A Class of P Automata for Characterizing Context-free Languages*

György Vaszil

Computer and Automation Research Institute
Hungarian Academy of Sciences
Kende utca 13-17, 1111 Budapest, Hungary
vaszil@sztaki.hu

Summary. We present a characterization of context-free languages in terms of a restricted class of P automata (P systems accepting strings of symbols using symport/antiport communication rules). The characterization is based on the form of the rules used by the system.

1 Introduction

Membrane systems, or P systems, are distributed computing models inspired by the functioning of the living cell. Their main components are membrane structures consisting of membranes hierarchically embedded in the outermost skin membrane. Each membrane encloses a region containing a multiset of objects and possibly other membranes. Each region has an associated set of operators working on the objects contained by the region.

The evolution of the objects inside the membrane structure from an initial configuration to a somehow specified end configuration corresponds to a computation having a result which is derived from some properties of the specific end configuration. Several variants of the basic notion have been introduced and studied proving the power of the framework; see the monograph [6] for a summary of notions and results of the area, and the web site [8] for recent developments. As the reader might notice, this machinery has relatively strong capabilities, so it might seem reasonable to look for as simple P system variants as possible, as for example systems using communication rules only. For more details on these variants, see [3], [5].

In the present paper we deal with the problem of characterizing language classes, other then the class of regular or recursively enumerable languages, in

* Research supported in part by the Hungarian Scientific Research Fund "OTKA" grant no. F037567 and by the Alexander von Humboldt Foundation of the Federal Republic of Germany. Completed during the author's stay at the Research Group Theoretical Computer Science of the Otto-von-Guericke-Universität, Magdeburg, Germany.

terms of P systems using only symport/antiport communication rules. We use the notion of P automata.

Besides their simplicity, the introduction of P automata in [1] was motivated by the idea of using P systems as language acceptors. The objects in a P automaton may move through the membranes from region to region, but they may not be modified during the functioning of the systems, and furthermore, the "words" accepted by a P automaton correspond to the sequences of multisets containing the objects entering from the environment in each step of the evolution of the system.

In what follows, after presenting the necessary definitions and notations, we first shortly review other characterizations of non-regular language classes which do not coincide with the class of recursively enumerable languages, in particular, a sub-logarithmic-space class from [2], and two characterizations of context-sensitive languages from [2] and from [4], and then we present our result, the characterization of context-free languages in terms of P automata.

2 Preliminaries and Definitions

We first recall the notions and the notations we use. The reader is assumed to be familiar with the basics of formal language theory (for details see [7]). Let Σ be a finite set of symbols called alphabet. Let Σ^* be the set of all words over Σ, that is, the set of finite strings of symbols from Σ, and let $\Sigma^+ = \Sigma^* - \{\varepsilon\}$ where ε denotes the empty word.

Let V be a set – the universe – of objects, and let $\mathbb{N}$ denote the set of natural numbers. A multiset is a pair $M = (V, f)$, where $f : U \to \mathbb{N}$ is a mapping which assigns to each object $a \in V$ its multiplicity. The support of $M = (V, f)$ is the set $supp(M) = \{a \in V \mid f(a) \geq 1\}$. If $supp(M)$ is a finite set, then M is called a finite multiset. The set of all finite multisets over the set V is denoted by V°.

We say that $a \in M = (V, f)$ if $a \in supp(M)$. $M_1 = (V, f_1) \subseteq M_2 = (V, f_2)$ if $supp(M_1) \subseteq supp(M_2)$ and for all $a \in V$, $f_1(a) \leq f_2(a)$. The union of two multisets is defined as $(M_1 \cup M_2) = (V, f')$ where for all $a \in V$, $f'(a) = f_1(a) + f_2(a)$, the difference is defined for $M_2 \subseteq M_1$ as $(M_1 - M_2) = (V, f'')$ where $f''(a) = f_1(a) - f_2(a)$ for all $a \in V$, and the intersection of two multisets is $(M_1 \cap M_2) = (V, f''')$ where for $a \in V$, $f'''(a) = min(f_1(a), f_2(a))$, $min(x, y)$ denoting the minimum of $x, y \in \mathbb{N}$. We say that M is empty, denoted (as in the case of sets) by $\emptyset$, if its support is empty, $supp(M) = \emptyset$.

A multiset M over the finite set of objects V can be represented as a string w over the alphabet V with $|w|_a = f(a)$ where $a \in V$ and where $|w|_a$ denotes the number of occurrences of the symbol a in the string w. In the following we sometimes identify the finite multiset of objects $M = (V, f)$ with the word w over V representing M, thus we write $w \in V^\circ$, or even $w_1 \cup w_2$ where $w,\ w_1,\ w_2$ are strings of symbols from V, or sometimes we enumerate the not necessarily distinct elements of $w = a_1 \dots a_t \in V^\circ$ as $\{\{a_1, \dots, a_t\}\}$.

A P system is a structure of hierarchically embedded membranes, each having a label and enclosing a region containing a multiset of objects and possibly other membranes. The outmost membrane which is unique and usually labeled with 1, is called the skin membrane. The membrane structure is denoted by a sequence of matching parentheses where the matching pairs have the same label as the membranes they represent. If membrane i contains membrane j, and there is no other membrane, k, such that k contains j and i contains k, then we say that membrane i is the parent membrane of j.

The evolution of the contents of the regions of a P system is described by rules associated to the regions. Applying the rules synchronously in each region, the system performs a computation by passing from one configuration to another one. In the following we concentrate on communication rules called symport or antiport rules.

A symport rule is of the form (x, in) or $(x, out), x \in V^\circ$. If such a rule is present in a region i, then the objects of the multiset x must enter from the parent region or must leave to the parent region, respectively. An antiport rule is of the form $(x, in; y, out), x, y \in V^\circ$, in this case, objects of x enter from the parent region and in the same step, objects of y leave to the parent region. All types of these rules might be equipped with a promoter or inhibitor multiset, denoted as $(x, in)|_Z, (x, out)|_Z$, or $(x, in; y, out)|_Z, x, y \in V^\circ, Z \in \{z, \neg z \mid z \in V^\circ\}$, in which case they can only be applied if region i contains the objects of multiset z, or, if $Z = \neg z$, then region i must not contain the elements of z. (For more on symport/antiport see [5], for the use of promoters see [3].)

A *P automaton* is a construct $\Gamma = (V, \mu, (w_1, P_1, F_1), \ldots, (w_n, P_n, F_n))$, where $n \geq 1$ is the number of membranes, V is a finite set of objects, μ is a membrane structure of n membranes with membrane 1 being the skin membrane, and for all $i, 1 \leq i \leq n$,

- $w_i \in V^\circ$ is the initial contents (state) of region i, that is, it is the finite multiset of all objects contained by region i;
- P_i is a finite set of communication rules associated to membrane i; they can be symport rules or antiport rules, with or without promoters or inhibitors, as above;
- $F_i \subseteq V^\circ$ is a finite set of finite multisets over V called the set of final states of region i; if $F_i = \emptyset$, then all the states of membrane i are considered to be final.

To simplify the notations we denote symport and antiport rules with or without promoters/inhibitors as $(x, in; y, out)|_Z$, $x, y \in V^\circ, Z \in \{z, \neg z \mid z \in V^\circ\}$ where we also allow x, y, z to be the empty string. If $y = \varepsilon$ or $x = \varepsilon$, then the notation above denotes the symport rule $(x, in)|_Z$ or $(y, out)|_Z$, respectively, and if $Z = \varepsilon$, then the rules above are without promoters or inhibitors.

The n-tuple of finite multisets of objects present in the n regions of the P automaton Γ describes a *configuration* of Γ; the n-tuple $(w_1, \ldots, w_n) \in (V^\circ)^n$ is the initial configuration.

The application of the rules can take place in a sequential, or in a maximally parallel manner.

The transition mapping of a P automaton is a partial mapping $\delta_X : V^\circ \times (V^\circ)^n \to 2^{(V^\circ)^n}$. These mappings are defined implicitly by the rules of the sets $P_i,\ 1 \leq i \leq n$. For a configuration $(u_1, \ldots, u_n)$,

$$(u'_1, \ldots, u'_n) \in \delta_X(u, (u_1, \ldots, u_n))$$

holds, that is, while reading the input $u \in V^\circ$ the automaton may enter the new configuration $(u'_1, \ldots, u'_n) \in (V^\circ)^n$, if there exist rules as follows.

- if $X = seq$, then for all $i, 1 \leq i \leq n$, there is a rule $(x_i, in; y_i, out)|_{Z_i} \in P_i$ with $z \subseteq u_i$ for $Z_i = z \in V^\circ$, and $z \cap u_i = \emptyset$ for $Z_i = \neg z, z \in V^\circ$, satisfying the conditions below, or
- if $X = par$, then for all $i, 1 \leq i \leq n$, there is a multiset of rules $R_i = \{\{r_{i,1}, \ldots, r_{i,m_i}\}\}$, where $r_{i,j} = (x_{i,j}, in; y_{i,j}, out)|_{Z_{i,j}} \in P_i$ with $z \subseteq u_i$ for $Z_{i,j} = z \in V^\circ$, and $z \cap u_i = \emptyset$ for $Z_{i,j} = \neg z, z \in V^\circ,\ 1 \leq j \leq m_i$, satisfying the conditions below, where $x_i,\ y_i$ denote the multisets $\bigcup_{1 \leq j \leq m_i} x_{i,j}$ and $\bigcup_{1 \leq j \leq m_i} y_{i,j}$, respectively. Furthermore, there is no rule occurrence $r \in P_j$, for any $j,\ 1 \leq j \leq n$, such that the rule multisets R'_i with $R'_i = R_i$ for $i \neq j$ and $R'_j = \{\{r\}\} \cup R_j$, also satisfy the conditions.

The conditions are given as

1. $x_1 = u$, and
2. $\bigcup_{parent(j)=i} x_j \cup y_i \subseteq u_i,\ 1 \leq i \leq n$,

and then the new configuration is obtained by

$$u'_i = u_i \cup x_i - y_i \cup \bigcup_{parent(j)=i} y_j - \bigcup_{parent(j)=i} x_j,\ 1 \leq i \leq n.$$

We define the sequence of multisets of objects accepted by the P automaton as the sequence of input multisets which appear in the environment and are consumed by the skin membrane in each computational step while the system reaches a final state, a configuration where for all j with $F_j \neq \emptyset$, the contents $u_j \in V^\circ$ of membrane j is "final", i.e., $u_j \in F_j$.

Let us extend δ_X to $\bar{\delta}_X,\ X \in \{seq, par\}$, a function mapping $(V^\circ)^*$, the sequences of finite multisets over V, and $(V^\circ)^n$, the configurations of Γ, to new configurations. We define $\bar{\delta}_X$ as

1. $\bar{\delta}_X(v, (u_1, \ldots, u_n)) = \delta_X(v, (u_1, \ldots, u_n)),\ v, u_i \in V^\circ,\ 1 \leq i \leq n$, and
2. $\bar{\delta}_X((v_1) \ldots (v_{s+1}), (u_1, \ldots, u_n)) = \bigcup \delta_X(v_{s+1}, (u'_1, \ldots, u'_n))$
   for all $(u'_1, \ldots, u'_n) \in \bar{\delta}_X((v_1) \ldots (v_s), (u_1, \ldots, u_n)),\ v_j, u_i, u'_i \in V^\circ$,
   $1 \leq i \leq n,\ 1 \leq j \leq s+1$.

Note that we use brackets in the multiset sequence $(v_1) \ldots (v_{s+1}) \in (V^\circ)^*$ in order to distinguish it from the multiset $v_1 \cup \ldots \cup v_{s+1} \in V^\circ$.

Let Γ be a P automaton as above with initial configuration $(w_1, \ldots, w_n)$, let Σ be an alphabet, and let $f : V^\circ \longrightarrow \Sigma \cup \{\varepsilon\}$ be a mapping with $f(x) = \varepsilon$ if and only if $x = \emptyset$.

The *language accepted by* Γ is defined as

$$L_X(\Gamma, f) = \{f(v_1) \dots f(v_s) \in \Sigma^* \mid (u_1, \dots, u_n) \in \bar{\delta}_X((v_1) \dots (v_s), (w_1, \dots, w_n)) \text{ with } u_j \in F_j \text{ for all } j \text{ with } F_j \neq \emptyset,\ 1 \leq j \leq n,\ 1 \leq s\},$$

for $X \in \{seq, par\}$. Obviously, the choice of the mapping f is essential. It has to be "easily" computable because the power of the P automaton should be provided by the underlying membrane system and not by f itself. The notion of "easiness", however, greatly depends on the context we are working in, so we do not give it a general specification here.

3 Related Results

P automata were already used to characterize interesting language classes. If the rule application is sequential, then the "problem" of mapping the set of possible input multisets to a finite alphabet does not appear since the number of possible inputs is finite. If $\{v_1, \dots, v_t\}$, $v_i \in V^\circ$ are the different multisets appearing in rules $(v_i, in; u_i, out) \in P_1,\ 1 \leq i \leq t$, then $\Sigma = \{a_1, \dots, a_t\}$, and $f(v_i) = a_i$ is a natural correspondence between the two sets.

In [2], P automata with sequential rule application were shown to characterize a language class which is strictly included in the class of languages accepted by Turing machines reading the separate input tape one way, and using logarithmic space for additional computations on the work-tapes. This sub-logarithmic-space language class still contains interesting languages, $\{a^n b^n c^n \mid n \geq 1\}$, for example.

If the rule application of P automata is maximally parallel, the situation is more complex. A mapping between the potentially infinite set of input multisets and a finite alphabet have to be specified. In [2] it was shown that if rules of the type (v, in) are not allowed in the skin membrane, and if the mapping $f : V^\circ \to \Sigma \cup \{\varepsilon\}$ is specified in such a way that $f(x) = \varepsilon$ implies $x = \emptyset$ and f is linear space computable by a Turing machine, then P automata accept exactly the class of context-sensitive languages. Moreover, it is also shown that all context-sensitive languages can be accepted using an f which maps an input multiset $x \in V^\circ$ nondeterministically to any element of $\{a \in V \mid a \in supp(x)\}$ and $f(\emptyset) = \varepsilon$, thus, it is also possible to give a characterization of context-sensitive languages in terms of P automata, which is "easy" to check, that is, to verify if a given systems conforms to this specification or not.

The authors of [4] also characterize the class of context-sensitive languages, using variants of so called symport/antiport acceptors called exponential symport/antiport acceptors. These systems have a separate terminal alphabet $\Sigma \subseteq V$, and the input is given as a sequence of terminal symbols provided by the environment. The symbols in this sequence of terminals have to be consumed by the symport/antiport rules of the system during the computation in the order they appear in the sequence. If the special end-marker is reached and the system halts, the string is accepted.

The results of [4] show that certain syntactic restrictions, that is, restrictions on the forms of the symport/antiport rules used by the system, can be given in such a way that the restricted exponential symport/antiport acceptors accept exactly the class of context-sensitive languages. Moreover, it is also shown that the number of nonterminal symbols in V with a fixed number of membranes, or the number of membranes with a fixed number of nonterminals, induces an infinite hierarchy of accepted languages.

4 P Automata for Context-free Languages

Now we introduce a restricted variant of P automata with parallel rule application which, as we show below, characterize the class of context-free languages.

Definition 1 A P stack-automaton is a construct $\Pi = (V, \mu, (w_1, P_1, F_1), \ldots, (w_n, P_n, F_n))$, a P automaton where V can be partitioned into disjoint subsets as

$$V = \Sigma \cup \{b\} \cup C \cup D,$$

the initial multisets are of the form

$$w_1 = b^i c \text{ for some } i \geq 0,\ c \in C,$$
$$w_j \in D^\circ \text{ for all } 2 \leq j \leq n,$$

the sets of final states are finite sets of finite multisets over D,

$$F_i \subset D^\circ \text{ for } 1 \leq i \leq n,$$

and there is a fixed positive integer k, such that all the rules of are one of the forms below.

For P_1:

- (a, out) where $a \in \Sigma$,
- $(b^{k^i}, in; b^k, out)|_c$ where $i \geq 0,\ c \in C$, and
- $(ab^l c_1, in; b^m c_2, out)$ where $a \in \Sigma,\ c_1, c_2 \in C,\ l, m \geq 0$,

and for $P_i,\ 2 \leq i \leq n$:

- $(b^i c^j, in; u, out)$ where $i \geq 1$, $c \in C,\ j \geq 0$, and $u \in D^+$.

The language accepted by the P stack-automaton Π over the finite alphabet Σ is the language $L(\Pi, f)$ accepted by the P automaton Π with a function f defined in such a way that, as usual, the empty multiset, and only the empty multiset is mapped to ε, and for nonempty input multisets, the image depends only on the occurrence or non-occurrence of certain symbols, but not on their multiplicity. That is

$$f : V^\circ \to \Sigma \cup \{\varepsilon\},\ \text{where for any } x \neq \emptyset,\ f(x) \neq \varepsilon,$$

and moreover, for any $x, y \in V^\circ$,

$$supp(x) = supp(y) \text{ implies that } f(x) = f(y).$$

Thus, a P stack-automaton is a P automaton having rules of a restricted form and a special, very simple mapping. Initially, the skin membrane contains a number of copies of the object b, plus one object from the set C, the other membranes contain finite multisets of objects over the set D. The rules are applied until a final configuration, a certain combination of a finite number of elements from the set D, is reached. Note that it is impossible that all copies of b and of objects from C leave the system, thus, there must be regions where the sets of given final states are empty, that is, where any state is considered to be final.

Theorem 1. *A language is context-free if and only if it is accepted by a P stack-automaton.*

Proof. First we prove that every context-free language can be accepted by a P stack-automaton.

Let L be a context-free language and $M = (\Sigma, \Gamma, Q, \delta, q, X_1)$ be a pushdown automaton with input alphabet Σ, stack alphabet Γ, set of states Q, transition mapping δ, initial state $q \in Q$, and initial stack contents X_1, accepting $L \setminus \{\varepsilon\}$ with empty stack. Let us assume, without the loss of generality, that $\Gamma = \{X_1, X_2, \ldots, X_t\}$, $Q = \{q\}$, and the transitions of M are of the form

$$(q, \alpha) \in \delta(q, a, X) \text{ where } q \in Q,\ \alpha \in \Gamma^*,\ a \in \Sigma,\ X \in \Gamma.$$

(Such a pushdown automaton can be obtained from a context-free grammar in Greibach normal form, i.e., having rules $X \to a\alpha$ where X is a nonterminal, a is a terminal, and α is a not necessarily nonempty string of nonterminals.) Let us also assume, again without the loss of generality, that X_1 only occurs as the initial contents of the stack, and if $(q, \alpha) \in \delta(q, a, X_1)$, then $|\alpha| = 1$.

Let $k = |\Gamma| + 1$, and let $t = max(\{|\alpha| \mid (q_2, \alpha) \in \delta(q_1, a, X)\})$. We construct a P stack-automaton accepting the same language as M. Let

$$\Pi = (V, [\ [\ ]_2]_1, (w_1, P_1, \{E\}), (w_2, P_2, \emptyset))$$

be a P stack-automaton with

$$V = \Sigma \cup I \cup B \cup \{T, E\},$$

where $I = \{(i) \mid 0 \le i \le t\}$, $B = \{b\}$, and let

$$\begin{aligned}
w_1 &= b^k b(1),\\
P_1 &= \{(b^{k^i}, in; b^k, out)|_{(i)}\}\\
&\quad \cup \{(a(j)b^l, in; b^m(i), out) \mid (q, \alpha) \in \delta(q, a, X_m),\ l = val(\alpha), i = |\alpha|\}\\
&\quad \cup \{(a, out) \mid a \in \Sigma\},\\
\\
w_2 &= TE,\\
P_2 &= \{(b, in; T, out)\} \cup \{(b(i), in; E, out)\} \cup P_2'.
\end{aligned}$$

where

$$P_2' = \begin{cases} \{(b^k b(1), in; E, out)\} & \text{if } \varepsilon \in L(M), \\ \emptyset & \text{otherwise.} \end{cases}$$

A configuration (w, γ) of M corresponds to the configuration (u_1, u_2) where the k-ary notation of $|u_1|_b$ is the string $1i_1i_2 \dots i_s$ where $X_{i_1}X_{i_2} \dots X_{i_s} = \gamma$, X_{i_s} being the topmost symbol, the digits 10 (denoting the integer k) corresponding to $\gamma = \varepsilon$.

In the following we will denote by $string(n)$ the sequence of pushdown symbols corresponding to the integer $n \geq k$.

Let us assume that

$$u_1 = c(i)b^n \tag{1}$$

where $c \in \Sigma$, $(i) \in I$, and b^n corresponds to the string $\gamma \in \Gamma^*$, $string(n) = \gamma$. Since there is only one occurrence of an object from I is present, the rules which could be used in the skin region are of the form

$$(b^{k^i}, in; b^k, out)|_{(i)} \text{ and } (a(j)b^l, in; b^m(i), out) \tag{2}$$

where there exists a transition among the transitions of the pushdown automaton M

$$tr : (q, \alpha) \in \delta(q, a, X_m),\ l = val(\alpha), i = |\alpha|.$$

If the application of two such rules consumes all the b objects of the skin region, then $string(n) = \gamma = \gamma' X_m$, and after the application of the rules we obtain a configuration (u_1', u_2) where $u_1' = a(j)b^p$ with $p = ((n-m)/k)k^i$, that is, an object $a \in \Sigma$ (and a number of b-s) has entered the system, the new string corresponding to the new number of b objects is $string(p) = \gamma'\alpha$, where $\gamma'\alpha$ is the new contents of the stack of M after executing the transition tr above.

If no rule-pair of the form (2) can be applied in the skin region or a rule-pair is applied in a way which does not consume all b objects, that is, when no transition of M is simulated, then, due to the maximal parallelism of rule application, a rule $(b, in; T, out)$ is applied in the second region sending the trap object T to the skin region which makes it impossible for Π to reach the final configuration.

As we have seen, a configuration $C = (u_1, u_2)$ with u_1 of the form (1) of Π corresponds to a configuration of M, and any configuration of Π following C is either corresponding to a possible next configuration of M, or the computation of Π cannot produce any result. No we explain how the computation of Π is started and finished.

The initial configuration of Π is

$$(b^k b(1), TE)$$

and the initial transition of M is of the form $(q, X_i) \in \delta(q, a, X_1)$, so it is correctly simulated by the rules

$$(b^k, in; b^k, out)|_{(1)} \text{ and } (a(j)b^i, in; b(1), out),$$

so after the first computational step, we either obtain a configuration with the trap symbol in the first region, or

$$(b^k b^i(j), TE).$$

This is a configuration corresponding to the configuration of M after the initial transition.

Let us assume now that Π is in a configuration

$$(vb^n(0), u_2)$$

where $string(n) = X_m$ and a possible transition of M is

$$(q, \varepsilon) \in \delta(q, a, X_m)$$

which finishes the computation by emptying the stack. This means that the rule-pair

$$(b, in; b^k, out)|_{(0)} \text{ and } (a(j), in; b^m(0), out)$$

is present and applicable in the first region, producing

$$(ab(j), u_2).$$

Now the computation can be finished by applying the rule (a, out) in the first region, and the rule $(b(j), in; E, out)$ in the second region, producing the final configuration

$$(E, b(j)u_2')$$

where $Eu_2' = u_2$.

As we have seen, Π either simulates a computation of M (or accepts the empty word, if $\varepsilon \in L$), or does not reach the final configuration.

To show that all languages accepted by P stack-automata are context-free, we present the following, rather informal argument.

Looking at the restrictions which define a P stack-automaton, we can notice that the regions other than the skin region, can only contain a limited number of objects which means that they can be in a finite number of different configurations. The skin region can also contain a finite number of objects other than b, thus, to record a configuration of the P stack-automaton, we need to record the potentially unbounded number of b-s in the skin membrane, plus a finite set of states to record the distribution of the objects in the other regions.

If we look at the rules of the skin region, we can also notice that the number of b-s can be recorded with a stack, thus, that a usual pushdown automaton is able to simulate the work of the P stack-automaton. To see this, consider the rules

$$(b^{k^i}, in; b^k, out)|_c \text{ where } i \geq 0,\ c \in C, \tag{3}$$

and

$$(ab^l c_1, in; b^m c_2, out) \text{ where } a \in \Sigma,\ c_1, c_2 \in C,\ l, m \geq 0. \tag{4}$$

Since there is at most one occurrence of one object from C which is present in the skin region, the rules of type (3) multiply the number of b-s by k^{i-1}, thus, if we consider the k-ary number denoting the number of b-s, the rules of type (3) append $i-1$ zeroes to the end of the string of k-ary digits (or, if $i=0$, add the value expressed by the last digit to the value obtained by erasing it from the end of the string). Since the length of the manipulated suffix of the digit string is bounded, the string of digits can be stored in a stack by introducing $k-1$ symbols to denote the digits. Moreover, since there is only one copy of a rule of type (4) can be used (because there is at most one copy of $c \in C$ is present), the changes of the number of b-s caused by these rules can be simulated using the finite control by modifying a bounded number of topmost symbols of the stack.

5 Conclusion

We have considered a problem left open in [4], namely the problem of characterizing the class of context-free languages in terms of symport/antiport P systems. We have introduced a restricted class of P automata called P stack-automata and shown that they accept exactly the class of context-free languages.

References

1. E. Csuhaj-Varjú, Gy. Vaszil: P automata. In *Pre-Proceedings of the Workshop on Membrane Computing WMC-CdeA 2002* (Gh. Păun, C. Zandron, eds.), Curtea de Argeş, Romania, August 19-23, 2002. Pub. No. 1 of MolCoNet-IST-2001-32008 (2002) 177–192, and also in *Membrane Computing* (Gh. Păun, G. Rozenberg, A Salomaa, C. Zandron, eds.), LNCS 2597, Springer, Berlin, 2003, 219–233.
2. E. Csuhaj-Varjú, O.H. Ibarra, Gy. Vaszil: On the computational complexity of P automata. In *DNA 10, Tenth International Meeting on DNA Computing* June 7-10, 2004, University of Milano-Bicocca, Preliminary Proceedings (C. Ferretti, G. Mauri, C. Zandron, eds.), University of Milano-Bicocca, 2004, 97–106.
3. C. Martín-Vide, A. Păun, Gh. Păun: On the power of P systems with symport rules. *Journal of Universal Computer Science*, 8 (2002), 317–331.
4. O.H. Ibarra, Gh. Păun: Characterizations of context-sensitive languages and other language classes in terms of symport/antiport P systems. *Theoretical Computer Science*, 2006.
5. A. Păun, Gh. Păun: The power of communication: P systems with symport/antiport. *New Generation Computing*, 20, 3 (2002), 295–306.
6. Gh. Păun: *Membrane Computing: An Introduction.* Springer, Berlin, 2002.
7. G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages.* Springer, Berlin, vol. 1-3, 1997.
8. The P Systems Web Page: `http://psystems.disco.unimib.it/`