ON STATELESS AUTOMATA AND P SYSTEMS*

Linmin Yang
School of Electrical Engineering and Computer Science
Washington State University, Pullman, Washington 99164, USA
lyang1@eecs.wsu.edu

Zhe Dang
School of Electrical Engineering and Computer Science
Washington State University, Pullman, Washington 99164, USA
zdang@eecs.wsu.edu

Oscar H. Ibarra†
Department of Computer Science
University of California, Santa Barbara, California 93106, USA
ibarra@cs.ucsb.edu

We introduce the notion of stateless multihead two-way (respectively, one-way) NFAs and stateless multicounter systems and relate them to P systems and vector addition systems. In particular, we investigate the decidability of the emptiness and reachability problems for these stateless automata and show that the results are applicable to similar questions concerning certain variants of P systems, namely, token systems and sequential tissue-like P systems.

Keywords: P system; stateless automaton; vector addition system; reachability problem.

1. Introduction

There has been a flurry of research activities in the area of membrane computing (a branch of molecular computing) initiated seven years ago by Gheorghe Paun [8]. Membrane computing identifies an unconventional computing model, namely a P system, from natural phenomena of cell evolutions and chemical reactions. It abstracts from the way living cells process chemical compounds in their compartmental structures. 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

*The work of Zhe Dang and Linmin Yang was supported in part by NSF Grant CCF-0430531. The work of Oscar H. Ibarra was supported in part by NSF Grants CCF-0430945 and CCF-0524136, and a Nokia Visiting Fellowhip at the University of Turku, Finland.
†Corresponding author.

diagram or a tree structure where one membrane may contain other membranes. By using the rules in a nondeterministic and maximally parallel manner, transitions between the system configurations can be obtained. A sequence of transitions shows how the system is evolving. At a high-level, a P system has the following key features:

- Objects are typed but addressless (i.e., without individual identifiers),
- Objects can transfer between membranes,
- Membranes themselves form a structure (such as a tree),
- Object transferring rules are in (either maximally or locally) parallel, and
- **The system is stateless**.

Biologically inspired computing models like P systems [9] are often stateless. This is because it is difficult and even unrealistic to maintain a global state for a massively parallel group of objects. Naturally, a membrane in a P system, which is a multiset of objects drawn from a given finite type set $\{a_1, \ldots, a_k\}$, can be modeled as having counters $x_1, \ldots, x_k$ to represent the multiplicities of objects of types $a_1, \ldots, a_k$, respectively. In this way, a P system can be characterized as a counter machine in a nontraditional form; e.g., without states, and with parallel counter increments/decrements, etc. The most common form of stateless counter machines are probably the Vector Addition Systems (VASs), which have been well-studied. Indeed, VASs have been shown to be intimately related to certain classes of P systems [5]. However, with new applications of P systems in mind [10], other classes of stateless automata deserve further investigation. In this paper, we present some results in this direction, with applications to reachability problems for variants of P systems, namely, token systems and sequential tissue-like P systems.

2. Preliminaries

A nondeterministic multicounter automaton is a nondeterministic automaton with a finite number of states, a one-way input tape, and a finite number of integer counters. Each counter can be incremented by 1, decremented by 1, or stay unchanged. Besides, a counter can be tested against 0. It is well-known that counter machines with two counters have an undecidable halting problem. Thus, to study decidable cases, we have to restrict the behaviors of the counters. One such restriction is to limit the number of reversals a counter can make. A counter is *n-reversal-bounded* if it changes mode between nondecreasing and nonincreasing at most n times. For instance, the following sequence of counter values:

$$0, 0, 1, 1, 2, 2, 3, 3, 4, 4, 3, 2, 1, 1, 1, 1, \ldots$$

demonstrates only one counter reversal. A counter is *reversal-bounded* if it is n-reversal-bounded for some fixed number n independent of computations. A *reversal-bounded nondeterministic multicounter automaton* is a nondeterministic multicounter automaton in which each counter is reversal-bounded.

Let Y be a finite set of variables over integers. For all integers a_y, with $y \in Y$, b and c (with $b > 0$), $\sum_{y \in Y} a_y y < c$ is an *atomic linear relation* on Y and $\sum_{y \in Y} a_y y \equiv_b c$ is a *linear congruence* on Y. A *linear relation* on Y is a Boolean combination (using $\neg$ and $\wedge$) of atomic linear relations on Y. A *Presburger formula* [2] on Y is a Boolean combination of atomic linear relations on Y and linear congruences on Y. A set P of tuples of nonnegative integers is *Presburger-definable* or a *Presburger relation* if there exists a Presburger formula $\mathcal{F}$ on Y such that P is exactly the set of the solutions for Y that make $\mathcal{F}$ true. It is well known that Presburger formulas are closed under quantification.

Let $\mathbb{N}$ be the set of nonnegative integers and n be a positive integer. A subset S of $\mathbb{N}^n$ is a *linear set* if there exist vectors $v_0, v_1, \ldots, v_t$ in $\mathbb{N}^n$ such that $S = \{v \mid v = v_0 + a_1 v_1 + \ldots + a_t v_t, \forall 1 \leq i \leq t, a_i \in \mathbb{N}\}$. S is a *semilinear set* if it is a finite union of linear sets. It is known that S is a semilinear set if and only if S is Presburger-definable [2].

Let Σ be an alphabet consisting of n symbols $a_1, \ldots, a_n$. For each string (word) w in Σ^*, we define the *Parikh map* of w, denoted by $p(w)$, as follows:

$$p(w) = (i_1, \ldots, i_n), \quad \text{where } i_j \text{ is the number of occurrences of } a_j \text{ in } w.$$

If L is a subset of Σ^*, the *Parikh map* of L is defined by $p(L) = \{p(w) \mid w \in L\}$. L is a *semilinear language* if its Parikh map $p(L)$ is a semilinear set.

The following result is known [4]:

Theorem 1. *$p(L(M))$ is an effectively computable semilinear set when M is a reversal-bounded nondeterministic multicounter automaton.*

Consider a reversal-bounded nondeterministic multicounter machine M (which is a reversal-bounded nondeterministic multicounter automaton without input). Let $(j, v_1, \ldots, v_k)$ denote the configuration of M when it is in state j and counter i has value v_i for $1 \leq i \leq k$. Define $R(M) = \{(\alpha, \beta) \mid$ configuration α can reach configuration β in 0 or more moves $\}$, which is called the reachability relation of M. Using Theorem 1, one can easily show the following result:

Theorem 2. *The reachability relation of a reversal-bounded nondeterministic multicounter machine is Presburger definable.*

An n-dimensional *vector addition system* (VAS) is specified by W, a finite set of vectors in $\mathbb{Z}^n$, where $\mathbb{Z}$ is the set of all integers (positive, negative, zero). For two vectors x and z in $\mathbb{N}^n$, we say that x can *reach* z if for some $j, z = x + v_1 + \ldots + v_j$, where, for all $1 \leq i \leq j$, each $v_i \in W$ and $x + v_1 + \ldots + v_i \geq 0$. The Presburger reachability problem for VAS is to decide, given two Presburger formulas P and Q, whether there are x satisfying P and z satisfying Q such that x can reach z. The following theorem follows from the decidability of the reachability problem for VASs (which are equivalent to Petri nets) [7].

Theorem 3. *The Presburger reachability problem for VAS is decidable.*

3. Stateless Multihead Two-way (One-way) NFAs/DFAs and Token Systems

Let Σ and Π be two alphabets. An object of some type in Σ (resp., Π) is called a *standard object* (resp., a *token*). Consider a chain (with length n) of membranes (i.e., membranes are organized as a linear structure)

$$A_1, \ldots, A_n \tag{1}$$

for some n. The chain is called *initial* if the following conditions are met:

1 Each A_i holds exactly one standard object,
2 A_1 contains one token of each type in Π; the rest of the A_i's do not contain any tokens,
3 The standard object in the first membrane A_1 is of type $\triangleright \in \Sigma$ and the standard object in the last membrane A_n is of type $\triangleleft \in \Sigma$; the membranes in between A_1 and A_n do not contain any $\triangleright$-objects and $\triangleleft$-objects.

Let $\Pi' \subseteq \Pi$ be given. The chain is called *halting* if we change the condition 2 above into "A_n contains one token of each type in Π'." A *rule* specifies how objects are transferred between two neighboring membranes (i.e., A_i and A_{i+1} for $1 \leq i \leq n-1$) and is in one of the following two forms:

- $(a, p)^{\rightarrow}$,
- $(a, p)^{\leftarrow}$,

where $a \in \Sigma$ and $p \in \Pi$. For instance, when $(a, p)^{\rightarrow}$ is applied on A_i, the A_i must contain a standard a-object and a p-token. The result is to move the token from A_i to A_{i+1} (where $1 \leq i \leq n-1$). The semantics of $(a, p)^{\leftarrow}$ is defined similarly but the token p moves from A_{i+1} to A_i. We are given a set of rules R which are applied sequentially; i.e., each time, a rule and an i are nondeterministically picked and the rule is applied on A_i. We are interested in studying the computing power of such *token* systems. Specifically, we focus on decision algorithms solving the following reachability problem: whether there is an $n \geq 1$ and an initial chain with length n such that, after a certain number of applications of rules in R, the initial chain evolves into a halting chain. Notice that an instance of a chain in the form of (1) is a special instance of tissue-like P systems [6] and in the future we will study intra-member structures more general than linear structures in (1), for example, graphs. Also note that, in the reachability problem, the chain is not given. Instead, we are looking for a desired chain. This is different from the case for a tissue-like P system where a concrete instance (with the n in (1) given) is part of the system definition.

One can generalize the aforementioned token systems by allowing some of the rules to be synchronized, where a *synchronized rule* is a rule of the following form

$$[r_1, r_2, \ldots, r_k]$$

for some k and distinct rules $r_1, \ldots, r_k$. The semantics of the synchronized rule is to apply each r_i at the same time. For instance, a synchronized rule $[(a,p)^{\rightarrow}, (b,q)^{\leftarrow}]$, when applied, is to nondeterministically pick an A_i and an A_j and apply the rule $(a,p)^{\rightarrow}$ on A_i and the rule $(b,q)^{\leftarrow}$ on A_j (assuming both are applicable). Such systems with synchronized rules are called *generalized* token systems and we can ask the same reachability problem for generalized token systems.

We first observe that the (generalized) token systems are essentially the same as *stateless multihead two-way NFAs* M studied in the following, where each input tape cell corresponds to a membrane in (1) and each token corresponds to a two-way head. The stateless NFA M is equipped with an input on alphabet Σ and heads $H_1, \ldots, H_k$ for some k. The heads are two-way, the input is read-only, and there are no states. An H_i-move (also called a *local move*) MOVE_i of the NFA can be described as a triple (H_i, a, D), where H_i is the head involved in the move, a is the input symbol under the head H_i, and $D \in \{+1, -1, 0\}$ meaning that, as a result of the move, the head H_i goes to the right, goes to the left, or simply stays. When a head H_i tries to execute a local move (H_i, a, D), the symbol under H_i must be a; otherwise M just crashes. A generalized move is in the form of $(H_i, \mathcal{S}, \mathcal{D})$, where $\mathcal{S}$ is a set of symbols, and $\mathcal{D}$ is a set of directions (i.e., $+1$, -1, 0). When executing a generalized move $(H_i, \mathcal{S}, \mathcal{D})$, the symbol H_i reads must belong to $\mathcal{S}$, and H_i nondeterministically picks a direction from $\mathcal{D}$.

Note that a local move is a special case of a generalized move. An *instruction* of M is a sequence of local or general moves, in the form of $[\text{MOVE}_{i_1}, \text{MOVE}_{i_2}, \ldots, \text{MOVE}_{i_m}]$, for some m, $1 \leq m \leq k$, and $i_1 < \ldots < i_m$. (If $m = 1$, the instruction is simply called a *local* instruction.) When the instruction is executed, the heads $H_{i_1}, \ldots, H_{i_m}$ perform the moves $\text{MOVE}_{i_1}, \ldots, \text{MOVE}_{i_m}$, respectively and simultaneously. Any head falling off the tape will cause M to crash. The NFA M has a finite set of such instructions. At each step, it nondeterministically picks a maximally parallel set of instructions to execute. M has a set of accepting heads $F \subseteq \{H_1, ..., H_k\}$. For most constructions in this paper, F consists of all the heads. Initially, all heads are at the leftmost cell of the input tape. *M halts and accepts* the input when the accepting heads are all at the rightmost cell. We assume that the input tape of M has a left end marker $\rhd$ and a right end marker $\lhd$. Thus, for any input $a_1...a_n$, $n \geq 2$, $a_1 = \rhd$, $a_n = \lhd$, and for $2 \leq i \leq n-1$, each a_i is different from the end markers.

We emphasize that in a stateless multihead one-way (two-way) NFA, at each step, the application of the instructions is "maximally parallel", i.e., all instructions that can be applied to the heads must be applied. Note that, in general, the set of instructions that can be applied maximally parallel need not be unique. If at most m instructions are applicable at each step, then we say the machine is m-maxpar.

A stateless one-way (two-way) DFA is one in which at each step of the computation, at most one maximally parallel set of instructions is applicable.

Our first result is the following:

Theorem 4. *The reachability problem for token systems is decidable. The problem is undecidable for generalized token systems.*

The first part follows from the fact that the emptiness problem for two-way NFAs is decidable. The second part of Theorem 4 is a direct consequence of the next theorem.

Theorem 5. *The emptiness problem for stateless (1-maxpar) 3-head one-way DFAs is undecidable.*

Proof. Given a deterministic TM Z, let $A_Z = C_1\#C_2\#...\#C_n$ be the halting computation of Z starting on blank tape. Hence C_1 is the initial configuration (on blank tape), C_n is the halting configuration, and C_{i+1} is the direct successor of C_i. We assume that $n \geq 2$. Let Γ and Q_Z be the tape alphabet and state set, respectively, of Z.

Clearly, from Z, we can construct a 2-head one-way DFA M_0 (with states) with heads H_1 and H_2 and input alphabet $\Sigma = \Gamma \cup Q_Z \cup \{\#\}$, which accepts a nonempty language (actually only the string A_Z) iff Z halts. Because from configuration C_i the next step of Z that results in configuration C_{i+1} may move its read-write head left, H_2 may not always move to the right at every step in M_0's computation. However, we can modify M_0 into another two-head one-way DFA M by putting "dummy" symbols α's on the tape so that H_2 can read these symbols instead of not moving right. H_1, of course, ignores these dummy symbols. M has now the property that H_2 always moves to the right at every step in the computation until M accepts. Clearly, $L(M_0) = \emptyset$ if and only if $L(M) = \emptyset$, and if and only if Z does not halt on blank tape. We may assume that M accepts with H_2 falling off the right end of the tape in a unique accepting state f. (This assumption on H_2 falling off the right end of the tape should not be confused with the condition that in a stateless automaton, a head falling off the tape will cause the machine to crash.) We also assume that there are no transitions from state f. Let Q_M be the state set of M.

Thus, if $\delta(q, a_1, a_2) = (p, d_1, d_2)$, then $d_2 = +1$. This transition is applicable if M is currently in the state q and the heads H_1 and H_2 are reading a_1 and a_2, respectively. When the transition is applied, H_2 moves right, H_1 moves right or remains stationary depending on whether d_1 is +1 or 0, and M enters state p.

We construct a stateless 3-head one-way DFA M' to simulate M. The heads of M' are H_1, H_2, and H_3. The input alphabet of M' is $(\Sigma \times Q_M \cup \{\rhd, \lhd\}\ )$ ($\rhd$ and $\lhd$ are left and right end markers for the input to M'.) The instructions of M' are as follows:

(1) $[(H_1, \rhd, 0), (H_2, \rhd, 0), (H_3, \rhd, +1)]$.
(2) $[(H_1, \rhd, +1), (H_2, \rhd, +1), (H_3, (a, q), +1)]$ for every $a \in \Sigma$ and every $q \in Q_M$.
(3) Suppose $\delta(q, a_1, a_2) = (p, d_1, d_2)$ and $p \neq f$, then $[(H_1, (a_1, s), d_1), (H_2, (a_2, q), +1), (H_3, (b, p), +1)]$ is a rule for every $s \in Q_M$ and every $b \in \Sigma$.

(4) Suppose $\delta(q, a_1, a_2) = (f, d_1, +1)$, then
$[(H_1, (a_1, s), d_1), (H_2, (a_2, q), +1), (H_3, \lhd, 0)]$ is a rule for every $s \in Q_M$.
(5) $[(H_1, (a, q), +1), (H_2, (b, p), +1), (H_3, \lhd, 0)]$ is a rule for every $a, b \in \Sigma$ and $q, p \in Q_M$.
(6) $[(H_1, (a, q), +1), (H_2, \lhd, 0), (H_3, \lhd, 0)]$ is a rule for every $a \in \Sigma$ and $q \in Q_M$.

M' accepts if and only if all three heads are on $\lhd$. Intuitively, each symbol read by the heads in M' is a pair of an input symbol and a state in M. M' is to simulate the execution of M. During the simulation, the symbol in a pair that the head H_1 in M' reads corresponds to the input symbol under the head H_1 in M. The head H_2 in M' corresponds to the head H_2 in M. In particular, the state in a pair that the head H_2 in M' reads corresponds to the current state of M. Finally, the head H_3 in M' reads the next step of M. From the construction, it is clear that M' is a stateless 3-head one-way DFA, and $L(M) = \emptyset$ if and only if Z does not halt on blank tape. The result follows from the undecidability of the halting problem for TMs on blank tape. □

It is an interesting open question whether the 3 heads in the above theorem can be reduced to 2, even if the 2 heads are allowed to be two-way. Note that in the theorem, all 3 heads are involved in every instruction. We can strengthen this result by a more intricate construction. Define a stateless k-head one-way 2-move NFA (DFA) to be one where in every instruction, at most two heads are involved. Then we have:

Theorem 6. *The emptiness problem for stateless 3-head one-way 2-move DFAs is undecidable.*

Proof. Let M be the 2-head one-way DFA **with states** constructed in the previous proof. The transition $\delta(q, a, b) = (p, d_1, +1)$ of M can be represented by the tuple

$$[q, (H_1, a, d_1), (H_2, b, +1), p]$$

Suppose M has n such transitions, and we number them as $1, \ldots, n$. We may assume that M starts its computation with rule number 1.

Note that H_2 moves to the right at every step, and that M accepts with H_2 falling off the right end of the tape in a unique accepting state f and there are no transitions from state f.

We say that transition numbers i and j are compatible if they correspond to transition instructions $[q, (H_1, a, d_1), (H_2, b, +1), p]$ and $[p, (H_1, a', d_1'), (H_2, b', +1), r]$, respectively, for some states q, p, r, symbols a, a', b, b', and directions d_1, d_1'.

The input alphabet of M' is $(\Sigma \times N \times \Delta) \cup \{\rhd, \lhd\}$, where $N = \{1, \ldots, n\}$ (set of transition numbers of M) and $\Delta = \{\delta_1, \delta_2\}$ ($\rhd$ and $\lhd$ are the end markers of M'). The heads of M' are H_1, H_2, H_3. The instructions of M' are defined as follows:

(1) $[(H_1, \rhd, +1)]$
(2) $[(H_3, \rhd, 0), (H_2, \rhd, +1)]$

(3) $[(H_3, \triangleright, +1), (H_2, (c, 1, \delta_1), 0)]$ for every $c \in \Sigma$.

Suppose transition number k corresponds to $[q, (H_1, a, d_1), (H_2, b, +1), p]$. Then the following instructions are in M':

4. $[(H_3, (b, k, \delta_1), 0), (H_2, (b, k, \delta_1), +1)]$
5. $[(H_3, (c, i, \delta_1), +1), (H_2, (d, i, \delta_2), 0)]$, for every $1 \le i \le n$ and every $c, d \in \Sigma$.
6. $[(H_1, (a, i, \delta), d_1), (H_2, (c, k, \delta_2), 0)]$, for every $1 \le i \le n$, every $c \in \Sigma$, and every $\delta \in \Delta$.
7. $[(H_3, (c, i, \delta_2), 0), (H_2, (c, i, \delta_2), +1)]$, for every $1 \le i \le n$ and every $c \in \Sigma$.
8. $[(H_3, (c, i, \delta_2), +1), (H_2, (d, j, \delta_1), 0)]$, for every $1 \le i, j \le n$ with i and j compatible, and every $c, d \in \Sigma$.
9. $[(H_3, (c, i, \delta_2), +1), (H_2, \triangleleft, 0)]$, every $c \in \Sigma$ and for every $1 \le i \le n$, with i corresponding to a transition of the form $[q, (H_1, a', d_1), (H_2, b', +1), f]$ for every state q and a', b' in Σ. (Note that f is the unique accepting state of M.)
10. $[(H_1, (c, i, \delta), +1), (H_3, \triangleleft, 0)]$, for every $1 \le i \le n$, every $c \in \Sigma$, and every $\delta \in \Delta$.

Define a homomorphism h that maps each symbol (α, i, δ) to (i, δ). Then we require that the homomorphic image of the input tape of M' (excluding the end markers) is a string in

$$(1, \delta_1)(1, \delta_2)\{(1, \delta_1)(1, \delta_2), ..., (n, \delta_1)(n, \delta_2)\}^*$$

so that a sequence of valid transitions can be executed properly. More precisely, each symbol read by the heads in M' is a triple of an input symbol (in M), transition number (in M), and either a δ_1 or δ_2. As shown above, a triple with δ_1 and a triple with δ_2 appear alternately and the head H_3 which is ahead of the head H_2, is used to check the format and to ensure the compatibility of two successive transitions. Finally, the head H_1 in M' functions similarly as in the previous proof.

M' accepts if and only if all heads are on the right end marker. ¿From the construction, it is clear that $L(M') = \emptyset$ iff $L(M) = \emptyset$. The undecidability follows.□

Next, we will study the emptiness problem when the inputs are restricted. Recall that a language is bounded if it is a subset of $a_1^* a_2^* \dots a_k^*$ for some given symbols $a_1, \dots, a_k$.

It is known [3] that if M is a multihead one-way NFA with states but with bounded input, the language it accepts is a semilinear set effectively constructable from M. In fact, this result holds, even if M has two-way heads, but the heads can only reverse directions from right to left or from left to right at most r times, for some fixed r independent of the input. It follows that Theorem 5 can not be strengthened to hold for one-way machines accepting bounded languages. However, for two-way machines, we can prove the following:

Theorem 7. *The emptiness problem for stateless 5-head two-way NFAs over bounded input is undecidable.*

Proof. We show how a stateless 5-head two-way NFA M' can simulate a 2-counter machine M. Suppose M has states $q_1, ..., q_n$, where we assume that $n \geq 3$, q_1 is the initial state, and q_n is the unique halting state. Assume that both counters are zero upon halting, and the number of steps is odd. The transition of M is of the form $\delta(q_i, s_1, s_2) = (q_j, d_1, d_2)$ where s_r (sign of counter r) = 0 or + and d_r (change in counter r) $= +1, 0, -1$ for $r = 1, 2$.

A valid input to M' is a string of the form $\rhd q_1 q_2 ... q_n a^d \lhd$ for some $d \geq 1$. We construct a stateless 5-head two-way NFA M' (with heads H_1, H_2, H_3, C_1, C_2) to simulate M. We begin with the following instructions:

$[(H_1, \rhd, 0), (H_2, \rhd, +1)]$
$[(H_1, \rhd, +1), (H_2, q_1, +1)]$
$[(H_1, q_1, +1), (H_2, q_2, +1)]$
...
...
$[(H_1, q_{n-1}, +1), (H_2, q_n, +1)]$
$[(H_1, q_n, +1), (H_2, a, +1)]$
$[(H_1, a, +1), (H_2, a, +1)]$
$[(H_1, a, +1), (H_2, \lhd, 0)]$

The instructions above check that the input is of the form $\rhd q_1 q_2 ... q_n a^d \lhd$ for some $d \geq 1$. At the end of the process H_1 and H_2 are on the right end marker $\lhd$. Next add the following instructions:

$[(H_1, \lhd, -1), (H_2, \lhd, -1), (H_3, \rhd, +1)]$
$[(H_1, t, -1), (H_2, t, -1), (H_3, q_1, 0)]$ for all $t \neq q_1$
$[(H_1, q_1, 0), (H_2, q_1, 0), (H_3, q_1, +1)]$
$[(H_2, q_k, +1), (H_3, q_2, 0)]$ for $1 \leq k \leq n-1$
$[(H_2, q_k, -1), (H_3, q_2, 0)]$ for $2 \leq k \leq n$
$[(H_1, q_k, +1), (H_3, q_3, 0)]$ for $1 \leq k \leq n-1$
$[(H_1, q_k, -1), (H_3, q_3, 0)]$ for $2 \leq k \leq n$

In the instructions above, if the symbol under H_3 is q_2 (resp., q_3) , the machine positions H_2 (resp., H_1) to some nondeterministically chosen state (for use below).

Let C_1 and C_2 be the counters of M. Initially they are set to zero. In the instructions below, we use heads C_1 and C_2 to correspond to the counters.

If $\delta(q_i, s_1, s_2) = (q_j, d_1, d_2)$ where $s_r = 0$ or + and $d_r = +1, 0, -1$, then add the following two instructions:

$[(H_1, q_i, 0), (H_2, q_j, 0), (H_3, q_2, +1), (C_1, t_1, d_1), (C_2, t_2, d_2)]$ and
$[(H_1, q_j, 0), (H_2, q_i, 0), (H_3, q_3, -1), (C_1, t_1, d_1), (C_2, t_2, d_2)]$
where $t_r = \rhd$ if $s_r = 0$ and $t_r \neq \rhd$ if $s_r = +$.

If $\delta(q_i, 0, 0) = (q_n, 0, 0)$ (i.e., the 2-counter machine M halts in the unique state q_n with both counters zero after an odd number of steps), then add the following instructions:

$[(H_1, q_i, 0), (H_2, q_n, 0), (C_1, \rhd, +1), (C_2, \rhd, +1)]$
$[(H_1, q_i, 0), (H_2, q_n, 0), (C_1, t, +1), (C_2, t, +1)]$ for all $t \neq \lhd$, and
$[(H_r, t, +1), (C_1, \lhd, 0), (C_2, \lhd, 0)]$ for all $t \neq \lhd$ and for $r = 1, 2, 3$.

M' accepts if all heads $(H_1, H_2, H_3, C_1, C_2)$ are on the right end marker. It can be easily verified that M' accepts the empty set if and only if M does not halt. □

The above theorem says that the emptiness problem is undecidable if the number of heads is 5 (i.e., fixed) but the size of the input alphabet is arbitrary. The next result shows that the emptiness problem is also undecidable if the size of input alphabet is fixed (in fact, can be unary) but the number of heads is arbitrary.

Theorem 8. *The emptiness problem for stateless multihead two-way NFAs is undecidable even when the input is unary but with the left end marker (resp., right end marker).*

Proof. We only show the case with the left end marker. (The construction can easily be modified for the case with the right end marker.) We assume that the input has length at least 1, excluding the left end marker. We use a stateless NFA M' (whose input is unary with a left end marker) to simulate a 2-counter machine M with counters C_1 and C_2, and states $q_0, q_1, \ldots, q_n$, $n \geq 1$. We assume that M starts in state q_0 with $C_1 = C_2 = 0$ and if it halts, it halts in state q_1 with $C_1 = C_2 = 0$.

The idea is to use $\lceil \log_2(n+1) \rceil$ heads, $H_1, \ldots, H_{\lceil \log_2(n+1) \rceil}$, to control the states, and another two heads $H_{\lceil \log_2(n+1) \rceil + 1}$ and $H_{\lceil \log_2(n+1) \rceil + 2}$ to control the value of counters C_1 and C_2. Initially, all heads of M' are at the leftmost cell (i.e., $\rhd$). If H_i, $1 \leq i \leq \lceil \log_2(n+1) \rceil$, is at $\rhd$, we consider it as 0; if H_i is at the first a, we consider it as 1. Hence $H_{\lceil \log_2(n+1) \rceil} \ldots H_1$ is a binary string (with $H_{\lceil \log_2(n+1) \rceil}$ as its most significant bit), which is used to encode the index of a state of M. Note that during the computation, H_i, $1 \leq i \leq \lceil \log_2(n+1) \rceil$, only moves between the left end marker and the first a. We define acceptance of M' to be when H_1 is on a, and all other heads are on the left end marker (note that this definition of acceptance is different from the original definition). This corresponds to the configuration when M halts in state q_1 with $C_1 = C_2 = 0$. We omit the details. □

Theorems 7 and 8 are best possible, since we can not fix both the number k of heads and the size n of the nput alphabet and get undecidability. This is because for fixed k and n, there are only a finite number of such stateless machines (also observed by Artiom Alhazov). Hence, the emptiness problem has a finite number of instances and therefore decidable.

A special case of Theorem 8 is when the input is unary and without end markers. In this case, the heads are initially at the leftmost input cell and the automaton accepts when the heads are all at the rightmost cell (we assume that there are at least two cells on the input). Using a VAS to simulate the multihead position changes,we have:

Theorem 9. *The emptiness problem for stateless multihead two-way NFAs is decidable when the input is unary and without end markers.*

Proof. Suppose we have a stateless NFA M with $H_1, \ldots, H_k$ for some k, and with unary input

$$a \ldots a$$

of length B for some B. We can construct a corresponding VAS $G = \langle x, W \rangle$, where $x \in \mathbb{N}^k$ is the start vector, and W is a finite set of vectors in $\mathbb{Z}^k$. Furthermore, we require that the maximal entry of any vector in G can not exceed $B - 1$; otherwise, G crashes. In other words, G is a bounded vector addition system. Since initially in M, all heads are at the leftmost cell, we set $x = (0, \ldots, 0)$ in G. If in M, there is an instruction $I =$[MOVE$_{i_1}$, MOVE$_{i_2}$, ..., MOVE$_{i_m}$], for some m, $1 \le m \le k$, and MOVE$_{i_j}$=(H_{i_j}, a, D), $1 \le j \le m$, then in G we have the following $v \in W$. For all j, $1 \le j \le m$, the i_j^{th} entry of v is 0 if $D = 0$. The entry is 1 if $D = +1$. And, the entry is -1 if $D = -1$. All other entries are 0.

Clearly, M accepts a nonempty language iff, for some B, $(B - 1, \ldots, B - 1)$ is reachable in G; directly from Theorem 3, the result follows. □

4. Sequential Tissue-Like P Systems and Stateless Multicounter Systems

We now generalize the rules in a token system by allowing multiple objects to transfer from one membrane to another in (1); the result is a variant of a tissue-like P system [6] with sequential applications of rules [1]. More precisely, let $\Sigma = \{a_1, \ldots, a_m\}$. A *sequential tissue-like P system* G is a directed graph with n nodes (for some n), where each node i is equipped with a membrane A_i which is a multiset of objects in Σ. In particular, we use m counters $\mathcal{X}_i = (x_{i1}, \ldots, x_{im})$ to denote the multiplicities of objects of types $a_1, \ldots, a_m$ in A_i, respectively. Each membrane A_i is also associated with a Presburger formula P_i, called a *node constraint*, over the m counters. Each edge (say, from node i to node j) in G is labeled with an *addition vector* Δ_{ij} in $\mathbb{N}^m$. Essentially, G defines a stateless multicounter system whose semantics is as follows. Intuitively, G specifies a multicounter system with n groups of counters with each group $\mathcal{X}_i$ of m counters. In the system, there are no states. Each instruction is the addition vector Δ_{ij} specified on an edge. The semantics of the instruction, when applied, is to decrement counters in group $\mathcal{X}_i$ by Δ_{ij} and increment counters in group $\mathcal{X}_j$ by Δ_{ij} (we emphasize the fact that each component in the vector Δ_{ij} is nonnegative by definition). When the system runs, an instruction is nondeterministically chosen and applied. Additionally, we require that at any moment during a run, for each i, the constraint P_i is true when evaluated on the counter values on group $\mathcal{X}_i$. Formally, a configuration is a tuple of n vectors $(V_1, \ldots, V_n)$ with each $V_i \in \mathbb{N}^m$ satisfying the node constraint $P_i(V_i)$. Given two configurations $\mathcal{C} = (V_1, \ldots, V_n)$ and $\mathcal{C}' = (V_1', \ldots, V_n')$, we say that $\mathcal{C}$ can reach $\mathcal{C}'$ in a *move*, written $\mathcal{C} \ \rightarrow_G \ \mathcal{C}'$, if there is an edge from node i to node j (for some i and j) such that $\mathcal{C}$ and $\mathcal{C}'$ are exactly the same except that $V_i - \Delta_{ij} = V_i'$ and $V_j + \Delta_{ij} = V_j'$. We say that $\mathcal{C}$ can *reach* $\mathcal{C}'$ in G, written

$$\mathcal{C} \ \leadsto_G \ \mathcal{C}'$$

if, for some t,

$$\mathcal{C}_0 \rightarrow_G \mathcal{C}_1 \ \ldots \ \rightarrow_G \mathcal{C}_t$$

where $\mathcal{C} = \mathcal{C}_0, \mathcal{C}_1, \ldots, \mathcal{C}_t = \mathcal{C}'$ are configurations. We now study the following *reachability problem*: given a G and two Presburger formulas $\mathcal{P}$ and $\mathcal{Q}$, whether there are $\mathcal{C}$ and $\mathcal{C}'$ such that $\mathcal{C} \ \leadsto_G \ \mathcal{C}'$ and, $\mathcal{C}$ and $\mathcal{C}'$ satisfy $\mathcal{P}$ and $\mathcal{Q}$, respectively.

One can show that the reachability problem is undecidable even under a special case:

Theorem 10. *The reachability problem for sequential tissue-like P systems G is undecidable even when G is a DAG.*

Proof. The proof idea is to use G to simulate a 2-counter machine M. M has two counters X_1 and X_2 (initially both are 0), and n states $s_1, \ldots, s_n$, where s_1 is the initial state and s_n is the accepting state. We can assume that M is essentially deterministic. That is, for any pair of states (s_i, s_j), there is at most one instruction that leads from s_i to s_j.

The graph G is illustrated in Figure 1. Suppose there are t instructions, $I_1, \ldots, I_t$, in M. In G, we totally have $(2t+1)$ nodes: the node $\hat{N}$, t suppliers and t depositories, each of which is equipped with $2n+3$ counters. Suppose that the counters in node $\hat{N}$ are $(x_{11}, \ldots, x_{1n}, y_{11}, \ldots, y_{1n}, z_1, z_2, c)$, where $x_{11}, \ldots, x_{1n}, y_{11}, \ldots, y_{1n}$ are used to simulate the states of M, z_1 and z_2 represent the values of counter X_1 and X_2, respectively, and c acts as a *control* counter to control the order of vectors to be fired. Supplier k $(1 \le k \le t)$ only has an outgoing edge to $\hat{N}$ decorated by Δ_k^2, which is to supply counters in $\hat{N}$ with objects specified by Δ_k^2. Similarly, depository k $(1 \le k \le t)$ only has an incoming edge from $\hat{N}$ decorated by Δ_k^1, and it only receives objects from $\hat{N}$ specified by Δ_k^1. Therefore suppliers and depositories will not communicate between themselves (So, each node either connects to or connects from (but not both) the node $\hat{N}$, and hence G is a DAG). If for some m, M is in state s_m, then in the node $\hat{N}$ there is one and only one counter x_{1m} among x_{1j}'s, $1 \le j \le n$, whose value is 1 and all the other x_{1j} counters are 0. Initially, $\hat{N}$ starts from $(x_{11}, \ldots, x_{1n}; y_{11}, \ldots, y_{1n}; z_1, z_2; c) = (1, 0 \ldots, 0; 1, 1, \ldots, 1; 0, 0; 1)$. We

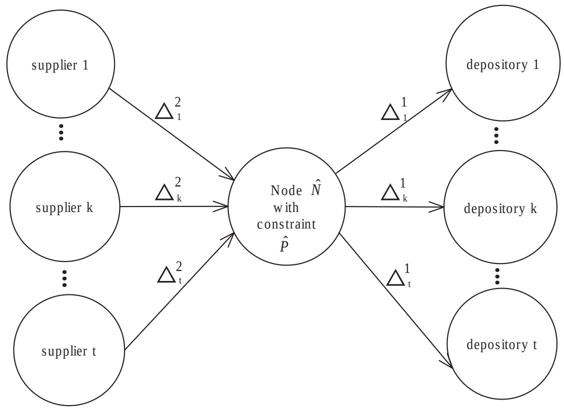

Fig. 1. The structure of the DAG G in the proof of Theorem 10. G simulates a deterministic 2-counter machine M. The instructions of M are $I_1, \ldots, I_k, \ldots, I_t$. Control counter c in $\hat{N}$ guarantees that Δ_i^1 and Δ_i^2 are executed alternately. When an instruction I_k is fired in M, our construction of G ensures that Δ_k^1 is the unique vector that can be applied, and the vector that can only be applied next is Δ_k^2. The executions of Δ_k^1 and Δ_k^2 in G together simulate the execution of the instruction I_k in M.

first define a Presburger formula, called P, as part of the constraint that we are going to apply over $\hat{N}$. The formula P is as follows:

$$\sum_{h=1}^{n} x_{1h} \leq 1 \wedge y_{11} \leq 1 \wedge \ldots \wedge y_{1n} \leq 1 \wedge (\sum_{h=1}^{n} y_{1h} = n \vee \sum_{h=1}^{n} y_{1h} = n-2) \wedge c \leq 1. \quad (2)$$

Consider an instruction I_k in M. If the instruction I_k is $I_k = (s_i : X_1 ++, \texttt{goto } s_j)$, in G we have an additional Presburger formula P_k that is simply *true*, besides, we have a Δ_k^1 decorating an outgoing edge of $\hat{N}$ (to depository k), and, in the vector Δ_k^1, the i^{th}, $(n+i)^{th}$, $(n+j)^{th}$ and $(2n+3)^{th}$ entries (corresponding to x_{1i}, y_{1i}, y_{1j} and c) are 1, and all other entries are 0; we also have another Δ_k^2 decorating an incoming edge of $\hat{N}$ (from supplier k), and, in the vector Δ_k^2, the j^{th}, $(n+i)^{th}$, $(n+j)^{th}, (2n+1)^{th}$ and $(2n+3)^{th}$ entries (corresponding to x_{1j}, y_{1i}, y_{1j}, z_1 and c) are 1, and all other entries are 0. If the instruction I_k is $I_k = (s_i : X_1 --, \texttt{goto } s_j)$, it can be handled in a similar way (i.e., P_k is *true*), except that the $(2n+1)^{th}$ entry of Δ_k^1 now is 1, and the $(2n+1)^{th}$ entry of Δ_k^2 now is 0. If the instruction I_k is $I_k = (s_i : X_1 == 0?, \texttt{goto } s_j)$, the P_k now is defined as the following formula:

$$((c = 0 \wedge y_{1i} = 0 \wedge y_{1j} = 0) \Rightarrow z_1 = 0). \quad (3)$$

In this case, the $(2n+1)^{th}$ entry for both Δ_k^1 and Δ_k^2 should be zero. Finally, the node constraint for $\hat{N}$ is $\hat{P} = P \wedge \bigwedge_k P_k$.

One can show that G can simulate a deterministic 2-counter machine in the following sense. We now define two Presburger formulas $\mathcal{P}$ and $\mathcal{Q}$ over all the counters in all of the $(2t+1)$ nodes in G. The formula $\mathcal{P}$ is to specify a condition on $\hat{N}$ corresponding to the initial configuration of M, as shown above,

$$(x_{11}, x_{12}, \ldots, x_{1n}; y_{11}, \ldots, y_{1n}; z_1, z_2; c) = (1, 0, \ldots, 0; 1, 1, \ldots, 1; 0, 0; 1).$$

The formula $\mathcal{Q}$ is to specify a condition on $\hat{N}$ corresponding to a halting configuration of M:

$$(x_{11}, x_{12}, \ldots, x_{1n}; y_{11}, \ldots, y_{1n}; c) = (0, 0, \ldots, 1; 1, 1, \ldots, 1; 1).$$

Clearly, the reachability problem of G wrt $\mathcal{P}$ and $\mathcal{Q}$ is equivalent to the halting problem of M. The result follows. □

We now consider the case when G is *atomic*; i.e., each node constraint P_i in G is a conjunction of atomic linear constraints, i.e., P_i is in the form of

$$\bigwedge(\sum_j a_{ij}x_{ij} \# c_i),$$

where $\# \in \{\leq, \geq\}$, a_{ij} and c_i are integral constants. Using a VAS to simulate an atomic sequential tissue-like P system, one can show:

Theorem 11. *The reachability problem for atomic sequential tissue-like P systems is decidable.*

Proof. In an atomic sequential tissue-like P system G, for the atomic linear constraint $\sum a_{ij}x_{ij} \leq c_i$ imposed upon node i, in VAS we have a counter (vector entry) v_i, whose value is equal to $c_i - \sum a_{ij}x_{ij}$; for $\sum a_{ij}x_{ij} \geq c_i$, we have v_i in VAS, and the value is equal to $\sum a_{ij}x_{ij} - c_i$. Hence $v_i \geq 0$. If the outgoing (resp. incoming) edge of node i is decorated by $\Delta = (d_1, \ldots, d_m)$, we have a vector v in VAS, whose i^{th} entry is $\sum_{j=1}^m a_{ij} * (-d_j)$ (resp. $\sum_{j=1}^m a_{ij} * d_j$), and all other entries are 0. For the conjunction of linear constraints, we simply have several v_i's, and the construction can easily be generalized. Clearly, this VAS can simulate G. The result follows. □

In fact, the converse of Theorem 11 can be shown, i.e., atomic sequential tissue-like P systems and VAS are essentially equivalent, in the following sense. Consider a VAS M with k counters $(x_1, \ldots, x_k)$ and a sequential tissue-like P system G with a distinguished node on which the counters are $(z_1, \ldots, z_l; x_1, \ldots, x_k)$. We further abuse the notation $\leadsto_G$ as follows. We say that $(z_1, \ldots, z_l; x_1, \ldots, x_k)$ *reaches* $(z'_1, \ldots, z'_l; x'_1, \ldots, x'_k)$ in G if there are $\mathcal{C}$ and $\mathcal{C}'$ such that $\mathcal{C} \leadsto_G \mathcal{C}'$ and, $\mathcal{C}$ and $\mathcal{C}'$, when projected on the distinguished node, are $(z_1, \ldots, z_l; x_1, \ldots, x_k)$ and $(z'_1, \ldots, z'_l; x'_1, \ldots, x'_k)$, respectively. We say that M can be *simulated* by G if, for all $(x_1, \ldots, x_k)$ and $(x'_1, \ldots, x'_k)$ in $\mathbb{N}^k$, $(x_1, \ldots, x_k)$ reaches $(x'_1, \ldots, x'_k)$ in M iff $(0, ..., 0; x_1, \ldots, x_k)$ reaches $(0, ..., 0; x'_1, \ldots, x'_k)$ in G. We say that G is *simple* if each constraint P_i in G is a conjunction of $x_{ij} \geq c$ (open constraint) or $x_{ij} \leq c$ (closed constraint), for some constant c and every j, where x_{ij} is a counter in node i . Notice that if G is simple then G must be atomic also. One can show:

Theorem 12. *Every VAS can be simulated by a sequential tissue-like P system G that is a DAG and simple.*

Proof. Suppose that we have n counters and t instructions in a VAS M. Then similar to the graph in Figure 1, we can construct a simple sequential tissue-like P system G, where each node has $n + t + 1$ counters and, in particular, in node $\hat{N}$, counter x_{11} serves as a *control* counter, counters $x_{12}, \ldots, x_{1(t+1)}$ correspond to the instructions of M, and counters $x_{1(t+2)}, \ldots, x_{1(n+t+1)}$ represent the n counters in M. Other nodes in G serve as either suppliers or depositories of objects for node $\hat{N}$, and they will not communicate between themselves. The node constraint for $\hat{N}$ is

$$\bigwedge_{j=1}^{t+1} x_{1j} \leq 1$$

Initially, $\hat{N}$ starts with $(0, \ldots, 0, v_{01}, \ldots, v_{0n})$, where $(v_{01}, \ldots, v_{0n})$ is the tuple of the initial value of counters in M. For any instruction v_i, $1 \leq i \leq t$, in M, we can always make it as a sum of two vectors v_{i1} and v_{i2}, where v_{i1} is nonnegative, and v_{i2} is nonpositive. For example, if $v_i = (3, -2, 6)$, we can get $v_{i1} = (3, 0, 6)$ and $v_{i2} = (0, -2, 0)$. For every v_i, we decorate an incoming edge of $\hat{N}$ with Δ_i^1. In the vector Δ_i^1, the first entry is 1, the $(i+1)^{th}$ entry is 1, the last n entries are v_{i1} , and all other entries are 0. We also decorate an outgoing edge of $\hat{N}$ with Δ_i^2. In the vector Δ_i^2, the first entry is 1, the $(i+1)^{th}$ entry is 1, the last n entries are $-v_{i2}$, and all other entries are 0. Clearly, such a simple sequential tissue-like P system (which is also a DAG) can simulate a VAS. □

For a sequential tissue-like P system G, a very special case is that there is only one counter in each node, and the instruction on an edge (i, j) is $I = 1$, which means that when I is executed, counter i is decremented by 1, and counter j is incremented by 1. We call such a G *single*. One can show that the reachability relation $\rightsquigarrow_G$ is Presburger definable if G is a DAG and single. The proof technique is to "regulate" the reachability paths in G and use reversal-bounded counter machine arguments, and then appeal to Theorem 2.

Theorem 13. *The reachability relation $\rightsquigarrow_G$ is Presburger definable when sequential tissue-like P system G is a DAG and single.*

Proof. Let G be a sequential tissue-like P system that is a DAG and single. Not that in G, each node i stores one counter x_i, and each node is either an *open node* (that is associated with an open constraint), or a *closed* node (that is associated with a closed constraint). Accordingly, we call x_i is an open (resp. closed) counter if node i is open (resp. closed). Now, we consider a procedure as follows, let *done* be a set of open nodes, initially, *done*$= \emptyset$. At each step, we iteratively pick a node i such that each of the ancestors of the node is either a closed node, or an open node in *done*. The procedure halts when *done* equals the set of all open nodes. For an open node i, we say, for two configurations, $\mathcal{C}_1$ and $\mathcal{C}_2$ of G, that $\mathcal{C}_1$ i-monotonically-reaches $\mathcal{C}_2$, written $\mathcal{C}_1 \stackrel{i}{\to} \mathcal{C}_2$, if there a sequence of moves that lead from $\mathcal{C}_1$ to $\mathcal{C}_2$, during which x_i is nonincreasing, and all the other open counters are nondecreasing. Clearly, from Theorem 2, $\stackrel{i}{\to}$ is presburger definable. Suppose that the sequence of open nodes picked in the aforementioned procedure are $i_1, \ldots, i_k$ (in this order), where k is the total number of open nodes in G. We use R to denote the result of "concatenating" all the relations $\stackrel{i_1}{\to}, \ldots, \stackrel{i_k}{\to}$ (Presburger definable relations are closed under existential quantifier elimination). One can show R equals $\rightsquigarrow_G$. The result follows. □

Currently, we do not know whether Theorem 13 still holds when the condition of G being a DAG is removed.

As we pointed out, sequential tissue-like P systems are essentially stateless. To conclude this section, we give an example where some forms of sequential tissue-like P systems become more powerful when states are added, and hence states matter (In contrast to this, VAS and VASS (by adding states to VAS) are known to be equivalent).

Consider a sequential tissue-like P system G where each node contains only one counter and, furthermore, G is a DAG. From Theorem 13, its reachability relation $\rightsquigarrow_G$ is Presburger definable. We now add states to G and show that the reachability relation now is not necessarily Presburger definable. G with states is essentially a multicounter machine M with k counters $(x_1, \ldots, x_k)$ and each counter is associated with a *simple* constraint defined earlier. Each instruction in M is in the following form:

$$(s_p, x_i, x_{i+1}, s_q)$$

where $1 \le i < k$ and, s_p and s_q are the states of M before and after executing the instruction. When the instruction is executed, x_i is decremented by 1, x_{i+1} is incremented by 1, and the simple constraint on each counter should be satisfied (before and after the execution).

Now, we show that an M can be constructed to "compute" the inequality $x * y \ge z$, which is not Presburger definable. We need 8 counters, $x_1, \ldots, x_8$ in M. The idea is that we use the initial value of x_3, x_5 and x_7 to represent x, y and z, respectively, and the remaining counters are auxiliary. In particular, x_1 acts as a "supplier" for supplying counter values. The constraint upon every counter is simply $x_i \ge 0$, $1 \le i \le 8$. Initially, the state is s_0, $x_2 = 0$, and all the other counters store some values. We have the following instructions:

$$I_1 = (s_0, x_3, x_4, s_1);$$

$$I_2 = (s_1, x_1, x_2, s_2);$$

$$I_3 = (s_2, x_7, x_8, s_0);$$

$$I_4 = (s_0, x_5, x_6, s_3);$$

$$I_5 = (s_3, x_2, x_3, s_0);$$

$$I_6 = (s_0, x_2, x_3, s_0).$$

Note that s_3 is the accepting state. I_1, I_2 and I_3 mean that, when x_3, which represents x, is decremented by 1, x_2 will record the decrement, and x_7, which represents z, will also be decremented by 1. I_4 says that during the decrement of x_3, x_5, which represents y, will be nondeterministically decremented by 1. I_5 and I_6 will restore the value of x_3, and after the restoration, the value of x_3 can never surpass the initial one (i.e., x). One can show that $x * y \geq z$ iff M can reach state s_3 (the accepting state) and, at the moment, $x_7 = 0$.

5. Conclusion

We introduced the notion of stateless multihead two-way (respectively, one-way) NFAs and stateless multicounter systems and related them to P systems and vector addition systems. In particular, we investigated the decidability of the emptiness and reachability problems for these stateless automata and showed that the results are applicable to similar questions concerning certain variants of P systems, namely, token systems and sequential tissue-like P systems. Many issues (e.g., the open problems mentioned in the previous sections) remain to be investigated, and we plan to look at some of these in future work.

We thank John Brzozowski at University of Waterloo and referees for the earlier version of this paper presented at ACM'07 for valuable comments.

References

[1] Z. Dang and O. H. Ibarra. On one-membrane p systems operating in sequential mode. *Int. J. Found. Comput. Sci.*, 16(5):867–881, 2005.

[2] S. Ginsburg and E. Spanier. Semigroups, presburger formulas, and languages. *Pacific J. of Mathematics*, 16:285–296, 1966.

[3] O. H. Ibarra. A note on semilinear sets and bounded-reversal multihead pushdown automata. *Inf. Processing Letters*, 3(1): 25-28, 1974.

[4] O. H. Ibarra. Reversal-bounded multicounter machines and their decision problems. *Journal of the ACM*, 25(1):116–133, 1978.

[5] Oscar H. Ibarra, Zhe Dang, and Ömer Egecioglu. Catalytic P systems, semilinear sets, and vector addition systems. *Theor. Comput. Sci.*, 312(2-3):379–399, 2004.

[6] C. Martin-Vide, Gh. Paun, J. Pazos, and A. Rodriguez-Paton. Tissue P systems. *Theor. Comput. Sci.*, 296(2):295–326, 2003.

[7] E. Mayr. An algorithm for the general Petri net reachability problem. *Proc. 13th Annual ACM Symp. on Theory of Computing*, 238–246, 1981.

[8] Gh. Paun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.

[9] Gh. Paun. *Membrane Computing, An Introduction.* Springer-Verlag, 2002.

[10] L. Yang, Z. Dang, and O.H. Ibarra. Bond computing systems: a biologically inspired and high-level dynamics model for pervasive computing. *Proc. of the 6th International Conference on Unconventional Computation (UC'07),* 2007. LNCS, 4618:226–241, Springer, 2007.