International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 4– No.8, December 2012 – www.ijais.org

Membrane Computing as Multi Turing Machines

Mahmoud Abdelaziz
Faculty of Computers and information, Cairo University

Amr Badr
Faculty of Computers and information, Cairo University

Ibrahim Farag
Faculty of Computers and information, Cairo University

ABSTRACT

A Turing machine (TM) can be adapted to simulate the logic of any computer algorithm, and is particularly useful in explaining the functions of a CPU inside a computer. Membrane computing aims to develop models and paradigms that are biologically motivated. It identifies an unconventional computing model, namely a P system, which abstracts from the way living cells process chemical compounds in their compartmental structure. These systems are a class of distributed systems, maximally parallel computing devices of a biochemical type. In this research, the research tries investigating a new view to show Membrane computing is a multi TM that communicate with each other. The main idea is that each membrane is a TM itself and each TM can communicate with other TM through communication channels under the structure of membranes (tree membranes structure) where membrane (TM) can send and receive string (multiset) to or from other membrane (TM). This TM is a TM with three tapes.

Keywords

Membrane computing, Turing machine, Persistent Turing Machines.

1. INTRODUCTION

Modern silicon computers have been rapidly advancing in power following Moore's law, but we are fast approaching the natural limits of current designs. In order to continue advancements in computing, we need to develop new computing models and the natural world is a logical step to look for inspiration. Various natural computations have been researched under the umbrella of molecular computing including a new branch known as membrane computing.

A Turing machine consists of two main parts: a 'tape' for storing the input, output and a set of internal states and transitions detailing how the machine should function [1].

A computation of a Turing machine is a (finite) sequence of valid applications of transitions. The Turing machine may end up in a state, above a symbol, for which no applicable transitions exist. The computation is finished and the Turing machine halts.

When a P system is viewed as a computing device, hence it is researched in terms of (theoretical) computer science. The main issues researches refer to the computing power and the computing efficiency (in comparison with standard models, especially Turing machines) [2] and the computing efficiency (how to using parallelism for solving hard problems).

2. PRELIMINARIES

The Membrane Computing or P Systems (created by [3]) are computation systems based on the bimolecular processes of living cells. According to this, the researchers are based on the idea that the simulation of the procedures that take place in nature and their application to machines, it can lead to find and to create new computation models that may lead to a new generation computers. These systems are computational equivalent to Turing machines, however, it's distributed and massively parallel. Until now, researches for implementing membrane systems have not yet reached the massively parallel of this computational model. These systems accomplish its computations through transitions between two consecutive configurations (as same as Turing machines model). Transitions between two consecutive configurations are performed by using evolution rules that present in each region. If the system reaches a configuration in which there are no applicable rules at any membrane, it is said that the system reaches a halting configuration and hence, the computation is successful. The Power of this model is that the evolution process is massively parallel in application rules phases and in communication phase.

This model can form as:

A P system of degree m is a structure

$\Pi = (O, \mu, w1, ..., wm, (R1 , \rho1), ..., (Rm , \rho m), io)$,

Where:

(i) O is an alphabet of objects, and μ is a membrane structure;

(ii) wi is the initial multisets over O associated with the regions defined by μ;

(iii) Ri are finite sets of evolution rules over O associated with the membranes, of typical form $u \rightarrow v$, where u is a multiset over O and V a multiset containing paired symbols of the form (c, here), (c, inj), (c, out) and the dissolving symbol δ;

(iv) ρ_i is a partial order relation over R_i, specifying a priority relation among the rules: $(r_1, r_2) \in \rho_i$ iff $r_1 > r_2$ (i.e., r_1 has a higher priority than r_2);

(v) i_0 is either a number between 1 and m specifying the output membrane of Π, or it is equal to 0 indicating that the output is the outer region.

3. P SYSTEM VS TM

Turing machine is a state transition machine $M = (S, \sum, \delta)$, with finite sets of states S and tape symbols ∑, and a state transition relation $\delta: S \times \sum \rightarrow S \times \{\sum, L, R\}$.

TMs transform finite input strings $x \in \sum^*$ to outputs $y = M(x)$ by a finite sequence of steps, staring in a unique starting state and ending when halting state is reached. At each step, M reads a tape symbol i, performs a state transition (s,i) →(s',o), writes a symbol o, and/or moves the reading head one position right or left [4]. Turing Machines is *non interactive computing devices*. Several classic extensions to the TM model have been proposed, such as *Persistent Turing machines* [5] which is an interactive computation model.

The Membrane computing gives a model of computation pointed to program, so the computing follows a structure similar to the classic algorithms, in which the operations in each step depend on the result get from the previous step.

In this model, the computing (likewise Turing machines), starts from an initial configuration (a structure of membranes) which evolves by means of rules of the system (reactions accepted in membranes). The rules are applied according to a special semantic; we can say that, the execution of such devices modifies the content of their components until arrive one of these states membrane dissolving or membrane division.

Transition P systems are computational equivalent to Turing Machine however; it's distributed and massively parallel. The research will compare between P systems and TM according to their structures and functions, and to simplify the comparison, the research has only two cases for comparing:

The first structure of the system has only one membrane as in Fig (1).

The second structure of the system has more than one membrane as in Fig (2).

Where each membrane has the following:-

- Input string (multiset)
- Rules (which it can be referenced by application phase)
- Initial state, the region (membrane) itself (the initial membrane structure)
- Final state, the region (membrane) itself if there is no any applicable rules
- Sent and receive data (multiset) between membranes (which it can be referenced by communication phase).

3.1 Structure of the system has only one membrane

Turing machine has a tape of input which is the multiset in the membrane. Turing machine has functions that are the rules in the membrane which changes each system from configuration to another.

But transition P systems are dynamics because chemical reactions produce elements that go through membranes to travel to other regions and produce new reactions.

3.2 Structure of the system has more than one membrane

This model has two challenges:-

First: The massively parallel in the two phases:

Application phase and Communication phase.

There are many algorithms for application phase and distributed computing models have been developed, more details about them could be found in ([6], [7]), ([8], [9]).

Second: The change in membrane structure due to membrane dissolving, division and creation.

3.3 Algorithmic vs. Interactive Computation

TMs model *algorithmic* behavior where as P System model *interactive behavior.*

Interactive computing devices: Interactive computing devices allow input /output actions during the process of computation [10, 5].

Table 1. Comparing between algorithmic and interactive computations

Algorithmic	Interactive
computation: finite transformation of input to output	computation: ongoing process which performs a task or delivers a service
input: finite-size (string or number)	Dynamically: generated stream of input and output
closed system: all input available at start, all output generated at end	Open system: input and output exchange on computation

Turing machines (TMs) and Persistent Turing Machines (PTMs) are abstract computing devices useful for representing different forms of computational behavior: TMs model depends on algorithmic behavior while PTMs model depends on sequential interactive behavior (SIMs) [11].

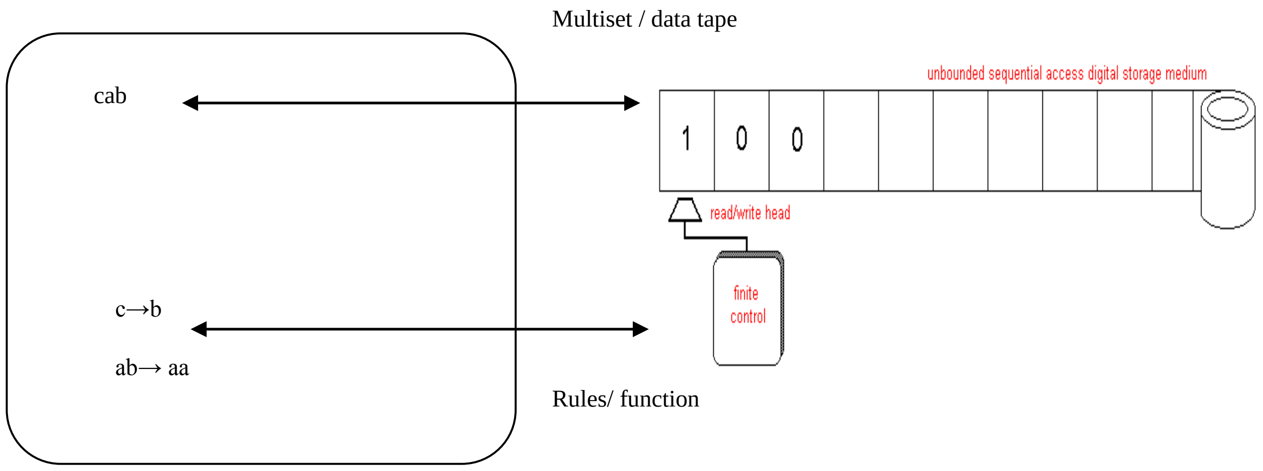

Fig 1 Comparing between TM and a P System of a one membrane

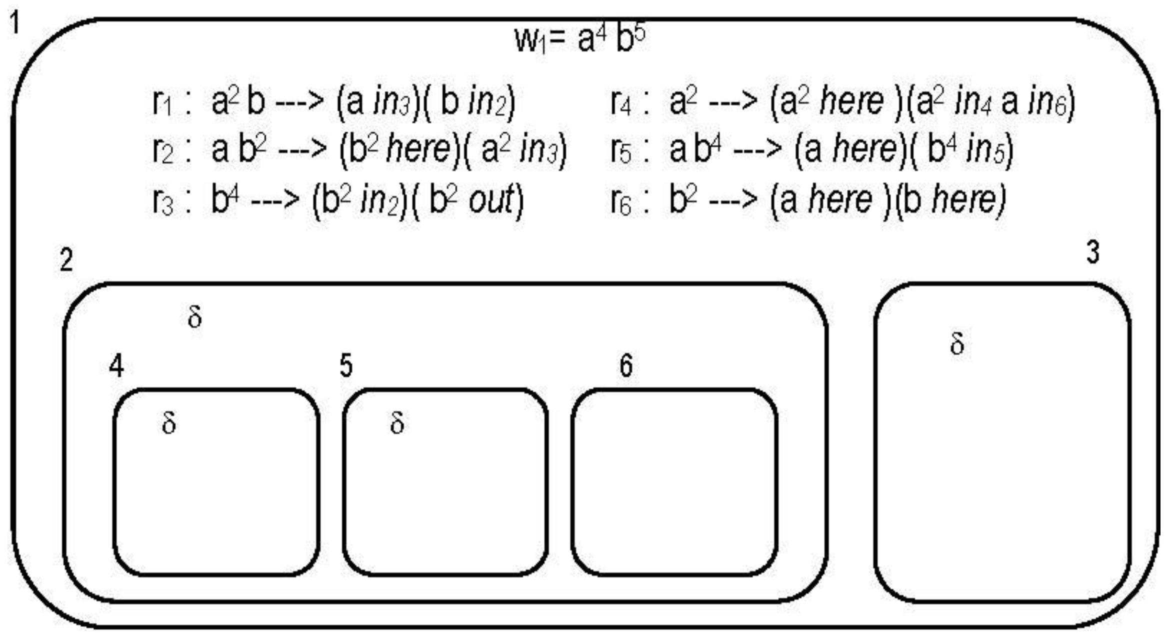
1
$w_1 = a^4 b^5$
$r_1 : a^2 b \dashrightarrow (a\ in_3)(b\ in_2)$
$r_2 : a b^2 \dashrightarrow (b^2\ here)(a^2\ in_3)$
$r_3 : b^4 \dashrightarrow (b^2\ in_2)(b^2\ out)$
$r_4 : a^2 \dashrightarrow (a^2\ here)(a^2\ in_4\ a\ in_6)$
$r_5 : a b^4 \dashrightarrow (a\ here)(b^4\ in_5)$
$r_6 : b^2 \dashrightarrow (a\ here)(b\ here)$
2
δ
3
δ
4
δ
5
δ
6

Fig 2 Membrane structure for more than one membrane

4. Membrane computing as multi TM

Membrane systems are dynamic behavior because chemical reactions create components that travel and exchange with other regions and produce new reactions. This dynamic behavior is possible to be sequenced in transactions of evolution transactions between one system configurations to another. These system configurations are occurred by the membrane structure and multisets that present inside membranes. Transition P systems model can be identified in two phases in each evolution step (Transition). These two phases are the *rules application phase* and *communication phase*. In application rules phase, rules of a membrane are applied in parallel to the multiset inside each membrane. Application phase may produce multisets that travel to other membrane (region). As soon as application rules phase is finished, communications phase work. These systems carry out computations through transitions (steps) between two consecutive configurations, which turn them into a computational model with the same capabilities as Turing machines.

Power of this model lies in the fact that the evolution process is massively parallel in application rules phase as well as in communication phase. The challenge for researchers is to achieve hardware and/or software implementations of P systems taking in consideration the massively parallelism in both phases. In order to produce such model of computation, the two phases of computation should be achieved (the application phase and communication phase). This research well goes through application phase only.

4.1 Description of the proposed TM.

We can say that membranes is a network tree of membranes, each membrane is a Turing machine which communicates with each other through communication device (channels under the tree membranes structure) so these Turing machines must accept and sent data (multisets) from and to its environment, this is the model of Turing machine that will achieve the computation of the membrane. The proposed model is model of Turing machine with three tapes, the first tape is for *input,* the second tape is for *work* and the third tape is for *output*:

- The *input tape* receives inputs (multisets) from its environment through the communication phase.
- The *work tape* will copy data from the input tape to itself, and then it will apply the rule producing output to the output tape.
- The *output tape* sends the output to its destination (here, in, out) through the communication phase.

The above model supports send and receives data to and from its environment; this is equivalent to Persistent Turing Machine since the described model cam send and receive data during processing.

Persistent Turing Machines (PTMs) are multi tape machines with a persistent work tape preserved between successive interactions; they are a minimal extension of Turing machines (TMs) that express interactive behavior [5].

A PTM computation is an (infinite) sequence of Turing computable steps, consuming input stream tokens and generating output stream tokens. The fact that input and output actions on streams take place in the middle of computation characterizes PTMs as interactive computing devices. The interaction of PTMs with their environment can be described by interaction streams, which interleave PTM's inputs and outputs.

4.2 The TM application phase

The *input tape* has its initial data and during computation of membranes it can receive an input (multiset) from its environment through the communication phase.

The data that on the input tape will be copied completely to the work tape at its end of data, and the copied data will be erased from the input tape. The input tape will always receive new data and again this new data will be copied to the work tape at its end.

The *work tape* read from the beginning of its data until finding the following:-

a. A data that can be used by one of the rules, after applying the rule, it produces output, then it sends the output to the output tape and erases the read data that has been used by the rule from the work tape, then the work tape head goes to the beginning of the work tape.
b. If the head of the TM doesn’t find a data that can be used by a rule till the end of the work tape, the head of the work tape will be moved to its beginning and looping until receiving new data from its environment (TM).

The *output tape* receive the output from work tape and sent it to its destination (here, in, out) through the communication phase.

This computation or process is repeated forever until the membrane dissolve or division.

According to the above we can say that the proposed PTM has its transition function which can be represented in a transition table and state transition graph as described in the next example.

4.3 Example

The research submits an example as shown in Fig (3) to show how membrane application phase works according to the proposed PTMs. To simplify the example will use two rules only from Fig (2).

$a^2b \rightarrow$ (a In 3) (b In 2)

$b^2\rightarrow$ (a here) (b here)

In the next page the draw of the state transition graph for the work tape of the proposed PTM in this example.

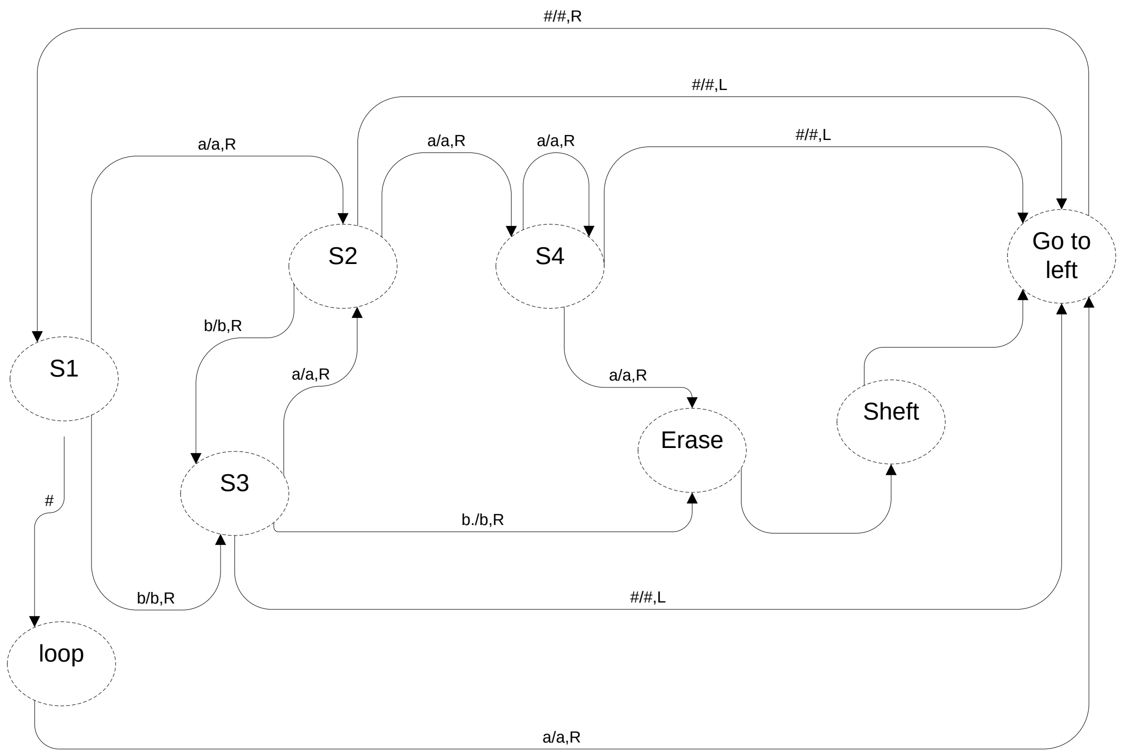

Fig 3 State transition graph for the work tape of the proposed PTM

5. Conclusion

Transition p system is an *interactive behavior* model since cells receive, send data and interact with their environment along their life, to simulate that, we must use a suitable model of computation that satisfy these characteristics. Since standard TMs model depends on *algorithmic behavior* so we didn’t use the standard model TMs and hence we used PTMs model which depends on *sequential interactive behavior* (SIMs). It works for ever and interacts with their environment along its life. We described the proposed PTMs focusing on application phase.

6. REFERENCES

[1] J.Copeland, “The Blackwell Guide to the Philosophy of Computing and Information” Blackwell, 2003.

[2] G. Păun. “Membrane Computing, Basic ideas, Results, Applications” Pre-Proceeding of First International Workshop on Theory and Application of P Systems, Timisoara (Romania), 2005.

[3] G. Păun. “Computing with membranes” In Turku University Computer Science Research Report No. 208, 1998.

[4] John Hopcroft, Jeffrey Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley 1979.

[5] Dina Goldin, Peter Wegner, Persistent Turing Machines, Brown University Technical Report, 1998.

[6] L. Fernández, F. Arroyo, J. Castellanos. et al (2006) “ New Algorithms for Application of Evolution Rules based on Applicability Benchmarks “ Las Vegas (USA).

[7] Tejedor, J, L. Fernández, F. Arroyo, et al (2007) “Algorithm of Active Rules Elimination for Application of Evolution Rules” 8th WSEAS, Vancouver (Canada), 2007.

[8] C. Li, Validating P system as Distributed Computing Models, master thesis, 2008.

[9] G. Ciobanu, Distributed Algorithms over Communication Membrane Systems, Bio Systems, 70(2):123-133, 2003.

[10] Peter Wegner, Interactive Foundations of Computing, Theoretical Computer Science, Feb. 1998.

[11] Peter Wegner, Dina Goldin, Coinductive Models of Finite Computing Agents, Proc. Coalgebra Workshop (CMCS ‘99), Electronic Notes in Theoretical Computer Science, Vol. 19, March 1999.