Journal of Universal Computer Science, vol. 5, no. 2 (1999), 33-49
submitted: 3/12/98, accepted: 5/2/99, appeared: 28/2/99 © Springer Pub. Co.

On the Power of Membrane Computing

Jürgen DASSOW
Fakultät für Informatik
Otto-von-Guericke-Universität Magdeburg
PSF 4120, D-39016 Magdeburg, Germany
E-mail: dassow@iws.cs.uni-magdeburg.de

Gheorghe PĂUN
Institute of Mathematics of the Romanian Academy
PO Box 1 – 764, 70700 Bucureşti, Romania
E-mail: gpaun@imar.ro

Abstract: We continue the investigation of the power of the computability models introduced in [12] under the name of *transition super-cell systems*. We compare these systems with classic mechanisms in formal language theory, context-free and matrix grammars, E0L and ET0L systems, interpreted as generating mechanisms of number relations (we take the Parikh image of the usual language generated by these mechanisms rather than the language). Several open problems are also formulated.

1 Introduction

A super-cell system is a distributed parallel computing model inspired from biochemistry and recently introduced in [12]. It is based on the notion of a *membrane structure*, of a kind similar to that used by the chemical abstract machine of [3]. Such a structure consists of several cell-like membranes, hierarchically embedded. Objects are placed in its regions, subject to given evolution rules. These rules are of the type used in multiset rewriting systems, see [1], [2]. An object can be transformed in other objects, can pass through adjacent membranes, and can dissolve the membrane in which it is placed. Such a construct leads to a computing device: start from an initial configuration and evolve until a halting configuration is obtained; (the number of) the objects in a specified output membrane is the result of the computation.

The objects can evolve in dependence to each other (we then say that the system is *cooperating*), or independently. A particular case of cooperating systems is that where one considers *catalysts*, certain objects which are allowed to participate to the evolution of other objects, but without being modified by this operation. It is proved in [12] that super-cell systems with catalysts, without using the membrane dissolving action, of degree two (composed of two membranes only) characterize the family of recursively enumerable sets of natural numbers.

The non-cooperating case is not investigated in [12]. We consider here systematically the eight possibilities: with or without priorities, with or without cooperation, with or without the membrane dissolving action. The eight hierarchies on the number of membranes used in a system are compared to classes of number relations associated with languages generated by devices in the Chomsky hierarchy and in the Lindenmayer area. Five of these hierarchies are proved to collapse. In the cooperating case when priorities are used, we give a proof of the equivalence with Turing machines shorter than the corresponding proof given in [12] for the weaker case of systems with catalysts (of course, systems with catalysts are particular cases of cooperating systems). The membrane dissolving action is not used, so in this way we settle the case of two hierarchies. When no priority, no cooperation, and no dissolving action is used, we get exactly the Parikh sets of context-free languages. This is a third hierarchy which we know that collapses. In general, when the dissolving action is not used, the region where an object is placed at any time can be identified by a subscript of its "name", hence again the hierarchy collapses (at level two, because we need an output membrane). This covers two more cases, different from the previously mentioned ones. For the other three hierarchies, this problem remains *open*. Also the properness of most of the inclusions we prove here remains *open*.

2 Super-cell Systems

We do not give a formal definition of a super-cell system; the reader is referred to [12] for details (and illustrating examples). Rather, we prefer an informal definition. For elements of formal language theory we refer to [16]. Here we only specify some notations.

For an alphabet V, we denote by V^* the free monoid generated by V under the operation of concatenation; the empty string is denoted by λ. For $x \in V^*$ and $a \in V$, we denote by $|x|$ the length of x and by $|x|_a$ the number of occurrences of the symbol a in x. If $L \subseteq V^*$, then the *length set* of L is $length(L) = \{|x| \mid x \in L\}$. The set of natural numbers is denoted by $\mathbf{N}$. If $V = \{a_1, \ldots, a_n\}$, then the *Parikh mapping* associated with V is $\Psi_V : V^* \longrightarrow \mathbf{N}^n$ defined by $\Psi_V(x) = (|x|_{a_1}, \ldots, |x|_{a_n})$ for $x \in V^*$. The mapping Ψ_V is extended in the natural way to languages. $\Psi_V(L)$ is called *the Parikh set* of $L \subseteq V^*$.

We denote by *REG, CF, CS, RE* the basic families in the Chomsky hierarchy: of regular, context-free, context-sensitive, and recursively enumerable languages, respectively. The family of Parikh sets of languages in a family F is denoted by PsF, while the family of length sets of languages in a family F is denoted by LsF.

A *multiset* over a given set U is a mapping $M : U \longrightarrow \mathbf{N}$. The set $\{a \in U \mid M(a) > 0\}$ is the *support* of M. The multiset with an empty support is said to be itself empty and denoted by $\emptyset$. Multisets of finite support are represented as strings which specify, in any order, the objects in the support and their multiplicities. The empty multiset is represented by the empty string λ.

A *membrane structure* is a construct consisting of several *membranes* placed in a unique "skin" membrane; we identify a membrane structure with a string of correctly matching parentheses, placed in a unique pair of matching parentheses. Graphically, a membrane structure is represented by a Venn diagram without intersection and with a unique superset (two sets can be either disjoint or one the subset of the other).

If in the regions delimited by the membranes we place multisets of objects from a specified finite set V, then we obtain a *super-cell*.

A *super-cell system* is a super-cell provided with evolution rules of objects and with a designated output membrane.

More formally, a *transition super-cell system* of degree $m, m \geq 1$, is a construct

$$\Pi = (V, \mu, M_1, \ldots, M_m, (R_1, \rho_1), \ldots, (R_m, \rho_m), i_0),$$

where:

(i) V is an alphabet; its elements are called *objects*;
(ii) μ is a membrane structure consisting of m membranes, with the membranes and the regions labeled in a one-to-one manner with elements in a given set Λ; here we always use the labels $1, 2, \ldots, m$;
(iii) $M_i, 1 \leq i \leq m$, are multisets over V associated with the regions $1, 2, \ldots, m$ of μ;
(iv) $R_i, 1 \leq i \leq m$, are finite sets of *evolution rules* over V associated with the regions $1, 2, \ldots, m$ of μ; ρ_i is a partial order relation over R_i, $1 \leq i \leq m$, specifying a *priority* relation among rules of R_i. This relation is conflictless (its graph contains no cycle).
An evolution rule is a pair (u, v), which we will usually write in the form $u \to v$, where u is a string over V and $v = v'$ or $v = v'\delta$, where v' is a string over

$$(V \times \{here, out\}) \cup (V \times \{in_j \mid 1 \leq j \leq m\}),$$

and δ is a special symbol not in V. The length of u is called *the radius* of the rule $u \to v$.
(v) i_0 is a number between 1 and m and it specifies the *output* membrane of Π.

When presenting the evolution rules, the indication "here" is often omitted.

Because we usually represent multisets by strings, we write $w_1, \ldots, w_m$ instead of $M_1, \ldots, M_m$, for $w_i \in V^*, 1 \leq i \leq m$.

If Π contains rules of radius greater than one, then it is said to be a system *with cooperation.*

The membrane structure and the multisets in Π constitute the *initial configuration* of the system. We can pass from a configuration to another one by using the evolution rules. This is done in parallel: all objects, from all membranes, which can be the subject of local evolution rules, as prescribed by the priority relation, should evolve simultaneously.

The priority checking is done as follows: we take a rule for which there is no rule of a higher priority and applicable and we assign to it the objects to which it can be applied; we repeat this operation with the rule of maximal priority which can be applied to the objects which were not assigned yet to rules (of course, the objects are assigned only once to a rule). That is, a rule can be used only if there are objects which are "free" in the moment when we check its applicability, and no rule of a higher priority can be applied at the same step, irrespective to which objects. Note that, because the priority relation is not circular, no conflict (loop) appears when checking it. (A possible variant is to use the priority relation only to settle the choice among rules which involve common objects in their left hand members, but this raises some difficulty in the cooperating case, because of the transitivity: consider $r_1 : ab \to v_1$, $r_2 : bc \to v_2$, and $r_3 : cd \to v_3$, and the relations $r_1 > r_2 > r_3$. Do we have to observe the relation $r_1 > r_3$, in spite of the fact that r_1 acts on ab and r_3 acts on the different objects cd ?)

The use of a rule $u \to v$ in a region with a multiset M means to subtract the multiset identified by u from M, then to follow the prescriptions of v: if an object appears in v in the form $(a, here)$, then it remains in the same region; if we have (a, out), then a copy of the object a will be introduced in the membrane immediately outside the region of the rule $u \to v$; if we have (a, in_i), then a copy of a is introduced in the membrane with the label i, providing that this membrane is adjacent to the region of the rule $u \to v$, otherwise the rule cannot be applied; if the special symbol δ appears in v, then the membrane which delimits the region of the rule $u \to v$ is dissolved; in this way, all the objects in this region become elements of the region placed immediately outside, while the rules of the dissolved membrane are removed. Several membranes can be dissolved at the same time, hence objects from several regions can come together in a higher region in one step.

We note the fact that we observe here no "conservation low". For instance, a rule of the form $a \to ab$ can be used an arbitrarily large number of times, producing an arbitrarily large number of copies of object b, without taking care of the "row materials" used for that. Otherwise stated, we use some biochemical inspiring details, but we work *in info* (as opposed to *in vivo* and to *in vitro*).

A sequence of transitions between configurations of a given super-cell system Π is called a *computation* with respect to Π. A computation is *successful* if and only if it halts, that is, there is no rule applicable to the objects present in the last configuration and the output membrane is present as an elementary membrane in this last configuration. The result of a successful computation is the total number of objects present in the output membrane of the halting configuration. We denote by $N(\Pi)$ the set of numbers computed by a super-cell system Π. We can also associate a relation to a system: consider a subset $T \subseteq V, T = \{a_1, \ldots, a_n\}$, as a designated output alphabet (and explicitly provided in the system: then we write the system in the form $\Gamma = (V, T, \mu, w_1, \ldots, w_m, (R_1, \rho_1), \ldots, (R_m, \rho_m), i_0))$; then *the Parikh set* computed by Π (with respect to T), denoted $P_T(\Pi)$, consists of all vectors $\Psi_T(x)$ for x describing the multiset present in the output membrane in a halting configuration.

The following example can clarify the way of defining the transition between configurations of a super-cell system, as well as the way of writing such systems. Consider the system:

$$
\begin{aligned}
\Pi &= (V, T, \mu, w_1, w_2, (R_1, \rho_1), (R_2, \rho_2), 2),\\
V &= \{a, b, c, d\},\\
T &= \{a, b\},\\
\mu &= [_1 [_2\ ]_2]_1,\\
w_1 &= ad,\\
w_2 &= \lambda,\\
R_1 &= \{r_1 : a \to aab,\ r_2 : a \to (a, in_2),\ r_3 : b \to (b, in_2),\\
&\qquad r_4 : c \to b,\ r_5 : d \to d,\ r_6 : d \to c\},\\
\rho_1 &= \{r_4 > r_1,\ r_5 > r_2,\ r_6 > r_2,\ r_5 > r_3,\ r_6 > r_3\},\\
R_2 &= \emptyset,\\
\rho_2 &= \emptyset.
\end{aligned}
$$

This is a super-cell system of degree two, of type $(Pri, nCoo, n\delta)$, with the output membrane labeled by 2.

The initial configuration of the system is better seen in the graphical representation from Figure 1. As long as the object d is present, the rules r_2, r_3 cannot be used. Thus, for evolving a we have to use the rule r_1, for all currently available copies of a. In this way, after n steps, we get 2^n copies of a and $2^n - 1$ copies of b. Let us suppose that in the meantime we have used $n-1$ times the rule r_5 and, at step n, the rule r_6. The multiset present in membrane 1 is (in the string representation) $a^{2^n} b^{2^n - 1} c$. Because rule r_4 can now be used, the rule r_1 cannot be used again, but all copies of a and b should now evolve using the available rules r_2, r_3, which can now be applied, because r_5 and r_6 aro no longer applicable. In this way, all copies of the objects a and b are sent to membrane 2. By using the rule r_4, in membrane 1 we produce one more copy of b; at the next step, this copy of b is also sent to the output membrane. No further transition can be done, the computation stops with the result $2 \cdot 2^n$. Therefore, $N(\Pi) = \{2^{n+1} \mid n \geq 1\}$.

If we distinguish the symbols in the output membrane, then the Parikh set computed by Π is $P_{\{a,b\}}(\Pi) = \{(2^n, 2^n) \mid n \geq 1\}$. $\square$

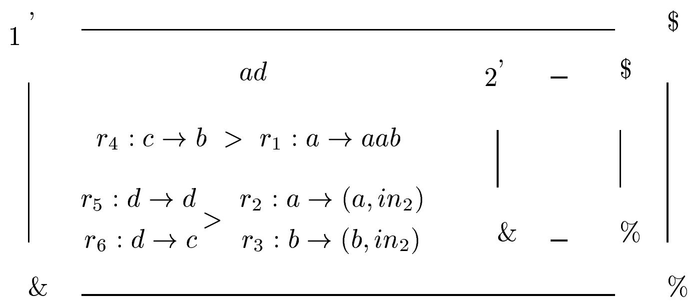

Figure 1. Example of a super-cell system.

We emphasize the fact that the objects we deal with in this paper are considered *atomic*, in the etymological sense: they have no parts, so we identify them with symbols. In [12] one also considers super-cell systems with the objects being strings and evolving either by means of rewriting rules, as usual in Chomsky grammars, or by means of splicing rules, as in DNA computing area (see, e.g., [13]). They are called *rewriting super-cell systems* and *splicing super-cell systems*, respectively, while the basic variant, introduced at the beginning of this section, is called a *transition super-cell system*. Because we work here only with transition systems, we simply call them *super-cell systems*.

The family of number relations $P_T(\Pi)$ generated by super-cell systems with priority, cooperation, and membrane dissolving action, and of degree at most $m, m \geq 1$, is denoted by $PsSC_m(Pri, Coo, \delta)$; when one of the features $\alpha \in \{Pri,\ Coo,\ \delta\}$ is not present, we replace it with $n\alpha$. The union of all families $PsSC_m(\alpha, \beta, \gamma), m \geq 1$, is denoted by $PsSC(\alpha, \beta, \gamma)$, for $\alpha \in \{Pri, nPri\}$, $\beta \in$

$\{Coo, nCoo\}$, $\gamma \in \{\delta, n\delta\}$. When dealing with the set $N(\Pi)$, we replace P in front of PsSC with N. In general, the results we obtain for PsSC families are the same with those for NsSC families.

The following inclusions directly follow from the definitions:

Lemma 1. *(i)* $PsSC_m(\alpha,\beta,\gamma) \subseteq PsSC_{m+1}(\alpha,\beta,\gamma)$, *and* $PsSC_m(\alpha,\beta,\gamma) \subseteq PsSC(\alpha,\beta,\gamma)$, *for all* $m \geq 1$ *and for all possible* α,β,γ.

(ii) $PsSC_m(\alpha',\beta',\gamma') \subseteq PsSC_m(\alpha,\beta,\gamma)$, *for all* $m \geq 1$ *and all possible* α,β,γ *such that* α',β',γ' *are either equal to* α,β,γ *or to* $n\alpha, n\beta, n\gamma$, *respectively, when* n *is not already present. The same assertion holds for families* $PsSC(\alpha,\beta,\gamma)$.

(iii) The previous assertions hold true also for families $NsSC_m(\alpha,\beta,\gamma), m \geq 1$, *and* $NsSC(\alpha,\beta,\gamma)$.

(iv) All these families are included in PsRE, LsRE, respectively.

3 A Basic Result

We start by giving a general result, which says that when the membrane dissolving action is not used, then all hierarchies collapse at the level two. Although intuitively simple, this result will considerably simplify the subsequent investigations.

Theorem 2. $PsSC(\alpha,\beta,n\delta) = PsSC_m(\alpha,\beta,n\delta)$, *for all* $\alpha \in \{Pri, nPri\}$, $\beta \in \{Coo, nCoo\}$, *and* $m \geq 2$.

Proof. Consider a super-cell system $\Pi = (V, T, \mu, w_1, \ldots, w_m, (R_1,\rho_1), \ldots, (R_m,\rho_m), i_0)$, of some degree $m \geq 1$, with the skin membrane labeled with 1. Assume that $m \geq 2$; then, because i_0 should be elementary, it is not the skin membrane, that is $i_0 \neq 1$. Let $a_i, 1 \leq i \leq m$, be new symbols associated with each $a \in V$, and h_i be the morphisms defined by $h_i(a) = a_i$, for $a \in V, 1 \leq i \leq m$. Assume that $R_i = \{r_{i,1}, \ldots, r_{i,t_i}\}$, with $t_i \geq 0, 1 \leq i \leq m$. Also, assume that Π does not contain rules which cannot be applied because of wrong target indications: if R_i contains a rule $a \to v$ with v introducing (b, in_j), then j is a membrane placed immediately inside i.

We construct the system

$$\Pi' = (V', T, [_1[_{i_0}\ ]_{i_0}]_1, w, w_{i_0}, (R'_1,\rho'_1), (R'_{i_0},\rho'_{i_0}), i_0),$$
$$V' = V \cup \{a_i \mid a \in V, 1 \leq i \leq m\},$$
$$w = h_1(w_1)\ldots h_{i_0-1}(w_{i_0-1})h_{i_0+1}(w_{i_0+1})\ldots h_m(w_m),$$
$$R'_1 = \{r'_{i,j} : h_i(u) \to v' \mid \text{for } r_{i,j} : u \to v \in R_i, 1 \leq i \leq m, i \neq i_0, 1 \leq j \leq t_i,$$

where v' is obtained from v in the following way:
each $(b, here)$ from v is replaced with b_i,
each (b, out) from v is replaced with b_j,
where j is the membrane outside i,
with the exception of $i = 1$, where (b, out) is left unchanged$\}$,
each (b, in_s) from v is replaced with b_s, for s a membrane

placed immediately inside i,
with the exception of $s = i_0$, when (b, in_{i_0}) is left unchanged,

$\rho'_1 = \{r'_{i,j} > r'_{i,k} \mid \text{for } r_{i,j} > r_{i,k}, 1 \leq i \leq m, i \neq i_0, 1 \leq j,k \leq t_i\}$,

$R'_{i_0} = \{r'_{i_0,j} : u \to v' \mid \text{for } r_{i_0,j} : u \to v \in R_{i_0}, 1 \leq j \leq t_{i_0}$, where
v' is obtained from v in the following way:
each $(b, here)$ from v remains unchanged,
each (b, out) from v is replaced with (b_j, out), where j
is the membrane outside i_0 in the system Π,
each (b, in_s) from v is replaced with (b_s, out) for s a membrane
placed immediately inside $i_0\}$,

$\rho'_{i_0} = \{r'_{i_0,j} > r'_{i_0,k} \mid \text{for } r_{i_0,j} > r_{i_0,k}, 1 \leq j,k \leq t_{i_0}\}$.

The itinerary of objects through membranes of Π is simulated by the subscripts of symbols used in Π'. The membrane i_0 is preserved, with the symbols representing its objects without having subscripts. Thus, the rules in R_{i_0} have the same left hand multiset (but a modified right hand side). The priorities imposed by $\rho_1, \ldots, \rho_m$ are captured by the relations in ρ'_1, ρ'_{i_0}, which act "locally", on the rules associated with each membrane of Π, because of the subscripts of symbols. In this way, any computation in Π can be simulated by an equivalent computation in Π', and conversely; the two computations stops in the same circumstances. In consequence, $P_T(\Pi) = P_T(\Pi')$ and we have the inclusion $PsSC(\alpha, \beta, n\delta) \subseteq PsSC_2(\alpha, \beta, n\delta)$. The converse inclusion was pointed out in Lemma 1.

4 Super-cell Systems with Cooperation

In this section we investigate the size of the families $PsSC_m(\alpha, Coo, \beta)$, for $\alpha \in \{Pri, nPri\}$ and $\beta \in \{\delta, n\delta\}$, in comparison with known families in formal language theory.

In [12] it is proved that even for a particular class of super-cell systems with cooperation, the systems with catalysts, as we have mentioned in the Introduction, we can characterize all recursively enumerable relations over $\mathbf{N}$. Systems with two membranes are sufficient. We give here a new proof of this result, much shorter, but directly for the cooperating systems, also proving that in this case one membrane is enough. As in [12], too, to this aim we use the notion of a *matrix grammar with appearance checking*.

Such a grammar is a construct $G = (N, T, S, M, F)$, where N, T are disjoint alphabets, $S \in N$, M is a finite set of sequences of the form $(A_1 \to x_1, \ldots, A_n \to x_n)$, $n \geq 1$, of context-free rules over $N \cup T$ (with $A_i \in N, x_i \in (N \cup T)^*$, in all cases), and F is a set of occurrences of rules in M (we say that N is the nonterminal alphabet, T is the terminal alphabet, S is the axiom, and the elements of M are called matrices.)

For $w, z \in (N \cup T)^*$ we write $w \Longrightarrow z$ if there is a matrix $(A_1 \to x_1, \ldots, A_n \to x_n)$ in M and the strings $w_i \in (N \cup T)^*, 1 \leq i \leq n+1$, such that $w = w_1, z = w_{n+1}$, and, for all $1 \leq i \leq n$, either $w_i = w'_i A_i w''_i, w_{i+1} = w'_i x_i w''_i$, for some $w'_i, w''_i \in (N \cup T)^*$, or $w_i = w_{i+1}$, A_i does not appear in w_i, and the rule

$A_i \to x_i$ appears in F. (The rules of a matrix are applied in order, possibly skipping the rules in F if they cannot be applied; we say that these rules are applied in the *appearance checking* mode.)

We denote by $\Longrightarrow^*$ the reflexive and transitive closure of the relation $\Longrightarrow$. The language generated by G is defined by $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$. The family of languages of this form is denoted by MAT_{ac}. When the set F is empty, hence all rules have to be applied effectively (the grammars are said to be *without appearance checking*), the obtained family is denoted by MAT.

It is known that $CF \subset MAT \subset MAT_{ac} = RE$. Further details about matrix grammars can be found in [7] and in [16].

Lemma 3. *$PsCF \subset PsMAT \subset PsMAT_{ac} = PsRE$, and $LsCF = LsMAT \subset LsMAT_{ac} = LsRE$.*

Proof. The inclusions follow from the remark above about the power of matrix grammars. It is known that $PsCF$ contains only semilinear sets, while MAT contains languages which are not semilinear (see Example 1.1.2 in [7]). On the other hand, it is known that the one-letter languages in the family MAT are regular ([9]), which implies that $LsCF = LsMAT$ and also that the inclusions $PsMAT \subset PsMAT_{ac}$, $LsMAT \subset LsMAT_{ac}$ are proper.

Theorem 4. *$PsRE = PsSC_m(Pri, Coo, n\delta) = PsSC_m(Pri, Coo, \delta) = PsSC(Pri, Coo, \delta) = PsSC(Pri, Coo, n\delta)$, for all $m \geq 1$. The similar equalities for length sets also hold true.*

Proof. According to Lemma 1, it is enough to prove the inclusion $PsRE \subseteq PsSC_1(Pri, Coo, n\delta)$. To this aim, we use the equality $PsRE = PsMAT_{ac}$ and show that each matrix grammar with appearance checking can be simulated, up to the order of symbols in the sentential forms, by a super-cell system of degree one with cooperation and priorities.

Take a matrix grammar with appearance checking, $G = (N, T, S, M, F)$ generating the language $L(G)$ such that $\Psi_T(L(G))$ is a given recursively enumerable subset of $\mathbf{N}^k$, for $k = card(T)$.

According to Lemma 1.3.7 in [7], without loss of generality we may assume that $N = N_1 \cup N_2 \cup \{S, \dagger\}$, with these three sets mutually disjoint, and that the matrices in M are of one of the following forms:

1. $(S \to XA)$, with $X \in N_1, A \in N_2$,
2. $(X \to Y, A \to x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*$,
3. $(X \to Y, A \to \dagger)$, with $X, Y \in N_1, A \in N_2$,
4. $(X \to x_1, A \to x_2)$, with $X \in N_1, A \in N_2$, and $x_1, x_2 \in T^*$.

Moreover, there is only one matrix of type 1 (we use then to write its rule in the form $S \to X_0A_0$, in order to stress that these symbols are fixed) and F consists exactly of all rules $A \to \dagger$ appearing in matrices of type 3; $\dagger$ is a trap-symbol, once introduced, it is never removed. A matrix of type 4 is used only once, at the last step of a derivation.

We construct the super-cell system of degree one

$$\Pi = (N \cup T, T, [_1\ ]_1, X_0A_0, (R_1, \rho_1), 1),$$

with the following rules:

1. $XA \to Yx$, for each matrix $(X \to Y, A \to x)$ in M of type 2;
2. $r : X \to Y$, $r' : A \to \dagger$, for each matrix $(X \to Y, A \to \dagger)$ in M of type 3; moreover $r' > r$ for each such a pair of rules;
3. $\dagger \to \dagger$;
4. $XA \to x_1x_2$, for each matrix $(X \to x_1, A \to x_2)$ in M of type 4;
5. $X \to X$, for all $X \in N_1$;
6. $A \to A$, for all $A \in N_2$.

It is easy to see that the cooperating rules of types 1 and 4 simulate the non-appearance checking matrices of M, while, because of the priority relation, the rules of type 2 correctly simulate the matrices with rules applied in the appearance checking mode: if A is present, then the trap-symbol † is introduced, which can evolve for ever. If not, then the change of X for Y is performed. Always we have only one copy of a symbol from N_1. If we reach a multiset where only terminal symbols appear, then no further step can be done; otherwise, the rules of types 5, 6 prevent terminating the computation. Note that after using a rule of type 4, no rule of type 1 or of the form $X \to Y$ can be used, and, conversely, if a string contains a symbol $X \in N_1$ and no symbol from N_2, then X can never be eliminated. Consequently, $\Psi_T(L(G)) = P_T(\Pi)$. Note that we do not use the membrane dissolving action.

Theorem 5. *$PsMAT \subseteq PsSC_1(nPri, Coo, \delta)$, and $PsMAT \subset PsSC_2(nPri, Coo, \delta)$, strict inclusion; $PsMAT \subseteq PsSC_1(nPri, Coo, n\delta)$; the same assertions are valid for families NsSC instead of PsSC.*

Proof. The inclusions follow as a particular case of the construction in the previous proof: if there is no matrix with appearance checking in the starting grammar G, then we need no priority in our super-cell system which simulates G.

Super-cell systems with two membranes which use the dissolving action can generate one-component non-semilinear sets even in the non-cooperating non-priority case: consider the system

$$\Pi = (\{a, b\}, [_1[_2\ ]_2]_1, \lambda, ab, (\{b \to b, b \to \delta, a \to aa\}, \emptyset), (\emptyset, \emptyset), 1).$$

We obtain $N(\Pi) = \{2^n \mid n \geq 1\}$: at each step we have to multiply by two all currently available copies of a, and this is done as long as the inner membrane exists, that is, as long as the rule $b \to b$ is used; after dissolving it, by using the rule $b \to \delta$, the computation halts.

Corollary 6. *$PsCF \subseteq PsSC_1(nPri, nCoo, n\delta)$.*

Proof. A construction similar to that in the proof of Theorem 4 can be carried out for a context-free grammar (which corresponds to a matrix grammar whose matrices have only one rule each and no appearance checking is present) and we get a super-cell system of degree one using no priority, no cooperation, and no membrane dissolving action.

5 Super-cell Systems without Cooperation

We now investigate the families $PsSC_m(\alpha, nCoo, \beta), NsSC_m(\alpha, nCoo, \beta)$, for $m \geq 1$ and $\alpha \in \{Pri, nPri\}, \beta \in \{\delta, n\delta\}$. We first complete the study of the families considered in Corollary 6. To this aim, we need the notion of a *cooperating distributed* (CD, for short) grammar system with maximal derivation, as introduced in [11], [4]; see details in [5].

A CD grammar system of degree $m, m \geq 1$, is a construct $\Gamma = (N, T, S, P_1, \ldots, P_m)$, where N, T are disjoint alphabets (of nonterminals and of terminals, respectively), $S \in N$ (axiom), and $P_i, 1 \leq i \leq m$, are finite sets of context-free rules over the alphabets N, T. Each set P_i is called a *component* of the system. With respect to each set P_i we define the derivation relation $\Longrightarrow_i$ over $(N \cup T)^*$ as usual. Then, a maximal derivation with respect to a component P_i is a derivation $x \Longrightarrow_i^* y$ for which there is no $z \in (N \cup T)^*$ such that $y \Longrightarrow_i z$. We write $x \Longrightarrow_i^t y$. (No rule of the component P_i can be applied to the string y.) The language generated by the system Γ is

$$\begin{aligned} L(\Gamma) = \{w \in T^* \mid S \Longrightarrow_{i_1}^t w_1 \Longrightarrow_{i_2}^t \ldots \Longrightarrow_{i_r}^t w_r = w, \\ \text{for some } r \geq 1, i_1, \ldots, i_r \in \{1, 2, \ldots, m\}\}. \end{aligned}$$

We denote by $CD_m, m \geq 1$, the family of languages generated by CD grammar systems of degree at most m and by CD the union of all these families.

It is known that $CF = CD_1 = CD_2 \subset CD_3 = CD = ET0L$. These relations extend also to Parikh sets and length sets.

For the sake of completeness, we specify that an ET0L system is a construct $G = (V, T, w, P_1, \ldots, P_m)$, where V is an alphabet, $T \subseteq V$, $w \in V^*$, and $P_i, 1 \leq i \leq m$, are finite sets of context-free rules over V such that for each $a \in V$ there is at least one rule $a \to x$ in each set P_i (we say that these set, called *tables*, are *complete*). In a derivation step, a table, nondeterministically chosen, is used; this means rewriting once all the symbols presents in the current sentential form. The language generated by G, denoted $L(G)$, consists of all strings over T which can be generated in this way starting from the axiom w. An ET0L system with only one table is called an E0L system. Details can be found, e.g., in [15].

Theorem 7. *$PsCF = PsSC_m(nPri, nCoo, n\delta) = PsSC(nPri, nCoo, n\delta)$, $LsCF = LsSC_m(nPri, nCoo, n\delta) = LsSC(nPri, nCoo, n\delta)$, $m \geq 1$.*

Proof. According to Lemma 1 and Corollary 6 we only have to prove the inclusion $PsSC(nPri, nCoo, n\delta) \subseteq PsCF$. To this aim, we will prove the inclusion $PsSC(nPri, nCoo, n\delta) \subseteq PsCD_2$.

Consider a super-cell system of degree $m \geq 1$, $\Pi = (V, T, \mu, w_1, \ldots, w_m, (R_1, \emptyset), \ldots, (R_m, \emptyset), i_0)$, without cooperation and without using de dissolving action. We assume the membranes labeled with the numbers $1, 2, \ldots, m$. We also assume that if a rule $a \to x$ in a set R_i contains a symbol (b, in_j) in x, then the membrane j is "accessible" to the rule, in the sense that the rule is placed in the region immediately outside membrane j (otherwise, the rule is never applicable and we can remove it; note that because we cannot dissolve membranes, the membrane structure is constant). Denote $T_i = \{a \in V \mid$ there is no rule $a \to x \in R_i\}$. For each symbol $a \in V$ we consider two new symbols, a_i and a_i', for each $i \in \{1, 2, \ldots, m\}$. We denote by h_i the morphisms defined by $h_i(a) = a_i'$, for all $a \in V, 1 \leq i \leq m$. If a symbol $a \in T_i$ appears in w_i, for $i \neq i_0$, then it is removed (it cannot evolve and cannot reach the output membrane).

We construct the CD grammar system of degree two $\Gamma = (N, T, S, P_1, P_2)$, where

$$N = \{a_i, a'_i \mid a \in V, 1 \leq i \leq m\} \cup (V - T_{i_0}) \cup \{S\},$$

and with the following components:

$P_1 = \{S \to h'_1(w_1)h'_2(w_2)\dots h'_m(w_m)\}$
$\cup \{a_i \to \bar{x} \mid a \to x \in R_i,$ where $\bar{x}$ is obtained from
x in the following way:
(a) if a symbol $(b, here)$ appears in x, then in $\bar{x}$
we put b'_i instead of it,
(b) if a symbol (b, out) appears in x, then in $\bar{x}$
we put b'_j instead of it, where j is the label
of the membrane immediately outside the membrane i;
in the case of the skin membrane, instead of (b, out) we put λ,
(c) if a symbol (b, in_j) appears in x, then in $\bar{x}$
we put b'_j instead of it;
moreover, in any of the cases (a), (b), (c), instead of any symbol
b'_j, for $b \in T_j$, we put λ in $\bar{x}$, with the exception
of the output membrane, where we put $b\}$,
$P_2 = \{a'_i \to a_i \mid 1 \leq i \leq m\}$.

The component P_1 simulates the transitions in the super-cell system Π. All symbols a_i such that a rule in R_i can be applied to a have to be rewritten, otherwise the derivation step is not done in the t mode. Because the use of a rule introduces primed symbols, we cannot rewrite repeatedly a symbol (this is possible in CD grammar systems, but it is not allowed in a transition in a super-cell system). The subscripts of the symbols indicate the membranes where they are placed in the same way as in the proof of Theorem 2 and they select the corresponding rules to be applied to these symbols; these subscripts are changed during the derivation according to the evolution rules of Π. All symbols which cannot evolve or are sent outside the skin membrane are removed, with the exception of the symbols reaching the output membrane. The component P_2 replaces primed symbols with non-primed symbols, still preserving their subscripts. Thus, the process can be iterated. The only way to reach a terminal string is to remove all symbols different from those from T which reach the output membrane. Consequently, $P_T(\Pi) = \Psi_T(\Gamma)$.

We do not know whether or not also the hierarchies not mentioned in Theorems 2, 4, and 7 collapse. At least for $PsSC_m(nPri, nCoo, \delta), m \geq 1$, we conjecture that we have an infinite hierarchy.

An estimation of families of Parikh sets in the remaining families is obtained by comparing them with the Parikh sets of languages in the L area. We start with lower estimations. First, we mention the few results about Parikh sets and length sets we know for E0L and ET0L languages.

Lemma 8. *$PsCF \subset PsE0L \subseteq PsET0L \subset PsCS$, $PsE0L - PsMAT \neq \emptyset$. The same relations hold also for length sets.*

Proof. The inclusions follow from the similar inclusions for the language families. Because $LsE0L$ contains one-component non-semilinear sets, we get the assertions for the CF and MAT cases. The language $\{a^n \mid n$ a is prime number$\}$ is known not to be an ET0L language (Exercise VI.2.6 in [15]), hence we have the strictness of the inclusions $PsET0L \subset PsCS, LsET0L \subset LsCS$.

Theorem 9. *$PsE0L \subseteq PsSC_2(nPri, nCoo, \delta)$ and similarly for the length sets.*

Proof. Having an E0L system $G = (V, T, w, P)$, we construct the super-cell system of degree two

$$\Pi = (V \cup \{d\}, T, [_1 [_2\ ]_2]_1, \emptyset, dw, (R_1, \emptyset), (R_2, \emptyset), 1),$$

with

$$R_1 = \{a \to a \mid a \in V - T\},$$
$$R_2 = \{d \to d,\ d \to \delta\} \cup P.$$

In the inner membrane one simulates derivation steps in G as long as the symbol d is present; when the membrane is dissolved, the simulation stops. If any symbol not in T is still present, then we can use the corresponding rule $a \to a$ in R_1 for ever. Thus, only the terminal derivations in G are simulated by halting computations in Π.

The result above cannot be improved by taking only one membrane: one membrane means no dissolving action; according to Theorem 7, we can then compute only context-free Parikh images.

It is also worth mentioning here the importance of a detail in the definition of super-cell systems: the output membrane is not necessarily an elementary one in the initial configuration, but it must be elementary in the halting configuration. This is the case for the system Π in the previous proof.

It is not possible to generate all E0L Parikh sets without using this feature. More specifically, systems of the form $\Pi = (V, T, [_1 [_2\ ]_2]_1,\ w_1, w_2,\ (R_1, \rho_1), (R_2, \rho_2), 2)$, without priority and without cooperation, do not generate all E0L Parikh sets. This is a consequence of Theorem 7: because neither the output membrane nor the skin membrane can be dissolved, the system is also not using the membrane dissolving action. Consequently, it generates only Parikh images of context-free languages.

When a priority relation is used, by super-cell systems of degree one we can simulate all Parikh sets of ET0L languages (without knowing whether or not this actually means more than Parikh sets of E0L languages: see again Lemma 8).

Theorem 10. *$PsET0L \subseteq PsSC_1(Pri, nCoo, n\delta) \subseteq PsSC_1(Pri, nCoo, \delta)$. Similar relations hold for length sets.*

Proof. It is known that each ET0L language can be generated by an ET0L system with two tables only. Let $G = (V, T, w, P_1, P_2)$ be such a system. For each letter $a \in V$, let a' be a new symbol, let V' be the set $\{a' \mid a \in V\}$, and h be the morphism defined by $h(a) = a', a \in V$. Let $d, d_1, d_2, e, \dagger$ be new symbols. Assume that $P_1 = \{p_1, \ldots, p_u\}$ and $P_2 = \{q_1, \ldots, q_v\}$.

We construct the super-cell system of degree 1

$$\Pi = (V \cup V' \cup \{d, d_1, d_2, e, \dagger\}, T, [_1\ ]_1, d\ h(w), (R_1, \rho_1), 1),$$

with the following rules:

$r_1 : d \to d_1$,
$r_2 : d \to d_2$,
$r'_1 : d_1 \to d$,
$r'_2 : d_2 \to d$,
$r_3 : d \to e$,
$r_4 : e \to (e, out)$,
$p'_i : a' \to h(x)$, for $p_i : a \to x \in P_1, 1 \leq i \leq u$,
$q'_i : a' \to h(x)$, for $q_i : a \to x \in P_2, 1 \leq i \leq v$,
$r_a : a' \to a$, for $a \in T$,
$r_a : a' \to \dagger$, for $a \in V - T$,
$r_\infty : \dagger \to \dagger$,

and the priority relations

$$\begin{aligned}
&r_1 > p'_i,\ r_2 > p'_i,\ r_3 > p'_i,\ r_4 > p'_i,\ 1 \leq i \leq u,\\
&r_1 > q'_i,\ r_2 > q'_i,\ r_3 > q'_i,\ r_4 > q'_i,\ 1 \leq i \leq v,\\
&r_1 > r_a,\ r_2 > r_a,\ r'_1 > r_a,\ r'_2 > r_a,\ r_3 > r_a,\ a \in V,\\
&r'_1 > q'_i,\ 1 \leq i \leq v,\\
&r'_2 > p'_i,\ 1 \leq i \leq u.
\end{aligned}$$

The system works as follows. To the multiset described by the string $d\ h(z)$ we can apply one of the rules r_1, r_2, r_3, and no rule can be applied to the objects in $h(z)$ because of the priority relation. The rules r_1, r_2 select the table to be simulated: in the presence of d_i only rules from P_i can be simulated, $i = 1, 2$. This is done in one step, in the same way as in an ET0L system (note that the completeness of tables ensures the fact that all objects evolve). After such a step, the symbols d_1, d_2 return to d and the process can be iterated. As long as one of the symbols d, d_1, d_2 is present, no rule r_a can be used. When we use the rule r_3, we pass to the final stage of the computation. In the presence of e no rule p'_i, q'_i can be used, but now the rules r_a can be used, because no rule with a higher priority can be applied. This means that at the same time when we remove the object e from the system, we have to replace with the trap-object $\dagger$ each a' such that $a \in V - T$; for the symbols in T we just remove their primes. Therefore, the computation stops when no symbol from $V - T$ is present and continues for ever in the opposite case. This is exactly the condition for including a string in the language generated by an ET0L system. Consequently, $\Psi_T(G) = P_T(\Pi)$.

We do not have similarly many upper approximations for families $PsSC(\alpha, \beta, \gamma)$ different from the cases $(\alpha, \beta, \gamma) \in \{(Pri, Coo, \delta), (Pri, Coo, n\delta), (nPri, nCoo, n\delta)\}$. We believe that most of the remaining families are included in $PsET0L$, but we have a proof only for one case.

Theorem 11. *$PsSC(nPri, nCoo, \delta) \subseteq PsET0L$. The same assertions are true for length sets.*

Proof. Let us consider a super-cell system of degree $m, m \geq 1$, without cooperation, $\Pi = (V, T, \mu, w_1, \ldots, w_m, (R_1, \rho_1), \ldots, (R_m, \rho_m), i_0)$, with the membranes labeled with the numbers $1, 2, \ldots, m$. For each membrane i and for each symbol $a \in V$ we consider a new symbol, a_i, $1 \leq i \leq m$. We define the morphisms h_i by $h_i(a) = a_i$, for all $a \in V, 1 \leq i \leq m$. Denote by T_i the sets $\{a \in V \mid$ there is no rule of the form $a \to x$ in $R_i\}$. We remove from w_i, for $i \neq i_0$, all symbols in T_i (they do not evolve and do not reach the output membrane).

For each $i = 1, 2, \ldots, m$, consider four new symbols $c_i, c'_i, \bar{c}_i, d_i$; c_i is meant to tell us that membrane i exists, c'_i, d_i will tell us that membrane i was just dissolved, while $\bar{c}_i$ will indicate the fact that membrane i was dissolved at a previous step.

We construct the ET0L system G with the total alphabet W defined by

$$W = \{a_i \mid a \in V, 1 \leq i \leq m\} \cup \{\dagger\} \cup \{c_i, c'_i, \bar{c}_i, d_i \mid 1 \leq i \leq m\} \cup \{a' \mid a \in T\} \cup T,$$

the terminal alphabet T, the axiom

$$w = h_1(w_1) \ldots h_n(w_m) c_1 c_2 \ldots c_m,$$

and the tables obtained in the following way.

Consider all sets of rules of the following form:

$$\begin{aligned} &\{c_{i_1} \to c_{i_1}, \ldots, c_{i_k} \to c_{i_k}, c_{j_1} \to \dagger, \ldots, c_{j_l} \to \dagger, \\ &\bar{c}_{i_1} \to \dagger, \ldots, \bar{c}_{i_k} \to \dagger, \bar{c}_{j_1} \to \bar{c}_{j_1}, \ldots, \bar{c}_{j_l} \to \bar{c}_{j_l}, \\ &d_1 \to \dagger, \ldots, d_m \to \dagger, c'_1 \to \dagger, \ldots, c'_m \to \dagger\}, \\ \text{for } &\{i_1, \ldots, i_k\}, \{j_1, \ldots, j_l\} \text{ a partition of } \{1, 2, \ldots, m\}. \end{aligned}$$

There are 2^m sets of this form. Let us enumerate them in any given order and denote them by $M_1, M_2, \ldots, M_{2^m}$. Let $E(M_j) = \{i \mid c_i \to c_i \in M_j, 1 \leq i \leq m\}$ (the set of membranes which exist according to M_j).

We consider all sets of rules

$$Q_j = M_j \cup \bigcup_{i \in E(M_j)} R_i, \ 1 \leq j \leq 2^m.$$

(Because $i \in E(M_j)$, the rules in R_i can be applied, the membrane with the label i still exists.)

For each such set Q_j we proceed as follows.

For each rule $a \to x \in R_i, i \in E(M_j)$, consider the rule $a_i \to \bar{x}$ constructed in a way similar to that in the proofs of Theorems 2 and 7 (with the differences imposed by the use of δ):

(a) if a symbol $(b, here)$ appears in x, then in $\bar{x}$ we put b_i instead of it;
(b) if a symbol (b, out) appears in x, then in $\bar{x}$ we put b_t instead of it, where t is the label of the membrane immediately outside the membrane i; in the case of the skin membrane, instead of (b, out) we put λ,
(c) if a symbol (b, in_t) appears in x, and M_j contains the rules $c_t \to c_t$, $\bar{c}_t \to \dagger$ (that is, membrane t exists), and, moreover, membrane t is adjacent to the rule $a \to x \in R_i$ (this can be checked by examining μ and the information given by M_j about the membranes still existing), then in $\bar{x}$ we put b_t instead of it; if membrane t is not adjacent to the rule, maybe because it was dissolved, then the rule $a \to \bar{x}$ is removed from the set we construct, it cannot be applied in the given circumstances.

Moreover, instead of any symbol b_j, for $b \in T_j$, we put λ in $\bar{x}$, with the exception of the output membrane, where we put b', for $b \in T$, and λ for $b \in T_{i_0} - T$.

If the rule $a \to x$ introduces the symbol δ, then instead of δ, in $\bar{x}$ we introduce the symbol d_i.

Let Q'_j be the set of rules obtained in this way. For each rule $c_i \to c_i$ appearing in Q'_j such that R_i contains a rule $a \to v\delta$, we either let $c_i \to c_i$ unchanged (this is meant to correspond to the case when the rule $a \to v\delta$ is not actually used), or we replace it with $c_i \to c'_i$ (this will correspond to the case when the rule $a \to v\delta$ is effectively used, that is the symbol d_i is also introduced in the current string), and we do this in all possible combinations; that is, we produce a series of sets $Q_{j,1}, \ldots, Q_{j,s}$, differing from Q'_j just by rules $c_i \to c_i, c_i \to c'_i$. Of course, for each i, each set $Q_{j,t}$ contains exactly one of these two rules.

To these sets we add the rules $a \to \dagger$, for $a \in T$, as well as completion rules $a \to a$ for all $a \in W - T$ for which there is no rule $a \to y$ already considered. The obtained sets – we still denote them with $Q_{j,1}, \ldots, Q_{j,s}$ – are tables of our ET0L system. We say that these tables are *of type Q*.

We also consider the following tables:

$$\begin{aligned}
&\{c_{i_1} \to c_{i_1}, \ldots, c_{i_k} \to c_{i_k}, c_{j_1} \to \dagger, \ldots, c_{j_l} \to \dagger, c_{r_1} \to \dagger, \ldots, c_{r_s} \to \dagger,\\
&\bar{c}_{i_1} \to \dagger, \ldots, \bar{c}_{i_k} \to \dagger, \bar{c}_{j_1} \to \dagger, \ldots, \bar{c}_{j_l} \to \dagger, \bar{c}_{r_1} \to \bar{c}_{r_1}, \ldots, \bar{c}_{r_s} \to \bar{c}_{r_s},\\
&c'_{i_1} \to \dagger, \ldots, c'_{i_k} \to \dagger, c'_{j_1} \to \bar{c}_{j_1}, \ldots, c'_{j_l} \to \bar{c}_{j_l}, c'_{r_1} \to \dagger, \ldots, c'_{r_s} \to \dagger,\\
&d_{i_1} \to \dagger, \ldots, d_{i_k} \to \dagger, d_{j_1} \to \bar{c}_{j_1}, \ldots, d_{j_l} \to \bar{c}_{j_l}, d_{r_1} \to \dagger, \ldots, d_{r_s} \to \dagger,\\
&a_p \to a_q \mid \text{ where: } p \in \{j_1, \ldots, j_l\},\\
&q \text{ is the membrane where a symbol from membrane } p\\
&\text{will be placed after dissolving membranes } j_1, \ldots, j_l,\\
&\text{for all } a \in V \text{ and all partitions}\\
&\{i_1, \ldots, i_k\}, \{j_1, \ldots, j_l\}, \{r_1, \ldots, r_s\} \text{ of } \{1, 2, \ldots, m\}\}\\
\cup\, &\{a \to \dagger \mid a \in T\}.
\end{aligned}$$

(Of course, we also add completions rules $a \to a$ for symbols $a \in W$ for which there is no a-rule already considered.) We say that these tables are *of type P*.

We also consider the following "final" table:

$$\begin{aligned}
T_f = &\{c_i \to \lambda, \mid 1 \leq i \leq m\}\\
&\cup \{\bar{c}_i \to \lambda \mid 1 \leq i \leq m, i \neq i_0\} \cup \{\bar{c}_{i_0} \to \dagger\}\\
&\cup \{a' \to a \mid a \in T\}\\
&\cup \{a \to \dagger \mid a \in W - (\{b' \mid b \in T\} \cup \{c_i, \bar{c}_i \mid 1 \leq i \leq m\})\}.
\end{aligned}$$

These are all the tables of our ET0L system G.

From the above construction, it is now clear that the use of a table of type Q in our ET0L system G is equivalent to a transition in the super-cell system Π. The difference is that after using a table $Q_{j,r}$, in the case of membrane dissolving, the subscripts of the symbols do not indicate correctly the membranes to which they belong. To this aim, we have to use a table of type P for performing the corrections indicated by the symbols d_t present in it (remember that these symbols tell us which are the membranes which were dissolved at the last transition).

On the other hand, as long as symbols c_t', d_t' are present, no table of type Q can be used: we have introduced rules $c_t' \rightarrow \dagger, d_t \rightarrow \dagger$ in all these tables, hence the trap-symbol will be produced. Similarly, for each string there precisely is only one table of type P which can be used without introducing the trap-symbol, because of the partition of the set $\{1, \ldots, m\}$ in the definition of tables of type P, in a set of existing membranes, a set of just dissolved membranes, and one of membranes which were dissolved at a previous transition.

Thus, we correctly simulate the transitions in Π (by two derivation steps in G), and, conversely, the derivations in G correspond to correct transitions in Π. Moreover, only halting computations in Π lead to terminal derivations in G: all symbols which can evolve in Π are nonterminal symbols in G, hence they should be removed in order to get a terminal derivation. After obtaining a terminal string (that is, after using the table T_f), no further derivation step is possible, because of the rules $a \rightarrow \dagger$, for $a \in T$, present in all tables different from T_f. In the output membrane we always collect terminal strings. If the trap symbol † is introduced, then it will never be eliminated.

Consequently, $P_T(\Pi) = \Psi_T(G)$.

6 Open Problems

We conclude this paper by pointing out a series of open problems. The first one is specific to super-cell systems: does the use of the membrane dissolving action makes any difference in what concerns the power of transition super-cell systems? The answer is negative for all cases considered in [12]: transition super-cell systems with catalysts (hence also cooperative systems), super-cell systems based on rewriting or on splicing (in the latter cases, the objects are strings over a given alphabet and the evolution rules are given as context-free rewriting rules or as splicing rules). In all these cases, one obtains characterizations of recursively enumerable sets of numbers or of languages without using the membrane dissolving action.

Can the result in Theorem 11 be extended to other classes of super-cell systems (that is, can ve also take into consideration systems with priority)? We expect a positive answer (maybe using ET0L systems with permitting random contexts, sets of symbols associated with tables and such that a table can be applied only if the associated symbols appear in the rewritten string; see details in [7]).

Which of the inclusions appearing in the lemmas and the theorems from the previous sections are proper? For instance, are the hierarchies on the number of membranes infinite in the cases not covered by Theorems 2 and 5? (A related question concerns the hierarchies on the depth of the membrane structure of super-cell systems. A partial answer can be found in [14].) For other inclusions (for instance, whether or not the inclusions in Theorems 9, 10 are proper), the answer could be based on an answer to the following problem of a "classic" flavour: how large is the family $PsET0L$ in comparison, for instance, with the family $PsE0L$. The same problem can be formulated for length sets. There are several papers about the length sets and the Parikh sets of L languages (see, e.g., [6], [8], [10]), but always dealing with non-extended systems. To our knowledge, the extended case was never investigated.

Note. Work supported by the Academy of Finland, Project 137358. Thanks are due to two referees who made several useful remarks about a previous version of the paper.

References

[1] J. P. Banâtre, A. Coutant, D. Le Metayer, A parallel machine for multiset transformation and its programming style, *Future Generation Computer Systems*, 4 (1988), 133–144.

[2] J. P. Banâtre, D. Le Metayer, Programming by multiset transformation, *Communications of the ACM*, 36 (1993), 98–111.

[3] G. Berry, G. Boudol, The chemical abstract machine, *Theoretical Computer Sci.*, 96 (1992), 217–248.

[4] E. Csuhaj-Varju, J. Dassow, On cooperating distributed grammar systems, *J. Inf. Process. Cybern., EIK*, 26, 1-2 (1989), 49–63.

[5] E. Csuhaj-Varju, J. Dassow, J, Kelemen, Gh. Păun, *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.

[6] J. Dassow, On Parikh-languages of L systems without interaction, *Rostock Math. Kolloq.*, 15 (1980), 103–110.

[7] J. Dassow, Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, Berlin, 1989.

[8] A. Ehrenfeucht, J. Karhumäki, G. Rozenberg, A note on D0L length sets, *Discrete Math.*, 6 (1978), 235-247.

[9] D. Hauschildt, M. Jantzen, Petri nets algorithms in the theory of matrix grammars, *Acta Inform.*, 31 (1994), 719–728.

[10] J. Karhumäki, On length sets of informationless L systems, in *Automata, Languages, Development* (A. Lindenmayer, G. Rozenberg, eds.), North Holland, Amsterdam, 1976, 227-242.

[11] R. Meersman, G. Rozenberg, Cooperating grammar systems, *Lect. Notes in Computer Sci.* 64, Springer-Verlag, Berlin, 1978, 364–373.

[12] Gh. Păun, Computing with membranes, submitted, 1998 (see also *TUCS Research Report* No. 208, November 1998, http://www.tucs.fi).

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

[14] I. Petre, A normal form for P systems, *Bulletin of the EATCS*, 67 (1999).

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

[16] G. Rozenberg, A. Salomaa, eds., *Handbook of Formal Languages*, Springer-Verlag, Heidelberg, 1997.