Chapter 1

Membrane Systems

1.1 Further Basics

In this section we introduce two further types of grammars. The common feature is that they use only context-free rules, however, by some restrictions in the application of rules a larger generative power than that of context-free grammars is obtained. These grammars will be used in the sequel to discuss the power of membrane systems which are the subject of this chapter.

We start with the definition of a matrix grammar[1]. Essentially instead of context-free rules finite sequences of context-free rules are considered and if one applies the first rule of such a sequence one has to apply the further rules of this sequence in the given order.

Definition 1.1 *i) A matrix grammar is a quintuple* $G = (N, T, M, S, F)$ *where*

- N, T *and* S *are specified as in a context-free grammar,*
- $M = \{m_1, m_2, \dots, m_n\}$ *is a finite set of finite sequence of context-free rules, i.e., for* $1 \leq i \leq n$,

  $$m_i = (A_{i,1} \rightarrow w_{i,1}, A_{i,2} \rightarrow w_{i,2}, \dots, A_{i,r_i} \rightarrow w_{i,r_i})$$

  for some $r_i \geq 1$, $A_{i,j} \in N$, $w_{i,j} \in (N \cup T)^*$, $1 \leq j \leq r_i$,
- F *is a subset of the rules occurring in the sequences* m_i, $1 \leq i \leq n$.

ii) For a matrix $m = (A_1 \rightarrow w_1, A_2 \rightarrow w_2, \dots, A_r \rightarrow w_r) \in M$, *we say that* x *derives* y *by* m, *written as* $x \Longrightarrow_m y$ *if there exist words* $x_1, x_2, \dots x_{r+1}$ *such that the following conditions hold:*

- $x = x_1$, $y = x_{r+1}$,
- *for* $0 \leq i \leq r-1$, $x_i = x_i' A_i x_i''$ *and* $x_{i+1} = x_i' w_i x_i''$ *or* A_i *does not occur in* x_i, $x_{i+1} = x_i$ *and* $A_i \rightarrow w_i \in F$.

[1]To be precise, we introduce matrix grammar with appearance checking and with erasing rules. Because the other more restricted types of matrix grammars will not be used we only use the term matrix grammar.

iii) The language $L(G)$ generated by G consists of all words $z \in T^*$ which have a derivation

$$S \Longrightarrow_{m_{i_1}} w_1 \Longrightarrow_{m_{i_2}} w_2 \Longrightarrow_{m_{i_3}} \ldots \Longrightarrow_{m_{i_t}} = w_t = z$$

where $t \geq 1$ and $m_{i_j} \in M$ for $1 \leq j \leq t$.

The sequences $m \in M$ are called matrices. By definition the rules of a matrix have to be applied in the given order and all matrices of a matrix have to be applied where applications means a usual application if the left hand side occurs in the sentential form or no change if the left hand side does not occur in the sentential form and the rule belongs to F.

By $\mathcal{L}(MAT)$ we denote the family of all languages which can be generated by matrix grammars.

We give two examples.

Example 1.2 Let $G_1 = (\{S, A, B\}, \{a, b, c\}, \{m_1, m_2, m_3\}, S, \emptyset)$ be a matrix grammar with

$$m_1 = (S \to AB),\ m_2 = (A \to aAb, B \to Bc),\ \text{and } m_3 = (A \to ab, B \to c).$$

Then any derivation has the form

$$\begin{aligned} S &\Longrightarrow_{m_1} AB \Longrightarrow_{m_2} aAbBc \Longrightarrow_{m_2} a^2Ab^2Bc^2 \Longrightarrow_{m_2} a^3Ab^3Bc^3 \Longrightarrow_{m_2} \cdots \\ &\Longrightarrow_{m_2} a^{n-1}Ab^{n-1}Bc^{n-1} \Longrightarrow_{m_3} a^nb^nc^n, \end{aligned}$$

which yields that

$$L(G_1) = \{a^nb^nc^n \mid n \geq 1\}.$$

Example 1.3 We consider the matrix grammar

$$G_2 = (\{S, A, B, X, Y, Z, \#\}, \{a\}, \{m_1, m_2, \ldots, m_8\}, S, \{A \to \#, B \to \#\})$$

where
$$\begin{array}{ll} m_1 = (S \to XA), & m_2 = (X \to X, A \to BB), \\ m_3 = (X \to Y, A \to \#), & m_4 = (Y \to Y, B \to A), \\ m_5 = (Y \to X, B \to \#), & m_6 = (Y \to Z, B \to \#), \\ m_7 = (Z \to Z, A \to a), & m_8 = (Z \to \lambda, A \to a). \end{array}$$

Let us assume, that we have a sentential form XA^n for some $n \geq 1$; note that by the application of the matrix m_1 (which has been used in the first step) we obtain such a word with $n = 1$. We cannot apply the matrix m_3 since it introduces the nonterminal $\#$ which cannot be replaced, i.e., the derivation cannot be terminated. Hence the only applicable rule is m_2 which gives $XA^{n_1}BBA^{n_2}$ with $n_1 + n_2 = n - 1$. Again, m_2 is the only applicable if $n - 1 > 1$; moreover, this situation holds as long as a letter A is present. Thus we get after n applications of m_2 the sentential form XB^{2n}. Now the only applicable matrix is m_3 where $A \to \#$ cannot be applied which is allowed by $A \to \# \in F$. Now we have to proceed with $2n$ application of m_4 which yields YA^{2n}. Now we have two possibilities; we use m_5 or m_6. In the former case we obtain the sentential form XA^{2n} which has the same form as our starting sentential form; only the number of occurrences of A is doubled. In the latter case we have to apply $2n - 1$ times the matrix m_7 and once m_8 which results in a^{2n} (note that m_8 cannot be applied earlier since we then obtain a sentential with no occurrence of X, Y, Z, i.e., the derivation is blocked). Thus we double the number of A's or we terminate. Therefore

$$L(G_2) = \{a^{2^n} \mid n \geq 1\}.$$

Obviously, if all matrices have length 1, i.e., they consist of one rule only, then the application of the matrix coincides with the application of its rule. Thus such matrix grammars generate only context-free languages and all context-free languages can be generated. The example shows that also non-context-free languages can be generated by matrix grammars. Without proof we give that the generative power of matrix grammars equals the power of arbitrary phrase structure grammars.

Theorem 1.4 *$\mathcal{L}(MAT) = \mathcal{L}(RE)$.* □

We now present a normal form for matrix grammars.

Definition 1.5 *A matrix grammar $G = (N, T, M, S, F)$ is in normal form if the following conditions hold:*

- $N = N_1 \cup N_2 \cup \{S, Z, \#\}$, $S, Z, \# \notin N_1 \cup N_2$, $N_1 \cap N_2 = \emptyset$
- *any matrix of M has one of the following forms*
  — $(S \to XA)$ *with* $X \in N_1$, $A \in N_2$,
  — $(X \to Y, A \to w)$ *with* $X, Y \in N_1$, $A \in N_2$, $w \in (N_2 \cup T)^*$,
  — $(X \to Y, A \to \#)$ *with* $X \in N_1$, $Y \in N_1 \cup \{Z\}$, $A \in N_2$,
  — $(Z \to \lambda)$,
- *there is only one matrix of the form $(S \to XA)$ in M,*
- *F consists of all rules of the form $A \to \#$ with $A \in N_2$.*

Moreover, in any derivation, the matrix $(Z \to \lambda)$ is only applied to a sentential form $w_1 Z w_2$ with certain $w_1 w_2 \in T^*$.

The following theorem shows that the naming normal form is used correctly.

Theorem 1.6 *For any recursively enumerable language L, there is a matrix grammar G in normal form such that $L(G) = L$.*

Proof. We first proof that the required special forms of matrices are sufficient. Let L be a recursively enumerable language. By Theorem 1.4, there is a matrix grammar $G' = (N, T, M, S', F)$ such that $L(G') = L$. We assume that

$$\begin{aligned} N &= \{A_1, A_2, \ldots, A_t\}, \\ M &= \{m_1, m_2, \ldots m_n\}, \\ m_i &= (A_{i,1} \to w_{i,1}, A_{i,2} \to w_{i,2}, \ldots, A_{i,r_i} \to w_{i,r_i}) \text{ for } 1 \leq i \leq n. \end{aligned}$$

We construct the matrix grammar G in normal form by the settings

$$\begin{array}{rcl}
N_1 & = & \{[i,j] \mid 1 \leq i \leq n,\ 1 \leq j \leq r_i\} \cup \{[k] \mid 1 \leq k \leq t, \\
N_2 & = & N, \\
& & \text{new letters } S, Z, \#, \\
(1) & & (S \to [i,1]S') \text{ for } 1 \leq i \leq n, \\
(2) & & ([i,j] \to [i,j+1], A_{i,j} \to w_{i,j}) \text{ for } 1 \leq i \leq n,\ 1 \leq j < r_i, \\
(3) & & ([i,j] \to [i,j+1], A_{i,j} \to \#) \text{ for } 1 \leq n,\ 1 \leq j < r_i,\ A_{i,j} \to w_{i,j} \in F, \\
(4) & & ([i,r_i] \to [i',1], A_{i,r_i} \to w_{i,r_i}) \text{ for } 1 \leq i \leq n,\ 1 \leq i' < n, \\
(5) & & ([i,r_i] \to [i',1], A_{i,r_i} \to \#) \text{ for } 1 \leq i \leq n,\ 1 \leq i' < n,\ A_{i,r_i} \to w_{i,r_i} \in F, \\
(6) & & ([i,r_i] \to [1], A_{i,r_i} \to w_{i,r_i}) \text{ for } 1 \leq i \leq n, \\
(7) & & ([i,r_i] \to [1], A_{i,r_i} \to \#) \text{ for } 1 \leq i \leq n,\ A_{i,r_i} \to w_{i,r_i} \in F, \\
(8) & & ([i] \to [i+1], A_i \to \#) \text{ for } 1 \leq i \leq t-1, \\
(9) & & ([t] \to Z, A_t \to \#), \\
(10) & & (Z \to \lambda).
\end{array}$$

We have $L(G') = L(G)$ by the following reasons. We start with an application of a matrix of type (1), which says that the application of the i-th matrix is started. The simulation is performed by applying in succession the rules of type (2) or (3) with left hand sides $[i,1], [i,2], \ldots, [i, r_i - 1]$ in their first rules and finishing the simulation with rules of type (4), (5), (6) or (7) with left hand side $[i, r_i]$ in its first rule. The matrices of types (3) and (5) can only be applied if the nonterminal $A_{i,j}$ and A_{i,r_i} does not occur in the sentential form since otherwise the nonterminal # is introduced which cannot be changed (there are no rules with left hand side #), i.e., we cannot derive a terminal word. After the simulation of a complete matrix of G', we start another simulation of a matrix if we applied a rule of type (4) or (5) and we start the applications of type (8) and (9) if we applied matrices of type (6) or (7). By the matrices of type (8) and (9) we check that no nonterminal is present in the sentential form (otherwise a # is introduced). Finally, we cancel the first letter Z. Thus any derivation consists of simulations of the application of matrices in G followed by a check that the word is terminal.

It remains to show that one rule of the form $(S \to XA)$ is sufficient. In order to prove this we change G' to $G'' = (N \cup \{S''\}, T, M \cup \{(S'' \to S'), S'', F)$. It is obvious that $L(G') = L(G'')$ since any derivation has to start with $S'' \Longrightarrow S'$. Moreover, there is a unique matrix $(S'' \to S')$ of G'' which has to be used in the first step. Such the construction of G as above starting from G'' requires only the matrix $(S \to [i,1]S'')$ where i refers to $(S'' \to S')$. □

The second concept is that of grammar systems[2] The basic idea can be illustrated as follows. Some (context-free) grammars are sitting around a table and a word is placed on the table. Now a grammar G can take the word and derive it as long productions of the grammar G are applicable. If no rule can be applied by G, then G puts the newly derived

[2]To be precise we consider here cooperating distributed grammar systems with terminating derivation mode t; however, since other types of grammar systems are not used, we use the term grammar system only.

word back to the table. Obviously, this process can be iterated. We have a cooperation of the grammars since rules of another grammar cannot be used if a grammar works.

We now give the formal definition.

Definition 1.7 *i) A grammar system with n components is an $(n+3)$-tuple*

$$G = (N, T, P_1, P_2, \ldots, P_n, S)$$

where

- *N, T, S are specified as in a context-free grammar,*
- *$P_1, P_2, \ldots, P_n$ are finite subsets of $N \times (N \cup T)^*$, i.e., P_i is a finite set of context-free rules for $1 \leq i \leq n$.*

ii) We say that x derives y by the set P_i, $1 \leq i \leq n$, written as $x \Longrightarrow_{P_i}^{t} y$ if $x \Longrightarrow_{P_i}^{*} y$, i.e., y can be obtained from x by a derivation which only uses rules from P_i, and no rule of P_i can be applied to y.

iii) The language $L(G)$ generated by the grammar system G consists of all word $z \in T^*$ which can be generated by a derivation of the form

$$S \Longrightarrow_{P_{i_1}}^{t} w_1 \Longrightarrow_{P_{i_2}}^{t} w_2 \Longrightarrow_{P_{i_3}}^{t} \ldots \Longrightarrow_{P_{i_s}}^{t} w_s = z$$

for some $t \geq 1$, $1 \leq i_j \leq n$, $1 \leq j \leq s$.

The sets $P_1, P_2, \ldots, P_n$ are called the components of the grammar system.

By $\mathcal{L}_n(CF)$ we denote the set of languages which can be generated by grammar systems with n components.

We now present two examples which generate the same languages as the matrix grammars considered in Examples 1.2 and 1.3.

Example 1.8 Let $G_1' = (\{S, A, B, A', B'\}, \{a, b, c\}, P_1, P_2, P_3, S)$ be a grammar system with the three components

$$P_1 = \{S \to AB, A \to aA'b, B \to B'c\},\ P_2 = \{A' \to A, B' \to B\},\ P_3 = \{A \to \lambda, B \to \lambda\}.$$

Obviously, any derivation in the grammar system G_1' has the form

$$\begin{aligned} S \ &\Longrightarrow_{P_1}^{t} \ aA'bB'c \Longrightarrow_{P_2}^{t} aAbBc \Longrightarrow_{P_1}^{t} a^2A'b^2B'c^2 \Longrightarrow_{P_2}^{t} a^2Ab^2Bc^2 \Longrightarrow_{P_1}^{t} \ldots \\ &\Longrightarrow_{P_2} \ a^nAb^nBc^n \Longrightarrow_{P_3} a^nb^nc^n, \end{aligned}$$

which gives

$$L(G_1' = \{a^nb^nc^n \mid n \geq 1\}.$$

Example 1.9 We consider the grammar system $G_2' = (\{S, S'\}, \{a\}, P_1, P_2, P_3, S)$ with the three components

$$P_1 = \{S \to S'S'\},\ P_2 = \{S' \to S\},\ P_3 = \{S \to a\}.$$

Then any derivation is of the form

$$\begin{aligned} S &\Longrightarrow_{P_1}^{t} S'S' \Longrightarrow_{P_2}^{t} SS \Longrightarrow_{P_1}^{t} (S')^4 \Longrightarrow_{P_2}^{t} S^4 \Longrightarrow_{P_1}^{t} (S')^8 \Longrightarrow_{P_2}^{t} S^8 \dots \\ &\Longrightarrow_{P_2}^{t} S^{2^t} \Longrightarrow_{P_3}^{t} a^{2^n} \end{aligned}$$

and, consequently,

$$L(G_2') = \{a^{2^n} \mid n \geq 0\}.$$

It is clear that a grammar system with one component is a context-free grammar. Therefore $\mathcal{L}_1(CF) = \mathcal{L}(CF)$. However, if we use three components, then non-context-free languages can be generated. The following theorem says that we need three components in order to generate non-context-free languages and that three components are sufficient to generate languages which can be obtained by an arbitrary number of components. We omit the proof which needs some knowledge on further closure properties of $\mathcal{L}(CF)$ and on extended tabled Lindenmayer systems.

Theorem 1.10 *i)* $\mathcal{L}(CF) = \mathcal{L}_1(CF) = \mathcal{L}_2(CF)$.
ii) For any $n \geq 3$, $\mathcal{L}_n(CF) = \mathcal{L}_3(CF)$. □

Let L be a language. Then we set

$$N(L) = \{n \mid n = |w| \text{ for some } w \in L\},$$

i.e., $N(L)$ is the set of all lengths of words in L.

Let X be a set of grammars. Then we set

$$N(X) = \{N(L) \mid L \in \mathcal{L}(X)\}.$$

Without proof we mention the following statements.

Theorem 1.11 *i)* $N(REG) = N(CF) \subset N(CS) \subset N(RE)$.
ii) A set M of natural numbers belongs to $N(CF)$ if and only if there are numbers $r, s, p, q_1, q_2, \dots q_r, p_1, p_2, \dots, p_s$ *such that* $r \geq 0$, $s \geq 0$, $p \geq 1$, $q_1 < q_2 < \dots < q_r < p_1 < p_2 < \dots < p_s$ *and*

$$M = \{q_1, q_2, \dots, q_r\} \cup \bigcup_{i=1}^{s} \{p_i + np \mid n \in \mathbf{N}_0\}.$$

In a subset M of a set U, any element of M occurs once, however, in many applications an element can occur more often. Thus we associate a number of occurrences with any element in a set. Formally, this leads to the concept of a multiset.

A multiset M over U is a mapping M of U into the set $\mathbf{N}_0 \cup \{\infty\}$ of non-negative integers and a symbol ∞ representing infinity. $M(x)$ is called the multiplicity of x.

A multiset M is called finite iff there is a finite subset U' of U such that $M(x) = 0$ for $x \notin U$ and $M(x) \neq \infty$ for $x \in U$. Then its cardinality is the sum of the multiplicities of the elements of U.

The cardinality and the length of a finite multiset M are defined as $\#(M) = \sum_{x \in U} M(x)$.

Let M be a finite multiset over a finite set U. Then we can build a word $w_M \in U^*$ such that $\#_x(w_M) = M(x)$ for all $x \in U$. Obviously, w_M is uniquely determined up to the order of the letters. For example, if M over $\{a, b, c, d\}$ is given by $M(a) = 2$, $M(b) = 3$, $M(c) = 1$, and $M(d) = 0$, then we can choose any of the words a^2b^3c, *abcabb*, *cbabab*, and *abcbab* as w_M.

Conversely, with any word w over a U, we can associate a multiset M by setting $M(x) = \#_x(w)$. Clearly, $M_w = w$ holds for this situation.

Thus, in the sequel, we shall identify a finite multiset M with an associated word w_M.

Obviously, $\#(M) = |w_m|$.

1.2 Basic Type of Membrane Systems and its Power

The idea of membrane systems is to model a biological cell as a computing device. A cell is considered as a membrane which contains further membranes which can contain membranes again. For instance the kernel of a cell gives a membrane contained in the skin membrane of the cell. Moreover, there is a change of the contents of each of the cells according to bio-chemical reactions inside a membrane, and there is an exchange of molecules through the membranes. If one considers the state of the cell, i.e., the molecules inside the membranes, as a configuration, then the above mentioned reactions lead to a change of the configuration. Therefore we have something which looks as a computation. However, inside of each membrane we only have a finite multiset of objects; therefore the computation is not done via words, it is done via multisets.

In Figure 1 we give a cell by the outer skin membrane 1 containing two membranes 2 and 3 and the membrane 2 contains a further membrane 4. Moreover, the content of the cell itself is the multiset with the associated word *abb*, the contents of the three membranes 2, 3, and 4 inside the cell are the multisets/words *bc*, *aac*, and *abc* respectively.

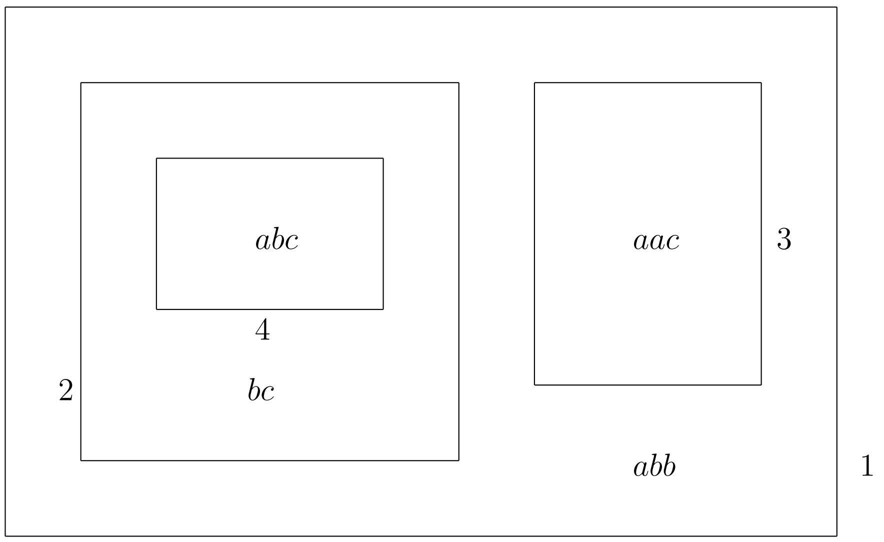
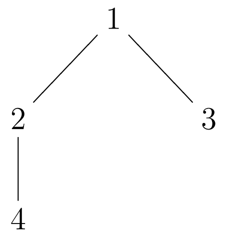


Figure 1.1: A membrane structure

The first problem is to describe the membrane structure. This can be done by a tree, where the outer skin membrane is the root and x is a son of y if and only if the membrane

y contains the membrane x. The membrane structure of the cell given in Figure 1.1 is then represented by



A further possibility to give a membrane structure is a correct sequence of indexed brackets where the index refers to the membrane. The outer membrane is represented by $[_1]_1$. If one has already a membrane structure where $[_i$ is followed by $]_i$, i.e., the sequence of brackets has the form $w[_i]_iw'$, and the i-th membrane contains the membranes $j_1, j_2, \ldots, j_s$, then we get a bracket word

$$w[_i[_{j1}]_{j1}[_{j2}]_{j2} \cdots [_{j_s}]_{j_s}]_i \ .$$

The structure given in Figure 1.1 is represented by $[_1[_2[_4]_4]_2[_3]_3]_1$.

A membrane is called *simple* if there is no membrane inside of it. In terms of trees which describe a membrane structure, the leaves correspond to simple membranes.

We also have to clarify the concept of a rule in a membrane system because we cannot only change a letter or a multiset of letters, i.e., a word, we can also move letters or multisets of letters through membranes. Obviously, a letter is kept in a membrane, or it can go out of the membrane, or it can move into a membrane which is inside the given membrane. Therefore we define the set Tar consisting of *here*, *out* and in_j where j refers to the j-th membrane. Thus we formally define a rule in a membrane system as a pair

$$(x_1x_2 \ldots x_n, (y_1, t_1)(y_2, t_2) \ldots (y_m, t_m))$$

where x_i and y_j are letters for $1 \leq i \leq n$ and $1 \leq j \leq m$, and $t_j \in Tar$ for $1 \leq j \leq m$. The application of this rule to the multiset $x_1x_2 \ldots x_n$ in membrane k is performed as follows: the multiset $x_1x_2 \ldots x_n$ is taken away from the multiset of membrane j, the letters y_q, $1 \leq q \leq m$,
— are added to the multiset in membrane j, if $t_q = here$,
— are added to the multiset in membrane k, if $t_q = out$ and membrane k contains membrane j,
— are given to the environment (and are lost) if $t_q = out$ and membrane j is the outer membrane,
— are added to the multiset in membrane p, if $t_q = in_p$ and membrane j contains membrane p. We note that, obviously, given a membrane j, the targets of the rules applicable to multisets in membrane j – besides *here* and *out* – can only be numbers of membranes which are contained in membrane j, i.e., which are sons of j in the tree describing the membrane structure. Moreover, *out* defines a unique membrane or the environment to which the letters have to go.

Again, we write $x_1x_2 \ldots x_n \rightarrow (y_1, t_1)(y_2, t_2) \ldots (y_m, t_m)$ for a rule.

In order to simplify the notation, we write a instead of $(a, here)$.

Thus we know, how to apply a rule. But in contrast to sequential grammars as context-free grammar or other phrase structure grammars, by the biological motivation, the rules have to be applied in parallel since some chemical reactions occur at the same moment. But we have a difference to L systems, too. In L systems, the rules are applied to letters – perhaps depending on the context – and thus the parallelism requires that to any object a rule has to be applied. In membrane systems, the rules are applied to multisets and thus it is possible that some objects remain to which no rule can be applied. For instance, let $p_1 = ab \to a(b, out)(a, in_3), p_2 = a \to bb, p_3 = bb \to (a, out)(a, in2)$ be the rules associated with the first membrane in the membrane structure given in Figure 1.1, where the membrane 1 contains the multiset represented by the word abb. If we apply the rule p_1 then we take from this multiset one occurrence of a and one occurrence of b such that one occurrence of b remains to which no rule is applicable. On the other hand, if we apply the second rule p_2, then two occurrences of b are not involved, and we can apply p_3 parallel to p_2. We require that we apply all rules in such a way that no rule is applicable to the remaining multiset. This is formally given in the following definition.

Definition 1.12 *Let a multiset M and a set $P = \{w_1 \to v_1, w_2 \to v_2, \ldots, w_m \to v_m\}$ of rules be given. We say that P is applied in a maximal parallel way iff the following conditions are satisfied:*
– M has a representation $w_M = w_{i_1}w_{i_2}\ldots w_{i_r}w'$,
– all rules $w_{i_k} \to v_{i_k}$, $1 \le k \le r$, are applied,
– no rule of P is applicable to w'.

Note that it is allowed that there is another representation $w_{j_1}w_{j_2}\ldots w_{j_s}w''$ of M with $s \neq r$ and/or $i_k \neq j_k$ and/or $w' \neq w''$ where we have to apply all rules $w_{j_k} \to v_{j_k}$, $1 \le k \le s$, and no rule of P is applicable to w''. Thus there is some nondeterminism in the definition of a maximal parallel derivation.

Before giving the formal definition of a membrane system we shortly discuss the problem of defining the generated language. Obviously, since the membranes contain multisets, only multisets can be generated. In a (context-free) grammar a derivation is finished iff the generated word contains only terminals, or in other words, no rule can be applied to the generated sentential forms. Therefore it is of interest to consider such multisets which are in the system if no rule is applicable. There are at least two possibilities for the choice of the generated multiset: take the union of all multisets present in the membranes or choose a special membrane and take the multiset in that membrane. We shall follow the second idea. Moreover, we shall not consider multisets, which count how often a letter occurs; we shall consider only the number of letters occurring in the multiset, that is the length of the word describing the multiset.

We now give the formal definition of a membrane system.

Definition 1.13 *i) A membrane system with m membranes is a $(2m+3)$-tuple*

$$\Gamma = (V, \mu, w_1, w_2, \ldots w_m, R_1, R_2, \ldots R_m, i)$$

where

- *V is a finite alphabet (of objects occurring in the membranes),*
- *μ is a membrane structure (of m membranes),*

- *for $1 \le j \le m$, w_j is a word over V (giving the initial content of membrane j),*
- *for $1 \le j \le m$, R_j is a finite set of rules which can be applied to words in membrane j,*
- *i is a natural number such that $1 \le i \le m$ and the membrane i is a simple membrane (the output membrane).*

ii) A configuration of Γ is an m-tuple of multisets/words.

For two configurations $C = (u_1, u_2, \ldots, u_m)$ and $C' = (u'_1, u'_2, \ldots, u'_m)$, we say that C is transformed to C' by Γ, written as $C \vdash C'$ if and only if C' is obtained from C by a maximal parallel application of rules of R_i to u_i for all i, $1 \le i \le m$, i.e., no rule of R_i can be applied to the multiset which remains after subtracting all sets to which rules are applied from u_i.

iii) A configuration $C = (u_1, u_2, \ldots, u_m)$ is called halting iff no rule of R_i is applicable to u_i for $1 \le i \le m$.

iv) The language $L(\Gamma)$ generated by a membrane system Γ is the set of all numbers n such that there is a halting configuration $C = (u_1, u_2, \ldots, u_m)$ of Γ with $|u_i| = n$.

We give two examples.

Example 1.14 We consider the membrane system

$$\Gamma_1 = (\{a, b, c\}, [_1[_2]_2]_1, a^2, \lambda, R_1, \emptyset, 2)$$

with

$$R_1 = \{a \to (a, here)(b, in_2)(c, in_2)^2,\ a^2 \to (a, out)^2\}.$$

Since, initially, we have two letters a in the membrane 1, we have two possibilities: we apply two times the rule $a \to (a, here)(b, in_2)(c, in_2)^2$ or we apply once the rule $a^2 \to (a, out)^2$. In the latter case both letters a are send in the environment and are lost such that the derivation stops since no further letters are in membrane 1. In the former case, two letters a remain in membrane 1 and two letter b and four letters c are send inside membrane 2. If we apply n times $a \to (a, here)(b, in_2)(c, in_2)^2$ and finish by one application of $a^2 \to (a, out)^2$, then we have finally $2n$ letters b and $4n$ letters c in membrane 2. Hence

$$L(\Gamma_1) = \{6n \mid n \ge 0\}.$$

Example 1.15 Let

$$\Gamma_2 = (\{A, B, D, E, X, Y, Z, a, \#\}, [_1[_2]_2]_1, XADE, \lambda, R_1, \emptyset, 2)$$

be a membrane systems with two membranes where

$$\begin{aligned} R_1 = \{ & XADE \to XBBDE,\ XE \to YE,\ AD \to \#, \# \to \#, \\ & YBDE \to YADE,\ YD \to YD,\ BE \to \#, \\ & YD \to Z,\ ZA \to Z(a, in_2)\ \} \end{aligned}$$

We note that any application of a rule requires an occurrence of X or Y or Z. the initial configuration contains one such letter, namely X, and each rule produces at most one such letter. Therefore only one such letter occurs in any configuration (and as we see below, hence we can only apply one rule of R_1 in each step). Furthermore, if the letter # is introduced by some rule, then we can apply the rule $\# \to \#$ at every moment and thus the system cannot reach a halting configuration, i.e., no word of $L(\Gamma_2)$ can be generated.

Let a configuration (XA^nDE, λ) with $n \geq 1$ be given: note that the initial configuration is given by $n = 1$. Then we cannot apply $XE \to YE$ since we also have to apply $AD \to \#$ by the maximal parallelism, which introduces #. This holds as long A is present in the first component of the configuration. Hence we get

$$(XA^nDE, \lambda) \vdash (XA^{n-1}B^2DE, \lambda) \vdash (XA^{n-2}B^4DE, \lambda) \vdash \ldots \vdash (XB^{2n}DE, \lambda).$$

Now we can use $XE \to YE$ (and only this rule is applicable) since it cannot be accompanied by $DA \to \#$. Thus we have $(YB^{2n}DE, \lambda)$. By arguments as above we have to replace all occurrences of B by A using the rule $YBDE \to YADE$. This yields $(YA^{2n}DE, \lambda)$. Now we have two cases for the continuation.

Case 1. We apply $YD \to XD$. Then we obtain the configuration $(XA^{2n}DE, \lambda)$ which has the form as the configuration from which we started and the process of doubling the A's can be iterated.

Case 2. We apply $YD \to Z$. We get $(ZA^{2n}E, \lambda)$. In this configuration only $ZA \to Z(a, in_2)$ is applicable. Thus we obtain

$$(ZA^{2n}E, \lambda) \vdash (ZA^{2n-1}E, a) \vdash (ZA^{2n-2}E, a^2) \vdash \ldots \vdash (ZE, a^{2n}).$$

The last configuration is a halting one and therefore a^{2n} belongs to $L(\Gamma_2)$. Therefore

$$L(\Gamma_2) = \{2^n \mid n \geq 1\}.$$

We ask the reader to note that the membrane systems Γ_2 works as the matrix grammar G_2. In both cases the introduction of # leads to a non-terminating derivation or only to non-halting configurations, and it is necessary to replace all A's or all B's, before X can be changed to Y or Y to X or Z, respectively.

A letter $c \in V$ is called a *catalyst* iff all rules where c occurs have the form $ca \to cw$ with $a \in V$ and $w \in (V \times Tar)^*$, i.e., the catalyst is not changed by the reaction, however, it is necessary that a can perform the change to w.

We say that a rule $u \to w$ with $w \in (V \times Tar)^*$ is called
– *non-cooperating* iff $u \in V$,
– *cooperating* iff $|u| \geq 2$,
– *catalytic* iff $u = ca$ and $w = cw'$ for some catalyst c, some $a \in V$ and some $w' \in (V \times Tar)^*$.
The notions non-cooperating and cooperating correspond to context-free and monotone in usual grammars. However since in a membrane system the words are interpreted as multisets we have no context in membrane systems and therefore we have only a cooperation between the letters of a multiset if the multiset is replaced.

We say that a membrane system is
– *non-cooperating* if all its rules are non-cooperating,

– *catalytic* if all its rules are non-cooperating or catalytic, and
– *catalytic* if it contains at least one rule which is cooperating and not catalytic.

By $\mathcal{L}_n(P, nco)$, $\mathcal{L}_n(P, cat)$, and $\mathcal{L}_n(P, coo)$ we denote the families of languages which can be generated by non-cooperating, catalytic, and cooperating membrane systems with at most n membranes, respectively. For $X \in \{nco, cat, coo\}$,

$$\mathcal{L}_*(P, X) = \bigcup_{n\geq 1} \mathcal{L}_n(P, X).$$

By definition, for $X \in \{nco, cat, coo\}$, we have

$$\mathcal{L}_1(P, X) \subseteq \mathcal{L}_2(P, X) \subseteq \mathcal{L}_3(P, X) \subseteq \ldots \subseteq \mathcal{L}_n(P, X) \subseteq \ldots \subseteq \mathcal{L}_*(P, X). \tag{1.1}$$

We first prove that the hierarchies given in (1.1) is finite for all X under consideration and has at most two levels.

Lemma 1.16 *For $X \in \{nco, cat, coo\}$ and $n \geq 2$,*

$$\mathcal{L}_1(P, X) \subseteq \mathcal{L}_2(P, X) = \mathcal{L}_n(P, X) = \mathcal{L}_*(P, X).$$

Proof. Obviously, by (1.1) it is sufficient to prove that $\mathcal{L}_*(P, X) \subseteq \mathcal{L}_2(P, X)$.

The idea of the proof consist in an indexing of letters in such a way that the index gives the membrane in which the letter is. Thus we set

$$V' = \{a_j \mid a \in V, 1 \leq j \leq m, j \neq i\}$$

and define for $1 \leq j \leq m$, $j \neq i$, the morphisms $h_j : V \to V'$ by $h(a) = a_j$.

Let $L \in \mathcal{L}_*(P, X)$. Then $L = L(\Gamma)$ for some membrane system Γ. Let

$$\Gamma = (V, \mu, w_1, w_2, \ldots, w_m, R_1, R_2, \ldots, R_m, i)$$

with $m \geq 3$ (if $m \leq 2$, then $L \in \mathcal{L}_2(P, X)$ by definition). We construct the membrane system

$$\Gamma' = (V' \cup V, [_1[_i]_i]_1, w_1', w_i, R_1', R_i', i)$$

with

$$w_1' = h_1(w_1)h_2(w_2)\ldots h_{i-1}(w_{i-1})h_{i+1}(w_{i+1})h_{i+2}(w_{i+2})\ldots h_m(w_m)$$

and R_1' and R_i' consisting of all rules which are constructed in the following way:

- If $u \to (b_1, t_1)(b_2, t_2)\ldots(b_s, t_s) \in R_k$ with $1 \leq k \leq m$, $k \neq i$, then $h_k(u) \to c_1c_2\ldots c_s \in R_1'$ where
  – $c_r = ((b_r)_k, here)$ if $t_r = here$
  – $c_r = ((b_r)_p, here)$ if $t_r = in_p$ and $p \neq i$,
  – $c_r = (b_r, in_i)$ if $t_r = in_i$,
  – $c_r = ((b_r)_l, here)$ if $t_r = out$ and l is the unique membrane which contains membrane k in μ.
- If $u \to (b_1, t_1)(b_2, t_2)\ldots(b_s, t_s) \in R_i$ with $1 \leq k \leq m$, $k \neq i$, then $h_k(u) \to c_1c_2\ldots c_s \in R_i'$ where
  – $c_r = (b_r, here)$ if $t_r = here$
  – $c_r = ((b_r)_{l'}, out)$ if $t_r = out$ and l' is the unique membrane which contains membrane i in μ.

By these definitions,

$$(v_1, v_2, \ldots, v_m) \vdash (v'_1, v'_2, \ldots, v'_m)$$

in Γ if and only if

$$(h_1(v_1) \ldots h_{i-1}(v_{i-1})h_{i+1}(v_{i+1}) \ldots h_m(v_m), v_i) \vdash (h_1(v'_1) \ldots h_{i-1}(v'_{i-1})h_{i+1}(v'_{i+1}) \ldots h_m(v'_m), v'_i)$$

in Γ'. Moreover, we have that $(v_1, v_2, \ldots, v_m)$ is a halting configuration of Γ if and only if $(h_1(v_1)h_2(v_2) \ldots h_{i-1}(v_{i-1})h_{i+1}(v_{i+1}) \ldots h_m(v_m), v_i)$ is a halting configuration of Γ'. Therefore the membrane i contains the same multisets if a halting configuration is obtained. Thus $L(\Gamma) = L(\Gamma')$. This implies $L = L(\Gamma') \in \mathcal{L}_2(P, X)$. $\square$

We now prove that Lemma 1.16 can be improved for *nco* and *coo* to $n \geq 1$. Moreover, we characterize $\mathcal{L}_*(P, nco)$ and $\mathcal{L}_*(P, coo)$.

Theorem 1.17 *For all* $n \geq 1$, $\mathcal{L}_1(P, nco) = \mathcal{L}_n(P, nco) = \mathcal{L}_*(P, nco) = N(CF)$.

Proof. By (1.1) and Lemma 1.16, it is sufficient to prove that $N(CF) \subseteq \mathcal{L}_1(P, nco)$ and $\mathcal{L}_2(P, nco) \subseteq N(CF)$.

Let $L \in N(CF)$. Then there is a context-free language L' such that $L = N(L')$. Let G be a context-free grammar generating L'. We construct the membran system $\Gamma = (N \cup T, [_1]_1, S, P, 1)$. Note that the rules of P in Γ are a short writing of rules where the target is *here* in all cases. It is obvious that a derivation $S \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow \ldots \Longrightarrow w_n$ in G corresponds to $(S) \vdash (w_1) \vdash (w_2) \vdash \ldots \vdash (w_n)$ in Γ (any configuration has only one component). Moreover, $z \in L(G)$ iff $z \in T^*$ iff no rule is applicable in G iff (z) is a halting configuration. Hence $L(\Gamma) = N(L(G)) = N(L') = N$. This proves $N(CF) \subseteq \mathcal{L}_1(P, nco)$.

Let $L = L(\Gamma)$ for some membrane system with 2 membranes, i.e.,

$$\Gamma = (V, [_1[_2]_2]_1, w_1, w_2, R_1, R_2, 2).$$

For $i \in \{1, 2\}$, let F_i be the set of all letters $a \in V$ such that there is no rule with left-hand side a in R_i. Without loss of generality we assume that w_1 contains no letter of F_1 since such letters cannot be changed by Γ, and therefore they are superfluous for $L(\Gamma)$. We set

$$V_i = \{a_i \mid a \in V\} \text{ and } V'_i = \{a'_i \mid a \in V\},$$

the define the homomorphisms

$$h_i : V \to V'_i,\ g_1 : V \times \{here, out, in_2\} \to F_2 \cup V'_1 \cup V'_2 \text{ and } g_2 : V \times \{here, out\} \to F_2 \cup V'_1 \cup V'_2$$

by

$$\begin{aligned}
h_i(a) &= a'_i, \\
g_1((b, here)) &= \begin{cases} \lambda & \text{if } b \in F_1 \\ b'_1 & \text{otherwise,} \end{cases} \\
g_1((b, out) &= \lambda, \\
g_1((b, in_2)) &= \begin{cases} b & \text{if } b \in F_2 \\ b'_2 & \text{otherwise,} \end{cases}
\end{aligned}$$

$$g_2((b,here)) = \begin{cases} b & \text{if } b \in F_2 \\ b_2' & \text{otherwise,} \end{cases}$$
$$g_2((b,here)) = \begin{cases} \lambda & \text{if } b \in F_1 \\ b_1' & \text{otherwise,} \end{cases}$$

and the grammar system $G = (N, V \setminus F_2, P_1, P_2, S)$ with two components by

$$\begin{aligned} N &= V_1 \cup V_2 \cup V_1' \cup V_2' \cup \{S\} \\ P_1 &= \{S \to h_1(w_1)h_2(w_2)\} \cup \{a_i \to g_i(x) \mid a \to x \in R_i, 1 \le i \le 2\}, \\ P_2 &= \{a_i' \to a_i \mid a \in V, 1 \le i \le 2\}. \end{aligned}$$

A configuration (w_1v, w_2u) of Γ with $w_1 \in (V \setminus F_1)^*$, $v \in F_1^*$, $w_2 \in (V \setminus F_2)^*$ and $u \in F_2^*$ is described in the grammar system G by a word $h_1(w_1)h_2(w_2)u$. Such a word cannot be processed by the first component of G and the second component of G cancels all the primes, i.e., we obtain the word v_1v_2u where v_1 is the variant of w_1 where all letters have the index 1 and v_2 is the variant of w_2 where all letters have the index 2. The first component of G transforms a word v_1v_2u with $v_1 \in V_1^*$ and $v_2 \in V_2^*$ in u_1u_2 where u_1 and u_2 are the indexed and primed versions of w_1' and w_2' with $(w_1, w_2u) \vdash (w_1', w_2'u)$ besides the letters of F_1 which are cancelled since they do not contribute to Γ and the letters of F_2 which remain in the second membrane. Therefore there are words $z_1 \in (V \setminus F_1)^*$, $z \in F_1^*$, $z_2 \in (V \setminus F_2)^*$ and $u' \in F_2^*$ such that $w_1' = z_1z$, $w_2 = z_2u'$ and

$$h(w_1)h(w_2)u \Longrightarrow_{P_1} v_1v_2u \Longrightarrow_{P_2} h_1(z_1)h_2(z_2)u'u$$

in G. Moreover, the derivation stops in G if and only if all letters belong to F_2, and a halting configuration in Γ is obtained if and only if all letters in membrane 1 belong to F_2 and all letters in membrane 2 belong to F_2. Taking into consideration that the letters of F_1 are cancelled in G, we obtain that $L(\Gamma) = N(L(G))$. By Theorem 1.10 i), $L(G)$ is a context-free language. Hence $N(L(G)) \in N(CF)$. Therefore we have $L(\Gamma) \in N(CF)$ and $\mathcal{L}_2(P, nco) \subseteq N(CF)$ is shown. □

Theorem 1.18 *For all $n \ge 1$, $\mathcal{L}_1(P, coo) = \mathcal{L}_n(P, coo) = \mathcal{L}_*(P, coo) = N(RE)$.*

Proof. By (1.1) it is sufficient to prove that $N(RE) \subseteq \mathcal{L}_1(P, coo)$.

Let $L \in N(RE)$. By Theorem 1.4, there is a matrix grammar $G = (N, T, M, S, F)$ such that $L = N(L(G))$. By Theorem 1.6, we can assume that G is in normal form. We construct the membrane system

$$\Gamma = (N_1 \cup N_2 \cup T \cup \{S, Z, \#, H, H', H''\} \cup \{H_A \mid A \in N_2\}, [_1]_1, S, R_1, 1)$$

with R_1 consisting of all rules of the forms

(1) $S \to HXA$ for $(S \to XA) \in M$,

(2) $HXA \to HYx$ for $(X \to Y, A \to x) \in M$,

(3) $HX \to H'H_AY,\ H_AA \to \#,\ \# \to \#, H' \to H'',\ H''H_A \to H$
for $(X \to Y, A \to \#) \in M$,

(4) $HZ \to \lambda$

Obviously, we have $S \Longrightarrow XA$ in G and $(S) \vdash (HXA)$ in Γ, i.e., besides the additional symbol H we have simulated a derivation step of G.

If we have a sentential form $w = Xw_1Aw_2$ in G, then we can apply a matrix of the form $(X \to Y, A \to x)$ and obtain Yw_1xw_2. In Γ we simulate this by applying $HXA \to HYx$ to HXw_1Aw_2 which gives HYw_1xw_2, i.e., the simulation is correct.

The matrix $(X \to y, A \to \#)$ is only applicable to Xw if A does not occur in w and results in Yw. Accordingly, if we apply $HX \to H'H_AY$ to Xw, we get $H'H_AYw$. If A is present, i.e. $w = w_1Aw_2$, we have to apply $H' \to H''$ and $H_AA \to \#$ in parallel (maximal parallelism) and get $H''\#w_1w_2$. However, now $\# \to \#$ can be applied at any moment and thus we cannot come to a halting configuration. If A is not present, we get

$$H'H_AYw \vdash H''H_AYw \vdash HYw,$$

i.e., again, the application of $(X \to y, A \to \#)$ is correctly simulated by the rules of (3).

If $Z \to \lambda$ is used in G we simulate this by $HZ \to lambda$.

By these explanations it follows that $L(\Gamma) = N(L(G)) = L$ and thus $L \in \mathcal{L}_1(P, coo)$ which proves $N(RE) \subseteq \mathcal{L}_1(P, coo)$.

□

For catalytic systems the situation is different since the hierarchy has two levels.

Theorem 1.19 *For all* $n \geq 2$, $\mathcal{L}_1(P, cat) \subset \mathcal{L}_2(P, cat) = \mathcal{L}_n(P, cat) = \mathcal{L}_*(P, cat) = N(RE)$.

Proof. We omit the proof of the strictness of the inclusion $\mathcal{L}_1(P, cat) \subset \mathcal{L}_2(P, cat)$.

Let L be the length set of a recursively enumerable language. Then there is a matrix grammar $G = (N_1 \cup N_2 \cup \{S, Z, \#\}, T, M, S, F)$ in normal form (see Definition 1.5 and Theorem 1.6) such that $L = N(L(G))$. Let $(X_i \to Y_i, A_i \to w_i)$, $1 \leq i \leq s$, and $(X_j \to Y_j, A_j \to \#)$, $s + 1 \leq j \leq t$, be the matrices of M which consist of two rules. We note that any sentential form of G, which does not contain S or Z, contains exactly one element of N_1. Since for membrane systems the order of the letters in a word is not important, we assume without loss of generality that, for $1 \leq i \leq s$, $w_i = w_i'w_i''$ with $w_i' \in N_2^*$ and $w_i'' \in T^*$. If all elements of the multiset/word $w = a_1a_2\ldots a_n$ are sent in a membrane i, we write (w, in_i) instead of $(a_1, in_i)(a_2, in_i)\ldots(a_n, in_i)$. We construct the catalytic membrane system

$$\Gamma = (V, [_1[_2]_2]_1, S, \lambda, R_1, \emptyset, 2)$$

with

$$\begin{aligned}
V &= N_1 \cup N_2 \cup \{S, Z, \#, \$_1, \$_2, \$_3, r, c_0, c, \S\} \cup \bigcup_{i=1}^{t}\{r_i, r_i', c_i, Q_i, Q_i'\} \cup \bigcup_{i=1}^{s}\{Q_i''\}, \\
R_1 &= \{S \to c\S c_0c_1\ldots c_t\S_1 r^t XA \mid (S \to XA) \in M\} \\
&\quad \cup \{c\S \to c, c\$_2 \to c, c_0r \to c_0\S, c_0Z \to c_0, \$_1 \to \$_2, \$_2 \to \$_3, \$_3 \to \$_1, \\
&\qquad\qquad r \to \#, \# \to \#, \S \to \#\} \\
&\quad \cup \{r_i \to \#, X_i \to \#, Q_i' \to Y_i, c_iX_i \to c_iX_i'r_i^t, c_ir \to c_i, c_0r_i \to c_0r_i'\S, \\
&\qquad\qquad c_0r_i' \to c_0r\S, c_i\$_2 \to c_i\# \mid 1 \leq i \leq t\}
\end{aligned}$$

$$\cup \{c_i r_j \to c_i r'_j, c_i r'_j \to c_i r \mid 1 \le i \le t, 1 \le j \le t, i \ne j\}$$
$$\cup \{Q_i \to Q'_i, c_i A_i \to c_i Q''_i, Q''_i \to w'_i(w''_i, in_2) \mid 1 \le i \le s\}$$
$$\cup \{c_i Q_i \to c_i Q'_i, c_i A_i \to c_i \# \mid s+1 \le i \le t\}.$$

The catalysts are given by $c, c_0, c_1, \dots, c_t$.

Let us consider a configuration $(c\S c_0 c_1 \dots c_t r^t \S_1 X_i A_i z_1, z_2)$ and assume that it corresponds to a sentential form which is up the order of the letters $X_i A_i z_1 z_2$ with $z_1 \in N_2^*$ and $z_2 \in T^*$.

We first discuss the case $i \le s$. To avoid the application of a rule $\S \to \#$, $r \to \#$, and $X_i \to \#$ we have to use the rules $c_0 r \to c_0 \S$, $c_j r \to c_j$ for $1 \le j \le t$, $j \ne i$, and $c_i X_i \to c_i Q_i r_i^t$ (and $\$_1 \to \$_2$). Thus we get

$$(c\S c_0 c_1 \dots c_t r^t \S_1 X_i A_i z_1, z_2) \vdash (c\S c_0 c_1 \dots c_t r_i^t \S_2 Q_i A_i z_1, z_2).$$

By similar reasons, we now obtain

$$(c\S c_0 c_1 \dots c_t r_i^t \S_2 Q_i A_i z_1, z_2) \vdash (c\S c_0 c_1 \dots c_t (r'_i)^t \S_3 Q'_i Q''_i z_1, z_2)$$

(note that we introduce $\#$ via $c_i \$_2 \to c_i \#$ if no A_i is present; in this case $(X_i \to Y_i, A_i \to w_i)$ is not applicable), and then

$$(c\S c_0 c_1 \dots c_t (r'_i)^t \S_3 Q'_i Q''_i z_1, z_2) \vdash (c\S c_0 c_1 \dots c_t r t \S_1 Y_i w'_i z_1, w''_i z_2)$$

(note that c_i is involved in no applied rule). The latter configuration corresponds to $Y_i w'_i z_1 w''_i z_2$ which is obtained from $X_i A_i z_1 z_2$ by application of $(X_i \to Y_i, A_i \to w_i)$.

If $s+1 \le i \le t$, we want to simulate the application of $(X_i \to Y_i, A_i \to \#)$. Thus A_i has not to be present in the sentential form (since otherwise $\#$ is produced in the matrix grammar and we cannot terminate). We get the following sequence of configurations:

$$\begin{aligned}(c\S c_0 c_1 \dots c_t r^t \S_1 X_i z_1, z_2) &\vdash (c\S c_0 c_1 \dots c_t r_i^t \S_2 Q_i z_1, z_2)\\ &\vdash (c\S c_0 c_1 \dots c_t (r'_i)^t \S_3 Q'_i z_1, z_2)\\ &\vdash (c\S c_0 c_1 \dots c_t r^t \S_1 Y_i z_1, z_2).\end{aligned}$$

If A_i is present, then we introduce $\#$ in the last step via $c_i A_i \to c_i \#$; if A_i is not present, then c_i is not involved in the last step. Obviously, the corresponding sentential form $Y_i z_1 z_2$ is obtained from $X_i z_1 z_2$ by application of $(X_i \to Y_i, A_i \to \#)$.

It remains the case where Z is present. Then our configuration is $(c\S c_0 c_1 \dots c_t r^t \S_1 Z, z)$, and the sentential form is Zz with $z \in T^*$. We obtain the transformations

$$c\S c_0 c_1 \dots c_t r^t \S_1 Z, z) \vdash (c c_0 c_1 \dots c_t \$_2, z) \vdash (c c_0 c_1 \dots c_t, z)$$

and the derivation $Zz \Longrightarrow z$ by application of $(Z \to \lambda)$. In both cases we have a halting configuration. Thus $L(\Gamma) = N(L(G)) = L$. □

For completeness we remark that we used $t+2$ catalysts where t is number of matrices consisting of two rules in the matrix grammar in normal form. In [10], it has been shown that two catalysts are sufficient to generate all recursively enumerable languages. It is open whether one catalyst is sufficient.

1.3 Membrane Systems with Symport/Antiport Rules

In this section we discuss membrane systems without the ability of changing objects. Hence only the moving through the membranes can be used for computation. Thus we have a process which only works by the exchange of information. Hence the study of these systems is also of interest from an information-theoretic point of view, since it is investigated the power of communication.

In biology it is known that there are many cases where two chemicals pass through a membrane at the same time with the help of each other. Both chemicals go in the same direction (this is called symport) or in opposite direction (called antiport). Formally, such movements can be written as (ab, in) or (ab, out) in symport case where both chemicals come in or leave out a membrane, respectively, or as $(b, out; a, in)$ denoting that a comes in and b leaves a given membrane.

Obviously, if one considers membrane systems where any rule is a symport or antiport rule, then no change of the involved chemicals occurs, and therefore finitely many objects are only moving around, which gives only a strongly limited power. Therefore we add symbols in the environment and assume that an infinite number of copies of each of these symbols is present in the environment.

We now give the formal definition of a membrane system with symport/antiport rules.

Definition 1.20 *i) A membrane system with m membranes and symport/antiport rules is a construct*

$$\Gamma = (V, \mu, E, w_1, w_2, \dots, w_m, R_1, R_2, \dots R_m, i)$$

where V, μ, $w_1, w_2, \dots w_m$, $R_1, R_2, \dots, R_m$ and i are specified as in membrane system, E is a subset of V and, for $1 \leq j \leq m$, R_j is a finite set of rules of the form (x, in) or (x, out) or $(x, out;, y, in)$ with $x, y \in V^+$.

ii) A configuration of a membrane system with symport/antiport rules is a m-tuple $C = (u_1, u_2, \dots, u_m)$ of words (or equivalently, multisets) over V.

Let j, $1 \leq j \leq m$, be a membrane and let j' be the unique membrane which contains membrane j. The application of a rule (x, in) of R_j to C results in taking the multiset x out $c_{j'}$ and adding to c_j; the application of (x, out) is performed by subtracting x from c_j and adding to $c_{j'}$; the application of $(x, out; y, in)$ consists in a parallel application of (x, out) and $y, in)$ as described. If j is the outer membrane, then E takes the rule of membrane j' where any element of E is present in E infinitely often.

The transformation of a configuration C into a configuration C' (written as $C \vdash C'$) is done by a maximal parallel application of the rules of all R_j, $1 \leq j \leq m$, to C.

A configuration C is called halting if no rules from the sets R_j, $1 \leq j \leq m$, can be applied to C

iii) The language $L(\Gamma)$ generated by a membrane system Γ with symport/antiport rules is the set of all numbers n such that there is a halting configuration $C = (u_1, u_2, \dots, u_m)$ of Γ with $|u_i| = n$.

Example 1.21 We consider the membrane system

$$\Gamma = (V, [_1[_2]_2]_1, E, ac, df, R_1, R_2, 2)$$

with

$$\begin{aligned}
V &= \{a,b,c,c',d,e,e',f,g,\#\},\\
E &= \{a,b,c,c',e,e',f,g,\#\},\\
R_1 &= \{(c,out;\#,in),\ (ca,out;cbb,in),\ (ca,out,c'bb,in),\ (da,out;\#,in)\\
&\qquad (c'd,out;e,in),\ (eb,out;ea,in),\ (eb,out;e'a,in),\ (fb,out;\#,in),\\
&\qquad (e'f,out;cdf,in),\ (e'f,out;g,in)\},\\
R_2 &= \{(d,out;c',in),\ (c',out),\ (f,out;e',in),\ (e',out),\ (df,in),\\
&\qquad (\#,in),\ (\#,out),\ (ga,in),\ (g,out)\}.
\end{aligned}$$

First we mention that the introduction of # is forbidden, again, because by the rules $(\#, out)$ and $(\#, in)$ in R_2 the symbol # can alternately moved from membrane 2 to membrane 1 and conversely such that no halting configuration can be reached. Therefore we have the following sequence of configurations (note that the case $n = 1$ is given initially)

$$\begin{array}{rcll}
(ca^n,df) & \vdash & (cbba^{n-1},df) & \text{by } (ca,out;cbb,in)\in R_1\\
& \vdash & (cb^4a^{n-2},df) & \text{by } (ca,out;cbb,in)\in R_1\\
& & \vdots & \\
& \vdash & (cb^{2n-2}a,df) & \text{by } (ca,out;cbb,in)\in R_1\\
& \vdash & (c'b^{2n},df) & \text{by } (ca,out;c'bb,in)\in R_1\\
& \vdash & (db^{2n},c'f) & \text{by } (d,out;c',in)\in R_2\\
& \vdash & (dc'b^{2n},f) & \text{by } (c',out)\in R_2\\
& \vdash & (eb^{2n},f) & \text{by } (dc',out;e,in)\in R_1\\
& \vdash & (eab^{2n-1},f) & \text{by } (eb,out;ea,in)\in R_1\\
& \vdash & (ea^2b^{2n-2},f) & \text{by } (eb,out;ea,in)\in R_1\\
& & \vdots & \\
& \vdash & (ea^{2n-1}b,f) & \text{by } (eb,out;ea,in)\in R_1\\
& \vdash & (e'a^{2n},f) & \text{by } (eb,out;e'a,in)\in R_1\\
& \vdash & (fa^{2n},e') & \text{by } (f,out;e',in)\in R_2\\
& \vdash & (e'fa^{2n},\lambda) & \text{by } (e',out)\in R_1.
\end{array}$$

Now we have two possibilities of continuation:

$$\begin{array}{rcll}
(e'fa^{2n},\lambda) & \vdash & (cdfa^{2n},\lambda) & \text{by } (e'f,out;cdf,in)\in R_1\\
& \vdash & (cbba^{2n-1},df) & \text{by } (ca,out;cbb,in)\in R_1,\ (df,in)\in R_2
\end{array}$$

which means that, essentially, we have doubled the number of occurrences of a in membrane 1 and can iterate this process, or

$$\begin{array}{rcll}
(e'fa^{2n},\lambda) & \vdash & (ga^{2n},\lambda) & \text{by } (e'f,out;g,in)\in R_1\\
& \vdash & (a^{2n-1},ga) & \text{by } (ga,in)\in R_2\\
& \vdash & (ga^{2n-1},a) & \text{by } (g,out)\in R_2\\
& \vdash & (a^{2n-2},ga^2) & \text{by } (ga,in)\in R_2\\
& \vdash & (ga^{2n-2},a^2) & \text{by } (g,out)\in R_2\\
& & \vdots & \\
& \vdash & (\lambda,ga^{2n}) & \text{by } (ga,in)\in R_2\\
& \vdash & (g,a^{2n}) & \text{by } (g,out)\in R_2
\end{array}$$

and a halting configuration is obtained. Therefore

$$L(\Gamma) = \{2^n \mid n \geq 1\}.$$

We now prove that membrane systems with symport/antiport rules, i.e., membrane systems which only work on the basis of communication, are able to generate all recursively enumerable sets of numbers.

Theorem 1.22 *For any set $L \in N(RE)$, there is a membrane system Γ with symport/antiport rules such that $L(\Gamma) = L$.*

Proof. By Theorems 1.4 and 1.6 there is a matrix grammar $G = (N, T, M, S, F)$ in normal form such that $L = N(L(G))$. Let $(S \to X'A')$ be the only matrix in G of this form. Let M have n matrices of the form $(X \to Y, A \to x)$ or $(X \to Y, A \to \#)$.

We define the membrane system

$$\Gamma = (V, [_1[_2]_2]_1, E, cX'A', \lambda, R_1, R_2, 2)$$

with

$$\begin{aligned}
V &= N_1 \cup N_2 \cup T \cup \{c, g, h, Z, \#\} \cup \bigcup_{i=1}^{n} \{c_i, c_i', d_i\},\\
E &= N_1 \cup N_2 \cup T \cup \{g, h, Z, \#\} \cup \bigcup_{i=1}^{n} \{c_i, c_i', d_i\},\\
Q_i &= \{(cX, out; c_iY, in),\ (c_iA, out; cc_i', in),\ (c_i', out; x, in),\ (c_i, out; \#, in)\}\\
&\qquad \text{for } m_i = (X \to Y, A \to x),\\
Q_i &= \{(cX, out; c_id_i, in),\ (d_i, out; Yh, in),\ (c_iA, out; \#, in),\ (h, out; cg, in),\ (c_ig, out)\}\\
&\qquad \text{for } m_i = (X \to Y, A \to \#),\\
R_1 &= \{(c, out; \#, in),\ (cZ, out)\} \cup \bigcup_{i=1}^{n} Q_i,\\
R_2 &= \{(\#, in),\ (\#, out)\} \cup \{(a, in) \mid a \in T\}.
\end{aligned}$$

We note, again, that introducing the symbol # does not allow reaching of a halting configuration since it can be move from the second membrane to the first membrane or conversely at every moment.

Moreover, the second membrane only collects the terminals occurring at some moment in the first membrane.

Therefore we now consider only the first component of a configuration. If it has the form $cXAw$ (as it is the case for the initial configuration), then we can without introducing # only perform the following steps

$$cXAw \vdash c_iYAw \vdash cc_i'Yw \vdash cYxw$$

which means that we have correctly applied the simulation of the matrix $m_i = (X \to Y, A \to x)$ and can proceed with the simulation of a further matrix.

If the configuration is cXw and A does not occur in w, then the following steps have to be done

$$cXw \vdash c_i d_i w \vdash c_i Y hw \vdash c_i Y cgw \vdash cYw$$

which is a correct simulation of the application of $(X \to Y, A \to \#)$.

Moreover, in both cases we stop if no nonterminal is present in the sentential form or in the first membrane (since the configuration (cZw, x) is only reachable if $w \in \lambda$). Therefore $L(\Gamma) = N(L(G)) = L$. $\square$

Bibliography

[1] L. M. Adleman, Molecular computation of solutions to combinatorial problems. *Science* **226** (1994) 1021-1024.

[2] C. Calude and Gh. Păun, *Computing with Cells and Atoms.* Taylor and Francis, London, 2001.

[3] J.W. Carlyle, S. Greibach and A. Paz, A two-dimensional generating system modelling growth and binary cell division. In: *Proc. 15th Annual Symp. Switching Automata Theory*, 1–12, 1974.

[4] E. Csuhaj-Varju, J. Kelemen, A. Kelemenova and Gh. Păun, Eco(grammar)systems - a grammatical framework for lifelike interaction. *Artificial Life* **24** (1997) 1–28.

[5] J. Dassow, Grammars with valuations - a discrete model for self-organization of biopolymers. *Discr. Appl. Math.* **4** (1982) 161–174.

[6] J. Dassow, *Theoretische Informatik*, Vorlesungsskript, Otto-von-Geuricke-Universität Magdeburg, 2001.

[7] J. Dassow, V. Mitrana and A. Salomaa, Operations and language generating devices suggested by genome evolution. *Theor. Comp. Sci.* **270** (2002) 701–738.

[8] J. Dassow and Gh. Păun, *Regulated Rerwriting in Formal Language Theory.* Springer-Verlag, Berlin, 1989 and Akademie-Verlag, Berlin, 1989.

[9] J. Dassow and Gy. Vaszil, Multiset splicing systems. *BioSystems* **74** (2004) 1–7.

[10] R. Freund, L. Kari, M. Oswald and P. Sosik, Computationally universal P systems without priorities: two catalysts are sufficient. *Theor. Comp. Sci.* **330** (2005) 251–266.

[11] T. Head, Formal language theory and DNA: An analysis of the generative capacity of specific recombinant behaviors. *Bull. Math. Biology* **49** (1987) 737–759.

[12] T. Head, Gh. Păun and D. Pixton, Language theory and molecular genetics. In: [30], Vol. II, Chapter 7, 295–360

[13] G.T. Herman and G. Rozenberg, *Developmental Systems and Languages.* North-Holland Publ. Co., Amsterdam, 1975.

[14] D. Janssens, G. Rozenberg and R. Verraedt, On sequential and parallel node-rewriting graph grammars. Part I, *Computer Graphics and Image Processing* **18** (1982) 279–304 and Part II, *Computer Vision, graphics and Image Processing* **23** (1983) 295–312.

[15] L. Kari, G. Rozenberg and A. Salomaa, L systems. In: [30], Vol. I, Chapter 5, 253–328.

[16] A. Lindenmayer, Mathematical models for cellular interaction in development I and II. *J. Theoret. Biol.* **18** (1968) 280–315.

[17] A. Lindenmayer and G. Rozenberg (eds.), *Automata, Languages, Development.* North-Holland Publ. Co., 1976.

[18] R. J. Lipton, Using DNA to solve NP-complete problems. *Science* **268** (1995) 542–545.

[19] J. Opatrny and K. Culik II, Time complexity of recognition and parsing of E0L languages. In [17], 243–250.

[20] A. Păun and Gh. Păun, The power of communication: P systems with symport and antiport. *New Generation Computing* **20** (2002) 295–306.

[21] Gh. Păun, Computing with membranes. *J. Comp. System Sci.* **61** (2000) 108–143.

[22] Gh. Păun, *Membrane Computing.* Springer-Verlag, Berlin, 2002.

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

[24] Gh. Păun and A.Salomaa, DNA computing based on the splicing operation. *Mathematica Japonica* **43** (1996) 607–632.

[25] P. Pruzinkiewicz and A. Lindenmayer, *The Algorithmic Beauty of Plants.* Springer-Verlag, Berlin, 1990.

[26] G. Rozenberg and A.Salomaa (eds.), *L Systems.* LNCS 15, Springer-Verlag, Berlin, 1974.

[27] G. Rozenberg and A.Salomaa, *The Mathematical Theory of L Systems.* Academic Press, New York, 1980.

[28] G. Rozenberg and A.Salomaa (eds.), *The Book of L.* Springer-Verlag, Berlin, 1985.

[29] G. Rozenberg and A.Salomaa (eds.), *Lindenmayer Systems.* Springer-Verlag, Berlin, 1992.

[30] G. Rozenberg and A.Salomaa (eds.), *Handbook of Formal Languages.* Vol I – III, Springer-Verlag, 1997.

[31] A. Salomaa, *Formal Languages*. Academic Press, New York, 1973 (german edition, Springer-Verlag, Berlin, 1978).

[32] A. Salomaa, *Jewels of Formal Language Theory*. Computer Science Press, Rockville, 1981.