Università degli Studi di Pisa

Dipartimento di Informatica
Dottorato di Ricerca in Informatica

Software Documentation

The Grid System Simulator

Version Date: June 3, 2015

Suryana Setiawan

Contents

1 Introduction **1**
1.1 Terminologies . . . 2
1.2 Software's Organization . . . 3
1.2.1 Data Structures . . . 3
1.3 Notes about Some Bugs . . . 4

2 User Interfaces **5**
2.1 The Dashboard . . . 5
2.1.1 "General" Mode . . . 5
2.1.2 "Object n Ranges" Mode . . . 6
2.1.3 "Trans. Rules" Mode . . . 8
2.1.4 "Init. State" Mode . . . 8
2.1.5 "Simulation" Mode . . . 10
2.2 A Practical Guideline . . . 11
2.2.1 Defining Grid's Dimension . . . 11
2.2.2 Defining Objects . . . 12
2.2.3 Defining Regions . . . 13
2.2.4 Defining Constants . . . 14
2.2.5 Defining Variation Ranges . . . 16
2.2.6 Defining Rules . . . 16
2.2.7 Defining Initial Objects . . . 19
2.2.8 Defining Links . . . 20
2.2.9 Defining Event Rules and Scheduling Events . . . 20
2.3 Simulating the Model . . . 22
2.3.1 "Growth" sub-mode . . . 23
2.3.2 "Animation" sub-mode . . . 23
2.3.3 "Tracing Reactions" sub-mode . . . 26
2.4 Generating LaTeX Codes of the Model . . . 29

3 Encoding the Model **33**
3.1 Basic Notations . . . 33
3.1.1 Document Root . . . 33
3.1.2 Dimension Part . . . 33
3.1.3 Author Part . . . 33
3.1.4 Description Part . . . 34
3.1.5 Objects Part . . . 34

3.1.6 Reaction Rules 35
3.1.7 Initial Configuration 36
3.1.8 Simulation Parameters 37
3.2 Advance Notations 38
3.2.1 Aliases Part 38
3.2.2 Region 39
3.2.3 Constant 40
3.2.4 Constantlist 40
3.2.5 Variation 41
3.2.6 Template of Rules 41
3.3 Scheduled Events 42
3.4 Document Type Definition (DTD) 43

4 Appendix A **47**

List of Figures

2.1 The Dashboard: “General” Mode 6
2.2 The Dashboard: “Objects n Ranges” Mode 7
2.3 The Dashboard: “Trans. Rules” Mode 8
2.4 The Dashboard: “Init. State” Mode 9
2.5 The Dashboard: “Simulation” Mode with “Growth” sub-mode 10
2.6 “Simulation” Mode with “Animation” sub-mode 11
2.7 “Simulation” Mode with “Trace Reactions” sub-mode 12
2.8 Defining the Dimension 12
2.9 Defining an Object 13
2.10 Defining a Region (“Edit Mode”) 14
2.11 Defining a Region (“Select Mode”) 15
2.12 Defining a Constant 16
2.13 Defining a Variation Range 16
2.14 Defining a Rule 17
2.15 Rule’s Variable 18
2.16 Rule’s reactants 18
2.17 Rule’s Products (or Promoters or Inhibitors) 19
2.18 Defining Initial Objects 20
2.19 Defining Initial Objects 21
2.20 Defining Initial Links 21
2.21 Output as Multisets 24
2.22 Output as Commas Separated Values (CSV) 24
2.23 Output as Chart 25
2.24 Animation of Simulation 25
2.25 Rules’ Frequencies 26
2.26 Current Populations 27
2.27 Displaying the Links 27
2.28 After Initialisation 28
2.29 State of a Membrane 29
2.30 Step-by-Step Tracing Reaction 30
2.31 On-going Reactions in a Membrane 30
2.32 LaTeX Documenting 31
2.33 Selecting Parts of Model for Generating LaTEX Codes 31
2.34 On-going Reactions in a Membrane 31

Chapter 1

Introduction

This documentation describes the use of Grid System Simulator. That simulator was initially a tool (also as a prototype) developed for writing the models and for verifying them during developing the author's PhD thesis. In order to enable our researches reported in the thesis to be repeatable by research community, the tool is made available to public, and in order to do that, this documentation is written for guiding in using the tool. Furthermore, the community can take advantage of the simulator to write and to simulate their the models of some formalisms that can be mapped to Grid Systems.

Because of the motivation in developing this tool as mentioned above, at the first place the early version of tool was intended primarily to provide functionalities instead of fancy features and user-friendliness like in commercial software. Even though, the tool works quite well and its simulation engine implement compute the semantics of Grid Systems precisely. Since then, some parts of the tool have been much improved, besides its remaining limitation, in order the tool to be made available to public.

We have planned to maintain continuously both its interfaces and its simulation engine to reach a reasonable level of the version. The tool should provide much graphical and interactive interfaces in order to ease the modellers in expressing the rule's behaviour. In this current version the engine implements the semantics in sequential manner instead of in parallel as the nature of the semantics. For the future version the engine should takes advantage of availability of parallel technology such GPGPUs, so that each membrane can be handled by one core.

The contents of the documentation are separated into several chapters. This introduction (Chapter 1) describes our intention in writing this documentation and summarizes some important terminologies related to Grid System modelling. Chapter 2 will describe general usages of the interfaces of the tool. The interfaces access functionalities from writing a new model from the scratch, defining the model's parts, defining data sets for several cases, running the simulation, setting its paramaters, tracing stepwise reactions, reporting utilities and maintaining the models.

In Chapter 3, a guideline for using the interfaces will be described through a step-by-step direction for defining the model parts. So that, the reader can be lead to follow it to understand how to define the parts through the interfaces. A brief description of the part will be supplied although understanding the concept of Grid Systems is a requirement in using the software.

Because of some limitation of the interfaces as explained above the modellers are recommended to know how the model will be formatted in its saved file. The content of model in the file will be structured using XML tags. Some necessary fixes that could not be easily made using the tool can be done by editing directly in the saved file while preserving the tagging syntax. In Chapter 4 the reference for working directly in the model's file will be elaborated. It will describe how the information of the model are formatted in the file as nested terms of XML tags, components and attributes. The use XML tagged file for models was initially due to shorten our developing time by taking advantage of the available packages provided by Java and W3C (`javax.xml.*`, `org.w3c.dom.*`). File content's readability helped to enable us in developing the engine earlier than the editing tool (the file was written using an ordinary text editor). The editing tool could be then developed in a function-by-function manner at a time as any function was needed.

As the last part of this documentation, an appendix will list the rules for above XML tagged file as a DTD.

1.1 Terminologies

The Grid Systems have 5 main components:

1. Grid name (G).
2. Set of object symbols (Σ).
3. Set of reaction rules (R).
4. Set of association of the rules and the membranes (A).
5. Initial configuration $C^{(0)}$.

The first component is just a unique identified to distinguish one another when they are discussed. However it is just the string when it is written in its file since currently a file can consist one model. Each object symbol ia a unique identifier representing the object that is involved in the model. A reaction rule describes a behaviour in the system involving objects of reaction (reactants, products, promoters, and inhibitors) and some other reaction parameters.

In Grid Systems the space where the system will evolve is divided as a grid of cells, that are also called local membranes. Each membrane is identified by its row-colum numbers in the grid started from [0,0] to [n-1, m-1], and both n and m are nonnegative integer numbers. Besides those local membranes, environmental entities will be placed in a global/environmental membrane that is identified by [E]. Formally, the dimension (n and m) of the grid is implicit. However, to ease the working of the engine in allocating the memory space, in this current version, the dimension should be supplied in the file.

An association is a relation between a rule r and a membrane M. It defines the rule r is applicable in M as long as the condition defined in the rule is satisfied. For a large grid dimension, defining such associations can be not quite practical. Therefore, above the definition of Grid Systems, we define a region as a set of local membranes. Association between a rule an a region will imply the associations between the rule and each membrane in that region.

Initial configuration defines initial state of the system in terms of the initial objects in each membranes. In general, a configuration defines a current objects in each membranes and the current on-going reactions (the reactions that have not completed yet). However, it is assumed that there is always no on-going reaction at t=0. The initial objects can be defined either explicitly in each membrane or implicitly in the region. Defining the initial objects in a region implies replication of the objects in each membrane of the region. When one membrane is included in more than one region, its initial objects are the addition from the objects defined in those regions.

Furthermore, the simulator will need two groups of running parameters (besides some others):

- Parameters for running the simulation.
  - Time limit. This information will stop the simulation when the time limit is exceeded.
  - Maximum number of evolution algorithm's iteration in simulation. Since some simulation may define zero-duration time of the reaction such that the time limit will never been reached, this parameter will limit the number of iterations.
  - Time Slice. As described in the thesis, time slice is the interval of time that all time points inside the interval will be considered the same. When time slice = 0, the interval is the time resolution of computer system clock. The small interval causes the simulation to gain more precision but costs in computation (space and time) cost.
- Parameters for observing the output.
  - Options for displaying the progress. By setting it true, the simulator will report the configuration periodically, and by setting its interval, the progress will be reported after each interval. Its default is not to report the progress, but only the final configuration.
  - The list of objects to be observed/reported.

The model and the parameters are encoded and bundled in one XML file.

1.2 Software's Organization

This section will describe in general how the software works with a model.

1.2.1 Data Structures

During its work the software maintains three different internal data structures:

- Internal XML structures: hierarchical structures reflecting the XML structures contained in its saved file.
- Raw Data Tables: the working structures for editing functionalities.
- Actual Data Tables: the work space for running the simulation.

Note that, the first data structures is resulted from the use the XML packages. Besides, in the future it is replaceable when another saved file format will be use to replace the XML tagged format. The raw tables are aimed to keep the parts as they are in the saved file, namely each rule is still written with its variables and constants. The actual tables keep the parts that are already expressed explicitly according to the definition of Grid System. The software has three modules:

- XML data handler module. It is a layer to access XML packages. It provides some methods for for loading the data tables from the internal XML structures, updating the structures after updating some parts of the model, besides, driving the XML package to load/save the internal XML structures from/to its saved file.
- Editor module. It contains editing interfaces and some verifier methods for verifying the input. In general each interface is specialized to handle editing one part of the model in its working data table in its own data space, then, when the interface is closed and the update is accepted, it will access the XML data handler for updating that part. It will then be followed by updating related tables. it will send the update to the XML input-output handler to update
- Simulation Module. It implements the Evolution Algorithm of Grid Systems. Other functionalities are added to control the running algorithm (stop, pause, resume, restart), as well as to inspect to current configuration data, and displaying the output as chart/pictures/tables.

1.3 Notes about Some Bugs

Graphical User Interfaces (GUIs) are implemented by using Swing (javax.swing.*) and AWT (java.awt.*) packages with the "Nimbus" look and feel. We should thanks for the goodies provided by the packages so that we did not need to develop those interface components by ourselves. However, we should always be warned that there are possibly some bugs from the packages. One of them that has been expectedly identified is in getting an item of a dropdown-menu (defined as a java.awt.Choice instance). Another one is in selecting a row of a table (defined as a javax.swing.jTable instance). To avoid having problem of this bug, a well-selected row entry is highlighted green. Instead of clicking mouse for selecting an item, Arrow keys can be used for moving the cursor toward the entry, followed by pressing Enter key for performing its selection.

The other bug that is being investigated is the instability of the system after running several instances of simulation. When the problem appears, a new simulation cannot be performed well. It is expectedly caused either from mistakenly handling the synchronization by multi-threaded processes or the delay of garbage-collection processes. However, it will happen only in instantiating a new simulation (not during a simulation run).

Chapter 2

User Interfaces

In this Part the user interfaces that provide the accesses to functionalities of the software will be described. The first section will describe how they are organized along with their respective functions. The second section will describe a step-by-step practice guideline in constructing in model by using the software. The third section will describe the interfaces and the ways how to control the simulation and setting-up the simulation output for observation.

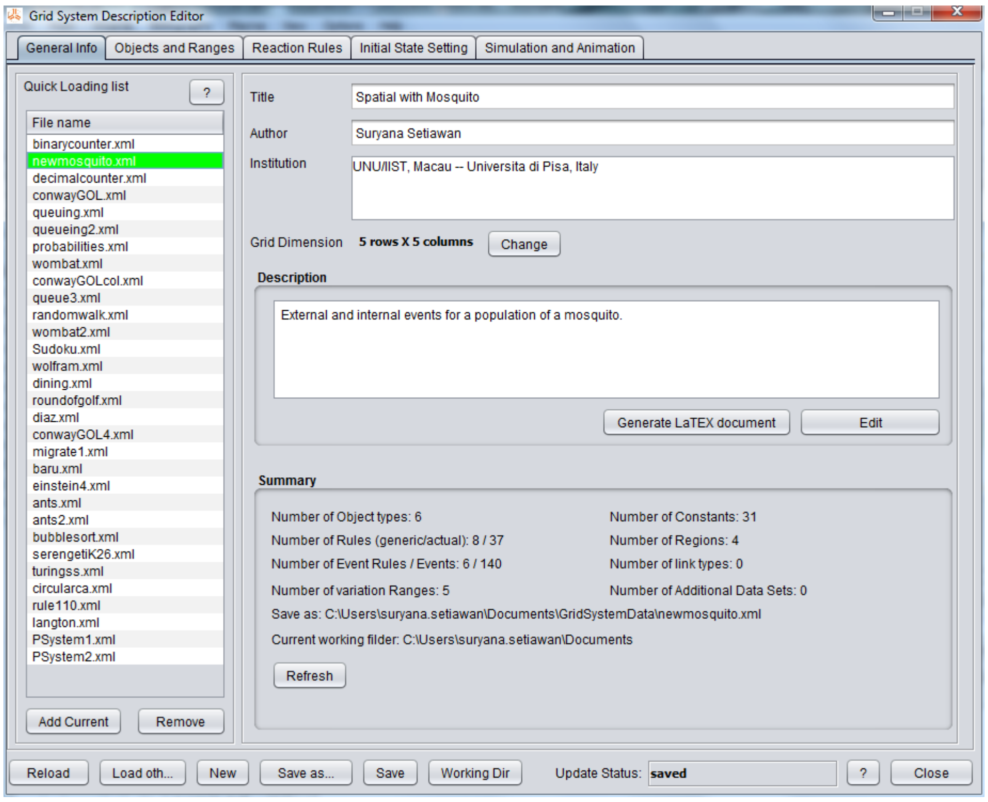
2.1 The Dashboard

The first appearance of the software is a dashboard containing some tabs at its top area, and some buttons at its bottom area as can be seen in Figure 2.1. The dashboard has five modes and it can be switch from one another mode by selecting its respective tab. The tabs/modes are "General", "Object and Ranges", "Trans. Rules", "Init. State", and "Simulation", respectively representing the groups of functionalities of the software that could be accessed.

2.1.1 "General" Mode

In this mode, the dashboard provides the interface for viewing general information of the being-edited model and some buttons to access the interfaces for modifying them. In this mode, the dashboard consists of two boxes: the center one that contains model's general information, and the left one that contains a list of names, as shown in Figure 2.1. Each name represents a model that the modeller has created. In the center box, there are some editable fields: the title, the author and the institution. A short description can be attached to the model and showing/editing the description can be performed by pressing "Show/Edit" button. After pressing it, a dialog box for reading/editing the description will be popped up. An additional button "Generate LaTeX..." is provided for generating LaTeX codes of the model components such that they can be copied and inserted in a LaTeX document. In the LaTeX codes the information (title, author, institution and description) could be inserted as well. The right LaTeX codes embedded into the description field will make the description well-formatted either.

The text fields containing numbers after Grid Dimension are not editable. Instead, by pressing "Change" button a dialog box for editing the dimension will be popped up.



File name
binarycounter.xml
newmosquito.xml
decimalcounter.xml
conwayGOL.xml
queuing.xml
queueing2.xml
probabilities.xml
wombat.xml
conwayGOLcol.xml
queue3.xml
randomwalk.xml
wombat2.xml
Sudoku.xml
wolfram.xml
dining.xml
roundofgolf.xml
diaz.xml
conwayGOL4.xml
migrate1.xml
baru.xml
einstein4.xml
ants.xml
ants2.xml
bubblesort.xml
serengetiK26.xml
turingss.xml
circularca.xml
rule110.xml
langton.xml
PSystem1.xml
PSystem2.xml

Title: Spatial with Mosquito
Author: Suryana Setiawan
Institution: UNU/IIST, Macau -- Universita di Pisa, Italy
Grid Dimension: **5 rows X 5 columns**

Description
External and internal events for a population of a mosquito.

Summary
Number of Object types: 6
Number of Constants: 31
Number of Rules (generic/actual): 8 / 37
Number of Regions: 4
Number of Event Rules / Events: 6 / 140
Number of link types: 0
Number of variation Ranges: 5
Number of Additional Data Sets: 0
Save as: C:\Users\suryana.setiawan\Documents\GridSystemData\newmosquito.xml
Current working filder: C:\Users\suryana.setiawan\Documents

Update Status: **saved**

Figure 2.1: The Dashboard: "General" Mode

"Summary" area is intended for showing the profile of the model. However, currently the area shows only a part of the profile that includes the current numbers of the model's components: objects, rules (a template as one rule), actual rules, external events, variations, constants, initial datasets, and regions. In the future, off course, this part can further be made to display more important information. In the left area the modeller can place the file name of being-edited model. The names in list act as short-cuts for accessing and loading their respective files quickly.

"New" button is for clearing the dashboard so that a new model can be written. When there is a model being edited, pressing the button will pop up a dialog box for saving being-edited model. "Reload" button is for restoring the saved version of the being-edited model when the changes have been made will be discarded. "Load other" button is for loading a saved model from its file and is used when the model is not listed in "Your Collection". Pressing this button will pop up a dialog box for selecting the file. "Save as" button is for saving the currently edited model in other file and "Save" button is for saving the currently edited model in its original file. "Work Dir" button is for changing the current folder so that saving/loading a file can be easily performed to/from that folder.

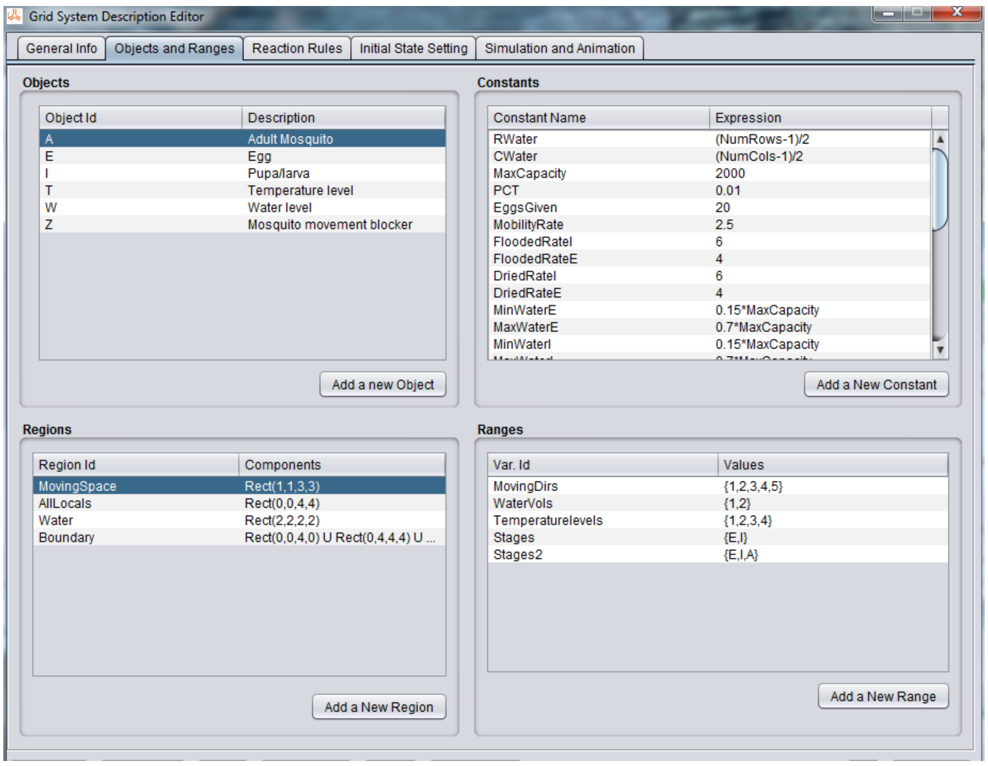
2.1.2 "Object n Ranges" Mode

In this mode the dashboard shows the entities currently defined in the model:

- the objects,

- the constants,
- the regions, and
- the variations,

as can be seen in Figure 2.2. The top-left area lists the objects that are already defined for



Objects

Object Id	Description
A	Adult Mosquito
E	Egg
I	Pupa/larva
T	Temperature level
W	Water level
Z	Mosquito movement blocker

Constants

Constant Name	Expression
RWater	(NumRows-1)/2
CWater	(NumCols-1)/2
MaxCapacity	2000
PCT	0.01
EggsGiven	20
MobilityRate	2.5
FloodedRateI	6
FloodedRateE	4
DriedRateI	6
DriedRateE	4
MinWaterE	0.15*MaxCapacity
MaxWaterE	0.7*MaxCapacity
MinWaterI	0.15*MaxCapacity

Regions

Region Id	Components
MovingSpace	Rect(1,1,3,3)
AllLocals	Rect(0,0,4,4)
Water	Rect(2,2,2,2)
Boundary	Rect(0,0,4,0) U Rect(0,4,4,4) U ...

Ranges

Var. Id	Values
MovingDirs	{1,2,3,4,5}
WaterVols	{1,2}
Temperaturelevels	{1,2,3,4}
Stages	{E,I}
Stages2	{E,I,A}

Figure 2.2: The Dashboard: "Objects n Ranges" Mode

the model. "Add a new Object" button is for popping up a dialog box for defining a new object. Double clicking over entry of the list will pop up a dialog box for editing/deleting its object definition. Of course, when this object is currently used somewhere in the model the object cannot be deleted. The top-right area lists the constants that are already defined for the model.

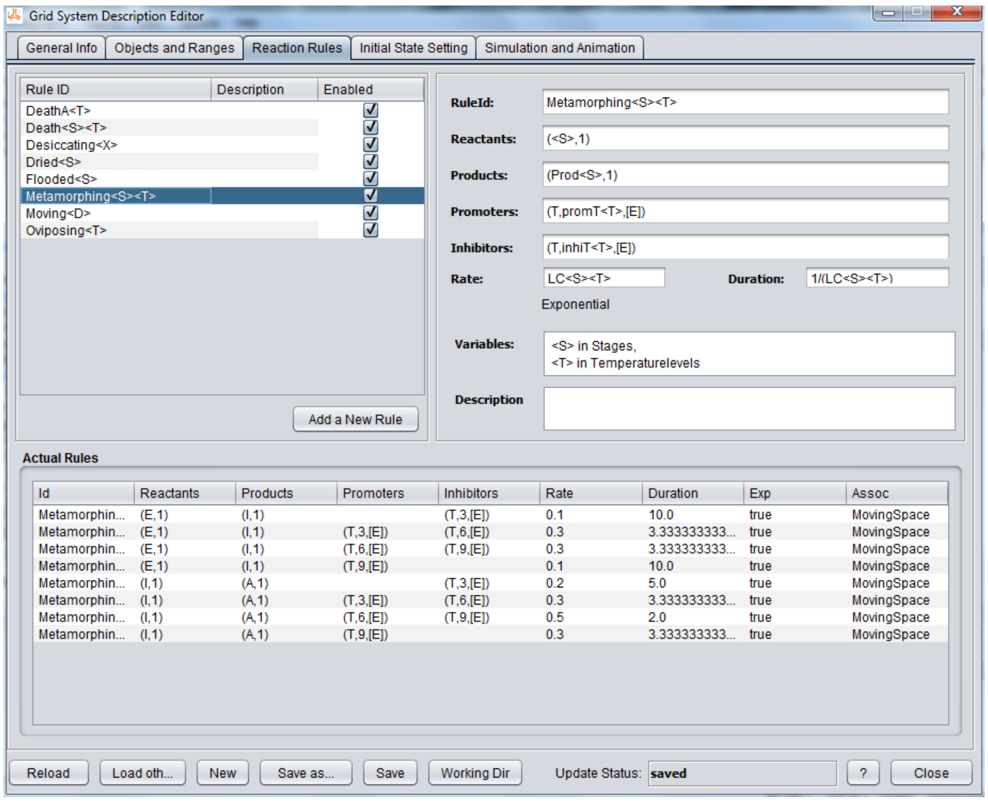
Pressing "Add a New Constant" will pop up a dialog box for defining a constant or a constant list. By the dialog box, the constant can be defined either textual or numeric. Deleting/updating a constant/constant list can be performed by double clicking over it in the list. As for objects, deletion can be performed only when the constant is not being used in any component of the model definition.

The bottom-left area lists the regions that are already defined for the model. Pressing "Add a New Region" will pop up a dialog box for defining a new region. Then, the dialog box requires defining the region interactively composed from rectangles and cells and other already-defined regions. Deleting/updating can be performed by double clicking over an entry of the list. The bottom-right area lists the variation ranges that are already defined for the model.

Pressing “Add a New Range” will pop up a dialog box for defining a new variation range. Then, the dialog box will require defining the variation id and its range. The range is defined as several strings separated by commas. Deleting/updating can be performed by double clicking over an entry of the list.

2.1.3 “Trans. Rules” Mode

In this mode the dashboard provides the interfaces for defining the transition rules as can be seen in Figure 2.3. In the top-left area all already defined rules are listed. One of them



Rule ID	Description	Enabled
DeathA<T>		✓
Death<S><T>		✓
Desiccating<X>		✓
Dried<S>		✓
Flooded<S>		✓
Metamorphing<S><T>		✓
Moving<D>		✓
Oviposing<T>		✓

RuleId:	Metamorphing<S><T>
Reactants:	(<S>,1)
Products:	(Prod<S>,1)
Promoters:	(T,promT<T>,[E])
Inhibitors:	(T,inhiT<T>,[E])
Rate:	LC<S><T>
Duration:	1/(LC<S><T>)
	Exponential
Variables:	<S> in Stages, <T> in Temperaturelevels
Description	

Actual Rules

Id	Reactants	Products	Promoters	Inhibitors	Rate	Duration	Exp	Assoc
Metamorphin...	(E,1)	(I,1)		(T,3,[E])	0.1	10.0	true	MovingSpace
Metamorphin...	(E,1)	(I,1)	(T,3,[E])	(T,6,[E])	0.3	3.333333333...	true	MovingSpace
Metamorphin...	(E,1)	(I,1)	(T,6,[E])	(T,9,[E])	0.3	3.333333333...	true	MovingSpace
Metamorphin...	(E,1)	(I,1)	(T,9,[E])		0.1	10.0	true	MovingSpace
Metamorphin...	(I,1)	(A,1)		(T,3,[E])	0.2	5.0	true	MovingSpace
Metamorphin...	(I,1)	(A,1)	(T,3,[E])	(T,6,[E])	0.3	3.333333333...	true	MovingSpace
Metamorphin...	(I,1)	(A,1)	(T,6,[E])	(T,9,[E])	0.5	2.0	true	MovingSpace
Metamorphin...	(I,1)	(A,1)	(T,9,[E])		0.3	3.333333333...	true	MovingSpace

Update Status: saved

Figure 2.3: The Dashboard: “Trans. Rules” Mode

is highlighted, and in the top-right area the detail of highlighted rule is shown. Another rule can be highlighted by clicking over its entry in the list will. If the rule is a template the bottom area will list all actual rules that can be generated from this template. Therefore, the modeller can preview all actual rules of the highlighted template rule. A new rule can be created by Pressing “Add a New Rule” button in the top-left area and its detail can be defined in the dialog box that will be popped up after Pressing the button. Double clicking over an entry (rule) in the list in top-left area will pop up a dialog box for re-editing the rule as well as for other purposes to the rule: deleting, enabling/disabling.

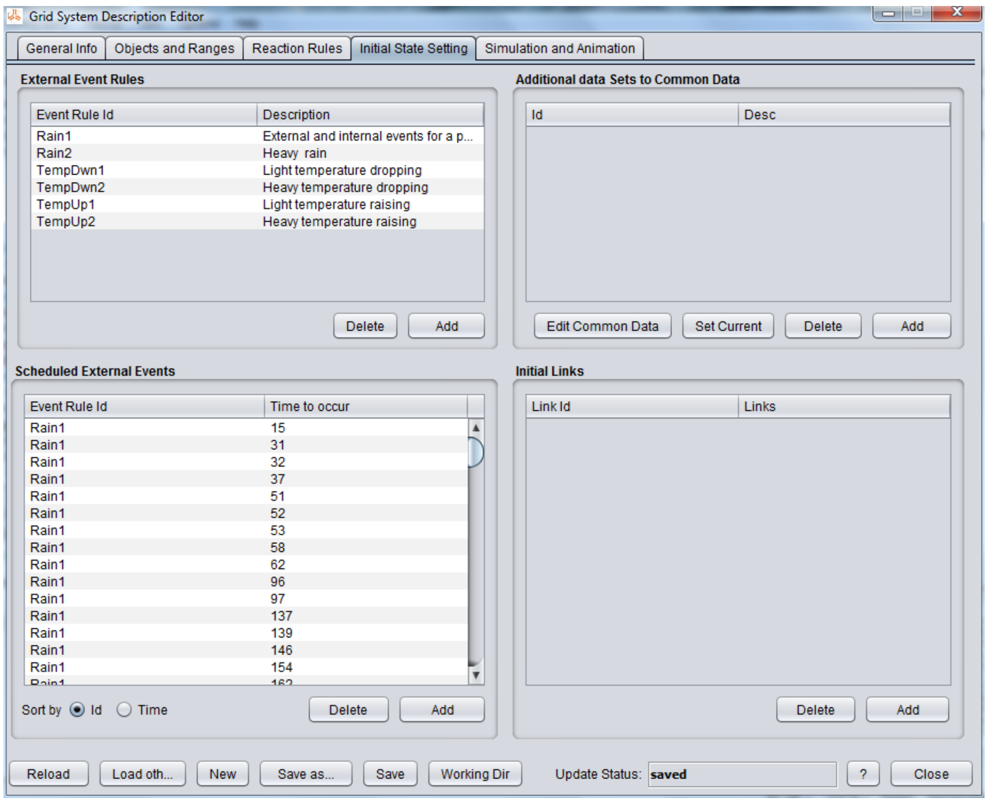
2.1.4 “Init. State” Mode

In this mode the dashboard provides five areas for relating aspects:

- list of datasets for different initial objects,

- links,
- external events, and
- rules for events,

as can be seen in Figure 2.4.



External Event Rules

Event Rule Id	Description
Rain1	External and internal events for a p...
Rain2	Heavy rain
TempDwn1	Light temperature dropping
TempDwn2	Heavy temperature dropping
TempUp1	Light temperature raising
TempUp2	Heavy temperature raising

Additional data Sets to Common Data

Id	Desc

Scheduled External Events

Event Rule Id	Time to occur
Rain1	15
Rain1	31
Rain1	32
Rain1	37
Rain1	51
Rain1	52
Rain1	53
Rain1	58
Rain1	62
Rain1	96
Rain1	97
Rain1	137
Rain1	139
Rain1	146
Rain1	154

Initial Links

Link Id	Links

Update Status: saved

Figure 2.4: The Dashboard: "Init. State" Mode

Scenarios are intended for defining different sets of intial objects so that its simulation can be conducted to the sets respectively for different observations. Currently, the interfaces for defining scenarios and the events are not developed well.

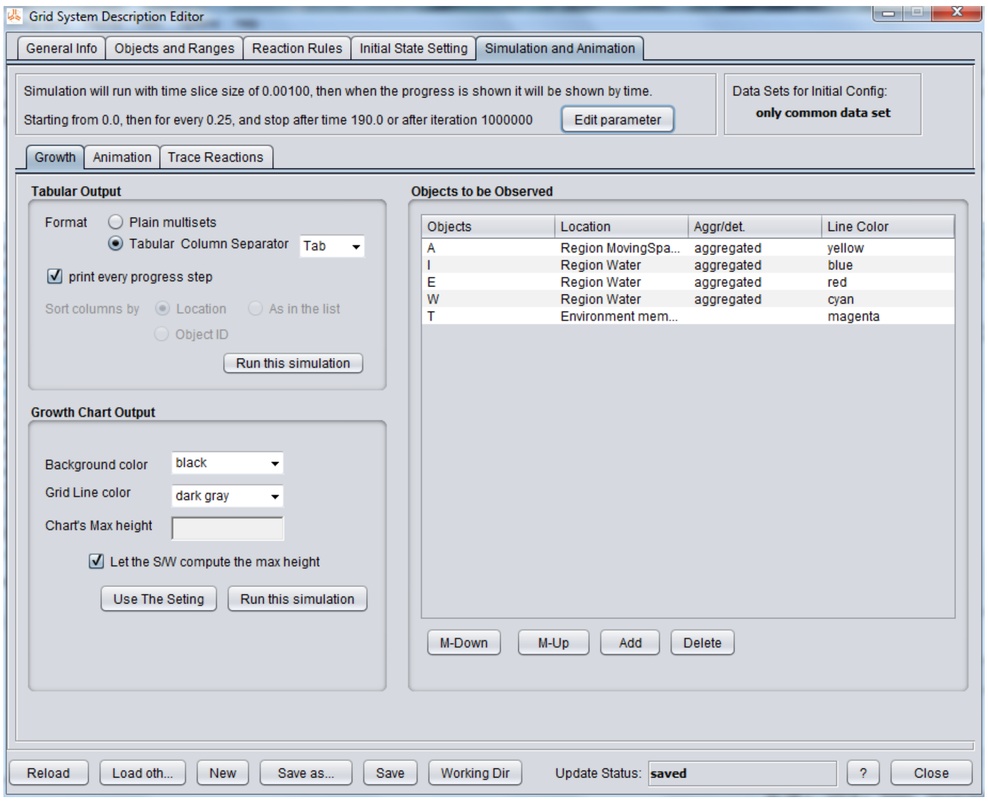
In the top-right box area already defined initial objects are listed. Each entry of the list specifies the location of initial objects that could be either a region or a local membrane or the environmental membrane. Pressing "Add" button in this box will pop up a dialog box for defining new initial objects in certain location as above. Double-clicking over an entry in the list will pop up a dialog box for updating/deleting the initial objects in that location.

In the model the links are grouped as link sets which are the names for the links. The Bottom-left area lists the link sets already defined in the model. Pressing "Add" button will pop up a dialog box for defining a new link set. The dialog box provides the interfaces for defining initial links of being-defined link set over the membranes in the grid. Double-clicking over an entry in the list will pop up the dialog box for editing/deleting the link set including the links already defined for the link set.

In the center there are two areas: "external event rules" (top-center) and "scheduled external events" (bottom-center). The former one lists the special rules for defining external events and the latter one list their scheduled events. By pressing "Add" button in the top-center area, a dialog box for defining an external event rule will be popped up. Editing one rule that is already in the list can be performed by double-clicking over its entry and the definition can be edited through its dialog box. By pressing "Add" button in the bottom-center area, a dialog box for creating the events will be popped up. Editing one event that is already scheduled in the list can be performed by double-clicking over its entry and a dialog box for editing it will be popped up.

2.1.5 "Simulation" Mode

In the simulation mode, the dashboard provides the accesses for software's functionalities in conducting the simulation as can be seen in Figure 2.5. There are three sub-modes



Simulation will run with time slice size of 0.00100, then when the progress is shown it will be shown by time.
Starting from 0.0, then for every 0.25, and stop after time 190.0 or after iteration 1000000

Data Sets for Initial Config:
only common data set

Format: Plain multisets [✓ Tabular] Column Separator: Tab
✓ print every progress step
Sort columns by [✓ Location] As in the list Object ID

Background color: black
Grid Line color: dark gray
Chart's Max height:
✓ Let the S/W compute the max height

Objects	Location	Aggr/det.	Line Color
A	Region MovingSpa...	aggregated	yellow
I	Region Water	aggregated	blue
E	Region Water	aggregated	red
W	Region Water	aggregated	cyan
T	Environment mem...		magenta

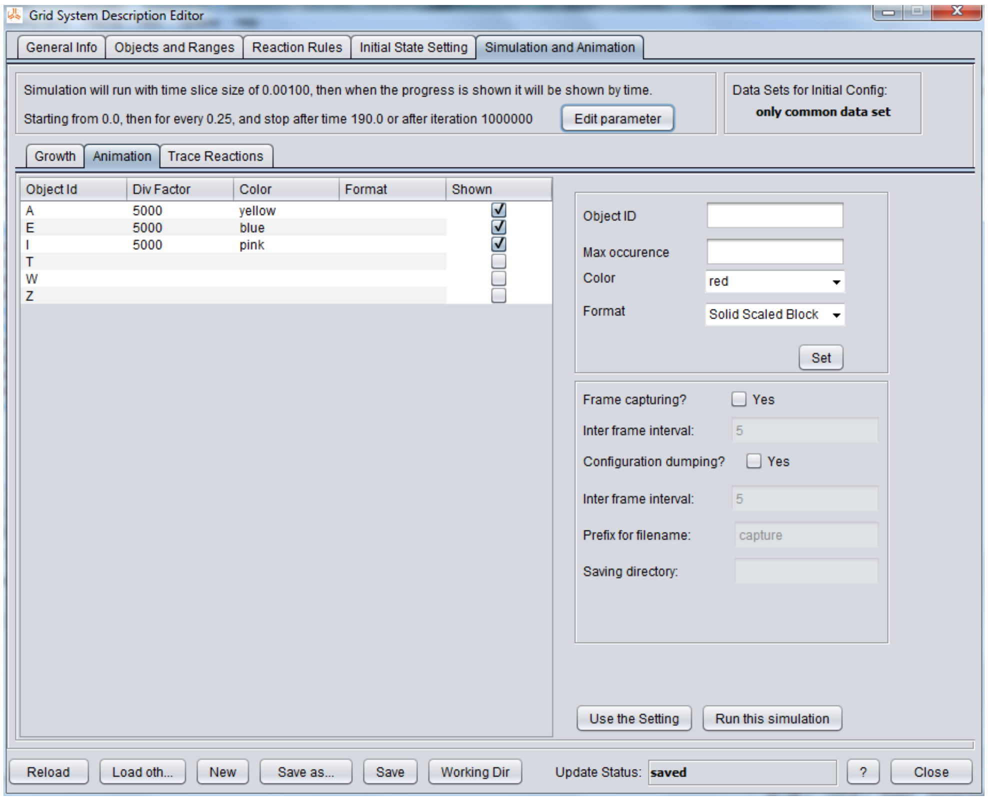
Update Status: **saved**

Figure 2.5: The Dashboard: "Simulation" Mode with "Growth" sub-mode

provided by the dashboard:

- "Growth "sub-mode, for observing the timely chart of the Growth functions resulted by the simulation,
- "Animation" sub-mode, for observing spatial dynamics of the model, and
- "Trace Reaction" sub-mode, providing an environment for "debugging the behaviour by a step-by-step simulation running.

Figure 2.6 and Figure 2.7 show the dashboard for the second and third submodes, respectively.



Simulation will run with time slice size of 0.00100, then when the progress is shown it will be shown by time.

Starting from 0.0, then for every 0.25, and stop after time 190.0 or after iteration 1000000

Data Sets for Initial Config: **only common data set**

Object Id	Div Factor	Color	Format	Shown
A	5000	yellow		✓
E	5000	blue		✓
I	5000	pink		✓
T				☐
W				☐
Z				☐

Object ID:

Max occurence:

Color: red

Format: Solid Scaled Block

Frame capturing? ☐ Yes

Inter frame interval: 5

Configuration dumping? ☐ Yes

Inter frame interval: 5

Prefix for filename: capture

Saving directory:

Update Status: **saved**

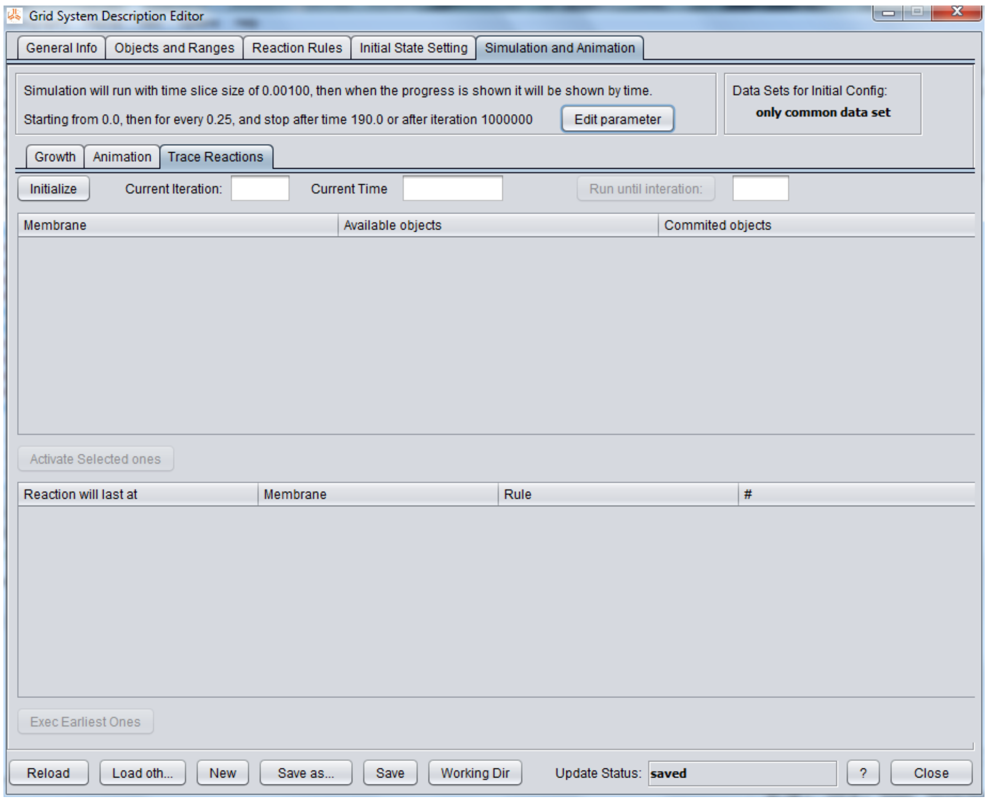
Figure 2.6: “Simulation” Mode with “Animation” sub-mode

2.2 A Practical Guideline

In this section a step-by-step practical guideline will be elaborated. It will start from an empty model as the first appearance when the software is started. It will subsequently followed by necessary actions in defining a model. Note that, before defining the model by the interfaces it will be better to have the model drafted well by hand. From the draft all major objects and regions are already identified so that the work will not be back and forth between defining the rules and defining the objects/regions.

2.2.1 Defining Grid’s Dimension

The first necessary step is defining the dimension of the grid. The dialog box for this will be shown after Pressing the “change” button as seen on Figure 2.8. The numbers for the rows and the columns of the grid should be specified as non-negative integers. Of course, once it has been defined, it can be modified later. However, after defining some rules and setting rule association to the membranes, modifying the dimension will be rejected if the new one excludes the membranes that are already associated with a rule. If such modification is neccessary, those regions/associations should be removed first. For



Simulation will run with time slice size of 0.00100, then when the progress is shown it will be shown by time.
Starting from 0.0, then for every 0.25, and stop after time 190.0 or after iteration 1000000

Data Sets for Initial Config:
only common data set

Membrane	Available objects	Commited objects

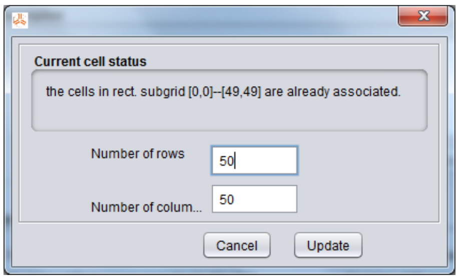
Reaction will last at	Membrane	Rule	#

Update Status: **saved**

Figure 2.7: "Simulation" Mode with "Trace Reactions" sub-mode

guiding this modification, the software provides the information of the largest row/column numbers that already associated to rules.

Figure 2.8 is the dialog box for modifying the dimension. It shows the the status that the membranes of subgrid [0,0]-[49,49] are already associated so that the dimension cannot be changed to the size less than 50 rows $\times$ 50 columns. The dimension can be changed to the lager size, instead.



Current cell status

the cells in rect. subgrid [0,0]--[49,49] are already associated.

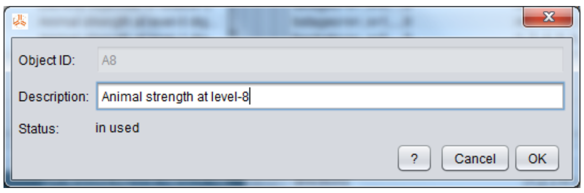
Number of rows: 50
Number of colum...: 50

Figure 2.8: Defining the Dimension

2.2.2 Defining Objects

The next step is to define the objects in the list. As described previously, a dialog box will be popped up when "Add New Object" button is pressed. The dialog box requires the modeller to enter two fields: the object identitty (ID) and the object description. The ID should be unique (the software will reject it if it is already defined/used in the model) and it is an alphanumeric string except the first character is a letter. The dialog box for editing already-defined object has the same appearance, except that the ID cannot be changed. Also, already used objects cannot be removed as well as its ID cannot be changed.

Figure 2.9 shows the dialog box for editing already-defined object which ID is "A1" and description is "Animal strength at level-1". A text line shows an information that this object is currently in use in some rules so that it cannot either be removed or be renamed.



Object ID: A8
Description: Animal strength at level-8
Status: in used

Figure 2.9: Defining an Object

2.2.3 Defining Regions

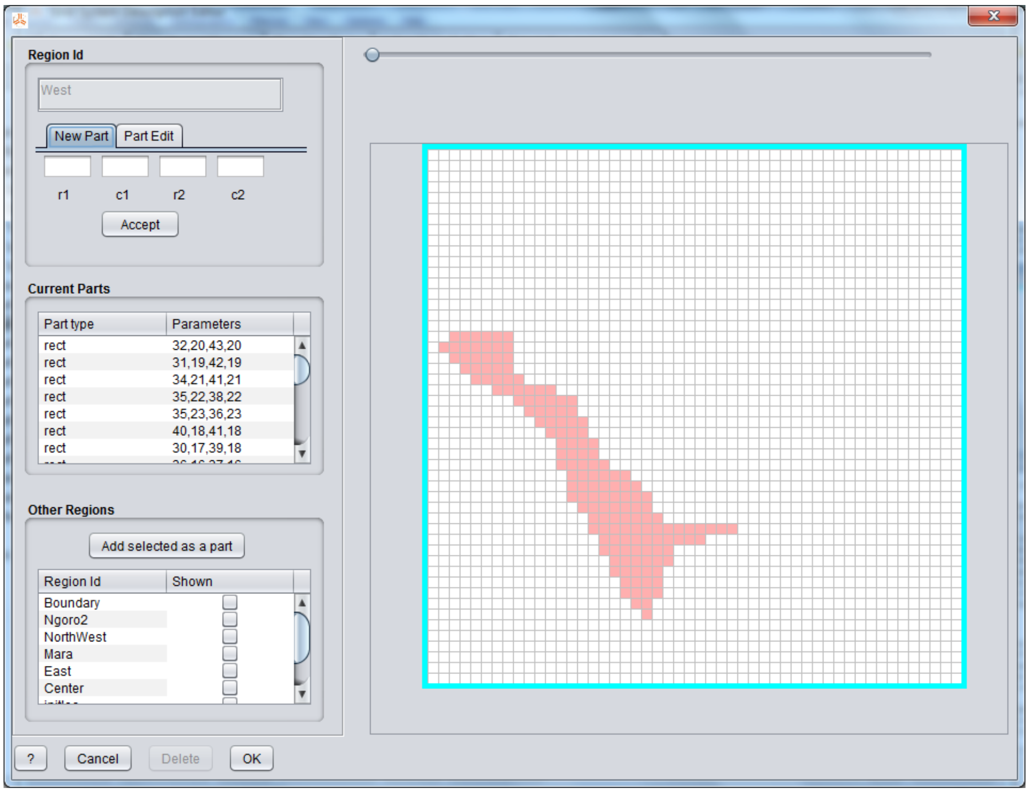
A region is a set of local membranes. Then, a region can be defined as a union (the union operation of sets) of any simpler regions besides of by its individual membranes. A rectangle is a rectangular region specified by 4-tuple $(r1, c1, r2, c2)$ defining rectangle's top-left $(r1, c1)$ and bottom-right $(r2, c2)$ membranes. All inclusive membranes of a rectangle $(r1, c1, r2, c2)$ are membranes (r, c) that $r1 \leq r \leq r2$ and $c1 \leq c \leq c2$.

The interface provides composing those parts to be a region by "drawing" them over a "canvas" representing the grid with the dimension as specified previously. The interface works in two modes: "Edit Mode" and "Select Mode". The former one is for defining and adding a new part into the current region. The latter is for modeller to select an already defined part for deletion. The modes are exclusive one another so clicking (enabling) the button "Edit Mode" will disable "Select Mode" and vice versa (by clicking "Select Mode" will disable "Edit Mode").

In "Edit Mode" a local membrane drawn as a small square in the canvas can be selected by double-clicking over a position, or alternatively, by writing directly the row and column numbers in respectively "r1" and "c1" text fields (inside "You are defining a new part (inclusive)" area). A rectangle is defined by pressing the mouse button at the position that will be one corner of the rectangle, and dragging the mouse to the opposite corner of the rectangle and then release the mouse button, or alternatively, by writing directly those positions in "r1", "c1", "r2", and "c2" fields as for membrane above. After either way the part will be recorded by pressing "Accept" button. When it is intentionally, or incidentally, written than r1=r2 and c1=c2, the software will assume this part is a membrane. Deletion a part can be performed after the interface is already in "Select mode". The part in the list should be selected by clicking over it (either in the canvas or in the list) and then pressing "Delete" button.

Figure 2.10 shows constructing a new region, while its ID is not typed yet in the text field at upper left corner. Of course, when it will be saved the software will require this

ID being given. The interface is in "Edit Mode" so manipulation on the canvas will create a part (either a membrane or a rectangle). There are two parts that have been created and accepted: membrane [5,5] and rectangle (21,7,24,13).



Part type	Parameters
rect	32,20,43,20
rect	31,19,42,19
rect	34,21,41,21
rect	35,22,38,22
rect	35,23,36,23
rect	40,18,41,18
rect	30,17,39,18

Region Id	Shown
Boundary	☐
Ngoro2	☐
NorthWest	☐
Mara	☐
East	☐
Center	☐

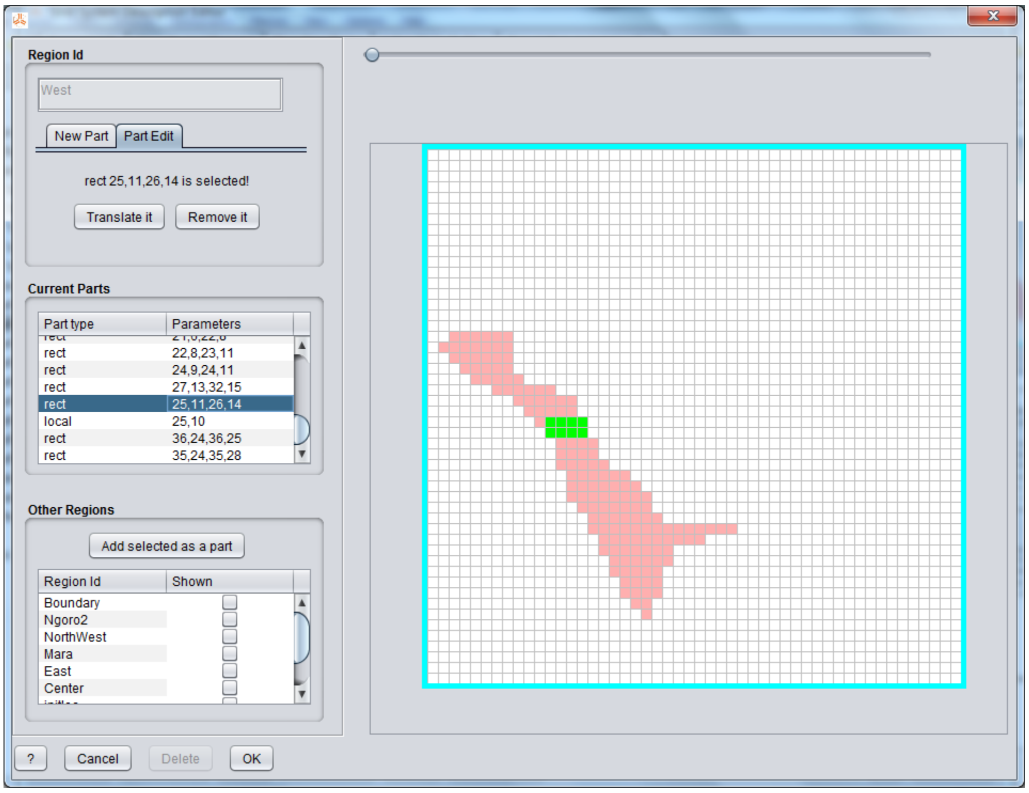
Figure 2.10: Defining a Region ("Edit Mode")

Figure 2.11 shows editing already developed region and the interface is in "Select Mode". So, clicking on a part in the canvas (or on an entry in the list) will select the part for an action, namely, deletion.

The feature for including already defined region is not implemented yet in the current version. This feature should enable to include the region by selecting it from the list in the bottom-left area. The software should detect a cyclical inclusion of regions (including a region that ultimately includes the current edited region) and then reject it. In the current version some regions can be shown overlaying with the currently edited region to guide in defining the region. Inclusion of other region can only be added by editing model's XML file manually since the simulation engine is able to recognize this.

2.2.4 Defining Constants

A constant is an identity (ID) having a value that can replace any occurrences of string that is the same with this ID in any value fields of later-defined constants or of rule definitons, by this constant's value. A numeric constant's value can be defined as an arithmetic expression of some literal values or values from previously-defined numeric constants. A textual constant's value is defined as a string or a substitutable string by other previously-defined textual constant.



Region Id

West

rect 25,11,26,14 is selected!

Current Parts

Part type	Parameters
rect	22,8,23,11
rect	24,9,24,11
rect	27,13,32,15
rect	25,11,26,14
local	25,10
rect	36,24,36,25
rect	35,24,35,28

Other Regions

Region Id	Shown
Boundary	☐
Ngoro2	☐
NorthWest	☐
Mara	☐
East	☐
Center	☐

Figure 2.11: Defining a Region ("Select Mode")

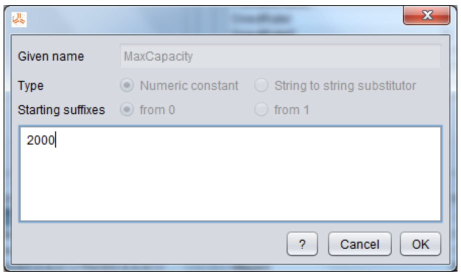
A constant list is indexed constant values. The indices of a constant list is defined by subsequent integers started either from 0 or from 1. A value can be referred by its ID and an index number. Just like a constant, a constant list can be numeric or textual.

The dialog box requires defining constant ID, its type (numeric or textual) and its value (values). If it is defined as one value it will be accepted as one constant (not a constant list). If it is defined with values (separated by commas, therefore, a comma is reserved symbol for this purpose) it will be accepted as a constant list. When it is a constant list, its first index should be defined either 0 or 1 (Note that, in the future version the first index could be any non-negative number as well as the indices could be a letter). The software will reject the definition when:

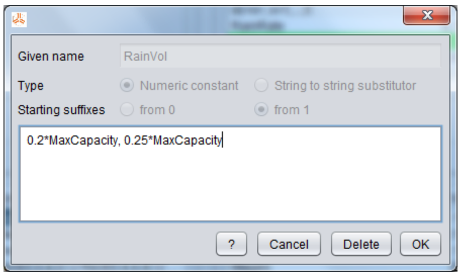
- it is an indexed constant list's ID that is already used as another constant. Eg., 'deathrate5' is already defined, then a constant list 'deathrate' indexed by 0,1,.., 9 will be rejected.
- a constant's ID that is already used as an indexed constant list ID. Eg., 'deathrate5' is rejected, when deathrate' indexed by 0,1,.., 9 has been defined.
- an indexed constant list's ID that is already used as another indexed constant list ID. Eg., a constantlist 'Ax1' indexed by 0,1,2 is rejected, when 'Ax' having indices 0,1,.., 15 has been defined.

By selecting "Numeric constant", the values are assumed as numbers. Otherwise, the values are assumed as strings. The software will reject when it is defined numeric but its values are not numbers, except if values are defined by a simple arithmetic expression (which can contain other constant ids). The dialog box for modifying the constant is just the same except the ID field could be not editable when it is already used somewhere in the model. The values of a constant list can be made shorter list than before as long as removed values are not used somewhere in the model yet.

Figure 2.2.4 shows a numerical constant list that is being modified containing 10 values and indexed from 0 to 9.


Given name: MaxCapacity
Type: (●) Numeric constant ( ) String to string substitutor
Starting suffixes: (●) from 0 ( ) from 1
2000

(a) A Constant (Single value)


Given name: RainVol
Type: (●) Numeric constant ( ) String to string substitutor
Starting suffixes: ( ) from 0 (●) from 1
0.2*MaxCapacity, 0.25*MaxCapacity

(b) A Constant List (set of values)

Figure 2.12: Defining a Constant

2.2.5 Defining Variation Ranges

The variation range is useful for defining the template of rules where some rules are implicitly defined as one template varied by one or more ranges. The different value/string for the rules can be still expressed in the template by the help of constant lists. After writing the id of the variation, the strings are defined by a set of string separated by commas (as for a constant list). Editing an existing variation will popped up similar dialog box except the id cannot be edited. Deleting an existing variation range can only be performed when its is not in used by the rules. Figure 2.13 shows a variation that has id 'declev' and ranges are 1,2,..., 10.

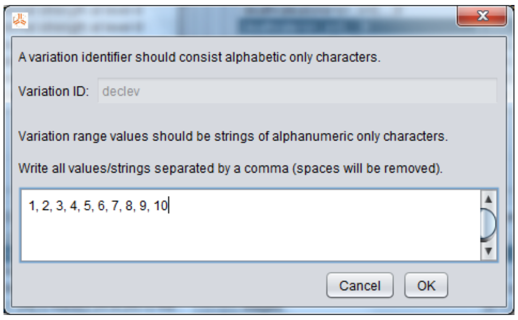
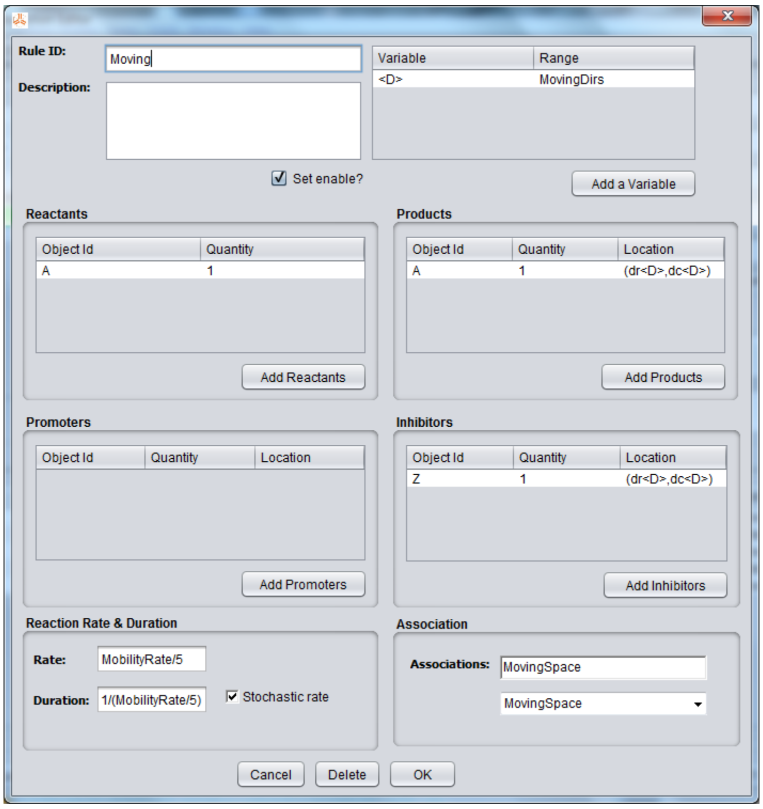

A variation identifier should consist alphabetic only characters.
Variation ID: declev
Variation range values should be strings of alphanumeric only characters.
Write all values/strings separated by a comma (spaces will be removed).
1, 2, 3, 4, 5, 6, 7, 8, 9, 10

Figure 2.13: Defining a Variation Range

2.2.6 Defining Rules

The dialog box contains some areas for defining components of a rule as shown in Figulre 2.14. The first thing to do for defining a rule is typing rule's identity (ID) in its field at the top-left corner of the dialog box. It becomes important to be defined earlier when the rule will be a template by appending one or more variables after the ID.



Rule ID: Moving

Description:

Variable	Range
<D>	MovingDirs

[✓] Set enable?

Reactants

Object Id	Quantity
A	1

Products

Object Id	Quantity	Location
A	1	(dr<D>,dc<D>)

Promoters

Object Id	Quantity	Location

Inhibitors

Object Id	Quantity	Location
Z	1	(dr<D>,dc<D>)

Reaction Rate & Duration

Rate: MobilityRate/5

Duration: 1/(MobilityRate/5) [✓] Stochastic rate

Association

Associations: MovingSpace

MovingSpace

Figure 2.14: Defining a Rule

The next thing to do is to append variables when the rule will be defined as a template. A variable is defined by a variation range. When it is appended to the rule's ID the variable functions as a stub to vary actual rule's IDs; each value in its range will replace this variable when the rule is interpreted becoming actual rules. The upper-left area of the Figure 2.14 contains a list for the variables already defined for this rule. In the figure, the rule is defined by ID 'MoveToGrass6' and it is appended by variables $\langle X \rangle$ and $\langle S \rangle$. Note that $\langle X \rangle$ is 'direction' (defined as range: {1,..., 8}) and $\langle S \rangle$ is 'stages' (defined as range {0,1,...,9}). Editing a variable that is already in list can be performed by double clicking over its entry in the list.

Button '+' below the variables is for appending another variable. By clicking it the dialog box will be popped up as shown in Figure 2.15. A letter is required by defining a variable and its variation range should be selected from a drop-down list of variation ranges. The variable letters for a rule should be distinct one another. Editing a variable that is already in the list can be performed by double clicking over its entry in the list.

Then, of course, rule's objects (reactants, products, promoters and inhibitors) are the next things to define. The middle-left area (labelled "Reactants") contains a list of already defined reactants. The figure shows a reactant that its ID ('A') is appended by variable $\langle S \rangle$. $\langle S \rangle$ is defined as 'stages' (range: {0,1,...,9 }) meaning that the reactants will be altered from 'A0, 'A1, ..., 'A9'.

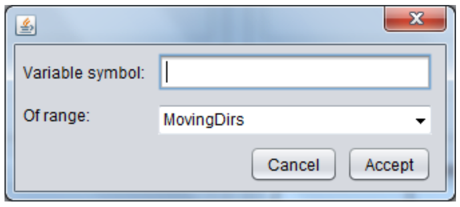


Figure 2.15: Rule's Variable

By clicking button "+" below the list, the dialog box for defining a new reactant will be popped up as shown in Figure 2.16. The dialog box will require an id that can be taken from a drop-down list, or by typing it especially when it will contain variable(s). When the number of objects is greater than one, this number should be typed in its "Quantity field".

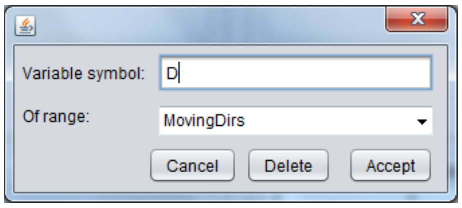


Figure 2.16: Rule's reactants

The areas labelled "Products", "Promoters", and "Inhibitors" contain lists of rule's products, promoters and inhibitors respectively. By clicking button "+" the dialog box for defining a new product/promoter/inhibitor will be popped up as shown in Figure 2.17. As for reactants the objects can be taken either from the drop-down list or by typing it. Especially when it contains variable(s) it should be typed. Also, the quantity of objects should be specified if the number is greater than one. Additional information can be supplied specifying the location of the object.

- For relative addressing (dr, dc) (the objects are at a relative (dr, dc) distance from the membrane where the rule will be applied), it should be typed as '(dr, dc)' (without quotes).
- For absolute addressing $[r, c]$ (the objects are exactly at membrane $[r, c]$), it should be typed as '[r, c]' (without quotes).
- For environment objects, it should be typed as '[E]' (without quotes).
- For linked addressing pointed by link *there*, it should be typed as 'L(there)' (without quotes).

As for object's id, the information can be appended by variables. Figure 2.17 shows Object 'G' that is located by a relative addressing. The row/column values of addressing are expressed as a constant id dr and dc by varied their id by appended variable $\langle X\rangle$ (range: {1,..., 8}). Therefore, it will take for row/column values in the addressing indexed values dr1, ..., dr8, and dc1,..., dc8, defined by constant lists dr and dc. In this Figure the object's

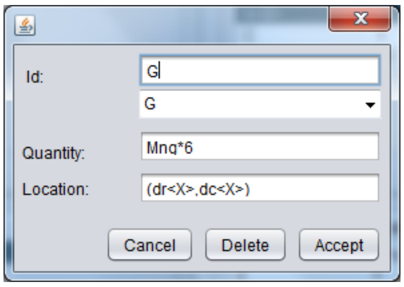

Id: G
G
Quantity: Mnq*6
Location: (dr<X>,dc<X>)

Figure 2.17: Rule's Products (or Promoters or Inhibitors)

quantity is defined by arithmetic expression 'Mng*6' which later will be interpreted to be a number when its actual rules is generated for the simulation.

The area labelled "Reaction Rate & Duration" is for defining reaction rate, duration, and the mode of duration. Reaction rate's default value is 1. When reaction rate is c, duration's default value is $1/c$. When stochastic rate is selected, the rule has exponentially distributed duration time with its mean is $1/c$. Otherwise, the rule has exact time duration $1/c$.

The area labelled "Association" is for specifying rules' associations. It shows a drop-down list containing regions that can be selected for association. The software limits the association to a single region only. Therefore, for more associations that already defined in the region, the modeller should define as another region.

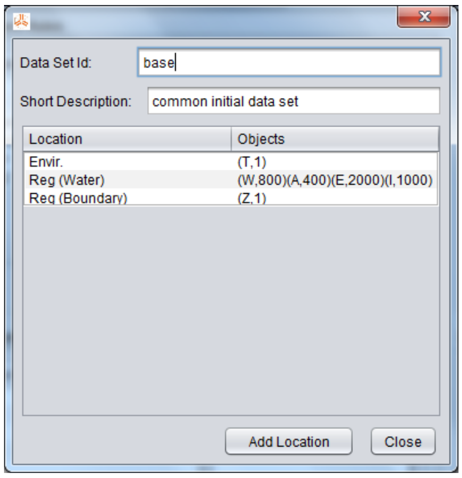
Description of the rule can be necessarily added for documentation of the model. Some special notes about the detail of the definition can be added, too. State of "Set Enable?" indicates whether the rule will be used or not for the moment. This state will be useful when an observation of certain rule(s) will be taken place by making off all other rules' states.

2.2.7 Defining Initial Objects

The initial objects are the ones that will exist in the model at $t = 0$. According to Grid System's definition the objects should be defined as multisets over objects in each membrane. For more practical work, defining by such a way may be exhaustive. Therefore, the software provides a way for defining them through regions as well directly to the membrane. When a multiset is defined for a region, each membrane of this region will copy the multiset into it. Each membrane that belongs to more than one region will have all multisets from every region. Figure 2.4 shows initial objects for locations: environmental membrane, region "Water" and region "Boundary". Figure 2.4 shows initial objects for three locations: environment membrane, region 'Water', and region 'Boundary'.

Creating new initial objects can be performed by clicking "Add" button and it will show a dialog box as seen in Figure 2.19 (a). In this Figure the objects will be initialized in the local membrane [3, 2]. If the objects will be initialized in a region "A region" option should be selected and it will require the region id to be selected from a drop-down list of available regions, as shown in Figure 2.19 (b).

For objects to be defined in environment membrane, option "Environment Membrane" should be selected. For all local membrane, option "All local membranes" should be selected. The multiset is defined by typing the positive numbers in the right column of



Data Set Id: base

Short Description: common initial data set

Location	Objects
Envir.	(T,1)
Reg (Water)	(W,800)(A,400)(E,2000)(I,1000)
Reg (Boundary)	(Z,1)

Figure 2.18: Defining Initial Objects

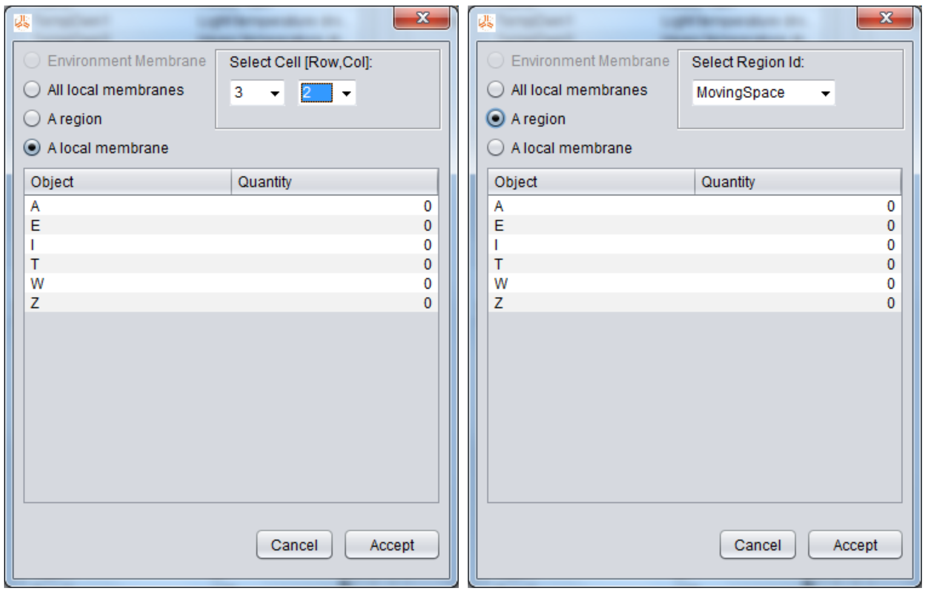
each respective object. Back to Figure 2.4, double clicking over Initial Objects' entry will pop up the dialog box for editing or deleting the entry. However, for updating the dialog box enables only the changes of multisets.

2.2.8 Defining Links

Links are special objects that carry the address of a certain membrane. Bottom-left area in Figure 2.4 provides the way for defining links and defining initial links. Clicking "Add" button will pop up a dialog box for defining the new link and its initial links. Double-clicking over an entry in the list of links will pop up the same dialog box for editing a link of the entry and its initial links. The dialog box is shown in Figure 2.20 is for link 'path'. The initial links are specified in the list by some columns. The numbers in first two columns are the row-column numbers of the link's location. The numbers in next two columns are the row-column numbers of the link's destination. The number in the last column is the quantity of the objects. Note that, when a membrane contains links 'path' of some different destinations, (i.e., some quantities), each destination will be weighted by the ration of its quantity to the total quantity. The 'canvas' in the right area shows the links drawn all over the grid. The dialog box has two modes: "Add" mode and "Delete" mode. When in "Add" mode, the initial links will be placed in a membrane by pressing the mouse button over the location where the link will be placed and dragging (while pressing the button) to the destination. If the destination is not adjacent to the location a prompt will appear questioning whether the link will be broken down becoming some links as it passes through adjacent membranes or the link will be kept as it is defined. This feature (distributing the link) is useful for defining a path.

2.2.9 Defining Event Rules and Scheduling Events

As described in the definition of Grid Systems, an event rule is just an ordinary reaction rule. It should be made so that it will be reacted at the time that it is previously scheduled. Since in Grid Systems the definition of the timer does not exist, to activate event E at time=d a timer should be defined as a reaction that



Environment Membrane | Select Cell [Row,Col]: 3, 2
All local membranes
A region
[✓ A local membrane]

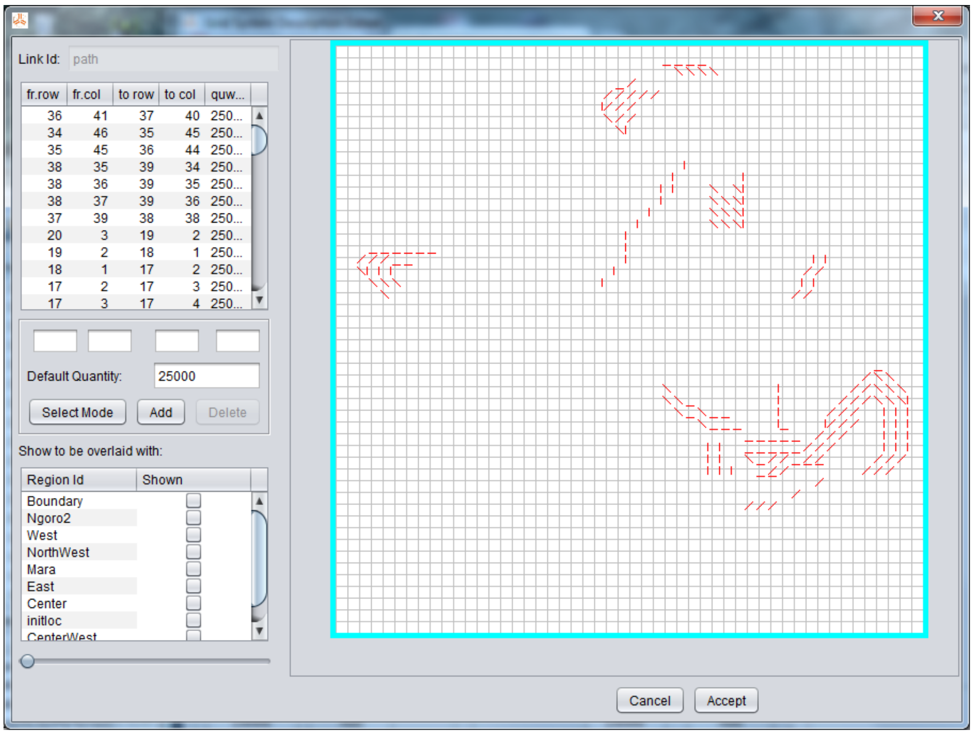
Object	Quantity
A	0
E	0
I	0
T	0
W	0
Z	0

Environment Membrane | Select Region Id: MovingSpace
All local membranes
[✓ A region]
A local membrane

Object	Quantity
A	0
E	0
I	0
T	0
W	0
Z	0

(a) In a membrane (b) In a region

Figure 2.19: Defining Initial Objects



Link Id: path

fr.row	fr.col	to row	to col	quw...
36	41	37	40	250...
34	46	35	45	250...
35	45	36	44	250...
38	35	39	34	250...
38	36	39	35	250...
38	37	39	36	250...
37	39	38	38	250...
20	3	19	2	250...
19	2	18	1	250...
18	1	17	2	250...
17	2	17	3	250...
17	3	17	4	250...

Default Quantity: 25000

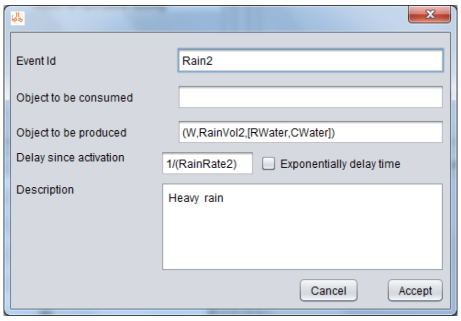
Show to be overlaid with:

Region Id	Shown
Boundary	☐
Ngoro2	☐
West	☐
NorthWest	☐
Mara	☐
East	☐
Center	☐
initloc	☐
CenterWest	☐

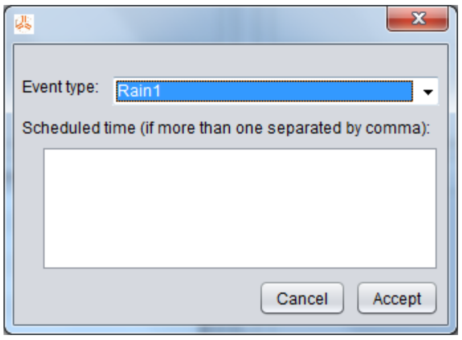
Figure 2.20: Defining Initial Links

- has duration d, and should be activated at time=0, such that,
- at time=d it will end and produce a special object that is required as a reactant by event E.

Therefore, the rules for an event should be defined to consume that special object. Note that, those are the things that should be defined according to the definition of Grid Systems. In its higher level definition (also, as adopted by the software) the event rule is specified by the product, as seen in Figure 2.2.9 (a). Since in the model the event


Event Id: Rain2
Object to be consumed:
Object to be produced: (W,RainVol2,[RWater,CWater])
Delay since activation: 1/(RainRate2) ☐ Exponentially delay time
Description: Heavy rain

(a) Defining an Event Rule


Event type: Rain1
Scheduled time (if more than one separated by comma):

(b) Scheduling Events

can happen in several points of time, the timer will be specified by (event rule id, set of time points) as shown in Figure 2.2.9 (b). In the dialog box the time points are positive numbers separated by commas.

2.3 Simulating the Model

When the model has been defined, the model can be simulated through several ways:

- Running the simulation and observing the configurations either as numbers or graphical chart.
- Running the simulation and animating the progress.
- Step-by-step running for observing certain model's behaviour and its effect to the others.

The software provide those in "Simulation" Mode and separated in three sub-modes: "Growth", "Animation", and "Trace Transitions". Interfaces of sub-modes are shown respectively in Figure 2.5, Figure 2.6, and Figure 2.28. For all sub-modes, on the top area some simulation parameters should be defined:

- Time slice of the simulation: the time interval that all points inside the interval that will be considered the same.
- Showing the progress either by time or by iteration.
- Start time to show the progress.
- Interval of observation (either time interval or the number of iterations).
- Time and iteration limits, both are required.

Changing those parameters can be performed through a dialog box that will be popped up after pressing button "Edit Parameters".

2.3.1 "Growth" sub-mode

This sub-mode is intended for observing how a model will change its state from time to time during the simulation. The software provides two different ways to observe it: (1) as plain textual output, and (2) as a chart. The first one can be controlled through the options in the upper-left area (labelled as "Tabular Output") and the second can be controlled through the options in the lower-left area (labelled "Growth Chart Output"). For either ways, the objects to be observed should be specified first through the right area (labelled "Object to be Observed."

The list in this area specifies which objects that will be observed.

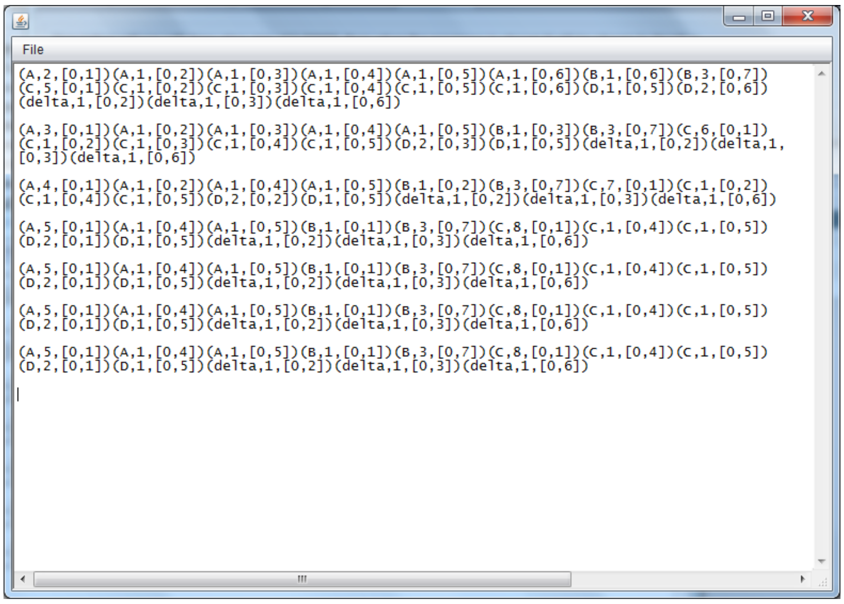
- The first column: the list specifies the object ID to be displayed.
- The second column: the list specifies the locations of respective object. The location can be either a certain membrane, or a certain region, or the environment membrane, or all membranes.
- The third column: the list specifies whether the objects are aggregated to be one number or not (one number from each membrane) when in the second column it is specified by a region.
- The fourth column: the list specifies the colour of the object when the output is a chart. The sequence of objects in the list can be used as the sequence of columns of output, therefore, some buttons are provided for arranging the sequence. "Tabular Output" produces textual output that can be opted either as multisets or column-wise numbers. Observing of small numbers of objects and locations as multisets could be useful.

When the output is needed for further analysis, namely using spreadsheets (CSV), column-wise numbers could be suitable by inserting appropriate separators between numbers (tabs, semicolons, commas, etc.). The appearance of the multisets/numbers can be sort following either their location, or their object ids, or as sequenced in the table at the right area. By pressing "Run this simulation" the simulation will be running and output will be shown in a dialog box.

The "Growth Chart Output" produces graphical output with each colour lines for each objects as specified in the table at the right area. Some other parameters can be specified to enhance the appearance of the chart. By pressing "Run this simulation" the simulation will be running, and at the same time, the chart will be drawn in a dialog box.

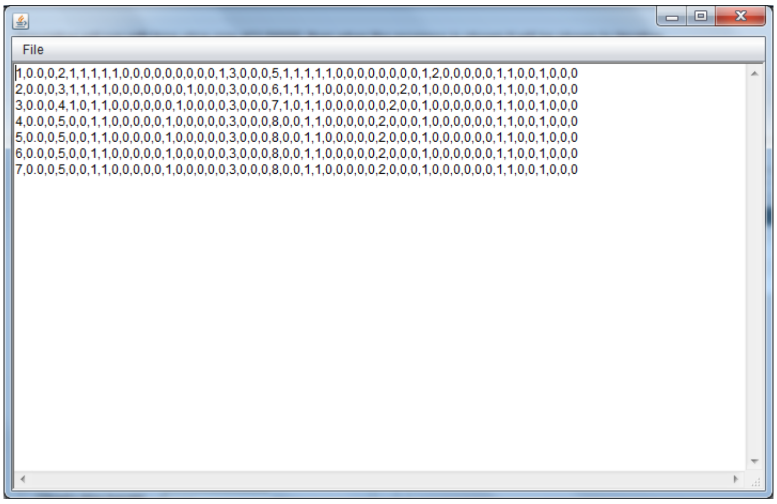
2.3.2 "Animation" sub-mode

Animating the simulation is more suitable for observing the spatial aspect of the model. By this sub-mode the state change of each membrane can be observed visually as seen in Figure 2.24. The software requires knowing which objects are to be shown, and drawn by what colour/shape if being shown. Therefore, the interface facilitates the modeller to define those properties. Besides, since the animation can take a long time, it could be useful to capture the frames of animation for later observation. Also, it could be useful to dump (saved all animation state in files) the simulation so that the simulation can be restarted from the dumped of certain time point. The modeller should supply further



```
(A,2,[0,1])(A,1,[0,2])(A,1,[0,3])(A,1,[0,4])(A,1,[0,5])(A,1,[0,6])(B,1,[0,6])(B,3,[0,7])
(C,5,[0,1])(C,1,[0,2])(C,1,[0,3])(C,1,[0,4])(C,1,[0,5])(C,1,[0,6])(D,1,[0,5])(D,2,[0,6])
(delta,1,[0,2])(delta,1,[0,3])(delta,1,[0,6])

(A,3,[0,1])(A,1,[0,2])(A,1,[0,3])(A,1,[0,4])(A,1,[0,5])(B,1,[0,3])(B,3,[0,7])(C,6,[0,1])
(C,1,[0,2])(C,1,[0,3])(C,1,[0,4])(C,1,[0,5])(D,2,[0,3])(D,1,[0,5])(delta,1,[0,2])(delta,1,
[0,3])(delta,1,[0,6])

(A,4,[0,1])(A,1,[0,2])(A,1,[0,4])(A,1,[0,5])(B,1,[0,2])(B,3,[0,7])(C,7,[0,1])(C,1,[0,2])
(C,1,[0,4])(C,1,[0,5])(D,2,[0,2])(D,1,[0,5])(delta,1,[0,2])(delta,1,[0,3])(delta,1,[0,6])

(A,5,[0,1])(A,1,[0,4])(A,1,[0,5])(B,1,[0,1])(B,3,[0,7])(C,8,[0,1])(C,1,[0,4])(C,1,[0,5])
(D,2,[0,1])(D,1,[0,5])(delta,1,[0,2])(delta,1,[0,3])(delta,1,[0,6])

(A,5,[0,1])(A,1,[0,4])(A,1,[0,5])(B,1,[0,1])(B,3,[0,7])(C,8,[0,1])(C,1,[0,4])(C,1,[0,5])
(D,2,[0,1])(D,1,[0,5])(delta,1,[0,2])(delta,1,[0,3])(delta,1,[0,6])

(A,5,[0,1])(A,1,[0,4])(A,1,[0,5])(B,1,[0,1])(B,3,[0,7])(C,8,[0,1])(C,1,[0,4])(C,1,[0,5])
(D,2,[0,1])(D,1,[0,5])(delta,1,[0,2])(delta,1,[0,3])(delta,1,[0,6])

(A,5,[0,1])(A,1,[0,4])(A,1,[0,5])(B,1,[0,1])(B,3,[0,7])(C,8,[0,1])(C,1,[0,4])(C,1,[0,5])
(D,2,[0,1])(D,1,[0,5])(delta,1,[0,2])(delta,1,[0,3])(delta,1,[0,6])
```

Figure 2.21: Output as Multisets

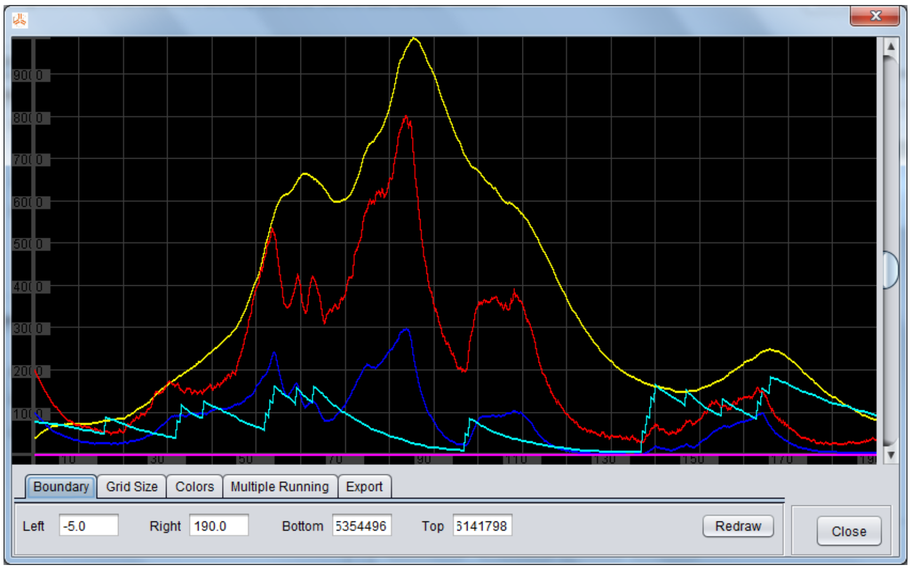


```
1,0,0,0,2,1,1,1,1,1,0,0,0,0,0,0,0,0,0,1,3,0,0,0,5,1,1,1,1,1,0,0,0,0,0,0,0,0,1,2,0,0,0,0,0,1,1,0,0,1,0,0,0
2,0,0,0,3,1,1,1,1,0,0,0,0,0,0,0,1,0,0,0,3,0,0,0,6,1,1,1,1,0,0,0,0,0,0,0,2,0,1,0,0,0,0,0,0,1,1,0,0,1,0,0,0
3,0,0,0,4,1,0,1,1,0,0,0,0,0,0,1,0,0,0,0,3,0,0,0,7,1,0,1,1,0,0,0,0,0,0,2,0,0,1,0,0,0,0,0,0,1,1,0,0,1,0,0,0
4,0,0,0,5,0,0,1,1,0,0,0,0,0,1,0,0,0,0,0,3,0,0,0,8,0,0,1,1,0,0,0,0,0,2,0,0,0,1,0,0,0,0,0,0,1,1,0,0,1,0,0,0
5,0,0,0,5,0,0,1,1,0,0,0,0,0,1,0,0,0,0,0,3,0,0,0,8,0,0,1,1,0,0,0,0,0,2,0,0,0,1,0,0,0,0,0,0,1,1,0,0,1,0,0,0
6,0,0,0,5,0,0,1,1,0,0,0,0,0,1,0,0,0,0,0,3,0,0,0,8,0,0,1,1,0,0,0,0,0,2,0,0,0,1,0,0,0,0,0,0,1,1,0,0,1,0,0,0
7,0,0,0,5,0,0,1,1,0,0,0,0,0,1,0,0,0,0,0,3,0,0,0,8,0,0,1,1,0,0,0,0,0,2,0,0,0,1,0,0,0,0,0,0,1,1,0,0,1,0,0,0
```

Figure 2.22: Output as Commas Separated Values (CSV)

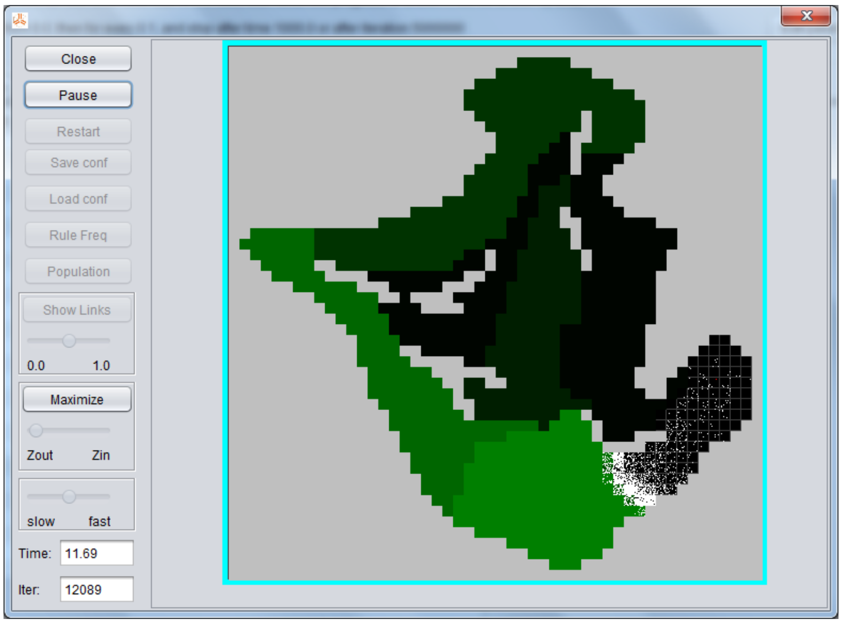
parameters for this: the inter-frame interval for capturing, inter-dumping interval, and prefix of file names for dumping as well as the directory for saving the files. After the simulation button is pressed, a dialog box is popped up. In the left area of the dialog box there are some control buttons and one of them, "Pause" button, is in the pressed state and the canvas in the right area shows initial state of simulation. The simulation will run and the animation will be shown up in the canvas after un-pressing the "Pause" button. During running, the only buttons that can be pressed are:

- "Close" button is for stopping the simulation and closing the box.
- "Pause" button is for pausing the simulation and then enabling other buttons.



Left -5.0 Right 190.0 Bottom 5354496 Top 5141798

Figure 2.23: Output as Chart



Time: 11.69

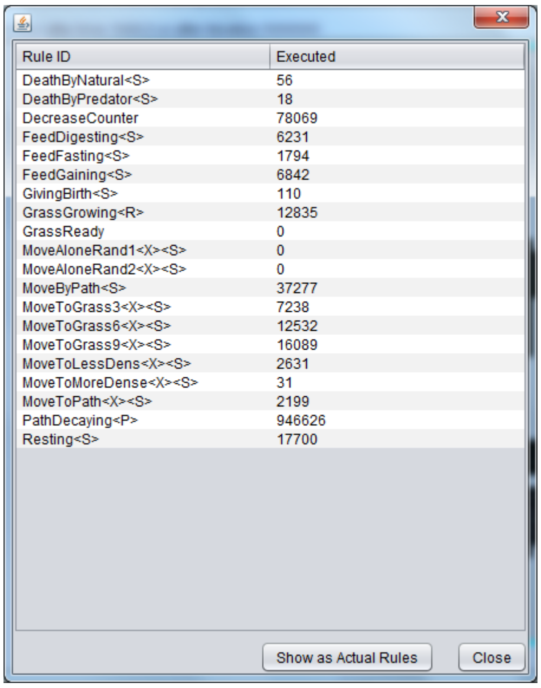
Iter: 12089

Figure 2.24: Animation of Simulation

- "Maximize" button is for enlarging the canvas to the largest that can draw in the screen.

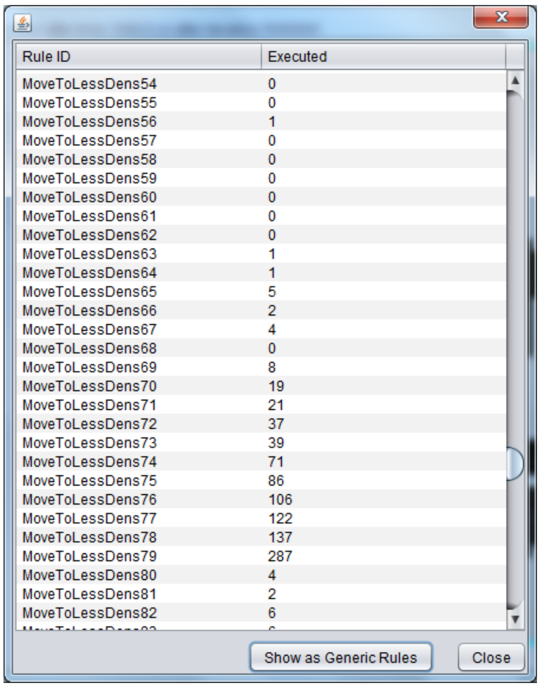
During being paused, some other buttons are enabled:

- "Restart" button is for restarting the simulation from initial configuration.
- "Save Config" button is for saving the current configuration to a file so that at any time later the simulation can be restored and restarted from this configuration. When this button is pressed the dialog box for saving into a file will be popped up.



Rule ID	Executed
DeathByNatural<S>	56
DeathByPredator<S>	18
DecreaseCounter	78069
FeedDigesting<S>	6231
FeedFasting<S>	1794
FeedGaining<S>	6842
GivingBirth<S>	110
GrassGrowing<R>	12835
GrassReady	0
MoveAloneRand1<X><S>	0
MoveAloneRand2<X><S>	0
MoveByPath<S>	37277
MoveToGrass3<X><S>	7238
MoveToGrass6<X><S>	12532
MoveToGrass9<X><S>	16089
MoveToLessDens<X><S>	2631
MoveToMoreDense<X><S>	31
MoveToPath<X><S>	2199
PathDecaying<P>	946626
Resting<S>	17700

(a) Template rules



Rule ID	Executed
MoveToLessDens54	0
MoveToLessDens55	0
MoveToLessDens56	1
MoveToLessDens57	0
MoveToLessDens58	0
MoveToLessDens59	0
MoveToLessDens60	0
MoveToLessDens61	0
MoveToLessDens62	0
MoveToLessDens63	1
MoveToLessDens64	1
MoveToLessDens65	5
MoveToLessDens66	2
MoveToLessDens67	4
MoveToLessDens68	0
MoveToLessDens69	8
MoveToLessDens70	19
MoveToLessDens71	21
MoveToLessDens72	37
MoveToLessDens73	39
MoveToLessDens74	71
MoveToLessDens75	86
MoveToLessDens76	106
MoveToLessDens77	122
MoveToLessDens78	137
MoveToLessDens79	287
MoveToLessDens80	4
MoveToLessDens81	2
MoveToLessDens82	6

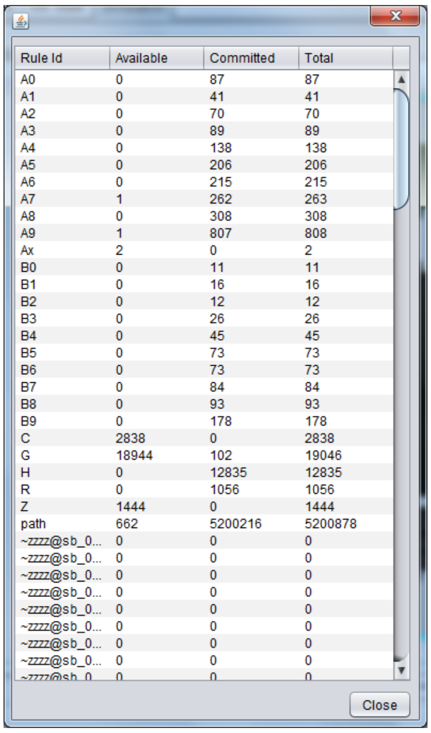
(b) Actual Rules

Figure 2.25: Rules' Frequencies

- "Load Config" button is for restoring the saved configuration.
- "Rule Freq" button is for inspecting the frequency of the use of each rules. This information is useful to inspect whether in the model there is a rule that is never or rarely used so far because of improper parameter values. Therefore, when it is the case, the modeller could adjust the parameter values. The frequencies can be shown based on either template rules, as shown in Figure 2.32 (a), or actual rules, as shown in Figure 2.32 (b).
- "Population" button is for inspecting the current total population of each objects as shown in Figure 2.26.
- "Show Links" button is for showing the current links as arrows shown in the canvas. Since the links have some different directions and each direction has its own weight, the interface provides a sliding bar to set a limit so that directions that will be shown should be above this limit. By this limitation the modeller could inspect on the major directions only (disregarding the minors). Also, the links can be shown either as being overlaid with the populations or as themselves by toggling this button as shown in Figure 2.31.

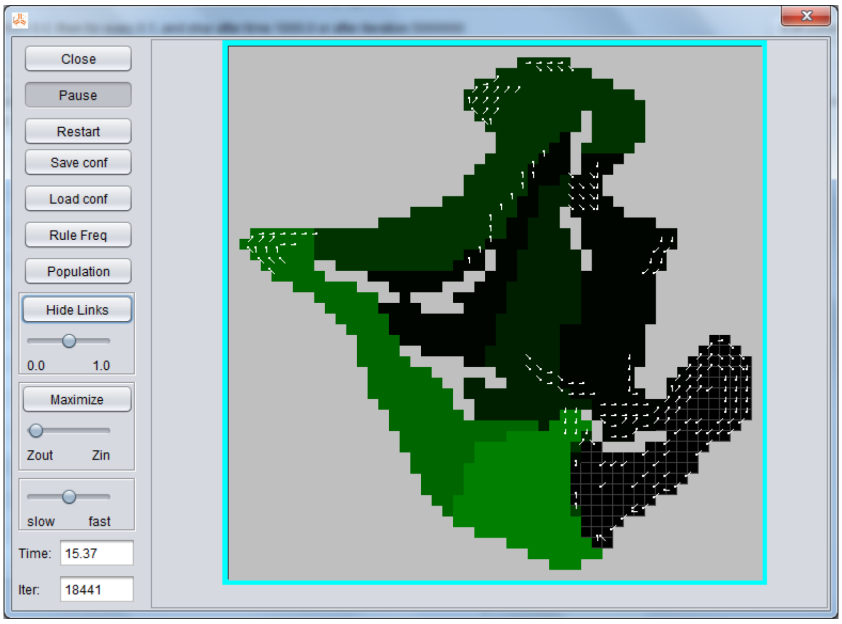
2.3.3 "Tracing Reactions" sub-mode

During developing a model the rules are constructed and their parameters are adjusted so that the model will behave as expected. This sub-mode can help modeller in verifying the behaviour of the rules by step-by-step tracing in the simulation. Figure 2.7 shows the



Rule Id	Available	Committed	Total
A0	0	87	87
A1	0	41	41
A2	0	70	70
A3	0	89	89
A4	0	138	138
A5	0	206	206
A6	0	215	215
A7	1	262	263
A8	0	308	308
A9	1	807	808
Ax	2	0	2
B0	0	11	11
B1	0	16	16
B2	0	12	12
B3	0	26	26
B4	0	45	45
B5	0	73	73
B6	0	73	73
B7	0	84	84
B8	0	93	93
B9	0	178	178
C	2838	0	2838
G	18944	102	19046
H	0	12835	12835
R	0	1056	1056
Z	1444	0	1444
path	662	5200216	5200878
~zzzz@sb_0...	0	0	0
~zzzz@sb_0...	0	0	0
~zzzz@sb_0...	0	0	0
~zzzz@sb_0...	0	0	0
~zzzz@sb_0...	0	0	0
~zzzz@sb_0...	0	0	0
~zzzz@sb_0...	0	0	0
~zzzz@sb_0...	0	0	0
~zzzz@sb_0	0	0	0

Figure 2.26: Current Populations

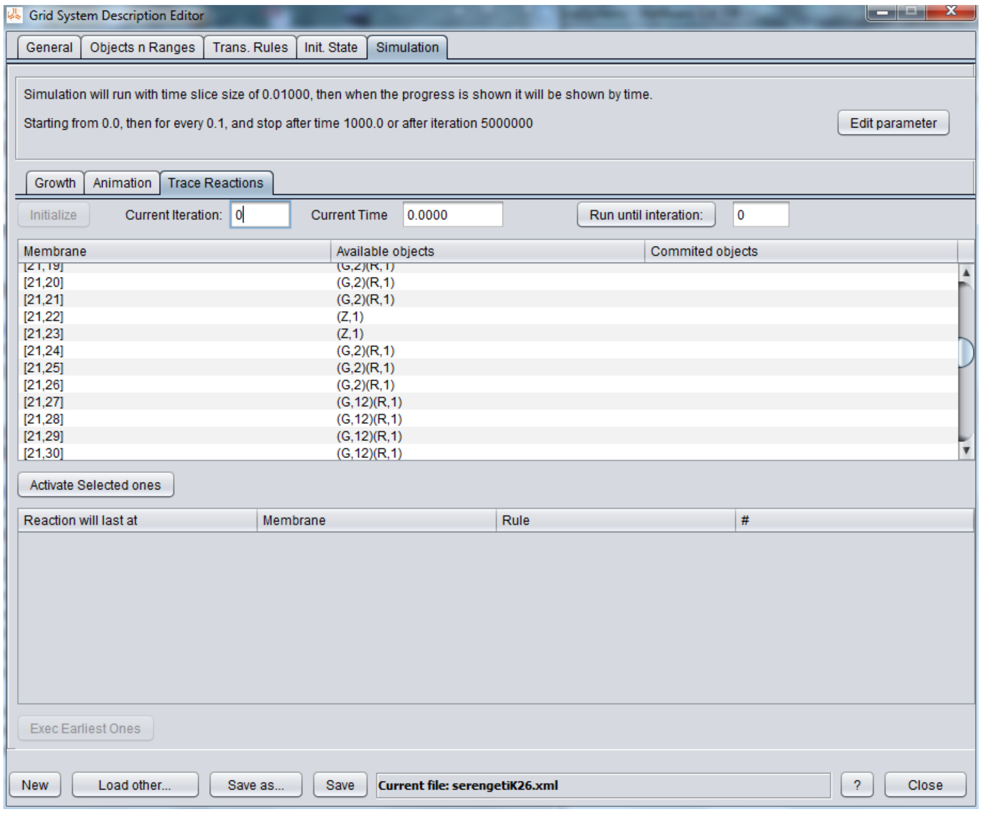


Time: 15.37

Iter: 18441

Figure 2.27: Displaying the Links

interface at the beginning in using it. “Initialize” button is to place the initial objects in each membrane so that the initialization can be inspected through upper list. Figure 2.28 shows the interface just after initialisation so that upper list is containing the initial multiset in each membrane. By double-clicking over an entry in the list, a dialog box



Simulation will run with time slice size of 0.01000, then when the progress is shown it will be shown by time.

Starting from 0.0, then for every 0.1, and stop after time 1000.0 or after iteration 5000000

Current Iteration: 0 Current Time 0.0000 Run until interation: 0

Membrane	Available objects	Commited objects
[21,19]	(G,2)(R,1)	
[21,20]	(G,2)(R,1)	
[21,21]	(G,2)(R,1)	
[21,22]	(Z,1)	
[21,23]	(Z,1)	
[21,24]	(G,2)(R,1)	
[21,25]	(G,2)(R,1)	
[21,26]	(G,2)(R,1)	
[21,27]	(G,12)(R,1)	
[21,28]	(G,12)(R,1)	
[21,29]	(G,12)(R,1)	
[21,30]	(G,12)(R,1)	

Reaction will last at	Membrane	Rule	#

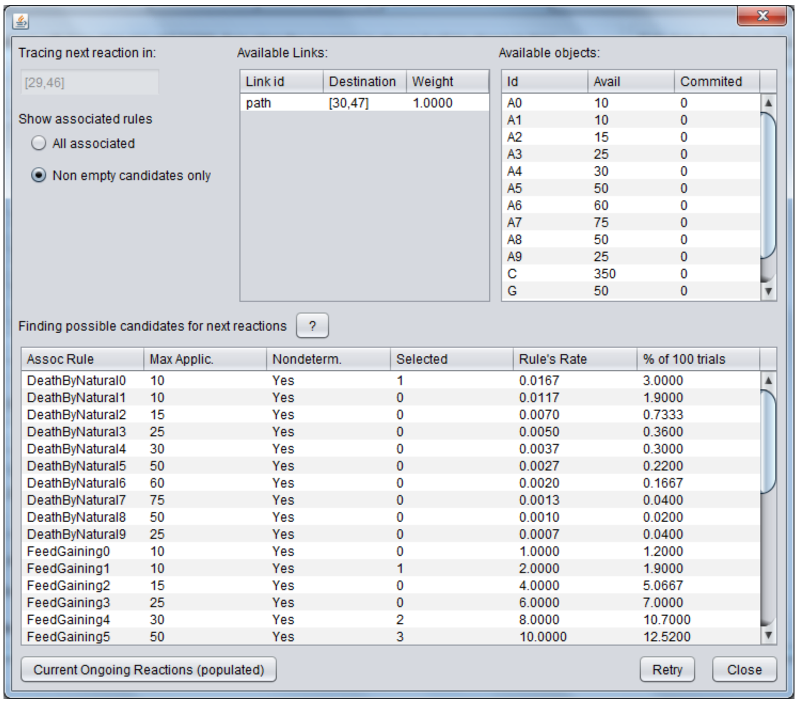
Current file: serengetiK26.xml

Figure 2.28: After Initialisation

is popped up as in Figure 2.29. The dialog box shows the objects currently alive in the membrane both available for reactions or being committed to on-going reactions (top-right list). It also shows the current paths including their directions and the weights (top-center list). The bottom list shows the result candidate reactions that were identified for current available objects containing columns:

- Assoc Rule: the rule associated to the membrane.
- Max Applic.: the maximm number of times that this rule can be applied.
- Nondeterm.: the status whether there is another rule competing for the same reactants (non-deterministic) or there is not.
- Selected: the number of reactions that “could be” selected by stochastic selection process out of the maximum in second column.
- Rule’s Rate: shows the rate of the rule (as defined to that rule).
- % of 100 trials: empirical possibility of the reaction to be selected effectively from 100 trials.

Note that the 5th and 6th columns are useful in evaluating the rates given to the rules. The difference is due to the stochastic selection process is based on the propensities of the rules


Tracing next reaction in:

[29,46]

Show associated rules

All associated

[✓ Non empty candidates only]

Available Links:

Link id	Destination	Weight
path	[30,47]	1.0000

Available objects:

Id	Avail	Commited
A0	10	0
A1	10	0
A2	15	0
A3	25	0
A4	30	0
A5	50	0
A6	60	0
A7	75	0
A8	50	0
A9	25	0
C	350	0
G	50	0

Finding possible candidates for next reactions

Assoc Rule	Max Applic.	Nondeterm.	Selected	Rule's Rate	% of 100 trials
DeathByNatural0	10	Yes	1	0.0167	3.0000
DeathByNatural1	10	Yes	0	0.0117	1.9000
DeathByNatural2	15	Yes	0	0.0070	0.7333
DeathByNatural3	25	Yes	0	0.0050	0.3600
DeathByNatural4	30	Yes	0	0.0037	0.3000
DeathByNatural5	50	Yes	0	0.0027	0.2200
DeathByNatural6	60	Yes	0	0.0020	0.1667
DeathByNatural7	75	Yes	0	0.0013	0.0400
DeathByNatural8	50	Yes	0	0.0010	0.0200
DeathByNatural9	25	Yes	0	0.0007	0.0400
FeedGaining0	10	Yes	0	1.0000	1.2000
FeedGaining1	10	Yes	1	2.0000	1.9000
FeedGaining2	15	Yes	0	4.0000	5.0667
FeedGaining3	25	Yes	0	6.0000	7.0000
FeedGaining4	30	Yes	2	8.0000	10.7000
FeedGaining5	50	Yes	3	10.0000	12.5200

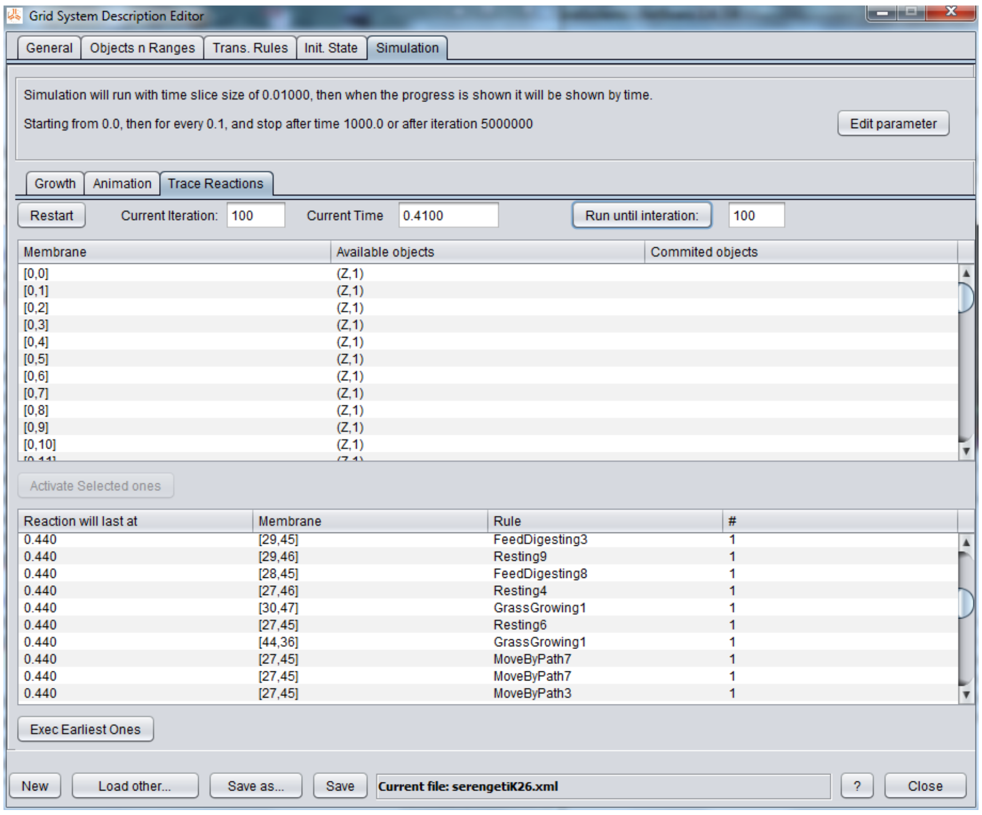
Figure 2.29: State of a Membrane

where the rates are only the factors for them besides the number of possible combinations of reactants. "Activate selected ones" in Figure 2.28 is the button to proceed reaction candidates above. After pressing the button, the simulation will put the reactions into the list of on-going reactions. Then, the list below will show the current on-going reactions sorted by the time of the reactions' termination. Figure 2.30 shows the list of multisets in each membrane that is the list of on-going reactions after 100 iterations.

2.4 Generating LATEX Codes of the Model

As described in Section 2.1.1, the software provides a function for exporting the model to LATEX codes so that it the codes can be inserted in a LATEX formatted report as well as a complete LATEX document. When the button "Generate LaTEX..." is pressed, a dialog box for selecting which part of model that will be outputted as shown in Figure 2.33 (a), and the codes that are generated as shown in Figure 2.33 (b). Then, in its output window, the output can be either saved in a file or copied and pasted the parts into the report.

Furthermore, the software provides a fuction to export the chart generated by the simulator to TikZ commands. This function can be accessed through pressing tab "Export" in the dialog box showing the output chart (at the bottom area) in Figure 2.23. Bypressing "Generate TikZ code", TikZ code will appear in a window as shown in Figure 2.34. Since the output is just TikZ codes, it needs to be inserted in a LATEX document.



Simulation will run with time slice size of 0.01000, then when the progress is shown it will be shown by time.

Starting from 0.0, then for every 0.1, and stop after time 1000.0 or after iteration 5000000

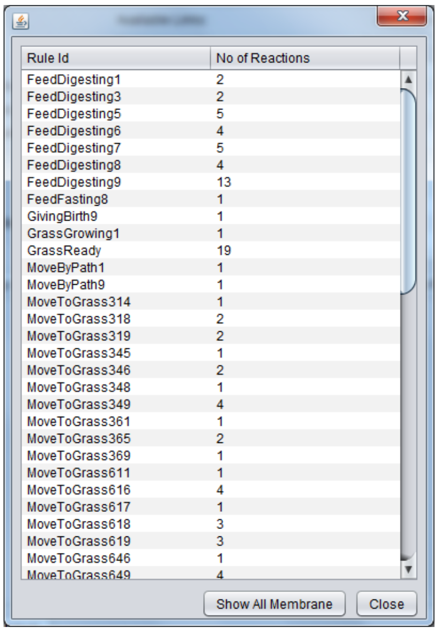
Current Iteration: 100 Current Time 0.4100 Run until interation: 100

Membrane	Available objects	Commited objects
[0,0]	(Z,1)	
[0,1]	(Z,1)	
[0,2]	(Z,1)	
[0,3]	(Z,1)	
[0,4]	(Z,1)	
[0,5]	(Z,1)	
[0,6]	(Z,1)	
[0,7]	(Z,1)	
[0,8]	(Z,1)	
[0,9]	(Z,1)	
[0,10]	(Z,1)	

Reaction will last at	Membrane	Rule	#
0.440	[29,45]	FeedDigesting3	1
0.440	[29,46]	Resting9	1
0.440	[28,45]	FeedDigesting8	1
0.440	[27,46]	Resting4	1
0.440	[30,47]	GrassGrowing1	1
0.440	[27,45]	Resting6	1
0.440	[44,36]	GrassGrowing1	1
0.440	[27,45]	MoveByPath7	1
0.440	[27,45]	MoveByPath7	1
0.440	[27,45]	MoveByPath3	1

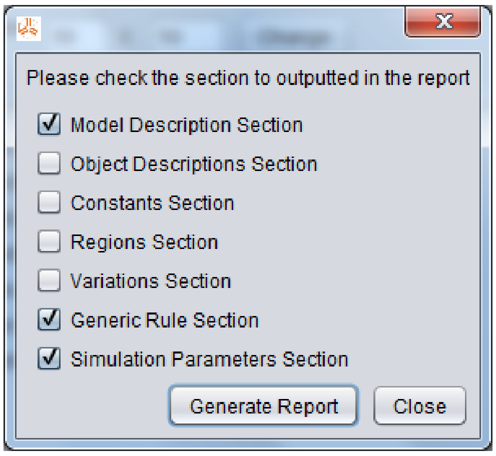
Current file: serengetiK26.xml

Figure 2.30: Step-by-Step Tracing Reaction

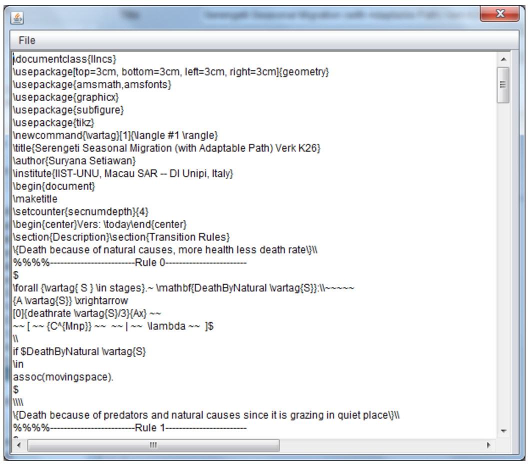


Rule Id	No of Reactions
FeedDigesting1	2
FeedDigesting3	2
FeedDigesting5	5
FeedDigesting6	4
FeedDigesting7	5
FeedDigesting8	4
FeedDigesting9	13
FeedFasting8	1
GivingBirth9	1
GrassGrowing1	1
GrassReady	19
MoveByPath1	1
MoveByPath9	1
MoveToGrass314	1
MoveToGrass318	2
MoveToGrass319	2
MoveToGrass345	1
MoveToGrass346	2
MoveToGrass348	1
MoveToGrass349	4
MoveToGrass361	1
MoveToGrass365	2
MoveToGrass369	1
MoveToGrass611	1
MoveToGrass616	4
MoveToGrass617	1
MoveToGrass618	3
MoveToGrass619	3
MoveToGrass646	1
MoveToGrass649	4

Figure 2.31: On-going Reactions in a Membrane


Please check the section to outputted in the report
☑ Model Description Section
☐ Object Descriptions Section
☐ Constants Section
☐ Regions Section
☐ Variations Section
☑ Generic Rule Section
☑ Simulation Parameters Section
Generate Report
Close

(a) Part Selection

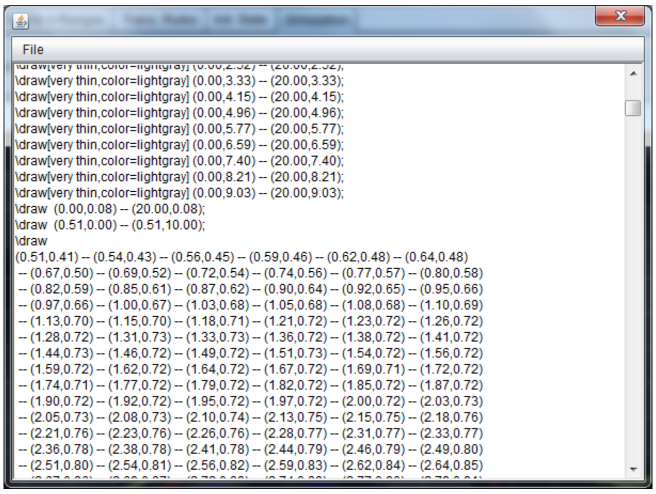


```
\documentclass{llncs}
\usepackage[top=3cm, bottom=3cm, left=3cm, right=3cm]{geometry}
\usepackage{amsmath,amsfonts}
\usepackage{graphicx}
\usepackage{subfigure}
\usepackage{tikz}
\newcommand{\vartag}[1]{\langle #1 \rangle}
\title{Serengeti Seasonal Migration (with Adaptable Path) Verk K26}
\author{Suryana Setiawan}
\institute{IIST-UNU, Macau SAR -- DI Unipi, Italy}
\begin{document}
\maketitle
\setcounter{secnumdepth}{4}
\begin{center}Vers: \today\end{center}
\section{Description}\section{Transition Rules}
\{Death because of natural causes, more health less death rate\}\\
%%%%------------------------Rule 0------------------------
$
\forall {\vartag{ S } \in stages}.~ \mathbf{DeathByNatural \vartag{S}}:\\~~~~~
{A \vartag{S}} \xrightarrow
[0]{deathrate \vartag{S}/3}{Ax} ~~
~~ [ ~~ {C^{Mnp}} ~~  ~~ | ~~  \lambda ~~  ]$
\\
if $DeathByNatural \vartag{S}
\in
assoc(movingspace).
$
\\\
\{Death because of predators and natural causes since it is grazing in quiet place\}\\
%%%%------------------------Rule 1------------------------
```

(b) LaTEX codes

Figure 2.32: LATEX Documenting

Figure 2.33: Selecting Parts of Model for Generating LaTEX Codes



```
\draw[very thin,color=lightgray] (0.00,3.33) -- (20.00,3.33);
\draw[very thin,color=lightgray] (0.00,4.15) -- (20.00,4.15);
\draw[very thin,color=lightgray] (0.00,4.96) -- (20.00,4.96);
\draw[very thin,color=lightgray] (0.00,5.77) -- (20.00,5.77);
\draw[very thin,color=lightgray] (0.00,6.59) -- (20.00,6.59);
\draw[very thin,color=lightgray] (0.00,7.40) -- (20.00,7.40);
\draw[very thin,color=lightgray] (0.00,8.21) -- (20.00,8.21);
\draw[very thin,color=lightgray] (0.00,9.03) -- (20.00,9.03);
\draw  (0.00,0.08) -- (20.00,0.08);
\draw  (0.51,0.00) -- (0.51,10.00);
\draw
(0.51,0.41) -- (0.54,0.43) -- (0.56,0.45) -- (0.59,0.46) -- (0.62,0.48) -- (0.64,0.48)
 -- (0.67,0.50) -- (0.69,0.52) -- (0.72,0.54) -- (0.74,0.56) -- (0.77,0.57) -- (0.80,0.58)
 -- (0.82,0.59) -- (0.85,0.61) -- (0.87,0.62) -- (0.90,0.64) -- (0.92,0.65) -- (0.95,0.66)
 -- (0.97,0.66) -- (1.00,0.67) -- (1.03,0.68) -- (1.05,0.68) -- (1.08,0.68) -- (1.10,0.69)
 -- (1.13,0.70) -- (1.15,0.70) -- (1.18,0.71) -- (1.21,0.72) -- (1.23,0.72) -- (1.26,0.72)
 -- (1.28,0.72) -- (1.31,0.73) -- (1.33,0.73) -- (1.36,0.72) -- (1.38,0.72) -- (1.41,0.72)
 -- (1.44,0.73) -- (1.46,0.72) -- (1.49,0.72) -- (1.51,0.73) -- (1.54,0.72) -- (1.56,0.72)
 -- (1.59,0.72) -- (1.62,0.72) -- (1.64,0.72) -- (1.67,0.72) -- (1.69,0.71) -- (1.72,0.72)
 -- (1.74,0.71) -- (1.77,0.72) -- (1.79,0.72) -- (1.82,0.72) -- (1.85,0.72) -- (1.87,0.72)
 -- (1.90,0.72) -- (1.92,0.72) -- (1.95,0.72) -- (1.97,0.72) -- (2.00,0.72) -- (2.03,0.73)
 -- (2.05,0.73) -- (2.08,0.73) -- (2.10,0.74) -- (2.13,0.75) -- (2.15,0.75) -- (2.18,0.76)
 -- (2.21,0.76) -- (2.23,0.76) -- (2.26,0.76) -- (2.28,0.77) -- (2.31,0.77) -- (2.33,0.77)
 -- (2.36,0.78) -- (2.38,0.78) -- (2.41,0.78) -- (2.44,0.79) -- (2.46,0.79) -- (2.49,0.80)
 -- (2.51,0.80) -- (2.54,0.81) -- (2.56,0.82) -- (2.59,0.83) -- (2.62,0.84) -- (2.64,0.85)
```

Figure 2.34: On-going Reactions in a Membrane

Chapter 3

Encoding the Model

In this part how the components are encoded in an XML tagged format will be described. To avoid going directly to its complexity we will present the description in several sections. Firstly, its basic notations that is used for encoding directly from the model will be presented. Advance notations that enables writing templates of rules and scheduled external events will be in the next section.

3.1 Basic Notations

3.1.1 Document Root

The data in one file contains only one description of Grid System. The root should be tagged by a pair of tags as follows:

`<gridsystem title="`*title*`">` *modelcomponents* `</gridsystem>`

Where *title* could be any string that will appear in the report/output, and *modelcomponents* is the place where all components and simulation parameters will be written. The following parts are syntax for writing the components and the simulation parameters.

3.1.2 Dimension Part

In this version of implementation the dimension of the grid should be declared to minimize the overhead cost time of the simulation. The expression when the numbers of rows is n and the number of columns is m, will be written as follows:

`<dimension NumRows="`*n*`" NumCols="`*m*`"/>`

For example, a grid of 10×10 membranes is defined.

```
<dimension NumRows="10" NumCols="10"/>
```

3.1.3 Author Part

Author name(s) and institution(s) can be placed in description data to attribute the authorship of the data. They are written as follows:

`<author institution="`*institutons*`" name="`*authornames*`"/>`

For example.

```
<author name="Suryana Setiawan"
    institution="Dipartimento di Informatica, Universit\'{a} di Pisa, Italy"
/>
```

3.1.4 Description Part

A textual description (and some model's notes) can be embedded into the model. It could be also exported for formatting by LaTeXas described in 2.4. Some LaTeXformatting codes can be inserted in this description for creating better formatted output.

It should be written in the tag pair as follows.

`<decription>` *modeldescription* `</decription>`

For example,

```
<description>
This model is the version without external events of
previously defined model (Mosquito vers 1.0.)
All external events are removed.
The objective is for a more focused observation on
mosquito's behaviours.
</description>
```

3.1.5 Objects Part

In its formal definition the model contains the second component Σ which is the set of objects. For this implementation all object's identifications should be declared in the beginning to ease the parser in its work. In its following syntax the modeller can put add object's description/annotation to each possibly cryptic id. However, the meaningful part to the simulator is just the id.

The list should be labelled by pair

`<objectlist>`*objectlist*`</objectlist>`

The objects are declared in *objectlist* by singleton

`<objectdef id="`*id*`" desc="`*description*`"/>`

where the *id* is the object identification that

For example,

```
<objectlist>
     <objectdef desc="Animal strength at level-0" id="A0"/>
     <objectdef desc="Animal strength at level-1" id="A1"/>
     <objectdef desc="One unit of Grass" id="G"/>
</objectlist>
```

3.1.6 Reaction Rules

The third and the fourth components of the model formal definition are the table of rules and the table of associations. The rules are written in a group delimited by a pair:

`<transitionrules>` *reactionrules* `</transitionrules>`

Each reaction rule in *reactionrules* is written as follows

```
<rule id=" id" rate="rate" duration="duration" mode="mode">
     <reactants>reactants</reactants>
     <products>products</products>
     <promoters>promoters</promoters>
     <inhibitors>inhibitors</inhibitors>
     <associations> associations</associations>
</rule>
```

rate is the rate of the reaction; when the attribute rate is omitted its value is 1.0. *duration* is the duration of the reaction. When the attribute duration is omitted its value is 1 over reaction rate. When *mode* is exp, the reaction has exponential distributed duration time with the expected rate *duration*, otherwise the reaction has a constant duration time which is *duration.*

products, *promoters*, and *inhibitors* are the multisets expressed by the list of object tags and each object tag is written as follows.

- For multiset (*objectid*, *n*), they are expressed by

  `<object id="objectid" quant="n"/>`

- For multiset (*objectid*, *n*, (*dr*,*dc*)), they are expressed by

  `<object id="objectid" quant="n" dr="dr" dc="dc" />`

- For multiset (*objectid*, *n*, *r,c*), they are expressed by

  `<object id="objectid" quant="n" r="r" c="c" />`

- For multiset (*objectid*, *n*, E), they are expressed by

  `<object id="objectid" quant="n" loc="E" />`

- For multiset (*objectid*, *n*, *link*), they are expressed by

  `<object id="objectid" quant="n" link=link" />`

When the value of attribute `quant` (the number of objects) is 1 (its default value) `quant` and its value can be omitted. The objects of *reactants* can only be written by the first of above since the reactants can only from the current membrane. The *associations* contains at least one membrane. When they are a list of membranes, each membrane will be expressed by singleton

<local r="r" c="c" />

For example, the following is rule table containing only a rule definition. The rule represents

$$\begin{array}{l} \mathit{Oviposing}1: \; A \xrightarrow{0.5,M} A\ E^{20}\ [WT_E^0 \mid T_E^3] \\ \text{if } \mathit{Oviposing}1 \in \text{Assoc}\{[0,0],[0,1],[0,2],[1,0],[1,1],[1,2],[2,0],[2,1],[2,2]\} \end{array}$$

```
<transitionrules>
    <rule id="Oviposing1" mode="exp" rate="0.05">
      <reactants>
         <object id="A"/>
      </reactants>
      <products>
         <object id="A"/>
         <object id="E" quant="EggsGiven"/>
      </products>
      <promoters>
         <object id="W"/>
         <object id="T" loc="E" quant="0"/>
      </promoters>
      <inhibitors>
         <object id="T" loc="E" quant="3"/>
      </inhibitors>
      <associations>
         <local r="0" c="0"/>
         <local r="0" c="1"/>
         <local r="0" c="2"/>
         <local r="1" c="0"/>
         <local r="1" c="1"/>
         <local r="1" c="2"/>
         <local r="2" c="0"/>
         <local r="2" c="1"/>
         <local r="2" c="2"/>
      </associations>
   </rule>
</transitionrules>
```

3.1.7 Initial Configuration

Initial configuration contains only the initial multisets over the objects in each membrane of the grid. They are written inside the following pair of tags.

<configuration> *configlist* </configuration>

For initial objects that are not links, *configlist* lists all nonempty membranes and each of them is written as follows.

<local r="r" c="c"> *multiset* </local>

Each object in *multiset* is written as follows.

```
<object id="objectid" quant="n"/>
```

Declaring the content of a membrane several times will be considered as the additional ones to the first one.

For links, *configlist* specify the link name first as follows.

```
<linkset id="linkname"> listoflinks</linkset>
```

listoflinks are one or more lines, each line specifies the number of rows and columns of the origin and destination membranes as well as their initial quantities, as follows.

```
<link fromrow="ro" fromcol="co" torow="rd" torow="cd" weight="w"/>
```

It specifies links from membrane [*ro*,*co*] to membrane [*rd*,*cd*]. The weights of the links in a membrane are computed as the proportion of the link quantities that is originated from hat membrane. Different link names are defined as different linksets.

3.1.8 Simulation Parameters

As described in previous section for simulation some parameters are needed both for running the simulation and for observing the output. All the parameters will be encoded inside the following tags.

```
<simulation> simulationpars </simulation>
```

Parameters for running simulation will be expressed as *imax*, *tmax*, and *tslice* as attribute values of the following singleton:

```
<limits iterationmax="imax" timemax="tmax" timeslice="tslice" />
```

The unit of time in *tlice* and *tmax* is rounded to one second.

The status for observing the output periodically is specified by adding a singleton:

```
<showprogress/>
```

When it is not specified, the simulation shows only the last configuration. The objects to be reported will be specified in the following singleton:

```
<displayobjects start="starttime"  step="steptime" bytime="val">
                          objectlist
                      </displayobjects>
```

starttime is the nonnegative real number representing the simulation time when the configuration will start to be shown. *steptime* is the interval of simulation time between two consecutive time points of showing the configuration. *bytime* is a boolean value when the value is false the *starttime* and *steptime* are based on the number of iterations instead of simulation time. *objectlist* lists the objects to display. Each object should be specified by the following singleton:

```
<dobject id="objid" loc="location" />
```

objid and *location* indicate the object and the membrane that object's status should be displayed. If attribute aggregate is given value "false" the status of the object in each membrane will be shown separately, otherwise they will be aggregated as one value.

```
<simulation>
    <limits iterationmax="1000000" timemax="190.0" timeslice="0.001"/>
    <showprogress/>
    <displayobjects bytime="true" start="0" step="0.25">
    <dobject id="A" loc="[2,2]" />
    <dobject id="I" loc="[2,2]" />
    <dobject id="E" loc="[2,2]" />
    <dobject id="W" loc="[2,2]" />
    <dobject id="T" loc="global"/>
    </displayobjects>
</simulation>
```

3.2 Advance Notations

3.2.1 Aliases Part

This part plays an important role to declare several aliases used in describing the reaction rules by using advance notation. All aliases are grouped and delimited by pair of tags as follows.

`<aliases>`*aliases*`</aliases>`

and *aliases* contains the declaration of each alias. There are some kinds of aliases:

- Region: defining a set of membranes. This notation will be useful in expressing the association of a rule to just a region instead of to each membrane of that region. It will be useful to declare initial multisets in the membranes of the region instead of declaring to each membrane.

- Constant: defining a value (could be either textual or number) by a constant name. This notation will be useful when a same value is given to many properties (the things that are given the values) in different parts of the model is replace by the constant. The changes of the value in the constant declaration will affect the values of the properties expressed by the constants. The other constants can be expressed by previously declared constants as well. Numerical constants can be defined by an arithmetic expression containing previously already defined numerical constants.

- Constant list: several constants which are named by the same identification by postfixed by series of number 0, 1, 2, ... (or 1, 2, 3, ...).

- Variation: a range of strings. This notation is useful for defining template of rules such that the rules whose similar components do not need be defined individually.

The rules for writing those aliases will be describe in the following subsections.

3.2.2 Region

A region is declared as follows.

`<region id="`*regionid*`">` *partlist* `</region>`

regionid is a unique string composed by alphanumeric characters started by a letter. *partlist* is a sequence of parts (either the membranes or other regions) such that the union operation of them contains the membrane of this region.

If the part is a membrane, it is expressed by.

`<local r="`*r*`" c="`*c*`"/>`

where r is the row number and c is the column number of the membrane in the grid. If the part is a rectangular region, it is expressed by

`<rect r1="`*r1*`" c1="`*c1*`" r2="`*r2* `" c2="`*c2*`"/>`

where *r1* and *r2* are the row numbers, and *c1* and *c2* are the column numbers such that each membrane addressed by $[r,c]$ in the grid is the part of this region, where $min(r1,r2) \leq r \leq max(r1,r2)$ and $min(c1,c2) \leq c \leq max(c1,c2)$. If the part is another region, it is expressed by

`<reg id="`*otherregionid*`"/>`

otherregionid is a unique string composed by alphanumeric characters started by a letter. It should have been declared previously.

For examples,

```
<region id="initloc">
     <rect c1="44" c2="47" r1="28" r2="28"/>
     <rect c1="45" c2="46" r1="27" r2="27"/>
     <rect c1="45" c2="46" r1="29" r2="29"/>
</region>
<region id="movingspace">
     <reg id="West"/>
     <reg id="Ngoro2"/>
     <reg id="NorthWest"/>
     <reg id="Mara"/>
     <reg id="East"/>
     <reg id="Center"/>
     <reg id="CenterWest"/>
</region>
```

The real uses of regions are in defining the associations of the rules and declaring initial configuration of membranes. The rule-membrane associations are defined as *associationlist* inside the element `<association>`*associationlist*`</associations>` as described in Section 3.1.6. If associated membranes to the rule are defined as a region, the rule can be associated to the region instead by the following element.

`<association><reg id="`*regid*`"/></associations>`

In Section 3.1.7, the configuration part declares the initial multisets in each non-empty membrane. In many cases some membranes have the same initial multisets. For such cases, instead of specifying the multisets in each membrane, the multiset can be specified inside the region of those membranes. Note that specified initial multiset will be replicated into those membranes. `<local r="`*r*`" c="`*c*`">` *multiset* `</local>`

3.2.3 Constant

A constant is declared by the following singleton.

`<constant id="`*constantid*`" val="`*value*`"/>`

constantid is a unique string composed by alphanumeric characters started by a letter. *value* can be either a literal number (real or integer) or a simple arithmetic expression that is formed by real numbers and/or integers and/or other constants (previously defined).

For example, constants `Mnp`, `Mng` and `Mnp2` are written as follows.

```
<constant id="Mnp" val="5"/>
<constant id="Mng" val="10"/>
<constant id="Mnp2" val="2*Mnp"/>
```

Note that `Mnp2` is defined by an expression containing other constant and the others are defined by literal values.

3.2.4 Constantlist

A constantlist defines several constants using a common prefix and suffixed differently by a sequence of numbers analogously as the index to an element of array. It is declared inside a pair or tags.

`<constantlist id="`*cname*`" type="`*type*`">` *listofconstantandvalues* `</constantlist>`

cname is a unique string composed by alphanumeric characters started by a letter, *type* can be either `txt` for textual values or `num` for numeric values. *listofconstantandvalues* contains elements and each one is written as singleton as follows `<mconst idx="`*index*`" val="`*val*`" />` for $index = a, a+1, a+2, ...$

For example, for constants `dc1=-1, dc2=0, dc3=4,...` and `bstages1="A0", bstages2="A0", bstages3="A0",...` they should be written as follows.

```
<constantlist id="dc" type="num">
    <mconst idx="1" val="-1" />
    <mconst idx="2" val="0" />
    <mconst idx="3" val="1" />
    <mconst idx="4" val="-1" />
    <mconst idx="5" val="1" />
    <mconst idx="6" val="-1" />
    <mconst idx="7" val="0" />
    <mconst idx="8" val="1" />
</constantlist>
<constantlist id="bstages" type="txt">
```

```
    <mconst idx="1" val="A0" />
    <mconst idx="2" val="A0" />
    <mconst idx="3" val="A0" />
    <mconst idx="4" val="A1" />
    <mconst idx="5" val="A2" />
    <mconst idx="6" val="A3" />
    <mconst idx="7" val="A4" />
    <mconst idx="8" val="A5" />
    <mconst idx="9" val="A6" />
</constantlist>
```

3.2.5 Variation

A variation defines a set of strings that can be used to vary the templates through substituting the stub in the template's identification by each string. The variation is defined as follows.

`<variation id="`*varname*`">`*listofstrings*`<variation/>`

where *vname* is a unique string composed by alphanumeric characters started by a letter, and *listofstrings* is a sequence of singleton elements of form:

`<string val="`*string1*`" />`

For example, variation `directions=`{`1,2,3,4,5,6,7,8`} should be written as follows.

```
<variation id="directions">
     <string val="1" />
     <string val="2" />
     <string val="3" />
     <string val="4" />
     <string val="5" />
     <string val="6" />
     <string val="7" />
     <string val="8" />
</variation>
```

3.2.6 Template of Rules

Several rules that have similar behaviour (components) can be defined as a template by varying it with one or more variables. Each rule takes on possible combination of values taken from variations. A template is written just like a rule: being defined inside the group of rules delimited by `transitionrules` tag and its attributes are expressed inside `rule` tag, and its components are expressed by `reactants`, `products`, `promoters`, `inhibitors` and `associations` tags. As a template, some letters (delimited by square brackets individually) representing variables are appended after *ruleid*. Each variable is described by

`<var varid="`*varid* `" varange="`*varange*`"/>`

inside the rule tag, where *vid* is the variable appended to *ruleid* and *varange* is its variation range id. Furthermore, a variable is written by a name delimited by square brackets `[`*S*`]`. All appearance of the variables in the body (attributes and/or components) will take the values defined in the variations when the rules are revealed from the template.

For example,

```
<rule id="DeathByNatural[S]"
   duration="0" rate="deathrate[S]/3">
     <var vid="S" vrange="stages"/>
     <description>Death because of natural causes,
         more health less death rate</description>
     <reactants><object id="A[S]"/></reactants>
     <products><object id="Ax"/></products>
     <promoters><object id="C" quant="Mnp"/></promoters>
     <inhibitors/>
     <associations><reg id="movingspace"/></associations>
</rule>
```

In the example above variable `S` is used. It takes the values of variation `stages` that is an example of Subsection Variation. The values are $\{0, 1, 2, 3, 4, 5, 6, 7, 8\}$ and each value will be taken to replace `[S]` on each appearance of it in the body for each generated rule. SInce there are 9 elements for `S` this template represents 9 rules that have id's are `DeathByNatural0`, `DeathByNatural1`,...,`DeathByNatural9`. For `S=5`, its id is `DeathByNatural5` and the atribute `rate` is revealed from simple arithmetic expression `"deathrate5/3"`. `deathrate5` should be a previously defined constant. Its reactant will be one object `A5`.

3.3 Scheduled Events

Scheduled events are basically defined by some event rules and set of time schedules when the event should be applied. An event rule is like an ordinary rule except it should specify a dummy object as the reactant to perform the reaction producing/consuming certain objects. Event rules should be placed altogether with ordinary rules/templates (inside `<transitionrules>... </transitionrules>`

For example,

```
<eventrule id="TempDwn1" rate="10">
    <description>Light temperature dropping</description>
    <reactants><object id="T"/></reactants>
</eventrule>
<eventrule id="TempUp1" rate="10">
    <description>Light temperature raising</description>
    <products><object id="T"/></products>
</eventrule>
```

A time schedule specifies the event rule *eventruleid* and the time point *timepoint* using the following singleton.

`<event ruleid="`*eventruleid*`" when="`*timepoint*`"/>`

For example,

```
<event ruleid="TempUp1" when="8"/>
<event ruleid="TempUp1" when="9"/>
<event ruleid="TempUp2" when="12"/>
<event ruleid="TempDwn2" when="13"/>
```

Note that, each event will be internally converted to be an adhoc rule that has deterministic time duration *timepoint* and will be reacted at time=0 (when the simulation is started). Then, this reaction at time=*timepoint* will produce the dummy object required by the event rule as the reactant.

3.4 Document Type Definition (DTD)

```
<!DOCTYPE gridsystem [
<!ELEMENT gridsystem (dimension,author,description,aliases,objectlist,transitionrules,co
<!ATTLIST gridsystem title CDATA #IMPLIED>

<!ELEMENT dimension EMPTY>
<!ATTLIST dimension
     NumCols CDATA #REQUIRED
     NumRows CDATA #REQUIRED>

<!ELEMENT author EMPTY>
<!ATTLIST author
     name CDATA #IMPLIED
     institution CDATA #IMPLIED>

<!ELEMENT description (#PCDATA)>

<!ELEMENT aliases ((variation)*|(constantlist)*|(constant)*|(region)*)*>

<!ELEMENT objectlist (objectdef)*>

<!ELEMENT transitionrules (rule)*>

<!ELEMENT variation (string)+>
<!ATTLIST variation id ID #REQUIRED>

<!ELEMENT constantlist (mconst)+>
<!ATTLIST constantlist
     type  (num|txt) #IMPLIED
     id ID #REQUIRED>

<!ELEMENT constant EMPTY>
<!ATTLIST constant
     id ID #REQUIRED
     val CDATA #REQUIRED>

<!ELEMENT region ((rect)+|(local)+|(reg)+)*>
<!ATTLIST region id ID #REQUIRED>

<!ELEMENT objectdef EMPTY>
<!ATTLIST objectdef
     id ID #REQUIRED
     desc CDATA #IMPLIED>

<!ELEMENT rule ((var)*,(description)?,reactants,products,promoters,inhibitors,associations
<!ATTLIST rule
     id CDATA #REQUIRED
     rate CDATA #IMPLIED
     duration CDATA #IMPLIED
     mode  (exp|const) #IMPLIED>


<!ELEMENT eventrule ((description)?,(reactants)?,(products)?)>
<!ATTLIST eventrule
     id CDATA #REQUIRED
     rate CDATA #IMPLIED>


<!ELEMENT var EMPTY>
<!ATTLIST var
     vid CDATA #REQUIRED
     vrange CDATA #REQUIRED>


<!ELEMENT reactants (object)+>

<!ELEMENT products (object)*>

<!ELEMENT promoters (object)*>

<!ELEMENT inhibitors (object)*>

<!ELEMENT associations ((local)*|(reg)*)>

<!ELEMENT configuration ((linkset)*|(reg)*|(local)*)*>

<!ELEMENT linkset (link)*>
<!ATTLIST linkset id ID #REQUIRED>


<!ELEMENT reg (object)*>
```

```
<!ATTLIST reg id CDATA #REQUIRED>


<!ELEMENT local (object)*>
<!ATTLIST local
     r CDATA #REQUIRED
     c CDATA #REQUIRED>


<!ELEMENT simulation (limits,(showprogress)?,displayobjects,animation)>

<!ELEMENT eventlist (event)*>

<!ELEMENT showprogress EMPTY>

<!ELEMENT event EMPTY>
<!ATTLIST event
     ruleid CDATA #REQUIRED
     when CDATA #REQUIRED>


<!ELEMENT object EMPTY>
<!ATTLIST object
     id CDATA #REQUIRED
     quant CDATA #IMPLIED
     r CDATA #IMPLIED
     c CDATA #IMPLIED
     dr CDATA #IMPLIED
     dc CDATA #IMPLIED
     loc CDATA #IMPLIED
     link CDATA #IMPLIED>


<!ELEMENT string EMPTY>
<!ATTLIST string val CDATA #REQUIRED>


<!ELEMENT mconst EMPTY>
<!ATTLIST mconst
     idx CDATA #REQUIRED
     val CDATA #REQUIRED>


<!ELEMENT rect EMPTY>
<!ATTLIST rect
     r1 CDATA #REQUIRED
     c1 CDATA #REQUIRED
     r2 CDATA #REQUIRED
     c2 CDATA #REQUIRED>


<!ELEMENT link EMPTY>
<!ATTLIST link
     fromrow CDATA #REQUIRED
     fromcol CDATA #REQUIRED
     torow CDATA #REQUIRED
     tocol CDATA #REQUIRED
     weight CDATA #REQUIRED>


<!ELEMENT limits EMPTY>
<!ATTLIST limits
     iterationmax CDATA #REQUIRED
     timemax CDATA #REQUIRED
     timeslice CDATA #REQUIRED>


<!ELEMENT displayobjects (dobject)*>
<!ATTLIST displayobjects
     bgcol CDATA #REQUIRED
     bytime CDATA #REQUIRED
     chartmax CDATA #REQUIRED
     step CDATA #REQUIRED
     gridcol CDATA #REQUIRED>


<!ELEMENT dobject EMPTY>
<!ATTLIST dobject
     id CDATA #REQUIRED
     col CDATA #REQUIRED
     loc CDATA #REQUIRED>


<!ELEMENT animation (vobject)*>
<!ELEMENT vobject EMPTY>
<!ATTLIST vobject
     id CDATA #REQUIRED
     max CDATA #REQUIRED
     color CDATA #REQUIRED
     format CDATA #REQUIRED>
]>
```

Chapter 4

Appendix A

(previous example should be revised according to the new DTD!)