International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 6 – No. 4, September 2013 – www.ijais.org

A Membrane Turing Machine

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
Membrane Computing (MC) (or P System theory) is a recent area of Natural Computing, the field of computer science that deals with computational techniques is inspired by the structure and functioning of living cells. P systems are massively parallel and distributed model of computation. Membrane Computing investigates models of computation inspired by the structure and functions of biological cells. There are some simulations models that have been developed but do not usually allow parallelism. Turing Machine (TM) and membrane computing are computation models; one of the main differences between them is the behavior of each other, since TM is algorithmic behavior while on the other hand Transition P Systems are Interactive computing behavior. This research investigates a Turing machine model of a special class of P system under a condition which is the rules are applied in a predefined order (which is applying rules priority). From this point of view, the P Systems could assume the same behavior of Turing machine in its sequential behavior. A single membrane can be considered as a machine (membrane devices) in the membrane structure of transition P Systems; hence the whole system of membrane structure consists of several machines that interact with each other. The interaction can be in the form of data passing. The aim of this research is designing a TM to simulate the behavior of a Transition P System. Also the research will show how Turing Machine can be partitioned into sub machines as well as the design of membrane machines can be partitioned into sub machines or sub modules.

Keywords
Membrane computing, P System, Turing machine, Persistent Turing Machines.

1. INTRODUCTION
Design of Modern computers has been growing but current design has a natural limitation. Nowadays, there is a need to find out a new alternative computational paradigm. New materials and technologies [1, 2] such as organic electronics, hybrid electronic biological machines, etc., and the ever increasing complexity and miniaturization of actual systems [3] require rethink the way computers can be built, A good example for such a new paradigm is Membrane computing. Membrane computing is a new branch of natural computing which is initiated by Paun at the end of 1998([4], [5]). This field works on computational models based on nature's behavior to process the information. Most of researchers worked on the idea of simulating the procedures that take place in nature and their application as machines, it can lead to find and create new computation models that may lead to a new computer generation. Transition P 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.

1.1 P Systems Theory
Natural computing is a research field that investigates both the computation designed by human being and computation taking place in nature [6]. Computational models inspired by natural systems such as neural computation, cellular automata and membrane computing. Membrane computing is a research field which deals with computational models inspired from bimolecular processes. Transition P Systems are a parallel and distributed computational model based on the biological membranes and try to simulate the cell behaviors and its functions according to the membrane structure. Each membrane determines a region that encloses a multiset of objects and evolution rules. There are some published deals with parallel implementation of membrane systems ([7], [8], [9]) and Hardware implementations ([10], [11]), and software simulations ([12], [13]). These systems perform its computations through transformation between two consecutive configurations (same as Turing machines model). Transforming occurs by applying evolution rules in each membrane. If the system reaches to a configuration in which there is no rules in any membrane can be applied, the system reaches a halting configuration, and hence the computation is successful end. The Power of this model is that the evolution process is massively parallel in application rules phases and in communication phase. The most attractive feature of membranes computational model is the possibility to represent in a formal way, the transition data and communication occurring into complex distributed computer systems.

Membrane computation works like other programs such as classical algorithms, each of them consists of group of sequence steps in which each step depends on the previous one. Membrane computing model (like Turing Machine model) starts from a specific configuration (a structure of membranes), then it turns into another configuration by executing some rules of the system (reactions accepted in membranes). The rules are applied according to a special semantic; so the execution of such devices modifies the content of their components until it reaches one of the following states, membrane halts stat, membrane dissolving state and membrane division state. This system can be represented in many ways such as Venn diagram or tree graph representation as shown in Fig.1 and Fig.2.

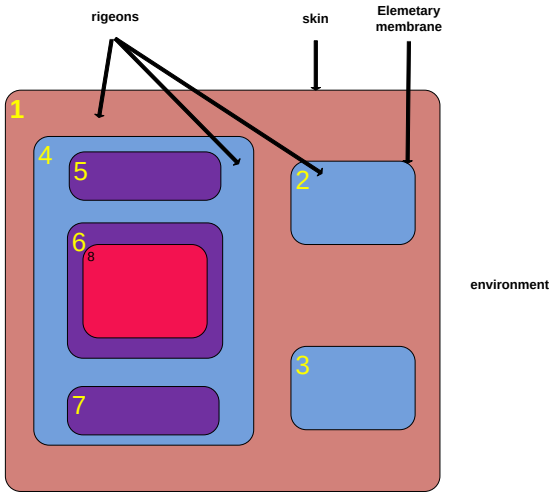

Fig.1. Venn Membrane Structure

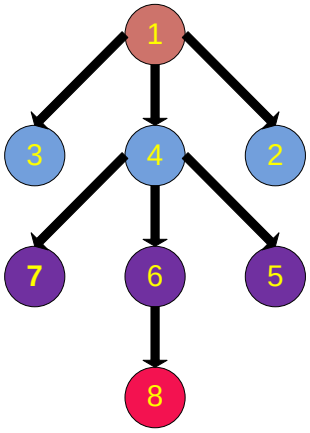

Fig.2. a Membrane Tree Structure for Fig.1

1.2 Turing Machine

Turing machine is a state transition machine M = (S, ∑, δ), with finite sets of states S and tape symbols ∑, and a state transition relation δ: S x ∑→ S x {∑, L, R}.

TMs transform finite input strings x Є ∑* 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 [14]. Several classic extensions to the TM model have been proposed, such as Persistent Turing machines [15] which model can send and receive data during processing so it become an interactive computation model (see Fig.3).

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) [16]. A PTM is a nondeterministic 3-tape Turing machine (N3TM) with a read-only input tape, a read/write work tape (work tape contents is maintained from one computation step to the next), and a write-only output tape. Upon receiving an input token from its environment on its input tape, a PTM computes for a while and then outputs the result to the environment on its output tape, and this process is repeated forever. a PTM performs persistent computations in the sense that the work tape contents is maintained from one computation step to the next, where each PTM computation step represents an N3TM computation.

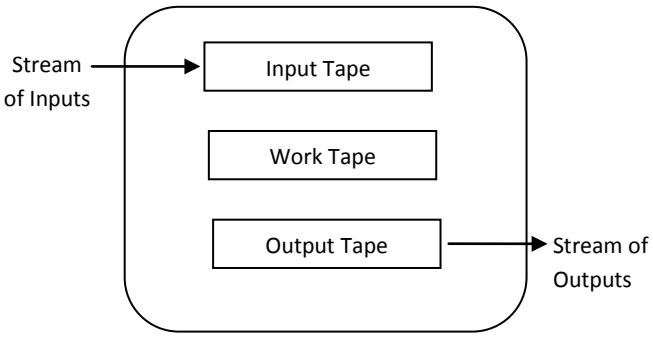

Fig.3. A Persistent Turing Machine

Suppose that the design of TM Machine is huge and more complicated it can be partitioned into sub-machines, each of them perform a specific task, for example if there is an equation like (x = a + b – c * d / e) where a, b, c, d, e are real numbers. This machine can be partitioned as the following: (see Fig.4)

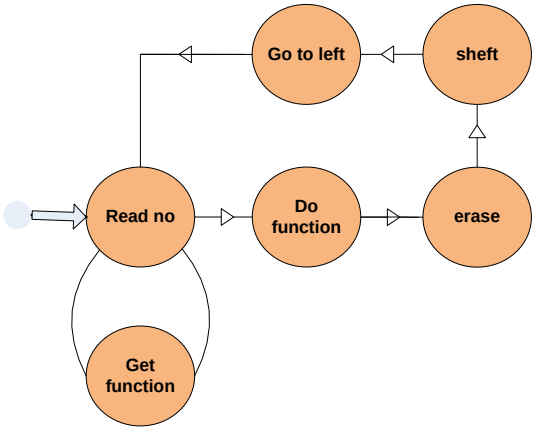

Fig.4 Structure of the exampled machine

- A submachine for reading number
- A sub machine for determining the function will be applied on the two numbers
- A Submachine for performing the function and produce the result
- A Submachine for erasing blanks, shifting data and going left to the beginning of data tape

2. The PROPOSED MEMBRANE TM

The rest of this paper is organized as follows: Section 2.1 Introduction of the proposed Membrane TM. Section 2.2 Data

representation. Section 2.3 Relation among Membrane Structure. Section 2.4 Description of the machine execution. Finally Section 3 Conclusions.

2.1 Introduction of the proposed Membrane TM

Membranes are 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 receive and send data (multisets) from and to its environment, which is PTM. Membrane computation model has two main phases, they are affects each other, those two phases are *communication phase* and *application or computation phase*.

Communication Phase: When some rules in the application phase are applied, this will lead to membrane dissolve or division that affects the structure of the communication tree, thus affecting the communication phase.

Application Phase: Some rules in this phase sends data to a connected membrane so these rules shouldn't be executed if its connections doesn't exist with this membrane, hence we can say that communication phase affect the application phase.

As noted from the above, there are two types of data that come to the membrane through its environment (communication channels which connects the membrane with other membranes):

- *Communication or network data*: Information about the connection channels which connected to the membrane with other membranes in the system (communication network data), this information constitute the membrane communication tree which control some rules to be executed or not according to the communication state, it changes the state of the rule to be useful or unuseful, this gives the system more power since it will decrease the selection time of those rules to be executed or not.
- *Incoming data object:* which are concatenated with the existing data in the membrane (membrane work tape), this data (multiset data) are consumed by some rules (useful rules) and produce outputs to the environment on the output tape.

So the proposed model worked under assumption which is each membrane has two types of input (input data object / communication network data), this assumption is applied to each membrane (machine) in the system. This assumption let all membrane work independent and this gives a power to the system because the system can have unlimited number of membranes. In this way the membrane has its input data so its application phase can perform its process over those data directly. We can simulate this computation by using any model of computation; the most important model in computation models is the Turing machine model which used in designing the computer. The membrane in this model is dealing with its environment as it is the supplier of its data and as a consumer to its output data regardless from where this input data comes or how the output data is transported to other membranes. The transportation process is related to the communication phase which is not included in this research since it concerns with the application phase only, in the application phase we concern only on what media the data (output and input) save on.

2.2 Data representation

Each symbol of data is represented and encoded as 8 bits according to an encoding table. The proposed machine consists of three tapes, one tape for its input data, one tape for its output data and one work tape. Data will be represented as the following.

2.2.1 Input tape

Input data which come from the membrane environment is saved on that tape either these data is communication data or object data, so the data representation is divided into two parts, the first one is the communication data part followed by the second part which is the object data part as it is shown in Fig. (5). This representation enables the machine to differentiate in dealing among these different data types when updating the work tape as will be explained later in this section.

Begin of Data	Data type	Data it self	---	End of Data

Fig.5 Machine input tape

2.2.2 Output tape

Output data produced from the execution of the membrane rules are saved on this tape, this output data can be sent to many membranes, so its representation is divided also into two parts, the first part specifies the target membrane and the second part represent the output data itself as it is shown in Fig. (6).

Target I	Data	Target J	data	---

Fig.6 Machine output tape

2.2.3 *Work tape*

This tape is divided into three parts as shown in Fig. (7), the first part represent the object data that will be input data for the rules that will be executed, in this part the symbols of data are represented in sequence. The second part represents information about the rules in the membrane that will be executed, these information are the rule no, rule itself and its state, sorted according to its execution priority, the third parts contains the total no of the rules that are valid to be executed at a specific moment in the membrane, the machine halts when the total number of rules to be executed at a specific moment equal to zero, this will solve the halting problem of the Turing machine. Also sorting the rules according to its priority in execution enables the machine to apply the rules according to a predefine order not in a maximal parallel manner as in p system so the machine work in a sequential manner.

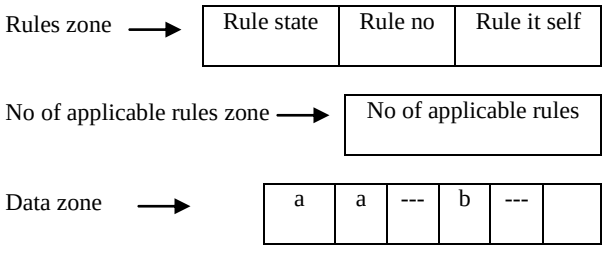

Fig.7 Machine work tape tape

2.3 Relation among Membrane Structure

The membrane structure is the inter connection and hierarchical organization among membranes (machine membrane). Any relation between two membranes is basically used when objects (input/output) are being transferred between the two membranes (membrane machine). The membrane machine can send and receive object (input/output) only from the membrane it is contained inside or from the membranes it contains as shown by the following example.

Example: membrane 4 of Fig. (1) Can send and receive objects from the membrane it is inside in which in this figure it is membrane No. 1 and from membranes 5, 6, and 7 that they are inside membrane 4. Membrane 4 can't directly communicate with membranes 8, 2 and 3. The connections are bidirectional in communication between any two membranes that have a relation as shown in Fig. (8) Where arrows indicate to the stream of input / output of objects between membranes.

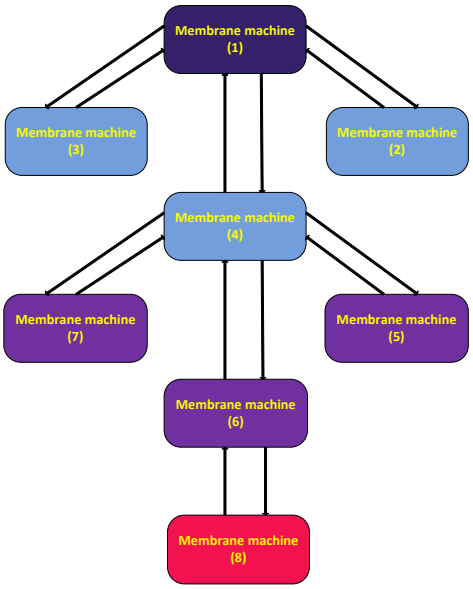

Fig.8 All Relation among Membrane Structure for Fig 1

2.4 Description of the machine execution.

Each machine applies its rules on the work tape, the selection of the rule that will be executed as the following:-

An evolution rule in membrane (i) can be applied in an evolution step if it fulfils

1- Useful: if all targets membranes are adjacent to membrane (i) (there exist a connection channels between membrane (i) and their targets membranes),So the machine puts (00) in the rule state to specify that this rule is out of execution since there are no communication channels between the membrane and its target membrane by the output data of this rule, this enables filtering the not applicable rules due to the lack of communication channels, this prevents the error of sending data to non existing targets.

For Example: In Fig. (9) Rule No. 4 in membrane No. 1 can't be applied since there is no connection pass between membrane 1 and membranes 4 or 6.

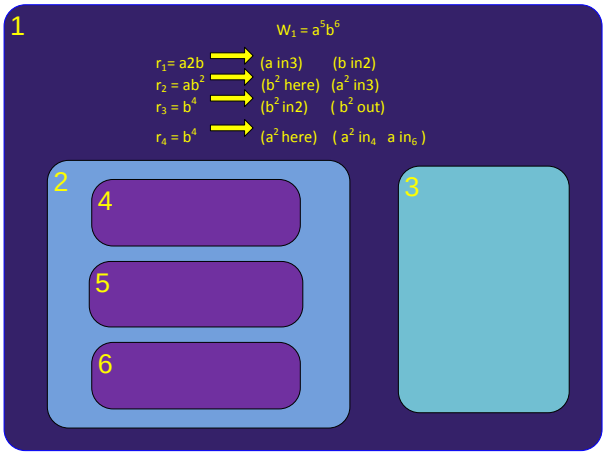

Fig.9 Membrane structure and rules

1- Applicable: if the membrane (i) have the data objects that the rule need to execute so the machine in the state rule (01) else if there is no data object that the rule need to execute the machine put (10) in the state rule.
2- Active: the rule has the turn to execute according to its priority so the machine put in the state rule (11).

According to this issue the machine works as described in the following Fig. (10).

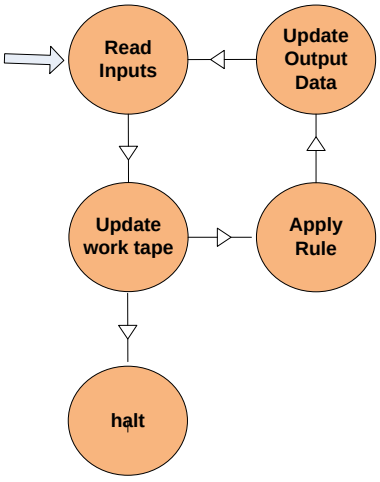

Fig.10 the propose Membrane Machine structure

In case of coming input data on the input tape. The machine reads it and updates the data on the work tape then deleting it from the input tape to make it ready for the new input data if exists. The machine before updating the data on the work tape

- If the data is communication data, the machine updates the states of its rules according to the existing or no existing channels among that membrane and other membranes, it changes the rule state from (10) to (00) if there is no communication channel with the membrane which will receive the output data of that rule, and it will change the rule state from (00) to (10) if there is a communication

channel with the membrane which will receive the output data of that rule and at the same time the machine updates the total number of the possible rules that can be executed.
- If the data is object data, the machine changes all waiting rules states from (10 the rule state of non existing object data) to be (01 the rule state of existing object data) since it assumes that object data is related to those waiting rules and at the same time the machine updates the total number of the possible rules that can be executed.

After ending the updating process on the work tape, the computation process on the work tape starts as the following:-

- The machine reads all the rules till it find a rule that has a state (01) then it changes it to (11) (the state of checking the applying the rule) and then the machine check possibility of applying this rule by searching on the work tape for its object data
- If the machine finds the object data of the rule then the rule will be executed and it deletes the used object data from the work tape and writes the output data according to its destinations (work tape or output tape or both of them).
- The machine loops the last step till it doesn't find any object data for the current applied rule then it changes its state from (11) to (10) and decrease the total number of the possible rules that can be executed by 1, after that the machine searches for next applied rule to be executed and loops again till it doesn't any rule to be executed, this check occurs by checking the total number of the possible rules that can be executed to be equal zero, in this case the machine halts, and thus the machine halts problem is being solved, the machine will continue in its halt state till new incoming input data then the whole previous process starts again.
- A general C code for the proposed machine introduced as follow.

```
// the following code is for each membrane
move_the_head_of_the_w_tape_to_the_first_R(w_tape,M_no)
While(;;)
{
    total_no_of_app_Rs=get_app_Rs(w_tape,M_no)

    if total_no_of_app_Rs > 0 then
        R_no = get_the_R(w_tape,M_no)

    x=check_comm_state_of_Ms(w_tape,M_no,R_no)
        if x = "00" then // no comm

        else

    y=check_data(w_tape,M_no,R_no)
        if y = false then //no data

    change_R_status(w_tape,M_no,R_no,"10")

    move_the_head_of_the_w_tape_to_the_next_R(w_tape,M_no)

        else

    execute_the_R(R_no,o_R_no,o_data)
        write_the o_data(o_tape,o_R_no,o_data)

        end if
        i_data= check_data_in_i_tape(i_tape,M_no)
        comm_data= check_comm_data_in_i_tape(i_tape,M_no)
        if i_data <> null then

    update_data_on_w_tape(w_tape,M_no,i_data)

    total_no_of_updated_Rs= update_all_related_Rs(w_tape,M_no,"01")

    update_no_of_app_Rs(w_tape,M_no,no_of_app_Rs,total_no_of_updated_Rs)
        end if
        if comm_data <> null then

    update_comm_data_on_w_tape(w_tape,M_no,i_data)

    total_no_of_updated_Rs= update_all_related_Rs_comm(w_tape,M_no)

    update_no_of_app_Rs(w_tape,M_no,no_of_app_Rs,total_no_of_updated_Rs)
        end if

    end if
    else
        wait(10000) // wait ten second
        i_data= check_data_in_i_tape(i_tape,M_no)
        comm_data= check_comm_data_in_i_tape(i_tape,M_no)
```

```
            if i_data <> null then

        update_data_on_w_tape(w_tape,M_no,i_data)
                                total_no_of_updated_Rs=
update_all_related_Rs(w_tape,M_no,"01")

        update_no_of_app_Rs(w_tape,M_no,no_of_app_Rs
,total_no_of_updated_Rs)
                    end if
                    if comm_data <> null then

        update_comm_data_on_w_tape(w_tape,M_no,i_dat
a)
                                total_no_of_updated_Rs=
update_all_related_Rs_comm(w_tape,M_no)

        update_no_of_app_Rs(w_tape,M_no,no_of_app_Rs
,total_no_of_updated_Rs)
                    end if


        end if


}
```

3. Conclusion

The physical limitations of current silicon hardware are one of the triggers for the development of alternative computation models. In particular, the scientific community is in increase with high interest in computation models inspired by nature. Developers of P system models require a computation model which is able to achieve the parallelism of the P System models. This research investigates a Turing machine model which simulates the P system model under a constrain which is the rules are applied in predefined order (priority). The proposed model of this research enables achieving the parallelism of the P system model in interactive sequential manner so we can simulate it in a computer application, more over the proposed module can be built as hardware machine.

4. REFERENCES

[1] N. Mathur, Beyond the Silicon Roadmap, Nature, 419, 6907, October 10, 2002, 573-575.

[2] S. De Franceschi, L. Kouwenhoven. Electronics and the Single Atom. Nature, 417, June 13 2002, 701-702.

[3] International Technology Roadmap for Semiconductors, Semiconductor Industry Association, http://public.itrs.net/Files/2001ITRS, 2001.

[4] G. Păun. "Computing with membranes" In Turku University Computer Science Research Report No. 208, 1998.

[5] Gheorghe Paun, A quick introduction to membrane computing, The Journal of Logic and Algebraic Programming, 79, 2010, 291-294

[6] Rozenberg G., B¨ack T., Kok J. N., eds. (2011) Handbook of Natural Computing, volume II. Springer.

[7] G.Ciobanu, G.Wenyuan, "A P System running on a cluster of computers", Proceedings of Membrane Computing. International Workshop, Tarragona (Spain). Lecture Notes in Computer Science, vol 2933 (2003) 123-150.

[8] A. Syropoulos, E.G. Mamatas, P.C. Allilomes, K.T.Sotiriades, "A distributed simulation of P systems". Preproceedings of the Workshop on Membrane Computing (A. Alhazov, C.Martin-Vide and Gh.Păun, eds); Tarragona, vol July 17-22 (2003), 455-460.

[9] J.Tejedor, L.Fernández, F.Arroyo, G.Bravo, An architecture for attacking the bottleneck communication in P systems. In: M. Sugisaka, H. Tanaka (eds.), Proceedings of the 12th Int. Symposium on Artificial Life and Robotics, Jan 25-27, 2007, Beppu, Oita, Japan, 500-505.

[10] Fernandez, L., Martinez, V. J., Arroyo, F., and Mingo,L.F.(2005). A hardware circuit for selecting active rules in transition p systems. Seventh International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Proceedings, 1:415-418.

[11] Martinez, V., Arroyo, F., Gutierrez, A., and Fernandez, L. (2007). Hardware implementation of a bounded algorithm for application of rules in a transition p-system. SYNASC 2006: Eighth International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Proceedings, 1:343-349

[12] Suzuki, Y. and Tanaka, H. (2000). On a lisp implementation of a class of p systems. In Romanian Journal of Information Science and Technology, volume 3, pages 173-186.

[13] Arroyo, F., Luengo, C., Baranda, A. V., and de Mingo, L. (2003). A software simulation of transition p systems in Haskell. Membrane Computing, 2597:19-32.

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

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

[16] 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.