Deterministic Non-cooperative P Systems with Strong Context Conditions

Artiom Alhazov[1], Rudolf Freund[2]

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

Summary. We continue the line of research of deterministic parallel non-cooperative multiset rewriting with control. We here generalize control, i.e., rule applicability context conditions, from promoters and inhibitors checking presence or absence of certain object up to some bound, to regular and even stronger predicates, focusing at predicates over multiplicity of one symbol at a time.

1 Introduction

It is known, see [7], that non-cooperative P systems with atomic promoters or atomic inhibitors characterize $ET0L$, while using either one catalyst, see [6], [3], or promoters or inhibitors of weight 2, see [4], leads to the computational completeness of non-cooperative P systems. A question about the power of deterministic systems was posed in [5], inspired by the fact that all identical objects have the same behavior in the same context. This question was answered in [1]: deterministic non-cooperative P systems have weak behaviour, namely, only accepting finite number sets and their complements, even using generalized context conditions (except the sequential case, when they keep the computational completeness).

Generalized context conditions of rule applicability are defined as a list of pairs (p_i, F_i), $1 \leq i \leq k$, applicable to a rule if at least one condition applies, in the following way: p_i, called promoter, must be a submultiset of the current configuration (or the contents of the current region), and none of the elements of F_i, called inhibitors, are allowed to be submultisets of the current configuration (or the contents of the current region). A subsequent paper, [2], precisely characterized the power of priorities alone, as well as established how much power of promoters and inhibitors is actually needed to reach $NFIN \cup coNFIN$. Already in [1] it has been shown that generalized context conditions are equivalent to arbitrary predicates on boundings, i.e., all boolean combinations over conditions $< m$ (and,

hence, also $\geq m$, $> m$, $\leq m$, $= m$ and $\neq m$) for multiplicities of symbols. In other words, generalized context conditions are able to check exactly the multiplicities of symbols up to an arbitrary fixed bound m. In this paper we consider stronger context conditions.

2 Definitions

Let O be a finite alphabet. In this paper we will not distinguish between a multiset, its string representation (having as many occurrences of every symbol as its multiplicity in the multiset, the order in the string being irrelevant), and a vector of multiplicities (assuming that the order of enumeration of symbols from O is fixed). By O° we denote the set of all multisets over O. By a strong context in this paper we mean a language of multisets, i.e., a subset of O°.

Let $a \in O$ and $u \in O^\circ$, then $a \to u$ is a non-cooperative rule. The rules are applied in the maximally parallel way, which in the case of our interest, i.e., for deterministic non-cooperative P systems, correspond to replacing every occurrence of each symbol a by the corresponding multiset u from the right side of the applicable rule (if there is any; no competition between different rules can happen due to the determinism). Let region j of a membrane system contain multiset w.

Then rule $a \to u$ with a strong context condition $C \subset O^\circ$ (written $a \to u|C$) is applicable if and only if $|w|_a > 0$ and $w \in C$. Consider the following examples:

- a singleton atomic promoter $s \in O$ corresponds to the context $+(s) = \{w \in O^\circ \mid |w|_s > 0\}$; we denote this feature by $pro_{1,1}$;
- a singleton atomic inhibitor $s \in O$ corresponds to the complementary context condition: $-(s) = \{w \in O^\circ \mid |w|_s = 0\}$;
- a singleton promoter $s \in O^\circ$ of a higher weight corresponds to the context $+(s) = \{w \in O^\circ \mid s \subseteq w\}$;
- a singleton inhibitor $s \in O^\circ$ of a higher weight corresponds to the complementary context condition: $-(s) = \{w \in O^\circ \mid s \not\subseteq w\}$;
- a (finite) promoter-set $S \subset O^\circ$ corresponds to the context $+(S) = \bigcup_{s \in S} +(s)$, i.e., at least one promoter must be satisfied;
- a (finite) inhibitor-set $S \subset O^\circ$ corresponds to the complementary $-(S) = \bigcap_{s \in S} -(s)$, i.e., any inhibitor can forbid the rule;
- a promoter-set P and an inhibitor-set Q together are called a *simple context condition*, written (P, Q); it corresponds to the strong context condition $+(P) \cap -(Q)$;
- context conditions as considered in [1] and [2] constitute a finite collection of simple context conditions $(P_1, Q_1), \cdots, (P_m, Q_m)$, they correspond to the strong context condition $\bigcup_{1 \leq i \leq m} (+(P_i) \cap -(Q_i))$, and were shown to be equivalent to predicates on boundings[3];

[3] the meaning of a promoter-set in [3] is different, but the computational power results are equivalent up to the descriptional complexity parameters such as number of promoters/inhibitors and their weights

- a bounding b_k is an operation on a multiset, for any symbol preserving its multiplicity up to k, or "cropping" it down to k otherwise; a predicate on bounding can be specified by a finite set M of multisets with multiplicities not exceeding k; it corresponds to a strong context condition $\{w \in O^\circ \mid b_k(w) \in M\}$, and can express precisely all boolean combinations of conditions $|w|_a < j$, $a \in O$, $1 \leq j \leq k$;
- a regular strong context condition can be specified by a regular multiset language, or as a Parikh image of a regular string language; e.g., $\mathtt{Eq}(a, b) = \{w \in O^\circ \mid |w|_a = |w|_b\}$ is an example; we denote the family of such conditions by $ctxt(REG)$;
- if a strong context condition only depends on the multiplicities of k symbols from O (and all other symbols do not affect the applicability), we represent this property by a superscript k of $ctxt$; for instance, if we denote the symbols mentioned above by $S = \{s_1, \cdots, s_k\}$, then $ctxt^k(REG) = \{\{u \cup v \mid u \in L,\ v \in (O \setminus S)^\circ\} \mid L \subseteq S^\circ,\ L \in PsREG\}$; hence $\mathtt{Eq}(a, b) \in ctxt^2(REG)$; by $ctxt(\mathtt{Eq})$ we denote being able to compare the multiplicities of two symbols (for different pairs of symbols separately) for being equal, together with the complementary condition;
- to stay within Turing computability of the resulting P systems, in this paper we only consider recursive context conditions, i.e., multiset languages with decidable membership, denoted by $ctxt(REC)$;
- if a one-symbol strong context condition only depends on the multiplicity of one symbol, it can be specified by a predicate over $\mathbb{N}$; e.g., $\mathtt{Sq}(a) = \{w \in O^\circ \mid |w|_a = k^2,\ k \geq 0\}$ and $\mathtt{Sq}'(a) = \{w \in O^\circ \mid |w|_a = k^2,\ k \geq 1\}$ are examples; hence, $\mathtt{Sq}, \mathtt{Sq}' \in ctxt^1(REC)$; by $ctxt(\mathtt{Sq})$ or $ctxt(\mathtt{Sq}')$ we denote being able to test the multiplicities (of different symbols separately) for squares (including zero or not, respectively), together with the complementary condition.

3 Regular conditions

Theorem 1. $Ps_aDOP_1(ncoo, ctxt^2(REG)) =$
$Ps_aDOP_1(ncoo, ctxt(\mathtt{Eq})) = PsRE.$

Proof. Consider an arbitrary register machine M with m registers. For each working register i, $1 \leq i \leq m$, we represent its value by the difference of the multiplicities of associated objects a_i and b_i. Hence, increment can be performed by producing one copy of a_i, decrement can be performed by producing one copy of b_i, and zero can be distinguished from non-zero by the following regular conditions:

$$Z_i = \{w \in O^\circ \mid |w|_{a_i} = |w|_{b_i}\} = \mathtt{Eq}(a_i, b_i),\ 1 \leq i \leq m,$$
$$P_i = \{w \in O^\circ \mid |w|_{a_i} \neq |w|_{b_i}\} = O^\circ \setminus \mathtt{Eq}(a_i, b_i),\ 1 \leq i \leq m,$$

We construct the following P system:

$$\Pi = (O, \Sigma, \mu = [\ ]_1, w_1 = q_0, R_1), \text{ where}$$

$$
\begin{aligned}
O &= Q \cup T \cup \{a_i, b_i \mid 1 \leq i \leq m\},\\
\Sigma &\subseteq \{a_i \mid 1 \leq i \leq m\},\\
R_1 &= \{q \to a_i q' \mid q : (ADD(i), q') \in P\}\\
&\quad \cup \{q \to b_i q' | P_i,\ q \to q'' | Z_i \mid q : (SUB(i), q', q'') \in P\}.
\end{aligned}
$$

□

If only regular conditions over *one* symbol are allowed, then we expect the power of such P systems to be much more limited.

4 Stronger Conditions

Consider one-symbol context conditions that are even stronger than regular.

It is expected that, with recursively enumerable conditions over one number we get something like $NRE \cup coNRE$, so we look at intermediate cases. We look at ways of obtaining RE by encoding a number by a multiplicity of *one* object, say, a_i, in such a way that increment and decrement are reasonably simple to perform by non-removable objects. We propose the following encoding: "ignoring the greatest square", i.e., number $n = k^2 + t$ encodes t if $0 \leq t < 2k + 1$. In this way, zero-test becomes a test whether the encoding number is a perfect square. Increment is performed as increment of the encoding number, followed by addition of $2k + 1$ if the next perfect square, i.e., $(k + 1)^2$, is reached. Decrement can thus be done by adding $2k$ to the encoding number. The value k can be stored as the multiplicity of another non-removable object, say, b_i, whose multiplicity should be incremented each time the encoding number is increased by $2k$ or by $2k + 1$. Putting it all together, the following construction is obtained:

$$
\begin{aligned}
Z_i &= \{w \in O^\circ \mid |w|_{a_i} = k^2,\ k \geq 0\} = \mathtt{Sq}(a_i),\ P_i = O^\circ \setminus Z_i,\ 1 \leq i \leq m,\\
&\quad \text{We construct the following P system:}\\
\Pi &= (O, \Sigma, \mu = [\ ]_1, w_1 = q_0, R_1), \text{ where}\\
O &= Q \cup T \cup \{a_i, b_i \mid 1 \leq i \leq m\},\\
\Sigma &\subseteq \{a_i \mid 1 \leq i \leq m\},\\
R_1 &= \{q \to a_i \tilde{q},\ \tilde{q} \to q' | P_i,\ \tilde{q} \to \hat{q} | Z_i,\ \hat{q} \to a_i b_i q',\ b_i \to a_i a_i b_i |_{\hat{q}}\\
&\quad \mid q : (ADD(i), q') \in P\}\\
&\quad \cup \{q \to q'' | Z_i,\ q \to \hat{q} | P_i,\ \hat{q} \to b_i q',\ b_i \to a_i a_i b_i |_{\hat{q}}\\
&\quad \mid q : (SUB(i), q', q'') \in P\}.
\end{aligned}
$$

Yet there is a major drawback of this result established above in comparison with the result from Theorem 1, as the input has to be encoded: given a number n_i for input register i, we have to compute numbers $n_i + k_i^2$ and k_i, such that $k_i^2 \leq n_i \leq k_i^2 + 2k_i$. But this is an algorithm which is not difficult to be implemented;

also our context condition for testing a number to be a perfect square does not require a difficult algorithm.

Hence, we have just shown the following result, where the index wa instead of a in $Ps_{wa}DOP_1(ncoo, pro_{1,1}, ctxt(\mathtt{Sq}))$ indicates weak computational completeness as for having to encode the input:

Theorem 2. $Ps_{wa}DOP_1(ncoo, ctxt^1(REC)) = PsPs_{wa}DOP_1(ncoo, pro_{1,1}, ctxt(\mathtt{Sq})) = PsRE.$

Adding rules $a_i \to \lambda|_{q_f}$, $b_i \to \lambda|_{q_f}$ and $q_f \to \lambda$ for $1 \leq i \leq m$, where q_f is the final state of the simulated register machines, we even obtain the clean result, i.e., halting without additional objects, still preserving determinism.

We can strengthen the claim of Theorem 2 by showing **strong** computational completeness (in the sense of deterministic acceptance and even deterministic way of computing functions). Without restricting the power of register machines, we assume that in the simulated register machine, the output registers are never decremented. Then, for the output registers, we replace the simulation of each increment instructions with a single rule $q \to a_i q'$, where $q : (ADD(i), q') \in P$ and i is an output register. In this way, the output will be produced without encoding.

It remains to show that P systems with strong context conditions over one symbol can simulate register machines where also the **input** is not encoded. We use the following idea. To represent the input N of a register in the way the P system constructed in the proof of Theorem 2 needs it, we first describe how to get two numbers x_N and y_N such that N is a function of x_N and y_N, and, moreover, by computing these two numbers from N, we get their representation in the form we need them as for the P system constructed in Theorem 2.

First we explain the algorithm how to obtain x_N and y_N: Starting with N represented by N copies of an object c_N, the multiplicity of these input objects is incremented until it becomes a perfect square (counting the increments, thus finally obtaining x_N), and then incrementing it (again counting the increments, thus finally obtaining y_N) until it again becomes a perfect square. From these two numbers x_N and $y_N$we can regain N by the formula computed in the following:

Given input N, the next perfect squares are $k_N^2 = N + x_N$ $(x_N \geq 0)$ and $(k_N + 1)^2 = N + x_N + y_N$, then $y_N = 2k_N + 1$, so $k_N = (y_N - 1)/2$, and $N = k_N^2 - x_N = (y_N - 1)^2/4 - x_N$. Of course, the function $f(x_N, y_N) = (y_N - 1)^2/4 - x_N$ decoding N from x_N and y_N can be implemented by a register machine and simulated by a P system as described in Theorem 2.

In the following example we specify more formally the precomputing block mentioned above.

Example 1. Encoding the input number N.

Let the input N be given as a multiplicity of symbol c_i, and we want to obtain values x_N and y_N described above in auxiliary registers j and l, respectively, but represented already in the way we need their contents x_N and y_N implemented

with the corresponding number of symbols a_j and b_j as well as a_l and b_l. We also use an additional starting object s_i and in sum the following rules:

$$
\begin{aligned}
&s_i \to c_i a_j \tilde{s}_i | P_i',\ \tilde{s}_i \to s' | P_j,\ \tilde{s}_i \to \hat{s}_i | Z_j,\ \hat{s}_i \to a_j b_j s',\ b_j \to a_j a_j b_j |_{\hat{s}_i},\\
&s_i \to c_i t_i | Z_i',\\
&t_i \to c_i a_l \tilde{t}_i | P_i',\ \tilde{t}_i \to t_i' | P_l,\ \tilde{t}_i \to \hat{t}_i | Z_l,\ \hat{t}_i \to a_l b_k t',\ b_k \to a_l a_l b_l |_{\hat{t}_i},\\
&t_i \to q_0^{(i)} | Z_i',\ \text{where}\\
&Z_i' = \{w \in O^\circ \mid |w|_{c_i} = k^2,\ k \geq 0\} = \mathtt{Sq}(c_i),\ P_i' = O^\circ \setminus Z_i'.
\end{aligned}
$$

Essentially, the rules above are exactly like increment instructions from Theorem 2, tracking how many times the multiplicity of the input object c_i has to be incremented to reach a perfect square and the next perfect square.

In the next phase of the encoding procedure, the P system should simulate a register machine which starts in state $q_0^{(i)}$ and computes the function $f(x_N, y_N) = (y_N - 1)^2/4 - x_N$, given x_N in register j and y_N in register l, producing the result (i.e., the value N of the input register i to be represented) in register i, represented by symbols a_i and b_i and thus in a suitable way to be the input for the P system constructed in Theorem 2.

Theorem 3. $Ps_aDOP_1(ncoo, ctxt^1(REC)) = Ps_aDOP_1(ncoo, pro_{1,1}, ctxt(\mathtt{Sq})) = PsRE.$

Proof. Clearly, any input vector can be processed accordingly in the way described in Example 1, and then a simulation of the register machine on these inputs as outlined in Theorem 2 completes the explanation of the following result. □

The construction in Theorem 3 may be adjusted so that we never rely on multiplicities of symbols a_i being zero, i.e., when starting with a value 0 in a register, we start with encoding it by 1. Moreover, testing for the appearance of a symbol which never appears more than once (which we needed for the symbols corresponding to the states of the simulated register machine) corresponds with testing for a perfect square of positive integers. Hence, for each checking set from $\mathtt{Sq}'$ (or its complement) or each singleton promoter used in the previous construction we can use a set from $\mathtt{Sq}'$ (or its complement) only. In sum we get:

Corollary 1. $Ps_aDOP_1(ncoo, ctxt(\mathtt{Sq}')) = PsRE.$

5 Conclusions

It was known that generalized context conditions are equivalent to predicates on boundings, and that using them in deterministic maximally parallel non-cooperative P systems still leaves their accepting power as low as $NFIN \cup coNFIN$. We have shown that regular context conditions yield computational

completeness of deterministic maximally parallel non-cooperative P systems, expecting that the power of P systems with regular context conditions over one symbol is still quite limited. However, we have shown computational completeness using a simple stronger one-symbol context condition, namely, $\{w \in O^\circ \mid |w|_{a_i} = k^2,\ k \geq 0\}$.

References

1. A. Alhazov, R. Freund: Asynchronous and Maximally Parallel Deterministic Controlled Non-Cooperative P Systems Characterize $NFIN$ and $coNFIN$. In: E. Csuhaj-Varjú, M. Gheorghe, G. Rozenberg, A. Salomaa, Gy. Vaszil: *Membrane Computing – 13th International Conference*, CMC13, Budapest, Revised Selected Papers, Lecture Notes in Computer Science **7762**, Springer, 2013, 101–111.
2. A. Alhazov, R. Freund: Priorities, Promoters and Inhibitors in Deterministic Non-Cooperative P Systems. In: M. Gheorghe, G. Rozenberg, A. Salomaa, P. Sosík, C. Zandron: *Membrane Computing - 15th International Conference*, CMC 2014, Prague, Revised Selected Papers, Lecture Notes in Computer Science **8961**, Springer, 2014, 86–98.
3. A. Alhazov, R. Freund, S. Verlan: Promoters and Inhibitors in Purely Catalytic P Systems. In: M. Gheorghe, G. Rozenberg, A. Salomaa, P. Sosík, C. Zandron: *Membrane Computing - 15th International Conference*, CMC 2014, Prague, Revised Selected Papers, Lecture Notes in Computer Science **8961**, Springer, 2014, 126–138.
4. 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, Milan, Revised Selected and Invited Papers, Lecture Notes in Computer Science **3365**, Springer, 2005, 178–189.
5. M. Gheorghe, Gh. Păun, M.J. Pérez-Jiménez, G. Rozenberg: Research Frontiers of Membrane Computing: Open Problems and Research Topics. *International Journal of Foundations of Computer Science* **24** (5), 2013, 547–624.
6. M. Ionescu, D. Sburlan: On P Systems with Promoters/Inhibitors. *Journal of Universal Computer Science* **10** (5), 2004, 581–599.
7. D. Sburlan: Further Results on P Systems with Promoters/Inhibitors. *International Journal of Foundations of Computer Science* **17** (1), 2006, 205–221.