Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 363 (2006) 149–161
Theoretical Computer Science
www.elsevier.com/locate/tcs

Deterministic catalytic systems are not universal

Oscar H. Ibarra[a,*,1], Hsu-Chun Yen[b,2]

[a]*Department of Computer Science, University of California, Santa Barbara, CA 93106, USA*
[b]*Department of Electrical Engineering, National Taiwan University, Taipei 106, Taiwan, ROC*

Abstract

We look at a 1-membrane catalytic P system with evolution rules of the form $Ca \to Cv$ or $a \to v$, where C is a catalyst, a is a noncatalyst symbol, and v is a (possibly null) string representing a multiset of noncatalyst symbols. (Note that we are only interested in the multiplicities of the symbols.) A catalytic system (CS) can be regarded as a *language acceptor* in the following sense. Given an input alphabet Σ consisting of noncatalyst symbols, the system starts with an initial configuration wz, where w is a fixed string of catalysts and noncatalysts not containing any symbol in z, and $z = a_1^{n_1} \cdots a_k^{n_k}$ for some nonnegative integers $n_1, \ldots, n_k$, with $\{a_1, \ldots, a_k\} \subseteq \Sigma$. At each step, a maximal multiset of rules is nondeterministically selected and applied in parallel to the current configuration to derive the next configuration (note that the next configuration is not unique, in general). The string z is accepted if the system eventually halts.

It is known that a 1-membrane CS is universal in the sense that any unary recursively enumerable language can be accepted by a 1-membrane CS (even by purely CSs, i.e., when all rules are of the form $Ca \to Cv$). A CS is said to be *deterministic* if at each step there is a *unique* maximally parallel multiset of rules applicable. It has been an open problem whether deterministic systems of this kind are universal. We answer this question negatively. We show that the membership problem for deterministic CSs is decidable. In fact, we show that the Parikh map of the language ($\subseteq a_1^* \cdots a_k^*$) accepted by any deterministic CS is a simple semilinear set which can be effectively constructed. Since nondeterministic 1-membrane CS acceptors (with two catalysts) are universal, our result gives the first example of a variant of P systems for which the nondeterministic version is universal, but the deterministic version is not.

We also show that for a deterministic 1-membrane CS using only rules of type $Ca \to Cv$, the set of reachable configurations from a given initial configuration is an effective semilinear set. The application of rules of type $a \to v$, however, is sufficient to render the reachability set nonsemilinear. Our results generalize to multimembrane deterministic CSs. We also consider deterministic CSs which allow rules to be prioritized and investigate three classes of such systems, depending on how priority in the application of the rules is interpreted. For these three prioritized systems, we obtain contrasting results: two are universal and one only accepts semilinear sets.

Keywords: Membrane computing; Deterministic catalytic system; Deterministic versus nondeterministic; Symport/antiport system; Counter machine; Semilinear set; Priority

* Corresponding author.
E-mail addresses: ibarra@cs.ucsb.edu (O.H. Ibarra), yen@cc.ee.ntu.edu.tw (H.-C. Yen).
[1] Research supported in part by NSF Grants CCR-0208595, CCF-0430945, IIS-0451097 and CCF-0524136.
[2] Research supported in part by NSC Grant 93-2213-E-002-003, Taiwan, ROC.

0304-3975/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2006.07.029

1. Introduction

There has been a great deal of research activities in the area of membrane computing (a branch of natural computing) initiated by Păun [10] in a seminal paper over 6 years ago (see also [11]). Membrane computing identifies an unconventional computing model, namely a P system, from natural phenomena of cell evolutions and chemical reactions. Due to the built-in nature of maximal parallelism inherent in the model, P systems have a great potential for implementing massively parallel systems in an efficient way that would allow us to solve currently intractable problems once future biotechnology (or silicon technology) gives way to a practical biorealization (or chip realization).

A P system is a computing model, which abstracts from the way the living cells process chemical compounds in their compartmental structure. Thus, regions defined by a membrane structure contain objects that evolve according to given rules. The objects can be described by symbols or by strings of symbols, in such a way that multisets of objects are placed in regions of the membrane structure. The membranes themselves are organized as a tree structure (this can be represented by a Venn diagram) where one membrane may contain other membranes. By using the rules in a nondeterministic, maximally parallel manner, transitions between the system configurations can be obtained. A sequence of transitions shows how the system is evolving. Various ways of controlling the transfer of objects from a region to another and applying the rules, as well as possibilities to dissolve, divide, or create membranes have been studied. P systems were introduced with the goal to abstract a new computing model from the structure and the functioning of the living cell (as a branch of the general effort of natural computing—to explore new models, ideas, paradigms from the way nature computes). Membrane computing has been quite successful: many models have been introduced, most of them Turing-complete and/or able to solve computationally intractable problems (NP-complete, PSPACE-complete) in a feasible time (polynomial), by trading space for time. (See the P system website at http://psystems.disco.unimib.it for a large collection of papers in the area, and in particular the monograph [12].)

In the standard semantics of P systems [11,12,14], each evolution step of a system P is a result of applying all the rules in P in a maximally parallel manner. More precisely, starting from the initial configuration, w, 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 multiset of rules, which are chosen nondeterministically. For example, a catalytic rule $Ca \to Cv$ in membrane m is applicable if there is a catalyst C and an object (symbol) a in the preceding configuration in membrane m. The result of applying this rule is the evolution of v from a. 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. Thus, in general, the number of times a particular rule is applied at anyone step can be unbounded. We require that the application of the rules is maximal: all objects, from all membranes, which *can be* the subject of local evolution rules *have to* evolve simultaneously. Configuration z is reachable (from the starting configuration) if it appears in some execution sequence; z is halting if no rule is applicable on z.

Two popular models of P systems are the catalytic system (CS) [11] and the symport/antiport system [9]. An interesting subclass of the latter was studied in [6]—each system is *deterministic* in the sense that the computation path of the system is unique, i.e., at each step of the computation, the maximal multiset of rules that is applicable is unique. [3] It was shown in [6] that any recursively enumerable unary language $L \subseteq o^*$ can be accepted by a deterministic 1-membrane symport/antiport system. Thus, for symport/antiport systems, the deterministic and nondeterministic versions are equivalent and they are universal. It also follows from the construction in [15] that for another model of P systems, called communicating P systems (CPs), the deterministic and nondeterministic versions are equivalent as both can accept any unary recursively enumerable language. However, the deterministic-versus-nondeterministic question was left open in [6] for the class of CSs (these systems have rules of the form $Ca \to Cv$ or $a \to v$), where the proofs of universality involve a high degree of parallelism [15,5]. For a discussion of this open question and its importance see [3,13]. We answer this question negatively in this paper. Since nondeterministic CSs are universal, our result also gives the first example of a variant of P systems for which the nondeterministic version is universal, but the deterministic version is not.

For a CS serving as a *language acceptor*, the system starts with an initial configuration wz, where w is a fixed string of catalysts and noncatalysts not containing any symbol in z, and $z = a_1^{n_1} \cdots a_k^{n_k}$ for some nonnegative integers $n_1, \ldots, n_k$, with $\{a_1, \ldots, a_k\}$ a distinguished subset of noncatalyst symbols (the input alphabet). At each step, a maximal

[3] *Deterministic P* systems were originally defined in [11], referring to those having in each moment at most one possible transition. There is a slightly different notion of 'determinism' reported in the literature [1] which does not require the maximally applicable multiset to be unique as long as the next configuration is unique. In this paper, we only consider the original notion of determinism given in [11].

multiset of rules are nondeterministically selected and applied in parallel to the current configuration to derive the next configuration (note that the next configuration is not unique, in general). The string z is accepted if the system eventually halts. Unlike nondeterministic 1-membrane CS acceptors (with two catalysts) which are universal, we are able to show using a graph-theoretic approach that the Parikh map of the language ($\subseteq a_1^* \cdots a_k^*$) accepted by any deterministic CS is a simple semilinear set which can also be effectively constructed. For deterministic 1-membrane CSs using only rules of type $Ca \to Cv$, we show the set of reachable configurations from a given initial configuration to be effectively semilinear. In contrast, the reachability set is no longer semilinear in general if rules of type $a \to v$ are also used. Our result generalizes to multimembrane CSs.

We also consider deterministic CSs which allow rules to be prioritized. Three such systems, namely, *statically prioritized*, *strongly prioritized*, and *weakly prioritized* CSs, are investigated. For statically prioritized systems, rules are divided into different priority groups, and if a rule in a higher priority group is applicable, then no rules from a lower priority group can be used. For both strongly prioritized and weakly prioritized systems, the underlying priority relation is a *strict partial order* (i.e., irreflexive, asymmetric, and transitive). Under the semantics of strong priority, if a rule with higher priority is used, then no rule of a lower priority can be used even if the two rules do not compete for objects. This notion of strong priority coincides with the semantics of the priority relation used in [11]. For weakly prioritized systems, a rule is applicable if it cannot be replaced by a higher priority one. For these three prioritized systems, we obtain contrasting results by showing that deterministic strongly and weakly prioritized CSs are universal, whereas statically prioritized systems only accept semilinear sets.

2. Nonuniversality of deterministic CSs

Consider a CS in which all rules are of the form: $Ca \to Cv$ or $a \to v$, where C is a catalyst, a is a noncatalyst symbol, and v is a (possibly null) string of noncatalyst symbols. Unless stated otherwise, we assume that CSs operate under the maximally parallel mode, i.e., at each step the maximal multiset of rules is applied. A CS is said to be *deterministic* if at each step, there is a *unique* maximally parallel multiset of rules applicable. A CS is referred to as a *purely* CS if only rules of the form $Ca \to Cv$ are used.

A *configuration* of a CS is a string of catalytic and noncatalytic symbols. (Note that we are only interested in the multiplicities of the symbols.) Given two configurations c and c', we write $c \xrightarrow{S} c'$ to denote that applying the multiset S at c yields c', and S is a maximally applicable multiset of rules at c. We also write $c \xrightarrow{S_1 \cdots S_k} c'$ to denote the reachability of c' from c through applying sequence $S_1 \cdots S_k$ of multisets of rules (or $c \xrightarrow{*} c'$ if the actual sequence is irrelevant). Given a configuration c , we write $\#_c$ to denote the Parikh map of c, and $\#_c(x)$ the number of occurrences of symbol x in c, where x is either a catalytic or a noncatalytic symbol. During the course of a computation, the number of occurrences of each catalytic symbol remains the same.

Next we recall the definition of a semilinear set [7]. Let N be the set of nonnegative integers and k be a positive integer. A subset R of N^k is a *linear set* if there exist vectors $v_0, v_1, \ldots, v_t$ in N^k such that $R = \{v \mid v = v_0 + m_1 v_1 + \cdots + m_t v_t,\ m_i \in N\}$. 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 R. The set $R \subseteq N^k$ is *semilinear* if it is a finite union of linear sets. The empty set is a trivial semilinear set. Every finite subset of N^k is semilinear—it is a finite union of linear sets whose generators are constant vectors. It is also clear that the semilinear sets are closed under (finite) union. It is also known that they are closed under complementation and intersection.

2.1. Deterministic purely CSs

We first consider deterministic purely CSs, i.e., all rules are of the form $Ca \to Cv$. Due to the nature of determinism as well as the number of catalysts being bounded, an infinite computation of a deterministic purely CS is 'periodic' in the sense stated in the following theorem.

Theorem 2.1. *Given a deterministic purely CS P and an initial configuration c_0, the following three statements are equivalent*:

(1) *P does not halt*,

(2) *there exist c and c' with $\#_{c'} \geqslant \#_c$ such that $c_0 \xrightarrow{*} c \xrightarrow{*} c'$*,

(3) *the computation of P is of the form $c_0 \overset{T_1\cdots T_r(S_1\cdots S_k)^\omega}{\to}$, where $T_1,\ldots, T_r, S_1,\ldots, S_k$ are multisets of rules.* (*That is, following a finite prefix the computation is 'periodic' with $S_1\cdots S_k$ repeating forever.*)

Proof. To proceed, we require the following claims.

Claim 1. *Suppose $c \overset{H}{\to} d, c' \overset{H'}{\to} d'$, and $H \subseteq H'$, where c, c', d, d' are configurations and H and H' are two multisets of rules. If rule $Ca \to Cv$ is in $H' - H$, then $\#_c(a) < \#_{c'}(a)$.*

Proof. The claim follows immediately from P being deterministic and operating under the maximally parallel mode. □

Claim 2. *Given a computation $c_1 \overset{H_1}{\to} c_2 \overset{H_2}{\to} \cdots c_{i-1} \overset{H_{i-1}}{\to} c_i$ and a configuration c'_1 with $\#_{c'_1} \geqslant \#_{c_1}$, then there exist multisets $H'_1, \ldots, H'_{i-1}$ and configurations $c'_2, \ldots, c'_i$ such that*

(i) $c'_1 \overset{H'_1}{\to} c'_2 \overset{H'_2}{\to} \cdots c'_{i-1} \overset{H'_{i-1}}{\to} c'_i$,
(ii) $H_j \subseteq H'_j, \forall j, 1 \leqslant j \leqslant i-1$, *and*
(iii) $\#_{c_j} \leqslant \#_{c'_j}, \forall j, 1 \leqslant j \leqslant i$.

Proof. The proof is done by induction on i.

Basis: The $i = 2$ case trivially holds.

Induction hypothesis: Assume that the assertion holds for $i = k$.

Induction step: Consider $c_1 \overset{H_1}{\to} c_2 \cdots c_k \overset{H_k}{\to} c_{k+1}$. Based on the induction hypothesis, there exist $c'_2, \ldots, c'_k, H'_1, \ldots, H'_{k-1}$ such that $c'_1 \overset{H'_1}{\to} c'_2 \overset{H'_2}{\to} \cdots \overset{H'_{k-1}}{\to} c'_k$ satisfies (ii) and (iii) above. Since $\#_{c'_k} \geqslant \#_{c_k}$, then all the rules in H_k are clearly applicable at c'_k. Since P is deterministic and operates under the maximally parallel mode, H_k must also be applied in its entirety in c'_k. Hence, $c'_k \overset{H'_k}{\to} c'_{k+1}$ must have $H_k \subseteq H'_k$. Following Claim 1, the existence of a rule $Ca \to Cv$ in $H'_k - H_k$ implies $\#_{c'_k}(a) > \#_{c_k}(a)$, meaning that the application of this additional rule only consumes an extra copy of symbol a in c'_k but not in c_k. Hence, $\#_{c_{k+1}} \leqslant \#_{c'_{k+1}}$. The assertion holds for $i = k+1$. □

Claim 3. *If $c_1 \overset{H_1\cdots H_k}{\to} c_2$, $\#_{c_2} \geqslant \#_{c_1}$, and $c_2 \overset{H_1\cdots H_k}{\to} c_3$, then it must be the case that $c_1 \overset{H_1\cdots H_k}{\to} c_2 \overset{H_1\cdots H_k}{\to} c_3 \cdots \overset{H_1\cdots H_k}{\to} c_i \cdots$, i.e., $H_1\cdots H_k$ repeats forever.*

Proof. The proof is done by contradiction. Let i and d be the smallest indices such that $c_1 \overset{H_1\cdots H_k}{\to} c_2 \overset{H_1\cdots H_k}{\to} c_3 \cdots \overset{H_1\cdots H_k}{\to} c_i \overset{H_1\cdots H'_d\cdots H'_k}{\to} c_{i+1}$ and $H'_d \neq H_d$, where $i \geqslant 3$ and $1 \leqslant d \leqslant k$. Consider the three consecutive segments $(c_{i-2} \overset{H_1\cdots H_{d-1}}{\to} e_{i-2} \overset{H_d\cdots H_k}{\to} c_{i-1})$, $(c_{i-1} \overset{H_1\cdots H_{d-1}}{\to} e_{i-1} \overset{H_d\cdots H_k}{\to} c_i)$, and $(c_i \overset{H_1\cdots H_{d-1}}{\to} e_i \overset{H'_d\cdots H'_k}{\to} c_{i+1})$. From Claim 2, we have $H_d \subseteq H'_d$. (Note that c'_1 in Claim 2 is in this case c_2.) This, together with the assumption that $H_d \neq H'_d$, implies the existence of a rule $Ca \to Cv$ in $H'_d - H_d$. Hence, $\#_{e_i}(a) > \#_{e_{i-1}}(a)$—Claim 1. Now observe that the same sequence of rule sets (i.e., $H_d \cdots H_k H_1 \cdots H_{d-1}$) is applied at both e_{i-2} and e_{i-1}, it follows that $\#_{e_i}(a) > \#_{e_{i-1}}(a) > \#_{e_{i-2}}(a)$. The set $H_d \cup \{Ca \to Cv\}$ is therefore applicable at e_{i-1} (since H_d was applied at e_{i-2})—a contradiction. □

We are now in a position to prove our theorem. We first show (1) $\Longrightarrow$ (2). Assume that P does not terminate. Let $c_0 \to c_1 \to \cdots \to c_l \to \cdots (l \in N)$ be an infinite computation. According to Dickson's lemma (see [4]), there exist $i < j$ such that $\#_{c_i} \leqslant \#_{c_j}$; hence, (2) holds.

Now we establish (2) $\Longrightarrow$ (3). Let $H_1 \cdots H_k$ be the sequence of rule sets such that $c_i \overset{H_1\cdots H_k}{\to} c_j$ and $\#_{c_i} \leqslant \#_{c_j}$. According to Claim 2, there are rule sets $H_1^t, H_2^t \cdots H_k^t$ and configurations $c_{j_t}, t \geqslant 1$, such that $c_i \overset{H_1\cdots H_k}{\to} c_j \overset{H_1^1\cdots H_k^1}{\to} c_{j_1} \overset{H_1^2\cdots H_k^2}{\to} c_{j_2} \cdots \overset{H_1^t\cdots H_k^t}{\to} c_{j_t} \cdots$. Furthermore, for all $1 \leqslant l \leqslant k$ and $1 \leqslant t$, $H_l^t \subseteq H_l^{t+1}$ and $\#_{c_{j_t}} \leqslant \#_{c_{j_{t+1}}}$. Since the number

of catalytic symbols (which bounds the degree of maximal parallelism) is a constant, for all $1 \leqslant l \leqslant k$ there must be a t_l such that $H_l^{t_l} = H_l^{t_l+1} = H_l^{t_l+2} = \cdots$. Choose t' to be the maximum among all t_l, such that $H_l^{t'} = H_l^{t'+1}$ for all $1 \leqslant l \leqslant k$. Now we have $c_{j_{t'}} \overset{H_1^{t'} \cdots H_k^{t'}}{\rightarrow} c_{j_{t'+1}} \cdots \overset{H_1^{t'} \cdots H_k^{t'}}{\rightarrow} c_{j_{t'+2}}$. By letting $S_l = H_l^{t'}$, $1 \leqslant l \leqslant k$, Claim 3 guarantees that $S_1 \cdots S_k$ repeat forever at $c_{j_{t'}}$. Therefore, (3) holds.

(3) $\Longrightarrow$ (1) is trivial. This completes the proof of the theorem. □

From (3) of Theorem 2.1, we immediately have:

Corollary 2.2. *Given a deterministic purely CS P and an initial configuration c_0, the reachability set $\{\#_c \mid c_0 \overset{*}{\rightarrow} c\}$ is semilinear.*

Corollary 2.3. *Given a deterministic purely CS P and an initial configuration c_0, the problem of determining whether P halts is decidable.*

Proof. According to Theorem 2.1, the computation of P does not halt iff there exist c and c' with $\#_{c'} \geqslant \#_c$ such that $c_0 \overset{*}{\rightarrow} c \overset{*}{\rightarrow} c'$. By simulating the computation of P, either it halts, or eventually c and c' are found (due to Dickson's lemma) which implies nontermination. Hence, the problem is decidable. □

Now consider the case when CSs serve as *language acceptors*. Consider a CS P with initial configuration $wo_1^{n_1} \cdots o_k^{n_k}$, where noncatalytic symbols $o_1, \ldots, o_k$ are distinguished input symbols not in w, and w is a fixed string (independent of $n_1, \ldots, n_k$) not containing any of the symbols $o_1, \ldots, o_k$. The word $o_1^{n_1} \cdots o_k^{n_k}$ is accepted if P halts. It is known [15] that even for $k = 1$, any unary RE language can be accepted by the a purely CS operating in a nondeterministic manner. Hence, nondeterministic purely CSs are universal. Surprisingly, however, deterministic purely CSs are not universal as the following result indicates.

Theorem 2.4. *Deterministic purely CSs are not universal.*

Proof. Based on Corollary 2.3, there exists a decision procedure A such that given a deterministic CS P and an input $wo_1^{n_1} \cdots o_k^{n_k}$, A can tell whether P halts on $wo_1^{n_1} \cdots o_k^{n_k}$ or not. It is obvious that a decision procedure A_P can be constructed such that A_P accepts $o_1^{n_1} \cdots o_k^{n_k}$ iff A answers 'yes' on input $(P, wo_1^{n_1} \cdots o_k^{n_k})$. Hence, the language $\{o_1^{n_1} \cdots o_k^{n_k} \mid A_P \text{ accepts } o_1^{n_1} \cdots o_k^{n_k}\} = \{o_1^{n_1} \cdots o_k^{n_k} \mid P \text{ halts on } wo_1^{n_1} \cdots o_k^{n_k}\}$ is recursive. □

2.2. Deterministic CSs

Now we consider the full class of deterministic CSs, where the rules are of the form $Ca \rightarrow Cv$ or $a \rightarrow v$. Intuitively, what makes the reachability set of a deterministic purely *CS* 'simpler' is that any infinite computation of such a system is *periodic* in the sense described in (3) of Theorem 2.1. Such a periodic behavior is partly due to the fact that the maximum degree of parallelism during the course of the computation of a deterministic purely CS is bounded by the number of catalytic symbols in the initial configuration. Note, however, that the degree of parallelism becomes unbounded if the CS uses rules of type $a \rightarrow v$. In fact, the semilinearity result no longer holds for the full class of deterministic CSs as the following example indicates. It is interesting to note that the degree of parallelism in this example is unbounded.

Example 2.5. Consider a CS with only one rule $a_1 \rightarrow a_1 a_1$ and initial configuration a_1. Then the Parikh map of the set of all reachable configurations is $\{2^n | n \geqslant 1\}$, which is clearly not semilinear.

Although the reachability set of a deterministic (not necessarily purely) CS is not semilinear in general, being deterministic does make the computational power of the model weaker than its nondeterministic counterpart. In what follows, we propose a graph-theoretic approach for reasoning about the behaviors of deterministic CSs.

Consider a deterministic CS P, in which $\{C_1, \ldots, C_k\}$ is the set of catalytic symbols, and $\Sigma = \{a_1, \ldots, a_m\}$ is the set of noncatalytic symbols. Let c_0 be the initial configuration which contains (possibly multiple copies of) C_i, $\forall 1 \leqslant i \leqslant k$.

Two rules r_1 and r_2 are said to be *in conflict* if one of the following holds:
- $r_1 : C_i a_t \to C_i w_1,\ r_2 : C_j a_t \to C_j w_2$, and either $w_1 \neq w_2$ or $i \neq j$,
- $r_1 : C_i a_t \to C_i w_1,\ r_2 : a_t \to w_2$,
- $r_1 : a_t \to w_1,\ r_2 : a_t \to w_2$, and $w_1 \neq w_2$.

In each of the above, rules r_1 and r_2 compete for the same noncatalyst a_t. (In this case, a_t is said to be *involved* in two conflicting rules.) At any point in time, a deterministic CS can never enable a rule that is in conflict with another rule. Under the unprioritized mode, conflicting rules can be removed without affecting the computation of the CS, regardless of the initial configuration. Note that rules $C_1 a_1 \to C_1 w_1$ and $C_1 a_2 \to C_1 w_2$ are not conflicting rules, and in fact, the absence of a_1 (resp., a_2) makes $C_1 a_2 \to C_1 w_2$ (resp., $C_1 a_1 \to C_1 w_1$) applicable.

In what follows, we employ a graph-theoretic approach to reasoning about the behaviors of deterministic CSs. We construct a directed labelled graph $G_{P,c_0} = (V, E)$, called the *execution graph*, such that $V = \Sigma$ and $E = \{(a_i, a_j)_r \mid$ there exists a rule r of the form $C_t a_i \to C_t w$ or $a_i \to w$, such that a_j is in w, and a_i is not involved in any conflicting rules$\}$. (The subscript r is the label of edge (a_i, a_j). We also write $a_i \overset{r}{\to} a_j$.) A careful examination of G_{P,c_0} reveals an important property: for each node a_i, the outgoing edges of a_i (if they exist) are of the same label.

To set the stage for the nonuniversality result of deterministic CSs, we require the following lemma.

Lemma 2.6. *Consider a deterministic CS P with $\{C_1, \ldots, C_k\}$ and $\{a_1, \ldots, a_m\}$ as the sets of catalysts and noncatalysts, respectively. Let c_0 be the initial configuration. Then*:
(1) *P does not halt on c_0 iff there is a reachable loop from some node a_{i_0} with $\#_{c_0}(a_{i_0}) > 0$ in G_{P,c_0}.*
(2) *Let c_0' be a configuration such that $(\forall 1 \leqslant i \leqslant k, \#_{c_0'}(C_i) = \#_{c_0}(C_i))$ and $(\forall 1 \leqslant j \leqslant m, (\#_{c_0}(a_j) > 0 \Longrightarrow \#_{c_0'}(a_j) = 1) \wedge (\#_{c_0}(a_j) = 0 \Longrightarrow \#_{c_0'}(a_j) = 0))$. Then P halts on c_0 iff P halts on c_0'.*
(3) *The problem of determining whether P halts on c_0 is decidable in polynomial time.*

Proof. We first show (1). The *only-if* part is obvious since without a reachable loop in G_{P,c_0}, the computation is clearly finite. For the *if* part, it suffices to show that for any path $a_{i_0} \overset{r_{i_1}}{\to} a_{i_1} \cdots \overset{r_{i_{h-1}}}{\to} a_{i_h}$ $(\#_{c_0}(a_{i_0}) > 0)$ in G_{P,c_0}, the first $h-1$ steps of the computation $c_0 \overset{H_1}{\to} c_1 \cdots \overset{H_{h-1}}{\to} c_h$ of P must have $\forall 0 \leqslant j \leqslant h, \#_{c_j}(a_{i_j}) > 0$ and $\forall 1 \leqslant j < h, r_{i_j} \in H_j$. The key to the above claim is that for each node a_{i_j} $(0 \leqslant j < h)$ along the above path, a_{i_j}'s outgoing edges are uniquely labelled with r_{i_j}, meaning that rule r_{i_j} is applicable when symbol a_{i_j} is present in configuration c_j of P. The if part can then be easily shown by induction.

(2) clearly follows from (1). The following *token game* also provides an intuitive idea for both (1) and (2). With the help of G_{P,c_0}, a way to visualize the computation of P is to place $\#_{c_0}(a_i)$ *tokens* in node a_i of G_{P,c_0} initially, and during the course of P's computation, if a token is in node a_i and '$C_t a_i \to C_t a_{j_1} \cdots a_{j_f}$' (or '$a_i \to a_{j_1} \cdots a_{j_f}$') is a rule, then in the next step a token is deposited to each of nodes $a_{j_1}, \ldots, a_{j_f}$ while a token is removed from a_i. All the applicable movements of tokens are carried out in parallel. Since the outgoing edges of each node are uniquely labelled, a loop can never become token-free once it contains a token. Clearly, the presence of a single copy of token in node a_{i_0} is sufficient to 'activate' the loop (mentioned in statement (1) of the lemma), making the CS nonterminating.

For (3), deciding nontermination is tantamount to checking whether in G_{P,c_0}, a reachable loop from some node a_{i_0} exists, which is clearly doable in polynomial time. This completes the proof. □

Deterministic CSs also have the following *monotonic property* regarding nonterminating computations.

Lemma 2.7. *Given a deterministic CS P, if P does not halt from configuration c, then P does not halt from any configuration c' such that $\#_{c'} \geqslant \#_c$.*

Proof. The proof is similar to the proof of Claim 2 in Theorem 2.1. □

Hence, we have the following result.

Theorem 2.8. *For a deterministic CS P and a fixed string w, the set $L = \{o_1^{n_1} \cdots o_k^{n_k} \mid P$ halts on $w o_1^{n_1} \cdots o_k^{n_k}\}$ is effective semilinear. In fact, L is either empty, or of the form $o_1^{n_1} \cdots o_k^{n_k}$, where $n_i = *$ or $0, 1 \leqslant i \leqslant k$.*

Proof. Let $\alpha, \beta \in \{o_1, \ldots, o_k\}^*$. It follows from Lemma 2.6 that P halts on $w\alpha$ and $w\beta$ iff P halts on $w\alpha\beta$. As a consequence, if P does not halt on w, then $L = \emptyset$; otherwise, $L = \{o_i \mid P \text{ halts on } wo_i\}^*$, which is clearly effectively constructible. The result is proven. □

We immediately have the following, which strengthens Theorem 2.4.

Corollary 2.9. *Deterministic CSs are not universal.*

In contrast, it is known that *nondeterministic* 1-membrane CSs are universal [5] (see also [8]) even operating under the 3-*Max-Parallel* mode. The universality result holds for either purely CSs with three catalysts, or CSs with two catalysts. In fact, to simulate a Turing machine M the 1-membrane CS needs no more than k noncatalysts for some *fixed* k, independent of M, as [8] shows.

It is also interesting to compare and contrast the model of deterministic CSs with that of the so-called deterministic CPS with only *one* membrane (called the skin membrane) [15]. Consider the following example.

Example 2.10. For deterministic CPS with only *one* membrane, the rules are of the form:
(1) $a \to a_x$,
(2) $ab \to a_x b_y$,
(3) $ab \to a_x b_y c_{\text{come}}$,
where a, b, c are objects, x, y (which indicate the directions of movements of a and b) can only be *here* (i.e., the object remains in the membrane) or *out* (i.e., the object is expelled into the environment). The third rule brings in an object c from the environment into the skin membrane. It is known that deterministic 1-membrane CPS are universal [15]. Note also that the universality result holds even when a bound (more precisely, 3) is imposed on the size of maximally parallel rules applicable at each step.

In spite of the similarity between rules of type (2) (also (3)) in CPS and rules of type $Ca \to Cw$ in CS, deterministic CS are less powerful than deterministic CPS, as our preceding discussion reveals. The disparity between CS and CPS rules lies in the fact that for an applicable rule $Ca \to Cw$ in CS, the catalytic symbol C is always present throughout the computation, whereas for a rule of type either $ab \to a_x b_y$ or $ab \to a_x b_y c_{\text{come}}$ in CPS, symbols a and b (with one seemingly playing the role of a catalytic symbol, to a certain degree) come in and out of the skin membrane. To get a better idea for the difference between deterministic CS and CPS, suppose we have the following rules in a CPS:

$$\begin{aligned} &ab \to a_{\text{here}} b_{\text{out}} d_{\text{come}};\\ &yd \to y_{\text{out}} d_{\text{out}};\\ &dx \to d_{\text{out}} x_{\text{out}};\\ &\vdots \end{aligned}$$

Now consider the following computation, where a, b, d, x, y are symbols, and w_1, w_2, w_3, w_4 are strings of symbols:
(1) $abw \to adw_1 \to adw_2 \to \cdots \to aydw_3 \to aw_4 \to \cdots \to abxw$.

From $abxw$, the computation continues in the following way:
(2) $abxw \to adxw_1 \to aw_2 \to \cdots$.

In the second step of (2), dx will be thrown away; but in (1) d remains until later on when y is brought in. Hence, the monotonic property stated in Claim 2 in the proof of Theorem 2.1 no longer holds, as $\#_{aw_2}$ in (2) $\not\geq \#_{adw_2}$ in (1) at the end of the second step of the computation, although (2) starts with $abxw$ which is greater than abw (the starting configuration of (1)). This is because the x (playing the role of a catalyst symbol, in some sense) that is going to couple with d is brought in later in the computation in (1). However, if x were a catalyst symbol in a CS, then x would have been in existence throughout the computation. Hence, the sequence of rules applied in (1) cannot be repeated in spite of having an extra symbol x in the end.

Consider the following extension of CS:

- *Multimembrane* CSs, where each rule in a membrane looks like: $Ca \to Cv$ or $a \to v$, where the symbols in v have designated target membranes specifying where they are to be moved. The catalyst C remains in the membrane containing the rule. In this case w represents the configurations $w_1, \ldots, w_m$ in the m membranes.

It turns out that our results obtained thus far can be extended to multimembrane CSs.

Theorem 2.11. *Deterministic multimembrane CSs are not universal.*

Theorem 2.1 and Corollary 2.2 (characterizing semilinear reachability sets) also hold for deterministic purely multimembrane CSs (i.e., without $a \to v$ type of rules). Using a reduction by encoding regions in objects, the proofs of the above are similar to that for the 1-membrane case.

3. Prioritized deterministic CSs

Now let us look at CSs which allow rules to be prioritized according to the following two types of priority relations. Let R be the set of rules of a CS. For a priority relation ρ over R, we write $\rho(r_1) < \rho(r_2)$ (or simply $r_1 < r_2$, if ρ is understood) to denote that $(r_1, r_2) \in \rho$, meaning that r_2 takes precedence over r_1. ρ is said to be of

- *Type A*: If ρ is irreflexive, asymmetric, transitive, and the complement of the symmetric closure of ρ, i.e., $\bar{\rho} = \{(r, r') \mid \neg((r, r') \in \rho) \wedge \neg((r', r) \in \rho)\}$, is an equivalence relation. Clearly, $\bar{\rho}$ induces equivalence classes $\Omega_1, \Omega_2, \ldots, \Omega_k$, for some k, such that $\forall 1 \leqslant i < j \leqslant k$, $\forall r \in \Omega_i, r' \in \Omega_j$, $\rho(r) < \rho(r')$. The subscript i of Ω_i can be thought of as the priority level of rules in Ω_i. For $r \in \Omega_i$, we also write $\bar{\rho}(r) = i$. (The interested reader is referred to [2] for an example of applying this notion of a priority relation to reasoning about concurrent systems.)
- *Type B*: If ρ is an irreflexive, asymmetric, and transitive relation. That is, ρ is a *strict partial order*.

Example 3.1. Consider a strict partial order ρ over $R = \{r_1, \ldots, r_6\}$: $(r_5 > r_3 > r_2 > r_1)$; $(r_6 > r_4 > r_2 > r_1)$; $(r_5 > r_4)$; and $(r_6 > r_3)$. Then $\bar{\rho} = \{(r_5, r_6), (r_6, r_5), (r_3, r_4), (r_4, r_3)\} \cup \{(r_i, r_i) \mid 1 \leqslant i \leqslant 6\}$, which is an equivalence relation. Hence, ρ is of type A. Furthermore, $\bar{\rho}$ partitions R into the following equivalence classes $\Omega_1 = \{r_1\}$, $\Omega_2 = \{r_2\}$, $\Omega_3 = \{r_3, r_4\}$, $\Omega_4 = \{r_5, r_6\}$, such that for $1 \leqslant i < j \leqslant 4$, rules in Ω_i have a lower priority than those in Ω_j.

3.1. Systems under type A priority relation

Let P be a deterministic CS, c and c' be two configurations, and H be a multiset of rules. With respect to a priority relation ρ of type A (with $\bar{\rho}$ inducing equivalence classes $\Omega_1, \Omega_2, \ldots, \Omega_k$):

(1) (*Static priority*): c' is said to follow c through the application of H under the *statically prioritized mode*, written as $c \xrightarrow{H}_t c'$, if H is the maximal multiset satisfying the following:
   (i) $\forall\, r_i, r_j \in H$, $\bar{\rho}(r_i) = \bar{\rho}(r_j)$ (i.e., r_i and r_j are in the same Ω_l, for some l),
   (ii) $\neg\, \exists\, r, r \notin H$, r is applicable in c and $\bar{\rho}(r) > \bar{\rho}(r')$ for some rule $r' \in H$.
   In words, H is the maximal multiset of rules such that if a rule in a higher priority group is applicable, then no rules from a lower priority group can be used.

We first show the following result which characterizes the computations of nonhalting CSs.

Lemma 3.2. *Given a deterministic purely CS P operating under the statically prioritized mode, and an initial configuration c_0, P does not halt iff there exist c and c' with $\#_{c'} \geqslant \#_c$ such that $c_0 \xrightarrow{*}_t c \xrightarrow{*}_t c'$.*

Proof. Let ρ be the underlying priority relation of type A. The *only-if* part follows from Dickson's lemma. It suffices to show the *if* part.

Consider a computation $c_0 \xrightarrow{*}_t c \xrightarrow{H_1}_t c_1 \xrightarrow{H_2}_t c_2 \cdots \xrightarrow{H_k}_t c'$ with $\#_{c'} \geqslant \#_c$. Suppose H_1' is the multiset applied at c'. Let $\bar{\rho}(H_1')$ and $\bar{\rho}(H_1)$ denote the priority levels of rules in H_1' and H_1, respectively. (Note that all the rules in H_1' (as well as H_1) have the same priority level.) Clearly $\bar{\rho}(H_1) \leqslant \bar{\rho}(H_1')$. Consider the following two cases:

(1) $\bar{\rho}(H_1) = \bar{\rho}(H_1')$: In this case, we have $c' \xrightarrow{H_1'}_t c_1'$, for some c_1', with $H_1 \subseteq H_1'$ and $\#_{c_1'} \geqslant \#_{c_1}$ (from Claim 2 in the proof of Theorem 2.1).
(2) $\bar{\rho}(H_1) < \bar{\rho}(H_1')$: Let $c' = cw$, for some string w of noncatalysts. It is fairly easy to observe that if $Ca \to Cv$ is a rule in H_1', then a cannot be in c; otherwise, $Ca \to Cv$ were applicable in c, violating the assumption that $\bar{\rho}(H_1) < \bar{\rho}(H_1')$. Hence, the presence of rule $Ca \to Cv$ in H_1' implies a in w, suggesting that the resulting configuration is of the form cw_1, for some string w_1 of noncatalysts (i.e., $c' \xrightarrow{H_1'}_t cw_1$). There are two cases regarding

the computation emanating from cw_1:

(a) $c' \xrightarrow{H_1'}_t cw_1 \xrightarrow{G_2}_t cw_2 \cdots \xrightarrow{G_i}_t cw_i \xrightarrow{G_{i+1}}_t \cdots$, and $\forall i \geqslant 2$, $\bar{\rho}(G_i) > \bar{\rho}(H_1)$. That is, only multisets of priority higher than $\bar{\rho}(H_1)$ are applied along the computation from cw_1. Then P is nonterminating. The theorem is proven.

(b) $c' \xrightarrow{H_1'}_t cw_1 \xrightarrow{G_2}_t cw_2 \cdots \xrightarrow{G_m}_t cw_m \xrightarrow{H_1''}_t c_1''$, for some m such that $\forall 2 \leqslant i \leqslant m$, $\bar{\rho}(G_i) > \bar{\rho}(H_1)$ and $\bar{\rho}(H_1'') = \bar{\rho}(H_1)$. That is, eventually a multiset of priority equal to $\bar{\rho}(H_1)$ becomes applicable along the computation. Again based on Claim 2 in the proof of Theorem 2.1, we have $H_1 \subseteq H_1''$ and $\#_{c_1''} \geqslant \#_{c_1}$.

In view of the above, either P is nonhalting (in the case of 2(a)), or $c_0 \xrightarrow{*}_t c \xrightarrow{H_1}_t c_1 \xrightarrow{H_2}_t c_2 \cdots \xrightarrow{H_k}_t c' \xrightarrow{*}_t \bar{c}_1$ and $\#_{\bar{c}_1} \geqslant \#_{c_1}$ (in either Case 1 or Case 2(b)). By inductively applying the above argument to the segments in which $H_2 \cdots H_k$ are applied, we immediately have $c' \xrightarrow{*}_s c''$, for some c'' such that $\#_{c''} \geqslant \#_{c'}$. Repeatedly applying the above reasoning to $c' \xrightarrow{*}_t c''$ and onwards, P is nonterminating. □

At this point, we do not know whether the reachability set of a deterministic statically prioritized purely CS is semilinear or not. Lemma 3.2, in conjunction with Dickson's lemma, is sufficient to yield the decidability of the halting problem for such prioritized CSs. Hence, we have the following, whose proof parallels that of Theorem 2.4.

Theorem 3.3. *Deterministic purely CSs under the statically prioritized mode are not universal.*

We now consider the full class of deterministic statically prioritized CSs with both catalytic and noncatalytic rules. It turns out that the graph-theoretic approach employed in Section 2.2 remains valid for this new class of CSs.

Let ρ be the underlying priority relation of type A. Given a deterministic statically prioritized CS P and an initial configuration c_0, we construct a directed labelled graph $G^t_{P,c_0} = (V, E)$, where V is the set of noncatalytic symbols, and $E = \{(a_i, a_j)_r \mid$ there exists a rule r of the form $C_t a_i \to C_t w$ or $a_i \to w$, such that a_j is in w, and no rule r' of equal or higher priority level (i.e., $\bar{\rho}(r') \geqslant \bar{\rho}(r)$) is in conflict with $r\}$. It is important to explain why E constructed above does not leave out edges corresponding to applicable rules. Suppose r is a rule in conflict with another rule of equal priority level in ρ. P being deterministic prohibits r from being enabled; hence, r can be dropped without affecting the computation of P. Similarly, if r is in conflict with a rule r' of higher priority level, then r can never be applied since r and r' become enabled simultaneously, and only the one of the higher priority level prevails. Again, r plays no role in P's computation in this case. It is therefore clear that like the execution graph in the unprioritized case, G^t_{P,c_0} also enjoys the property that for each node in V the outgoing edges of the node are uniquely labelled. What makes this property critical is that if a noncatalyst a_i is in the current configuration of P, the only way to prevent the unique rule associated with a_i (in the execution graph) from being applied indefinitely is for P to apply rules of higher priority level forever, implying P to be nonhalting. Therefore, it becomes fairly easy to see that P is nonhalting iff G^t_{P,c_0} has a reachable loop from some node whose corresponding symbol appears in c_0. In view of this key observation, Lemmas 2.6, 2.7, and Theorem 2.8 also hold for deterministic statically prioritized CSs. Hence, we have:

Theorem 3.4. *For a deterministic statically prioritized CS P and a fixed string w, the set $L = \{o_1^{n_1} \cdots o_k^{n_k} \mid P$ halts on $wo_1^{n_1} \cdots o_k^{n_k}\}$ is effective semilinear. In fact, L is either empty, or of the form $o_1^{n_1} \cdots o_k^{n_k}$, where $n_i = *$ or 0, $1 \leqslant i \leqslant k$.*

3.2. Systems under type B priority relation

Again, let P be a deterministic CS, c and c' be two configurations, and H be a multiset of rules. With respect to a priority relation ρ of type B, the following two notions of priority are considered.

(1) (*Strong priority*): c' is said to follow c through the application of H under the *strongly prioritized mode*, written as $c \xrightarrow{H}_s c'$, if H is the maximal multiset satisfying the following:
   (a) $\forall r_1 \in H, \neg\exists r_2 \notin H$ such that $\rho(r_1) < \rho(r_2)$ and $(H - \{r_1\}) \cup \{r_2\}$ is still applicable,
   (b) $\forall r_1, r_2 \in H, \neg(\rho(r_1) < \rho(r_2))$.
   In words, if a rule with higher priority is used, then no rule of a lower priority can be used, even if the two rules do not compete for objects. Note that this priority notion coincides with the one used in [11].

(2) (*Weak priority*): c' is said to follow c through the application of H under the *weakly prioritized mode*, written as $c \overset{H}{\rightarrow}_w c'$, if H is the maximal multiset satisfying the following:

$\forall r_1 \in H, \neg\exists r_2 \notin H$ such that $\rho(r_1) < \rho(r_2)$ and $(H - \{r_1\}) \cup \{r_2\}$ is still applicable.

In words, none of the rules in H can be replaced by a higher priority one.

Note that P is deterministic if at any time the applied multiset is always unique. We use the following simple example to illustrate the difference between the above two notions of priority.

Example 3.5. Consider a deterministic CS P with the following rules:

$$\text{Rule } r_1 : Cb_1 \to Cb_2; \quad \text{Rule } r_2 : Ca_1 \to Ca_2; \quad \text{Rule } r_3 : Da_1 \to Da_3;$$

$$\text{Rule } r_4 : Ec_1 \to Ec_2; \quad \text{Rule } r_5 : Fc_1 \to Fc_3; \quad \text{Rule } r_6 : Gd_1 \to Dd_2.$$

(*Priority relation*): $\rho(r_1) > \rho(r_2) > \rho(r_3)$; $\rho(r_4) > \rho(r_5)$.

Then

$$CDEFGa_1b_1c_1d_1 \overset{\{r_1,r_4,r_6\}}{\rightarrow}_s CDEFGa_1b_2c_2d_2$$

$$CDEFGa_1b_1c_1d_1 \overset{\{r_1,r_3,r_4,r_6\}}{\rightarrow}_w CDEFGa_3b_2c_2d_2.$$

Note that under the weak priority semantics, the application of r_1 makes r_3 applicable, since r_2 (competing for the catalyst C with r_1) is 'disabled' by r_1. Under the strong priority semantics, however, the application of r_1 disables r_3 (since $\rho(r_3) < \rho(r_1)$) even though these two rules do not compete for objects.

In contrast to Theorem 2.4 (also Theorem 2.8) that deterministic unprioritized CSs are not universal, allowing strongly or weakly prioritized rules boosts the computational power as the following result shows.

Theorem 3.6. *Deterministic purely CSs under the weakly prioritized* (*or strongly prioritized*) *mode are universal.*

Proof. The proof involves the construction of a purely CS that simulates a given deterministic k-counter machine which starts with one counter having value n and the other counters empty. We only consider the case $k = 2$, the generalization for any k being straightforward.

Let M be a deterministic two-counter machine. Each of M's transitions is of one of the following forms:

- (*Increment*) s: $c++$, `goto` s' (on state s, increment counter c by one and move to state s').
- (*Test-for-zero/decrement*) s: If $c = 0$, `goto` s_1 else $c--$, `goto` s_2 (on state s, if counter c is zero, go to state s_1; otherwise, decrement counter c by one and move to state s_2.

We show how to construct a deterministic purely CS P under either the strongly or the weakly prioritized mode such that starting with one counter empty and the other counter having value n, M halts iff P halts on the initial configuration $w(o_1)^n$, where w is a string of catalytic and noncatalytic symbols not including the symbol o_1. Let the two counters of M be c_1 and c_2.

At any point in time, the configuration of P is of the form $C_1C_2D_1D_2s(o_1)^{n_1}(o_2)^{n_2}t$, where C_1, D_1 (resp., C_2, D_2) are catalysts associated with the simulation of M's transitions operating on counter c_1 (resp., c_2), s represents the current state of M, n_1 and n_2 keep track of the values of counters c_1 and c_2, respectively, and t is a noncatalyst whose purpose will be explained later.

We are now in a position to see how the two types of M's transitions are simulated. Without loss of generality, we assume the operations to be simulated operate on counter c_1; the cases on counter c_2 are similar. Let $C_1C_2D_1D_2s(o_1)^{n_1}(o_2)^{n_2}t_2$ be the current configuration.

- Transition s: c_1++, `goto` s' (assuming that from s', the next transition operates on counter c_1).

  P utilizes the following rules:

  $$r_1 : C_1s \to C_1q_so_1; \quad r_2 : C_1q_s \to C_1q'_s; \quad r_3 : C_1q'_s \to C_1s't_2;$$
  $$h'_1 : C_2t_2 \to C_2t'_2; \quad h'_2 : C_2t'_2 \to C_2t''_2; \quad h'_3 : C_2t''_2 \to C_2.$$

  Note that symmetrically we also have rules $h_1 : C_1t_1 \to C_1t'_1$; $h_2 : C_1t'_1 \to C_1t''_1$; $h_3 : C_1t''_1 \to C_1$. If the next transition to be executed on state s' operates on counter c_2, then rule r_3 becomes $C_1q'_s \to C_1s't_1$.

Using the above rules, incrementing counter c_1 is simulated through the following sequence:

$$\begin{aligned} & C_1C_2D_1D_2s(o_1)^{n_1}(o_2)^{n_2}t_2 \\ \overset{\{r_1,h_1'\}}{\rightarrow}_w\ & C_1C_2D_1D_2q_s(o_1)^{n_1+1}(o_2)^{n_2}t_2' \\ \overset{\{r_2,h_2'\}}{\rightarrow}_w\ & C_1C_2D_1D_2q_s'(o_1)^{n_1+1}(o_2)^{n_2}t_2'', \\ \overset{\{r_3,h_3'\}}{\rightarrow}_w\ & C_1C_2D_1D_2s'(o_1)^{n_1+1}(o_2)^{n_2}t_2. \end{aligned}$$

It will be seen later that the length of the above sequence (i.e., three steps) is exactly the same as that of simulating a test-for-zero/decrement.

- Transition s: If $c_1 = 0$, goto s_1 else $c_1 - -$, goto s_2.

P has the following rules, in addition to the h_1', h_2', and h_3' defined above. Assume that from s_1 and s_2, M's transitions operate on counter c_1; the other cases are similar.

$$f_1 : C_1s \to C_1q_{s_2}b; \quad f_2 : C_1o_1 \to C_1; \quad f_3 : C_1q_{s_2} \to C_1q_{s_1}; \quad f_4 : D_1q_{s_1} \to D_1s_1\,t_2;$$

$$f_5 : D_1q_{s_2} \to D_1s_2\,t_2; \quad f_6 : D_1b \to D_1d; \quad f_7 : C_1d \to C_1.$$

The priority relation has

$$f_1 > f_2 > f_3; \quad \{h_1, h_2, h_3\} > f_2; \quad f_7 > f_2 > f_3; \quad f_6 > \{f_4, f_5\}; \quad \{r_1, r_2, r_3\} > f_2.$$

Care has to be taken regarding f_2, which decrements counter c_1. The $\{r_1, r_2, r_3\} > f_2$ is to prevent f_2 from being falsely applied when simulating an 'increment'. Note that f_3 and f_5 are conflicting rules. The simulation involves the following sequence:

$$\begin{aligned} C_1C_2D_1D_2s(o_1)^{n_1}(o_2)^{n_2}t_2 \overset{\{f_1,h_1'\}}{\rightarrow}_w\ & C_1C_2D_1D_2q_{s_2}b(o_1)^{n_1}(o_2)^{n_2}t_2', \\ \overset{\{f_2,f_6,h_2'\}}{\rightarrow}_w\ & C_1C_2D_1D_2q_{s_2}d(o_1)^{n_1-1}(o_2)^{n_2}t_2'', \\ \overset{\{f_5,f_7,h_3'\}}{\rightarrow}_w\ & C_1C_2D_1D_2s_2(o_1)^{n_1-1}(o_2)^{n_2}t_2 \quad \text{provided } n_1 > 0, \\ & \qquad \text{or} \\ \overset{\{f_3,f_6,h_2'\}}{\rightarrow}_w\ & C_1C_2D_1D_2q_{s_1}d(o_2)^{n_2}t_2'', \\ \overset{\{f_4,f_7,h_3'\}}{\rightarrow}_w\ & C_1C_2D_1D_2s_1(o_2)^{n_2}t_2 \qquad \text{provided } n_1 = 0. \end{aligned}$$

In the second step of the above sequence, the application of f_6 disables f_5, allowing f_3 to be applied if counter c_1 is zero. In the third step, the use of f_7 disables both f_2 and f_3, while allowing either f_4 or f_5 to be applied.

Now we explain the role played by noncatalyst t_2 and rules h_1', h_2', and h_3'. Like f_2, there is a rule $C_2o_2 \to C_2$ in existence in P for simulating the operation of decrementing counter c_2. To prevent such a rule from being falsely applied in the course of simulating a transition operating on counter c_1 (such as the two cases detailed above), rules h_1', h_2', and h_3', which take precedence over $C_2o_2 \to C_2$, are designed to prevent $C_2o_2 \to C_2$ from being applied. Likewise, there is a noncatalyst t_1, along with rules h_1, h_2, h_3, associated with the simulation of a transition operating on counter c_2.

Clearly, M halts iff P terminates. It is also obvious that P is deterministic. It is easy to observe that the above argument also works for deterministic purely CSs under the strongly prioritized mode. □

A careful examination of the proof of Theorem 3.6 reveals that the constructed relation ρ is not of type A, assuming the computation involved in the simulation to be under the static priority semantics. If otherwise, f_2 (or f_3) and h_2', which are applied simultaneously, must be in the same equivalence class, so are f_2' (or f_3') and h_2, where f_2' and f_3' are the analogies of f_2 and f_3, respectively, used in simulating an operation on counter $_2$. But in our priority relation ρ, $f_3 < f_2 < \{h_1, h_2, h_3\}$ and $f_3' < f_2' < \{h_1', h_2', h_3'\}$, we immediately have a contradiction since $\bar{\rho}(f_2) = \bar{\rho}(h_2')$ (or $\bar{\rho}(f_3) = \bar{\rho}(h_2')$), $\bar{\rho}(f_2') = \bar{\rho}(h_2)$ (or $\bar{\rho}(f_3') = \bar{\rho}(h_2)$), $\bar{\rho}(h_2) > \bar{\rho}(f_2)$, and $\bar{\rho}(h_2') > \bar{\rho}(f_2')$.

Finally, we use the following example to explain in an intuitive fashion why strongly and weakly prioritized deterministic CSs are more powerful than statically prioritized ones.

Example 3.7. Consider the following CS P:

$$\text{Rule } r_1 : Cb_1 \to Cb_2; \quad \text{Rule } r_2 : Ca_1 \to Ca_2; \quad \text{Rule } r_3 : Da_1 \to Da_3; \quad \text{Rule } r_4 : Ec_1 \to Ec_2;$$

$$\text{Rule } r_5 : Fc_1 \to Fc_3; \quad \text{Rule } r_6 : Ed_1 \to Ed_2; \quad \text{Rule } r_7 : Fe_1 \to Fe_2.$$

(*Priority relation*): $\rho(r_1) > \rho(r_2) > \rho(r_3)$; $\rho(r_6) > \rho(r_4)$; $\rho(r_7) > \rho(r_5)$.

First consider r_1, r_2, and r_3 under the weakly prioritized mode. With the presence of a noncatalyst a_1, the applicability of r_2 or r_3 depends on whether r_1 is applied. Hence, neither r_2 nor r_3 can be removed at node a_1 in constructing the execution graph of P. (Note that under the strongly prioritized mode, the application of r_1 blocks rules r_2 and r_3.) For rules r_4–r_6 under the strongly prioritized mode, as long as exactly one of d_1 and e_1 exists in the configuration, either r_4 or r_5 becomes applicable depending on whether r_7 or r_6 is used. Again, both labels r_4 and r_5 have to be kept at node c_1 in the execution graph of P.

Unlike the unique labelling property under the semantics of static priority, the presence of mixed labels in both the strongly and weakly prioritized cases, in some sense, introduces 'nondeterminism' (in an implicit fashion) to the computation of the CS, which seems to be one of the reasons behind which weakly and strongly prioritized CSs are more powerful than statically prioritized ones. Another reason is that under either strong or weak priority, it is possible to apply $\{r_5, r_6\}$ (and $\{r_4, r_7\}$) in parallel. If this is also the case for static priority, however, it becomes impossible to assign priority levels (as in the case of a type A relation) to r_4, r_5, r_6, r_7 while respecting the semantics of static priority. In this regard, both strong and weak priorities allow the priority of a rule to be *dynamic*, whereas in the static priority case, the priority of a rule is *static* (set in the beginning). We feel that such a disparity between dynamic and static priority assignments also plays a key role in the difference in power between the three prioritized systems.

4. Conclusion

It has been an open problem whether deterministic CSs are universal or not. We answered this question negatively in this paper. Our result gives the first example of a variant of P systems for which the nondeterministic version is universal, but the deterministic version is not. For deterministic purely CSs, we were able to establish semilinearity of their reachability sets. Finding a bound for the size of such a semilinear representation remains an interesting open question. We also considered deterministic CSs which allow rules to be prioritized under three notions of priority relations, namely, *static priority*, *strong priority*, and *weak priority*. We obtained contrasting results by showing that deterministic strongly and weakly prioritized CSs are universal, whereas statically prioritized systems only accept semilinear sets.

In view of the somewhat surprising result that deterministic CSs are weaker than their nondeterministic counterparts, it is of importance and interest to investigate in the future the computational powers of deterministic versions of other variants of P systems.

Acknowledgments

The authors thank the anonymous referees for their comments and suggestions, which improved the presentation of this paper.

References

[1] A. Alhazov, On the power of deterministic EC P systems, Second Brainstorming Week on Membrane Computing 2004, Sevilla, Spain, February 2–7, 2004.
[2] F. Bause, On the analysis of Petri nets with static priorities, Acta Inform. 33 (7) (1996) 669–685.
[3] C. Calude, G. Păun, Computing with Cells and Atoms: After Five Years (new text added to Russian edition of the book with the same title first published by Taylor and Francis Publishers, London, 2001), Pushchino Publishing House, submitted for publication.
[4] L. Dickson, Finiteness of the odd perfect and primitive abundant numbers with distinct factors, Amer. J. Math. 35 (4) (1913) 413–422.
[5] R. Freund, L. Kari, M. Oswald, P. Sosik, Computationally universal P systems without priorities: two catalysts are sufficient, Theoret. Comput. Sci. 330 (2) (2005) 251–266.
[6] R. Freund, G. Păun, On deterministic P systems. See P systems web page at: ⟨http://psystems.disco.unimib.it⟩.
[7] S. Ginsburg, The Mathematical Theory of Context-Free Languages, McGraw-Hill, New York, 1966.

[8] O. Ibarra, H. Yen, Z. Dang, The power of maximal parallelism in *P* systems, in: Proc. Eighth Internat. Conf. on Developments in Language Theory (DLT'04), Lecture Notes in Computer Science, Vol. 3340, Springer, Berlin, 2004, pp. 212–224.
[9] A. Păun, G. Păun, The power of communication: *P* systems with symport/antiport, New Generation Comput. 20 (3) (2002) 295–306.
[10] G. Păun, Computing with membranes, Research Report No. 208, Turku University of Computer Science, 1998.
[11] G. Păun, Computing with membranes, J. Comput. System Sci. 61 (1) (2000) 108–143.
[12] G. Păun, Membrane Computing: an Introduction, Springer, Berlin, 2002.
[13] G. Păun, Further twenty six open problems in membrane computing, Written for the Third Brainstorming Week on Membrane Computing, Sevilla, Spain. See *P* systems web page at: ⟨http://psystems.disco.unimib.it⟩, 2005.
[14] G. Păun, G. Rozenberg, A guide to membrane computing, Theoret. Comput. Sci. 287 (1) (2002) 73–100.
[15] P. Sosik, *P* systems versus register machines: two universality proofs, in: Pre-Proc. of Workshop on Membrane Computing (WMC-CdeA2002), Curtea de Argeş, Romania, 2002, pp. 371–382.