ISSN 0975-3303
Mapana J Sci, **10**, 2(2011),63-74
https://doi.org/10.12725/mjs.19.6

Parallel Communicating String - Graph P System

Meena Parvathy Sankar* and N.G. David**

Abstract

The concept of parallel communicating grammar systems generating string languages is extended to string-graph P systems and their generative power is studied. It is also established that for every language L generated by a parallel communicating grammar system there exists an equivalent parallel communicating string-graph P system generating the string-graph language corresponding to L.

Keywords: String Grammar, PC Grammar System, Membrane Computing

1. Introduction

Membrane computing is an area of computer science aiming to abstract computing ideas and models from the structure and the functioning of living cells, as well as from the way the cells are organized in tissues or higher order structures, also known as P System.

The membrane structure of a P system is a hierarchical arrangement of membranes, embedded in a skin membrane - the one which separates the system from its environment. A membrane without any membrane inside is called elementary. Each membrane defines a region. A membrane structure is pictorially represented by an Euler-Venn diagram or it can be

* Madras Christian College, Chennai – 600 059; meenaparvathysankar@gmail.com
**Madras Christian College, Chennai – 600 059; ngdmcc@gmail.com.

Received: March 2011; Revised: August 2011

mathematically represented by a tree, or by a corresponding string of matching parenthesis.

Each region consists of a multiset of objects and a set of evolution rules. The objects are represented by symbols from a given alphabet. An evolution rule from region r is in general of the form $ca \rightarrow cb_{in_j} d_{out} d_{here}$, and it says that a copy of the object a, in the presence of a copy of the catalyst c (this is an object which is never modified, it only assists the evolution of other objects), is replaced by a copy of the object b and two copies of the object d. Moreover, the copy of b has to immediately enter the inner membrane of region r labeled by j (hence to enter region j), a copy of object d is sent out through the membrane of region r, and a copy of d remains in region r.

In natural languages, there occur phenomena like multiple agreements, crossed agreements and replication. These aspects are represented in formal language theory by the three languages $K_1 = \{a^n b^n c^n / n \geq 1\}$, $K_2 = \{a^n b^m c^n d^m / m, n \geq 1\}$ and $K_3 = \{ww / w \in \{a, b\}^+\}$ respectively. It is known that these languages can be generated by grammar systems with regular grammar components.

A grammar system is a set of grammars working together, according to a specified protocol, to generate one language. There are two basic classes of grammar systems: sequential and parallel.

A cooperating distributed (CD) grammar system is a sequential grammar system, in which all component grammars have a common sentential form. Whereas, a parallel communicating (PC) grammar system is parallel in nature and, each component has its own sentential form. Within each time unit each component applies a rule, rewriting its own sentential form. The key feature of a PC grammar system is its 'communications through queries' mechanism. Special (query) symbols are provided with each system pointing to a component of the system. When a component i introduces the query symbol Q_j, the current sentential form of the component j will be sent to the component i, replacing all the occurrences of Q_j. The grammar j resumes rewriting beginning again from its axiom.

On the other hand, string-graphs (or linear graphs or labeled paths) represent strings over an alphabet and hyper-graph grammars generating string-graph languages are well studied in the literature [4]. In this paper, we define string-graph P system as an outcome of string-graphs, PC grammar system and P system and provide some results.

2. Basic Concepts

In this section, we review the concepts of grammar systems, membrane systems, and string graph grammars.

2.1 Grammar System

In this section, we recall the concepts related to PC grammar system [1, 2].

Definition 2.1.1 A PC grammar system of degree $n \geq 1$, is an $(n+3)$ tuple $\Gamma = (N, K, T, (S_1, P_1), (S_2, P_2), \ldots, (S_n, P_n))$, where N is a non terminal alphabet, T is a terminal alphabet, $K = \{Q_1, Q_2, \ldots, Q_n\}$ (the sets N, K, T are mutually disjoint), P_i is a finite set of rewriting rules over $V_\Gamma = N \cup K \cup T$ and $S_i \in N$, for all $1 \leq i \leq n$.

The sets P_i, $1 \leq i \leq n$, are called the components of the system, and the elements $Q_1, Q_2, \ldots, Q_n$ of K are called query symbols; the index i of Q_i points to the i^{th} component P_i of Γ.

Given a PC grammar system $\Gamma = (N, K, T, (S_1, P_1), (S_2, P_2), \ldots, (S_n, P_n))$, for two n-tuples $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$, with $x_i, y_i \in V_\Gamma^*$, $1 \leq i \leq n$, where $x_1 \notin T^*$, we write $(x_1, x_2, \ldots, x_n) \Rightarrow (y_1, y_2, \ldots, y_n)$, if one of the following cases holds:

(i) For each i, $1 \leq i \leq n$, $|x_i|_k = 0$, $1 \leq i \leq n$, and for each i, $1 \leq i \leq n$, we have either $x_i \Rightarrow y_i$ by a rule in P_i, or $x_i \Rightarrow y_i \in T^*$.

(ii) There is i, $1 \leq i \leq n$ such that $|x_i|_k > 0$. Let for each i, $x_i = z_1 Q_{i_1} z_2 Q_{i_2} \ldots z_t Q_{i_t} z_{t+1}$, $t \geq 1$, for $z_j \in (N \cup T)^*$, $1 \leq i \leq t+1$. If $|x_{i_j}|_k = 0$, for all j, $1 \leq j \leq t$, then $y_i = z_1 x_{i_1} z_2 x_{i_2} \ldots z_t x_{i_t} z_{t+1}$ and $y_{i_j} = S_{i_j}$, $1 \leq j \leq t$. If for some j, $1 \leq j \leq t$, $|x_{i_j}|_k \neq 0$, then $y_i = x_i$. For all i, $1 \leq i \leq n$, such that y_i is not specified above, we have $y_i = x_i$.

A PC grammar system deadlocks in two cases:

(1) when no query symbol is present, a component x_i of the current configuration $(x_1, x_2, \ldots, x_n)$ is not a terminal string and no rule of p_i can be applied to it. (This can happen both after a rewriting and after a communication), and

(2) when a circular query appears: P_{i_1} introduces Q_{i_2}, P_{i_2} introduces Q_{i_3}, and so on until $P_{i_{k-1}}$ introduces Q_{i_k}, and P_{i_k} introduces Q_{i_1} no derivation is possible (the communication has priority), but no communication is possible (only strings without occurrences of query symbols are communicated).

Definition 2.1.2 The language generated by a PC grammar system Γ is $L(\Gamma) = \{x \in T^* / (S_1, S_2, \ldots, S_n) \Rightarrow^* (x, \alpha_2, \ldots, \alpha_n), \alpha_i \in V_{\Gamma^*}, 2 \leq i \leq n\}$.

Let $\Gamma = (N, K, T, (S_1, P_1), (S_2, P_2), \ldots, (S_n, P_n))$ be a PC grammar system. If only P_1 is allowed to introduce query symbols (formally, $P_i \subseteq (N \cup T)^* \times (N \cup T)^*$ *for* $2 \leq i \leq n$), then we say that Γ is a centralized PC grammar system, otherwise Γ is non-centralized,

A PC grammar system is said to be returning (to axiom) if, after communicating, each component which has sent its string to another component returns to axiom.

A PC grammar system is non-returning if after communicating, the component does not return to its axiom, but rather it continues to process the current string.

A PC grammar system is said to be regular, linear, context-free, context-sensitive, λ-free, etc. when the rules in its components are of the corresponding types.

2.2 Membrane System

In this section, we review the notions of P system and Rewriting P system [3].

Definition 2.2.1 A membrane system called a P system is a construct $\Pi = (V, T, C, \mu, w_1, w_2, \ldots, w_m, (R_1, \rho_1), \ldots, (R_m, \rho_m))$, where V is an alphabet –its elements are called objects; $T \subseteq V$ (the output alphabet); $C \subseteq V - T$ (catalysts); μ is a membrane structure

consisting of m membranes injectively labeled by the elements of a given region of m labels; m is called the degree of Π; w_i, $1 \le i \le m$, are strings which represents multisets over V associated with the regions $1, 2, \ldots, m$ of μ.

An evolution rule is a pair (u, v) written as $u \to v$, where u is a string over V and $v = v'$ *or* $v = v'\delta$, where v' is a string over $\{a_{here}, a_{out}, a_{in_j} / a \in V, 1 \le j \le m\}$, and δ is a special symbol not in V. The length of u is called the radius of the rule $u \to v$. R_i, $1 \le j \le m$, are finite sets of evolution rules over V each R_i is associated with the region i of μ; ρ_i is a partial order relation over R_i, called a priority relation.

Definition 2.2.2 A rewriting P system of degree $n \ge 1$ is a construct $\Pi = (V, \mu, M_1, \ldots, M_n, (R_1,\rho_1), \ldots, (R_n,\rho_n), i_0)$, where V is an alphabet, μ is a membrane structure consisting of n membranes (labeled with $1, 2, \ldots, n$), M_i, $1 \le i \le n$ are finite languages over V, R_i, $1 \le i \le n$, are finite sets of context free evolution rules. The rules of the form $X \to v(tar)$, where X *is a symbol of* V *and* $v = v'$ *or* $v = v'\delta$, *or* $v = v'\tau$, where v' is a string over V and δ, τ are symbols not in V. $tar \in \{here, out\} \cup \{in_m \mid 1 \le m \le n\}$, represents the target membrane, (i.e.) the membrane where the string produced with this rule will go. ρ_i, $1 \le i \le n$, are partial order relations over R_i representing priorities among rules , i_0 is the output membrane.

The membrane structure μ and the finite languages $M_1, \ldots, M_n$ constitute the initial configuration of the system. Membranes can have two different thickness levels. We can pass from one configuration to another by applying in parallel the evolution rules to all strings which can be rewritten, obeying the priority relations. The set of strings generated are collected in a designated membrane, the output one, and hence the language generated by Π .

2.3 String Graph Grammar

In this section, we provide the necessary definitions related to the study of string graph grammar. For the unexplained notions, we refer to [4].

Definition 2.3.1 Let C be an arbitrary, but fixed set, called set of labels (or colors). A (directed hyperedge-labeled) hypergraph over C is a system (V, E, s, t, l) where V is a finite set of nodes (or vertices), E is a finite set of hyperedges $s : E \to V^* \text{ and } t : E \to V^*$ are two mappings assigning a sequence of sources s(e) and a sequence of targets $t(e)$ to each $e \in E$, and $l : E \to C$ is a mapping labeling each hyperedge.

For $e \in E$, the set of nodes occurring in the sequence $att(e) = s(e).t(e)$ is called the set of attachment nodes of e and is denoted by $att(e)$.

A hyperedge $e \in E$ is called an (m, n) – edge for some $m,n \in \mathbf{N}$ if $|\, s(e) \,| = m$ and $|\, t(e) \,| = n$. The pair (m, n) is the type of e, denoted by type(e).

Definition 2.3.2 A multi-pointed hypergraph over C is a system $H = (V, E, s, t, l, begin, end)$ where (V, E, s, t, l) is a hypergraph over C and $begin, end \in V^*$, components of H are denoted by V_H, E_H, s_H, t_H, l_H, $begin_H$, end_H, respectively. The set of all multi-pointed hypergraphs over C is denoted by $\mathcal{H}_C$.

For $H \in \mathcal{H}_C$, the set of nodes occurring in the sequence $ext_H = begin_H.\, end_H$ is called the set of external nodes of H and is denoted by EXT_H.

$H \in \mathcal{H}_C$ is said to be an (m, n) – hypergraph for some $m, n \in N$ if $|\, begin_H \,| = m$ and $|\, end_H \,| = n$. The pair (m, n) is the *type of H*, denoted by $type(H)$.

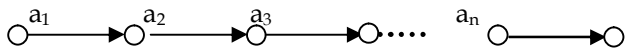
An (m, n) hypergraph H over C is said to be an (m, n) graph if $|\, V_H \,| \geq 1$ and all hyperedges of H are $(1, 1)$ edges. The set of all $(1, 1)$ graphs over C is denoted by $\mathcal{G}_C$.

Definition 2.3.3 $H \in \mathcal{H}_C$ is said to be a singleton if $V_H = EXT_H$ and $|\, E_H \,| = 1$. In this case, $e(H)$ refers to the single hyperedge of H and $l(H)$ to its label.

A singleton H with $E_H = \{e\}$, $s_H(e) = begin_H$, and $t_H(e) = end_H$ is said to be a handle. If $l_H(e) = A$, $type(e) = (m, n)$ for some $m, n \in N$, and the nodes in $ext_H = begin_H.end_H$ are pairwise distinct, then H is said to be an (m,n) – handle induced by A and is denoted by $(A, (m, n))^\bullet$ or $A(m, n)^\bullet$

Definition 2.3.4 A (1, 1) hypergraph H over C is called a string graph if it is of the form $H = (\{v_0, v_1, \ldots, v_n\}, \{e_1, \ldots, e_n\}, s, t, l, (v_0, v_n))$ where $v_0, v_1, \ldots, v_n$ are pair wise distinct, $s(e_i) = v_{i-1}$, and $t(e_i) = v_i$ for $i = 1, 2, \ldots, n$. If $w = l(e_1), \ldots, l(e_n)$, then the *(1, 1)* hypergraph is called string graph induced by w and it is denoted by w*.

A string graph of the form



provides a unique graph representation of the string $a_1 a_2 \ldots a_n \in C^+$.

Definition 2.3.5 A hypergraph language L is said to be a string-graph language if all $H \in L$ are string graphs. The class of all string-graph languages is denoted by L_{string}.

Definition 2.3.6 A hyperedge replacement grammar $HRG = (N, T, P, Z)$ is said to be context–free string–graph grammar if the right hand sides of the productions in P as well as the axiom Z are string graphs. The class of all string-graph languages generated by a context-free string–graph grammar is denoted by *CFL*.

Fact 2.3.7 [4]

The string graph language $L = \{(a^n b^n c^n)^\bullet \mid n \geq 1\}$ cannot be generated by a hypergraph grammar of order < 4.

3. String-graph P system

In this section, we extend the concept of parallel communicating grammar system generating string languages to string-graph P system and study its generative power.

Definition 3.1 A string-graph P system is a construct $\Pi = (N, K, T, \mu, w_1, w_2, \ldots, w_m, R_1, R_2, \ldots, R_m)$ where N is a set of nonterminals; T is a set of terminals, distinct from N; $K = \{Q_1, Q_2, \ldots, Q_n\}$, is a distinct subset of N, called the set of query symbols; μ is a membrane structure consisting of m membranes injectively labeled by the elements of a given region of m labels; w_i is a string-

graph associated with the region i, $1 \leq i \leq m$ of μ; R_i are the string-graph rewriting rules associated with the region i, $1 \leq i \leq m$ of μ.

Theorem 3.1 For every language L generated by a PC grammar system there exists an equivalent parallel communicating string-graph P system generating the string-graph language corresponding to L.

Proof Consider a PC grammar system $\Gamma = (N, K, T, (S_1, P_1), (S_2, P_2),\ldots, (S_n, P_n))$ with degree $n \geq 1$, generating a language L. We give below the method of constructing an equivalent parallel communicating string-graph P system generating the string-graph language $L\bullet$.

The required PC string-graph P system is $\prod = (N, K, T, \mu, w_1, w_2, \ldots, w_n, R_1, R_2, \ldots, R_n)$, where N, T, K are same as in Γ, μ is the membrane structure of the regions $1, 2, \ldots, n$: $[_1[_2]_2[_3]_3\ldots]_1$. $w_i = (S_i)\bullet$, the initial string-graph present in the region i and R_i contains the string-graph rewriting rules corresponding to the string rules of P_i in Γ for $i = 1, 2, \ldots, n$.

Initially, the derivation starts with the axiom S_i, in the region i, and continuing the derivation with rules of R_i, for each $i = 1, 2, \ldots, n$. Only the skin membrane produces the query symbols using the rules in R_1. When the skin membrane produces the query symbol Q_i, for some i, it performs the communication step of replacing the query symbol Q_i by the communicated string which is derived in region i.

After performing a sequence of steps, and if there is no query symbol present in the skin membrane the corresponding string-graph is sent out of the membrane and collected in the language.

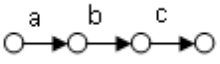
It can be easily seen that corresponding to a successful derivation in Γ, there exists a corresponding successful derivation in the string-graph P system $\prod$.

Example 3.1 illustrates the Theorem 3.1.

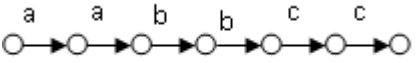
Example 3.1

Consider the string-graph P system $\Pi = (N, K, T, \mu, w_1, w_2, w_3, R_1, R_2, R_3)$, where $N = \{S_1, S_1', S_2, S_3, Q_2, Q_3\}$, $T = \{a, b, c\}$, $\mu = [_1 [_2]_2 [_3]_3]_1$,

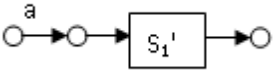
$K = \{Q_2, Q_3\}$ is the set of query symbols, w_1, w_2, w_3 are the initial string-graphs present in regions 1, 2 and 3 respectively and the string-graph rewriting rules R_1, R_2, R_3 are defined as follows:

R_1= {S_1::=begin 

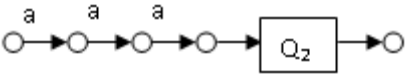
end ,

S_1 ::=begin 

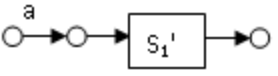
end ,

S_1 ::=begin 

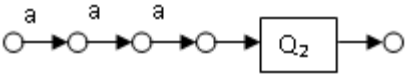
end ,

S_1::=begin 

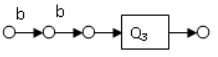
end ,

S_1'::= begin 

end ,

S_1'::= begin 

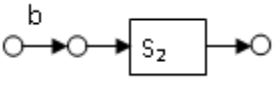
end ,

S_2 ::= begin 

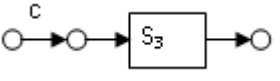
end ,

S_3 ::= begin 

end }

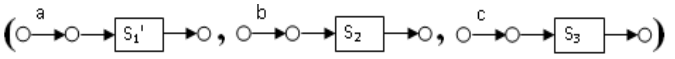
R_2 = S_2 ::= {begin 

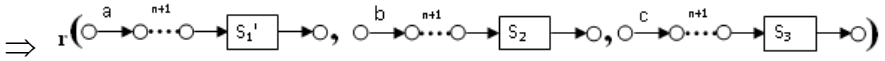
end}

R_3 = S_3 ::= { begin 

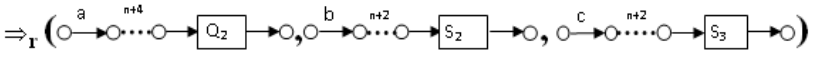
end }

First, we start with (S_1, S_2, S_3). Applying the third rule of R_1 first and then the fifth rule in $R_1$and the unique rules in R_2, R_3, for n ≥ 0 steps, we get the following derivation,

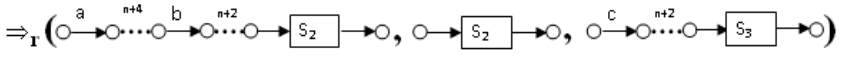
$(S_1, S_2, S_3) \Rightarrow_r$ 

$\Rightarrow_r$ 

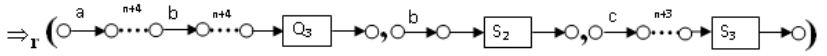
Next applying the sixth rule of R_1, we get the following

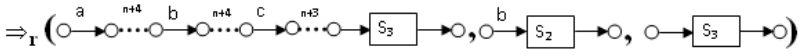
$\Rightarrow_r$ 

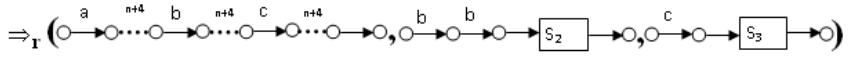
Since, the query symbol Q_2 is present in the first component, we perform a communication step by sending $(b^{n+2}S_2)^\bullet$ present in the second component to the first component replacing $(Q_2)^\bullet$ and we obtain the following,

$\Rightarrow_r$ 

After using the seventh rule of R_1, followed by the communication step for Q_3 and finally apply the terminating rule for S_3 in R_1, we get the following derivation steps

$\Rightarrow_r$ 

$\Rightarrow_r$ 

$\Rightarrow_r$ 

Hence, all string-graphs $(a^n\ b^n\ c^n)^\bullet$, $n \geq 4$ can be produced in this way.

The following corollary is interesting because of the Fact 2.3.7

Corollary 3.1

The string graph language $L = \{(a^n b^n c^n)^\bullet / n \geq 1\}$ can be generated by a string-graph P system using hypergraphs of order 2.

Remarks

This corollary is significant in implementation as it requires only hypergraphs of order two, in contrast with Fact 2.3.7

Conclusion

In this paper, we have defined string-graph P System and obtained its generative power. In future, we study further the other properties of this system.

References

[1] J. Dassow and B. Truthe, "On the number of components for some parallel communicating grammar systems." *Theoretical Computer Science,* vol. 387 pp 136-146, 2007.

[2] J. Dassow, G. Paun and G. Rosenberg. "Grammar systems." *Handbook of formal languages*, Springer, pp 155-213, 1997.

[3] G. Paun and G. Rozenberg. "A Guide to membrane computing." *Theoretical Computer Science*, vol. 287, pp. 73-100, 2002.

[4] A. Habel. "Hyperedge replacement: Grammars and Languages." *Lecture notes in Computer Science,* vol. 643, Springer-Verlag, New York.