

# A Note on Parallel Rewriting in P Systems

Shankara Narayanan KRISHNA, Raghavan RAMA

Department of Mathematics, Indian Institute of Technology  
Madras, Chennai-36, Tamilnadu, India  
E-mail: ramar@iitm.ac.in

**Abstract.** We consider a variant of rewriting P systems, called *with parallel parallelism*, where at each step, one occurrence of each symbol which can be rewritten is rewritten. We prove that such systems, with no bound on the number of membranes, characterize the family of recursively enumerable languages, while systems with four membranes can generate non-matrix languages.

## 1 Introduction

The P systems are distributed computing models introduced in [2], which mimic the way the alive cells process chemical compounds. These systems are based on a membrane structure, consisting of several hierarchically embedded membranes, which delimit regions; in these regions certain objects evolve according to given rules, also associated with the regions. Several variants have been considered; we refer to [2], [3], [4] for details.

For instance, the objects placed in the regions of a P system can be strings, processed by rewriting rules. The case when these rules are used in a sequential manner (all strings are processed in parallel, but each string is rewritten by only one rule, only one symbol of the string is rewritten) was investigated already in [2]. The case of parallel rewriting of the strings was considered in [1]. In the parallel variant, all symbols which can be rewritten should be rewritten.

In this paper we consider an intermediate case, where only a partial parallelism is enforced: for each string, each symbol which can be rewritten must be rewritten, but if several occurrences of a symbol exist, then only one is rewritten. It is somewhat surprising to find that such P systems can characterize the recursively enumerable languages in a rather natural/easy manner.

## 2 P Systems with Parallel Rewriting

We refer to [5] for all elements of formal language theory we use and to the cited papers from the P area for basic elements of P systems theory. We introduce here only the variant of P systems we are going to investigate.

A *rewriting P system with partial parallelism* (of degree  $m \geq 1$ ) is a construct

$$\Pi = (V, T, \mu, M_1, \dots, M_m, R_1, \dots, R_m),$$

where  $V$  is an alphabet,  $T \subseteq V$  is the terminal alphabet of  $\Pi$ ,  $\mu$  is a membrane structure with  $m$  membranes, labelled with  $1, 2, \dots, m$ ,  $M_1, \dots, M_m$  are finite languages over  $V$

associated with the regions of  $\mu$ , and  $R_1, \dots, R_m$  are finite sets of evolution rules associated with the regions of  $\mu$ .

The rules are of the form  $a \rightarrow u(\text{tar})$ , where  $a \in V, u \in V^*$ , and  $\text{tar} \in \{\text{here}, \text{out}\} \cup \{\text{in}_j \mid 1 \leq j \leq m\}$ .

The work of such a P system is defined as usual in P systems area: in each time unit, each string which can be rewritten is rewritten; the strings and the rules are localized to regions, so the strings in a given region are rewritten only by rules in that region; starting from a given configuration, we pass to another configuration; a sequence of transitions form a computation, and we consider as successful computations only the halting ones (the computations which reach a configuration where no rule can be applied); the result of a halting computation consists of all strings over  $T$  which are sent out of the system during the computation.

What is specific to our systems is the way of rewriting the strings and to pass them through membranes. Specifically, each string is rewritten in a partially parallel manner: take a string  $w$  in a region  $i$ ; for each symbol  $a \in V$  which appears in  $w$  and for which there are rules of the form  $a \rightarrow u(\text{tar})$  in the set  $R_i$ , we have to rewrite exactly one occurrence (nondeterministically chosen) of  $a$  in  $w$  (by a rule nondeterministically chosen). Assume that we rewrite in this way  $n$  symbols, by rules which have associated targets of the form  $\text{here}, \text{out}, \text{in}_j$  of several types and that there are  $n_{\text{tar}}$  targets of each type. Then, the string obtained by rewriting is sent to the membrane indicated by the target with the maximal  $n_{\text{tar}}$  (the target which appears the maximal number of times in the used rules). As usual, when several targets have the same maximal number of occurrences, then one of them is nondeterministically chosen.

We denote by  $L(\Pi)$  the language generated by a rewriting P system  $\Pi$ ; the family of all languages of this type, generated by systems with at most  $m$  membranes, is denoted by  $EPPP_m$ ; if no bound on the number of membranes is considered, then the subscript  $m$  is replaced by  $*$ .

Note that we do not use further features, such as priority relations among the rules in each region, or the possibility of controlling the membrane permeability.

### 3 The Generative Power

In the case of sequential rewriting and using priorities, a characterization of recursively enumerable ( $RE$ ) languages by means of P systems with two membranes is obtained. The proof is rather involved, see [2]. For our systems, a characterization of  $RE$  is obtained in a surprisingly easy manner, even without using a priority relation – but without an upper bound on the number of membranes.

We note that we consider two languages equal if they differ at most by the empty string.

**Theorem 1.**  $RE = EPPP_*$ .

*Proof.* Let us consider a matrix grammar  $G = (N, T, S, M, F)$  with appearance checking, in the binary normal form, that is with  $N = N_1 \cup N_2 \cup \{S, \#\}$ , and with the matrices in  $M$  of the following forms:

1.  $(S \rightarrow XA), X \in N_1, A \in N_2,$
2.  $(X \rightarrow Y, A \rightarrow x), X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*,$
3.  $(Y \rightarrow Y, A \rightarrow \#), X, Y \in N_1, A \in N_2,$
4.  $(X \rightarrow \lambda, A \rightarrow x), X \in N_1, A \in N_2, x \in T^*.$

There is only one matrix of type 1, and  $F$  consists of exactly all rules of the form  $A \rightarrow \#$  in matrices of type 3;  $\#$  is a trap-symbol, once introduced it is never removed. Each derivation ends with the use of a matrix of type 4.

Assume that the matrices of types 2 and 4 are injectively labelled with  $m_2, \dots, m_k,$  and the matrices of type 3 are labelled with  $m_{k+1}, \dots, m_n.$  We construct the partially parallel rewriting P system (of degree  $n$ )

$$\Pi = (V, T, \mu, M_1, M_2, \dots, M_n, R_1, R_2, \dots, R_n),$$

with:

$$\begin{aligned} V &= T \cup N \cup \{X_i \mid 2 \leq i \leq n\} \cup \{\#\}, \\ \mu &= [{}_1[{}_2]_2[{}_3]_3 \cdots [{}_n]_n]_1, \\ M_1 &= \{XA\}, \text{ for } (S \rightarrow XA) \text{ being the initial matrix of } G, \\ M_i &= \emptyset, 2 \leq i \leq n, \\ R_1 &= \{X \rightarrow \alpha(in_i) \mid m_i : (X \rightarrow \alpha, A \rightarrow \beta) \in M, 2 \leq i \leq k\} \\ &\cup \{X \rightarrow X_i(in_i) \mid m_i : (X \rightarrow Y, A \rightarrow \#) \in M, k+1 \leq i \leq n\} \\ &\cup \{a \rightarrow a(out) \mid a \in T\}, \\ R_i &= \{A \rightarrow x(out) \mid m_i : (X \rightarrow \alpha, A \rightarrow x) \in M, 2 \leq i \leq k\} \\ &\cup \{X_i \rightarrow Y(out), A \rightarrow \#(out) \mid m_i : (X \rightarrow Y, A \rightarrow \#) \in M, k+1 \leq i \leq n\}, \\ &\text{ for all } 2 \leq i \leq n. \end{aligned}$$

The system works as follows. Always, we have only one string in the system; initially, this is  $XA$ , for the initial matrix  $(S \rightarrow XA)$  of  $G$ . In the skin membrane, the string is rewritten either in the leftmost position, by a rule of the form  $X \rightarrow u(in_i)$ , or by a rule  $a \rightarrow a(out)$ . In the second case, the resulting string is sent out of the system and it is accepted in  $L(\Pi)$  only if it is terminal. In the former case, we have two possibilities. If we use a rule  $X \rightarrow \alpha(in_i)$  corresponding to the matrix  $m_i : (X \rightarrow \alpha, A \rightarrow x)$ , of type 2 or of type 4, then the string is sent to membrane  $i$ , where the only existing rule is  $A \rightarrow x$ . If the symbol  $A$  appears in the string, then the rule can be applied, hence exactly one occurrence of  $A$  is replaced by  $x$  and the string is sent back to the skin membrane. If there is no occurrence of the symbol  $A$  in the string, then no rewriting is possible, the string remains forever in membrane  $i$ , and we get no output. Therefore, the matrix  $m_i$  is correctly simulated. If we start by using a rule  $X \rightarrow X_i(in_i)$ , for some  $i \geq k+1$ , then we simulate the matrix  $m_i : (X \rightarrow Y, A \rightarrow \#)$ : the string is sent to membrane  $i$ , where we can rewrite for sure the symbol  $X_i$  and, if also  $A$  appears in the string, then also the rule  $A \rightarrow \#(out)$  must be used. Both rules have associated the target  $out$ , hence in both cases the string will immediately exit membrane  $i$ . If the trap-symbol  $\#$  has been introduced, then it will never be removed, hence the computation will not lead to a terminal string.

If the symbol  $\#$  is not introduced, then we continue. Therefore, we can simulate in  $\Pi$  all derivations of  $G$ , which implies the equality  $L(\Pi) = L(G)$ .  $\square$

It is an *open problem* whether or not a characterization of recursively enumerable languages can be obtained by systems with a small number of membranes, as often happens in the P area. Anyway, the partially parallel manner of rewriting looks rather powerful, systems with a small number of membranes can generate “complex” languages.

**Theorem 2.**  $EPPP_1 - CF \neq \emptyset$ .

*Proof.* Let us consider the system

$$\Pi = (\{a, a', b, b', c, c'\}, \{a, a', b, b', c, c'\}, [1]_1, \{a'b'c'\}, R_1),$$

with the following rules:

$$R_1 = \{a' \rightarrow aa'(tar), b' \rightarrow bb'(tar), c' \rightarrow cc'(tar) \mid tar \in \{here, out\}\}.$$

At each step we have to use one rule for each symbol  $a', b', c'$ . If at least two of these rules have associated the target *here*, then we have to continue. If at least two targets are equal to *out*, then the string has to be sent out of the system. Because at each step we increase by one the number of all symbols  $a, b, c$ , we obtain  $L(\Pi) = \{a^n a' b^n b' c^n c' \mid n \geq 1\}$ , which is not a context-free language.  $\square$

**Theorem 3.**  $EPPP_4 - MAT \neq \emptyset$ .

*Proof.* Consider the P system

$$\Pi = (\{a, b, X, X', X'', Z\}, \{b\}, [1[2]_2[3[4]_4]_3]_1, \{Xa\}, \emptyset, \emptyset, \emptyset, R_1, R_2, R_3, R_4),$$

with the following sets of rules:

$$R_1 = \{a \rightarrow bb(here), X \rightarrow X(here), X \rightarrow X'(in_2), X'' \rightarrow X''(in_3), X \rightarrow \lambda(out)\},$$

$$R_2 = \{a \rightarrow Z(out), X' \rightarrow X''(out)\},$$

$$R_3 = \{b \rightarrow a(here), X'' \rightarrow X''(here), X'' \rightarrow X'(in_4), X \rightarrow X(out)\},$$

$$R_4 = \{b \rightarrow Z(out), X' \rightarrow X(out)\}.$$

Assume that in the skin membrane we have a string  $Xa^n$ ; initially,  $n = 1$ . We have to use once the rule  $a \rightarrow bb(here)$  and one of the  $X$ -rules. If we use  $X \rightarrow \lambda(out)$ , then the string can remain in membrane 1 (and then it will remain forever here, hence we get no output), or can exit, but it is accepted in  $L(\Pi)$  only if it consists only of occurrences of  $b$ . If we use the rule  $X \rightarrow X'(in_2)$  and the string remains in the skin membrane, then we will get no output, because the string will remain forever here. If the string goes to membrane 2, and at least one occurrence of  $a$  is still present, the the rule  $a \rightarrow Z(out)$  must be used, and the symbol  $Z$  will never be removed. Therefore, we have to use the rules  $X \rightarrow \lambda(out)$ ,  $X \rightarrow X'(in_2)$  exactly in the moment when we rewrite by  $a \rightarrow bb(here)$  the last occurrence of  $a$  in the string. In the first case we send out the string  $b^{2n}$ , which is accepted in  $L(\Pi)$ ,

in the second case we send to membrane 2 the string  $X'b^{2n}$ , which is immediately send back to the skin membrane, by using the rule  $X' \rightarrow X''(out)$ . The only applicable rule is now  $X'' \rightarrow X''(in_3)$ , and the string is sent to membrane 3, where we replace each  $b$  by  $a$ , and only when no copy of  $b$  is present we can send the string to membrane 4, by using the rule  $X'' \rightarrow X'(in_4)$ . If any occurrence of  $b$  is present, then in membrane 4 we introduce the trap-symbol  $Z$ , otherwise the string is sent back to membrane 3, with  $X'$  replaced by  $X$ . Using the rule  $X \rightarrow X(out)$ , the string is sent to membrane 1. We have started from  $Xa^n$  and we return  $Xa^{2n}$ . The process can be iterated. Consequently, we have  $L(\Pi) = \{b^{2^n} \mid n \geq 1\}$ , and this is not a matrix language.  $\square$

The power of partially parallel rewriting P systems with a small number of membranes remains to be more closely investigated.

## 4 Final Remarks

Several further topics remain to be investigated about P systems with parallel rewriting. For instance, it seems to be of interest to consider other, weaker, ways of controlling the communication of strings, either starting from target indications of the form *here*, *out*, *in* (that is, without indicating the label of the target membrane), or changing the manner of synthesizing the target from the indications of the rules used for rewriting a string; at the limit, the communication can be done nondeterministically, which probably will decrease the power of the systems. It also remains to systematically investigate the systems where a full parallelism is used – in comparison with families in the L area, the natural test bed for such a case.

## References

- [1] S. N. Krishna, R. Rama, On the power of P systems with sequential and parallel rewriting, *Intern. J. Computer Math.*, 77, 1-2 (2000), 1–14.
- [2] Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143 (see also *Turku Center for Computer Science-TUCS Report No 208*, 1998, [www.tucs.fi](http://www.tucs.fi)).
- [3] Gh. Păun, Computing with membranes. An introduction, *Bulletin of the EATCS*, 67 (Febr. 1999), 139–152.
- [4] Gh. Păun, G. Rozenberg, A. Salomaa, Membrane computing with external output, *Fundamenta Informaticae*, 41, 3 (2000), 259–266,
- [5] G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages*, 3 volumes, Springer-Verlag, Berlin, 1997.