MILIEU-M: VISUAL MANIPULATION AND PROGRAMMING FOR MULTI-MEMBRANES.

Rosario Lombardo, Vincenzo Manca

***Abstract**: In the context of membrane computing, the new notion of multi-membrane was introduced where the junction of membranes is used together with the inclusion of membranes. In multi-membranes a deterministic computation model can be defined for computing arithmetical functions by designing a sort of circuits in pure geometrical forms articulated at different distributed levels.*

In this article we present Milieu-M, a software tool for the visual manipulation of calculi on membranes and multi-membranes offering a rich and interactive Graphical User Interface. The visual programming approach is used to create topological forms entirely describing an algorithm in terms of the Pure Graphs formalism. The visual programming formalism and the textual multi-membrane programming language can be used jointly in Milieu-M to work on the same model and for seamlessly generating source code from the visual graphs and vice versa.

***Keywords**: Membrane Computing, MP-systems, Multi-Membranes, P-systems, Programming languages, Visual programming.*

***ACM Classification Keywords**: D.1.7 Visual Programming, D.1.m Miscellaneous – Natural computing, D.2.2 Design Tools and Techniques – User interfaces, H.5.2 User Interfaces – User-centered design.*

Introduction

Membrane computing is a branch of natural computing that investigates and abstracts computing ideas and models from the structure and functioning of living cells [Păun 2000]. P systems are the computational membrane structures containing multisets of objects and where evolution rules are applied non-deterministically and in a maximally parallel way [Păun 2002]. Non-deterministic evolution strategies allow P systems to obtain different successful computations and are thus very suited in generating or recognizing languages, depending on the way their output is defined.

Metabolic P systems, based on Gh. Păun's membrane computing theoretical framework, were introduced to represent metabolic processes in a discrete mathematical setting. MP systems [Manca 2009] are deterministic single-membrane dynamical systems where the multiset rewriting rules are regulated by functions depending on the state of the system. The encouraging results obtained by applying MP systems to the modeling of biological phenomena and to function approximation suggested the idea of defining a novel model of computation which is deterministic and based on rules regulated by fluxes. The deterministic model of computation inspired to MP systems is developed for a new type of membrane structure called *Multi-Membranes* where calculi are essentially carried out by matter transferals [Manca, Lombardo 2011].

Since the P systems were presented, several variants of the model were defined and many software applications have been used in many computational and applicative contexts [Ciobanu, Păun, Pérez-Jiménez, (Eds.) 2006]. Recently the P-Lingua programming language was introduced to cover a variety of non-deterministic models [Díaz-Perniland, Pérez-Hurtado, Pérez-Jiménez, Riscos-Núñez 2009], but still in a large number of software applications available to the community, there is the lack of important user-centric aspects regarding the ease of use and the graphical user interface (GUI) that could potentially open up the membrane community to a wider audience (for a review of available simulators and applications see [Gutíerrez–Naranjo, Pérez-Jiménez, Riscos–Núñez 2006][P systems page]).

In this paper we present Milieu-M, a software application intended to be a tool for the visual manipulation of calculi defined on membrane structures and for displaying non-terminating dynamics. In Milieu-M the human-machine interaction aspects were taken into account since the early stages of its engineering. The design of a GUI that had to be user-friendly and effective at the same time was possible by implementing the *Pure Graphs* formalism, a language for describing multi-membranes that is formal but also completely visual [Manca, Lombardo 2011].

For the following discussion, and to keep the document self-contained, it is useful to briefly recall some dynamical aspects of the Metabolic P systems used in the computation model behind the Multi-membranes. MP Systems are deterministic dynamical systems where the rules are interpreted as reactions specifying variations of reactants and products. A multiset rule such as $2a+b \rightarrow c$ means that a number $2 \square n$ of molecules of kind a and a number n of molecules b are replaced by n molecules of type c. The value of n is the *flux* of the rule application and is provided by a function called *regulator* of the reaction. Regulators take as arguments the state of the system, that is, substance quantities (or physical parameters such as temperature, pressure) and provides the fluxes that the reactions have to apply. The dynamics of an MP system is deterministic at population level and, at each step, is governed by a partition of matter determined by the fluxes of the rules consuming it.

Example. An MP system is specified entirely by an *MP grammar*, where reactions are given with the corresponding regulators. The empty multiset $\square$ on the left side (resp. right side) of the rule is used to specify introduction (resp. expulsion) of matter.

$r_1 : \square \rightarrow x$	$\varphi_1 = 2$	(φ_1 is constant)
$r_2 : x \rightarrow y$	$\varphi_2 = 2y + x$	(φ_2 depends on populations x and y)
$r_3 : y \rightarrow \square$	$\varphi_3 = 1$	(φ_3 is constant)

Consider the system at some time steps $0, 1, 2, \ldots, t$ and consider the substance x that is produced by rules r_1, r_3 and is consumed by rule r_2. If $X[i]$ is the global state of the system at step i, then $u_1[i] = \varphi_1(X[i]),\ u_2[i] = \varphi_2(X[i]),\ u_3[i] = \varphi_3(X[i])$ are the fluxes of the rules r_1, r_2, r_3 respectively. Therefore in the transition from step i to step $i+1$, the variation of substance x is given by $x[i+1] = x[i] + u_1[i] - u_2[i] + u_3[i]$ or, more explicitly written as $x[i+1] = x[i] + 2 - (2y[i] + x[i]) + 1$.

Multi-Membranes

The notion of Multi-Membranes [Manca, Lombardo 2011] is presented along with the associated deterministic computation model that allows the combination of computations articulated at different distributed levels. In multi-membranes, together to the usual notion of *inclusion* among membranes, is ascertained the new *junction* relation among membranes, which holds when two membranes share a portion of their borders (they are joined). An *elementary multi-membrane* is obtained by joining many elementary membranes. Starting from elementary multi-membranes, by iterating membrane inclusion and junction, general multi-membranes can be obtained (Figure 1).

An elementary multi-membrane has a *central structure* if there is one central elementary membrane, to which are joined two or more elementary membranes, called the *frontier membranes*. A general multi-membrane with central structure is obtained, starting from an elementary multi-membrane with central structure, by means of hierarchical inclusions, into the central components, of membranes or multi-membranes having central structure as in Figure 2.

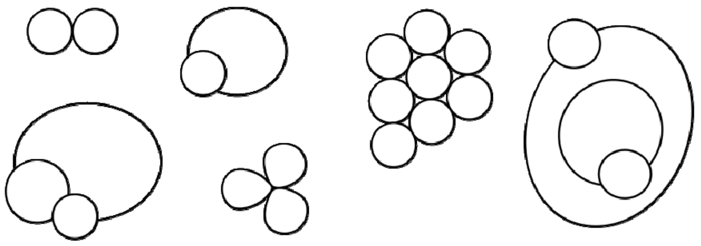


Figure 1: General multi-membranes obtained by iterating inclusions and junctions operations.

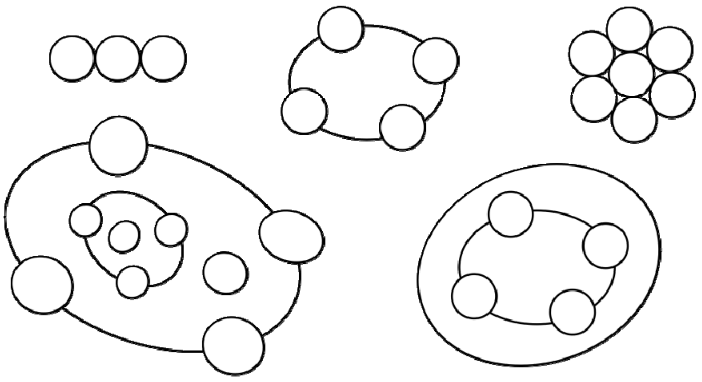


Figure 2: Multi-membranes with central structure.

Multi-membranes with central structure are used for defining a notion of deterministic computation, which is essentially based on the transfer of objects among membranes. A membrane is reachable from a region if a portion of its border confines with the region. In a region from which two membranes are reachable a *channel* can be put between them, which goes from one of them, the source, to the other, the target. Along a channel, objects contained in the source membrane can be transferred to the target membrane. The important aspect of a multi-membrane with central structure is the role of the *frontier* membranes which are joined to the central one. In fact they are reachable from the region within the central membrane but also from the points which are external to the central membrane and to other frontier membranes (see Figure 3).

In a multi-membrane, a (transfer) rule is a channel between two membranes. We denote a rule in the following way, where a, b are membrane labels and φ is the flux, that is, an expression that assumes a value in correspondence to the values of its variables:

$$a \rightarrow b \ \# \varphi \tag{1}$$

The effect of such a rule is the passage of φ objects from the membrane with label a to the membrane with label b. In this paper we consider the function φ to be either a natural number or the label of some membrane. In the latter case the value of the flux is then the number of objects inside the region of that membrane.

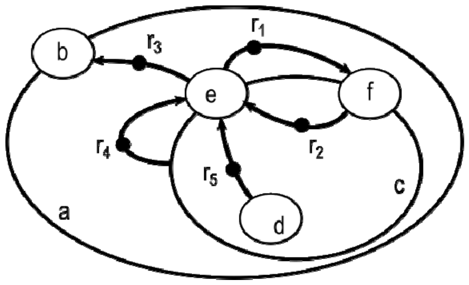


Figure 3: A multi-membrane structure with channels representing rules for matter transferal.

Table 1: Graphical elements denoting the frontier membranes in a multi-membrane computation system.

Graphical symbol	Description	
⟲	IN	
⟳	OUT	
⟲⟳	IN-OUT	
◎	START	
⊙	HALT	
○•→○	Rule	$a \to b \ \# \varphi$
→▷	Expulsion rule	$a \to \emptyset \ \# \varphi$
▷→	Introduction rule	$\emptyset \to b \ \# \varphi$

Definition 1. A Multi-Membrane Computation System with central structure is a construct

$$M = (L, R, \mu)$$

where (see also Table 1):

L is a set of membrane

R is a set of multi-membrane rules having the following forms with $a, b \in L$ and $\varphi \in N$ or $\varphi \in L$:

trans: $a \to b \quad \# \varphi$

extra-in: $\emptyset \to a \quad \# \varphi$

extra-out: $a \to \emptyset \quad \# \varphi$

μ is the initial multi-membrane configuration with only one type of objects, that is, a multi-membrane with central structure where some initial objects are inside certain membranes (by default assumed empty).

In each multi-membrane some frontier membranes are marked by:

START	one and only one membrane is marked by START;
HALT	one and only one is marked by HALT;
IN, OUT	some membranes are marked by IN and some with OUT, but these marks may not occur.

Rules in the system are applied according to the following general principle. The rules sharing a common source membrane are simultaneously applied if all their fluxes are *defined* and if the objects that they would globally extract from the source are less than the number of objects inside the source membrane. If this is not the case, all the rules sharing a common source cannot be applied. Moreover, a flux given by the content of a frontier membrane is *defined* only when the multi-membrane of this frontier membrane has one object in its Halt membrane.

The semantics of computation of a multi-membrane system is given in the following definition.

Definition 2. (MM-computable function) A function $f : N^k \to N^h$ such that $f(x_1, \ldots, x_k) = (y_1, \ldots, y_h)$ is MM-computable if there exist a multi-membrane System M with central structure where:

- Frontier membranes marked by IN contain the values $x_1, \ldots, x_k$ respectively.
- The contents of frontier membranes START and HALT are 0 and the system is setup with its initial configuration.
- Provided that START is only referred as target from outer region and only as source from inner region (and reversed for HALT), after posing one object in the membrane marked START (as indication of "computation in progress") the rules of the system get applied according to Definition 1. The system M eventually ends when concomitantly START is emptied and HALT receives one object (as indication of

"end of computation"). In this case, the values $y_1, \ldots, y_h$ which are in the frontier membranes marked by Out provide the results of the computation, and all internal rules become inapplicable.

- Moreover, the HALT membrane never contains an object *iff* $f(x_1, \ldots, x_k)$ is undefined.

Representation formalisms for Multi-membranes

A large number of membrane structure models are customarily represented textually with variants of the boundary rule notation [Bernardini, Manca 2003], which naturally captures the idea that reactions take place on the inner membranes of a cell, maybe depending on the contents of both the inner and outer region adjacent to that membrane.

Bracketed notation, a variant of the boundary notation, provides a complete textual specification for Multi-membranes and constitutes the specification for the textual multi-membrane programming language. In bracketed notation the rules are syntactically localized in a multi-compartmental configuration μ that has been extended with the definition of the frontier membranes (Figure 4, at the top), that is, the labelled membranes (of types START, HALT, IN, OUT) that appear after the at-symbol @ and are joined to the central membrane.

In *Annotated graphs* (at the bottom in Figure 4), the graphical elements of the multi-membranes from Table 1 are accompanied by textual annotations denoting the membrane labels and flux definitions. The correspondence of elements from the annotated graphs and the bracketed notation is depicted in Figure 4.

The third equivalent formalism is defined by the *pure multi-membrane graphs* that completely express multi-membrane algorithms using a pure visual formalism where there is no single textual representation. Indeed a flux defined in terms of another membrane (e.g. $\varphi = \#m$, if m is the label of a membrane) is denoted by a dotted line connecting the membrane m to the rule, while a flux defined by a constant number (e.g. $\varphi = \#3$) is replaced by a reference to an isolated membrane containing the same number of objects (e.g. with $\varphi = \#t$ and $[_t 3]_t$). The pure graph defined in this manner is a topological form that entirely describes an algorithm in terms of a pure visual formalism.

Milieu-M, introduced in the next section, implements the pure graphs as a visual programming language for multi-membranes.

Visual programming with Milieu-M

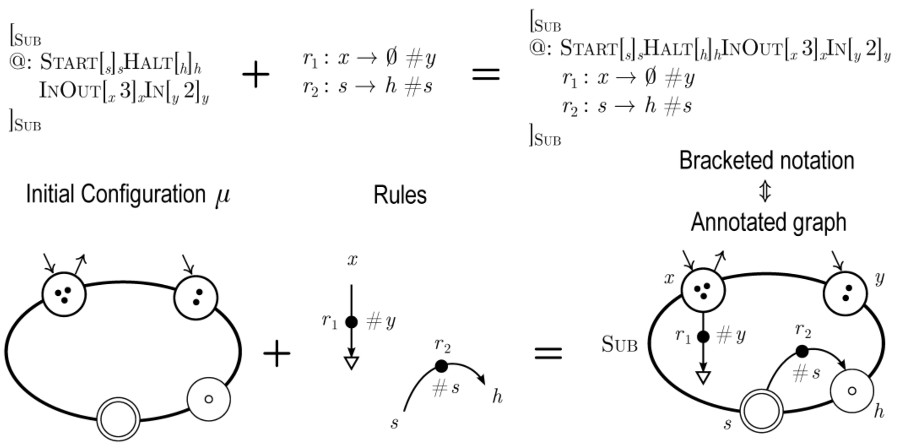
Milieu-M is an Integrated Development Editor (IDE) written in Java and following a user-centered design. Its functionalities are bundled in one single cross-platform installer making easy the installation at the first try (no development or additional libraries are required apart the installer itself). The application implements most of modern compiler's techniques [Aho, Sethi, Ullman 1986], by using the ANTLR[1] v3 parser generator for high performances source code parsing, and the SWT[2] widget toolkit for ensuring consistent high-level look-and-feel across different platforms for the graphical user interface. The IDE offers syntax-highlighting text-editors with undo-history and the GUI is organized into convenient tabs allowing to work on several, possibly connected, documents at once (see Figure 5). In the following an overview of Milieu-M is described while the complete details of functioning and user interface will be published shortly in the user manual.

At the time of writing Milieu-M supports two *dialects*, one for the Multi-Membranes and the other for the LAMP framework where the early investigation results [Lombardo, Manca 2011] of the computational MP approach

[1] ANTLR, ANother Tool for Language Recognition, version 3. http://www.antlr.org/

[2] The Standard Widget Toolkit, http://www.eclipse.org/swt/

where developed in terms of substance transformations rather than matter exchange among membranes. In this regard the architecture has been designed in the attempt to be as easier as possible to integrate with other frameworks or membrane models whenever their source code is available, but this has to be actually tested on the field in further steps.

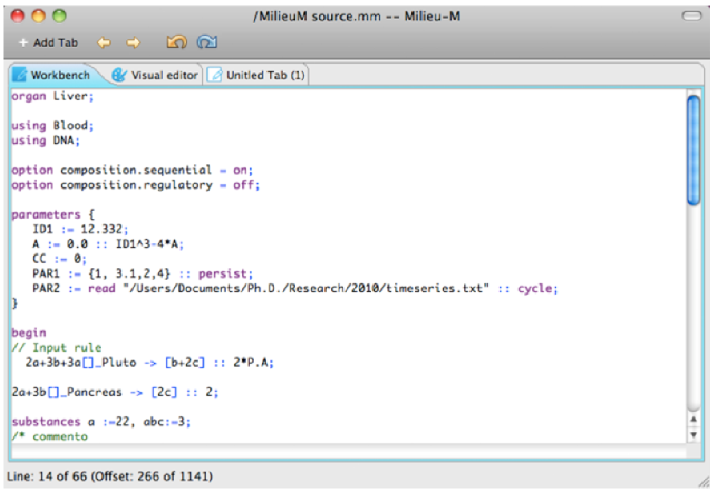


$[_{\textsc{Sub}}$ @: $\textsc{Start}[_s]_s\textsc{Halt}[_h]_h$ $\textsc{InOut}[_x 3]_x\textsc{In}[_y 2]_y$ $]_{\textsc{Sub}}$
$+$
$r_1 : x \to \emptyset \;\#y$
$r_2 : s \to h \;\#s$
$=$
$[_{\textsc{Sub}}$ @: $\textsc{Start}[_s]_s\textsc{Halt}[_h]_h\textsc{InOut}[_x 3]_x\textsc{In}[_y 2]_y$ $r_1 : x \to \emptyset \;\#y$ $r_2 : s \to h \;\#s$ $]_{\textsc{Sub}}$
Initial Configuration μ
Rules
Bracketed notation ⇕ Annotated graph
x
r_1
$\#y$
r_2
$\#s$
s
h
SUB
y

Figure 4: Bracketed notation combines the initial configuration μ with the rules of the system. Frontier membranes x and y (prefixed by the @ symbol in bracket notation) initially contain 3 and 2 objects respectively, while rules r_1 and r_2 are localized in membrane SUB*, that computes the limited subtraction $x - y$, if $x > y$ and x otherwise.*

According to *Definitions 1* and 2, the general concept of sub-routine calling in multi-membranes is not a procedural call but can be interpreted as a *regulatory composition* that regulates the reaction rate of a rule by using the computational OUTPUT of an adjacent system. Therefore the deterministic model of computation built-in in Milieu-M is compositional because multi-membrane systems can be connected together via composition.

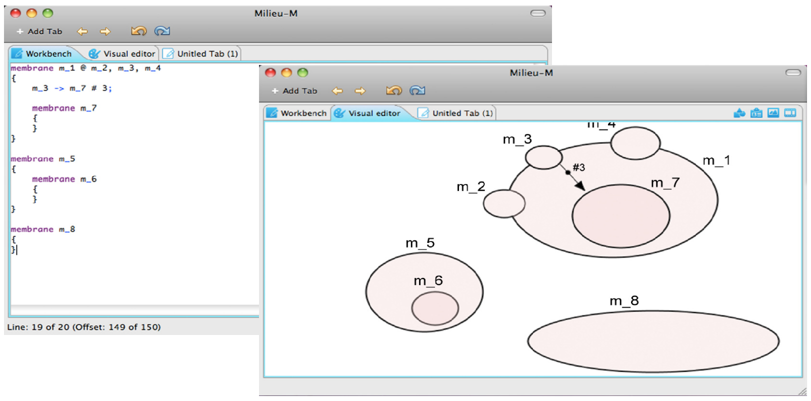
The application provides two predefined non-closable tabs: the “Workbench” tab is the textual area where to start typing source code while the “Visual editor” tab is used for interactive visual programming and animations. An important aspect of the two predefined tabs is that they represent the same model with different, but equivalent, formalisms. In fact a first effort has been made to keep the textual representation in the Workbench synchronized with the graphical representation in the Visual editor. One single model is managed by the application in the internal data structures allowing the user to seamlessly change “perspective” from the visual representation to the textual one and vice versa as shown in Figure 6, where Milieu-M is interactively generating source code from the visual program in foreground.



```
organ Liver;

using Blood;
using DNA;

option composition.sequential = on;
option composition.regulatory = off;

parameters {
   ID1 := 12.332;
   A := 0.0 :: ID1^3-4*A;
   CC := 0;
   PAR1 := {1, 3.1,2,4} :: persist;
   PAR2 := read "/Users/Documents/Ph.D./Research/2010/timeseries.txt" :: cycle;
}

begin
// Input rule
  2a+3b+3a[]_Pluto -> [b+2c] :: 2*P.A;

2a+3b[]_Pancreas -> [2c] :: 2;

substances a :=22, abc:=3;
/* commento
```

Line: 14 of 66 (Offset: 266 of 1141)

Figure 5: Milieu-M as appearing on a Mac. The tabbed GUI allows working on several syntax-highlighted tabs at once. The "Visual Editor" tab is devoted to the visual manipulation of calculi on membranes.



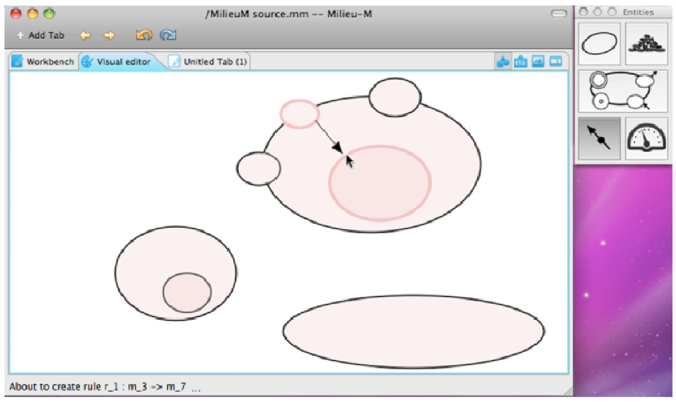
```
membrane m_1 @ m_2, m_3, m_4
{
    m_3 -> m_7 # 3;

    membrane m_7
    {
    }
}

membrane m_5
{
    membrane m_6
    {
    }
}

membrane m_8
{
}|
```

Line: 19 of 20 (Offset: 149 of 150)

Figure 6: Two screenshots of the same application instance. In foreground there is the "Visual editor" tab where a simple model was created. Milieu-M interactively generates source code in the "Workbench" tab as the newly graphical elements are designed.

When building a model graphically, the Visual Editor applies real-time checks to user's actions ensuring that the interactive creation and manipulation is made only on valid models stored in the internal representation. Starting from the source code the process can not be interactive, but first the source has to be correctly parsed and "translated" into the same internal data structures. From the internal representation is possible then to generate the graphical layout or the textual source back.

The Visual editor has its own toolbar as can be seen in the right-most part of the tab container in Figure 7. From left to right, each of the four buttons opens a different toolbox: 1) Entities toolbox: containing the available entities for a given visual language. 2) Properties toolbox: shows table-like window where the properties of selected entities in Visual editor can be displayed and modified. 3) Visualization toolbox: contains commands for changing the view of the model like zoom, pan, etc. 4) Animation toolbox: provides the basic actions to start and control animated simulations of the models.



About to create rule r_1 : m_3 -> m_7 ...

Figure 7: In the Visual Editor tab the user interactively designs membrane structures – which are visual algorithms – by drag-and-drop operations: when "drawing" a rule the involved membranes gets highlighted. The status bar on the bottom provides interactive information to the user about his actions.

Visual Programming

The founding principles of the visual approach are the formalness, user-friendliness and correctness. The visual language is formal because is based on the Visual Graphs (or Annotated Graphs when the appropriate textual labels are shown). Additionally it has to be user-friendly but no visual model can be wrong if it was created with the visual formal language. To this aim every user action is interactively managed by the application, which makes real-time checks on the drawn elements before they can be created in the internal data structures and automatically provides each newly created entity with a progressive name (m_1, m_2... for membranes, r_1, r_2… for rules etc.).

Visual programming allows to interactively “draw” complex membrane structures with drag-and-drop operations from the Entities toolbox (the floating window on the right side of Figure 7). The basic entities in the toolbox are i) membranes, ii) substances iii) rules and iv) parameters. Additionally a multi-membrane button is shown, in the middle of the toolbox, allowing the guided creation of multi-membrane templates with central structure. The process of creating a membrane consists in picking the membrane tool from the toolbox, and then drawing them on the Visual Editor. General multi-membranes can be created by drawing new membranes upon the existing ones interactively: membrane operations of inclusion and junction are interactively supported when drawing new membranes inside or upon other membrane borders allowing to create morphological forms of multi-membranes in 2D.

Analogously, after selecting the rule button from the toolbox, it is sufficient for the user to start dragging from the border of a membrane to pull an arrow from the border outward to the mouse pointer. At the same time the selected membrane is highlighted as source membrane. Moving the mouse pointer around, other membranes are highlighted as possible targets when passing the arrow tip over their borders (Figure 7). The regulator for a rule can be created by a double click on the rule that allows setting an expression or a membrane label as the regulator definition. The same result can be obtained by visual programming. After selecting the rule tool from the toolbox, dragging the mouse pointer from the border of a membrane a new rule arrow is spawn as before, but this time it will turn into a dashed arrow (regulator) when moving upon another rule. Releasing the mouse the regulator of the rule will be defined as the contents of the membrane. Once multi-membranes and rules have been defined it is possible to add matter by clicking on a membrane region with the appropriate substance tool.

Even more interactive checks are made on the theoretical definitions of multi-membranes being drawn providing simultaneous visual clues and textual suggestions in the status bar. In multi-membranes the checks include, but is not limited to, the number and quality of frontier membranes (eg.: if there is a START there must be also an HALT) or selectable target membranes (eg.: no START membrane can be used as source membrane for a rule going out of the central membrane).

Simulations and Animations

Simulations of the deterministic model are carried out applying the rules with the corresponding fluxes, as illustrated in the example at page 2, and according to the Definitions 1 and 2. The evolution of the system dynamics can be visually followed on screen during its progress using the Animation toolbox where are available commands to play, pause, step forth and tune the animation speed. At each step the active entities are highlighted making visible the source/target membranes across which matter is moved. Each simulation creates time series for all involved membranes and substances that can be graphically plotted alone or in combination as charts or phase diagrams.

Conclusion

In this paper we introduce Milieu-M, a software tool for the visual manipulation of calculi on multi-membranes, a recent membrane structure providing mechanisms for combining deterministic computations articulated at different distributed levels. In Milieu-M the multi-membrane operations of junction and inclusion of membranes are carried out graphically by *visual programming* in the Visual editor where the real-time checks performed by the application ensure to build valid models. The visual algorithm can be graphically executed producing time series that are plotted in charts or phase diagrams. Additional peculiarity is that the membrane model can be created and managed coherently both in visual programming and source code with the ability to generate transparently one from the other.

Topics of future improvement include i) supporting and integrating with additional membrane models and their textual languages in Visual programming mode and ii) enhancements in the automatic visual layout routines starting form the source code.

Bibliography

[Aho, Sethi, Ullman, 1986] A. Aho, R. Sethi, and J. Ullman. Compilers: principles, techniques, and tools. Reading, MA,, 1986.

[Bernardini, Manca, 2003] F. Bernardini, V. Manca. Dynamical aspects of P systems. *Biosystems*, 70(2):85 – 93, 2003. Cell Computing.

[Ciobanu, Păun, Pérez-Jiménez, (Eds.), 2006] G. Ciobanu, Gh. Păun, M. Pérez-Jiménez, (Eds.). Applications of Membrane Computing, Springer Verlag. *Natural Computing Series*, Berlin, October, 2006.

[Díaz-Perniland, Pérez-Hurtado, Pérez-Jiménez, Riscos-Núñez, 2009] D. Díaz-Perniland, I. Pérez-Hurtado, M. Pérez-Jiménez, A. Riscos-Núñez. A P-Lingua programming environment for Membrane Computing. *Membrane Computing*,

[Freund, Verlan, 2007] R. Freund and S. Verlan. A Formal Framework for Static (Tissue) P Systems. Membrane Computing, *WMC 2007. LNCS, vol. 4860*, pp. 271–284. Springer, Heidelberg, 2007.

[Gutíerrez–Naranjo, Pérez-Jiménez, Riscos–Núñez, 2006] M.A. Gutíerrez–Naranjo, M.J. Pérez-Jiménez, A. Riscos–Núñez, Available membrane computing software. In: [Ciobanu, Păun, Pérez-Jiménez, (Eds.) 2006], pp. 411–436. (2006).

[Lombardo, Manca, 2011] R. Lombardo and V. Manca. Arithmetical Metabolic P Systems. *Foundations on Natural and Artificial Computation, LNCS 6686*. J. Ferràndez at al. (Eds.). Springer Berlin / Heidelberg, 2011.

[Manca, 2009] V. Manca. Fundamentals of Metabolic P Systems. *The Oxford Handbook of Membrane Computing, Ch. 19*. Gh. Păun, G.Rozenberg, A. Salomaa (Eds.). Oxford University Press, 2009.

[Manca, Lombardo, 2011] Computing with Multi-Membranes. *12th Conference on Membrane Computing, LNCS* (in press).

[Păun, 2000] Gh. Păun. Computing with Membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.

[Păun, 2002] Gh. Păun. Membrane Computing. In *Fundamentals of Computation Theory*, volume 2751 of LNCS, pages 177–220. Springer Berlin / Heidelberg, 2002.

[P systems page] P systems web page, http://ppage.psystems.eu/

Authors' Information

Vincenzo Manca *– Full Professor, Department of Computer Science, University of Verona, Strada Le Grazie 15, Ca Vignal 2, I-37174 Verona, Italy; e-mail: vincenzo.manca@univr.it*
Major Fields of Scientific Research: Natural Computing - Molecular, DNA, Membrane, and Cellular Computing; Unconventional Computation Models and Formal Language Theory.

Rosario Lombardo *Ph.D. Student, Department of Computer Science, University of Verona, Strada Le Grazie 15, Ca Vignal 2, I-37174 Verona, Italy; e-mail: rosario.lombardo@univr.it*
Major Fields of Scientific Research: Computing Methodologies – Simulation and Modeling; Unconventional Computation Models.