Characterizations of Catalytic Membrane Computing Systems [⋆]

(Extended Abstract)

Oscar H. Ibarra[1], Zhe Dang[2], Omer Egecioglu[1], and Gaurav Saxena[1]

[1] Department of Computer Science
University of California
Santa Barbara, CA 93106, USA
[2] School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164, USA

Abstract. We look at 1-region membrane computing systems which only use rules of the form $Ca \to Cv$, where C is a catalyst, a is a noncatalyst, and v is a (possibly null) string of noncatalysts. There are no rules of the form $a \to v$. Thus, we can think of these systems as "purely" catalytic. We consider two types: (1) when the initial configuration contains only one catalyst, and (2) when the initial configuration contains multiple (not necessarily distinct) catalysts. We show that systems of the first type are equivalent to communication-free Petri nets, which are also equivalent to commutative context-free grammars. They define precisely the semilinear sets. This partially answers an open question in [19]. Systems of the second type define exactly the recursively enumerable sets of tuples (i.e., Turing machine computable). We also study an extended model where the rules are of the form $q : (p, Ca \to Cv)$ (where q and p are states), i.e., the application of the rules is guided by a finite-state control. For this generalized model, type (1) as well as type (2) with some restriction correspond to vector addition systems.

Keywords: membrane computing, catalytic system, semilinear set, vector addition system, reachability problem.

1 Introduction

In recent years, there has been a burst of research in the area of membrane computing [16], which identifi es an unconventional computing model (namely a P system) from natural phenomena of cell evolutions and chemical reactions [2]. Due to the built-in nature of maximal parallelism inherent in the model, P systems have a great potential for implementing massively concurrent systems in an effi cient way, once future bio-technology (or silicon-technology) gives way to a practical bio-realization (or a chip-realization). In this sense, it is important to study the computing power of the model.

[⋆] This research was supported in part by NSF Grants IIS-0101134 and CCR02-08595. Contact author: Oscar H. Ibarra, Email: ibarra@cs.ucsb.edu, Fax: 805-893-8553.

Two fundamental questions one can ask of any computing device (such as a Turing machine) are: (1) What kinds of restrictions/variations can be placed on the device without reducing its computing power? (2) What kinds of restrictions/variations can be placed on the device which will reduce its computing power? For Turing machines, the answer to (1) is that Turing machines (as well as variations like multitape, nondeterministic, etc.) accept exactly the recursively enumerable (r.e.) languages. For (2), there is a wide spectrum of well-known results concerning various sub-Turing computing models that have been introduced during the past half century – to list a few, there are finite automata, pushdown automata, linearly bounded automata, various restricted counter automata, etc. Undoubtedly, these sub-Turing models have enhanced our understanding of the computing power of Turing machines and have provided important insights into the analysis and complexity of many problems in various areas of computer science. We believe that studying the computing power of P systems would lend itself to the discovery of new results if a similar methodology is followed. Indeed, much research work has shown that P systems and their many variants are universal (i.e., equivalent to Turing machines) [4, 16, 17, 3, 6, 8, 19] (surveys are found in [12, 18]). However, there is little work in addressing the sub-Turing computing power of restricted P systems. To this end, we present some new results in this paper, specifically focusing on catalytic P systems.

A P system S consists of a finite number of membranes, each of which contains a multiset of objects (symbols). The membranes are organized as a Venn diagram or a tree structure where one membrane may contain 0 or many membranes. The dynamics of S is governed by a set of rules associated with each membrane. Each rule specifies how objects evolve and move into neighboring membranes. The rule set can also be associated with priority: a lower priority rule does not apply if one with a higher priority is applicable. A precise definition of S can be found in [16]. Since, from a recent result in [19], P systems with one membrane (i.e., 1-region P systems) and without priority are already able to simulate two counter machines and hence universal [14], for the purposes of this paper, we focus on catalytic 1-region P Systems, or simply catalytic systems (CS's) [16, 19].

A CS S operates on two types of symbols: catalytic symbols called *catalysts* (denoted by capital letters C, D, etc) and noncatalytic symbols called *noncatalysts* (denoted by lower case letters a, b, c, d, etc). An evolution rule in S is of the form $Ca \rightarrow Cv$, where C is a catalyst, a is a noncatalyst, and v is a (possibly null) string (an obvious representation of a multiset) of noncatalysts. A CS S is specified by a finite set of rules together with an initial multiset (configuration) w_0, which is a string of catalysts and noncatalysts. As with the standard semantics of P systems [16], each evolution step of S is a result of applying all the rules in S in a maximally parallel manner. More precisely, starting from the initial configuration w_0, the system goes through a sequence of configurations, where each configuration is derived from the directly preceding configuration in one step by the application of a subset of rules, which are chosen nondeterministically. Note that a rule $Ca \rightarrow Cv$ is applicable if there is a C and an a in the preceding configuration. The result of applying this rule is the replacement of a by v. If there is another occurrence of C and another occurrence of a, then the same rule or another rule with Ca on the left hand side can be applied. We require that the chosen subset of rules to apply must be maximally parallel in the sense that no other applicable rule can be added to the subset. Configuration w is reachable if it appears in some execution sequence. w is halting if none of the rules is applicable. The set of all reachable configurations is denoted by $R(S)$. The set of all halting reachable configurations (which is a subset of $R(S)$) is denoted by $R_h(S)$.

We show that CS's, whose initial configuration contains only one catalyst, are equivalent to communication-free Petri nets, which are also equivalent to commutative context free grammars [5, 11]. They define precisely the semilinear sets. Hence $R(S)$ and $R_h(S)$ are semilinear. This partially answers an open problem in [19], where it was shown that when the initial configuration contains six catalysts, S is universal, and [19] raised the question of what is the optimal number of catalysts for universality. Our result shows that one catalyst is not enough. We also study an extended model where the rules are of the form $q: \ (p, \ Ca \rightarrow Cv)$ (where q and p are states), i.e., the application of the rules is guided by a finite-state control. For this generalized model, systems with one catalyst in its initial configuration as well as systems with multiple catalysts in its initial configuration but with some restriction correspond to vector addition systems.

We conclude this section by recalling the definitions of semilinear sets and Parikh maps [15].

Let $\mathbf{N}$ be the set of nonnegative integers and k be a positive integer. A set $S \subseteq \mathbf{N}^k$ is a *linear set* if there exist vectors $v_0, v_1, \ldots, v_t$ in $\mathbf{N}^k$ such that $S = \{v \mid v = v_0 + a_1 v_1 + \ldots + a_t v_t, \ a_i \in \mathbf{N}\}$. The vectors v_0 (referred to as the *constant vector*) and $v_1, v_2, \ldots, v_t$ (referred to as the *periods*) are called the *generators* of the linear set S. A set $S \subseteq \mathbf{N}^k$ is *semilinear* if it is a finite union of linear sets. The empty set is a trivial (semi)linear set, where the set of generators is empty. Every finite subset of $\mathbf{N}^k$ is semilinear – it is a finite union of linear sets whose generators are constant vectors. Clearly, semilinear sets are closed under union and projection. It is also known that semilinear sets are closed under intersection and complementation.

Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be an alphabet. For each string w in Σ^*, define the *Parikh map* of w to be $\psi(w) = (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_n})$, where $|w|_{a_i}$ is the number of occurrences of a_i in w. For a language (set of strings) $L \subseteq \Sigma^*$, the *Parikh map* of L is $\psi(L) = \{\psi(w) \mid w \in L\}$.

2 1-Region Catalytic Systems

In this section, we study 1-region membrane computing systems which use only rules of the form $Ca \rightarrow Cv$, where C is a catalyst, a is a noncatalyst, and v is a (possibly null) string of noncatalysts. Note that we do not allow rules of the form $a \rightarrow v$ as in a P System. Thus, we could think of these systems as "purely" catalytic. As defined earlier, we denote such a system by CS.

Let S be a CS and w be an initial configuration (string) representing a multiset of catalysts and noncatalysts. A configuration x is a reachable configuration if S can reach x starting from the initial configuration w. Call x a halting configuration if no rule is applicable on x. Unless otherwise specified, "reachable configuration" will mean any reachable configuration, halting or not. Note that a non-halting reachable configuration x is an intermediate configuration in a possibly infinite computation. We denote by

$R(S)$ the set of Parikh maps of reachable configurations with respect to noncatalysts only. Since catalysts do not change in a computation, we do not include them in the Parikh map. Also, for convenience, when we talk about configurations, we sometimes do not include the catalysts. $R(S)$ is called the reachability set of R. $R_h(S)$ will denote the set of all halting reachable configurations.

2.1 The Initial Configuration Has Only One Catalyst

In this subsection, we assume that the initial configuration of the CS has only *one* catalyst C.

A noncatalyst a is *evolutionary* if there is a rule in the system of the form $Ca \to Cv$; otherwise, a is *non-evolutionary*. Call a CS *simple* if each rule $Ca \to Cv$ has at most one evolutionary noncatalyst in v. Our first result shows that semilinear sets and simple CS's are intimately related.

Theorem 1. *1. Let $Q \subseteq \mathbf{N}^k$. If Q is semilinear, then there is a simple CS S such that Q is definable by S, i.e., Q is the projection of $R_h(S)$ on k coordinates.*
2. Let S be a simple CS. Then $R_h(S)$ and $R(S)$ are semilinear.

Later, in Section 4, we will see that, in fact, the above theorem holds for any CS whose initial configuration has only one catalyst.

Suppose that we extend the model of a CS so that the rules are now of the form $q : (p,\ Ca \to Cv)$, i.e., the application of the rules is guided by a finite-state control. The rule means that if the system is in state q, application of $Ca \to Cv$ will land the system in state p. We call this system a CS with states or CSS. In addition, we allow the rules to be prioritized, i.e., there is a partial order on the rules: A rule r' of lower priority than r cannot be applied if r is applicable. We refer to such a system as a CSSP. For both systems, the computation starts at (q_0, w), where q_0 is a designated start state, and w is the initial configuration consisting of catalyst C and noncatalysts. In Section 4, we will see that a CSS can define only a recursive set of tuples. In contrast, the following result shows that a CSSP can simulate a Turing machine.

Theorem 2. *Let S be a CSSP with one catalyst and two noncatalysts. Then S can simulate a Turing machine.*

Directly from Theorem 2, we have:

Corollary 1. *Let S be a CSSP with one catalyst and two noncatalysts. Then $R(S) \subseteq \mathbf{N}^2$ need not be a semilinear set.*

We will see later that in contrast to the above result, when the rules are not prioritized, i.e., we have a CSS S with one catalyst and two noncatalysts, $R(S)$ is semilinear.

2.2 The Initial Configuration Has Multiple Catalysts

In this subsection, we assume that initial configuration of the CS can have *multiple* catalysts.

In general, we say that a noncatalyst is k-bounded if it appears at most k times in any reachable configuration. It is bounded if it is k-bounded for some k.

Consider a CSSP whose initial configuration has multiple catalysts. Assume that except for one noncatalyst, all other noncatalysts are bounded or make at most r (for some fixed r) alternations between nondecreasing and nonincreasing multiplicity in any computation. Call this a reversal-bounded CSSP.

Corollary 2. *If S is a reversal-bounded CSSP, then $R_h(S)$ and $R(S)$ are semilinear.*

Without the reversal-bounded restriction, a CSSP can simulate a TM. In fact, a CS (with multiple catalysts in its initial configuration) can simulate a TM. It was shown in [19] that a CS augmented with noncooperating rules of the form $a \to v$, where a is a noncatalyst and v is a (possibly null) string of noncatalysts is universal in the sense that such an augmented system with 6 catalysts can define any recursively enumerable set of tuples. A close analysis of the proof in [19] shows that all the rules can be made purely catalytic (i.e., of the form $Ca \to Cv$) using at most 8 catalysts. Actually, this number 8 can be improved further using the newest results in [7]:

Corollary 3. *A CS with 7 catalysts can define any recursively enumerable set of tuples.*

There is another restriction on a CSSP S that makes it define only a semilinear set. Let T be a sequence of configurations corresponding to some computation of S starting from a given initial configuration w (which contains multiple catalysts). A noncatalyst a is *positive* on T if the following holds: if a occurs in the initial configuration or does not occur in the initial configuration but later appears as a result of some catalytic rule, then the number of occurrences (multiplicity) of a in any configuration after the first time it appears is at least 1. (There is no bound on the number of times the number of a's alternate between nondecreasing and nonincreasing, as long there is at least 1.) We say that a is *negative* on T if it is not positive on T, i.e., the number of occurrences of a in configurations in T can be zero. Any sequence T of configurations for which every noncatalyst is bounded or is positive is called a *positive computation*.

Corollary 4. *Any semilinear set is definable by a CSSP where every computation path is positive.*

Conversely, we have,

Corollary 5. *Let S be a CSSP. Suppose that every computation path of S is positive. Then $R_h(S)$ and $R(S)$ are semilinear.*

The previous corollary can further be strengthened.

Corollary 6. *Let S be a CSSP. Suppose we allow one (and only one) noncatalyst, say a, to be negative. This means that a configuration with a positive occurrence (multiplicity) of a can lead to a configuration with no occurrence of a. Suppose that every computation path of S is positive, except for a. Then $R_h(S)$ and $R(S)$ are semilinear.*

3 Characterizations in Terms of Vector Addition Systems

An n-dimensional *vector addition system* (VAS) is a pair $G = \langle x, W \rangle$, where $x \in \mathbf{N}^n$ is called the *start point* (or *start vector*) and W is a finite set of vectors in $\mathbf{Z}^n$, where $\mathbf{Z}$ is the set of all integers (positive, negative, zero). The *reachability set* of the VAS $\langle x, W \rangle$ is the set $R(G) = \{z \mid \text{ for some } j,\ z = x + v_1 + ... + v_j$, where, for all $1 \leq i \leq j$, each $v_i \in W$ and $x + v_1 + ... + v_i \geq 0\}$. The *halting reachability set* $R_h(G) = \{z \mid z \in R(G), z + v \not\geq 0$ for every v in $W\}$.

An n-dimensional *vector addition system with states* (VASS) is a VAS $\langle x, W \rangle$ together with a finite set T of transitions of the form $p \rightarrow (q, v)$, where q and p are states and v is in W. The meaning is that such a transition can be applied at point y in state p and yields the point $y + v$ in state q, provided that $y + v \geq 0$. The VASS is specified by $G = \langle x, T, p_0 \rangle$, where p_0 is the starting state.

The *reachability problem* for a VASS (respectively, VAS) G is to determine, given a vector y, whether y is in $R(G)$. The *equivalence problem* is to determine given two VASS (respectively, VAS) G and G', whether $R(G) = R(G')$. Similarly, one can define the reachability problem and equivalence problem for halting configurations.

We summarize the following known results concerning VAS and VASS [20, 9, 1, 10, 13]:

Theorem 3. *1. Let G be an n-dimensional VASS. We can effectively construct an $(n+3)$-dimensional VAS G' that simulates G.*
2. If G is a 2-dimensional VASS G, then $R(G)$ is an effectively computable semilinear set.
3. There is a 3-dimensional VASS G such that $R(G)$ is not semilinear.
4. If G is a 5-dimensional VAS G, then $R(G)$ is an effectively computable semilinear set.
5. There is a 6-dimensional VAS G such that $R(G)$ is not semilinear.
6. The reachability problem for VASS (and hence also for VAS) is decidable.
7. The equivalence problem for VAS (and hence also for VASS) is undecidable.

Clearly, it follows from part 6 of the theorem above that the halting reachability problem for VASS (respectively, VAS) is decidable.

3.1 The Initial Configuration Has Only One Catalyst

We first consider CSS (i.e., CS with states) whose initial configuration has only one catalyst. There is an example of a 3-dimensional VASS G in [10] such that $R(G)$ is not semilinear: $G =< x, T, p >$, where $x = (0, 0, 1)$, and the transitions in T are:

$p \rightarrow (p, (0, 1, -1))$ $p \rightarrow (q, (0, 0, 0))$ $q \rightarrow (q, (0, -1, 2))$ $q \rightarrow (p, (1, 0, 0))$ Thus, there are only two states p and q. The following was shown in [10]:

1. (x_1, x_2, x_3) is reachable in state p if and only if $0 < x_2 + x_3 \leq 2^{x_1}$.
2. (x_1, x_2, x_3) is reachable in state q if and only if $0 < 2x_2 + x_3 \leq 2^{x_1+1}$.

Hence $R(G)$ is not semilinear. From this example, we can show:

Corollary 7. *There is CSS S with 1 catalyst, 3 noncatalysts, and two states such that $R(S)$ is not semilinear.*

In fact, as shown below, each CSS corresponds to a VASS and vice versa.

Lemma 1. *1. Let S be a CSS. We can effectively construct a VASS G such that $R(G) = R(S)$. 2. Every VASS can be simulated by a CSS.*

From Theorem 3 part 6, we have:

Corollary 8. *The reachability problem for CSS is decidable.*

Clearly a reachable configuration is halting if no rule is applicable on the configuration. It follows from the above result that the halting reachability problem (i.e., determining if a configuration is in $R_h(S)$) is also decidable.

A VASS is *communication-free* if for each transition $q \to (p, (j_1, ..., j_k))$ in the VASS, at most one j_i is negative, and if negative its value is -1. From Lemma 1 and the observation that the VASS constructed for the proof of Lemma 1 can be made communication-free, we have:

Theorem 4. *The following systems are equivalent in the sense that each system can simulate the others: CSS, VASS, communication-free VASS.*

Now consider a communication-free VASS without states, i.e., a VAS where in every transition, at most one component is negative, and if negative, its value is -1. Call this a *communication-free VAS*. Communication-free VAS's are equivalent to communication-free Petri nets, which are also equivalent to commutative context-free grammars [5, 11]. It is known that they have effectively computable semilinear reachability sets [5]. It turns out that communication-free VAS's characterize CS's.

Theorem 5. *Every communication-free VAS G can be simulated by a CS, and vice versa.*

Corollary 9. *If S is a CS, then $R(S)$ and $R_h(S)$ are effectively computable semilinear sets.*

The following is obvious, as we can easily construct a VAS from the specification of the linear set.

Corollary 10. *If Q is a linear set, then we can effectively construct a communication-free VAS G such that $R(G) = Q$. Hence, every semilinear set is a union of the reachability sets of communication-free VAS's.*

From the NP-completeness of the reachability problem for communication-free Petri nets (which are equivalent to commutative context-free grammars) [11, 5], we have:

Corollary 11. *The reachability problem for CS is NP-complete.*

We have already seen that a CSS S with prioritized rules (CSSP) and with two noncatalysts can simulate a TM (Theorem 2); hence $R(S)$ need not be semilinear. Interestingly, if we drop the requirement that the rules are prioritized, such a system has a semilinear reachable set.

Corollary 12. *Let S be a CSS with two noncatalysts. Then $R(S)$ and $R_h(S)$ are effectively computable semilinear sets.*

Open Problem: Suppose S has only rules of the form $Ca \to Cv$ whose initial configuration has exactly one catalyst. Suppose the rules are prioritized. How is $R(S)$ related to VASS?

3.2 The Initial Configuration Has Multiple Catalysts

We have seen that a CS with multiple catalysts can simulate a TM. Consider the following restricted version: Instead of "maximal parallelism" in the application of the rules at each step of the computation, we only allow "limited parallelism" by organizing the rules to apply in one step to be in the following form (called a matrix rule):

$(D_1b_1 \to D_1v_1, ..., D_sb_s \to D_sv_s)$

where the D_i's are catalysts (need not be distinct), the b_i's are noncatalysts (need not be distinct), the v_i's are strings of noncatalysts (need not be distinct), and s is the degree of the matrix. The matrix rules in a given system may have different degrees. The meaning of a matrix rule is that it is applicable if and only if each component of the matrix is applicable. The system halts if no matrix rule is applicable. Call this system a *matrix CS*, or MCS for short. We shall also consider MCS with states (called MCSS), where now the matrix rules have states and are of the form:

$p: \ (q, (D_1b_1 \to D_1v_1, ..., D_sb_s \to D_sv_s))$

Now the matrix is applicable if the system is in state p and all the matrix components are applicable. After the application of the matrix, the system enters state q.

Lemma 2. *Given a VAS (VASS) G, we can effectively construct an MCS (MCSS) S such that $R(S) = R(G) \times \{1\}$.*

Lemma 3. *Given an MCSS S over n noncatalysts, we can effectively construct an $(n+1)$-dimensional VASS G such that $R(S) = proj_n(R(G) \cap (\mathbf{N}^n \times \{1\}))$.*

The VASS in Lemma 3 can be converted to a VAS. It was shown in [10] that if G is an n-dimensional VASS with states $q_1, ..., q_k$, then we can construct an $(n+3)$-dimensional VAS G' with the following property: If the VASS G is at $(i_1, ..., i_n)$ in state q_j, then the VAS G' will be at $(i_1, ..., i_n, a_j, b_j, 0)$, where $a_j = j$ for $j = 1$ to k, $b_k = k+1$ and $b_j = b_{j+1}+k+1$ for $j = 1$ to $k-1$. The last three coordinates keep track of the state changes, and G' has additional transitions for updating these coordinates. However, these additional transitions only modify the last three coordinates. Define the finite set of tuples $F_k = \{(j, (k-j+1)(k+1)) \mid j = 1, ..., k\}$ (note that k is the number of states of G). Then we have:

Corollary 13. *Given an MCSS S over n noncatalysts, we can effectively construct an $(n+4)$-dimensional VAS G' such that $R(S) = proj_n(R(G') \cap (\mathbf{N}^n \times \{1\} \times F_k \times \{0\}))$, for some effectively computable k (which depends only on the number of states and number of rules in G).*

From Theorem 4, Lemmas 2 and 3, and the above corollary, we have:

Theorem 6. *The following systems are equivalent in the sense that each system can simulate the others: CSS, MCS, MCSS, VAS, VASS, communication-free VASS.*

Corollary 14. *It is decidable to determine, given an MCSS S and a configuration α, whether α is a reachable configuration (halting or not).*

Corollary 15. *It is decidable to determine, given an MCSS S and a configuration α, whether α is a halting reachable configuration.*

From Lemma 2 and Theorem 3 part 7, we have:

Corollary 16. *The equivalence and containment problems for MCSS are undecidable.*

4 Closure Properties

Let S be a catalytic system of any type introduced in the previous sections. For the purposes of investigating closure properties, we will say that S *defines* a set $Q \subseteq \mathbf{N}^k$ (or Q is *definable* by S) if $R_h(S) = Q \times \{0\}^r$ for some given r. Thus, the last r coordinates of the $(k+r)$-tuples in $R_h(S)$ are zero, and the first k-components are exactly the tuples in Q.

Fixed the noncatalysts to be $a_1, a_2, a_3, \ldots$. Thus, any system S has noncatalysts $a_1, \ldots, a_t$ for some t. We say that a class of catalytic systems of a given type is closed under:

1. *Intersection* if given two systems S_1 and S_2, which define sets $Q_1 \subseteq \mathbf{N}^k$ and $Q_2 \subseteq \mathbf{N}^k$, respectively, there exists a system S which defines $Q = Q_1 \cap Q_2$.
2. *Union* if given two systems S_1 and S_2, which define sets $Q_1 \subseteq \mathbf{N}^k$ and $Q_2 \subseteq \mathbf{N}^k$, respectively, there exists a system S which defines $Q = Q_1 \cup Q_2$
3. *Complementation* if given a system S which defines a set $Q \subseteq \mathbf{N}^k$, there exists a system S' which defines $Q' = \mathbf{N}^k - Q$.
4. *Concatenation* if given two systems S_1 and S_2, which define sets $Q_1 \subseteq \mathbf{N}^k$ and $Q_2 \subseteq \mathbf{N}^k$, respectively, there exists a system S which defines $Q = Q_1 Q_2$, where $Q_1 Q_2 = \{(i_1 + j_1, \ldots, i_k + j_k) \mid (i_1, \ldots, i_k) \in Q_1, (j_1, \ldots, j_k) \in Q_2\}$.
5. *Kleene* + if given a system S which defines a set $Q \subseteq \mathbf{N}^k$, there exists a system S' which defines $Q' = \bigcup_{n \geq 1} Q^n$.
6. *Kleene* * if given a system S which defines a set $Q \subseteq \mathbf{N}^k$, there exists a system S' which defines $Q' = \bigcup_{n \geq 0} Q^n$.

Other unary and binary operations can be defined similarly.

Theorem 7. *The class CS with only one catalyst in the initial configuration is closed under intersection, union, complementation, concatenation, and Kleene$^+$ (or Kleene*).*

Investigation of closure properties of other types of catalytic systems is a subject for future research.

Acknowledgment

We would like to thank Dung Huynh and Hsu-Chun Yen for their comments and for pointing out some of the references concerning vector addition systems. We also appreciate the comments and encouragement of Gheorghe Paun and Petr Sosik on this work.

References

1. H. G. Baker. Rabin's proof of the undecidability of the reachability set inclusion problem for vector addition systems. In *C.S.C. Memo 79, Project MAC, MIT*, 1973.
2. G. Berry and G. Boudol. The chemical abstract machine. In *POPL'90*, pages 81–94. ACM Press, 1990.
3. P. Bottoni, C. Martin-Vide, Gh. Paun, and G. Rozenberg. Membrane systems with promoters/inhibitors. *Acta Informatica*, 38(10):695–720, 2002.
4. J. Dassow and Gh. Paun. On the power of membrane computing. *Journal of Universal Computer Science*, 5(2):33–49, 1999.
5. J. Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. In *FCT'95*, volume 965 of *LNCS*, pages 221–232. Springer, 1995.
6. R. Freund and M. Oswald. P Systems with activated/prohibited membrane channels. In *WMC-CdeA'02*, volume 2597 of *LNCS*, pages 261–269. Springer, 2003.
7. R. Freund, M. Oswald, and P. Sosik. Reducing the number of catalysts needed in computationally universal P systems without priorities. In *the 5th Descriptional Complexity of Formal Systems Workshop (DFCS), July 12-14, 2003, Budapest, Hungary.*
8. P. Frisco and H. Jan Hoogeboom. Simulating counter automata by P Systems with symport/antiport. In *WMC-CdeA'02*, volume 2597 of *LNCS*, pages 288–301. Springer, 2003.
9. M. H. Hack. The equality problem for vector addition systems is undecidable. In *C.S.C. Memo 121, Project MAC, MIT*, 1975.
10. J. Hopcroft and J.-J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *TCS*, 8(2):135–159, 1979.
11. D.T. Huynh. Commutative grammars: The complexity of uniform word problems. *Information and Control*, 57:21–39, 1983.
12. C. Martin-Vide and Gh. Paun. Computing with membranes (P Systems): Universality results. In *MCU*, volume 2055 of *LNCS*, pages 82–101. Springer, 2001.
13. E. Mayr. Persistence of vector replacement systems is decidable. *Acta Informatica*, 15:309–318, 1981.
14. M. Minsky. Recursive unsolvability of Post's problem of Tag and other topics in the theory of Turing machines. *Ann. of Math.*, 74:437–455, 1961.
15. R. Parikh. On context-free languages. *Journal of the ACM*, 13:570–581, 1966.
16. Gh. Paun. Computing with membranes. *JCSS*, 61(1):108–143, 2000.
17. Gh. Paun. Computing with membranes (P Systems): A variant. *International Journal of Foundations of Computer Science*, 11(1):167–181, 2000.
18. Gh. Paun and G. Rozenberg. A guide to membrane computing. *TCS*, 287(1):73–100, 2002.
19. P. Sosik and R. Freund. P Systems without priorities are computationally universal. In *WMC-CdeA'02*, volume 2597 of *LNCS*, pages 400–409. Springer, 2003.
20. J. van Leeuwen. A partial solution to the reachability problem for vector addition systems. In *STOC'74*, pages 303–309.