Individual memory about the 14^{th} Brainstorming Week on Membrane Computing

Ariadna Ribes Metidieri

Universitat de Barcelona
Email: aribesmetidieri@gmail.com

Summary. The main objective of this memory is to stand out one of the research methods for developing new P system models observed during the 14^{th} Brainstorming Week on Membrane Computing. Firstly, a general overview of P systems is provided. To continue, the use of register machines in order to justify completeness and universality is justified. And to end up, an example of the method is provided.

1 Motivation and experience

The motivation for investigating further into this subject arose when observing that new computational model could be proposed. However, computability (the model's capability of acting as a computer) was always required, meaning that each new proposed model was tested versus Turing completeness. i.e., every proposed model should be demostrated to be capable of performing a computation.

The proof of the universality of the P systems can be attained using different methods, but the utilization of the Register Machines was widely promoted. Moreover, through '*Rudi's fancy homework*' we learned that register machines were in fact, simple but really useful interesting devices.

2 General Overview

Membrane computing is a biologically-inspired research branch in the field of computer science which starts from the assumption that processes taking place in the structure of a living cell can be interpreted as a computation and it gathers the study of different kinds of P systems. P systems are the devices used in this new computing paradigm which performs calculations based on the idea of a hierarchical arrangement of membranes acting as channels of communication. These systems are inspired in cellular structures, being the cell-like, tissue-like and Spiking Neural P systems current developed models [1]. All three models are based on cells, but seems important to recall that they are formal models which should not be considered as representations of the truth.

Membrane computer models share the same structure composed by membranes, objects, catalysts and a multiset of rules (i.e, evolution, communication, dissolution or division rules)[2] which are applied on the objects in each region delimited by a membrane. The computation works from an initial starting state or configuration to an end state through a number of discrete steps or transitions between configurations. The evolution rules are used in a non-deterministic and maximally parallelism way, i.e., in any computational step of the P system Π, a multiset of rules from the sets R_1,...,R_m is chosen in a non-deterministic way such that no further rule can be added to it. The obtained multiset will still be applicable to the existing objects in the membrane regions 1, . . . ,m. When no more rules can be applied, the computation ends (it is said to halt), leaving the result of the process in a given membrane or in the environment.[3]

Membrane computing was first developed in order to solve NP-complete problems. The research in this field moves in two different directions. On the one hand, theoretical models are being developed. This branch of research tries to find a theoretical foundation for new P system models and works in computational complexity, which tries to find an efficient solution to hard problems and works on the P conjecture. On the other hand, a practical approach is postured, including simulations in silico (for example, using MeCoSim)[4] as well as research in order to implement P systems in vitro.

3 Register machines as reference model for computational completeness and universality

Most P system variants (such as purely catalytic P systems, extended Spiking Neural P systems, P systems with anti-matter...) can be demonstrated to be computationally universal or Turing complete, i.e, the system of data-manipulation rules can be used to simulate a single-taped Turing machine.

A Turing machine is a hypothetical device with an infinite memory capacity, which manipulates symbols on a supposedly infinite strip of tape according to a set of rules. The Church-Turing thesis conjectures that any function whose values can be computed by an algorithm can be computed by a Turing machine, and therefore that any real computer is equivalent to a Turing machine.

The register machines are known to be computationally complete and equal in power to (non-deterministic) Turing machines. Consequently, register machines provide a simple universal computational model, which can be used to provide the proofs of the computational completeness of P systems based on the simulation of this kind of machines.

Formally, a register machine is a tuple $M = (m, B, l_0, l_h, P)$, where m is the number of registers, b is the set of labels, $l_0 \in B$ is the initial label, $l_h \in B$ is the final label and P is the set of instructions bijectivly labeled by elements of B. The instructions of M can be of the following forms:

- l_1: $(ADD(j), l_2, l_3)$ with $l_1 \in B \backslash \{l_h\}, l_2, l_3 \in B, 1 \leq j \leq m$.
  Increases the value of the register j by one, followed by a non-deterministic jump to instructions l_2 or l_3. This instruction is usually called *increment*.
- l_1: $(SUB(j), l_2, l_3)$ with $l_1 \in B \backslash \{l_h\}, l_2, l_3 \in B, 1 \leq j \leq m$.
  If the value of the register j is 0 then jumps to l_3 (instruction called *zero-test*), otherwise the value of the register j is decreased by one, followed by a jump to instruction l_2 (*decrement*).

- l_2: HALT: stops the execution of the register machine.

A specific model of a P system should be called computationally complete or universal if for any (generating, accepting, computing) register machine M we can effectively construct an equivalent P system Π of that type simulating each step of M in a bounded number of steps and yielding the same result.[5]

Once a new P system model has been proposed, the main goal to achieve is to determine that effectively it can perform all the calculations computable by a real computer and not just the operation it was first thought to perform.

The rule complexity of universal P systems depends on the objects as well as on the specific types of rules.

3.1 Example: SN P systems with States

Let's consider a particular SN P system with states and a single neuron stP_1 Π. It's formal definition is given by

$$\Pi = (1, O = O_T = \{a\}^*, Q = B, \delta, f_I, f_O, q_i = l_0, F = l_h, C_i = 0) \quad (1)$$

The stP_1 starts with the initial configuration computed by the input function f_I, the initial state $q_i = l_0$ and the input object $a \in O$, which are equal to the set of terminal objects O_T. The transitions between configurations and states are computed by δ to the new ones until the computation reaches a final state $f = l_h \in F$.

The computations of the register machine $M = (m, B, l_0, l_h, P)$ can be simulated by the stP_1 Π working with multisets as follows (the states of a single neuron represent the instruction labels of the register machine) [6]

$$\delta(p, (w)) = \{(\{q, s\}, \{(a \to a^{p_r}, maxpar)\})\} \quad (2)$$

for p: (ADD(r),q,s) ∈P, w∈ $\{a\}^*$

$$\delta(p, (w)) = \{(q, \{(a^{p_r} \to a, maxpar)\})\} \quad (3)$$

for p: (SUB(r),q,s)∈P, $p_r/|w|$

$$\delta(p, (w)) = \{(s, 0)\} \quad (4)$$

for p: (SUB(r),q,s) ∈ P, not $p_r/|w|$

To sum up, as can be seen from the example above, a SN P system acting in the maximally parallel derivation mode (*maxpar*) is in fact computationally complete, as the rules which define the system can be simulated using a Turing machine.

References

1. Gh. Păun *Membrane Computing: An Introduction*
2. Claudio Zandron, Alberto Leporati, Luca Manzoni, Giancarlo Mauri and Antonio E. Porreca *P systems with Active Membranes working in Sublinear Space*
3. Mario J. Pérez-Jiménez *A Bioinspired Computing Approach to Model Complex Systems*

4. J. M. Cecilia, J. M. García, G. D. Guerrero, M. A. Martínez-del-Amor, I. Pérez-Hurtado, M. J. Pérez-Jiménez *Simulation of P systems with active membranes on CUDA*
5. Artiom Alhazov, Rudolf Freund, and Sergey Verlan *Computational Completeness of P Systems Using Maximal Variants of the Set Derivation Mode*
6. Artion Alhazov, Rudolf Freud, Sergiu Ivanov and Marion Oswald *P systems with States*