The Power of Maximal Parallelism in P Systems ⋆

Oscar H. Ibarra[1⋆⋆], Hsu-Chun Yen[2], and Zhe Dang[3]

[1]Department of Computer Science
University of California
Santa Barbara, CA 93106, USA

[2]Department of Electrical Engineering
National Taiwan University
Taipei, Taiwan 106, R.O.C.

[3]School of Electrical Engineering and Computer Science
Washington State University
Pullman, WA 99164, USA

Abstract. We consider the following definition (different from the standard definition in the literature) of "maximal parallelism" in the application of evolution rules in a P system G: Let $R = \{r_1, ...r_k\}$ be the set of (distinct) rules in the system. G operates in maximal parallel mode if at each step of the computation, a maximal subset of R is applied, and at most one instance of any rule is used at every step (thus at most k rules are applicable at any step). We refer to this system as a maximally parallel system. We look at the computing power of P systems under three semantics of parallelism. For a positive integer $n \leq k$, define:

n-**Max-Parallel:** At each step, nondeterministically select a maximal subset of at most n rules in R to apply (this implies that no larger subset is applicable).

$\leq n$-**Parallel:** At each step, nondeterministically select any subset of at most n rules in R to apply.

n-**Parallel:** At each step, nondeterministically select any subset of exactly n rules in R to apply.

In all three cases, if any rule in the subset selected is not applicable, then the whole subset is not applicable. When $n = 1$, the three semantics reduce to the **Sequential** mode.

We focus on two popular models of P systems: multi-membrane catalytic systems and communicating P systems. We show that for these systems, n-**Max-Parallel** mode is strictly more powerful than any of the following three modes: **Sequential**, $\leq n$-**Parallel**, or n-**Parallel**. For example, it follows from the result in [10] that a maximally parallel communicating P system is universal for $n = 2$. However, under the three limited modes of parallelism, the system is equivalent to a vector addition system, which is known to only define a recursive set. These generalize and refine the results for the case of 1-membrane systems recently reported in [4]. Some of the present results are rather surprising. For example, we show that a **Sequential** 1-membrane communicating P system can only generate a semilinear set, whereas with k membranes, it is equivalent to a vector addition system for any $k \geq 2$ (thus the hierarchy collapses at 2 membranes - a rare collapsing result for nonuniversal P systems). Another unexpected result is the following: the reachability problem for **Sequential** multi-membrane catalytic systems with prioritized rules is NP-complete. This contrasts the known result [9] that a 1-membrane catalytic system with only 3 catalysts and (non-prioritized) catalytic rules operating under 3-**Max-Parallel** mode can simulate any 2-counter machine M. We also give another proof (using vector addition systems) of this later result, but unlike in [9], our catalytic system needs only a *fixed* number of noncatalysts, independent of M.

A simple cooperative system (SCO) is a P system where the only rules allowed are of the form $a \rightarrow v$ or of the form $aa \rightarrow v$, where a is a symbol and v is a (possibly null) string of symbols not containing a. We show that a 9-**Max-Parallel** 1-membrane SCO is universal.

⋆ The research of Oscar H. Ibarra was supported in part by NSF Grants IIS-0101134 and CCR02-08595.

⋆⋆ Corresponding author (ibarra@cs.ucsb.edu).

1 Introduction

There has been a flurry of research activities in the area of membrane computing (a branch of molecular computing) initiated five years ago by Gheorghe Paun [17]. 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 concurrent systems in an efficient way that would allow us to solve currently intractable problems (in much the same way as the promise of quantum and DNA computing) once future bio-technology (or silicon-technology) gives way to a practical bio-realization (or chip-realization).

The Institute for Scientific Information (ISI) has recently selected membrane computing as a fast "Emerging Research Front" in Computer Science (see http://esi-topics.com/erf/october2003.html). 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 Venn diagram or a tree structure 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.unimb/it for a large collection of papers in the area, and in particular the monograph [18].)

As already mentioned above, in the standard semantics of P systems [17–19], each evolution step of a system G is a result of applying all the rules in G 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 multi-set of rules, which are chosen nondeterministically. For example, a catalytic rule $Ca \rightarrow Cv$ in membrane q is applicable if there is a catalyst C and an object (symbol) a in the preceding configuration in membrane q. 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.

In this paper, we study a different definition of maximal parallelism. Let G be a P system and $R = \{r_1, ..., r_k\}$ be the set of (distinct) rules in all the membranes. (Note that r_i uniquely specifies the membrane the rule belongs to.) We say that G operates in maximal parallel mode if at each step of the computation, a maximal subset of R is applied, and at most one instance of any rule is used at every step (thus at most k rules are applicable at any step). For example, if r_i is a catalytic rule $Ca \rightarrow Cv$ in membrane q and the current configuration has two C's and three a's in membrane q, then only one a can evolve into v. Of course, if there is another rule r_j, $Ca \rightarrow Cv'$, in membrane q, then the other a also evolves into v'. Throughout the paper, we will use this definition of maximal parallelism. Here, we look at the computing power of P systems under three semantics of parallelism. For a positive integer $n \leq k$, define:

n-**Max-Parallel:** At each step, nondeterministically select a maximal subset of at most n rules in R to apply (this implies that no larger subset is applicable).
$\leq n$-**Parallel:** At each step, nondeterministically select any subset of at most n rules in R to apply.
n-**Parallel:** At each step, nondeterministically select any subset of exactly n rules in R to apply.

In all three cases, if any rule in the subset selected is not applicable, then the whole subset is not applicable. When $n = 1$, the three semantics reduce to the **Sequential** mode.

In the next four sections, we investigate the computing power of two popular models of P systems with respect to the above semantics of parallelism – the catalytic P systems and the communicating P systems.

We should mention some related work on P systems operating in sequential and limited parallel modes. Sequential variants of P systems have been studied, in a different framework, in [8]. There, generalized P systems (GP-systems) were considered and were shown to be able to simulate graph controlled grammars. Our notion of limited parallelism seems to correspond to "cooperation modes" in cooperating distributed grammar systems, investigated in [3].

Some of the proofs can be found in the Appendix, which may be read by the PC members at their discretion.

2 Multi-Membrane Catalytic Systems

2.1 Maximally Parallel CS

First we recall the definition of a catalytic system (CS). The membranes (regions) are organized in a hierarchical (tree) structure and are labeled 1, 2, .., m for some m, with the outermost membrane (the skin membrane) labeled 1. At the start of the computation, there is a distribution of *catalysts* and *noncatalysts* in the membranes (the distribution represents the initial configuration of the system). Each membrane may contain a finite set of catalytic 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. When this rule is applied, the catalyst remains in the membrane the rule is in, symbol a is deleted from the membrane, and the symbols comprising v (if nonnull) are transported to other membranes in the following manner. Each symbol b in v has a designation or target, i.e., it is written b_x, where x can be $here$, out, or in_j. The designation $here$ means that the object b remains in the membrane containing it (we usually omit this target, when it is understood). The designation out means that the object is transported to the membrane directly enclosing the membrane that contains the object; however, we do not allow any object to be transported out of the skin membrane. The designation in_j means that the object is moved into a membrane, labeled j, that is directly enclosed by the membrane that contains the object.

It is important to note that our definition of catalytic system is different from what is usually called catalytic system in the literature. Here, we do not allow rules without catalysts, i.e., rules of the form $a \rightarrow v$. Thus our systems use only purely catalytic rules.

Suppose that S is a CS with m membranes. Let $\{a_1, ..., a_n\}$ be the set of noncatalyst symbols (objects) that can occur in the configurations of S. Let $w = (w_1, ..., w_m)$ be the initial configuration, where w_i represents the catalysts and noncatlysts in membrane i. (Note that w_i can be null.) Each reachable configuration of S is an nm-tuple $(v_1, ..., v_m)$, where v_i is an n-tuple representing the multiplicities of the symbols $a_1, ..., a_n$ in membrane i. Note that we do not include the catalysts in considering the configuration as they are not changed (i.e., they remain in the membranes containing them, and their numbers remain the same during the computation). Hence the set of all reachable configurations of S, denoted by $R(S)$ is a subset of $\mathbf{N}^{mn}$. The set of all halting reachable configurations is denoted by $R_h(S)$.

2.2 Sequential CS

In a sequential multi-membrane CS, each step of the computation consists of an application of a single nondeterministically chosen rule, i.e., the membrane and rule within the membrane to apply are chosen nondeterministically. We show below that sequential multi-membrane CS's define exactly the semilinear sets.

We need the definition of a vector addition system. 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 $R(G) = \{z \mid$ 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 $W\}$. For convenience, we write

$z \overset{v_1 v_2 \ldots v_j}{\rightarrow} z'$ ($z, z' \in N^n$, $j \geq 0$) if $\forall i, 1 \leq i \leq j, v_i \in W$, $z + v_1 + ... + v_i \geq 0$ and $z' = z + v_1 + ... + v_j$. We also use $z \overset{*}{\rightarrow} z'$ to denote the existence of a $\sigma \in W^*$ such that $z \overset{\sigma}{\rightarrow} z'$.

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.

The following summarizes the known results concerning VAS and VASS [21, 12, 1, 13, 16]:

Theorem 1. *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.

A *communication-free VAS* is a VAS where in every transition, at most one component is negative, and if negative, its value is -1. Communication-free VAS's are equivalent to communication-free Petri nets, which are also equivalent to commutative context-free grammars [5, 14]. It is known that they have effectively computable semilinear reachability sets [5].

Our first result shows that a sequential CS is weaker than a maximally parallel CS.

Theorem 2. *Every sequential multi-membrane CS S can be simulated by a communication-free VAS G, and vice versa.*

Proof. Let S be an m-membrane CS with noncatalysts $a_1, ..., a_n$. Suppose that the start configuration of $w = (w_1, ..., w_m)$ has k catalysts $C_1, ..., C_k$. We may assume, without loss of generality by adding new catalysts and rules if necessary, that each C_i occurs at most once in w_i ($1 \leq i \leq m$). Number all the rules in S by $1, ..., s$. Note that the rule number uniquely determines the membrane where the rule is applicable.

We first transform S to a new system $\mathcal{S}$ by modifying the rules and the initial configuration w. $\mathcal{S}$ will now have catalysts $C_1, ..., C_k, Q_1, ..., Q_s$ and noncatalysts $a_1, ..., a_n, d_1, ..., d_s$. The component w_q of the initial configuration in membrane q will now be w_q plus each Q_h for which rule number h is in membrane q. The rules of S' are defined as follows:

Case 1: Suppose that $C_j a_i \rightarrow C_j v$ is a rule in membrane q of S, and a_i does not appear in v with designation (target) $here$. Then this rule is in membrane q of S'.

Case 2: Suppose that $C_j a_i \rightarrow C_j a_i^t v$ is rule number r and $t \geq 1$. Suppose that this rule is in membrane q, with the target of each a_i in a_i^t being $here$, and v does not contain any a_i with target $here$. Then the following rules are in membrane q of S':

$C_j a_i \rightarrow C_j d_r^t v$ and $Q_r d_r \rightarrow Q_r a_i$.

In the above rules, the target for a_i and each d_r in the right-hand side of the rules is $here$.

Clearly, S' simulates S, and S' has the property that in each rule $Xb \rightarrow Xv$ (where X is a catalyst, b is a noncatalyst, and v a string of noncatalysts), v does not contain a b with target $here$. It is now obvious that each

rule $Xb \to Xv$ in S' can be transformed to a VAS transition rule of $mn + s$ components, where the component of the transition corresponding to noncatalyst b is -1, and the other components (corresponding to the target designations in v) are nonnegative. Thus, the VAS is communication free.

Conversely, let G be a communication-free VAS. We construct a sequential 1-membrane CS S which has one catalyst C, noncatalysts $\#, a_1, ..., a_k$, and starting configuration $C\#w$, where w corresponds to the starting vector of G. Suppose that $(j_1, ..., j_{m-1}, j_m, j_{m+1}, ..., j_k)$ is a transition in G.

Case 1: $j_m = -1$ and all other j_i's are nonnegative. Then the following rule is in S:
$Ca_m \to Ca_1^{j_1} ... a_{m-1}^{j_{m-1}} a_{m+1}^{j_{m+1}} ... a_k^{j_k}$.

Case 2: All the j_i's are nonnegative. Then the following rule is in S:
$C\# \to C\#a_1^{j_1} ... a_k^{j_k}$.

Clearly, S simulates G. In fact, $R(G) = R(S) \times \{1\}$. ∎

For the proof of the next result, see the Appendix.

Corollary 1. *1. If S is a sequential multi-membrane CS, then $R(S)$ and $R_h(S)$ are effectively computable semilinear sets.*
2. The reachability problem (whether a given configuration is reachable) for sequential multi-membrane CS's is NP-complete.

Since a communication-free VAS can be simulated by a sequential 1-membrane CS (from part 2 of the proof of Theorem 2), we have:

Corollary 2. *The following are equivalent: communication-free VAS, sequential multi-membrane CS, sequential 1-membrane CS.*

2.3 CS Under Limited Parallelism

Here we look at the computing power of the multi-membrane CS under three semantics of parallelism. For a positive integer n, define:

1. n-**Max-Parallel:** At each step, nondeterministically select a maximal set of at most n rules to apply.

2. $\leq n$-**Parallel:** At each step, nondeterministically select any set of at most n rules to apply.

3. n-**Parallel:** At each step, nondeterministically select a set of exactly n rules to apply.

In all three cases above, if any rule in the set selected is not applicable, then the whole set is not applicable. Note that when $n = 1$, the three semantics reduce to the **Sequential** mode.

Theorem 3. *For $n = 3$, a 1-membrane CS operating under the n-**Max-Parallel** mode can define any recursively enumerable set. For any n, a multi-membrane CS operating under $\leq n$-**Parallel** mode or n-**Parallel** mode can be simulated by a VASS.*

Proof. The first part follows from a recent result in [9], where it was shown that a 1-membrane CS with three catalysts (even when each catalyst appears exactly once in the initial configuration) are already sufficient for universality. It remains an interesting open question as to whether the three catalysts can be reduced to two. One catalyst is not enough as was shown in [15].

For the case of $\leq n$-**Parallel** mode, given a multi-membrane CS S, we construct a VASS G to simulate S as follows. G has a unique state p such that G is always in this state at the beginning of every simulation of a parallel step of S. Using several new states, G nondeterministically picks $k \leq n$ rules of the form $Ca_i \to Cv_i$ (k itself is also nondeterministically chosen) and apply each rule sequentially by subtracting 1 from the coordinate corresponding to a_i in an n-tuple transition rule. G then sequentially updates the coordinates as given in the right-hand sides of the rule $Ca_i \to Cv_i$ (G needs additional new states to do this) and enters state p. Clearly, a tuple is reachable in S if and only if it is reachable in G in state p. We omit the details. The case of n-**Parallel** mode follows directly in that G chooses k to be exactly n. ∎

2.4 3-Max-Parallel 1-Membrane CS

As noted above, it is known that a 3-**Max-Parallel** 1-membrane CS is universal [9] in that it can simulate any 2-counter machine M. Here we provide another proof of this result in terms of communication-free VAS. Later we improve this result by showing that, in fact, the 1-membrane CS need no more than k noncatalysts for some *fixed* k, independent of M.

Consider an n-dimensional communication-free VAS $G = \langle x, W \rangle$ with its set of addition vectors W partitioned into three disjoint groups W_1, W_2 and W_3, i.e., $W = W_1 \cup W_2 \cup W_3$ and $W_i \cap W_j = \emptyset$ for all $1 \leq i, j \leq 3, i \neq j$. (For convenience, we write $G = \langle x, (W_1, W_2, W_3) \rangle$ to denote such a VAS.) For $w \in W$, let w^- be a vector such that $w^-(i) = w(i)$ if $w(i) \leq 0$; otherwise $w^-(i) = 0$. A vector $z \in \mathbf{N}^n$ is said to follow $z' \in \mathbf{N}^n$ under the 3-**Max-Parallel** mode if there exist $w_i \in (W_i \cup \{0\}), 1 \leq i \leq 3$, such that

(1) $z + (w_1)^- + (w_2)^- + (w_3)^- \geq 0$,
(2) if $w_i = 0$, then there is no other $w_i' \in W_i$ such that (1) holds if w_i is replaced by w_i', and
(3) $z' = z + w_1 + w_2 + w_3$.

In this case, we write $z \overset{(w_1, w_2, w_3)}{\Rightarrow} z'$. Intuitively, the semantics of 3-**Max-Parallelism** requires that at each step, nondeterministically a maximal set of at most 3 addition vectors be applied simultaneously to yield the next vector; however, from each group $W_i, 1 \leq i \leq 3$, at most one addition vector can be chosen.

Acting as either *acceptors* or *generators*, we show communication-free VAS's under the 3-**Max-Parallel** mode to be computationally equivalent to 2-counter machines. First recall that a transition of a 2-counter machine is of one of the following three forms:

- $q_i \overset{C_d := C_d - 1}{\rightarrow} q_j$ (d=1 or 2): in state q_i, decrement counter C_d by one, and then go to state q_j, provided that $C_d > 0$,
- $q_i \overset{C_d := C_d + 1}{\rightarrow} q_j$: in state q_i, increment counter C_d by one, and then go to state q_j,
- $q_i \overset{C_d = 0}{\rightarrow} q_j$: in state q_i, go to state q_j if counter C_d is empty.

The following result shows the equivalence of 2-counter machines and communication-free VAS operating under the 3-**Max-Parallel** mode. See the Appendix for the details of the proof.

Theorem 4. *Let M be a 2-counter machine with two counters C_1 and C_2. There exist an n-dimensional VAS $G = \langle x, (W_1, W_2, W_3) \rangle$ under the* 3-**Max-Parallel** *mode and a designated coordinate l such that M accepts on initial counter values $C_1 = m$ and $C_2 = 0$ iff*

1. *(generator:) $m \in \{v(l) \mid v \in R_h(G)\}$;*
2. *(acceptor:) from start vector x with $x(l) = m$, $R_h(G) \neq \emptyset$, i.e., G has a halting computation.*

Clearly, from the construction in the proof of Theorem 4, we can assign for each $1 \leq i \leq 3$, a catalyst C_i for the set of addition vectors W_i, define a distinct noncatalyst symbol for each position in the addition vector, and convert each vector in W_i to a rule of the form $C_i a \rightarrow C_i v$, where a is a noncatalyst and v is a (possibly null) string of noncatalysts. Hence, the following corollary:

Corollary 3. *Let M be a 2-counter machine with two counters. There exists a 1-membrane* 3-**Max-Parallel** *CS S with catalysts C_1, C_2, C_3 and n noncatalysts with a designated noncatalyst symbol a_l such that M accepts on initial counter values m and 0, respectively, iff*

1. *(generator:) $m \in \{\#_{a_l}(y) \mid y \in R_h(S)\}$;*
2. *(acceptor) if S starts with initial configuration $(a_l)^m y$, for some y not containing a_l, then $R_h(S) \neq \emptyset$.*

We note that in the corollary above, the CS operates in 3-**Max-Parallel** mode. Now the catalyst C_1 (resp. C_2) is needed to make sure that at most one addition vector in W_1 (resp. W_2) is simulated by the CS at each step. However, catalyst C_3 is not really needed in that we can convert each addition vector in W_3 to a rule of the form $a \rightarrow v$, i.e., a *noncooperative rule* (without a catalyst). Thus, the system can be constructed to

have only *two* catalysts with catalytic rules and noncooperative rules. This was also shown in [9]. However, the degree of maximal parallelism in the system is no longer 3 (because now more than one noncooperative rule may be applicable at each step). In fact, a careful examination of the proof of Theorem 4 reveals that at any point, no more than 3 noncooperative rules are applicable. This in turn implies that the degree of maximal parallelism now becomes 5 (two catalysts plus 3 noncooperative rules). Note also that n, the dimension of the communication-free VAS, which translates to the number of noncatalysts for the system, is also a function of the number of states, hence is unbounded.

We can improve the above results. We need the following lemma whose proof is in the Appendix.

Lemma 1. *There exists a 2-counter machine U with counters C_1 and C_2 that is universal in the following sense. When U is given a description of an arbitrary 2-counter machine M as a positive integer in C_2 and an input m in C_1, U accepts iff M with input m on its first counter and 0 on its other counter accepts.*

Let s be the number of states of U in the above lemma. Looking at the construction in the proof of Theorem 4, we see that n, the dimension of the communication-free VAS is bounded by a function of s. Then the following corollary follows (one need only modify the first part of the proof of Theorem 4, since now, we use the universal 2-counter machine U, where initially, $C_1 = m$, and $C_1 =$ positive integer description of the 2-counter machine M).

Corollary 4. *There exists a* fixed positive integer n *such that if $L \subseteq \mathbf{N}$ is* any *recursively enumerable set of nonnegative integers, then:*

1. *L can be generated (accepted) by a 1-membrane* 3-**Max-Parallel** *CS with 3 catalysts and n noncatalysts.*
2. *L can be generated (accepted) by a 1-membrane* 5-**Max-Parallel** *P system with 2 catalysts and n noncatalysts with catalytic and noncooperative rules.*

2.5 9-Max-Parallel 1-Membrane CS with One Catalyst

We now look at a model of a 1-membrane CS with only *one* catalyst C with initial configuration $C^k x$ for some string x of noncatalysts (thus, there are k copies of C). The rules allowed are of the form $Ca \to Cv$ or of the form $Caa \to Cv$, i.e., C catalizes two copies of an object. Clearly the system operates in maximally parallel mode, but uses no more than k rules in any step. We call this system 1GCS. This system is equivalent to a restricted form of cooperative P system [17, 18]. A simple cooperative system (SCO) is a P system where the rules allowed are of the form $a \to v$ or of the form $aa \to v$. Moreover, there is some fixed integer k such that the system operates in maximally parallel mode, but uses no more that k rule instances in any step. We can show the following (see Appendix):

Theorem 5. *1GCS (hence, also SCO) operating under the* 9-**Max-Parallel** *mode is universal.*

2.6 Sequential Multi-Membrane CS with Prioritized Rules

In this section, we briefly consider the model of a sequential multi-membrane CS where the rules are prioritized. Specifically, there is a priority relation on the rules: A catalytic rule R' of lower priority than R cannot be applied if R is applicable. We refer to this system as prioritized CS. We know that the reachability set of a sequential multi-membrane CS is semilinear and, hence, its reachability problem is NP-complete. In [15], the status of the reachability problem for systems with prioritized rules was left open. Here we show that the reachability problem is also NP-complete.

Taking advantage of the equivalence between sequential multi-membrane CS and communication-free VAS, we first show the reachability problem for prioritized VAS (which will be defined in detail below) to be NP-complete, which immediately yields the mentioned complexity result for prioritized sequential multi-membrane CS.

Given a communication-free VAS $G = \langle x, W \rangle$, a *priority relation* ρ over W is an irreflexive, asymmetric, and transitive relation such that v_2 takes precedence over v_1 if $(v_1, v_2) \in \rho$, meaning that v_1 cannot be applied

if v_2 is applicable. Due to the nature of communication-freeness, we further assume ρ to satisfy a property that if v and v' subtract from the same coordinate, neither (v, v') nor (v', v) is in ρ; otherwise, one of the two could never be applied. Let $\bar{\rho}$ denote $\{(v, v') | (v, v') \notin \rho$ and $(v', v) \notin \rho\}$. In this paper, $\bar{\rho}$ is assumed to be an equivalence relation, which partitions W into a number of *equivalence classes* $\Omega_1, ..., \Omega_d$, for some d ($d \leq |W|$). Intuitively, each $\Omega_i, 1 \leq i \leq d$, represents a set of vectors having the same priority. For every $v \in \Omega_i$ and $v' \in \Omega_j$ ($i \neq j$), either $(v, v') \in \rho$ or $(v', v) \in \rho$ (but not both); we write $\Omega_i < \Omega_j$ (resp., $\Omega_j < \Omega_i$) if $(v, v') \in \rho$ (resp., $(v', v) \in \rho$). Without loss of generality, we assume that $\Omega_1, ..., \Omega_d$ be enumerated in increasing priority throughout the rest of this paper. Given a $v \in W$ and a ρ, we let $class(v) = i$ if $v \in \Omega_i$ (i.e., $class(v)$ is the index of the equivalence class containing v).

Given a vector $z \in \mathbf{N}^n$, an addition vector v can be applied at z under priority relation ρ if $z + v \geq 0$ and no other $v' \in W$ such that $z + v' \geq 0$ and $class(v) < class(v')$, i.e., no vector of higher priority is applicable at z. We write $z \xrightarrow{v}_{\rho} z'$, where $z' = z + v$. Let $\xrightarrow{*}_{\rho}$ be the reflexive and transitive closure of $\rightarrow_{\rho}$. The reachability set of G under ρ is $R_{\rho}(G) = \{z \mid x \xrightarrow{*}_{\rho} z\}$. The proof of the following lemma is in the Appendix.

Lemma 2. *Given a communication-free VAS $G = \langle x, W \rangle$ and a priority relation ρ, if $z \xrightarrow{\sigma} z'$ ($\sigma \in W^*$) and for every v applicable at z', $v \in \Omega_1$ (i.e., v is in the lowest priority class induced by $\bar{\rho}$), then $z \xrightarrow{\sigma'}_{\rho} z'$, for some permutation σ' of σ.*

For related results concerning other types of prioritized concurrent models, the reader is referred to [2, 23]. We also need the following known result [11, 22] saying that checking reachability for communication-free VAS can be equated with solving the integer linear programming problem.

Lemma 3. *Given a communication-free VAS $G = \langle x, W \rangle$, there exists a system of linear inequalities $\mathcal{L}(G, x')$ of polynomial size such that $x' \in R(G)$ iff $\mathcal{L}(G, x')$ has an integer solution. Furthermore, $\mathcal{L}(G, x')$ remains linear even if x and x' are replaced by variables.*

Based upon the above lemmas, we have the following result. Again, the proof is in the Appendix.

Theorem 6. *The reachability problem for prioritized communication-free VAS's is NP-complete.*

According to Theorem 2, every sequential multi-membrane CS S can be simulated by a communication-free VAS G, and vice-versa. Consider a priority relation for multi-membrane CS such that the prioritizing is only between rules in the same membrane, and if $Ca \rightarrow v$ and $C'a \rightarrow v'$ are rules in the same membrane, then neither the first rule takes precedence over the second rule nor the second rule takes precedence over the first rule. Examining the proof of Theorem 2 shows that every sequential multi-membrane CS S with prioritized rules can be simulated by a communication-free VAS G with prioritized rules, and vice-versa. Therefore, we immediately have:

Corollary 5. *The reachability problem for sequential multi-membrane CS with prioritized rules is NP-complete.*

3 Sequential 1-Membrane Communicating P Systems

Consider the model of a communicating P system (CPS) with only *one* membrane, called the skin membrane [20]. The rules are of the form:

1. $a \rightarrow a_x$
2. $ab \rightarrow a_x b_y$
3. $ab \rightarrow a_x b_y c_{come}$

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. In the sequel, we omit the designation $here$, so

that objects that remain in the membrane will not have this subscript. There is a fixed finite set of rules in the membrane. At the beginning, there is a fixed configuration of objects in the membrane.

Assume that the computation is *sequential*; i.e., at each step there is only one application of a rule (to one instance). So, e.g., if nondeterministically a rule like $ab \rightarrow a_{here}b_{out}c_{come}$ is chosen, then there must be at least one a and one b in the membrane. After the step, a remains in the membrane, b is thrown out of the membrane, and c comes into the membrane. There may be several a's and b's, but only one application of the rule is applied. Thus, there is *no* parallelism involved. The computation halts when there is no applicable rule. We are interested in the multiplicities of the objects when the system halts.

One can show that a 1-membrane CPS can be simulated by a vector addition system (VAS) (this is a special case of a theorem in the next section). However, the converse is not true – it was shown in [4] that a sequential 1-membrane CPS can only define a semilinear set.

4 Sequential 1-Membrane Extended CPS (ECPS)

We have seen in the previous section that 1-membrane CPS's operating sequentially define only semilinear sets. In contrast, we shall see in the next section that sequential 2-membrane CPS's are equivalent to VASS.

There is an interesting generalization of a 1-membrane CPS, we call extended CPS (or ECPS) – we add a fourth type of rule of the form: $ab \rightarrow a_x b_y c_{come} d_{come}$. That is, two symbols can be imported from the environment. We shall see below that ECPS's are equivalent to VASS's.

Let G be an n-dimensional VASS. Clearly, by adding new states, we may assume that all transitions in G have the form:

$p_i \rightarrow (p_j, +1_h)$,
$p_i \rightarrow (p_j, -1_h)$.

The above is a short-hand notation. The $+1_h$ is addition of 1 to the h-th coordinate, and -1_h is subtraction of 1 from the h-th coordinate. All other coordinates are unchanged. Note at each step, the state uniquely determines whether it is a '+1 transition' or a '-1 transition'.

For constructing the ECPS S equivalent to G, we associate symbol p_i for every state of the VASS, a_h for every coordinate (i.e., position) h in the transition. We also define a new special symbol c. So the ECPS has symbols $p_1, ...p_s$ (s is the number of states), $a_1, ..., a_n$ (n is dimension of the VASS), and c.

Then a transition of the form $p_i \rightarrow (p_j, +1_h)$ in G is simulated by the following rule in S:

$p_i c \rightarrow p_{i(out)} c_{here} p_{j(come)} a_{h(come)}$

A transition of the form $p_i \rightarrow (p_j, -1_h)$ in G is simulated by the following rule in S:

$p_i a_h \rightarrow p_{i(out)} a_{h(out)} p_{j(come)}$

If the VASS G has starting point $\langle p_1, v \rangle$, where $v = (i_1, ..., i_n)$ and p_1 is the start state, then ECPS S starts with the word $p_1 a_1^{i_1} ... a_n^{i_n} c$. Clearly, S simulates G.

Conversely, suppose we are given an ECPS S over symbols $a_1, ..., a_n$ with initial configuration w and rules $R_1, ..., R_k$. The VASS G has states $R_0, R_1, R_1', ...R_k, R_k'$ and starting point $\langle R_0, v_0 \rangle$, where v_0 is the n-dimensional vector in $\mathbf{N}^n$ representing the multiplicities of the symbols in the initial configuration w. The transitions of G are defined as follows:

1. $R_0 \rightarrow (R_i, zero)$ for every $1 \leq i \leq k$ is a transition, where $zero$ represents the zero vector.
2. If R_i is a rule of the form $a_h \rightarrow a_{hx}$, then the following are transitions:
   $R_i \rightarrow (R_i', -1_h)$
   $R_i' \rightarrow (R_j, d_{hx})$ for every $1 \leq j \leq k$, where $d_{hx} = 0_h$ if $x = (out)$ and $d_{hx} = +1_h$ if $x = (here)$.
   (As before, $-1_h, 0, +1_h$ mean subtract 1, add 0, add 1 to h, respectively; all other coordinates are unchanged.)

3. If R_i is a rule of the form $a_h a_r \rightarrow a_{hx} a_{ry}$, then the following are transitions:
   $R_i \rightarrow (R_i', -1_h, -1_r)$
   $R_i' \rightarrow (R_j, d_{hx}, d_{ry})$ for every $1 \leq j \leq k$, where d_{hx} and d_{ry} are as defined above.
   (Note that if $h = r$, then $(R_i', -1_h, -1_r)$ means $(R_i', -2_h)$, i.e., subtract 2 from coordinate h.)
4. If R_i is a rule of the form $a_h a_r \rightarrow a_{hx} a_{ry} a_{s(come)}$, then the following are transitions:
   $R_i \rightarrow (R_i', -1_h, -1_r)$
   $R_i' \rightarrow (R_j, d_{hx}, d_{ry}, +1_s)$ for every $1 \leq j \leq k$, where d_{hx} and d_{ry} are as defined above.
5. If R_i is a rule of the form $a_h a_r \rightarrow a_{hx} a_{ry} a_{s(come)} a_{t(come)}$, then the following are transitions:
   $R_i \rightarrow (R_i', -1_h, -1_r)$
   $R_i' \rightarrow (R_j, d_{hx}, d_{ry}, +1_s, +1_t)$ for every $1 \leq j \leq k$, where d_{hx} and d_{ry} are as defined above.

It follows from the construction above that G simulates S. Thus, we have:

Theorem 7. *Sequential 1-membrane ECPS and VASS are equivalent.*

We can generalize rules of an ECPS further as follows:

1. $a_{i_1} ... a_{i_h} \rightarrow a_{i_1 x_1} ... a_{i_1 x_h}$
2. $a_{i_1} ... a_{i_h} \rightarrow a_{i_1 x_1} ... a_{i_1 x_h} c_{j_{1come}} ... c_{j_{lcome}}$

where $h, l \geq 1$, and $x_m \in \{here, out\}$ for $1 \leq m \leq h$, and the a's and c's are symbols. Call this system ECPS+. Generalizing the constructions in the proof of Theorem 7, we can show ECPS+ is still equivalent to a VASS. Thus, we have:

Corollary 6. *The following systems are equivalent: Sequential 1-membrane ECPS, sequential 1-membrane ECPS+, and VASS.*

Using rules of types of 1 and 2 above, we can define the three versions of parallelism as in Section 2.3, and we can prove the following result. The first part was shown in [10]. The proof of the second part follows the strategy described in the proof of Theorem 3.

Theorem 8. *For $n = 2$, a 1-membrane CPS (and, hence, also 1-membrane ECPS+) operating under the n-***Max-Parallel** *mode can define a recursively enumerable set. For any n, a 1-membrane ECPS+ operating under $\leq n$-***Parallel** *mode or n-***Parallel** *mode is equivalent to a VASS.*

5 Multi-Membrane CPS and ECPS

In this section, we look at CPS and ECPS with multiple membranes. Now the subscripts x, y in the CPS rules $a \rightarrow a_x$, $ab \rightarrow a_x b_y$, $ab \rightarrow a_x b_y c_{come}$ (and $ab \rightarrow a_x b_y c_{come} d_{come}$ in ECPS) can be $here$, out, or in_j. As before, $here$ means that the object remains in the membrane containing it, out means that the object is transported to the membrane directly enclosing the membrane that contains the object (or to the environment if the object is in the skin membrane), and $come$ can only occur within the outermost region (i.e., skin membrane). The designation in_j means that the object is moved into a membrane, labeled j, that is directly enclosed by the membrane that contains the object.

5.1 Sequential 2-Membrane CPS

In Section 3, we saw that a sequential 1-membrane CPS can only define a semilinear set. We now show that if the system has two membranes, it can simulate a vector addition system.

Theorem 9. *A sequential 2-membrane CPS S can simulate a VASS G.*

Proof. Let G be an n-dimensional VAS. Again, we may assume that all transitions in G have the form:

$p_i \to (p_j, +1_h)$,
$p_i \to (p_j, -1_h)$.

Moreover, by adding states, we can tag each state that corresponds to a '+1 transition' as odd or $even$ with the following meaning: a state that is tagged odd (resp. $even$) means that the VASS has so far performed an even (resp. odd) number of '+1 transitions'.

The CPS G has two membranes: membrane 1 (the skin membrane) contains membrane 2. Initially, membrane 2 contains the object (symbol) d. Membrane 1 has initially $p_1 a_1^{i_1} ... a_n^{i_n} c$.

As in the construction in the proof of Theorem 7, a transition of the form $p_i \to (p_j, -1_h)$ in G is simulated by the following rule in membrane 1 of S:

$p_i a_h \to p_{i(out)} a_{h(out)} p_{j(come)}$

A transition of the form $p_i \to (p_j, +1_h)$ in G is simulated by the following rules in S. The rules are partitioned into two cases.

Case: p_i is tagged *odd*.

Rules in membrane 1:

$p_i c \to p_{i(out)} c_{here} p^h_{j(come)}$
$p^h_j c \to p^h_{j(in_2)} c_{(in_2)} a_{h(come)}$
$p^h_j d \to p^h_{j(out)} d_{(here)} p_{j(come)}$

Rule in membrane 2:

$p^h_j d \to p^h_{j(out)} d_{(out)}$

In the rules above, p^h_j is also tagged *odd*.

Case 2: p_i is tagged $even$.

Then we include a similar set of rules for membranes 1 and 2 as above, except that the roles of symbols c and d are switched, i.e., we replace the occurrences of c (resp. d) by d (resp. c) in the rules, and p^h_j is tagged $even$.

It is easily verified that S simulates G. ∎

5.2 Sequential Multi-Membrane ECPS

In Theorem 7, we saw that a sequential 1-membrane ECPS can be simulated by a VASS. The construction can be extended to multi-membrane ECPS. Recall that we now allow rules of the form: $ab \to a_x b_y c_{come} d_{com}$.

Suppose that S has m membranes. Let $\{a_1, ..., a_n\}$ be the set of symbols (objects) that can occur in the configurations of S. Then each reachable configuration of S is an mn-tuple $(v_1, ..., v_m)$, where v_q is an n-tuple representing the multiplicities of the symbols $a_1, ..., a_n$ in membrane q. Then the set of all reachable configurations of S is a subset of $\mathbf{N}^{mn}$. Let $R_1, ..., R_k$ be the rules in S. Note that R_i not only gives the rule but also the membrane where it appears. The construction of the VASS G simulating S is similar to the construction described in the second part of the proof of Theorem 7. In fact the construction also works for ECPS+. Since a sequential 2-membrane CPS can simulate a VASS, we have:

Theorem 10. *The following are equivalent: VASS, sequential 2-membrane CPS, sequential 1-membrane ECPS, sequential multi-membrane ECPS, and sequential multi-membrane ECPS+.*

Finally, we observe that Theorem 8 extends to multi-membrane CPS:

Theorem 11. *For any n, a multi-membrane ECPS+ operating under $\leq n$-***Parallel** *mode or n-***Parallel** *mode is equivalent to a VASS.*

6 Conclusion

We showed in this paper that P systems that compute in sequential or limited parallel mode are strictly weaker than systems that operate with maximal parallelism for two classes of systems: multi-membrane catalytic systems and multi-membrane communicating P systems. Our proof techniques can be used to show that many of the P systems that have been studied in the literature (including ones with membrane dissolving rules) operating under sequential or limited parallelism with unprioritized rules can be simulated by vector addition systems.

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. F. Bause. On the analysis of Petri net with static priorities. *Acta Informatica*, 33, 1996.
3. E. Csuhaj-Varju, J. Dassow, J. Kelemen, and Gh. Paun. *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*. Gordon and Breach Science Publishers, Inc., 1994.
4. Z. Dang and O. H. Ibarra. On P systems operating in sequential and limited parallel modes. In *Pre-Proc. 6th Workshop on Descriptional Complexity of Formal Systems*, 2004.
5. J. Esparza. Petri nets, commutative context-free grammars, and basic parallel processes. In *Proc. Fundamentals of Computer Theory*, volume 965 of *Lecture Notes in Computer Science*, pages 221–232. Springer, 1995.
6. P. C. Fischer. Turing machines with restricted memory access. *Information and Control*, 9:364–379, 1966.
7. P. C. Fischer, A. R. Meyer, and A. L. Rosenberg. Counter machines and counter languages. *Mathematical Systems Theory*, 2:265–283, 1968.
8. R. Freund. Sequential P-systems. Available at *http://psystems.disco.unimib.it*, 2000.
9. R. Freund, L. Kari, M. Oswald, and P. Sosik. Computationally universal P systems without priorities: two catalysts are sufficient. Available at *http://psystems.disco.unimib.it*, 2003.
10. R. Freund and A. Paun. Membrane systems with symport/antiport rules: universality results. In *Proc. WMC-CdeA2002*, volume 2597 of *Lecture Notes in Computer Science*, pages 270–287. Springer, 2003.
11. L. Fribourg and H. Olsén. Proving safety properties of infinite state systems by compilation into Presburger Arithmetic. In *CONCUR 1997*, volume 1243 of *Lecture Notes in Computer Science*, pages 213–227. Springer-Verlag, 1997.
12. M. H. Hack. The equality problem for vector addition systems is undecidable. In *C.S.C. Memo 121, Project MAC, MIT*, 1975.
13. J. Hopcroft and J.-J. Pansiot. On the reachability problem for 5-dimensional vector addition systems. *Theoretical Computer Science*, 8(2):135–159, 1979.
14. D.T. Huynh. Commutative grammars: The complexity of uniform word problems. *Information and Control*, 57:21–39, 1983.
15. O. Ibarra, Z. Dang, and O. Egecioglu. Catalytic P systems, semilinear sets, and vector addition systems. *Theoretical Computer Science*, 11(1):167–181, 2004.
16. E. Mayr. Persistence of vector replacement systems is decidable. *Acta Informat.*, 15:309–318, 1981.
17. Gh. Paun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.
18. Gh. Paun. *Membrane Computing: An Introduction*. Springer-Verlag, 2002.
19. Gh. Paun and G. Rozenberg. A guide to membrane computing. *Theoretical Computer Science*, 287(1):73–100, 2002.
20. P. Sosik. P systems versus register machines: two universality proofs. In *Pre-Proceedings of Workshop on Membrane Computing (WMC-CdeA2002), Curtea de Arges, Romania*, pages 371–382, 2002.
21. J. van Leeuwen. A partial solution to the reachability problem for vector addition systems. In *Proceedings of STOC'74*, pages 303–309.
22. H. Yen. On reachability equivalence for BPP-nets. *Theoretical Computer Science*, 179, 1997.
23. H. Yen. Priority conflict-free Petri nets. *Acta Informatica*, 35(8), 1998.

APPENDIX

Proof of Corollary 1

Proof. For part 1, let G be the communication-free VAS constructed in the first part of the proof of Theorem 2. Then $R(G) \subseteq \mathbf{N}^{mn+s}$ is semilinear (since a communication-free VAS has a semilinear reachability set). Now the first mn components of each vector in $R(G)$ correspond to the multiplicities of $a_1, ..., a_n$ for each of the m membranes, and the remaining s components correspond to $d_1, ..., d_s$. Clearly, from the proof of the theorem, $R(S) = proj_{mn}(R(G) \cap (\mathbf{N}^{mn} \times \{0\}^s))$, where $proj_{mn}$ is the projection of the tuples on the first mn coordinates. Since $\mathbf{N}^{mn} \times \{0\}^s$ is semilinear and semilinear sets are closed under intersection and projection, it follows that $R(S)$ is semilinear.

Now consider $R_h(S)$. A noncatalyst b is useful for membrane q if there is a rule in membrane q with Cb on the left hand side for some catalyst C. Without loss of generality (by simple relabeling), assume that the last t positions in the $(mn + s)$-tuples of transition rules correspond to useful symbols. Note that the same symbol b can not be useful for more than one membrane because in the configurations, we distinguish a symbol occurring in membrane q from the same symbol occurring in another membrane p. Clearly, a reachable configuration in $R(S)$ is halting if coordinates $(mn - t + 1), ..., mn$ (corresponding to the useful symbols) are zero. Hence $R_h(S) = R(S) \cap (\mathbf{N}^{mn-t} \times \{0\}^t)$, which is semilinear.

Part 2 follows from the NP-completeness of the reachability problem for communication-free Petri nets (which are equivalent to commutative context-free grammars) [14, 5]. ∎

Proof of Theorem 4

Proof. Let the set of states of M be $\{q_1, ..., q_k\}$, where q_1 the initial state and q_k the final state. W.l.o.g., we assume that whenever M enters an accepting state, both counters are empty. Furthermore, M runs forever if it does not accept. In what follows, we show how to construct an n-dimensional communication-free VAS $G = \langle x, (W_1, W_2, W_3) \rangle$ meeting Statement (1) of the theorem. To better explain how VAS G functions, we divide the n coordinates into the following:

- l: the designated coordinate,
- c_1 and c_2: keep track of the values of the two counters C_1 and C_2, respectively,
- y and y': serve for the purpose of generating an arbitrary number in c_1 and l for the simulation to start,
- c_1' and c_2': serve as temporary holders for decrementing C_1 and C_2, respectively, or testing C_1 and C_2 for zero, respectively,
- $p_1, ..., p_k$: simulate the k states of M in such a way that at any instant, only one of $p_1, ..., p_k$ is non-zero (indicating the current state of M). Furthermore, if from state p_i the next transition to simulate is of type 'increment', then p_i = 2; otherwise, p_i = 1. (More will be said later about how this is done.)
- $s^h_{(i,j),d}, 1 \leq h \leq 5$ ($d = 1$ or 2): serve for simulating transition $q_i \overset{C_d:=C_d-1}{\rightarrow} q_j$,
- $t^1_{(i,j),d}$ and $t^2_{(i,j),d}$ ($d = 1$ or 2): serve for simulating transition $q_i \overset{C_d=0}{\rightarrow} q_j$,
- f and f': record the occurrence of an illegitimate simulation step during the course of the simulation.

The start vector x of G is $x(y) = 1$, while the remaining coordinates are zero. By using the set of addition vectors given in Table 1, an arbitrary value m can be generated in both c_1 and l through the sequence listed in Table 2. Note that in Table 2, only the values of coordinates y, y', c_1, l, p_1, f are shown; the rest are zero throughout.

The coordinate f plays the role of enforcing nontermination of G whenever f =1. To this end, the addition vectors listed in Table 3 are used.

Regarding c_1, c_2, c_1' and c_2', the addition vectors listed in Table 4 are involved. It is worth mentioning that during the course of Steps 2 - 2m+1 in Table 2, r_1 and h_1 (or h_2) are both enabled, and the two are of the same type W_1. If r_1 (feasible when $c_1 > 0$) is ever applied at some point in time, then a type W_3 vector α_1 (or α_2) is forced to take place due to the max-parallelism rule – yielding $f = 1$.

addition vector	W_1	W_2	W_3	y	y'	c_1	l	p_1	f
h_1	*			-1	1	1	1		
h_2	*			1	-1				
h_3	*			-1				1	
α_1			*	-1					1
α_2			*		-1				1

Table 1. Addition vectors associated with the initial phase of the simulation. A '*' in columns W_1, W_2, W_3 indicates the type to which the addition vector belongs. In each of columns y, y', c_1, l, p_1 and f, a blank denotes a zero.

step	addition vectors used	y	y'	c_1	l	p_1	f
0		1	0	0	0	0	0
1	(h_1,0,0) $\Rightarrow$	0	1	1	1	0	0
2	(h_2,0,0) $\Rightarrow$	1	0	1	1	0	0
3	(h_1,0,0) $\Rightarrow$	0	1	2	2	0	0
4	(h_2,0,0) $\Rightarrow$	1	0	2	2	0	0
...							
2m-1	(h_1,0,0) $\Rightarrow$	0	1	m	m	0	0
2m	(h_2,0,0) $\Rightarrow$	1	0	m	m	0	0
2m+1	(h_3,0,0) $\Rightarrow$	0	0	m	m	1	0

Table 2. A sequence generating an arbitrary number m in coordinates c_1 and l.

addition vector	W_1	W_2	W_3	f	f'
z			*	-1	1
z'			*	1	-1

Table 3. Forcing G to be nonterminating when $f = 1$.

addition vector	W_1	W_2	W_3	c_1	c_1'	c_2	c_2'	f
r_1	*			-1	1			
r_2		*				-1	1	
r_1'	*				-1			
r_2'		*					-1	
β_1			*		-1			1
β_2			*				-1	1

Table 4. Addition vectors associated with the two counters.

addition vector	W_1	W_2	W_3	p_i	$s^1_{(i,j),2}$	$s^2_{(i,j),2}$	$s^3_{(i,j),2}$	$s^4_{(i,j),2}$	$s^5_{(i,j),2}$	p_j	f
w_1	*			-1	1	1					
w_2	*				-1		1				
w_3		*				-1		1			
w_4	*						-1		1		
w_5	*							-1			
w_6		*							-1	1 (@)	
γ_1			*	-1							1
γ_2			*		-1						1
γ_3			*				-1				1
γ_4			*					-1			1
γ_5			*						-1		1

Table 5. Addition vectors associated with the simulation of $q_i \overset{C_d:=C_d-1}{\rightarrow} q_j$. (@): Note that if from p_j an 'add one to a counter' is the next transition, then the '1' in the p_j's column is replaced by '2'.

We are now in a position to see how the three types of transitions of M can be faithfully simulated in G. First consider $q_i \overset{C_d:=C_d-1}{\rightarrow} q_j$. Addition vectors shown in Table 5 serve for the purpose of simulating the such a transition. Note that Table 5 is for $d = 2$; $d = 1$ is symmetric.

To see how $q_i \overset{C_2:=C_2-1}{\rightarrow} q_j$ can be simulated faithfully, we focus on the following 12 coordinates that are involved in the simulation, i.e.,

$$c_1, c_2, c_1', c_2', p_i, s^1_{(i,j),d}, s^2_{(i,j),d}, s^3_{(i,j),d}, s^4_{(i,j),d}, s^5_{(i,j),d}, p_j, f$$

Suppose that the simulation begins with vector $(...a, b, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0...)$ denoting a configuration of M in which the values of counters C_1 and C_2 are a and b, respectively, and the current state is q_i. First consider the case when $a > 0$ and $b > 0$. In this vector, only r_1, r_2, γ_1 and w_1 are enabled. Since r_1 and w_1 belong to the same group W_1, only one of them is feasible. Hence, there are two feasible steps:

$$(...a, b, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0...) \overset{(r_1,r_2,\gamma_1)}{\Rightarrow} (...a-1, b-1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 1...) \text{ or}$$
$$(...a, b, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0...) \overset{(w_1,r_2,0)}{\Rightarrow} (...a, b-1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0...)$$

Clearly the former gives rise to a vector in which f=1, rendering the simulation nonterminating. From $(...a, b-1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0...)$, it is easy to see that the sequence of steps listed in Table 6 is legitimate.

step	addition vectors used	c_1	c_2	c_1'	c_2'	p_i	$s^1_{(i,j),d}$	$s^2_{(i,j),d}$	$s^3_{(i,j),d}$	$s^4_{(i,j),d}$	$s^5_{(i,j),d}$	p_j	f
0		a	b	0	0	1	0	0	0	0	0	0	0
1	$(w_1, r_2, 0) \Rightarrow$	a	b-1	0	1	0	1	1	0	0	0	0	0
2	$(w_2, r_2', 0) \Rightarrow$	a	b-1	0	0	0	0	1	1	0	0	0	0
3	$(w_4, w_3, 0) \Rightarrow$	a	b-1	0	0	0	0	0	0	1	1	0	0
4	$(w_5, w_6, 0) \Rightarrow$	a	b-1	0	0	0	0	0	0	0	0	1	0

Table 6. A successful simulation of $q_i \overset{C_2:=C_2-1}{\rightarrow} q_j$.

A careful examination of Table 4 reveals that along the sequence in Table 6, r_1 or r_2 might be enabled in each of the intermediate vectors during Steps 2-4. However, executing r_1 or r_2 in Steps 2-4 forces one of $\beta_2, \gamma_1, ..., \gamma_5$ to take place under the 3-**Max-Parallelism** mode, which in turn yields $f = 1$. In view of the above, the sequence listed above is the only way that the computation of G has a potential to terminate, and

such a computation corresponds to a faithful simulation of $q_i \overset{C_2:=C_2-1}{\rightarrow} q_j$. The case when $a = 0$ and $b > 0$ is similar. Now consider the case when $C_2 = 0$ (i.e., b=0) to begin with. Then again w_1 must be involved in the first step (otherwise, γ_1 will result in $f = 1$) which inhibits r_1 from being executed. So we have the sequence in Table 7.

step	addition vectors used	c_1	c_2	c_1'	c_2'	p_i	$s^1_{(i,j),d}$	$s^2_{(i,j),d}$	$s^3_{(i,j),d}$	$s^4_{(i,j),d}$	$s^5_{(i,j),d}$	p_j	f
0		a	0	0	0	1	0	0	0	0	0	0	0
1	$(w_1, 0, 0) \Rightarrow$	a	0	0	0	0	1	1	0	0	0	0	0
2	$(w_2, w_3, 0) \Rightarrow$	a	0	0	0	0	0	0	1	1	0	0	0
3	$(w_4, 0, \gamma_4), or$												1
3'	$(w_5, 0, \gamma_3), or$												1
3"	$(r_1, 0, \gamma_4) \Rightarrow$												1

Table 7. A fail simulation of $q_i \overset{C_2:=C_2-1}{\rightarrow} q_j$.

In $(...a, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0...)$, both w_4 and w_5 are enabled, yet only one of them can be executed due to the fact that both are of type W_2. Hence, one of γ_3 and γ_4 is forced to take place, resulting in $f = 1$. From the discussion above, $f = 1$ will always happen during the course of the simulation. This in turn implies that simulating $q_i \overset{C_2:=C_2-1}{\rightarrow} q_j$ while $C_2 = 0$ always results in G being nonterminating.

The crux of the simulation of $q_i \overset{C_2:=C_2-1}{\rightarrow} q_j$ lies in that if $b > 0$, then r_2, together with w_1, has to be applied in the first step. With $c_2' = 1$ (as a result of applying r_2), r_2' can then be applied to 'delay' the execution of w_3, since both are of type W_2. Notice that $s^2_{(i,j),2}$ is a coordinate at which no type W_3 vector, i.e., adding one to f, is defined. Hence, its outgoing vector can be delayed as long as it is needed. On the other hand, if $b = 0$ initially, then w_2 and w_3 must be applied at the same time in the second step, which eventually results in $f = 1$ as our earlier discussion concludes.

Now we turn our attention to simulating $q_i \overset{C_d=0}{\rightarrow} q_j$. To this end, Table 8 lists the vectors used for the case $d = 2$. ($d = 1$ is similar.)

addition vector	W_1	W_2	W_3	p_i	$t^1_{(i,j),2}$	$t^2_{(i,j),2}$	p_j	f
v_1	*			-1	1	1		
v_2		*			-1		1(@)	
v_3	*					-1		
π_1			*	-1				1
π_2			*		-1			1
π_3			*			-1		1

Table 8. Addition vectors associated with $q_i \overset{C_2=0}{\rightarrow} q_j$. (@): Note that if from p_j an 'add one to a counter' is the next transition, then the '1' in the p_j's column is replaced by '2'.

In what follows, we focus on the following 9 coordinates that are relevant to the simulation of $q_i \overset{C_2=0}{\rightarrow} q_j$: $c_1, c_2, c_1', c_2', p_i, t^1_{(i,j),2}, t^2_{(i,j),2}, p_j, f$. Suppose that the simulation begins in vector $(...a, b, 0, 0, 1, 0, 0, 0, 0...)$. First consider the case when $b = 0$ and $a > 0$. In this case, Table 9 records a success in simulating $q_i \overset{C_2=0}{\rightarrow} q_j$ since p_j (the coordinate corresponding to state q_j) becomes 1. Notice that v_1, v_2, v_3 must be applied in such a sequence; otherwise, one of π_1, π_2, π_3 will be executed, resulting in $f = 1$ in the end. Now suppose in the start vector, $C_2 \neq 0$ (i.e., $b > 0$). Operating in the 3-Max-Parallel mode forces the step

$(...a, b, 0, 0, 1, 0, 0, 0, 0...) \overset{(v_1, r_2, 0)}{\Rightarrow} (...a, b-1, 0, 1, 0, 1, 1, 0, 0...)$. Now in $(...a, b-1, 0, 1, 0, 1, 1, 0, 0...)$, one of β_2 and π_2 has to be taken since r_2' and v_2 are of type W_2. This makes $f = 1$.

step	addition vectors	c_1	c_2	c_1'	c_2'	p_i	$t^1_{(i,j),d}$	$t^2_{(i,j),d}$	p_j	f
0		a	0	0	0	1	0	0	0	0
1	$(v_1, 0, 0) \Rightarrow$	a	0	0	0	0	1	1	0	0
2	$(v_3, v_2, 0) \Rightarrow$	a	0	0	0	0	0	0	1	0

Table 9. A successful simulation of $q_i \overset{C_2=0}{\rightarrow} q_j$.

Now we show how $q_i \overset{C_d:=C_d+1}{\rightarrow} q_j$ is simulated. Table 10 shows the addition vectors involved for the case $d = 2$. ($d = 1$ is similar.) Since in our design we always have $p_i = 2$ prior the simulation of this transition, one must apply both x_1 and x_2 simultaneously; otherwise, δ_1 is forced to be taken which renders $f = 1$. The applications of x_1 and x_2 not only increment c_2 by one, but also inhibit r_1 and r_2 from been executed.

addition vector	W_1	W_2	W_3	p_i	c_2	p_j	f
x_1		*		-1	1	1 (@)	
x_2	*			-1			
δ_1			*	-1			1

Table 10. Addition vectors associated with $q_i \overset{C_2:=C_2+1}{\rightarrow} q_j$. (@): Note that if from p_j an 'add one to a counter' is the next transition, then the '1' in the p_j's column is replaced by '2'.

Finally, an addition vector decrementing p_k (the coordinate corresponding to the accepting state of M) is included in W to force halting of G as soon as M reaches an accepting state. Upon halting, the value of l (i.e., the designated coordinate) is m. Clearly, G does not halt if M never accepts, or during the course of the simulation, an illegitimate simulation step is carried out. This completes the proof of Statement (1) of the theorem. The proof of (2) can be carried out along a similar line as that of (1), and hence, is left to the reader. ∎

Proof of Lemma 1

Proof. We sketch the construction of U. Let $L \subseteq \mathbf{N}$ be a recursively enumerable set of nonnegative integers accepted by a deterministic TM M with a *one-way* unary read-only input (with endmarkers) and one (two-way) read-write worktape, i.e., M when given n on its input tape (in unary with endmarkers), computes and accepts if n is in L.

It is well known that we can construct a universal deterministic TM U_1 with two read-only input tapes and one read-write tape that operates as follows. U_1, when given n and a description x_M over the binary alphabet $\{1, 2\}$ of a deterministic TM M on its two input tapes, will simulate the computation of M on n and accepts iff M on input n accepts.

First, we convert U_1 to an equivalent universal deterministic TM U_2, where the description of x_M is given as a unary string $Tally(x_M) = 1^{Num(x_M)}$, where $Num(x_M)$ is x_M interpreted as a number in 2-adic notation. The idea is for U_2 to read $Tally(x_M)$ on the input and convert it into 2-adic representation x_M on the first track of the worktape. Then U_2 uses x_M to simulate U on n using the second track of the worktape. So now U_2 has two read-only unary input tapes (with endmarkers) and a worktape.

Next, we convert the worktape of U_2 to a finite number of counters. Three counters are sufficient to simulate the TM worktape, as shown in [6, 7]. The new machine U_3 will now have two unary inputs and and three counters.

Then we convert U_3 to a deterministic counter machine U_4 which will use two counters to simulate the two unary input tapes (note that the unary input tapes are one-way) and three working counters, all of which are initially zero.

Thus, U_4 has 5 counters. Initially, the first counter contains n, the second counter contains $Tally(x_M)$, and the other 3 counters are zero.

Finally, we convert U_4 to a universal 2-counter machine U. Call the two counters of U C_1 and C_2. Initially C_1 has value n and $C_2 = p_1^0 p_2^{Tally(x_M)} p_3^0 p_4^0 p_5^0$. Here $p_1, ..., p_5$ are the first 5 prime numbers, and the values of the 5 counters of U_4 will be represented as the exponents of the prime numbers.

U first encode n (the initial value of C_1) into counter C_2, so that at the end of the process, $C_1 = 0$ and C_2 contains $p_1^n p_2^{Tally(x_M)} p_3^0 p_4^0 p_5^0$. (Thus, the 'exponent' of p_1 will now represent the value of the first counter). Then U simulates the computation of U_4 using only the two counters C_1 and C_2 and accepts iff U_4 accepts. The simulation is straightforward. ∎

Proof of Theorem 5

Proof. (Proof sketch) To demonstrate universality, we show the ability for this type of CS to simulate arbitrary 2-counter machines. Let M be a 2-counter machine with two counters C_1 and C_2, and the set of states of M be $\{q_1, ..., q_k\}$, where q_1 the initial state and q_k the final state. A configuration of M is a triple (q, i, j), where q is a state and i and j ($\in N$) are the values of counters C_1 and C_2, respectively. W.l.o.g., we assume that whenever M enters an accepting state, both counters are empty. In the remainder of the proof, we show how to construct a CS S with initial configuration C^9x in such a way that M accepts on initial counter values $C_1 = m$ and $C_2 = 0$ iff for some designated noncatalyst a,

1. (generator:) $m \in \{\#_a(y) \mid y \in R_h(S)\}$,
2. (acceptor:) initial configuration is $d^n y$ and $R_h(S) \neq \emptyset$, for some y not containing a.

We first show the *generator* case. The meanings of the noncatalysts and rules used in the construction will be elaborated in detail as our discussion progresses. There is a special noncatalyst f in the CS S, which serves as an 'error' indicator. Whenever f appears in some configuration, CS S never halts. This can be guaranteed by including rule $Cf \to Cf$ in S. Furthermore, f ever appears iff during the course of the simulation, an illegitimate step is taken. To keep track of the values of the two counters C_1 and C_2 and the states of M, we introduce the following noncatalysts:

- c_1, c_1', c_2, c_2': if C^9x corresponds to configuration (q, i, j) of M during the course of the simulation, then $\#_{c_1}(x) + \#_{c_1'}(x) = i$, $\#_{c_1'}(x) \leq 1$, $\#_{c_2}(x) + \#_{c_2'}(x) = j$, $\#_{c_2'}(x) \leq 1$. In words, the number of occurrences of c_1 and c_1' (resp., c_2 and c_2') together records the value of counter C_1 (resp., C_2).
- a_1 and a_2: their presence or absence reflects the status of the counters, i.e., whether C_1 and C_2, respectively, are empty or not. In any reachable configuration C^9x, $\#_{a_1}(x), \#_{a_2}(x) \leq 2$.
- p_1, ..., p_k: associated with states of M. At any point, one and only one of p_1, ..., p_k appears in the configuration of S in the course of the simulation.

We also have the following rules associated with counters C_1 and C_2:

- r_1: $Cc_1 \to Cc_1'a_1$
- r_1': $Cc_1' \to Cc_1$
- r_2: $Cc_2 \to Cc_2'a_2$
- r_2': $Cc_2' \to Cc_2$
- α_1: $Ca_1a_1 \to Cf$
- α_2: $Ca_2a_2 \to Cf$

To give the reader a better feel for how the construction works, we first give the intuitions in the following. Like VAS (or Petri nets), catalytic systems also lack an explicit 'zero-test' capability, i.e., there is no rule whose application relies on a certain noncatalyst being absent in a configuration. As a result, the ability to simulate 'test-for-zero' by taking advantage of the system operating under the max-parallel mode is crucial.

Take counter C_1 for instance. The way we accomplish 'test counter C_1 for zero' is by periodically applying rules r_1 and r_1' alternatively to interchange c_1 (if it exists) with c_1'. While in each of such alternations, a noncatalyst a_1 is created by rule r_1, serving as an indicator telling whether counter C_1 is zero or not. In the process of simulating either 'incrementing' or 'decrementing' counter C_1, such an a_1 (possibly multiple copies) will always be synchronized with some noncatalysts evolving from the 'state' noncatalyst from which the simulation of the current transition begins. One can think of the simulation of a transition (from state q) involves two *threads* evolving from the noncatalyst representing q. At some point in time, the two threads are to be synchronized with those a_1 and a_2 possibly spun off from c_1 and c_2, respectively. a_1 and a_2 will then be consumed while the action (either an increment or a decrement)on the counter is taken. However, if '$C_1 = 0$? is the underlying transition to simulate, in our design the generated a_1 (indicating a nonempty C_1) has no chance to be synchronized properly, and then eventually a_1 contributes to the occurrence of an f making S nonterminating. With the presence of a_1, simulating 'test C_1 for zero' eventually fails.

To illustrate how such a synchronization is carried out, consider the case when a transition of M from state q_i to q_j is to be simulated and the CS S starts with confi guration $C^9 p_i (c_1)^{m_1} (c_1')^{n_1} (a_1)^{l_1} (c_2)^{m_2} (c_2')^{n_2} (a_2)^{l_2} x$, for some $m_1, n_1, l_1, m_2, n_2, l_2 \in N, n_1, n_2 \leq 1, l_1, l_2 \leq 2$. Depending on the type of M's transition, what follows explains the evolution of the catalytic system with the operations on the counters omitted.

First, two noncatalysts d_1 and d_2 are generated through a rule of the form $Cp_i \to Cd_1d_2$, and subsequently, d_i ($i = 1$ or 2) evolves into new noncatalysts $d_i^1, d_i^2, d_i^3, ...d_i^h$ until the number of a_i becomes 2. (Notice that for different values of m_i, n_i, l_i, the number of steps (i.e., h) for a_i to become 2 varies.) Using rules $Ca_ia_i \to Cd_i'$ and $Cd_i^h \to Cd_i'$, a confi guration containing $(d_i')^2$ can be reached, and then a rule of type $Cd_i'd_i' \to Cp_j$ marks the end of the simulation of the current transition. By introducing rule $Cd_i' \to Cf$, rules $Ca_ia_i \to Cd_i'$ and $Cd_i^h \to Cd_i'$ must be applied simultaneously (in order to create two d_i' for $Cd_i'd_i' \to Cp_j$ to be applicable next); otherwise, $Cd_i' \to Cf$ is forced to be taken (guaranteed by max-parallelism) if only one d_i' occurs, resulting in the occurrence of an f.

We now begin showing the rest of the CS S, and how M's transitions are simulated.

(Increment counter C_1: $q_i \overset{C_1:=C_1+1}{\to} q_j$)

It is important to point out that each of the newly introduced noncatalyst should be annotated by i and j (i.e., for different i, j, a new set of noncatalysts is used); nevertheless, for the sake of clarity in notation, such indices are omitted in our subsequent discussion. Suppose that the current confi guration of S is $C^9 x$. With respect to the numbers of occurrences of c_1, c_1', a_1, c_2, c_2', a_2 in x, we consider the following cases:

• Case I-1: $(\#_{c_1}(x) > 0) \wedge (\#_{c_1'}(x) = 1) \wedge (\#_{a_1}(x) = 2) \wedge (\#_{c_2}(x) > 0) \wedge (\#_{c_2'}(x) = 1) \wedge (\#_{a_2}(x) = 2)$: In this case, two additional noncatalysts d_1^1 and d_2^1 together with the following rules are in S: (Notice that in those noncatalysts listed below, each superscript indicates the type of the case.)

- Rule A_1^1: $Cp_i \to Cd_1^1 d_2^1$
- Rule A_2^1: $Ca_1a_1 \to Cd_1^1$
- Rule A_3^1: $Ca_2a_2 \to Cd_2^1$
- Rule A_4^1: $Cd_1^1 d_1^1 \to Cc_1p_j$
- Rule A_5^1: $Cd_2^1 d_2^1 \to C$
- Rule β_1: $Cd_1^1 \to Cf$
- Rule β_2: $Cd_2^1 \to Cf$

Suppose that we begin in confi guration $C^9 p_i (c_1)^m c_1' (a_1)^2 (c_2)^n c_2' (a_2)^2$. Operating in the max-parallel mode yields the following computation:

step	rules applied	confi guration
0		$C^9 p_i (c_1)^m c_1' (a_1)^2 (c_2)^n c_2' (a_2)^2$
1	$r_1, r_1', r_2, r_2', A_1^1, A_2^1, A_3^1$	$C^9 (c_1)^m c_1' a_1 (d_1^1)^2 (c_2)^n c_2' a_2 (d_2^1)^2$
2	$r_1, r_1', r_2, r_2', A_4^1, A_5^1$	$C^9 p_j (c_1)^{m+1} c_1' (a_1)^2 (c_2)^n c_2' (a_2)^2$

It is easy to observe that if any of A_2^1, A_3^1, A_4^1 and A_5^1 is not applied in Steps 1 and 2, then one of α_1, α_2, β_1, β_2, or a rule replacing a_1a_1 (or a_2a_2) by a noncatalyst in simulating a different transition must take place – eventually leading to the creation of an f.

• Case I-2: $(\#_{c_1}(x) > 1) \wedge (\#_{c'_1}(x) = 0) \wedge (\#_{a_1}(x) = 0) \wedge (\#_{c_2}(x) > 0) \wedge (\#_{c'_2}(x) = 1) \wedge (\#_{a_2}(x) = 2)$: The following rules are in S:

- Rule A_1^2: $Cp_i \to Cd_{1,1}^2 d_{2,1}^2$
- Rule A_2^2: $Ca_2a_2 \to Cd_{2,1}^2$
- Rule A_3^2: $Cd_{1,1}^2 \to Cd_{1,2}^2$
- Rule A_4^2: $Cd_{2,1}^2 d_{2,1}^2 \to Cd_{2,2}^2$
- Rule A_5^2: $Cd_{1,2}^2 \to Cd_{1,3}^2$
- Rule A_6^2: $Cd_{2,2}^2 \to Cd_{2,3}^2$
- Rule A_7^2: $Ca_1a_1 \to Cd_{1,3}^2$
- Rule A_8^2: $Ca_2a_2 \to Cd_{2,3}^2$
- Rule A_9^2: $Cd_{1,3}^2 d_{1,3}^2 \to Cc_1p_j$
- Rule A_{10}^2: $Cd_{2,3}^2 d_{2,3}^2 \to C$
- Rules β: $Cd_{1,h}^2 \to Cf, Cd_{2,h}^2 \to Cf, 1 \le h \le 3$

Suppose that we begin in configuration $C^9p_i(c_1)^m(c_2)^n c'_2(a_2)^2$. Operating in the max-parallel mode yields the following computation:

step	rules applied	configuration
0		$C^9p_i(c_1)^m(c_2)^n c'_2(a_2)^2$
1	$r_1, r_2, r'_2, A_1^2, A_2^2,$	$C^9(c_1)^{m-1}c'_1a_1d_{1,1}^2(c_2)^n c'_2a_2(d_{2,1}^2)^2$
2	$r_1, r'_1, r_2, r'_2, A_3^2, A_4^2,$	$C^9(c_1)^{m-1}c'_1(a_1)^2d_{1,2}^2(c_2)^n c'_2(a_2)^2d_{2,2}^2$
3	$r_1, r'_1, r_2, r'_2, A_5^2, A_6^2, A_7^2\ A_8^2$	$C^9(c_1)^{m-1}c'_1a_1(d_{1,3}^2)^2(c_2)^n c'_2a_2(d_{2,3}^2)^2$
4	$r_1, r'_1, r_2, r'_2, A_9^2\ A_{10}^2$	$C^9p_j(c_1)^m c'_1(a_1)^2(c_2)^n c'_2(a_2)^2$

Again, if any one of $A_1^2 ... A_{10}^2$ does not appear in the above sequence, one of those rules in β has to be applied due to max-parallelism. Also notice that if, instead of A_2^2, A_7^2, A_8^2, one of A_2^1 and A_3^1 is applied in steps 1 and 3 (i.e., a_1 or a_2 is synchronized with the wrong case of the simulation), then it is reasonably easy to see that one of α or β rules has to be applied.

There are several other cases, which are all similar to the above.

(Decrement counter C_1: $q_i \overset{C_1:=C_1-1}{\to} q_j$)

Consider

• Case D-1: $(\#_{c_1}(x) > 0) \wedge (\#_{c'_1}(x) = 1) \wedge (\#_{a_1}(x) = 2) \wedge (\#_{c_2}(x) > 0) \wedge (\#_{c'_2}(x) = 1) \wedge (\#_{a_2}(x) = 2)$: In this case, three additional noncatalysts e_1^1, e_2^1, e_3^1 together with the following rules are in S:

- Rule B_1^1: $Cp_i \to Ce_1^1e_2^1e_3^1$
- Rule B_2^1: $Ca_1a_1 \to Ce_1^1$
- Rule B_3^1: $Ca_2a_2 \to Ce_2^1$
- Rule B_4^1: $Cc'_1 \to Ce_3^1$
- Rule B_5^1: $Ce_1^1e_1^1 \to Cp_j$
- Rule B_6^1: $Ce_2^1e_2^1 \to C$
- Rule B_7^1: $Ce_3^1e_3^1 \to C$
- Rule γ_1: $Ce_1^1 \to Cf$
- Rule γ_2: $Ce_2^1 \to Cf$
- Rule γ_3: $Ce_3^1 \to Cf$

Suppose that we begin in configuration $C^9p_i(c_1)^m c'_1(a_1)^2(c_2)^n c'_2(a_2)^2$. Operating in the max-parallel mode yields the following computation:

step	rules applied	configuration
0		$C^9p_i(c_1)^m c'_1(a_1)^2(c_2)^n c'_2(a_2)^2$
1	$r_1, r_2, r'_2, B_1^1, B_2^1, B_3^1, B_4^1$	$C^9(c_1)^{m-1}c'_1a_1(e_1^1)^2(c_2)^n c'_2a_2(e_2^1)^2(e_3^1)^2$
2	$r_1, r'_1, r_2, r'_2, B_5^1, B_6^1, B_7^1$	$C^9p_j(c_1)^{m-1}c'_1(a_1)^2(c_2)^n c'_2(a_2)^2$

It should be noted that if $c'_1 = 0$ (i.e., c'_1 is absent) to start with, then after step 1, there is only one occurrence of e_3^1 in the configuration, which in turn forces γ_3 to be taken next. (Rule B_7^1 requires two copies of e_3^1.) The

absence of c_1' could be the result of either counter C_1 being empty, or the wrong case of the simulation being involved. In either case, CS S then becomes nonterminating.

The remaining cases are similar to the above, and hence, are left to the reader. For the case similar to Case I-2 (i.e., the second case of 'increment'), the degree of maximal parallelism needed to simulate 'decrement' is 9, one plus the maximum number utilized in I-2. The extra one comes from the fact that to simulate a 'decrement', three *threads* (captured by e_1^1, e_2^1, e_3^1) are required as opposed to two threads as was the case in I-2.

Finally, we consider the simulation of a 'test-for-zero' transition.

(Test counter C_1 for zero: $q_i \overset{C_1=0}{\rightarrow} q_j$)

Consider

• Case Z-1: $(\#_{c_1}(x) > 0) \wedge (\#_{c_1'}(x) = 0) \wedge (\#_{a_1}(x) = 0) \wedge (\#_{c_2}(x) > 0) \wedge (\#_{c_2'}(x) = 1) \wedge (\#_{a_2}(x) = 2)$: In this case, the following rules are in S:

- Rule T_1^1: $Cp_i \rightarrow Cg_{2,1}^1 g_{3,1}^1$
- Rule T_2^1: $Ca_2a_2 \rightarrow Cg_{2,1}^1$
- Rule T_3^1: $Cg_{2,1}^1 g_{2,1}^1 \rightarrow Cg_{2,2}^1$
- Rule T_4^1: $Cg_{3,1}^1 \rightarrow Cg_{3,2}^1$
- Rule T_5^1: $Ca_2a_2 \rightarrow Cg_{2,3}^1$
- Rule T_6^1: $Cg_{2,2}^1 \rightarrow Cg_{2,3}^1$
- Rule T_7^1: $Cg_{3,2}^1 \rightarrow Cg_{3,3}^1$
- Rule T_8^1: $Cg_{2,3}^1 g_{2,3}^1 \rightarrow Cp_j$
- Rule T_9^1: $Cg_{3,3}^1 \rightarrow C$
- Rules π: $Cg_{h,2}^1 \rightarrow Cf, Cg_{h,3}^1 \rightarrow Cf, 1 \leq h \leq 3$

Suppose that the current configuration is $C^9 p_i c_1 (c_2)^n c_2' (a_2)^2$ (i.e., counter C_1 is not zero). Operating in the max-parallel mode yields the following computation:

step	rules applied	configuration
0		$C^9 p_i c_1 (c_2)^n c_2' (a_2)^2$
1	$r_1, r_2, r_2', T_1^1, T_2^1$	$C^9 c_1' a_1 (g_{2,1}^1)^2 (c_2)^n c_2' a_2 g_{3,1}^1$
2	$r_1', r_2, r_2', T_3^1, T_4^1$	$C^9 c_1 a_1 g_{2,2}^1 (c_2)^n c_2' (a_2)^2 g_{3,2}^1$
3	$r_1, r_2, r_2', T_5^1, T_6^1, T_7^1$	$C^9 c_1' (a_1)^2 (g_{2,3}^1)^2 (c_2)^n c_2' a_2 g_{3,3}^1$
4	$r_1', r_2, r_2', T_8^1, T_9^1$	$C^9 p_j c_1 x (c_2)^n c_2' (a_2)^2$

Clearly in step 4, some rule of the form $Ca_1a_1 \rightarrow Cx$ has to be applied. However, our design ensures that either an f is generated (i.e., $x = f$) directly, or a noncatalyst x belonging to some other simulation stage is created. Either case leads to an f eventually. While $C_1 \neq 0$, the simulation of $q_i \overset{C_1=0}{\rightarrow} q_j$ fails. Note that using a sufficient amount of 'delay' (i.e., 4 stages in our design), the number of a_1s will be at least 2, provided that the counter is not zero.

Now suppose $C_1 = 0$, then the following describes a successful simulation:

step	rules applied	configuration
0		$C^9 p_i (c_2)^n c_2' (a_2)^2$
1	r_2, r_2', T_1^1, T_2^1	$C^9 (g_{2,1}^1)^2 (c_2)^n c_2' a_2 g_{3,1}^1$
2	r_2, r_2', T_3^1, T_4^1	$C^9 g_{2,2}^1 (c_2)^n c_2' (a_2)^2 g_{3,2}^1$
3	$r_2, r_2', T_5^1, T_6^1, T_7^1$	$C^9 (g_{2,3}^1)^2 (c_2)^n c_2' a_2 g_{3,3}^1$
4	$r_2, r_2', T_8^1, T_9^1, T_7^1$	$C^9 p_j (c_2)^n c_2' (a_2)^2$

Again, the remaining cases of 'test-for-zero' are similar to the above, and are left to the reader.

Finally, rules $Cb \rightarrow Cbc_1a; Cb \rightarrow Cp_1$ are used to initialize the initial value of counter C_1, where p_1 corresponds to the initial state, and a is the designated noncatalyst from which the output number is observed. Note that the number of a's (= the initial value of counter C_1) remains the same throughout the entire simulation. The above completes the proof for the *generator* case. The *acceptor* case is similar. ∎

Proof of Lemma 2

Proof. Along $z \xrightarrow{\sigma} z'$, let z'' be the leftmost vector at which the priority requirement is violated. Let v_1 be the addition vector applied at z'', and v_2 be one of the highest priority applicable at z''. We claim that there exists a v_2', $(v_2, v_2') \in \bar{\rho}$ and v_2' is present in the segment from z'' to z'; otherwise, v_2 would still be applicable at z' (due to the communication-freeness nature of G) – violating the assumption of the lemma. Since v_2' and v_1 do not subtract from the same coordinate, applying v_2' at z'' followed by v_1 remains a valid computation. By repeatedly applying such a rearrangement to the remaining sequence, a computation meeting the priority requirement can be constructed. ∎

Proof of Theorem 6

Proof. The lower bound follows immediately from the NP-hardness of checking reachability for the basic model of communication-free VAS, as they are special cases of their prioritized counterparts (with an empty priority relation).

To show the upper bound, suppose $x \xrightarrow{\sigma}_\rho z$ is a computation reaching z in a prioritized communication-free VAS $G = \langle x, W \rangle$ with priority relation ρ. Listing in increasing priority, let the equivalence classes induced by $\bar{\rho}$ be $\Omega_1, ..., \Omega_d$, for some d ($\leq |W|$). We let $z_1, \ldots, z_s$, $z_1', \ldots, z_s'$ be vectors and $v_1, \ldots, v_s$ be addition vectors along σ satisfying the following: (We assume $z_0' = x$, $z_s' = z$, and $v_i \in \Omega j_i, 1 \leq j_i \leq d$.)

1. $v_i \in W$ is the vector applied at z_i in σ, and $z_i \xrightarrow{v_i}_\rho z_i', \forall 1 \leq i \leq s$.
2. $\forall 1 \leq i \leq s$,
   (a) $class(v_1) < class(v_2) < \ldots < class(v_s)$,
   (b) $\forall v$ applied in $(z_{i-1}' \xrightarrow{*}_\rho z)$, $class(v) \geq j_i$, and $\forall v$ applied in $(z_i' \xrightarrow{*}_\rho z)$, $class(v) > j_i$. (In words, v_i is the rightmost occurrence among the lowest priority transitions in $z_{i-1}' \xrightarrow{*}_\rho z$.)

The crux of our subsequent analysis lies in the fact that in $z_{i-1}' \xrightarrow{*}_\rho z_i, 1 \leq i \leq s$, v_i is of the lowest priority in VAS G_i with its set of addition vectors restricted to $W_i = W - (\bigcup_{l=1,\ldots,j_{i-1}} \Omega_l)$ (The start vector of G_i is not important here.) . By Lemma 2, $z_{i-1}' \xrightarrow{*} z_i$ iff $z_{i-1}' \xrightarrow{*}_\rho z_i$ in G'. This, in conjunction with Lemma 3, enables us to set up a system of linear inequalities to capture the reachability of z from x:

We begin by guessing the following:

1. transitions $v_1, \ldots, v_s$ with $class(v_1) < \ldots < class(v_s)$,
2. $\forall 1 \leq i \leq s$, a set of coordinates P_i such that v subtracts from some coordinate in P_i, for every v with $class(v) > class(v_i)$. (P_i is the set of coordinates that are zero so that no vector of higher priority than v_i is applicable in z_i.

Then the system of linear inequalities is set up as follows:

(A0) $x_0' = x$,
(A1) $x_i(j) = 0, \forall j \in P_i$,
(A2) $\forall 1 \leq i \leq s$, $\mathcal{L}(\langle x_{i-1}', W_i \rangle, x_i)$ — a system of linear inequalities stated in Lemma 3,
(A3) $x_i + v_i \geq 0$ and $x_i' = x_i + v_i$,
(A4) $x_s' = z$.

In the above inequalities, $x_0', x_1, x_1', \ldots x_s, x_s'$ are vector variables representing the values of markings x, z_1, z_1', … z_s, z_s', respectively, mentioned in our earlier discussion. (A0) is trivial. What (A1) says is that at x_i, no vectors with priorities higher than v_i are applicable. By Lemma 3, (A2) is sufficient to imply $x_{i-1}' \xrightarrow{*} x_i$. Lemma 2, in conjunction with (A1) and (A2), further implies $x_{i-1}' \xrightarrow{*}_\rho x_i$. (A3) and (A4) are again trivial.

Based on our earlier discussion, it is then straightforward that the above system of linear inequalities has an integer solution iff $z \in R_\rho(G)$. Hence, the reachability problem is in NP. ∎