Generalized Homogeneous P-Systems

Rudolf FREUND
Institut für Computersprachen
Technische Universität Wien
Karlsplatz 13
A-1040 Wien, Austria
email: rudi@logic.at
tel.: ++43 1 58801 18542

Franziska FREUND
De La Salle Schools
Strebersdorf
Anton Böck-Gasse 37
A-1215 Wien, Austria
email: ffreund@logic.at
tel.: ++43 1 58801 18542

Abstract. Recently Gheorghe Păun introduced P-systems as a new model for computations based on membrane structures. Many variants of P-systems were shown to have universal computational power. Using the membranes as a kind of filter for specific objects when transferring them into an inner compartment or out into the surrounding compartment turned out to be a very powerful mechanism in combination with suitable rules to be applied within the membranes in the model of generalized P-systems, GP-systems for short. GP-systems were shown to allow for the simulation of graph controlled grammars of arbitrary type based on productions working on single objects; moreover, various variants of GP-systems using splicing or cutting and recombination of strings were shown to have universal computational power, too. In this paper, we consider GP-systems with homogeneous membrane structures, GhP-systems for short, using cutting and recombination of string objects with specific markers at the ends of the strings that can be interpreted as electrical charges; these electrical charges determine the permeability of the membranes to the string objects. We show that GhP-systems have universal computational power; moreover, a very restricted variant of GhP-systems characterizes the minimal linear languages.

1 Introduction

The most important feature in the model of P-systems – as introduced by Gheorghe Păun in [8] – is the membrane structure (for a chemical variant of this idea see [1]) consisting of membranes hierarchically embedded in the outermost *skin* membrane. Every membrane encloses a *region* possibly containing other membranes; the part delimited by the membrane labelled by k and its inner membranes is called *compartment* k. A region delimited by a membrane not only may enclose other membranes but also specific objects and operators, which in general are considered as multisets, as well as evolution rules, which in *generalized P-systems (GP-systems)* as introduced in [3] and [4] are evolution rules for the operators. In GP-systems, ground operators as well as transfer operators (simple rules of that kind are called travelling rules in [11]) are taken into account; these transfer operators transfer objects or operators (or even rules) either to the outer compartment or to an inner compartment delimited by a membrane of specific kind with also checking for some permitting and/or forbidding conditions on the objects to be transferred. In that way, the membranes act as a filter like in test tube systems (e.g., see [9] and [5]). In [4] it was shown how GP-systems

with splicing or cutting and recombination rules can simulate test tube systems using the corresponding type of molecular rules.

In the specific model of generalized P-systems with special *h*omogeneous membr*a*ne structures considered in this *pa*p*e*r, the objects (which are assumed to be available in an unbounded number) can pass the membranes at a specific depth of the membrane structure depending on their electrical charges onl*y*. In contrast to the original definition of P-systems, in G(h)P-systems no priority relations on the rules are used, and we do not enforce parallelism guarded *b*y a un*i*versal clock, because these featu*r*es are not motivated biologically and, moreover, *th*eir implementation at least "in vitro" seems to be very *d*ifficult ("in silicio" these features might be implemented in an easier w*ay*). On the other hand, both the membrane structure as well as the operations used therein are motivated by nature; yet despite this biological background, real implementations in the lab ("in vitro") or in electronic media ("in silicio") remain topics for future interdisciplinary research.

In the following section we start with some preliminary notions from formal language theory and then give a general *de*finition of a molecul*ar* system that also captures the notion of cutting/recombination systems of strings. In the third section we introduce the model of *Gh*P-syst*e*ms considered in this paper and, as a first result, we sh*o*w that a *r*estricted variant of GhP-systems characterizes the minimal linear languages. Our main result showing that *Gh*P-syst*e*ms have universal computational power is elaborated in the fourth section. A short summary and an outlook to future research topics conclude the paper.

2 Preliminary Definitions

For an alphabet V, by V^* we denote the free monoid generated by V under the operation of concatenation; the *empty string* is denoted by λ, and $V^* \setminus \{\lambda\}$ is denoted by V^+. Any subset of V^+ is called a *λ-free (string) language*. $\mathbf{N}$ denotes the set of non-negative integers.

A minimal linear grammar G is a quadruple $(\{S\}, V_T, P, S)$, where S is the only non-terminal symbol and the start symbol of the grammar, V_T is the terminal alphabet, and P is the set of linear productions of the forms $S \to w$ with $w \in V_T^+$ or $S \to uSw$ with $uw \in V_T^+$.

A *molecular system* is a quadruple $\sigma = (B, B_T, P, A)$, where B and B_T are sets of *objects* and *terminal objects,* respectively, with $B_T \subseteq B$, P is a set of *productions,* and A is a set of axioms from B. A production p in P in general is a partial recursive relation $\subseteq B^k \times B^m$ for some $k, m \geq 1$, where we also demand that the domain of p is recursive (i.e., given $w \in B^k$ it is decidable if there exists some $v \in B^m$ with $(w, v) \in p$) and, moreover, that the range for every w is finite, i.e., for any $w \in B^k$, $card\,(\{v \in B \mid (w, v) \in p\}) < \infty$. For any two sets L and L' over B, we say that L' is computable from L by a production p if and only if $\{w_1, ..., w_k\} \subseteq L$ and $L' = L \cup \{v_1, ..., v_m\}$ for some $(w_1, ..., w_k) \in B^k$ and $(v_1, ..., v_m) \in B^m$ with $(w_1, ..., w_k, v_1, ..., v_m) \in p$; we also write $L \Longrightarrow_p L'$ and $L \Longrightarrow_\sigma L'$. A computation in σ is a sequence $L_0, ..., L_n$ such that $L_i \subseteq B$, $0 \leq i \leq n$, $n \geq 0$, as well as $L_i \Longrightarrow_\sigma L_{i+1}$, $1 \leq i \leq n$; in this case we also write $L_0 \Longrightarrow_\sigma^n L_n$, and moreover, we write $L_0 \Longrightarrow_\sigma^* L_n$ if $L_0 \Longrightarrow_\sigma^n L_n$ for some $n \geq 0$. The *language generated by σ* is

$$L(\sigma) = \{w \mid A \Longrightarrow_\sigma^* L,\ w \in L \cap B_T\}.$$

The special productions on string objects we shall consider in the following are the cutting and recombination operations:

A *cutting/recombination scheme* (a *CR-scheme* for short) is a quadruple (V, M, C, R), where V is a finite alphabet; M is a finite set of *markers;* V and M are disjoint sets; C is a set of *cutting rules* of the form $u\#l\$m\#v$, where $u \in V^* \cup MV^*$, $v \in V^* \cup V^*M$, and $m, l \in M$, and $\#, \$$ are special symbols not in $V \cup M$; $R \subseteq M \times M$ is the recombination relation representing the *recombination rules.* Cutting and recombination rules are applied to objects from $O(V, M)$, where we define

$$O(V, M) = V^+ \cup MV^* \cup V^*M \cup MV^*M.$$

For $x, y, z \in O(V, M)$ and a cutting rule $c = u\#l\$m\#v$ we define $x \Longrightarrow_c (y, z)$ if and only if for some $\alpha \in V^* \cup MV^*$ and $\beta \in V^* \cup V^*M$ we have $x = \alpha uv\beta$ and $y = \alpha ul$, $z = mv\beta$. For $x, y, z \in O(V, M)$ and a recombination rule $r = (l, m)$ from R we define $(x, y) \Longrightarrow_r z$ if and only if for some $\alpha \in V^* \cup MV^*$ and $\beta \in V^* \cup V^*M$ we have $x = \alpha l$, $y = m\beta$, and $z = \alpha\beta$. For a CR-scheme $\sigma = (V, M, C, R)$ and any language $L \subseteq O(V, M)$, $\sigma(L)$ then denotes the set of all objects obtained by applying one cutting or one recombination rule to objects from L. We also define $\sigma^0(L) = L$ and $\sigma^{i+1}(L) = \sigma\left(\sigma^i(L)\right)$ for all $i \geq 0$, as well as $\sigma^{(0)}(L) = L$ and $\sigma^{(i+1)}(L) = \sigma^{(i)}(L) \cup \sigma\left(\sigma^{(i)}(L)\right)$ for all $i \geq 0$; moreover, we denote $\sigma^*(L) = \bigcup_{i=0}^{\infty} \sigma^{(i)}(L)$. An *extended CR-system* is a molecular system of type $CR\ \sigma$, $\sigma = (O(V, M), O(V_T, M_T), P, A)$, where $V_T \subseteq V$ is the set of terminal symbols, $M_T \subseteq M$ is the set of terminal markers, A is the set of axioms, P is the union of the relations (productions) defined by the cutting rules from C $(\subseteq O(V, M) \times O(V, M)^2)$ and the recombination rules from R $(\subseteq O(V, M) \times O(V, M))$, and (V, M, C, R) is the underlying CR-scheme.

Throughout this paper we shall restrict ourselves to markers that can be interpreted as electrical charges of ions, i.e., we shall write $[+k]$ and $[-k]$, $k \in \mathbf{N}$, for these special markers. In that sense, the recombination rules we use will be of the simple forms $([+k], [-k])$ and $([-k], [+k])$, $k \in \mathbf{N}$.

In [5] we showed that test tube systems with only two test tubes using cutting and recombination rules in the tubes and filters of a special type between the tubes have the computational power of arbitrary grammars and Turing machines, respectively (which also holds true for test tube systems with splicing rules).

3 Generalized homogeneous P-systems

In this section we quite informally describe the model of GhP-systems discussed in this paper. Only the features not captured by the original model of P-systems as described in [2], [7], and [8] will be defined in more details. In these papers, only the number of symbols is counted in the multiset sense, whereas in [10] at least the outputs are strings. In generalized P-systems (for the basic definition of GP-systems the reader is referred to [3] and [4]) the objects usually are strings or graphs, etc.

The basic ingredient of a (G(h))P-system is a *membrane structure* consisting of several membranes placed within one unique surrounding membrane, the so-called skin membrane.

All the membranes can be labelled (in a one-to-one manner) by natural numbers; the outermost membrane (skin membrane) always is labelled by 0. In that way, a membrane structure can uniquely be described by a string of correctly matching parentheses, where each pair corresponds to a membrane. For example, the membrane structure depicted in Figure 1, which within the skin membrane contains two inner membranes labelled by 1, containing membrane 3, and 2, containing membrane 4, is described by $[_0[_1[_3]_3]_1[_2[_4]_4]_2]_0$. Figure 1 also shows that a membrane structure graphically can be represented by a Venn diagram, where two sets can either be disjoint or one set be the subset of the other one. In this representation, every membrane encloses a *region* possibly containing other membranes; the part delimited by the membrane labelled by k and its inner membranes is called *compartment* k in the following. The space outside the skin membrane is called *outer region.*

Informally, in [7] and [8] *P-systems* were defined as membrane structures containing multisets of objects in the compartments k as well as evolution rules for the objects. A priority relation on the evolution rules guarded the application of the evolution rules to the objects, which had to be affected in parallel (if possible according to the priority relation). The output was obtained in a designated compartment from a halting configuration (i.e., a configuration of the system where no rules can be applied any more).

A *generalized homogeneous P-system (GhP-system) of molecular type* X is a construct γ, $\gamma = (B, B_T, P, A, \mu, I, in, out)$, where

- (B, B_T, P, A) is a molecular system of type X (we shall restrict ourselves to molecular systems of type CR in the following);
- μ is a membrane structure (with the membranes labelled by natural numbers $0, ..., n$);
- $I = (I_0, ..., I_n)$, where I_k is the initial contents of compartment k containing a set of objects from A as well as a set of rules from P;
- in for each $j \in \{0, 1, ..., n\}$ specifies a condition an object must fulfill in order to be able to pass into the inner compartment of a membrane k, $k \in \{1, ..., n\}$, in the region enclosed by membrane j;
- out specifies a condition an object must fulfill in order to be able to pass a membrane k, $k \in \{1, ..., n\}$, into its surrounding compartment.

According to the definition given above, we only allow one unique out-condition, whereas the in-conditions specified by in may depend on the region the membrane to be passed lies in; yet in the following we will also demand that the in-conditions of regions at the same level of the membrane structure coincide.

A *computation* in γ starts with the initial configuration with I_k being the contents of compartment k. A transition from one configuration to another one is performed by applying a rule (from P) in I_k to an object in compartment k or by moving an element out of a compartment k or into a compartment k according to the conditions given by out and in. The language generated by γ is the set of all terminal objects $w \in B_T$ obtained in the terminal compartment 0 by some computation in γ.

Example 1. The main ingredients of a GhP-system (of type CR) generating the language $\{ww^r \mid w \in \{0,1\}^+\}$ (where w^r denotes the mirror image of w) are depicted in Figure 1. Moreover, the permeability of the membranes is specified in such a way that objects with a sum of electrical charges equal to +1 can leave every compartment, whereas objects

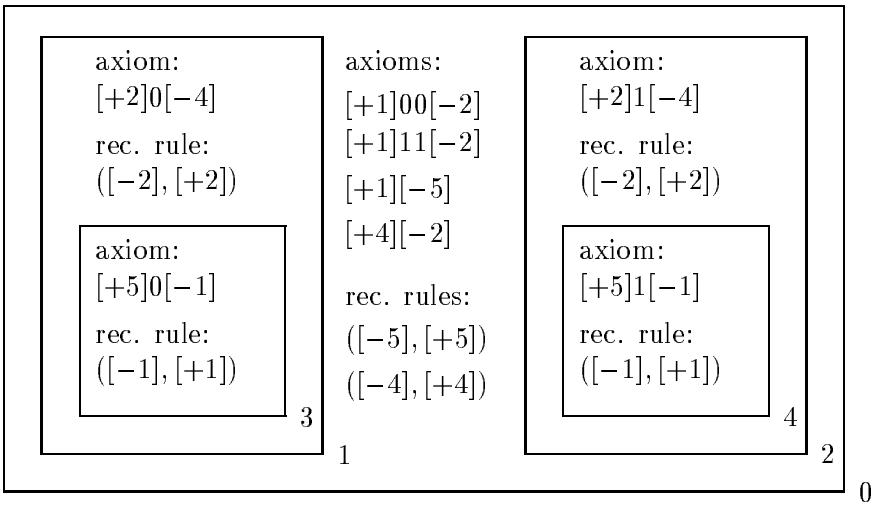

Figure 1: GhP-system generating $\{ww^r \mid w \in \{0,1\}^+\}$.

with a sum of electrical charges equal to -1 and -3 can enter compartments $1, 2$ and $3, 4$, respectively.

In the skin membrane we start with (terminal) objects of the form $[+1]ww^r[-2]$ for some $w \in \{0,1\}^+$; these can enter compartment 1 (or 2), where the application of the recombination rule yields $[+1]ww^r0[-4]$ ($[+1]ww^r1[-4]$), which can pass into compartment 3 (4), where the application of the corresponding recombination rule yields $[+5]0ww^r0[-4]$ ($[+5]1ww^r1[-4]$). These objects can pass back the membranes to compartment 0, where the objects are reduced to the terminal objects $[+1]0ww^r0[-2]$ ($[+1]1ww^r1[-2]$). □

It is easy to see that due to the different electrical charges of the axioms and the intermediate objects evolving in compartments 1 and 3 (2 and 4) we could even take the union of these compartments and work in the simpler membrane structure $[_0[_1]_1[_2]_2]_0$ (compare with Figure 2).

Lemma 2. Any minimal linear language generated by a minimal linear grammar G, $G = (\{S\}, V_T, P, S)$, can be generated by a GhP-system with membrane structure $[_0[_1]_1...[_n]_n]_0$, where n is the number of linear productions of the form $S \to uSw$ with $uw \in V_T^+$ in P.

Proof. As permeability conditions for the membranes we specify that objects with a sum of electrical charges equal to $+1$ can leave every compartment k, $1 \leq k \leq n$, whereas objects with a sum of electrical charges equal to -1 can enter any compartment k. The main parts of the GhP-system are depicted in Figure 2. The terminal objects $w \in L(G)$ can be extracted from compartment 0 in the form $[+1]w[-2]$. □

Without proof we mention that GhP-systems of the form as constructed in the preceding lemma exactly characterize the minimal linear languages.

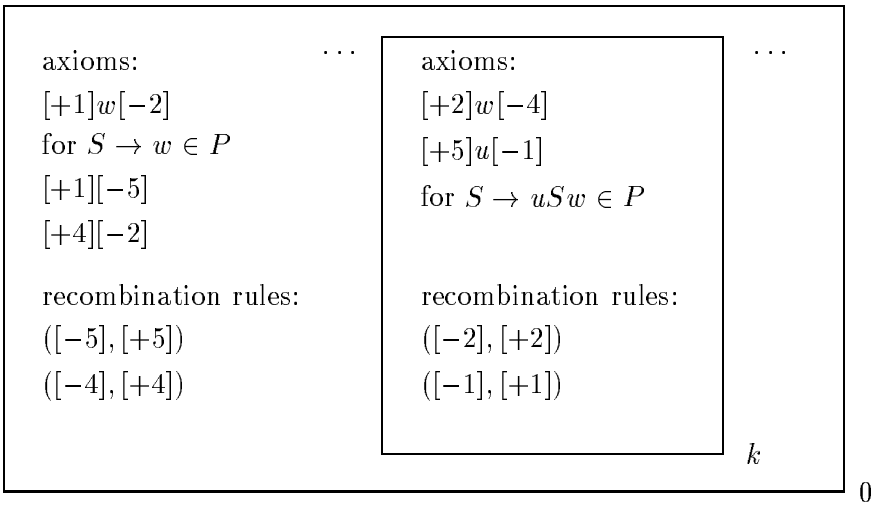

Figure 2: GhP-system generating $L(G)$, where $G = (\{S\}, V_T, P, S)$ is a minimal linear grammar.

4 The computational power of GhP-systems

The main result we prove in the following establishes the universal computational power of GhP-systems.

Theorem 3. Any recursively enumerable language L can be generated by a GhP-system γ with membrane structure

$$[_0[_1[_{n+1}]_{n+1}]_1 \cdots [_n[_{2n}]_{2n}]_n]_0.$$

Proof. The proof idea to simulate the productions of a grammar generating L like in Post systems in normal form has already been used in many papers on splicing systems and cutting/recombination systems (see [9] and [5]). Instead of a grammar G' generating L, we consider a grammar G generating the language $L\{d\}$, where the end marker d in any derivation of a word $w'd$, for $w' \in L$, is generated exactly in the last step of this derivation in G. A string w appearing in a derivation of such a grammar G, $G = (V_N, V_T, P, S)$, generating $L\{d\}$, is represented by its rotated versions $[+1]\,Xw_2Bw_1Y\,[-1]$, where $w = w_1w_2$ and B is a special symbol indicating the beginning of the string within the rotated versions and X, Y are special symbols marking the ends of the strings. Final strings first appear in the form $[+1]\,XBw'dY\,[-1]$, where w' is the final result from L which we want to get, and finally appear as $[+1]\,w'\,[-1]$ in compartment 0.

In compartment 0 we start with the axiom $[+1]XBSY[-1]$; the objects passed back to compartment 0 are objects of the form $[+1]\,Xw_2Bw_1Y\,[-1]$, $w_2w_1 \in (V_N \cup V_T \cup \{d\})^+$, or terminal objects of the form $[+1]w'[-1]$, $w' \in V_T^+$. Whereas the last region $[_n[_{2n}]_{2n}]_n$ is used to obtain the terminal objects $[+1]\,w'\,[-1]$, the other regions $[_k[_{n+k}]_{n+k}]_k$, $1 \leq k < n$, are used to simulate a production $\alpha \to \beta$ or the rotation of a symbol b (thus simulating a production $b \to b$) in the following way:

In compartment k, $1 \leq k < n$, we use the cutting rules $u\#[-2]\$[+2]\#\alpha Y[-1]$ and $[+1]X\#[-4]\$[+4]\#v$ for suitable strings u, v in order to obtain $[+4]w[-2]$ from $[+1]Xw\alpha Y[-1]$. This object $[+4]w[-2]$ with the sum of electrical charges equal to $+2$ can pass through membrane $n+k$. In compartment $n+k$ we then obtain the string object $[+1]X\beta wY[-1]$ by using the axioms $[+1]X\beta[-4]$ and $[+2]Y[-1]$ as well as the corresponding recombination rules $([-4],[+4])$ and $([-2],[+2])$.

The "neutral" objects with the sum of electrical charges equal to 0 can leave every compartment j, $1 \leq j \leq 2n$, and re-enter every compartment k, $1 \leq k \leq n$, from compartment 0.

Due to lack of space, the remaining technical details of the proof have to be left to the reader. □

Obviously, we can encode arbitrary additional data u in the axiom, i.e., we can take $[+1]XBSuY[-1]$ instead of $[+1]XBSY[-1]$ in compartment 0; hence, from the preceding theorem we also obtain the following result:

Corollary 4. Any partial recursive function can be computed by a GhP-system.

5 Conclusion

The idea of membrane structures offers a nearly unlimited variety of variants as it was already pointed out in [8]. In this paper we considered homogeneous (static) membranic structures with cutting and recombination rules inside and with the permeability of the membranes to the objects depending on the electrical charges of the string objects only. The formal definition of molecular systems would also allow us to consider other objects than strings, e.g., graphs and the corresponding cutting and recombination rules (see [6]). A thorough investigation of the generative power of such variants of GhP-systems and their complexity for solving special problems remains for future research.

Acknowledgements

We gratefully acknowledge all the interesting and fruitful discussions with Gheorghe Păun concerning his brilliant ideas for various models of P-systems.

References

[1] G. Berry and G. Boudol, *The chemical abstract machine,* Theoretical Computer Science 96 (1992), pp. 217-248.

[2] J. Dassow and Gh. Păun, *On the power of membrane computing,* TUCS Research Report No. 217 (Dec. 1998).

[3] R. Freund, *Generalized P-systems,* FCT'99, Iasi, Romania, September 1999.

[4] R. Freund, *Generalized P-systems with splicing and cutting/recombination,* WFLA'99, Iasi, Romania, September 1999.

[5] R. Freund and F. Freund, *Test tube systems: When two tubes are enough,* DLT'99, Aachen, Germany, July 1999.

[6] R. Freund and F. Freund, *Cutting and recombination of graphs,* AFL'99, Hungary, August 1999.

[7] Gh. Păun, *Computing with membranes,* TUCS Research Report No. 208 (Nov. 1998).

[8] Gh. Păun, *Computing with membranes: an introduction,* Bulletin EATCS **67** (Febr. 1999), pp. 139-152.

[9] Gh. Păun, G. Rozenberg, and A. Salomaa, *DNA Computing: New Computing Paradigms* (Springer-Verlag, Berlin, 1998).

[10] Gh. Păun, G. Rozenberg, and A. Salomaa, *Membrane computing with external output,* TUCS Research Report No. 218 (Dec. 1998).

[11] I. Petre, *A normal form for P-systems,* Bulletin EATCS **67** (Febr. 1999), pp. 165-172.