Chapter 35
Software for Membrane Computing

Andrei George Florea
Politehnica University of Bucharest, Romania

Cătălin Buiu
Politehnica University of Bucharest, Romania

ABSTRACT

In order to use membrane computing models for real life applications there is a real need for software that can read a model from some form of input media and afterwards execute it according to the execution rules that are specified in the definition of the model. Another requirement of this software application is for it to be capable of interfacing the computing model with the real world. This chapter discusses how this problem was solved along the years by various researchers around the world. After presenting notable examples from the literature, the discussion continues with a detailed presentation of three membrane computing simulators that have been developed by the authors at the Laboratory of Natural Computing and Robotics at the Politehnica University of Bucharest, Romania.

INTRODUCTION

Membrane computing has attracted a notable interest from the scientific community, both in the theoretical foundations and in the applications in multiple domains. Shortly after the introduction of P systems as a distributed and parallel computing model, P systems simulators have been developed, tested, and made available to the membrane computing community. The purpose of this chapter is to give an overview of existing membrane computing simulators and to discuss their capabilities. Included in this discussion are detailed presentations of the simulators we have designed: *SNUPS* (for simulating standard and enzymatic numerical P systems), *Lulu* (for simulating P colonies and P swarms) and *PeP* (an on-going open-source project to develop a standard/enzymatic numerical P system simulator). These membrane computing simulators required tackling several design and implementation challenges that are described

DOI: 10.4018/978-1-7998-1754-3.ch035

in *Simulator Design and Implementation Challenges*, found within this chapter. The understanding of this chapter is based upon the assimilation of the fundamental principles of membrane computing presented in Chapter 2.

MEMBRANE COMPUTING SIMULATORS

In general, the existing software for simulating membrane computing models can be divided into three categories taking into account the paradigm they use: (1) sequential (C, Java, C++, etc.), (2) software-based parallelization (of which some of the best known are *Open MPI* and *OpenMP*), and (3) hardware-based parallelization (*FPGAs*, etc.). *Graphics Processing Units* (GPUs) follow a hybrid paradigm and offer a many-core platform with high parallelism at low cost (Martínez-del-Amor, Macías-Ramos, Valencia-Cabrera, Riscos-Núñez, & Pérez-Jiménez, 2014). Our discussion and presentation will follow this categorization.

One of the earliest works in the area of simulating P systems is reported in (Ciobanu & Paraschiv, 2002) and describes the *Membrane Simulator* which provides a graphical simulation for two basic P systems (the hierarchical cell system and the active membrane system).

An ANSI C library with simple data structures that facilitated the *in silico* study of P systems was proposed in (Nicolau Jr, Solana, Fulga, & Nicolau, 2002). There can be implemented both active and non-active membranes, and actions for dissolving, dividing and creating new membranes.

(Borrego et al., 2007) continued the work dedicated to the graphical simulation of P systems and presented a tool called *Tissue Simulato*r which supports the understanding of the basic structure and functioning of tissue P systems with cell division. This tool was developed in Java (to analyze the input data) and C# (used for the graphical user interface and for the kernel of the application). This software is no longer available to the community. A software tool for assisting the formal verification of spiking neural P systems (*SNPS*) has been proposed in (Gutiérrez-Naranjo, Pérez-Jiménez, & Ramírez-Martínez, 2008).

The membrane computing paradigm lies at the basis of *Cyto-Sim* (Sedwards & Mazza, 2007) which included a formal language and also a Java stochastic simulator of membrane systems. *Cyto-Sim* allowed the use of Petri nets based models and was able to import and export SBML files and to export MATLAB files. In (Spicher, Michel, Cieslak, Giavitto, & Prusinkiewicz, 2008) there is a report on an implementation of stochastic P systems using *MGS*, a spatially explicit programming language.

Psim, a simulation tool based on metabolic algorithms, was presented and discussed in (Bianco & Castellini, 2007), allowing the simulation of metabolic P systems. *MetaPlab* is an interesting development and presents itself in the form of a virtual laboratory implemented in Java, available at http://mplab.sci.univr.it/ and allowing the understanding and simulation of the internal mechanisms of biological systems (Castellini & Manca, 2008).

SNUPS, a Java software tool for modelling and simulation of standard numerical P systems and of enzymatic numerical P systems (Arsene, Buiu, & Popescu, 2011) is presented together with a detailed example in (Buiu, Arsene, Cipu, & Patrascu, 2011). This application software allows the development of a wide range of applications from modelling and simulation of ordinary differential equations, to the design and simulation of computational blocks for cognitive architectures and of membrane controllers for autonomous mobile robots. In the case of enzymatic numerical P systems, *SNUPS* simulates