ROMANIAN JOURNAL OF INFORMATION SCIENCE AND TECHNOLOGY
Volume **10**, Number 2, 2007, 137–144

On the Power of P Systems with Parallel Rewriting and Conditional Communication

K. G. SUBRAMANIAN[1], S. HEMALATHA[2],
C. SRI HARI NAGORE[2], M. MARGENSTERN[3]

[1]Department of Computer Science,
Sri Muthukumaran Institute of Technology,
Chennai 600 069, India
E-mail: kgsmani1948@yahoo.com

[2]Department of Applied Sciences and Humanities,
Madras Institute of Technology, Anna University,
Chennai 600 044, India

[3]LITA, Universite Paul Verlaine-Metz,
Ile du Saulcy, 57045 Metz Cedex, France

Abstract. In this paper we consider the application of rules in the regions of a parallel rewriting P system with the feature of unique parallelism and conditional communication. We call the resulting P system parallel rewriting P system with conditional communication. We examine the generative power of this P system by comparing it with other string generating devices.

1. Introduction

Inspired from the behavior of living cells, a new computability model, called P system [3, 4] was introduced by Păun which has proved to be a rich theoretical frame work to study many computational problems besides giving a new impetus to formal language theory. One of the several kinds of P systems is a rewriting P system with string objects and rewriting rules in its regions whose computational power has been investigated extensively. Extending the application of the rules in a region from sequential to parallel rewriting, parallel rewriting P systems were introduced in [1] with different kinds of parallelism methods and the computational power was analyzed with respect to Lindenmayer systems [5]. Several problems related to these P systems have been investigated in the literature. On the other hand among many variations of the

rewriting P systems, conditional communication is an interesting feature introduced in [2] resulting in rewriting P systems with conditional communication.

In this paper we consider application of rules in the regions of the rewriting P systems with the simplest type of conditional communication, namely empty or symbol conditions and with the feature of unique parallelism. We examine the generative power of the resulting P system, called parallel rewriting P system with conditional communication by comparing with other string-generating devices.

2. Preliminaries

We first recall the Lindenmayer systems [5] that are needed in the sequel, the unique parallelism mode used in parallel rewriting P systems [1] and the rewriting P systems with conditional communication [2]. We refer [5, 6] for notions of formal language theory.

A 0L system is a construct G =(V,w,P) where V is an alphabet, $w \in V^+$ is the axiom and P is a finite set of rules of the form $a \to v$, with $a \in V$, $v \in V^*$, such that for each $a \in V$ there is at least one rule $a \to v$ in P. For $w_1, w_2 \in V^*$, $w_1 \Rightarrow w_2$ if $w_1 = a_1...a_n$, $w_2 = v_1...v_n$, for $a_i \to v_i \in P, 1 \leq i \leq n$. The language generated by G is $L(G) = \{x \in V^* | w \Rightarrow^* x\}$. If for each $a \in V$ there is exactly one rule $a \to v$ in P, then G is said to be deterministic. If $T \subseteq V$, consists of terminal symbols then G is said to be extended and $L(G) = \{x \in T^* | w \Rightarrow^* x\}$.

The family of languages generated by 0L systems, extended 0L systems, extended and deterministic 0L systems, extended tabled 0L systems, extended deterministic tabled 0L systems are respectively denoted by 0L, E0L, ED0L, ET0L, EDT0L. In fact in an ET0L System [5], the set of rewriting rules is divided into subsets of rules, called tables of rules. In one step, only the rules from a table (nondeterministically chosen) can be applied, and these rules must be applied in parallel on all occurrences of all symbols in the string rewritten, but not necessarily following the order of the rules appearing in the table. Moreover, if some rules in the chosen table are defined over symbols not in the string, or if the number of rules in the table exceeds the length of the string, then we skip those rules without forbidding the application of the entire table.

Different modes of parallel rewriting have been introduced in the study of parallel rewriting P systems [1].

We refer to [2, 4] for the notion of rewriting P systems with conditional communication in unique parallelism mode. But we restrict ourselves to only two modes of permitting and forbidding conditions, namely, empty and symbol checking conditions.

3. Parallel Rewriting P Systems with Conditional Communication

We now introduce a new kind of rewriting P System, called parallel rewriting P system with conditional communication, in which rewriting of strings is in parallel mode as in [1], and communication is conditional as in [2]. We use only unique parallelism mode and empty and symbol checking conditions.

An extended parallel rewriting P system with conditional communication of degree n is defined by the construct
$\prod = (V, T, \mu, M_1, M_2, ..., M_n, (R_1, P_1, F_1), ..., (R_n, P_n, F_n), (n, d))$
where V is the alphabet of the system; $T \subseteq V$ is the terminal alphabet; μ is the membrane structure with n membranes and depth d, which are injectively labelled by numbers in the set $\{1, ..., n\}$. The skin membrane is always labelled as 1; M_i's are finite languages over V, representing strings initially present in the regions $1, 2, ..., n$ of the system; R_i's are finite sets of context-free rules of the form $a \rightarrow \alpha$, with $a \in V$, $\alpha \in V^*$, associated with the regions of μ; The rules are used in the regions in the unique parallelism mode as follows:

A rewriting step with unique parallelism (U) involves substitution of all occurrences of exactly one symbol according to exactly one rule, which is nondeterministically chosen between all rules that can be applied to that symbol. That is, given a string $w = x_1 a x_2 a ... x_n a x_{n+1}$ with $x_i \in (V \backslash \{a\})^*$, $\forall\ i = 1, 2, ..., n+1$, and one context-free rule $r : a \rightarrow \alpha$,we obtain the string $w' = x_1 \alpha x_2 \alpha ... x_n \alpha x_{n+1}$ in one parallel rewriting step.

P_i's and F_i's are permitting and forbidding conditions associated with the regions $1, 2, ..., n$ of the forms empty or symbol checking given as follows:

1. empty: no restriction is imposed on strings; they either exit the current membrane or enter any of the directly inner membranes freely; we denote an empty permitting condition by (true,α), $\alpha \in \{in, out\}$, and an empty forbidding condition by (false,notα), $\alpha \in \{in, out\}$.

2. symbols checking: each P_i is a set of pairs (a, α), $\alpha \in \{in, out\}$ for $a \in V$ and each F_i is a set of pairs (b, notα), $\alpha \in \{in, out\}$ for $b \in$ V; a string w can go to a lower membrane only if there is a pair $(a, in) \in P_i$ with $a \in alph(w)$, and for each $(b, notin) \in F_i$, $b \notin alph(w)$; similarly the string goes out of membrane i , if there is at least one pair $(a, out) \in P_i$ and $(b, notout) \in F_i$ for all $b \notin alph(w)$.

A system is said to be non-extended if $V = T$.

The transitions in the system are defined in the following way. In each region, each string which can be rewritten is rewritten by a rule from that region. The rule to be applied and the non-terminal it rewrites are nondeterministically chosen; Each string obtained in this way is checked against the conditions P_i, F_i from that region. If it fulfills the requested conditions, then it will be immediately sent out of the membrane or to an inner membrane if any exists; if it fulfills both in and out conditions, then it is sent either out of the membrane or to an inner membrane (nondeterministically choosing any of the available inner membranes). If a string does not fulfill any condition, or it fulfills only "in" conditions and there is no inner membrane , then the string remains in the same region. A string which is rewritten and a string which is sent to another membrane is "consumed", no copy of it is available in the next step in the same membrane. If a string cannot be rewritten then it is directly checked against the communication conditions, and, as above, it leaves the membrane (or remains inside forever) depending on the result of this checking. That is rewriting has priority over communication.

As usual, a sequence of transitions forms a computation and the result of a halting computation is the set of strings over T(terminal symbols) sent out of the system during the computation. A computation which never halts gives no output. The strings that contain only terminal symbols and which are sent out of the skin membrane contribute to the language of the system. The language generated by a system $\prod$ in this way is denoted by $L(\prod)$.

We denote by $EUPRP_n^m(\alpha, \beta)$ the family of languages generated by extended rewriting P systems of degree n and depth m. Here degree n is the total number of membranes in the system; Depth d is the maximum number of membranes in the the nesting of membranes in the whole system . $\alpha, \beta \in \{empty, symbol\}$ denote the permitting and forbidding conditions for communication. If the system is non-extended then we write $URP_m^n(\alpha, \beta)$. If there is no upper bound on the number of membranes then we replace m by $*$.

Remark 1. Although we have allowed a finite number of strings in each membrane initially, we require only one string initially in a region in all the results in this paper.

Example 1. *Consider the non-context free language* $L_1 = \{a^{2^n} | n \geq 0\}$. *We construct a P system* $\prod_1 = (V, T, [_1]_1, \{Xa\}, (R_1, P_1, F_1), (1, 1))$,
$V = \{X, a\}, T = \{a\}$
$R_1 = \{X \rightarrow \lambda, a \rightarrow aa\}$
$P_1 = \{(true, out)\}$
$F_1 = \{(X, notout)\}$.
then $L(\prod_1) = L_1$.

Initially, the string Xa is in the skin membrane, If the rule $X \rightarrow \lambda$ is applied, the resulting string a, is sent out of the membrane. On the other hand when the rule $a \rightarrow aa$ is applied, a is doubled, The string can leave the membrane only if there exists no X in the string due to the forbidding condition. Hence only after applying the rule $X \rightarrow \lambda$ strings are sent out. Again on applying the rule $a \rightarrow aa$, Every a in the string is further doubled and checked by the conditions and at any stage the rule $X \rightarrow \lambda$ can be applied to send the string out of the membrane which contributes to the language L_1. Thus strings of the form a^{2^n} are generated.

Theorem 1. $EDT0L \subseteq EUPRP_*^2(empty, symbol)$

Proof. Consider an EDT0L system $G = (V, T, w, T_1, T_2, ..., T_n)$ where V is the alphabet, $T \subseteq V$ is the terminal alphabet and each table T_i has exactly one rule of the form $A \rightarrow x$, for every $A \in V$ and for some $x \in V^*$. We construct an equivalent parallel rewriting P system with permitting conditions in empty mode and forbidding conditions in symbol mode and rewriting in unique parallelism mode. The required P system is:
$\prod = (V', T, \mu, M_1, M_2, ..., M_{n+2}, (R_1, P_1, F_1), (R_2, P_2, F_2), ..., (R_{n+2}, P_{n+2}, F_{n+2}),$
$(n + 2, 2))$ where $V' = V \cup \{X\} \cup \{\overline{a}_j | a_j \in V\}$ with X and each $\overline{a}_j$ being new symbols; $\mu = [_1[_2[_3 ... [_{n+2}]_{n+2}]_1$ consisting of n+1 membranes placed inside the skin membrane; $M_1 = \{Xw\}$, $M_i = \phi, i = 2, ..., n + 2$. The skin membrane consists of the following rules:

$R_1 = \{X \to X, X \to \lambda\}$
$P_1 = \{(true, in), (true, out)\}$
$F_1 = \{(X, notout)\}$.
Each inner membrane m_i, $(2 \leq i \leq n+1)$ simulates the table $T_i, i = 2, ..., n+1$. The rules are given below:
$R_i = \{A \to \overline{x} | A \to x \in T_i, A \in V$ and $\overline{x}$ is obtained from x by replacing each symbol a in x by $\overline{a}\}$
$P_i = \{(true, out)\}$
$F_i = \{(A, notout) | A \in V\}$.
Finally, the rules in m_{n+2} are as follows:
$R_{n+2} = \{\overline{a} \to a | a \in V\}$
$P_{n+2} = \{(true, out)\}$
$F_{n+2} = \{(\overline{a}, notout) | a \in V\}$. □

The system works in the following way. Initially, the string Xw is in the skin membrane, If the rule $X \to \lambda$ is used, the resulting string w is sent out of the membrane. If $X \to X$ is used the string Xw is sent to any of the inner membranes nondeterministically chosen. If it enters the membrane m_i, for some i, $2 \leq i \leq n+1$, then the rules of R_i are used in unique parallelism mode. The rewritten string is not sent out of membrane m_i until all the symbols are changed into barred symbols by the application of the rules. Note that there is exactly one rule $A \to \overline{x}$ for every $A \in V$ in R_i and so all the A's in the string rewritten are replaced by $\overline{x}$ due to unique parallelism and this process is repeated for other symbols in the string rewritten, until all the symbols in the resulting string are barred symbols, due to the forbidding conditions in F_i. This captures the rewriting in an EDTOL system.

Once the string is sent out of the membrane m_i, into the skin membrane, the application of the rule $X \to X$ can send it again to any of the inner membranes but when it enters the membrane m_{n+2}, all the barred symbols are changed back into non-barred symbols and sent back to the skin membrane from which the application of the rule $X \to \lambda$ will send it out of the skin membrane and this string will be collected in the language generated. Any premature application of the rule $X \to \lambda$, causing the string to be sent out of the skin membrane will not contribute to the language generated if the string contains nonterminal symbols.

Remark 2. It is not known whether the inclusion is proper in Theorem 1.

Theorem 2. $FIN = [E]UPRP_1^1(empty, empty)$

Proof. When we have only one membrane, each string can evolve only once under unique parallelism mode, because it has to exit the system after one evolution step, hence the inclusion $[E]UPRP_1^1(empty, empty) \subseteq FIN$ is obvious.
Also the converse inclusion holds: any given finite language $L \subset V^*$,
$L = \{x_1, x_2, ..., x_n\}$, is generated by the system
$\prod_1 = (V \cup \{S\}, V, [_1]_1, \{S\}, (\{S \to x_i | 1 \leq i \leq n\}, \{(true, out)\}\{(false, notout)\}))$.
Here $S \notin V$. □

Theorem 3. $[E]UPRP_1^1(empty, symbol) - CF \neq \phi$
The result follows from example 1.

Theorem 4. $UPRP_2^2(empty, empty) - CF \neq \phi$

Proof. Consider the non-context free language $L_3 = \{a^{2^n} b^{2^n} | n \geq 0\}$.
We construct a P system
$\prod_3 = (V, T, [_1[_2]_2]_1, \{X\}, \phi, (R_1, P_1, F_1), (R_2, P_2, F_2), (2, 2))$,
$V = \{X, a, b\}, T = \{a, b\}$
$R_1 = \{X \rightarrow ab, a \rightarrow aa\}$
$P_1 = \{(true, in)(true, out)\}$
$F_1 = \{(false, notin), (false, notout)\}$.
$R_2 = \{b \rightarrow bb\}$
$P_2 = \{(true, out)\}$
$F_2 = \{(false, notout)\}$.
then $L(\prod_3) = L_3$. Initially the string X is in the skin membrane. If the rule $X \rightarrow ab$ is applied, the string can be sent out collected in the language or can be sent an inner membrane where b is doubled and sent back to membra1, Here a is doubled giving $aabb$. This can be sent out or sent to the inner membrane. The process repeats. □

Theorem 5. $EUPRP_1^1(empty, symbol) \subset EUPRP_2^2(empty, symbol)$

Proof. Consider the non-context free language $L_4 = \{a^{2^n} b^{2^n} c^{2^n} | n \geq 0\}$.
We construct a P system
$\prod_4 = (V, T, [_1[_2]_2]_1, \{Xabc\}, \phi, (R_1, P_1, F_1), (R_2, P_2, F_2), (2, 2))$,
$V = \{X, a, b, c, \overline{a}, \overline{b}, \overline{c}\}, T = \{a, b, c\}$
$R_1 = \{X \rightarrow X, X \rightarrow \lambda, \overline{a} \rightarrow a, \overline{b} \rightarrow b, \overline{c} \rightarrow c\}$
$P_1 = \{(true, in)(true, out)\}$
$F_1 = \{(\overline{a}, notin), (\overline{b}, notin), (\overline{c}, notin)\}$.
$R_2 = \{a \rightarrow \overline{aa}, b \rightarrow \overline{bb}, c \rightarrow \overline{cc}\}$
$P_2 = \{(true, out)\}$
$F_2 = \{(a, notout), (b, notout), (c, notout)\}$. then $L(\prod_4) = L_4$.
The Language L_4 is in $EUPRP_2^2(empty, symbol)$ but cannot be generated by $EUPRP_1^1$ $(empty, symbol)$. Note that every word in L_4 has for some $n \geq 0$, 2^n, a's followed by, 2^n, b's and 2^n, c's. If there is only one membrane all the rules are to be kept in this membrane, when each a is doubled by unique parallelism mode, $b's$ and $c's$ need not be equally doubled and vice versa. This will give rise to words not in L_4. So one membrane is not enough to generate L_4. □

Theorem 6. $LIN \subset EUPRP_1^1(empty, symbol)$

Proof. Since in a derivation in a linear grammar, there is atmost one nonterminal in any sentential form, the application of rules is indeed in unique parallelism mode. All the rules of a linear grammar can be kept in a single membrane with empty permitting condition and symbol forbidding condition for every nonterminal in the linear grammar. The language $L_5 = \{a^n c b^n a^m d b^m | m, n \geq 0\}$ is a non- linear language. Consider a P system
$\prod_5 = (V, T, [_1]_1, XAB, (R_1, P_1, F_1))$
$V = \{X, a, b, c, d, A, B\}, T = \{a, b, c, d\}$,
$R_1 = \{A \rightarrow aAb, B \rightarrow aBb, A \rightarrow c, B \rightarrow d, X \rightarrow X, X \rightarrow \lambda\}$ $P_1 = \{(true, out)\}$
$F_1 = \{X, notout\}$ then $L(\prod_5) = L_5$. Hence the theorem. □

Theorem 7. For $n \geq 3$, $EUPRP^2_{n-1}(empty, symbol) \subset EUPRP^2_n(empty, symbol)$

Proof. Consider a P System $\prod^{'} = (V \cup \{X_i : 1 \leq i \leq n-1\} \cup \{\overline{a}_j : 1 \leq j \leq n-1\}, T = \{a_1, a_2, ..., a_n\}, [_1[_2]_2...[_{n+1}]_{n+1}]_1,$
$Xa_1a_2...a_n, (R_1, P_1, F_1), ..., (R_{n+1}, P_{n+1}, F_{n+1}), (n+1, 2))$
$R_1 = \{X_{i-1} \rightarrow \lambda, a_{i-1} \rightarrow \overline{a}_{i-1}, X \rightarrow X_{i-1} : 2 \leq i \leq n+1\}$
$P_1 = \{(true, out)(true, in)\}$
$F_1 = \{(a_j, notin), (X, notin) : 2 \leq j \leq n\}$
The elementary membranes inside the skin membrane contain the rules as given below:
For $2 \leq i \leq n+1$,
$R_i = \{\overline{a}_i \rightarrow a_i a_i\} \cup \{\overline{a}_j \rightarrow a_j : \ 1 \leq j \leq n, j \neq i\}$
$P_i = \{(true, out)\}$
$F_i = \{(X_j, notout) : 1 \leq j \leq n \ j \neq i\} \cup \{(\overline{a}_j, notout) : 1 \leq j \leq n\}$
$\prod^{'}$ generates the language $\{a_1^{2^m} a_2...a_n | m \geq 1\} \cup \{a_1 a_2^{2^m} ...a_n | m \geq 1\} \cup ... \cup \{a_1 a_2 ...a_n^{2^m} | m \geq 1\}$. In each of the strings in the language, only one symbol among the n symbols is raised to a power of 2 whereas the remaining symbols are not increased in number. Hence it is clear that this language cannot be generated with lesser number of membranes in depth 2. □

Remark 3. For every extended P system $\prod$ in $EUPRP^n_m(\alpha, \beta)$ generating the language L, there exists an equivalent non-extended P system $\prod^{'}$ in $UPRP^{n+1}_{m+1}$ (α, β) generating the same language L. This is due to the fact that any extended system can be simulated by a non-extended one by just considering an additional external membrane with a condition of the form $\{(a, notout) | a \in V - T\}$.

4. Conclusion

A beginning has been made here to combine parallel rewriting with conditional communication restricting to unique parallelism mode and empty and symbol checking conditions. Other parallelism modes and conditions can be considered, besides examining depth d of the system for $d \geq 2$.

Acknowledgements. The authors are thankful to the referees for their useful comments. The author K. G Subramanian is grateful to Prof. M. Margenstern, University of Paul Verlaine-Metz, France, for his invitation and the University for the support to visit the university during May-June, 2006. Part of this work was done during this visit.

References

[1] BESOZZI D., MAURI G., ZANDRON C., *Parallel Rewriting P Systems without Target Conflicts*, Lecture Notes in Computer Science, pp. 119–133, **2597**, 2003.

[2] BOTTONI P., LABELLA A., MARTIN-VIDE C., PĂUN GH., *Rewriting P Systems with Conditional Communication*, Lecture Notes in Computer Science, Springer, Berlin, pp. 325–353, **2300**, 2002.

[3] PĂUN GH., *Computing with Membranes*, Journal of Computer and System Sciences, pp. 108–143, **61**, 1, 2000, and Turku Center for Computer Science - TUCS Report No. 208, 1998.

[4] PĂUN GH., *Membrane Computing, An Introduction*, Springer, 2002.

[5] ROZENBERG G., SALOMAA A., eds., *Mathematical Theory of L Systems*, Academic Press, New York, 1980.

[6] ROZENBERG G., SALOMAA A., eds., *Handbook of Formal Languages*, Springer-Verlag, Heidelberg, 1997.