

# Generalized P-Systems with Splicing and Cutting/Recombination

Rudolf FREUND

Institut für Computersprachen, Technische Universität Wien,  
Resselg. 3, A-1040 Wien, Austria  
email: rudi@logic.at, tel.: ++43 1 58801 18542

**Abstract.** P-systems recently were introduced by Gheorghe Păun as a new model for computations based on membrane structures. Using the membranes as a kind of filter for specific objects when transferring them into an inner 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. In general, GP-systems allow for the simulation of graph controlled grammars of arbitrary type based on productions working on single objects. In this paper we consider GP-systems as computing devices using splicing or cutting and recombination of strings. Various variants of such systems are proved to have universal computational power, e.g., we show how test tube systems based on splicing or cutting and recombination of strings can be simulated by the corresponding GP-systems.

## 1 Introduction

In the model of P-systems as introduced in [19], the most important feature is the membrane structure (for a chemical variant of this idea see [2]) 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 [8] 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 [24]) 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, see [21]). In contrast to the original definition of P-systems, in GP-systems no priority relations on the rules are used, because this feature can be captured in another way by using the transfer conditions in the transfer operators. Moreover, in general the objects are not affected in parallel by the rules; the proofs of the results established so far in various papers on P-systems, see [6], [18], and [22], show that only bounded parallelism is needed.

Since Adleman's paper [1], the field of molecular computing and DNA computing has developed rapidly, and many interesting theoretical results could be obtained; a comprehensive overview is given in [21]. Hence, it is quite natural to consider GP-systems with splicing or cutting and recombination rules; these systems can be seen as a new model in the field of molecular computing that uses these operations of splicing or cutting and recombination (well-known in the area of DNA computing) within a membrane structure. 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 definition of a molecular system that captures the notions of splicing systems and cutting/recombination systems of strings; moreover, we define test tube systems based on these operations. In the third section we introduce the model of GP-systems used in this paper, especially GP-systems with splicing or cutting/recombination. The first results on GP-systems with splicing or cutting/recombination rules working in the multiset manner are established in the fourth section. Our main results showing that GP-systems with splicing or cutting/recombination rules allow for the simulation of the corresponding test tube systems based on splicing or cutting/recombination are elaborated in the fifth section. A short summary of the main results of this paper and an outlook to future research topics conclude the paper.

## 2 Preliminary Definitions

First, we recall some basic notions from the theory of formal languages (for more details, the reader is referred to [5]).

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  $\lambda$ , and  $V^* \setminus \{\lambda\}$  is denoted by  $V^+$ . Any subset of  $V^+$  is called a  $\lambda$ -free (*string*) *language*.  $\mathbf{N}$  denotes the set of non-negative integers.

A multiset over  $B$  is a function  $M : B \rightarrow \mathbf{N} \cup \{\infty\}$ ;  $(M, x)$  is the number of copies of  $x \in B$  in the multiset  $M$ . All the multisets we consider are supposed to be defined by recursive mappings  $M$ . The set  $\{w \in B \mid (M, w) > 0\}$  is called the support of  $M$  and it is denoted by  $\text{supp}(M)$ . A usual set  $S \subseteq B$  is interpreted as the multiset defined by  $S(x) = 1$  for  $x \in S$ , and  $S(x) = 0$  for  $x \notin S$ . For two multisets  $M_1, M_2$  we define their *union* by  $(M_1 + M_2, x) = (M_1, x) + (M_2, x)$ , and their *difference* by  $(M_1 - M_2, x) = (M_1, x) - (M_2, x)$ ,  $x \in B$ , provided  $(M_1, x) \geq (M_2, x)$  for all  $x \in B$ . Usually, a multiset with finite support,  $M$ , is presented as a set of pairs  $(x, (M, x))$ , for  $x \in \text{supp}(M)$ .

In order to prove our results in a quite general setting, we use the following general notion of a molecular system:

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 (finite) set of *productions*, and  $A$  is a (finite) multiset 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  $w \in B^k$ ,  $\text{card}(\{v \in B^m \mid (w, v) \in p\}) < \infty$ . For any two multisets  $L$  and  $L'$  over  $B$ , we say that  $L'$  is computable from  $L$  by a production  $p$  if and only if

$$L' = (L - (w_1 + \dots + w_k)) + (v_1 + \dots + v_m)$$

for some  $(w_1, \dots, w_k) \in B^k$  and  $(v_1, \dots, v_m) \in B^m$  with  $(w_1, \dots, w_k, v_1, \dots, v_m) \in p$ ; we also write  $L \Rightarrow_p L'$  and  $L \Rightarrow_\sigma L'$ . A computation in  $\sigma$  is a sequence

$$L_0, \dots, L_n$$

such that the  $L_i$ ,  $0 \leq i \leq n$ ,  $n \geq 0$ , are multisets over  $B$  as well as  $L_i \xRightarrow{\sigma} L_{i+1}$ ,  $1 \leq i \leq n$ ; in this case we also write  $L_0 \xRightarrow{\sigma} L_n$ , and moreover, we write  $L_0 \xRightarrow{\sigma^*} L_n$  if  $L_0 \xRightarrow{\sigma^n} L_n$  for some  $n \geq 0$ . The *language generated by  $\sigma$*  is

$$L(\sigma) = \{w \in B_T \mid A \xRightarrow{\sigma^*} L, (L, w) \geq 1\}.$$

## 2.1 Splicing systems and cutting/recombination systems

The special string productions we shall consider in the following are the splicing as well as the cutting and recombination operations:

A *splicing scheme* (an *H-system* for short) is a pair  $(V, P)$ , where  $V$  is an alphabet and  $P \subseteq V^* \# V^* \$ V^* \# V^*$ ;  $\#, \$$  are special symbols not in  $V$ ;  $P$  is the set of *splicing rules*. For  $x, y, z \in V^+$  and  $r = u_1 \# u_2 \$ u_3 \# u_4$  in  $P$  we define  $(x, y) \xRightarrow{r} z$  if and only if  $x = x_1 u_1 u_2 x_2$ ,  $y = y_1 u_3 u_4 y_2$ , and  $z = x_1 u_1 u_4 y_2$  for some  $x_1, x_2, y_1, y_2 \in V^*$ . For any language  $L \subseteq V^+$ ,  $\sigma(L)$  denotes the language obtained from  $L$  by any single application of rules from  $\sigma$  to a string from  $L$ . We also define  $\sigma^0(L) = L$  and  $\sigma^{i+1}(L) = \sigma(\sigma^i(L))$  for all  $i \geq 0$ , as well as  $\sigma^{(0)}(L) = L$  and  $\sigma^{(i+1)}(L) = \sigma^{(i)}(L) \cup \sigma(\sigma^{(i)}(L))$  for all  $i \geq 0$ ; moreover, we denote  $\sigma^*(L) = \bigcup_{i=0}^{\infty} \sigma^{(i)}(L)$ . An *extended mH-system* (or *extended splicing system with multisets*) is a molecular system  $\sigma$ ,  $\sigma = (V^*, V_T^*, P, A)$ , where  $V_T \subseteq V$ ,  $P$  is a set of splicing rules  $p \subseteq (V^* \times V^*) \times V^*$ , and  $A$  is the multiset of axioms.

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  $x \xRightarrow{c} (y, z)$  if and only if for some  $\alpha \in V^* \cup MV^*$  and  $\beta \in V^* \cup V^*M$  we have  $x = \alpha u v \beta$  and  $y = \alpha u l$ ,  $z = m v \beta$ . For  $x, y, z \in O(V, M)$  and a recombination rule  $r = (l, m)$  from  $R$  we define  $(x, y) \xRightarrow{r} z$  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  $L$ ;  $\sigma^i(L)$ ,  $\sigma^{(i)}(L)$ , and  $\sigma^*(L)$  are defined as above for H-systems. An *extended mCR-system* is a molecular system  $\sigma$ ,  $\sigma = (O(V, M), O(V_T, M_T), P, A)$ , where  $V_T \subseteq V$  is the set of terminal symbols,  $M_T \subseteq M$  is the set of terminal markers,  $A$  is the multiset 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.

In [3] and [13] it was proved that extended mH-systems with a finite multiset of axioms and a finite number of splicing rules have the computational power of arbitrary grammars and Turing machines, respectively. Similar results for CR-systems were proved in [14].

## 2.2 Test tube systems with filters

In the test tube systems considered in [4] and [9], the contents of the tubes is redistributed to all other tubes according to specific input or output filters. In

fact, most of the test tube systems to be found in literature work in such a manner that the contents of the test tubes is only distributed to selected test tubes that are prescribed, i.e., there are programs how to redistribute the contents of test tubes, e.g., see [1]. Hence, we use the following general definition for test tube systems as introduced in [10]:

A *test tube system* (a *TTS* for short)  $\sigma$  is a septuple  $(B, B_T, n, A, \rho, D, E)$ , where

1.  $B$  is a set of *objects*;
2.  $B_T$  is a set of *terminal objects*,  $B_T \subseteq B$ ;
3.  $n, n \geq 1$ , is the number of test tubes in  $\sigma$ ;
4.  $A = (A_1, \dots, A_n)$  is a sequence of sets of *axioms*, where  $A_i \subseteq B$ ,  $1 \leq i \leq n$ ;
5.  $\rho$  is a sequence  $(\rho_1, \dots, \rho_n)$  of sets of *test tube operations*, where  $\rho_i$  contains specific operations for the test tube  $T_i$ ,  $1 \leq i \leq n$ ;
6.  $D$  is a (finite) set of *prescribed output/input relations* between the test tubes in  $\sigma$  of the form  $(i, F, j)$ , where  $1 \leq i \leq n$ ,  $1 \leq j \leq n$ , and  $F$  is a (recursive) subset of  $B$ ;  $F$  is called a *filter* between the test tubes  $T_i$  and  $T_j$ .
7.  $E \subseteq \{i \mid 1 \leq i \leq n\}$  specifies the set of final test tubes.

In the interpretation used in this paper, the computations in the system  $\sigma$  run as follows: At the beginning of each computation step the axioms are distributed over the  $n$  test tubes according to  $A$ , hence, test tube  $T_i$  starts its first computation step with  $A_i$ . Now let  $L_i$  be the contents of test tube  $T_i$  at the beginning of a derivation step. Then in each test tube the rules of  $\rho_i$  operate on  $L_i$ , i.e., we obtain  $\rho_i^*(L_i)$ . The next substep is the redistribution of the  $\rho_i^*(L_i)$  over all test tubes according to the corresponding output/input relations from  $D$ , i.e., if  $(i, F, j) \in D$ , then the test tube  $T_j$  from  $\rho_i^*(L_i)$  gets  $\rho_i^*(L_i) \cap F$ , whereas the rest of  $\rho_i^*(L_i)$  that cannot be distributed to other test tubes remains in  $T_i$ . The final result of the computations in  $\sigma$  consists of all terminal objects from  $B_T$  that can be extracted from a *final test tube*  $f$  from  $E$ , i.e., we take  $\rho_f^*(L_f) \cap B_T$ .

In the following, the abbreviation CRTTS denotes a TTS using cutting and recombination rules in its test tubes, and the abbreviation HTTS denotes a TTS using splicing rules in its tubes.

In [9] test tube systems using cutting and recombination rules in the test tubes and filters of the forms  $V^+$ ,  $\{m\}V^*$ ,  $V^*\{n\}$ , and  $\{m\}V^*\{n\}$  for some markings  $m, n$  were explored, and such test tube systems with only three test tubes were shown to have the computational power of arbitrary grammars (Turing machines), the first tube being used for extracting the terminal results. In [11] this result was slightly improved: Any recursively enumerable language  $L$  can be generated by a CRTTS

$$\sigma = (MW^*M, [e]W^+[f], 2, (A_1, \emptyset), (C_1 \cup R_1, C_2), \{(1, F_1, 2), (2, F_2, 1)\}, \{2\})$$

with only two tubes and the filters  $F_1, F_2$  being a finite union of subsets of the form  $mW^+n$  with  $m, n \in M$ .

The theoretical features of test tube systems using splicing rules in the test tubes and regular filters of the form  $W^*$  for some alphabet  $W$  were investigated in [4]. In [11] a result similar to that stated above for CRTTS was shown for HTTS: Any recursively enumerable language  $L$  can be generated by an HTTS

$$\sigma = (W^*, EW^+F, 2, (A_1, A_2), (R_1, R_2), \{(1, F_1, 2), (2, F_2, 1)\}, \{2\})$$

with only two tubes and the filters  $F_1, F_2$  being a finite union of subsets of the form  $AW^+B$  with  $A, B \in W$ .

In both cases, two tubes were shown to be enough for all the calculations needed for yielding the terminal objects in the second tube. The proof idea used there, i.e., to simulate the productions of the underlying grammar on the right-hand side of the object in the test tube system and to rotate the string (which already goes back to Post systems in normal form) had already been used in many papers on splicing systems (see [17], [21]) and cutting/recombination systems (e.g., see [14]). The representation of a string  $w$  by objects  $[l] w [r]$  with specific markers  $[l]$  and  $[r]$  on the left-hand side and on the right-hand side had already been used by Dennis Pixton in [23].

### 3 Generalized P-systems (GP-systems)

In this section we quite informally describe the model of generalized P-systems discussed in this paper. Only the features not captured by the original model of P-systems as described in [18] and [19] will be defined in more details. For the basic definition of generalized P-systems the reader is referred to [8].

The basic part of a (G)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 (throughout this paper, 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 (with membrane 6 inside) and membrane 4, and 2 (containing membrane 5) is described by  $[_0 [_1 [_3 [_6]_6]_3 [_4]_4]_1 [_2 [_5]_5]_2]_0]$ . 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 [18] and [19] *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 P-system (GP-system) of molecular type  $X$*  is a construct  $G_P$  of the following form:

$$G_P = (B, B_T, P, A, \mu, I, O, R, f)$$

where

- $(B, B_T, P, A)$  is a molecular system of type  $X$ ;
- $\mu$  is a membrane structure (with the membranes labelled by natural numbers  $0, \dots, n$ );
- $I = (I_0, \dots, I_n)$ , where  $I_k$  is the initial contents of compartment  $k$  containing a (finite) multiset of objects from  $B$  as well as a (finite) multiset of operators from  $O$  and of rules from  $R$ ;

[tbph]

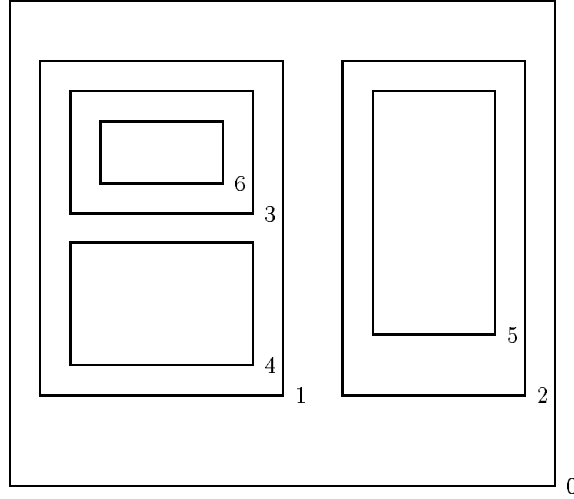


Fig. 1. Membrane structure:  $\left[ {}_0 \left[ {}_1 \left[ {}_3 \left[ {}_6 \right]_3 \left[ {}_4 \right]_4 \right]_1 \left[ {}_2 \left[ {}_5 \right]_5 \right]_2 \right]_0 \right.$

- $O$  is a finite set of operators (which will be described in detail below);
- $R$  is a finite set of (evolution) rules of the form  $(op_1, \dots, op_k; op'_1, \dots, op'_m)$  with  $k \geq 1$  and  $m \geq 0$ , where  $op_1, \dots, op_k, op'_1, \dots, op'_m$  are operators from  $O$ ;
- $f \in \{1, \dots, n\}$  is the label of the final compartment;

The main power of GP-systems lies in the operators, which can be of the following types:

- $P \subseteq O$ , i.e., the productions working on the objects from  $B$  are operators;
- $O_0 \subseteq O$ , where  $O_0$  is a finite set of special symbols, which are called *ground operators*;
- $Tr_{in} \subseteq O$ , where  $Tr_{in}$  is a finite set of transfer operators on objects from  $B$  of the form  $(\tau_{in,k}, E, F)$ ,  $1 \leq k \leq n$ ,  $E \subseteq P$ ,  $F \subseteq P$ ; the operator  $(\tau_{in,k}, E, F)$  transfers an object  $w$  from  $B$  being in compartment  $m$  into compartment  $k$  provided
  1. region  $m$  contains membrane  $k$ ,
  2. every production from  $E$  could be applied to  $w$  (hence,  $E$  is also called the permitting transfer condition),
  3. no production from  $F$  can be applied to  $w$  (hence,  $F$  is also called the forbidding transfer condition);
- $Tr_{out} \subseteq O$ , where  $Tr_{out}$  is a finite set of transfer operators on objects from  $B$  of the form  $(\tau_{out}, E, F)$ ,  $1 \leq k \leq n$ ,  $E \subseteq P$ ,  $F \subseteq P$ ; the operator  $(\tau_{out}, E, F)$  transfers an object  $w$  from  $B$  being in compartment  $m$  into compartment  $k$  provided
  1. region  $k$  contains membrane  $m$ ,
  2. every production from  $E$  could be applied to  $w$ ,
  3. no production from  $F$  can be applied to  $w$ ;
- $Tr'_{in} \subseteq O$ , where  $Tr'_{in}$  is a finite set of transfer operators working on operators from  $P$ ,  $O_0$ ,  $Tr_{in}$ , and  $Tr_{out}$  or even on rules from  $R$ ; a transfer operator

- $\tau_{in,k}$  moves such an element in compartment  $m$  into compartment  $k$  provided region  $m$  contains membrane  $k$ ;
- $Tr'_{out} \subseteq O$ , where  $Tr'_{out}$  is a finite set of transfer operators working on operators from  $P$ ,  $O_0$ ,  $Tr_{in}$ , and  $Tr_{out}$  or even on rules from  $R$ ; a transfer operator  $\tau_{out}$  transfers such an element in compartment  $m$  into the surrounding compartment.

In sum, the multiset  $O$  is the (disjoint) union of the sets  $P$ ,  $O_0$ , and  $Tr$ , where  $Tr$  itself is the (disjoint) union of the sets of transfer operators  $Tr_{in}$ ,  $Tr_{out}$ ,  $Tr'_{in}$ , and  $Tr'_{out}$ . In the following we shall assume that the transfer operators in  $Tr'_{in}$ , and  $Tr'_{out}$  do not work on rules from  $R$ ; hence, the distribution of the evolution rules is static and given by  $I$ . If in all transfer operators the permitting and the forbidding sets are empty, then  $G_P$  is called a *GP-system without transfer checking*. In contrast to the results elaborated in [8], in this paper we shall only consider GP-systems without ground operators.

A *computation* in  $G_P$  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 evaluating one evolution rule  $(op_1, \dots, op_k; op'_1, \dots, op'_m)$  in some compartment  $k$ , which means that the operators  $op_1, \dots, op_k$ , are applied to suitable elements in compartment  $k$  in the multiset sense (i.e., the operators as well as the objects the operators are applied to, are “consumed” by the usage of the rule; observe that ground operators have no arguments and are simply consumed in that way); thus we may obtain a new object by the application of a production and/or we may move elements out or into inner compartments by the corresponding transfer operators; yet we also obtain the operators  $op'_1, \dots, op'_m$  (in the multiset sense) in compartment  $k$ .

The language generated by  $G_P$  is the set of all terminal objects  $w \in B_T$  obtained in the terminal compartment  $f$  by some computation in  $G_P$ .

## 4 GP-systems working in the multiset manner

The first result we can show immediately is how molecular systems of type  $X$  working in the multiset manner can be simulated by GP-systems of the corresponding types.

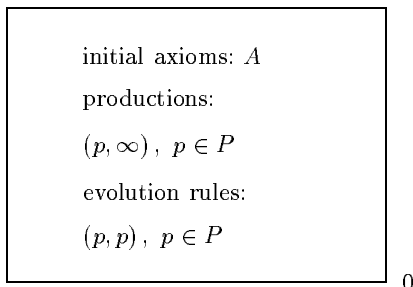
**Theorem 1.** *Any molecular system of type  $X$  working in the multiset manner can be simulated by a GP-system of the same type with the simplest membrane structure  $[0]_0$ .*

*Proof.* Let  $\sigma = (B, B_T, P, A)$  be a molecular system of type  $X$ . In a depictive way, the corresponding GP-system  $(B, B_T, P, A, [0]_0, I, P, R, 0)$ ,  $I = A \cup \{(p, \infty) \mid p \in P\}$ ,  $R = \{(p, p) \mid p \in P\}$  is illustrated in Figure 2. The productions are used by the evolution rules in a catalytic way only, i.e., they are not affected when being applied according to the evolution rule  $(p, p)$ . Hence, we would obtain the desired result even with  $I = A \cup \{(p, 1) \mid p \in P\}$ .  $\square$

Obviously, this theorem applies to extended mH-systems as well as to extended mCR-systems. In these cases it is even sufficient that at each time only one production is present in order to be evaluated by an evolution rule, which then has to generate the next production to be applied, which is stated in the following corollary:

**Corollary 2.** *Any recursively enumerable string language can be generated by a GP-system  $(B, B_T, P, A, [0]_0, I, P, R, 0)$  with the simplest membrane structure*

[tbph]



**Fig. 2.** Simulation of molecular system by GP-system

$[0]_0$  and splicing or cutting/recombination rules such that  $I = A \cup \{(p_0, 1)\}$  and  $R = \{(p, q) \mid p, q \in P\}$ .

Whereas in the GP-systems constructed in Theorem 1 an unbounded (infinite) number is assumed to be available, in the GP-systems constructed in Corollary 2, only one production is present at any time. For proving Corollary 2, we only have to find suitable first productions  $p_0$ : Inspecting the standard proofs as elaborated in [13], [21] or [14], we know that we start with an axiom of the form  $XBSY$  in splicing systems and of the form  $[x]XBS[y]$  in CR-systems, respectively. In the splicing case, we just can use the splicing rule  $XBS\#Y\$Z\#Y$  as  $p_0$  together with the axiom  $ZY$ ; the application of  $XBS\#Y\$Z\#Y$  does not affect the start axiom  $XBSY$ . In the case of CR-systems, we have to use a sequence of a cutting rule  $p_0$  and a suitable recombination rule  $p'_0$ , e.g.,  $p_0 = XBS\#[z]\$[z']Y$  and  $p'_0 = ([z], [z'])$ . The remaining details of the GP-systems then are rather obvious when using the constructions given in [13] and [14] and therefore left to the reader.

## 5 GP-systems Simulating Test Tube Systems

In this section we show how test tube systems based on molecular (DNA) productions (i.e., splicing or cutting and recombination) can be simulated by GP-systems of the corresponding type. Moreover, throughout this section we will assume that all operators and objects are available in an unbounded number.

**Theorem 3.** *Any HTTS*

$$\sigma = (W^*, EW^+F, n, A, P, D, F')$$

with  $n$  tubes and each filter between two tubes being a finite union of subsets of the form  $AW^+B$  with  $A, B \in W$  (special symbols only occurring at the ends of a string) can be simulated by a GP-system with splicing rules.

*Proof.* The main ingredients of the corresponding GP-system

$$(W^*, EW^+F, P', A', [0]_1 \dots [n]_n [n+1]_{n+1} [0]_0, I, P'', R, n+1)$$

are depicted in Figure 3.

The initial axioms in compartment  $i$ ,  $I_i$ , are the axioms  $A_i$  from  $A$  and some additional axioms needed for the marking rules and demarking rules (which will

be listed later), the splicing rules  $p_{i,j}, 1 \leq j \leq m_i$ , from  $P$ , the marking rules  $\mu_k^{A,u}, \mu_k^{v,B}$  for  $(i, AW^*B, k) \in D$ ,  $u, v, A, B \in W$ , the demarking rules  $\delta_i^{A,u}, \delta_i^{v,B}$ ,  $u, v, A, B \in W$ , and the transfer operators  $\tau_{out,k}^{A,u,v,B}$  for  $(i, AW^*B, k) \in D$ ,  $A, B \in W$ , and  $\tau_{out,f}^{u,v}$  for  $(i, EW^*F, f) \in D$  with  $f \in F$ ,  $u, v \in W$ .

For each partial filter  $(i, AW^*B, k)$  from tube  $i$  to tube  $k$  the marking rules  $\mu_k^{A,u}, \mu_k^{v,B}$  can mark suitable strings for being transferred from compartment  $i$  to compartment  $k$  via the “communication channel”  $\emptyset$ :

$$\mu_k^{A,u} = A_k \# Z \$ A \# u, \mu_k^{v,B} = v \# B \$ Z \# B_k; \text{ axioms: } A_k Z, Z B_k.$$

After such transfer, the demarking rules in the destination compartment remove the markers at the ends of the string again:

$$\delta_k^{A,u} = A \# Z \$ A_k \# u, \delta_k^{v,B} = v \# B_k \$ Z \# B; \text{ axioms: } AZ, ZB.$$

The marking rules and the demarking rules only change the first and the last character of a string; therefore, they cannot have an undesired effect in the middle of the strings. The transfer operators  $\tau_{out,k}^{A,u,v,B}$  (and  $\tau_{out,f}^{u,v}$ ) then control the passage through membrane  $i$  (to the desired compartment  $k$ ):

$$\begin{aligned} \tau_{out,k}^{A,u,v,B} &= (\tau_{out}, \{A_k \# Z \$ A_k \# u, v \# B_k \$ Z \# B_k\}, \emptyset), \\ \tau_{out,f}^{u,v} &= (\tau_{out}, \{E \# Z \$ E \# u, v \# F \$ Z \# F\}, \emptyset). \end{aligned}$$

Observe that the application of the permitting (enforcing) rules as  $A_k \# Z \$ A_k \# u, v \# B_k \$ Z \# B_k$  would not change the underlying string.

The evolution rules in compartment  $i$ ,  $1 \leq i \leq n$ , (there are no evolution rules in compartment  $n+1$ ) are all using the operators in a catalytic way:

$$\begin{aligned} &(p_{i,j}, p_{i,j}), 1 \leq j \leq m_i; \\ &\left(\mu_k^{A,u}, \mu_k^{A,u}\right), \left(\mu_k^{v,B}, \mu_k^{v,B}\right), \text{ for } (i, AW^*B, k) \in D, u, v, A, B \in W; \\ &\left(\delta_i^{A,u}, \delta_i^{A,u}\right), \left(\delta_i^{v,B}, \delta_i^{v,B}\right), \text{ for } u, v, A, B \in W; \\ &\left(\tau_{out,k}^{A,u,v,B}, \tau_{out,k}^{A,u,v,B}\right), \text{ if } (i, AW^*B, k) \in D, u, v, A, B \in W; \\ &\left(\tau_{out,f}^{u,v}, \tau_{out,f}^{u,v}\right), \text{ for } (i, EW^*F, f) \in D \text{ with } f \in F, u, v \in W. \end{aligned}$$

The “communication channel”  $\emptyset$  contains the following transfer operators to transfer the marked objects into the corresponding compartments:

$$\begin{aligned} \tau_{in,k}^{A,u,v,B} &= (\tau_{in,k}, \{A_k \# Z \$ A_k \# u, v \# B_k \$ Z \# B_k\}, \emptyset); \\ \tau_{in,n+1}^{u,v} &= (\tau_{in,n+1}, \{E \# Z \$ E \# u, v \# F \$ Z \# F\}, \emptyset). \end{aligned}$$

The evolution rules in compartment  $0$  are

$$\begin{aligned} &\left(\tau_{in,k}^{A,u,v,B}, \tau_{in,k}^{A,u,v,B}\right), \text{ if } (i, AW^*B, k) \in D \text{ for some } i \text{ with } 1 \leq i \leq n, \text{ and} \\ &\left(\tau_{in,n+1}^{u,v}, \tau_{in,n+1}^{u,v}\right), \text{ if } (i, EW^*F, f) \in D \text{ for some } f \in F \text{ and some } i \text{ with} \\ &1 \leq i \leq n, u, v \in W. \end{aligned}$$

According to the given construction, the compartment  $0$  only acts as a communication channel between the compartments  $i$ , which represent the corresponding tubes  $i$ ; the additional compartment  $n+1$  only collects the terminal objects, i.e., in this case no final intersection with  $B_T (= EW^+F)$  is necessary any more.

It should also be pointed out that instead of the complex transfer operators  $\tau_{out,k}^{A,u,v,B}$  in compartment  $0$  we could simply use  $(\tau_{out}, \emptyset, \emptyset)$ , because the transfer operators  $\tau_{in,k}^{A,u,v,B}$  in compartment  $0$  were sufficient to control the flow between the compartments.  $\square$

As it was proved in [11], any recursively enumerable language  $L$  can be generated by an HTTS

$$\sigma = (W^*, EW^+F, 2, (A_1, A_2), (R_1, R_2), \{(1, F_1, 2), (2, F_2, 1)\}, \{2\})$$

with only two tubes and the filters  $F_1, F_2$  being a finite union of subsets of the form  $AW^+B$  with  $A, B \in W$ . Following the construction given in the proof of Theorem 3, we then obtain the following result:

**Corollary 4.** *Any recursively enumerable language  $L$  can be generated by a GP-system with splicing*

$$(W^*, EW^+F, P', A', [{}_0[{}_1]{}_1]{}_0, I, P'', R, 0).$$

The details of the proof of Corollary 4 are rather obvious and therefore omitted. We just would like to mention that now compartment 0 not only works as a communication channel, but represents one of the two tubes, which also collects the terminal objects.

Similar results as Theorem 3 and Corollary 4 for splicing systems also hold true for cutting/recombination systems. Again, the complete construction is rather obvious and therefore left to the reader. For example, a left marking rule  $\mu_k^{A,u}$  now is the cutting rule  $[x] \# [z] \$ [x_k] \# u$ .

## 6 Summary and Future Research

As already pointed out in [19], the idea of membrane structures offers a nearly unlimited variety of variants. In this paper, we have considered generalized P-systems ([8]) in connection with productions (splicing or cutting and recombination rules) and systems (e.g., test tube systems) motivated from DNA computing. In the multiset case, GP-systems equipped with splicing or cutting and recombination rules even need no specific membrane structure and only have to make use of simple evolution rules for being able to generate any recursively enumerable string language. Without using the multiset counting feature, GP-systems with splicing or cutting and recombination rules can simulate the corresponding type of test tube systems even in the simplest non-trivial membrane structure. The advantage of GP-systems in comparison with the corresponding test tube systems is that no additional “mechanical” handling as with distributing the tubes is necessary, instead, GP-systems manage the transfer of objects themselves.

In contrast to the original definition of P-systems we do not enforce parallelism guarded by a universal clock, because this feature is not motivated biologically and, moreover, its implementation at least “in vitro” seems to be very difficult (“in silicio” this feature might be implemented in an easier way). On the other hand, the strictly sequential formal definition does not prohibit (at least “semi-”)parallel interpretations: In all variants considered in this paper, we could allow an arbitrary number of steps to happen in parallel using all the resources in the multiset manner. Even an enforced parallelism, i.e., all evaluations of evolution rules being possible with respect to multiset use of the current resources have to be carried out in one such parallel step, would not change the obtained results. Such an enforced parallelism in (G)P-systems, at least theoretically, becomes important if the complexity of the computations in these systems is considered; for example, in [20] a special variant of P-systems is shown to be able to solve the NP-complete problem SAT in linear time. Finding some interesting special problems to be solved in the framework of GP-systems as well as investigating the complexity of these computations in GP-systems might be topics for future research.

The formal definition of molecular systems also allows to consider other objects than strings, e.g., graphs and the corresponding splicing or cutting and recombination rules (see [7] and [12]). A thorough investigation of the generative power of such variants of GP-systems and their complexity for solving special problems remains for future research.

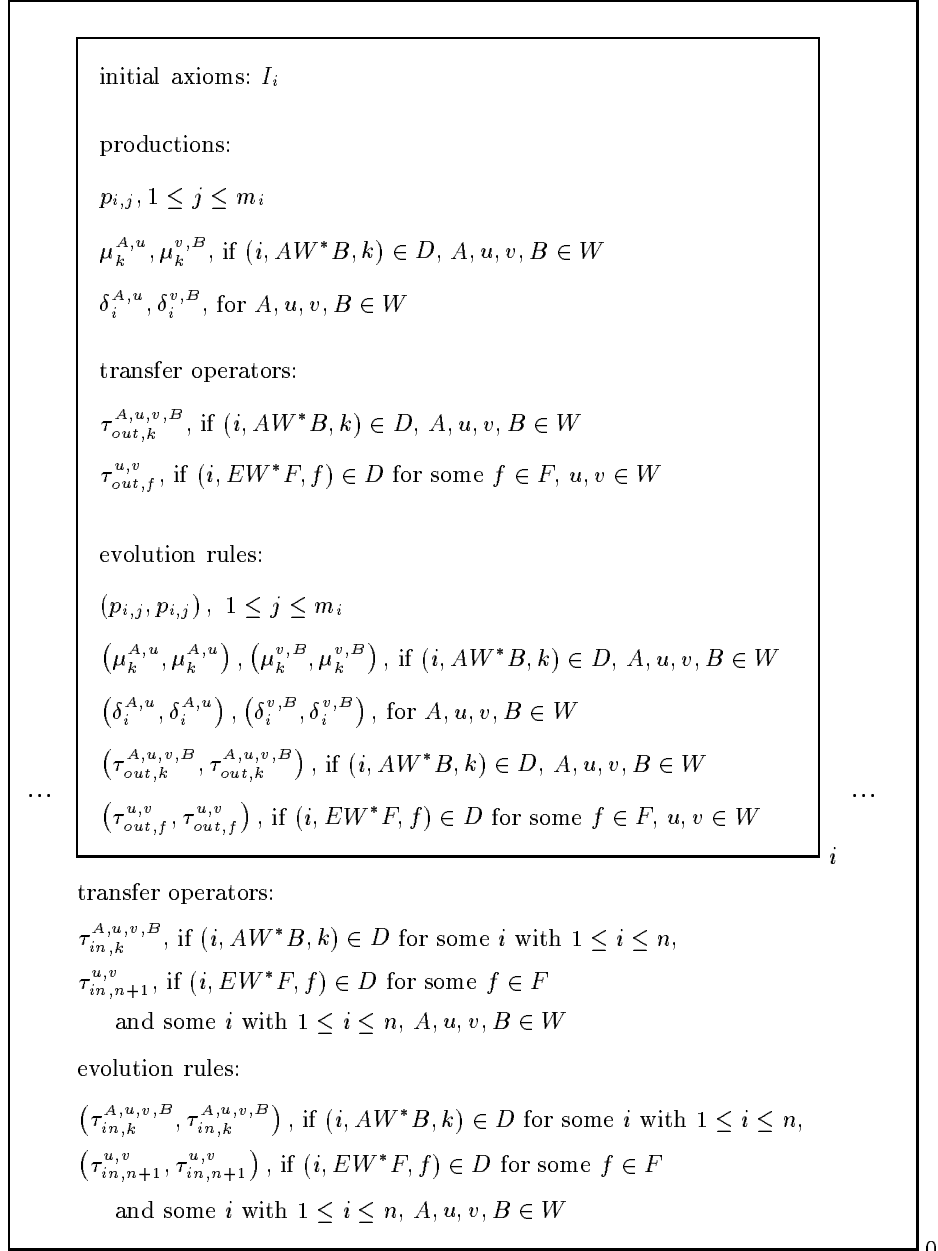
## **Acknowledgements**

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

## References

1. L. M. Adleman, *Molecular computation of solutions to combinatorial problems*, Science, 226 (Nov. 1994), pp. 1021-1024.
2. G. Berry and G. Boudol, *The chemical abstract machine*, Theoretical Computer Science 96 (1992), pp. 217-248.
3. E. Csuhaj-Varjú, R. Freund, L. Kari, and Gh. Păun, *DNA computing based on splicing: Universality results*, in: L. Hunter and T. Klein (eds.), Pacific Symposium on Biocomputing'96, World Scientific (1996), pp. 179-190.
4. E. Csuhaj-Varjú, L. Kari, and Gh. Păun, *Test tube distributed systems based on splicing*, Computers and Artificial Intelligence, Vol. 15 (2) (1996), pp. 211-232.
5. J. Dassow and Gh. Păun, *Regulated Rewriting in Formal Language Theory* (Springer, Berlin, 1989).
6. J. Dassow and Gh. Păun, *On the power of membrane computing*, TUCS Research Report No. 217 (Dec. 1998).
7. R. Freund, *Splicing systems on graphs*, International IEEE Symposium on Intelligence in Neural and Biological Systems, Herndon, VA, USA, May 1995, pp. 189 – 194.
8. R. Freund, *Generalized P-systems*, FCT'99, Iasi, Romania, September 1999.
9. R. Freund, E. Csuhaj-Varjú, and F. Wachtler, *Test tube systems with cutting/recombination operations*, Proceedings PSB'97, World Scientific (1997), pp. 163-174.
10. R. Freund and F. Freund, *Test tube systems or How to bake a DNA cake*, Acta Cybernetica, 12 (1996), pp. 445-459.
11. R. Freund and F. Freund, *Test tube systems: When two tubes are enough*, DLT'99, Aachen, Germany, July 1999.
12. R. Freund and F. Freund, *Cutting and recombination of graphs*, AFL'99, Hungary, August 1999.
13. R. Freund, L. Kari, and Gh. Păun, *DNA computing based on splicing: The existence of universal computers*, Theory of Computing Systems, Vol. 32, (1999), pp. 69-112.
14. R. Freund and F. Wachtler, *Universal systems with operations related to splicing*, Computers and Artificial Intelligence, Vol. 15 (4) (1996), pp. 273-294.
15. T. Head, Gh. Păun, and D. Pixton, *Language theory and molecular genetics*, chapter 7 in: G. Rozenberg and A. Salomaa (eds.), Handbook of Formal Languages, Vol. 2, Springer-Verlag (1997), 295-360.
16. M. Margenstern and Y. Rogojine, *Time-varying distributed H-systems of degree 2 generate any r.e. language*, MFCS'98 Satellite Workshop on Frontiers between Decidability and Undecidability, Brno (1998).
17. Gh. Păun, *Regular extended H systems are computationally universal*, Journal of Automata, Languages and Combinatorics, Vol. 1, Nr. 1 (1996), pp. 27-37.
18. Gh. Păun, *Computing with membranes*, TUCS Research Report No. 208 (Nov. 1998).
19. Gh. Păun, *Computing with membranes: an introduction*, Bulletin EATCS 67 (Febr. 1999), pp. 139-152.
20. Gh. Păun, *P systems with active membranes: Attacking NP complete problems, submitted*.
21. Gh. Păun, G. Rozenberg, and A. Salomaa, *DNA Computing: New Computing Paradigms* (Springer-Verlag, Berlin, 1998).
22. Gh. Păun, G. Rozenberg, and A. Salomaa, *Membrane computing with external output*, TUCS Research Report No. 218 (Dec. 1998).
23. D. Pixton, *Splicing in abstract families of languages*, Technical Report SUNY Univ. at Binghamton, New York, (1997).
24. I. Petre, *A normal form for P-systems*, Bulletin EATCS 67 (Febr. 1999), pp. 165-172.

[tbph]



**Fig. 3.** Simulation of DNA test tube system by GP-system.