Priorities, Promoters and Inhibitors in Deterministic Non-Cooperative P Systems

Artiom Alhazov[1], Rudolf Freund[2]

[1] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Academiei 5, Chişinău MD-2028 Moldova
artiom@math.md
[2] Faculty of Informatics, Vienna University of Technology
Favoritenstr. 9, 1040 Vienna, Austria
rudi@emcc.at

Summary. Membrane systems (with symbol objects) are distributed controlled multiset processing systems. Non-cooperative P systems with either promoters or inhibitors (of weight not restricted to one) are known to be computationally complete. Since recently, it is known that the power of the deterministic subclass of such systems is subregular. We present new results on the weight of promoters and inhibitors, as well as for characterizing the systems with priorities only.

1 Introduction

The most famous membrane computing model where determinism is a criterion of universality versus decidability is the model of catalytic P systems, see [3] and [6].

It is also known that non-cooperative rewriting P systems with either promoters or inhibitors are computationally complete, [2]. Moreover, the proof satisfies some additional properties:

- Either promoters of weight 2 or inhibitors of weight 2 are enough.
- The system is non-deterministic, but it restores the previous configuration if the guess is wrong, which leads to correct simulations with probability 1.

Recently, in [1] it was shown that the computational completeness cannot be achieved by deterministic non-cooperative systems with promoters, inhibitors and priorities (in maximally parallel or asynchronous mode, unlike the sequential mode), and characterizations of the corresponding classes were obtained:

$$\begin{aligned} NFIN \cup coNFIN &= N_{deta}OP_1^{asyn}\left(ncoo, pro_{1,*}, inh_{1,*}\right) \\ &= N_{deta}OP_1^{maxpar}\left(ncoo, pro_{1,*}\right) \\ &= N_{deta}OP_1^{maxpar}\left(ncoo, inh_{1,*}\right) \\ &= N_{deta}OP_1^{asyn}\left(ncoo, \left(pro_{*,*}, inh_{*,*}\right)_*, pri\right) \\ &= N_{deta}OP_1^{maxpar}\left(ncoo, \left(pro_{*,*}, inh_{*,*}\right)_*, pri\right), \text{ but} \\ NRE &= N_{deta}OP_1^{sequ}\left(ncoo, pro_{1,1}, inh_{1,1}\right). \end{aligned}$$

A few interesting questions have been left open. For instance, what is the power of P systems, e.g., in the maximally parallel mode, when we only use priorities, or when we restrict the weight of the promoting/inhibiting multisets. These are the questions we address in this paper.

2 Definitions

An *alphabet* is a finite non-empty set V of abstract *symbols*. The free monoid generated by V under the operation of concatenation is denoted by V^*; the *empty string* is denoted by λ, and $V^* \setminus \{\lambda\}$ is denoted by V^+. The set of non-negative integers is denoted by $\mathbb{N}$; a set S of non-negative integers is called *co-finite* if $\mathbb{N} \setminus S$ is finite. The family of all finite (co-finite) sets of non-negative integers is denoted by $NFIN$ ($coNFIN$, respectively). The family of all recursively enumerable sets of non-negative integers is denoted by NRE. In the following, we will use $\subseteq$ both for the subset as well as the submultiset relation.

Since flattening the membrane structure of a membrane system preserves both determinism and the model, in the following we restrict ourselves to consider membrane systems as one-region multiset rewriting systems.

A *(one-region) membrane system (P system)* is a tuple

$$\Pi = (O, \Sigma, w, R'),$$

where O is a finite alphabet, $\Sigma \subseteq O$ is the input sub-alphabet, $w \in O^*$ is a string representing the initial multiset, and R' is a set of rules of the form $r : u \to v$, $u \in O^+$, $v \in O^*$.

A configuration of the system Π is represented by a multiset of objects from O contained in the region, the set of all configurations over O is denoted by $\mathbb{C}(O)$. A rule $r : u \to v$ is applicable if the current configuration contains the multiset specified by u. Furthermore, applicability may be controlled by *context conditions*, specified by pairs of sets of multisets.

Definition 1. *Let P_i, Q_i be (finite) sets of multisets over O, $1 \leq i \leq m$. A* rule with context conditions $(r, (P_1, Q_1), \cdots, (P_m, Q_m))$ *is applicable to a configuration C if r is applicable, and there exists some $j \in \{1, \cdots, m\}$ for which*

- *there exists some $p \in P_j$ such that $p \subseteq C$ and*
- *$q \not\subseteq C$ for all $q \in Q_j$.*

In words, context conditions are satisfied if there exists a pair of sets of multisets (called *promoter set* and *inhibitor set*, respectively) such that at least one multiset in the promoter set is a submultiset of the current configuration, and no multiset in the inhibitor set is a submultiset of the current configuration.

Definition 2. *A* P system with context conditions and priorities on the rules *is a construct*

$$\Pi = (O, \Sigma, w, R', R, >),$$

where (O, Σ, w, R') *is a (one-region) P system as defined above,* R *is a set of rules with context conditions and* $>$ *is a priority relation on the rules in* R*; if rule* r' *has priority over rule* r*, denoted by* $r' > r$*, then* r *cannot be applied if* r' *is applicable.*

Throughout the paper, we will use the word *control* to mean that at least one of these features is allowed (context conditions or promoters or inhibitors only and eventually priorities).

In the *sequential mode* (*sequ*), a computation step consists in the non-deterministic application of one applicable rule r, replacing its left-hand side ($lhs\,(r)$) with its right-hand side ($rhs\,(r)$). In the *maximally parallel mode* (*maxpar*), a multiset of applicable rules may be chosen non-deterministically to be applied in parallel to the underlying configuration to disjoint submultisets, possibly leaving some objects idle, under the condition that no further applicable rule can be added to that multiset (i.e., no supermultiset of the chosen multiset is applicable to the same configuration). Maximal parallelism is the most common computation mode in membrane computing, see also Definition 4.8 in [5]. In the *asynchronuous mode* (*asyn*), any positive number of applicable rules may be chosen non-deterministically to be applied in parallel to the underlying configuration, to disjoint submultisets. The computation step between two configurations C and C' is denoted by $C \to C'$, thus yielding the binary relation $\Rightarrow: \mathbb{C}(O) \times \mathbb{C}(O)$. A computation halts when there are no rules applicable to the current configuration (*halting configuration*) in the corresponding mode.

The computation of a *generating* P system starts with w, and its result is $|x|$ if it halts, an *accepting* system starts with wx, $x \in \Sigma^*$, and we say that $|x|$ is its results – is accepted – if it halts. The set of numbers generated/accepted by a P system working in the mode α is the set of results of its computations for all $x \in \Sigma^*$ and denoted by $N_g^\alpha(\Pi)$ and $N_a^\alpha(\Pi)$, respectively. The family of sets of numbers generated/accepted by a family of (one-region) P systems with context conditions and priorities on the rules with rules of type β working in the mode α is denoted by $N_\delta OP_1^\alpha\left(\beta, (pro_{k,l}, inh_{k',l'})_d, pri\right)$ with $\delta = g$ for the generating and $\delta = a$ for the accepting case; d denotes the maximal number m in the rules with context conditions $(r, (P_1, Q_1), \cdots, (P_m, Q_m))$; k and k' denote the maximum number of promoters/inhibitors in the P_i and Q_i, respectively; l and l' indicate the maximum of weights of promotors and inhibitors, respectively. If any of these numbers k, k', l, l' is not bounded, we replace it by $*$. As types of rules we are going to distinguish between cooperative ($\beta = coo$) and non-cooperative (i.e., the left-hand side of each rule is a single object; $\beta = ncoo$) ones.

In the case of accepting systems, we also consider the idea of determinism, which means that in each step of any computation at most one (multiset of) rule(s) is applicable; in this case, we write *deta* for δ.

In the literature, we find a lot of restricted variants of P systems with context conditions and priorities on the rules, e.g., we may omit the priorities or the context conditions completely. If in a rule $(r, (P_1, Q_1), \cdots, (P_m, Q_m))$ we have $m = 1$, we say that $(r, (P_1, Q_1))$ is a rule with a *simple context condition*, and we omit the inner parentheses in the notation. Moreover, context conditions only using promoters are denoted by $r|_{p_1,\cdots,p_n}$, meaning $(r, \{p_1, \cdots, p_n\}, \emptyset)$, or, equivalently, $(r, (p_1, \emptyset), \cdots, (p_n, \emptyset))$; context conditions only using inhibitors are denoted by $r|_{\neg q_1,\cdots,\neg q_n}$, meaning $(r, \lambda, \{q_1, \cdots, q_n\})$, or $r|_{\neg\{q_1,\cdots,q_n\}}$. Likewise, a rule with both promoters and inhibitors can be specified as a rule with a simple context condition, i.e., $r|_{p_1,\cdots,p_n,\neg q_1,\cdots,\neg q_n}$ stands for $(r, \{p_1, \cdots, p_n\}, \{q_1, \cdots, q_n\})$. Finally, promoters and inhibitors of weight one are called *atomic*.

Remark 1. If we do not consider determinism, then (the effect of) the rule $(r, (P_1, Q_1), \cdots, (P_m, Q_m))$ is equivalent to (the effect of) the collection of rules $\{(r, P_j, Q_j) \mid 1 \leq j \leq m\}$, no matter in which mode the P system is working (obviously, the priority relation has to be adapted accordingly, too).

Remark 2. Let $(r, \{p_1, \cdots, p_n\}, Q)$ be a rule with a simple context condition; then we claim that (the effect of) this rule is equivalent to (the effect of) the collection of rules

$$\{(r, \{p_j\}, Q \cup \{p_k \mid 1 \leq k < j\}) \mid 1 \leq j \leq m\}$$

even in the the case of a deterministic P system: If the first promoter is chosen to make the rule r applicable, we do not care about the other promoters; if the second promoter is chosen to make the rule r applicable, we do not allow p_1 to appear in the configuration, but do not care about the other promoters p_3 to p_m; in general, when promoter p_j is chosen to make the rule r applicable, we do not allow p_1 to p_{j-1} to appear in the configuration, but do not care about the other promoters p_{j+1} to p_m; finally, we have the rule $\{(r, \{p_m\}, Q \cup \{p_k \mid 1 \leq k < m\})\}$. If adding $\{p_k \mid 1 \leq k < j\}$ to Q has the effect of prohibiting the promotor p_j from enabling the rule r to be applied, this makes no harm as in this case one of the promoters p_k, $1 \leq k < j$, must have the possibility for enabling r to be applied. By construction, the domains of the new context conditions now are disjoint, so this transformation does not create (new) non-determinism. In a similar way, this transformation may be performed on context conditions which are not simple. Therefore, without restricting generality, the set of promoters may be assumed to be a singleton. In this case, we may omit the braces of the multiset notation for the promoter multiset and write (r, p, Q).

Remark 3. As in a P system $(O, \Sigma, w, R', R, >)$ the set of rules R' can easily be deduced from the set of rules with context conditions R, we omit R' in the description of the P system. Moreover, for systems having only rules with a simple

context condition, we omit d in the description of the families of sets of numbers and simply write

$$N_\delta OP_1^\alpha\left(\beta, pro_{k,l}, inh_{k',l'}, pri\right).$$

Moreover, each control mechanism not used can be omitted, e.g., if no priorities and only promoters are used, we only write $N_\delta OP_1^\alpha\left(\beta, pro_{k,l}\right)$.

3 Results

3.1 Recent results

We first recall from [1] the *bounding* operation over multisets, with a parameter $k \in \mathbb{N}$ as follows:

$$\text{for } u \in O^*,\ b_k(u) = v \text{ with } |v|_a = \min(|u|_a, k) \text{ for all } a \in O.$$

The mapping b_k "crops" the multisets by removing copies of every object a present in more than k copies until exactly k remain. For two multisets u, u', $b_k(u) = b_k(u')$ if for every $a \in O$, either $|u|_a = |u'|_a < k$, or $|u|_a \geq k$ and $|u'|_a \geq k$. Mapping b_k induces an equivalence relation, mapping O^* into $(k+1)^{|O|}$ equivalence classes. Each equivalence class corresponds to specifying, for each $a \in O^*$, whether no copy, one copy, or $\cdots\ k-1$ copies, or "k copies or more" are present. We denote the range of b_k by $\{0, \cdots, k\}^O$.

Lemma 1. *[1] Context conditions are equivalent to predicates defined on boundings.*

Theorem 1. *[1] Priorities are subsumed by conditional contexts.*

Remark 4. It is worth to note, see also [4], that if no other control is used, the priorities can be mapped to sets of atomic inhibitors. Indeed, a rule is inhibited precisely by the left side of each higher priority rule. This is straightforward in case when the priority relation is assumed to be a partial order.

If it is not, then both the semantics of computation in P systems and the reduction of priorities to inhibitors is a bit more complicated, but the claim still holds.

Fix an arbitrary deterministic controlled non-cooperative P system. Take k as the maximum of size of all multisets in all context conditions. Then, the bounding does not influence applicability of rules, and $b_k(u)$ is halting if and only if u is halting. We recall that bounding induces equivalence classes preserved by any computation.

Lemma 2. *[1] Assume $u \to x$ and $v \to y$. Then $b_k(u) = b_k(v)$ implies $b_k(x) = b_k(y)$.*

Corollary 1. *[1] If $b_k(u) = b_k(v)$, then u is accepted if and only if v is accepted.*

Finally, the "at most $NFIN \cup coNFIN$" part of characterizing

$$N_{deta}OP_1^{maxpar}\left(ncoo, (pro_{*,*}, inh_{*,*})_*, pri\right)$$

(the main theorem of [1]) is shown with the following argument:

> Each equivalence class induced by bounding is completely accepted or completely rejected. If no infinite equivalence class is accepted, then the accepted set is finite (containing numbers not exceeding $(k-1)\cdot|O|$). If at least one infinite equivalence class is accepted, then the rejected set is finite (containing numbers not exceeding $(k-1)\cdot|O|$).

3.2 Priorities only

We start with an example how to deterministically rewrite an object t depending on the presence or absence of object a.

Example 1.

$$\Pi = (\{a, A, A', t, t', t_+, t_-\}, \{a\}, tA, R, R, >), \text{ where}$$
$$R = \{1 : t \to t',\ 2 : a \to \lambda,\ 3 : A \to A',\ 4 : t' \to t_+,\ 5 : t' \to t_-,\ 6 : A' \to \lambda\},$$
$$> = \{a \to \lambda > A \to A',\ A \to A' > t' \to t_-,\ A' \to \lambda > t' \to t_+\}.$$

Indeed, object t waits for one step by becoming t', while A has to change to A' or wait, depending on the presence of a. Then, object t' becomes either t_+ or t_-, depending on whether A or A' is present. Notice, e.g., how adding either rule $t_+ \to t_+$ or rule $t_- \to t_-$ leads to a system accepting $\{0\}$ or $\mathbb{N} \setminus \{0\}$. Of course, accepting only zero could instead be done by a trivial one-rule system, but this example is important because such a deciding subsystem can be used, with suitable delays, as a building block for checking combinations of presence/absence of multiple symbols.

We now proceed with characterizing systems with priorities only.

Theorem 2. $N_{deta}OP_1^{maxpar}(ncoo, pri) = \{\mathbb{N}_k, \mathbb{N}_k \cup \{0\} \mid k \geq 0\} \cup \{\{0\}, \emptyset\}$.

Proof. We already know that the priorities correspond to sets of atomic inhibitors. This means that each system accepts a union of some equivalence classes induced by bounding b_1 (i.e., checking presence/absence). Note that various combinations of "$= 0$" and "≥ 1" yield numeric sets $\{0\}$ and $\mathbb{N}_k$ (where $k > 0$ is the number of different symbols present). The family of all unions of these sets is

$$F_{pri} = \{\mathbb{N}_k, \mathbb{N}_k \cup \{0\} \mid k \geq 0\} \cup \{\{0\}, \emptyset\}.$$

It follows that $N_{deta}OP_1^{maxpar}(ncoo, pri) \subseteq F_{pri}$.

We proceed with the converse inclusion. Let $\Pi_0 = (\{a, t\}, \{a\}, t, R, R, >)$, then $R = \{t \to t\}$ and empty relation $>$ yields $\emptyset$. To accept $\{0\}$, we instead take $R = \{a \to a\}$ and empty relation $>$.

Now suppose we want to accept $\mathbb{N}_k$. It would suffice to count that we have at least one of each objects $a_1, \cdots, a_k$ (we recall that we need to accept *at least one* input of size j for each $j \geq k$, or reject the input if $j > k$). To accept $\mathbb{N}_k \cup \{0\}$ instead, we may first perform a simultaneous check for the absence of all input symbols.

Using the idea from Example 1, we construct the system

$$
\begin{aligned}
\Pi_1 &= (O, \Sigma = \{a_{i,0} \mid 1 \leq i \leq k\}, tA_{0,0} \cdots A_{k,0}, R, R, >), \text{ where} \\
O &= \{a_{i,j} \mid 1 \leq i \leq k,\ 0 \leq j \leq i+1\} \cup \{A_{i,j} \mid 0 \leq i \leq k,\ 0 \leq j \leq i+2\} \\
&\quad \cup \{t, z, p\} \cup \{t_i \mid 0 \leq i \leq i+1\}, \\
R &= \{1 : a_{i,j} \to a_{i,j+1} \mid 1 \leq i \leq k,\ 0 \leq j \leq i\} \\
&\quad \cup \{2 : A_{i,j} \to A_{i,j+1} \mid 1 \leq i \leq k,\ 0 \leq j \leq i+1\} \\
&\quad \cup \{3 : t \to t_0,\ 4 : t_0 \to z,\ 5 : t_0 \to t_1,\ 6 : p \to p\} \\
&\quad \cup \{7 : t_i \to t_{i+1},\ 8 : t_i \to p \mid 1 \leq i \leq k\}, \\
> &= \{a_{i,0} \to a_{i,1} > A_{0,0} \to A_{0,1} \mid 1 \leq i \leq k\} \\
&\quad \cup \{A_{0,0} \to A_{0,1} > t_0 \to z,\ A_{0,1} \to A_{0,2} > t_0 \to t_1\} \\
&\quad \cup \{a_{i,i} \to a_{i,i+1} > A_{i,i} \to A_{i,i+1} \mid 1 \leq i \leq k\} \\
&\quad \cup \{A_{i,i} \to A_{i,i+1} > t_i \to p,\ A_{i,i+1} \to A_{i,i+2} > t_i \to t_{i+1}\}.
\end{aligned}
$$

Such system accept exactly $\mathbb{N}_k \cup \{0\}$. Indeed, after first step, $A_{0,0}$ is present if all input symbols were absent, otherwise $A_{0,1}$ is present instead. For any i, $1 \leq i \leq k$, after step $1+i$, object $A_{i,i}$ is present if input symbol $a_{i,0}$ was present in the input, and otherwise $A_{i,i+1}$ is present instead. These "decision symbols" are used by t_i, $0 \leq i \leq k$, to build the "presence picture". We recall that it suffices to accept when all input symbols are present, or when none of them is present. In the first case, t_0 becomes z, and the computation only continues by rules from groups 1 and 2, leading to halting. Let us assume that the first s of the input symbols are present, $s < k$. Then, t_0 becomes t_1, and then $\cdots$, t_s, and then the absence of t_{s+1} will change t_s into p, leading to an infinite computation. Finally, if all input symbols are present, then the computation will halt with t_{k+1}.

It remains to notice that accepting $\mathbb{N}_k$, $k \geq 1$, can be done by simply adding a rule $z \to z$. □

3.3 Promoters or inhibitors of weight 2

We start from examples, illustrating deterministic choice of rewriting p, depending on whether object a is absent, occurs exactly once, or occurs multiple times.

Example 2. Symbols A, B are primed if input is present (multiple input symbols are present). Then primed and unprimed symbols form mutually exclusive conditions.

$$\begin{aligned}
&\Pi = (O = \{p, p', p'', p_>, p_1, p_0, A, B, a\}, \Sigma = \{a\}, pAB, R', R), \text{ where}\\
&R' = \{1 : p \to p',\ 2 : A \to A',\ 3 : B \to B',\\
&\qquad 4 : p' \to p_>,\ 5 : p' \to p'',\ 6 : p'' \to p_1,\ 7 : p'' \to p_0\},\\
&R' = \{1 : p \to p',\ 2 : A \to A'|_a,\ 3 : B \to B'|_{aa},\\
&\qquad 4 : p' \to p_>|_B,\ 5 : p' \to p''|_{B'},\ 6 : p'' \to p_1|_A,\ 7 : p'' \to p_0|_{A'}\}.
\end{aligned}$$

Example 3. Notice that if we replace all promoters by inhibitors with the **same** context, the effect of blocking rules will be reversed, but the result will be the same. Indeed, the role of A' and B' will switch from found a and found aa, respectively, to not found a and not found aa, respectively.

$$\begin{aligned}
R' = \{&1 : p \to p',\ 2 : A \to A'|_{\neg a},\ 3 : B \to B'|_{\neg aa},\\
&4 : p' \to p_>|_{\neg B},\ 5 : p' \to p''|_{\neg B'},\ 6 : p'' \to p_1|_{\neg A},\ 7 : p'' \to p_0|_{\neg A'}\}.
\end{aligned}$$

We now proceed with characterizing systems with context of weight two. Notice that we already know that their power does not exceed $NFIN \cup coNFIN$.

Theorem 3. $N_{deta}OP_1^{maxpar}(ncoo, pro_2) =$
$N_{deta}OP_1^{maxpar}(ncoo, inh_2) = NFIN \cup coNFIN.$

Proof. We use the technique from Example 2 for all input symbols and combine the extracted information. Consider an arbitrary finite set M, and let $\max(M) = n$. We will use the following strategy: to accept a number $j \in M$, we will accept an input multiset with exactly j symbols appearing once, and nothing else. To accept the complement of M, we split it into sets $M'' = \{j \mid j > n\}$ and $M' = \{j \mid j \leq n,\ j \notin M\}$. While M' is treated similarly to M, it only remains to accept M'', which is covered by equivalence classes when all symbols are present, and at least one is present more than once.

$$\begin{aligned}
&\Pi = (O, \Sigma = \{a_i \mid 1 \leq i \leq n\}, tA_1 \cdots A_n B_1 \cdots B_n, R', R), \text{ where}\\
&O = \{t_{i,j}, T_{i,j}, t'_{i,j}, T'_{i,j} \mid 1 \leq i \leq n+1,\ 0 \leq j \leq n\}\\
&\quad \cup \{A_i, A'_i, B_i, B'_i \mid 1 \leq i \leq n\} \cup \{t, \#\},\\
&R' = \{t_{i,j} \to T_{i+1,j+1},\ T_{i,j} \to T_{i+1,j+1},\ t_{i,j} \to t'_{i,j},\ T_{i,j} \to T'_{i,j},\\
&\qquad t'_{i,j} \to t_{i+1,j+1},\ T'_{i,j} \to T_{i+1,j+1},\ t'_{i,j} \to t_{i+1,j},\ T'_{i,j} \to T_{i+1,j},\\
&\qquad A_i \to A'_i,\ B_i \to B'_i \mid 1 \leq i \leq n\} \cup \{t \to t_{1,0}\ \# \to \#\}\\
&\quad \cup \{T_{i,n+1} \to \# \mid 1 \leq i \leq n\} \cup \{t_{i,n+1} \to \# \mid i \notin M\},\\
&R = \{t_{i,j} \to T_{i+1,j+1}|_{B_i},\ T_{i,j} \to T_{i+1,j+1}|_{B_i},\ t_{i,j} \to t'_{i,j}|_{B'_i},\ T_{i,j} \to T'_{i,j}|_{B'_i},\\
&\qquad t'_{i,j} \to t_{i+1,j+1}|_{A_i},\ T'_{i,j} \to T_{i+1,j+1}|_{A_i},\ t'_{i,j} \to t_{i+1,j}|_{A'_i},\ T'_{i,j} \to T_{i+1,j}|_{A'_i},\\
&\qquad A_i \to A'_i|_{a_i},\ B_i \to B'_i|_{a_i a_i} \mid 1 \leq i \leq n\} \cup \{t \to t_{1,0},\ \# \to \#\}\\
&\quad \cup \{T_{i,n+1} \to \# \mid 1 \leq i \leq n\} \cup \{t_{i,n+1} \to \# \mid i \notin M\}.
\end{aligned}$$

The meaning of $T_{i,n+1}$ is that exactly i input symbols are present, and at least one of them is present multiple times. The meaning of $t_{i,n+1}$ is that the input consisted of exactly i different symbols. This is how an arbitrary finite set is accepted. To accept instead of M its complement, replace $i \notin M$ by $i \in M$ and remove rule $T_{n,n+1} \to \#$. Therefore, deterministic P systems with promoters of weight two accept exactly $NFIN \cup coNFIN$.

For the inhibitor counterpart, notice that the computation of the number of different symbols present, as well as checking if any symbol is present multiple times, stays correct by simply changing promoters to the inhibitors with the same condition, just like in Example 3. Rules processing objects $t_{i,n+1}$ and $T_{i,n+1}$ will have an opposite effect, accepting the complement of the set accepted by the system with promoters, again yielding $NFIN \cup coNFIN$. □

It is still open whether only inhibitors in the rules or only promoters in the rules are sufficient to yield $NFIN \cup coNFIN$ with the asynchronuous mode, too.

4 Conclusion

We have shown the characterizations of deterministic non-cooperative P systems with inhibitors of weight 2, with promoters of weight 2, and with priorities. The first two cases did not reduce the accepting power with respect to unrestricted weight.

References

1. A. Alhazov, R. Freund: Asynchronuous and Maximally Parallel Deterministic Controlled Non-Cooperative P Systems Characterize $NFIN$ and $coNFIN$. *The Tenth Brainstorming Week in Membrane Computing*, vol. 1, Sevilla, 2012, 25–34, and *Membrane Computing - 13th International Conference*, CMC13, Budapest (E. Csuhaj-Varjú, M. Gheorghe, G. Rozenberg, A. Salomaa, Gy. Vaszil, Eds.), *Lecture Notes in Computer Science* **7762**, 2013, 101-111.
2. A. Alhazov, D. Sburlan: Ultimately Confluent Rewriting Systems. Parallel Multiset-Rewriting with Permitting or Forbidding Contexts. In: G. Mauri, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg, A. Salomaa: *Membrane Computing, 5th International Workshop*, WMC 2004, Milano, Revised Selected and Invited Papers, *Lecture Notes in Computer Science* **3365**, Springer, 2005, 178–189.
3. R. Freund, L. Kari, M. Oswald, P. Sosík: Computationally Universal P Systems without Priorities: Two Catalysts are Sufficient, *Theoretical Computer Science* **330**, 2, 2005, 251–266.
4. R. Freund, M. Kogler, M. Oswald, A General Framework for Regulated Rewriting Based on the Applicability of Rules. In: J. Kelemen, A. Kelemenová, *Computation, Cooperation, and Life*, Springer, *Lecture Notes in Computer Science* **6610**, 2011, 35–53.

5. R. Freund, S. Verlan: A Formal Framework for Static (Tissue) P Systems. Membrane Computing, 8th International Workshop, WMC 2007 Thessaloniki, 2007, Revised Selected and Invited Papers (G. Eleftherakis, P. Kefalas, Gh. Păun, G. Rozenberg, A. Salomaa, Eds.), *Lecture Notes in Computer Science* **4860**, 2007, 271–284.
6. O.H. Ibarra, H.-C. Yen: Deterministic Catalytic Systems are Not Universal, *Theoretical Computer Science* **363**, 2006, 149–161.
7. M.L. Minsky: *Finite and Infinite Machines*, Prentice Hall, Englewood Cliffs, New Jersey, 1967.
8. Gh. Păun: *Membrane Computing. An Introduction*, Springer, 2002.
9. Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *The Oxford Handbook of Membrane Computing*, Oxford University Press, 2010.
10. G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages*, 3 vol., Springer, 1997.
11. P systems webpage. `http://ppage.psystems.eu`