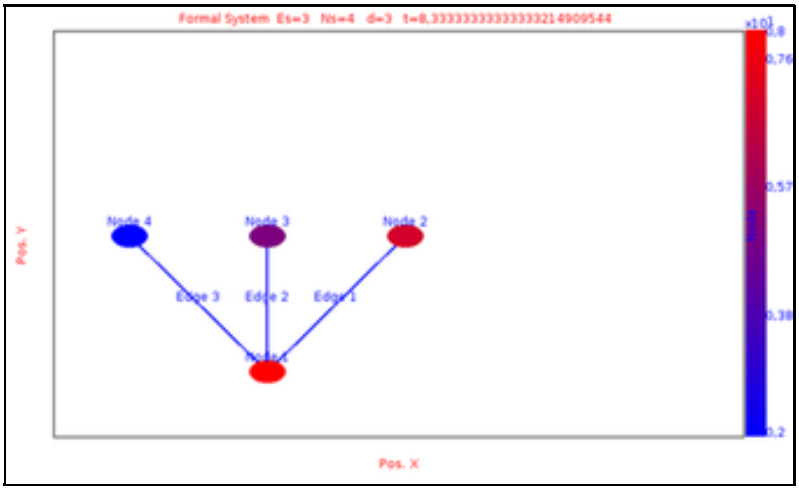
xj=2;

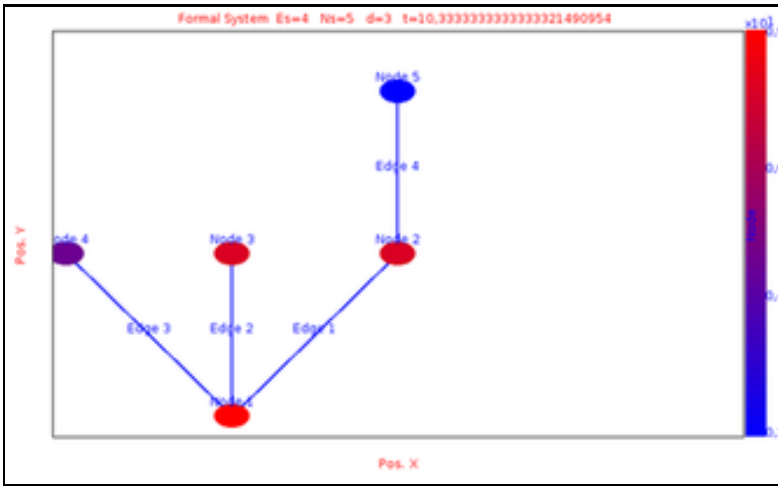
In this case, we do not want to initialize all the state to a predefined value. So the global state initialisation is not activated.

We simulate the evolution of the system between time 0 and time 13 with max time step sets to 1 and min time step sets to 0.1
We obtain the following results at different time by choosing the layout equals to Hierarchical Layout:









14

An example case 2

We want to simulate a dynamical system with only one state on nodes called mX and one state on edge called mY

The equations of the state on the node i is:
dmX/dt=6/n where n is the number of connection of node i.

The Fi function code is:

```
double fi=0;

vector<double> xjs=xj.find("mX")->second;

int n=xjs.size();

if (n>0) fi=6/n;

return fi;
```

The state eqution of edge is given by the equation: dmY/dt=1
The Gij function code is:

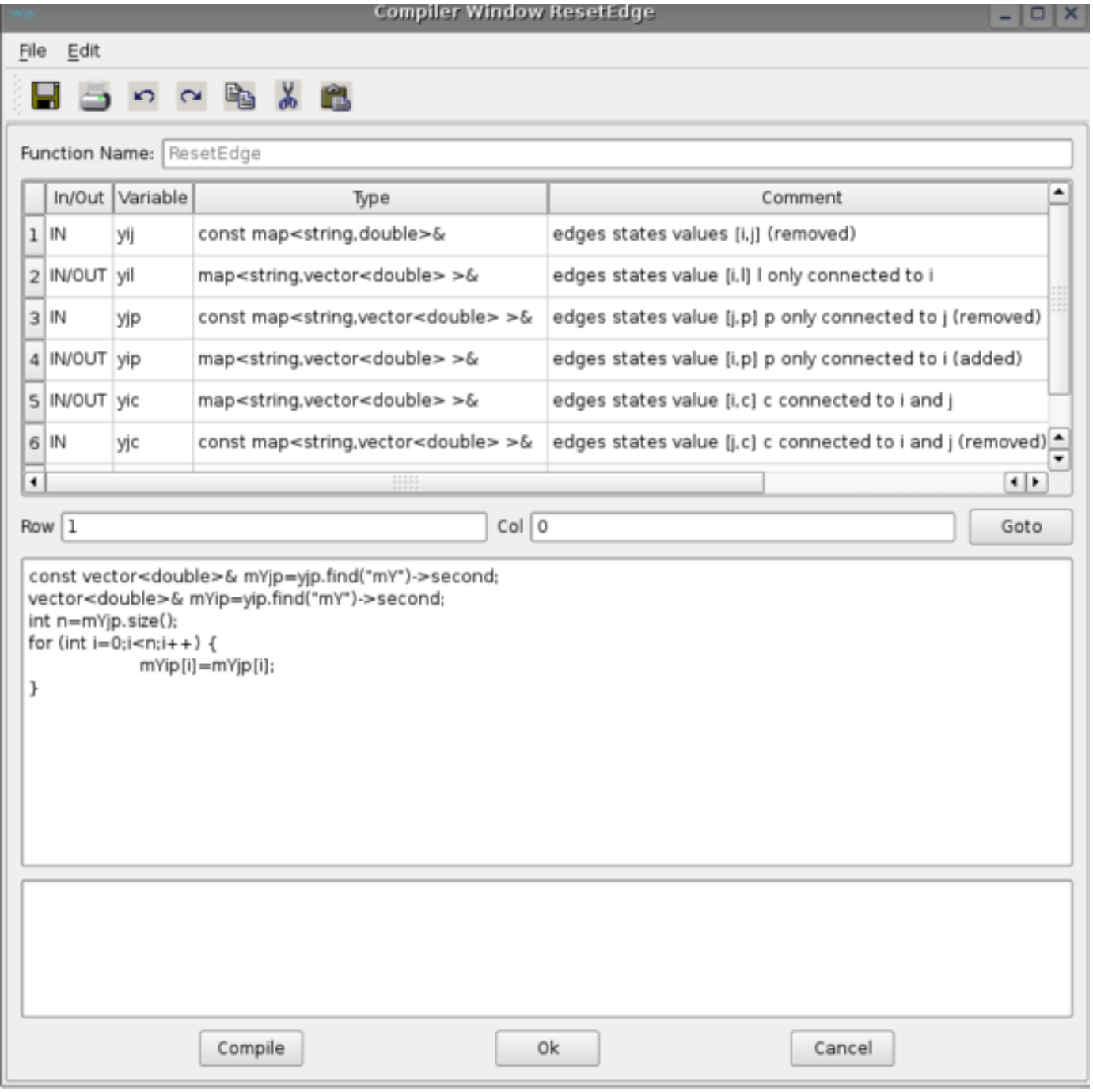
```
double gij=1;

return gij;
```

There are 2 constraints. the first constraint is an edge fusion that means that if the edge [i,j] verifies its test function then , the node j disapears and all the nodes connected to j are connected to i the value of the new edge created is the same as the old edge deleted. The function test is a follow:

```
double y_ij=yij.find("mY")->second;

bool isVerified=false;

if (y_ij>=5) {

isVerified=true;

} else {

double der=der_yij.find("mY")->second;

if (der!=0) dt=(5-y_ij)/der;

}

return isVerified;
```

the reset edge function is as follow:

15



The secund constraint is an edge splitting: the edge [i,j] verified its test function then a node k is created and also 2 edges between node i and k and node j and k.
The test function is as following:

```
double x_i=xi.find("mX")->second;

double x_j=xj.find("mX")->second;

double xprime_i=der_xi.find("mX")->second;

double xprime_j=der_xj.find("mX")->second;

bool isVerified=false;

if ((x_i+x_j) >=12) {
```


16

isVerified=true;

} else {

double der=(xprime_i+xprime_j);

if (der!=0) dt=(12-x_i-x_j)/(xprime_i+xprime_j);

}

return isVerified;

The state of the added edges are set to 0.

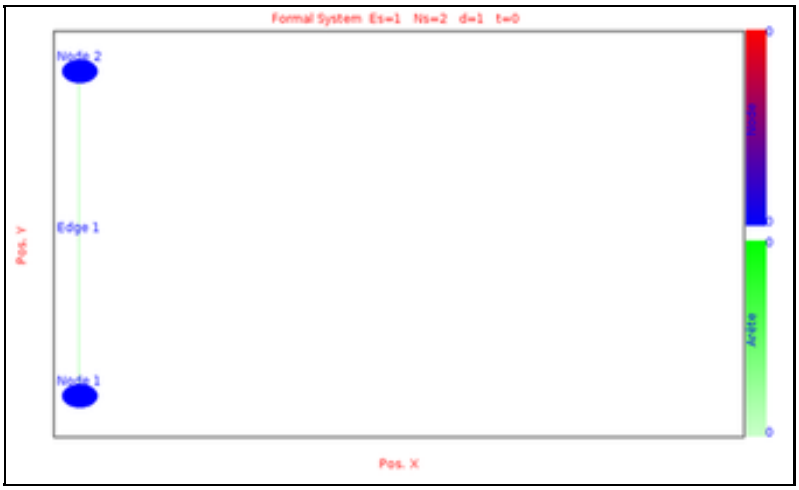
yil=0;

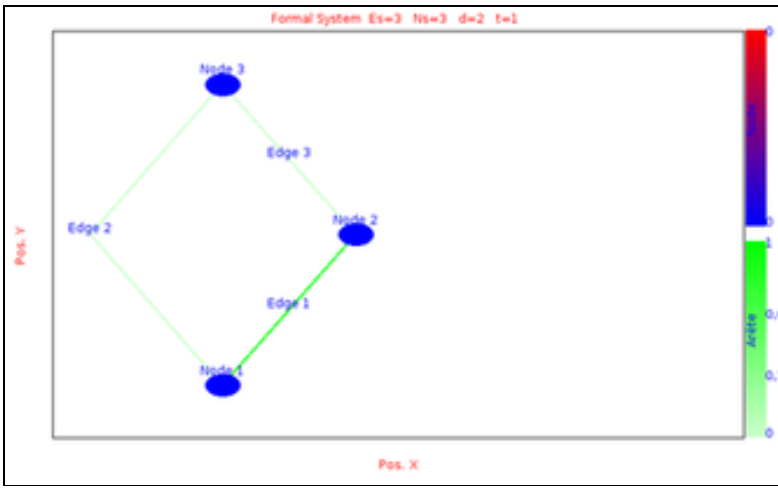
yjl=0;

The state of all nodes are set to 0.

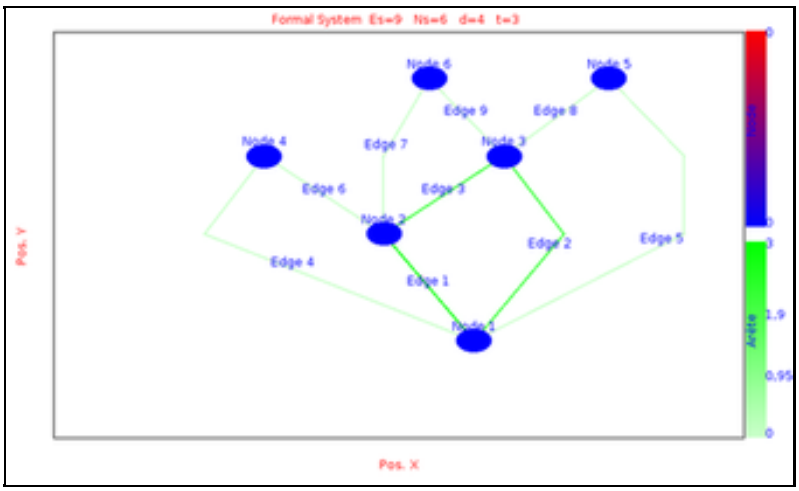
So the global state initialisation is activated and at each graph transformation the node are set to 0.

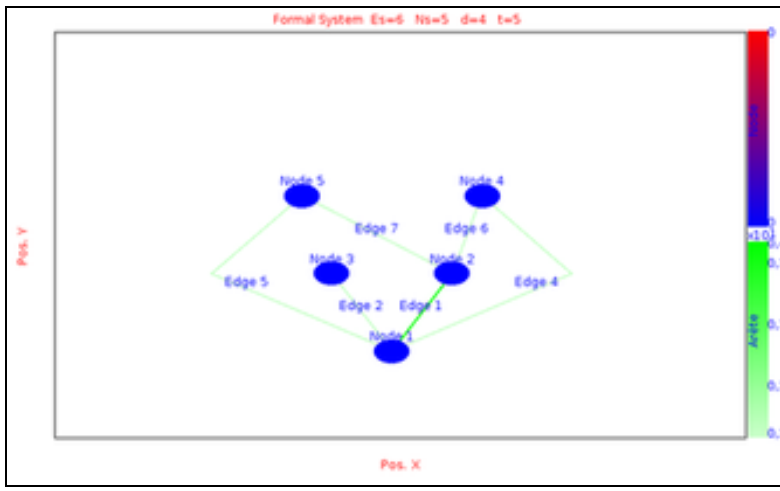
We simulate the evolution of the system between time 0 and time 15 with max time step sets to 1 and min time step sets to 0.1
We obtains the following results at different time by choosing the layout equals to Hierarchical Layout:

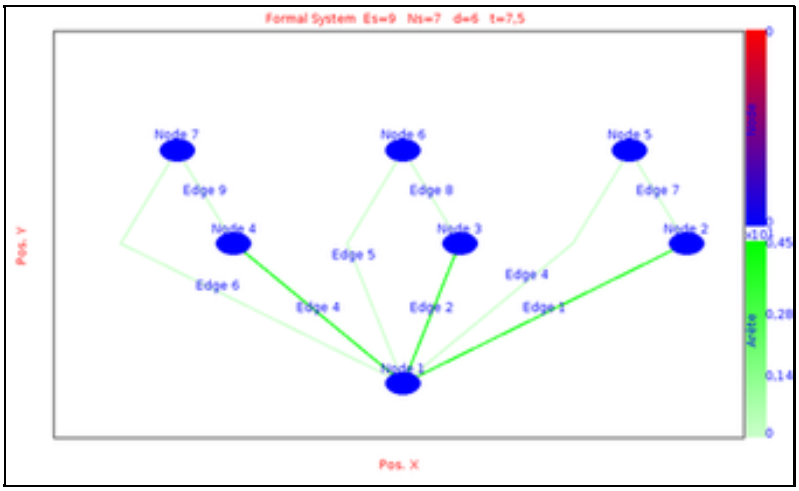


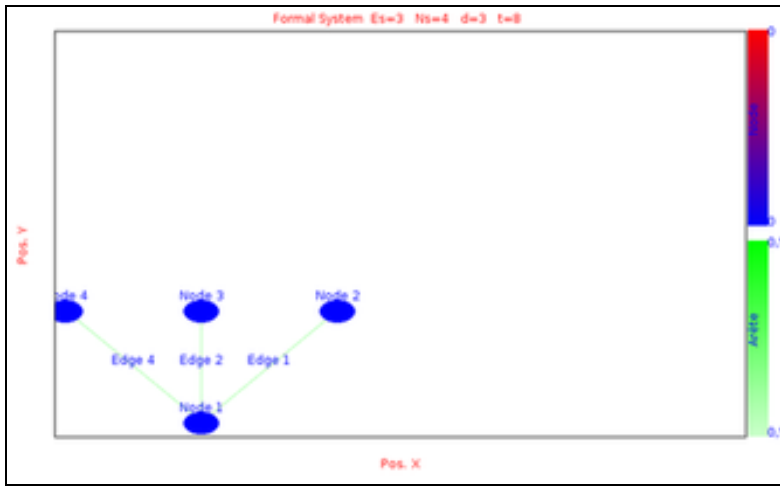


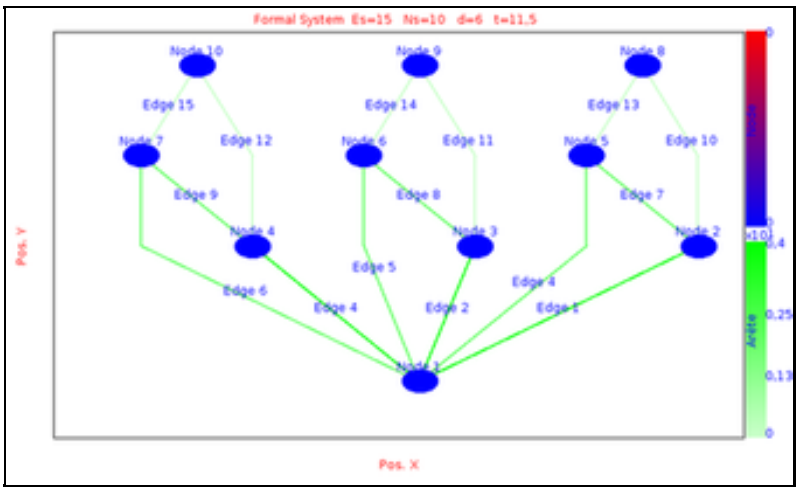
17

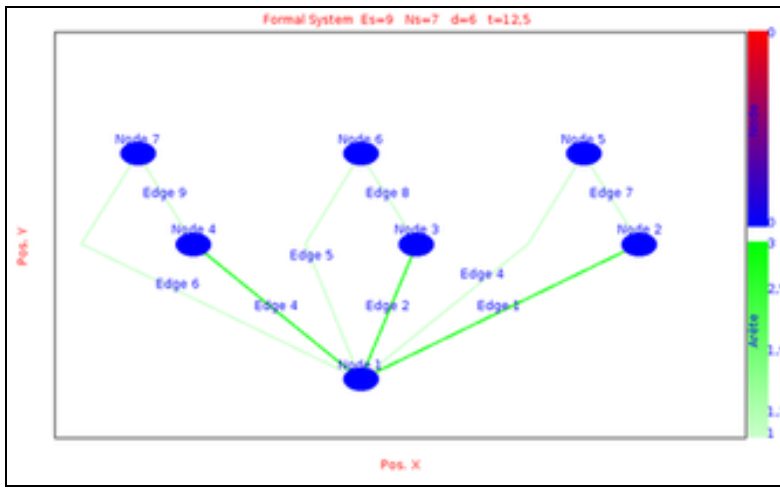












We notice that at time t=14.5 is the same graph as time t=6.

18

Dynsys

Downloading and installing DynSys

Linux

Download the archive file.

Installation:

untar the tar.gz file and execute ./install.sh to create the links to libraries needed by dynsys.exe.
To launch it, execute the command ./dynsys.sh .
To work, you need to install a g++ compiler to compile the state equations, mencoder to register images into a film and mplayer to visualize the film.
If any problem, contact Dynsys team

Window

Download and execute the setup file.
To launch the application click on the DynSys icon on the desktop.

Installation:

A C++ compiler is needed (MinGW is available in install path). The mencoder exe file is also provided in exe path. You have to install mplayer to visualize the graph movie.

the pdf documentation is available here.

Dynsys releases:

0.24 linux version - **0.24 window version** (15/03/10)
- Euler Method is re-introduced

0.23 linux version - **0.23 window version** (15/01/10)
- Euler Method are replaced by Runge Kutta 5 or radau 5 methods
- dtmin is the minimum time step for graph transformation detection
- import and export file with gml format
- random initialisation of value of state nodes and edges

0.22 linux version 0.22 window version (22/09/09)
- Bug correction for fusion edge constraint

0.21 linux version 0.21 window version (02/06/09)
initial version.

linux version 0.10 (26/09/08)
Old version

If any bug found, sent information to Dynsys team
Title:
Version:0.20
OS: Windows XP, Windows Vista...
Priority: (low, normal, high)
Severity: (low, normal, high)
Description: a little description of the bug. You need also to send the project file project.pj

19