Control Words of String Rewriting P Systems

Atulya K. Nagar · Ajeesh Ramanujan · K.G. Subramanian

Abstract P systems with controlled computations have been introduced and investigated in the recent past, by assigning labels to the rules in the regions of the P system and guiding the computations by control words. Here we consider string rewriting cell-like transition P system with label assigned rules working in acceptor mode and compare the obtained family of languages of control words over the rule labels with certain well-known language families. An application to chain code picture generation is also pointed out.

1 Introduction

Among several directions of study related to the bio-inspired computing model of P system in the area of membrane computing [11–13, 21] which was initiated by Paun [11], a fairly recent area of research is the concept of control language in a P system [14]. This control language is defined in terms of control words that are sequences of labels of rules used in the steps of a computation in a P system with the labels being symbols of an alphabet or the empty word λ.

P systems with label restricted transitions where in each computation step, all rules used have either the same label, or the empty label λ as well as P systems

Atulya K. Nagar
Department of Mathematics and Computer Science,
Faculty of Science, Liverpool Hope University,
Hope Park, Liverpool L16 9JD, UK

Ajeesh Ramanujan
Department of Computer Science And Engineering
College of Engineering, Thiruvananthapuram,
Kerala 695016 India

K.G. Subramanian*
Department of Mathematics and Computer Science,
Faculty of Science, Liverpool Hope University,
Hope Park, Liverpool L16 9JD, UK
* Honorary Visiting Professor

with the computations controlled by languages are considered in [8] and relationships between the families of sets of numbers computed by the various classes of controlled P systems are investigated. On the other hand, spiking neural P systems [4] with labels associated with the rules and referred to as labeled spiking neural P systems, are considered in [16] and the family of control languages of these systems is studied in comparison with other well-known families while tissue P systems [9] with label assigned rules are considered in [17] and a study of control languages of these systems is done.

Ramanujan [15] has introduced another variant by considering P system working in an accepting mode with label assigned rules in the regions and studied such a system as acceptor of a language of control words over the alphabet of rule labels. In fact the system reads the input control word from left to right, one symbol at a time, and uses in a maximally parallel way, the rules labeled with the symbol read or rules having λ label. If the input control word is processed completely with the P system reaching a configuration from a given finite set of final configurations, the input control word is accepted and otherwise the input word is rejected.

String rewriting cell-like transition P system with structured finite strings as objects and context-free rules in the regions with an evolved object being communicated as a whole between regions, was first introduced by Paun [11] and subsequently several variants have been considered and investigated [1–3, 6]. Following [11], here we consider string rewriting cell-like transition P system with label assigned rules working in acceptor mode analogous to [15]. We study the accepting power of such a P system by comparison of the family of languages of control words over the rule labels with certain well-known language families.

2 String Rewriting P System with control

We first recall a few needed notions of formal language theory [19].

A word w over an alphabet Σ is a string of symbols of the form $w = a_1 a_2 \cdots a_n$, $a_i \in \Sigma$, $1 \leq i \leq n$. The length of a word w is denoted by $|w|$. The set of all words over an alphabet Σ is denoted by Σ^*, which includes the empty word λ with no symbols. We write $\Sigma^+ = \Sigma^* - \{\lambda\}$. We denote by RE, CSL, CFL, REG respectively, the recursively enumerable, context-sensitive, context-free and regular families of languages in the Chomsky hierarchy [19].

We now consider a string rewriting cell-like transition P system and define label restricted computations that accept control words input to the system.

Let

$$\Pi = (V, \mu, A_1, \cdots, A_m, R_1, \cdots, R_m)$$

be a string rewriting transition P system with a membrane structure $\mu = [_1 [_2 \cdots [_m]_m \cdots]_2]_1$ of degree m in which the skin membrane and the membranes inside it are labelled $1, \cdots, m$ in a one-to-one manner; each A_i, $1 \leq i \leq m$ is a finite language, which can be empty, initially present in the region i consisting

of finite structured strings over V; the regions can have rewriting rules of the form $\alpha \to \beta$ where α ($\neq \lambda$) and β are words over V with R_i, $1 \leq i \leq m$ being the set of rules in region i; a string in a region is rewritten by only one rule at a time but all the strings in a region to which rules could be applied are to be written in a maximal parallel way; each rule has a target indication *here, in or out* with the usual meaning: *here* means the evolved string remains in the same region while *in* means that the evolved string is sent to the directly inner region and *out* means that the evolved string is sent to the directly outer region. The target *here* is not mentioned in the rule and is understood. Note that the evolved string, when communicated to another region, is sent as a whole.

As described in [15], writing $R = R_1 \cup \cdots \cup R_m$, we consider a labelling function $l : R \to \Sigma \cup \{\lambda\}$ that assigns a label from the finite set Σ, to every rule in the set R of the P system Π. For a rule $\alpha \to \beta$ with label $r \in \Sigma \cup \{\lambda\}$, we write the rule as $r : \alpha \to \beta(tar)$ where tar is the target associated with the rule. We also consider a finite set F of final configurations of the P system Π.

We denote by (Π, Σ, F) the P system Π with Σ and F defined as described. With such a system, we associate a language as follows: A configuration of the P system Π is $c = (L_1, \cdots, L_m)$ where L_i consists of the strings in region i. For two configurations c_1, c_2 of the system, c_1 yields c_2, written $c_1 \Rightarrow^b c_2$, $b \in \Sigma \cup \{\lambda\}$ and called a label restricted transition, only when rules with the same label b are used in evolving strings, if any, in the regions. In a label restricted transition, we say that the symbol b is an input symbol which is "consumed" if the label b of the rules used belongs to Σ and if the label is λ, then the input symbol is not consumed. A string w of input symbols over Σ is said to be accepted if, in a computation, all the symbols of w are consumed and a halting final configuration belonging to the set F is reached. If no rule with label λ is allowed in a computation, we call it a $\lambda-$restricted computation. Note that the language accepted depends on the final configuration being reached with the computation in the P system halting and so there is no need for mentioning any output membrane.

The language consisting of all the strings accepted by the P system Π with $\lambda-$restricted computations is denoted by $L_a(\Pi, \Sigma, F)$. If rules with λ labels are allowed in the computations in the P system Π, then the language accepted is denoted by $L_a^{\lambda}(\Pi, \Sigma, F)$. The family of languages $L_a(\Pi, \Sigma, F)$ (respectively $L_a^{\lambda}(\Pi, \Sigma, F)$) accepted by such P systems with at most $m\,(m \geq 1)$ membranes is denoted by $\mathcal{L}_a P_m$ (respectively $\mathcal{L}_a^{\lambda} P_m$).

We illustrate with a few examples of a P system with $\lambda-$restricted computations .

Example 1

Consider the string rewriting transition P system

$$\Pi_1 = (\{A, A', C\}, [_1\,]_1, \{A\}, R_1)$$

with the set of rule labels $\Sigma = \{a\}$ and the set of final configurations $F_1 = \{(\{C\})\}$ where

$$R_1 = \{a : A \to A', a : A' \to A, a : A' \to C\}$$

The initial configuration of the system is $(\{A\})$.

For an input control word of the form a^{2n}, a computation in Π_1 is done as follows: Starting from A in region 1, application of the rules $a : A \to A', a : A' \to A$ (with the former followed by the latter) and finally $a : A \to A'$ followed by the application of the rule $a : A' \to C$ once consumes a^{2n} in the input control word and results in a halting final configuration $(\{C\})$. Thus the language accepted by the system (Π_1, Σ, F_1) is $\{a^{2n} \mid n \geq 1\}$.

Note that if any word over $\{a\}$ which is not of the form a^{2n} for some $n \geq 1$, is given as input control word, then the corresponding computation in the P system cannot reach the final configuration specified in the system. The language accepted is $L_a(\Pi_1, \Sigma, F_1) = \{a^{2n} \mid n \geq 1\}$.

Example 2
Consider the language

$$L = \{ww^R \mid w \in \{a, b\}^+\}.$$

We construct a string rewriting transition P system Π_2 accepting L. Let

$$\Pi_2 = (\{S, A, B, C\}, [_1\,]_1, \{S\}, R_1)$$

with one membrane and the set of rule labels $\Sigma = \{a, b\}$ where

$$R_1 = \{a : S \to SA, a : S \to CA, b : S \to SB, b : S \to CB,$$

$$a : CA \to C, b : CB \to C\}.$$

The halting final configuration is $(\{C\})$. It can be seen that only for an input word of the form ww^R, $w \in \{a, b\}^+$, the system Π_2 carries out a successful computation starting from the initial configuration $(\{S\})$ and ends in a halting final configuration $(\{C\})$ consuming the input control word. Note that for an input control word ww^R, the system Π_2 produces in the generated string, the symbol A on consuming a and produces the symbol B on consuming b on reading the first-half w while the system Π_2 deletes the symbol A (respy. B) on consuming the corresponding a (respy. b) on reading the second-half w^R.

Remark
The study of Sureshkumar and Rama [20] on array P systems with the evolution of arrays being related to control strings of labels of array-rewriting rules, motivated us to consider string rewriting P systems in the accepting mode for defining string languages. Thus both the P system in [20] and the P system considered here, work in the accepting mode but the objects and the types of rules used are different.

We give certain comparison results comparing the family $\mathcal{L}_a P_m$ with other language families.

Theorem 1

$$REG \subset \mathcal{L}_a P_1$$

Proof
Let L be a regular language over an alphabet T generated by a regular grammar $G = (N, T, R, S)$ where N is the set of nonterminals with start symbol $S \in N$ and R is the set of right-linear rules of the forms $A \to aB, A \to a$ where $A, B \in N$ and $a \in T$. We construct a string rewriting transition P system

$$\Pi_3 = (N \cup T \cup \{d\}, [_1\,]_1, \{S\}, R_1)$$

($d \notin N \cup T$) with only one membrane and the set of rule labels $\Sigma = T$ where

$$R_1 = \{a : A \to B, a : A \to d \mid A \to aB \,\text{and}\, A \to a \in R\}$$

The halting final configurations is $(\{d\})$.

An input word $w = a_1 a_2 \cdots a_n$, $w \in L_1$, with a derivation $S \Rightarrow^* w$ in G, is accepted by a computation in Π_3. In fact a corresponding computation in Π_3 starts with S and is continued with the application of a sequence of rules in R_1 with labels $a_1, a_2, \cdots, a_{n-1}$ in order and also consuming these symbols in w, yielding finally a nonterminal, say, X. There will be a rule $X \to a_n$ in G as the last symbol in $w \in L$ is a_n. Hence $a_n : X \to d$ will be in the region 1 and hence the computation in Π_3 reaches the final halting configuration $(\{d\})$. Thus Π_3 accepts the language L. This proves the inclusion.

For proper inclusion, consider the non-regular language $L' = \{a^n b^n \mid n \geq 1\}$ over the alphabet $\Sigma = \{a, b\}$. We construct a string rewriting transition P system $\Pi_4 = (\{S_1, S_2, A\}, [_1\,]_1, \{S_1\}, R_1)$ with only one membrane and the set of rule labels being Σ where

$$R_1 = \{a : S_1 \to AS_1, a : S_1 \to AS_2, b : AS_2 \to S_2\}$$

The halting final configuration is $(\{S_2\})$. It can be seen that Π_4 accepts the language L'. In fact for an input control word $a^n b^n$, the system Π_4 generates the word $A^n S_2$ while consuming a^n using the rules $a : S_1 \to AS_1, a : S_1 \to AS_2$, after which b^n is consumed using the rule $b : AS_2 \to S_2$ with the system Π_4 finally yielding S_2 Any other control word not in L' will either be not consumed completely or the system will not enter the final configuration. □

A class of grammars richer than context-free grammars and called simple matrix grammars of degree $n \geq 1$ ($n - SMG$) has been considered in [7] and investigated further in many studies (see, for example, [5,7,18]). The family of languages generated by simple matrix grammars of degree $n \geq 1$ is denoted by $\mathcal{S}(n)$. A hierarchy of classes of languages generated by these grammars has been established in [7] by showing a language in $\mathcal{S}(n+1)$ which is not in $\mathcal{S}(n)$ [7]. We now show that $L_n \in \mathcal{L}_a P_2$.

Theorem 2
For $n \geq 1$,

$$\mathcal{L}_a P_1 - \mathcal{S}(n) \neq \emptyset.$$

Proof
The language

$$L_n = \{a_1^k a_2^k \cdots a_n^k b^k c_n^k c_{n-1}^k \cdots c_1^k \mid k \geq 1\},$$

with alphabet
$\{a_1, \cdots, a_n, b, c_1, \cdots, c_n\}$ is shown in [7, page 373] to be not in $\mathcal{S}(n)$.
We consider the case when n is even. The case when n is odd can be analogously dealt with. Let $n = 2m$, $m \geq 1$. We construct a string rewriting transition P system

$$\Pi_5 = (\{A_i, X_i, C_i, Y_i, B \mid 1 \leq i \leq 2m\}, [_1\,]_1, \{A_1\}, R_1).$$

The rule set

$$R_1 = \{a_1 : A_1 \to A_1X_1, a_1 : A_1 \to A_2X_1, a_2 : A_2X_1 \to X_2A_2, a_2 : A_2X_1 \to X_2A_3,$$
$$a_3 : X_2A_3 \to A_3X_3, a_3 : X_2A_3 \to A_4X_3, \cdots, a_{2m} : A_{2m}X_{2m-1} \to X_{2m}A_{2m},$$
$$a_{2m} : A_{2m}X_{2m-1} \to X_{2m}B, b : X_{2m}B \to BY_{2m}, b : X_{2m}B \to C_{2m}Y_{2m},$$
$$c_{2m} : C_{2m}Y_{2m} \to Y_{2m-1}C_{2m}, c_{2m} : C_{2m}Y_{2m} \to Y_{2m-1}C_{2m-1}, \cdots,$$
$$c_3 : Y_3C_3 \to C_3Y_2, c_3 : Y_3C_3 \to C_2Y_2, c_2 : C_2Y_2 \to Y_1C_2,$$
$$c_2 : C_2Y_2 \to Y_1C_1, c_1 : Y_1C_1 \to C_1.\}$$

The halting final configuration is $(\{C_1\})$. It can be seen that L_{2m} is accepted by Π_5. □

We now obtain certain comparison results comparing the family $\mathcal{L}_a^{\lambda}P_m$ with other language families.

Theorem 3

$$CF \subset \mathcal{L}_a^{\lambda}P_2$$

Proof
Let L_{CF} be a context-free language over an alphabet T generated by a context-free grammar $G = (N, T, R, S)$ in Chomsky normal form, where N is the set of nonterminals with start symbol $S \in N$ and R is the set of context-free rules of the forms $A \to BC, A \to a$ where $A, B, C \in N$ and $a \in T$. For every rule $A \to a$ in R, we include in N, a new nonterminal A_a and include in R, the rule $A_a \to a$ replacing $A \in N$ and the rule $A \to a \in R$.
We construct a string rewriting transition P system

$$\Pi_6 = (N \cup T \cup \{E_1, E_2\}, [_1\,[_2\,]_2\,]_1, \{E_1SE_2\}, R_1, R_2)$$

$(E_1, E_2 \notin N \cup T)$ with two membranes and the set of rule labels $\Sigma = T$ where

$$R_1 = \{\lambda : A \to BC, \lambda : A_aE_2 \to A_a(in) \mid A \to BC \,\text{and}\, A_a \to a \in R\}$$

$$R_2 = \{a : E_1A_a \to E_1 \mid A_a \in N\}.$$

The halting final configuration is $(\emptyset, \{E_1\})$. Note that the rules in region 1 with label λ generate a word corresponding to the given input word w belonging to L_{CF} in the computation in the system Π_6 but no input symbol is consumed till

the evolved string enters region 2 and a rule of the form $a : E_1 A_a \to E_1$ is applied, after which a successful computation consumes w and reaches the halting final configuration $(\emptyset, \{E_1\})$. Thus Π_6 accepts the language L_{CF}. This proves the inclusion.

For proper inclusion, consider the non-CF language $L'' = \{a^n b^n c^n \mid n \geq 1\}$. In order to accept L'' we construct a string rewriting transition P system

$$\Pi_7 = (\{S_1, S_2, A, B, C\}, [_1\,]_1, \{S_1\}, R_1)$$

with the set of rule labels $\Sigma = \{a, b, c\}$ and the final configuration $(\{B\})$ where

$$R_1 = \{a : S_1 \to AS_1C, a : S_1 \to AS_2C, b : AS_2 \to S_2, b : AS_2 \to B, c : BC \to B\}.$$

The initial configuration of the system is $(\{S_1\})$.

For an input control word of the form $a^n b^n c^n$, a computation in Π is done as follows: Starting from S_1 in the only region 1, application of the rule $a : S_1 \to AS_1C$ $(n-1)$ times followed by the application of the rule $a : S_1 \to AS_2C$ once consumes a^n in the input control word and yields the string $A^n S_2 C^n$. Then the application of the rule $b : AS_2 \to S_2$ $(n-1)$ times followed by the rule $b : AS_2 \to B$ once, consumes b^n in the control input word and yields the string BC^n. At this stage the application of the rule $c : BC \to B$, n times, consumes c^n in the input control word and results in a halting final configuration $(\{B\})$. Thus the language accepted by the system Π_6 is $\{a^n b^n c^n \mid n \geq 1\}$.

Note that if any word over $\{a, b, c\}$ which is not of the form $a^n b^n c^n$ for some $n \geq 1$, is given as control input word, then either the corresponding computation in the P system cannot be completed in the sense of consuming all the symbols in the input word or the computation cannot reach a final configuration specified in the system. □

Theorem 4

$$CS \subseteq \mathcal{L}_a^{\lambda} P_2$$

Proof
Let L_{CS} be a context-sensitive language over an alphabet T generated by a context-sensitive grammar $G = (N, T, R, S)$ in Kuroda normal form, where N is the set of nonterminals with start symbol $S \in N$ and R is the set of rules of the forms $AB \to CD$, $A \to BC$, $A \to B$, $A \to a$ where $A, B, C, D \in N$ and $a \in T$. For every rule $A \to a$ in R, we include in N, a new nonterminal A_a and include in R, the rule $A_a \to a$ replacing $A \in N$ and the rule $A \to a \in R$.
We construct a string rewriting transition P system

$$\Pi_8 = (N \cup T \cup \{F_1, F_2\}, [_1\, [_2\,]_2\,]_1, \{F_1 S F_2\}, R_1, R_2)$$

$(F_1, F_2 \notin N \cup T)$ with two membranes and the set of rule labels $\Sigma = T$ where

$$R_1 = \{\lambda : AB \to CD, \lambda : A \to BC, \lambda : A \to B, \lambda : A_a F_2 \to A_a(in) \mid AB \to CD,$$

$$A \to BC,\ A \to B \,\text{and}\, A_a \to a \in R\}$$

$$R_2 = \{a : F_1 A_a \to F_1 \mid A_a \in N\}.$$

The halting final configurations is $(\emptyset, \{F_1\})$. Note that the rules in region 1 with label λ generate a word corresponding to the given input word w belonging to L_{CS} in the computation in the system Π_8 but no input symbol is consumed till the evolved string enters region 2 and a rule of the form $a : F_1 A_a \to F_1$ is applied, after which a successful computation consumes w and reaches the halting final configuration $(\emptyset, \{F_1\})$. Thus Π_8 accepts the language L_{CS}. This proves the inclusion. □

Theorem 5

$$\mathcal{L}_a^{\lambda} P_3 = RE$$

Proof

We only have to prove the inclusion $\subseteq$, the opposite one is a consequence of the Turing-Church thesis.

Let L_{RE} be a recursive language over an alphabet T generated by a type 0 grammar $G = (N, T, R, S)$ in Penttonen normal form, with the non context-free rules from R labeled into a one-to-one manner, where N is the set of nonterminals with start symbol $S \in N$ and R is the set of rules of the forms $A \to \lambda$, $AB \to AC$, $A \to BC$, $A \to a$ where $A, B, C \in N$ and $a \in T$. For every rule $A \to a$ in R, we include in N, a new nonterminal A_a and include in R, the rule $A_a \to a$ replacing $A \in N$ and the rule $A \to a \in R$. We construct a string rewriting transition P system

$$\Pi_9 = (N \cup T \cup \{F_1, F_2\} \cup \{A' \mid A \in N\} \cup \{A_r \mid r : AB \to AC\}, [_1\, [_2 [_3\,]_3\,]_2\,]_1,$$

$$\{F_1 S F_2\}, \emptyset, \emptyset, R_1, R_2, R_3)$$

$(F_1, F_2, A' \notin N \cup T)$ with three membranes and the set of rule labels $\Sigma = T$ where

$$R_1 = \{\lambda : A \to BC, \lambda : A \to \lambda, \lambda : A_a F_2 \to A_a(in) \mid A \to BC,\ A \to \lambda, A_a \to a \in R\}$$

$$\cup\{\lambda : B \to B_r(in) \mid r : AB \to AC \in R\}$$

$$R_2 = \{a : F_1 A_a \to F_1 \mid A_a \in N\} \cup \{\lambda : A' \to A(out) \mid A \in N\}$$

$$\cup\{\lambda : A_r \to A_r \mid A \in N\}$$

$$\cup\{\lambda : AB_r \to AB_r(in) \mid r : AB \to AC \in R\}$$

$$R_3 = \{\lambda : B_r \to C'(out) \mid r : AB \to AC \in R\}.$$

The halting final configurations is $(\emptyset, \{F_1\}, \emptyset)$.

The system works as follows:

We can simulate the context free rules from R with out any difficulty. For simulating rules of the form $r : AB \to AC \in R$, assume that we have a string $w_1 A B w_2$ in membrane 1. In order to simulate the rule, we apply the $B \to B_r$ on the string so that the resulting string is sent to membrane 2. The string is sent to membrane 3 only if it has a substring of the form AB_r such that $r : AB \to AC \in R$.

Otherwise, by repeated application of the rule $B_r \rightarrow B_r$ in membrane 2, the computation goes on forever. In membrane 3, we replace the symbol B_r with C' and send the resulting string to membrane 2. From membrane 2, the string is sent to membrane 1 by applying the rule $C' \rightarrow C$. In this way we complete the simulation of the rule. The process is continued until no nonterminals other than $A_a, a \in T$ are present in the sentential form. We can see that the rules with λ label generate a word corresponding to the given input word w belonging to L_{RE} in the computation in the system Π_9 but no input symbol is consumed till the evolved string enters membrane 2 and a rule of the form $a : F_1 A_a \rightarrow F_1$ is applied, after which a successful computation consumes w and reaches the halting final configuration $(\emptyset, \{F_1\}, \emptyset)$. Thus Π_9 accepts the language L_{RE}. This proves the inclusion. □

3 An Application to Chain Code Pictures

A chain code picture [10], in a basic form, is composed of horizontal or vertical lines of unit length in the two dimensional plane with their ends at points with integer coordinates. The drawing of a unit length line to the left, right, up or down directions, starting from a point z with integer coordinates, is represented by the symbols l, r, u, d which are referred to as chain code symbols. A chain code picture itself can be represented by a word over the set $\Sigma = \{l, r, u, d\}$ by tracing the picture starting from an end of a unit line in the picture and covering all the unit lines that make up the picture.

For example, Fig. 1 shows a chain code picture in the shape of a star and can be represented by the word *rrrrlluudddd* starting from the left end of the horizontal line and ending at the bottom end of the vertical line. This representation is only one among the many words representing the picture.

Chain code picture grammars with context-free rules and the symbols l, r, u, d as terminal symbols, have been introduced in [10] and extensively studied by others subsequently. The language of such a grammar is called a chain code picture language.

Here we give an illustration of generating a chain code picture language $L_s = \{r^{2n} l^n u^n d^{2n} \mid n \geq 1\}$ with string rewriting P system accepting control words over $\Sigma = \{l, r, u, d\}$. For example, for $n = 2$, the word $r^4 l^2 u^2 d^4$ describes the star shaped picture in Fig. 1.

Consider the string rewriting cell-like transition P system

$$\Pi_s = (\{A, A', B, C, D, E, X\}, [_1\, [_2\,]_2\,]_1, \{A\}, \emptyset, R_1, R_2)$$

with the set of rule labels $\Sigma = \{l, r, u, d\}$ and the set of final configurations $F = \{(\emptyset, \{E\})\}$ where

$$R_1 = \{r : A \rightarrow A', r : A' \rightarrow AC, r : A' \rightarrow BC(in),$$

$$u : D \rightarrow DXX, u : D \rightarrow EXX(in)\},$$

$$R_2 = \{l : BC \to B, l : BC \to D(out), d : EX \to E\}.$$

For an input control word of the form $r^{2n}l^nu^nd^{2n}$, a computation in Π_s is done as follows: Starting from A in region 1, application of the rules $r : A \to A', r : A' \to AC$, with the former followed by the latter and finally applying the rule $r : A \to A'$ followed by the application of the rule $r : A' \to BC$ once consumes r^{2n} in the input control word and yields the string BC^n which is sent to region 2 due to the target indication (in) in the rule $r : A' \to BC$. In region 2, the application of the rule $l : BC \to B$ $(n-1)$ times followed by the rule $l : BC \to D$ once, consumes l^n in the control input word and yields the string D which is sent back to membrane 1 as the rule $l : BC \to D$ has target out. The application of the rule $u : D \to DXX$, $n-1$ times, followed by the rule $u : D \to EXX$ once, consumes u^n in the input control word and the resulting string EX^{2n} is sent again to region 2 due to the target in in the rule $u : D \to EXX$. Now the application of the rule $d : EX \to E$ $2n$ times consumes d^{2n} in the input control word resulting in the halting final configuration $(\emptyset, \{E\})$ which is in F. Thus the language accepted by the system (Π_1, Σ, F) is $\{r^{2n}l^nu^nd^{2n} \mid n \geq 1\}$.

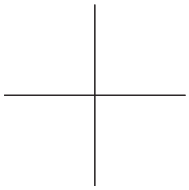

Figure 1: A star-shaped chain code picture corresponding to the word $r^4l^2u^2d^4$

4 Conclusion

String rewriting P systems with control are considered here for defining string languages in accepting mode and comparison of the family of languages accepted by such P systems with certain well-known families of languages is done. The comparison study can be extended to other families of languages as for example, the families of languages generated by L systems [19]. The study here and in [20] consider defining string languages in accepting mode by controlling computations in P systems by control strings although with different kinds of objects. It will be of interest to compare both these studies for their descriptional complexity in terms of number of membranes used in the constructions or the number of rules used and other kinds of measures.

Acknowledgements The authors thank the reviewers for their very useful comments which helped in improving the presentation

References

1. Besozzi, D., Mauri, G., Zandron, C., Hierarchies of parallel rewriting P systems - A survey, New Generation Comput., 22(4), 331–347 (2004).
2. Besozzi, D., Ferretti, C., Mauri, G., Zandron, C., Parallel rewriting P systems with deadlock, Lecture Notes Comput. Sci., 2568, 302–314 (2003).

3. Bottoni, P., Labella, A., Martin-Vide,C., Păun, Gh., Rewriting P systems with conditional communication, Lecture Notes Comput. Sci., 2300, 325–353 (2002).
4. Chen, H., Freund, R., Ionescu, M., Păun, G., Pérez-Jiménez, M.J., On string languages generated by spiking neural P systems, Fund. Inform., 75(1-4), 141–162 (2007).
5. Fernau, H.: Even Linear Simple Matrix Languages: Formal Language Properties and Grammatical Inference, Theor. Comput. Sci. 289, 425–456 (2002).
6. Ferretti, C., Mauri, G., Păun, Gh., Zandron, C., On three variants of rewriting P systems, Theor. Comput. Sci. 301, 201–215 ((2003).
7. Ibarra, O.H.: Simple Matrix Languages. Inf. Contro. 17, 359–394 (1970)
8. Krithivasan, K., Păun, Gh., Ramanujan, A., On controlled P systems. Fundam. Inform., 131(3-4), 451–464 (2014).
9. Martín-Vide, C., Păun, Gh., Pazos, J., Rodrguez Patón, A., Tissue P systems, Theor. Comp. Sci., 296(2), 295-326 (2003).
10. Maurer, H.A., Rozenberg, G. Welzl, E., Using string languages to describe picture languages, Inform. Control 54 (1982) 155–185.
11. Păun, Gh., Computing with membranes, J. Comp. System Sci., 61, 108–143 (2000).
12. Păun, Gh., Membrane Computing: An Introduction, Springer-Verlag Berlin, Heidelbrg, (2000).
13. Păun, Gh., Rozenberg, G., Salomaa, A., The Oxford Handbook of Membrane Computing, Oxford University Press, Inc., New York, NY, USA, 2010.
14. Păun, Gh., Pérez Jiménez, M. J., Languages and P Systems: Recent Developments, Computer Science Journal of Maldova, 20, 112–132 (2012),
15. Ramanujan, A., Control Languages in P Systems, Ph.D Thesis, Indian Institute of Technology Madras, 2014.
16. Ramanujan, A., Krithivasan, K., Control Languages Associated with Spiking Neural P Systems, Romanian J. Inform. Sci. Tech., 15(4), 301-318 (2012).
17. Ramanujan, A., Krithivasan, K., Control languages associated with tissue P systems, Proc. Twelfth Int. Conf. Unconventional Computation and Natural Computation 2013, Lecture Notes in Comp. Sci., 7956, 186–197 (2013).
18. Rosebrugh, R.D., Wood, D., Image Theorems for Simple Matrix Languages and n-parallel Languages, Math. Syst. Theor., 8(2), 150–155 (1975).
19. Rozenberg, G., Salomaa, A. (Eds.), Handbook of Formal Languages, Vol. 1-3, Springer, Berlin, 1997.
20. Sureshkumar, W., Rama, R., Regulating a distributed computing model via Chomsky hierarchy, GSTF Journal of Mathematics, Statistics and Operations Research, 3(1), 21–29 (2015).
21. Zhang, G., Pan, L.: A survey of membrane computing as a new branch of natural computing, Chinese Journal of Computers **33(2)**, 208–214 (2010).