Note on some recursive family of P systems with active membranes

Adam Obtułowicz
Institute of Mathematics, Polish Academy of Sciences
Śniadeckich 8, P.O. Box 137
00-950 Warsaw, Poland
e-mail: adamo@impan.gov.pl

July 25, 2001

Abstract

There is given an outline of a multitape Turing machine which generates in a logarithmic space of working tapes programs for some family of deterministic P systems with active membranes applied to solve SAT problem in a linear time.

Introduction

P systems with active membranes applied to solve SAT problem in linear time and described in [1] and [3] constitute families of P systems Π^{Φ} depending on propositional formulas Φ in conjunctive normal form. Intuitively these families are recursive families, i.e. one can define a Turing machine which for any input Φ generates an output in the form of a program for P system Π^{Φ}. We explain that a program for a P system Π with active membranes is meant to be a sequence of lists of data which determine Π including the list of evolution rules of Π. Thus one can ask about estimates of time and space of computations of a Turing machine which for any input Φ generates a program for Π^{Φ}.

We show in the note that for the family of P systems Π^{Φ} described in [1] it suffices to apply a multitape Turing machine which for any input Φ generates in a logarithmic space a program for Π^{Φ} as an output. Needless to say that logarithmic space concerns here working tapes (counters) of a multitape Turing machine which reads-only its input tape and writes-only its output tape. It is known fact that if such a machine computes in a logarithmic space of working tapes, then this machine halts after a polynomial number of steps, cf. [2].

Therefore the family of P systems Π^{Φ} described in [1] reaches the standards of uniform programs for Parallel Random Access Machines discussed in [2], where uniform programs are claimed to be generated in a logarithmic space by multitape Turing machines.

Analogous facts concerning the family of P system Π^Φ described in [3] can be proved in a similar way as in the present note.

In Section 1 we recall P systems Π^Φ described in Obtułowicz (2001) and then we introduce a concept of a program for Π^Φ to be a succinct description of Π^Φ. Section 2 contains an outline of a Turing machine which generates in logarithmic space the programs for P system Π^Φ described in Section 1.

For all unexplained terms concerning computational complexity and multitape (multistring) Turing machines we refer the reader to [2].

1 P systems solving SAT problem and programs for P systems

We recall in this Section deterministic P systems Π^Φ applied to solve SAT problem in linear time and described in [1]. Then we describe programs for these P systems.

By a propositional formula in the *conjunctive normal form* of n variables $x_1, \ldots, x_n$ and of m clauses we mean a propositional formula

$$\Phi = C_1 \wedge \ldots \wedge C_m$$

for C_i $(1 \leq i \leq m)$ being propositional formulas, called *clauses* of Φ, such that they are of the form of disjunctions

$$C_i = \alpha_1^i \vee \ldots \vee \alpha_{j_i}^i,$$

where α_j^i $(1 \leq j \leq j_i)$, called *components* of C_i, are propositional variables x_k themselves or negations $\neg(x_k)$ for $1 \leq k \leq n$.

For a formula Φ in the conjunctive normal form of n variables $x_1, \ldots, x_n$ and m clauses $C_1, \ldots, C_m$ we define a P system $\Pi^\Phi = (\mathcal{S}^\Phi, \mathcal{R}^\Phi)$ such that

— the membrane system $\mathcal{S}^\Phi$, called initial configuration in [3], is such that its underlying set $\mathbb{B}$ of balls contains exactly three balls $b_2 \subsetneq b_1 \subsetneq b_0$, the set L of labels of $\mathcal{S}^\Phi$ is the set $\{0,1,2\}$, and $l(b_i) = i$, $e(b_i) = 0$ for all $i \in \{0,1,2\}$, where the balls represent membranes of initial configuration and $e(b_i) = 0$ means that the electric charge of the membrane corresponding to a ball b_i is 0,

— the set O of objects of $\mathcal{S}^\Phi$ is the set of ordered triples (σ, x, y), written $(\sigma)_y^x$, such that $\sigma \in \{0,1\}$, $x \in \{1, \ldots, m\} \cup \{\perp, -@, @, +@\}$, and $y \in \{0, 1, \ldots, n\} \cup \{\perp\}$ if x is not a natural number, where the four different elements $\perp$, $-@$, $@$, $+@$ are not natural numbers,

— the labelling function $M : \mathbb{B} \to N^O$ of $\mathcal{S}^\Phi$ valued in the set N^O of multisets is defined in the manner of [1] by

$$M(b_0) \;=\; M(b_1) = \mathbb{O},$$

$$M(b_2) \;=\; \langle (0)_0^{\perp} \rangle + \Big(\sum_{j=1}^{n-1} \langle (1)_j^{\perp} \rangle \Big) + \langle (1)_n^{@} \rangle,$$

where the labelling function M and the whole initial configuration $\mathcal{S}^{\Phi}$ can be described simply by the following parenthesis expression

$$[_0 [_1 [_2 (0)_0^{\perp} (1)_1^{\perp} \ldots (1)_{n-1}^{\perp} (1)_n^{@}]_2^0]_1^0]_0^0$$

written in the manner of [3],

— the set $\mathcal{R}^{\Phi}$ of rules contains the rules given by the following schemes:

(S_1) $[_2 (1)_i^{@}]_2^0 \to [_2 (1)_i^{-@}]_2^- [_2 (1)_i^{+@}]_2^+$ for $1 \leq i \leq n$,

(S_2) $[_2 (\sigma)_{i-1}^{\perp} (1)_i^{-@} \to (\sigma)_{i-1}^{@} (0)_i^{\perp}]_2^-$, for $1 \leq i \leq n$ and $\sigma \in \{0,1\}$,

(S_3) $[_2 (\sigma)_{i-1}^{\perp} (1)_i^{+@} \to (\sigma)_{i-1}^{@} (1)_i^{\perp}]_2^+$, for $1 \leq i \leq n$ and $\sigma \in \{0,1\}$,

(S_4) $[_1 [_2 \,]_2^- \, [_2 \,]_2^+]_1^0 \to [_1 [_2 \,]_2^0]_1^0 \, [_1 [_2 \,]_2^0]_1^0$,

(S_5) $[_2 (0)_0^{@}]_2^0 \to (0)_0^{@}$,

(U_1) $[_1 (1)_j^{\perp} \to (\sigma_1)_{\perp}^1 \ldots (\sigma_m)_{\perp}^m]_1^0$ for $1 \leq j \leq n$ and for a binary string $\sigma_1 \ldots \sigma_m$ such that

$$\sigma_k = \begin{cases} 1 & \text{if the variable } x_j \text{ is a component of the clause } C_k, \\ 0 & \text{otherwise,} \end{cases}$$

(U_2) $[_1 (0)_j^{\perp} \to (\sigma'_1)_{\perp}^1 \ldots (\sigma'_m)_{\perp}^m]_1^0$ for $1 \leq j \leq n$ and for a binary string $\sigma'_1 \ldots \sigma'_m$ such that

$$\sigma'_k = \begin{cases} 1 & \text{if the formula } \neg(x_j) \text{ is a component of the clause } C_k, \\ 0 & \text{otherwise,} \end{cases}$$

(W) $[_1 (1)_{\perp}^1 \ldots (1)_{\perp}^m \to (1)_{\perp}^{\perp}]_1^0$,

(T_1) $[_1 (1)_{\perp}^{\perp}]_1^0 \to [_1 \,]_1^+ (1)_{\perp}^{\perp}$,

(T_2) $[_0 (1)_{\perp}^{\perp}]_0^0 \to [_0 \,]_0^+ (1)_{\perp}^{\perp}$.

There is no any other rule in $\mathcal{R}^{\Phi}$ than that described by the above schemes.

Theorem 1 *The P system $\Pi^{\Phi} = (\mathcal{S}^{\Phi}, \mathcal{R}^{(\Phi)}$ is a deterministic system and the process of evolution generated by it stops after the number of steps no greater than $2n + 5$. Moreover, the object $(1)_{\perp}^{\perp}$ is sent out of the system (skin) iff the formula Φ is satisfied for some valuation of variables occurring in it.*

Proof. The notions of a deterministic P system and a process generated by a P system are explained in [1]. The proof of above Theorem 1 is contained in [1]. □

By a *program* for a P system $\Pi = (\mathcal{S}, \mathcal{R})$ we mean a succinct description of data determining Π. A program for Π can be written as a sequence of strings including a parenthesis expression describing membrane system $\mathcal{S}$, called initial configuration in [3], and lists of other data determining Π like the list of objects of $\mathcal{S}$. Such a sequence should contain the list of rules belonging to the set $\mathcal{R}$ which is a core of a program for Π.

In the case of P system $\Pi^\Phi = (\mathcal{S}^\Phi, \mathcal{R}^\Phi)$ described above for a propositional formula Φ in conjunctive normal form of n variables and m clauses the list of rules belonging to $\mathcal{R}^\Phi$ can be presented in the following succinct form.

Since the rules defined by schemes (S_1)–(S_3) have a uniform shape not dependent on Φ and the scopes of application of the schemes depend only on the number n, one can rewrite these schemes instead of writing a string of rules, where the scopes of application of the schemes should be indicated by writing in binary the number n.

The rules defined by the schemes (U_1) and (U_2) depend on the shape of Φ and their list can be presented by two binary matrices

$$A = [\sigma_{ij}]_{m\times n} \text{ and } B = [\sigma'_{ij}]_{m\times n},$$

respectively, of n rows and m columns for $\sigma_{ij}, \sigma'_{ij} \in \{0, 1\}$. The rows of matrices A and B represent the rules defined by the schemes (U_1) and (U_2), respectively, where a row corresponds to the string of objects next to symbol $\rightarrow$ in a rule and the index of a row corresponds to the object followed by $\rightarrow$ in a rule.

We assume that a binary matrix is presented by a sequence of binary strings of the same length which correspond to the rows of a matrix.

2 Turing machine generating programs for P systems

We present in this Section a multitape Turing machine which for any propositional formula Φ in conjunctive normal form and of some regular shape generates in a logarithmic space a program for P system Π^Φ described in Section 1.

Propositional formulas of regular shape are described in the following way.

We use propositional variables, called *regular variables*, which are expressions $X\{i\}$, where X is capital Latin X and i is a binary presented natural number greater than 0 called the *index* of variable $X\{i\}$.

A propositional formula Φ is called *regular* if every propositional variable occurring in Φ is regular and its index i is such that

$$|i| \leq C \log |\Phi|$$

for some constant C, where $|i|$ and $|\Phi|$ denote the length of binary presented i and the length of Φ as a string of symbols, respectively. Thus by a *regular formula of n variables* we mean a regular propositional formula such that n is the greatest index of variables occurring in it.

The following theorem concerns programs for P systems Π^Φ described in Section 1.

Theorem 2 *There exists a multitape Turing machine T with a input tape, an output tape, and working tapes (counters) such that for any regular propositional formula Φ in conjunctive normal form written in the input tape machine T reads-only its input tape, writes-only its output tape, and generates in a logarithmic space of working tapes a program for P system $\Pi^{\Phi} = (\mathcal{S}^{\Phi}, \mathcal{R}^{\Phi})$ by writing it in the output tape.*

Proof. We outline in the proof only those procedures realized by claimed machine T which use working tapes (counters).

To write the parenthesis expression presenting $\mathcal{S}^{\Phi}$ and scopes of application of rules (S_1)–(S_3), machine T counts in binary the greatest index n of variables occurring in Φ by using one counter in the following way.

Machine T reads from left to right the occurrences of variables in Φ and computes storing in a counter the current greatest index of the already read variables. After reading the last occurrence of a variable in Φ the counter contains the greatest index n of variables occurring in Φ.

Since the indices of variables are presented in binary also in a counter, the greatest index of variables occurring in Φ is counted in a logarithmic space because Φ is regular.

To write the matrices A and B presenting the rules given by schemes (U_1) and (U_2), machine T uses one counter in the following way.

Each of the matrices A and B is computed and written in n stages such that i-th row of a matrix is computed and written during i-th stage, where n is the greatest index of variables occurring in Φ. In the $i+1$-th stage, next to i-th stage, binary presented i is incremented by one in binary and machine T tests the occurrences of the variable $X\{i+1\}$ in the clauses of Φ by reading the clauses from left to right in Φ. Thus the matrices A and B are computed and written in the output tape by using logarithmic space of a counter.

Therefore machine T generates programs for P systems Π^{Φ} in a logarithmic space of working tapes. □

Corollary *For every language $L \subseteq \Sigma^*$ belonging to complexity class NP there exists a Turing machine T_L such that for any input string w in Σ^* machine T_L generates in a logarithmic space of working tapes a program for P system Π^w which decides in linear time whether $w \in L$.*

Proof. Let R be a reduction of a language $L \subseteq \Sigma^*$ to SAT problem, where the values $R(w)$ are regular propositional formulas in conjunctive normal form. By Theorem 1 the P systems $\Pi^{R(w)}$ defined for the formulas $R(w)$ as in Section 1 decide in linear time whether $w \in L$.

One can construct claimed machine T_L from a Turing machine T_R computing in a logarithmic space the reduction R and from the machine T outlined in the proof of Theorem 2. A construction of T_L from the machines T_R and T is analogous to the construction of a Turing machine which computes in logarithmic space the composition of two reductions of decision problems, cf. [2]. □

References

[1] A. Obtułowicz, *Deterministic P systems for solving SAT problem*, to be published in Romanian Journal of Information Science and Technology.

[2] Ch. P. Papadimitriou, *Computational Complexity*, Reading, Mass., 1994.

[3] Gh. Păun, *P-Systems with Active Membranes: Attacking NP Complete Problems*, Journal of Automata, Languages and Combinatorics 6 (2000), 75–90.

[4] Gh. Păun, *Computing with membranes*, Journal of Computer and System Science 61 (2000), 108–143.