Playing with Derivation Modes

Rudolf Freund

TU Wien, Institut für Logic and Computation
Favoritenstraße 9–11, 1040 Wien, Austria
rudi@emcc.at

Summary. In the area of P systems, besides the standard maximally parallel derivation mode, many other derivation modes have been investigated, too. In this paper, many variants of hierarchical P systems and tissue P systems using different derivation modes are considered and the effects of using different derivation modes, especially the maximally parallel derivation modes and the maximally parallel set derivation modes, on the generative and accepting power are illustrated. Moreover, an overview on some control mechanisms used for (tissue) P systems is given.

1 Introduction

The basic model of *P systems* as introduced in [16] can be considered as a distributed multiset rewriting system, where all objects – if possible – evolve in parallel in the membrane regions and may be communicated through the membranes. But also P systems operating on more complex objects (e.g., strings, arrays) are often considered, too, for instance, see [6].

Besides the maximally parallel derivation mode, many other derivation modes have been investigated during the last two decades. Thus in this paper the definitions of the standard derivation modes used for P systems and tissue P systems are recalled. Various interpretations of derivation modes known from the P systems area are illustrated and well-known results are presented in a different manner.

Overviews on the field of P systems can be found in the monograph [17] and the Handbook of Membrane Computing [18]; for actual news and results we refer to the P systems webpage [20] as well as to the Bulletin of the International Membrane Computing Society.

With this paper being presented at CMC 20, the 20th anniversary edition of the meeting of the membrane systems community, the reader is assumed to be very familiar with the basic definitions and notations of P systems and tissue P

2 Prerequisites

The set of integers is denoted by $\mathbb{Z}$, and the set of non-negative integers by $\mathbb{N}$. Given an alphabet V, a finite non-empty set of abstract symbols, the free monoid generated by V under the operation of concatenation is denoted by V^*. The elements of V^* are called strings, the empty string is denoted by λ, and $V^*\backslash\{\lambda\}$ is denoted by V^+. For an arbitrary alphabet $V = \{a_1, \ldots, a_n\}$, the number of occurrences of a symbol a_i in a string x is denoted by $|x|_{a_i}$, while the length of a string x is denoted by $|x| = \sum_{a_i \in V} |x|_{a_i}$. The Parikh vector associated with x with respect to $a_1, \ldots, a_n$ is $(|x|_{a_1}, \ldots, |x|_{a_n})$. The Parikh image of an arbitrary language L over $\{a_1, \ldots, a_n\}$ is the set of all Parikh vectors of strings in L, and is denoted by $Ps(L)$. For a family of languages FL, the family of Parikh images of languages in FL is denoted by $PsFL$, while for families of languages over a one-letter (d-letter) alphabet, the corresponding sets of non-negative integers (d-vectors with non-negative components) are denoted by NFL (N^dFL).

A (finite) multiset over a (finite) alphabet $V = \{a_1, \ldots, a_n\}$, is a mapping $f : V \to \mathbb{N}$ and can be represented by $\langle a_1^{f(a_1)}, \ldots, a_n^{f(a_n)} \rangle$ or by any string x for which $(|x|_{a_1}, \ldots, |x|_{a_n}) = (f(a_1), \ldots, f(a_n))$. In the following we will not distinguish between a vector $(m_1, \ldots, m_n)$, a multiset $\langle a_1^{m_1}, \ldots, a_n^{m_n} \rangle$ or a string x having $(|x|_{a_1}, \ldots, |x|_{a_n}) = (m_1, \ldots, m_n)$. Fixing the sequence of symbols $a_1, \ldots, a_n$ in an alphabet V in advance, the representation of the multiset $\langle a_1^{m_1}, \ldots, a_n^{m_n} \rangle$ by the string $a_1^{m_1} \ldots a_n^{m_n}$ is unique. The set of all finite multisets over an alphabet V is denoted by V°.

The family of regular, context-free, and recursively enumerable string languages is denoted by REG, CF, and RE, respectively. For example, $PsREG = PsCF$, which is the reason why in the area of multiset rewriting CF plays no role at all, and in the area of membrane computing we usually get characterizations of $PsREG$ and $PsRE$.

For more details of formal language theory the reader is referred to the monographs and handbooks in this area as [5] and [19].

Register machines

A *register machine* is a tuple $M = (m, B, l_0, l_h, P)$, where m is the number of registers, B is a set of labels, $l_0 \in B$ is the initial label, $l_h \in B$ is the final label, and P is the set of instructions bijectively labeled by elements of B. The instructions of M can be of the following forms:

- $l_1 : (ADD(j), l_2, l_3)$, with $l_1 \in B\backslash\{l_h\}$, $l_2, l_3 \in B$, $1 \leq j \leq m$.
  Increases the value of register j by one, followed by a non-deterministic jump to instruction l_2 or l_3. This instruction is usually called *increment*.
- $l_1 : (SUB(j), l_2, l_3)$, with $l_1 \in B\backslash\{l_h\}$, $l_2, l_3 \in B$, $1 \leq j \leq m$.
  If the value of register j is zero, then a jump to instruction l_2 is performed, otherwise the value of register j is decreased by one, and a jump to instruction l_3 is performed.
  The two cases of this instruction are usually called *zero-test* and *decrement*, respectively.
- $l_h : HALT$. Stops the execution of the register machine.

A *configuration* of a register machine is described by the contents of each register and by the value of the current label, which indicates the next instruction to be executed. Computations start by executing the instruction l_0 of P, and terminate with reaching the HALT-instruction l_h.

M is called deterministic if in all ADD-instructions $p : (\texttt{ADD}\,(r), q, s)$, it holds that $q = s$; in this case we write $p : (\texttt{ADD}\,(r), q)$.

For useful results on the computational power of register machines, we refer to [15]; for example, deterministic register machines can accept all recursively enumerable sets of vectors of natural numbers with k components using precisely $k + 2$ registers.

3 A General Model for Tissue and Hierarchical P Systems

We now recall the main definitions of the general model for tissue P systems and hierarchical P systems and the basic derivation modes as defined, for example, in [13].

A *(hierarchical) P system (with rules of type X)* working in the derivation mode δ is a construct

$$\Pi = (V, T, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m, f, \Longrightarrow_{\Pi,\delta}) \text{ where}$$

- V is the alphabet of *objects*;
- $T \subseteq V$ is the alphabet of *terminal objects*;
- μ is the hierarchical membrane structure (a rooted tree of membranes) with the membranes uniquely labeled by the numbers from 1 to m;
- $w_i \in V^*$, $1 \leq i \leq m$, is the *initial multiset* in membrane i;
- R_i, $1 \leq i \leq m$, is a finite set of *rules of type X* assigned to membrane i;
- f is the label of the membrane from which the result of a computation has to be taken from (in the generative case) or into which the initial multiset has to be given in addition to w_f (in the accepting case),
- $\Longrightarrow_{\Pi,\delta}$ is the derivation relation under the derivation mode δ.

The symbol X in "rules of type X" may stand for "evolution", "communication", "membrane evolution", etc.

In hierarchical P systems, the membranes are arranged in a tree structure. If we allow arbitrary graphs as communication structure, with the membranes now also called *cells* floating in the environment instead of being enclosed in the skin

A *(static) tissue P system (with rules of type X)* working in the derivation mode δ is a construct

$$\Pi = (V, T, m, w_1, \ldots, w_m, R_1, \ldots, R_m, f, \Longrightarrow_{\Pi,\delta}) \text{ where}$$

- V is the alphabet of *objects*;
- $T \subseteq V$ is the alphabet of *terminal objects*;
- m is the number of cells uniquely labeled by the numbers from 1 to m;
- $w_i \in V^*$, $1 \leq i \leq m$, is the *initial multiset* in cell i;
- R_i, $1 \leq i \leq m$, is a finite set of *rules of type X* assigned to cell i;
- f is the label of the cell from which the result of a computation has to be taken from (in the generative case) or into which the initial multiset has to be given in addition to w_f (in the accepting case),
- $\Longrightarrow_{\Pi,\delta}$ is the derivation relation under the derivation mode δ.

Each of the cells may have assigned its own set of rules R_i, but in the most general case the rules (for multisets) are of the form

$$(u_1) \ldots (u_m) \rightarrow (v_1) \ldots (v_m)$$

where $(u_1), \ldots, (u_m)$ and $(v_1), \ldots, (v_m)$ are multisets over V, and then instead of $R_1, \ldots, R_m$ we specify only one set of rules R for the whole tissue P system Π.

A configuration is a list of the contents of each cell or membrane region, respectively; a sequence of configurations $C_1, \ldots, C_k$ is called a *computation* in the derivation mode δ if $C_i \Longrightarrow_{\Pi,\delta} C_{i+1}$ for $1 \leq i < k$. The derivation relation $\Longrightarrow_{\Pi,\delta}$ is defined by the set of rules in Π and the given derivation mode which determines the multiset of rules to be applied to the multisets contained in each membrane or cell or even in the overall tissue P system.

The *language generated by* Π is the set of all terminal multisets which can be obtained in the output membrane / cell f starting from the initial configuration $C_1 = (w_1, \ldots, w_m)$ using the derivation mode δ in a halting computation, i.e.,

$$L_{gen,\delta}(\Pi) = \left\{ C(f) \in T^\circ \mid C_1 \overset{*}{\Longrightarrow}_{\Pi,\delta} C \wedge \neg \exists C' : \; C \Longrightarrow_{\Pi,\delta} C' \right\},$$

where $C(f)$ stands for the multiset contained in the output membrane or cell f of the configuration C. The configuration C is halting, i.e., no further configuration C' can be derived from it.

The family of languages of multisets generated by P systems and tissue P systems of type X with at most n membranes / cells in the derivation mode δ is denoted by $Ps_{gen,\delta}OP_n(X)$ and $Ps_{gen,\delta}OtP_n(X)$, respectively.

We may also consider (tissue) P systems as accepting mechanisms: in membrane / cell f, we add the input multiset w_0 to w_f in the initial configuration $C_1 = (w_1, \ldots, w_m)$ thus obtaining $C_1[w_0] = (w_1, \ldots, w_f w_0, \ldots, w_m)$; the input

$$L_{acc,\delta}(\Pi) = \left\{ w_0 \in T^\circ \mid \exists C : \; \left(C_1[w_0] \overset{*}{\Longrightarrow}_{\Pi,\delta} C \wedge \neg \exists C' : \; C \Longrightarrow_{\Pi,\delta} C' \right) \right\}.$$

Then the family of languages of multisets accepted by P systems and tissue P systems of type X with at most n membranes / cells in the derivation mode δ is denoted by $Ps_{acc,\delta}OP_n(X)$ and $Ps_{acc,\delta}OtP_n(X)$, respectively.

We finally mention that (tissue) P systems can also be used to compute functions and relations, with using f both as input and output membrane / cell or even using two different membranes / cells for the input and the output. Yet in this paper we will mainly focus on the generating case.

3.1 Derivation Modes

The set of all multisets of rules applicable in a (tissue) P system to a given configuration C is denoted by $Appl(\Pi, C)$ and can be restricted by imposing specific conditions, thus yielding the following basic derivation modes (for example, see [13] for formal definitions):

- asynchronous mode (abbreviated *asyn*): at least one rule is applied;
- sequential mode (*sequ*): only one rule is applied;
- maximally parallel mode (*max*): a non-extendable multiset of rules is applied;
- maximally parallel mode with maximal number of rules (max_{rules}): a non-extendable multiset of rules of maximal possible cardinality is applied;
- maximally parallel mode with maximal number of objects ($max_{objects}$): a non-extendable multiset of rules affecting as many objects as possible is applied.

In [2], these derivation modes are restricted in such a way that each rule can be applied at most once, thus yielding the set modes *sasyn*, *smax*, $smax_{rules}$, and $smax_{objects}$ (the sequential mode is already a set mode by definition):

- asynchronous set mode (abbreviated *sasyn*): at least one rule is applied, but each rule at most once;
- maximally parallel set mode (*smax*): a non-extendable set of rules is applied;
- maximally parallel set mode with maximal number of rules ($smax_{rules}$): a non-extendable set of rules of maximal possible cardinality is applied;
- maximally parallel set mode with maximal number of objects ($smax_{objects}$): a non-extendable set of rules affecting as many objects as possible is applied.

Let us denote the set of all multisets (possibly only sets) of rules applicable in a (tissue) P system Π to a given configuration C in the derivation mode δ by $Appl(\Pi, C, \delta)$. We immediately observe that $Appl(\Pi, C, asyn) = Appl(\Pi, C)$.
To collect the set and multiset derivation modes, we use the following notations:

$D_S = \{sequ, sasyn, smax, smax_{rules}, smax_{objects}\}$ and
$D_M = \{asyn, max, max_{rules}, max_{objects}\}$.

3.2 Standard Rule Variants

Non-cooperative rules have the form $a \to w$, where a is a symbol and w is a multiset, catalytic rules have the form $ca \to cw$, where the symbol c is called the *catalyst*, and cooperative rules have no restrictions on the form of the left-hand side. These types of rules will be denoted by *ncoo* (*non-cooperative*), *pcat* (*purely catalytic*), and *coo* (*cooperative*); if both non-cooperative and catalytic rules are allowed, we write *cat* (*catalytic*).

If the P system has more than one membrane, each symbol on the right-hand side may have assigned a target where the symbol has to be sent after the application of the rule. In tissue P systems this target is simply the number of the cell, whereas in hierarchical P systems the targets take into account the tree structure of the membranes:

here the symbol stays in the membrane where the rule is applied;
out the symbol is sent to the outer membrane, i.e., the membrane enclosing the membrane where the rule is applied;
in the symbol is sent to an inner membrane, i.e., a membrane enclosed by the membrane where the rule is applied;
in_j the symbol is sent to the inner membrane labeled by j.

3.3 Flattening

As many variants of P systems can be *flattened* to only one membrane, see [9], we often may assume the simplest membrane structure of only one membrane which in effect reduces the P system to a multiset processing mechanism, and, observing that $f = 1$, in what follows we then will use the reduced notation

$$\Pi = (V, T, w, R, \Longrightarrow_{\Pi,\delta}).$$

For a one-membrane system, the definitions for the *language generated by* Π and the *language accepted by* Π can be written in an easier way, i.e.,

$$L_{gen,\delta}(\Pi) = \left\{ v \in T^\circ \mid w \overset{*}{\Longrightarrow}_{\Pi,\delta} v \wedge \neg\exists z : \ v \Longrightarrow_{\Pi,\delta} z \right\} \text{ and}$$

$$L_{acc,\delta}(\Pi) = \left\{ w_0 \in T^\circ \mid \exists v : \ \left(w w_0 \overset{*}{\Longrightarrow}_{\Pi,\delta} v \wedge \neg\exists z : \ v \Longrightarrow_{\Pi,\delta} z \right) \right\}.$$

The family of languages of multisets generated by one-membrane P systems of type X in the derivation mode δ is denoted by $Ps_{gen,\delta}OP(X)$.

The family of languages of multisets accepted by one-membrane P systems of type X in the derivation mode δ is denoted by $Ps_{acc,\delta}OP(X)$.

In the following, we will mainly focus on the generative case, and when writing $Ps_\delta OP(X)$ we by default will mean $Ps_{gen,\delta}OP(X)$.

4 Some Well-Known Results

4.1 Non-Cooperative Rules

Using only non-cooperative rules leaves us on the level of semi-linear sets, as for the derivation with context-free rules (and non-cooperative rules correspond to those), the resulting derivation tree does not depend on an interpretation of a sequential or a parallel derivation of any kind. Moreover, context-free (string or multiset) languages are closed under projections, hence, taking (even only terminal) results out from a specific output membrane / cell does not make any difference. Therefore, we may state the following result:

Theorem 1. *For any* $Y \in \{N, Ps\}$ *and any* $n \geq 1$ *as well as any any derivation mode* $\delta \in D_S \cup D_M$,

$$Y_{gen,\delta}OP_n(ncoo) = Y_{gen,\delta}OtP_n(ncoo) = YREG.$$

4.2 The Importance of Using Catalysts

If in a one-membrane system we only have one catalyst c and only catalytic rules assigned to c, then this corresponds to a sequential use of noncooperative rules, which together with Theorem 1 yields the following result:

Theorem 2. *For any* $Y \in \{N, Ps\}$ *and any derivation mode* $\delta \in D_S \cup D_M$,

$$Y_{gen,\delta}OP(pcat_1) = Y_{gen,sequ}OP(pcat_1) = Y_{gen,sequ}OP(ncoo) = YREG.$$

Without additional control mechanisms, at least three catalysts are needed to obtain computational completeness for purely catalytic P systems using the derivation mode max, see [8]. In a more general way, the following results were already proved there:

Theorem 3. *For any* $d \geq 1$ *and any* $k \geq d + 2$,

$$Ps_{acc,max}OP(pcat_{k+1}) = Ps_{acc,max}OP(cat_k) = N^dRE.$$

Although not yet stated [8], we mention that these results are also valid when replacing the derivation mode max by any other maximally parallel (set) derivation mode, i.e., for any δ in

$$\{max, max_{rules}, max_{objects}, smax, smax_{rules}, smax_{objects}\}.$$

The complexity of the construction, for all these derivation modes, has been considerably reduced since the original paper from 2005, for example, see [2].

These results are obtained by simulating register machines, which in fact means

$\delta \in \{max, max_{rules}, max_{objects}\}$, for decrementing the number of a symbol a_r to carry out the decrement case of a SUB-instruction of a register machine, we cannot do this by a non-cooperative rule $a_r \to \lambda$, instead we have to use a catalytic rule $ca_r \to c$.

What happens in the case of two catalysts in purely catalytic P systems (and one catalyst in the case of catalytic P systems), is one of the most intriguing open problems in the area of P systems since long time, e.g., see [12], where it is shown that catalytic P systems with one catalyst can simulate partially blind register machines and partially blind counter automata.

With respect to the importance of using catalytic rules, the set derivation modes offer new opportunities, i.e., using specific control mechanisms they are not needed any more, as eliminating only one symbol a_r to carry out the decrement case of a SUB-instruction of a register machine now *can* be done by a non-cooperative rule $a_r \to \lambda$, because due to the set restriction this rule is not applied more than once.

5 Control Mechanisms

To reduce the number of catalysts needed for obtaining computational completeness, specific control mechanisms can be used. Some of these control mechanisms are considered in this section. For example, label selection or control languages allow for using only one catalyst (two catalysts) in (purely) catalytic P systems for getting computational completeness, for instance, see [7, 10, 11, 2]. With target agreement and maximally parallel set derivation modes, catalysts can even be avoided completely, only non-cooperative rules are needed.

5.1 P Systems with Label Selection

For all the variants of (tissue) P systems of type X, we may consider labeling all the rules in the sets $R_1, \ldots, R_m$ in a one-to-one manner by labels from a set H and taking a set W containing subsets of H. In any transition step of a *(tissue) P system with label selection* Π we first select a set of labels $U \in W$ and then, in the given derivation mode, we apply a non-empty multiset R of rules such that all the labels of these rules from R are in U.

The families of sets $Y_{\gamma,\delta}(\Pi)$, $Y \in \{N, Ps\}$, $\gamma \in \{gen, acc\}$, and $\delta \in D_M \cup D_S$ computed by (tissue) P systems with label selection with at most m membranes and rules of type X are denoted by $Y_{\gamma,\delta}OP_m(X, ls)$ $(Y_{\gamma,\delta}OtP_m(X, ls))$.

Theorem 4. *$Y_{\gamma,\delta}OP(cat_1, ls) = Y_{\gamma,\delta}OP(pcat_2, ls) = YRE$ for any $Y \in \{N, Ps\}$, $\gamma \in \{gen, acc\}$, and any maximally parallel (set) derivation mode δ.*

The proof given in [11] for the maximally parallel mode *max* can be taken over for the other maximally parallel (set) derivation modes word by word; the only difference again is that in set derivation modes, in non-successful computations where more than one trap symbol # has been generated, the trap rule $\# \to \#$ is only applied once.

5.2 Controlled (Tissue) P Systems and Time-Varying (Tissue) P Systems

Another method to control the application of the labeled rules is to use control languages (see [14] and [1]).

In a *controlled (tissue) P system* Π, in addition we use a set H of labels for the rules in Π, and a string language L over 2^H (each subset of H represents an element of the alphabet for L) from a family FL. Every successful computation in Π has to follow a control word $U_1 \ldots U_n \in L$: in transition step i, only rules with labels in U_i are allowed to be applied (in the underlying derivation mode, for example, *max* or *smax*), and after the n-th transition, the computation halts; we may relax this end condition, i.e., we may stop after the i-th transition for any $i \leq n$, and then we speak of *weakly controlled P systems*. If $L = (U_1 \ldots U_p)^*$, Π is called a *(weakly) time-varying (tissue) P system*: in the computation step $pn + i$, $n \geq 0$, rules from the set U_i have to be applied; p is called the *period*.

The family of sets $Y_{\gamma,\delta}(\Pi)$, $Y \in \{N, Ps\}$, computed by (weakly) controlled P systems and (weakly) time-varying P systems with period p, with at most m membranes and rules of type X as well as control languages in FL is denoted by $Y_{\gamma,\delta}OP_m(X, C(FL))$ $(Y_{\gamma,\delta}OP_m(X, wC(FL)))$ and $Y_{\gamma,\delta}OP_m(X, TV_p)$ $(Y_{\gamma,\delta}OP_m(X, wTV_p))$, respectively, for $\gamma \in \{gen, acc\}$ and $\delta \in D_M \cup D_S$. Similar notations hold for tissue P systems.

Theorem 5. *$Y_{\gamma,\delta}OP(cat_1, \alpha TV_6) = Y_{\gamma,\delta}OP(pcat_2, \alpha TV_6) = YRE$, for any $\alpha \in \{\lambda, w\}$, $Y \in \{N, Ps\}$, $\gamma \in \{gen, acc\}$, and*

$$\delta \in \{max, max_{rules}, max_{objects}, smax, smax_{rules}, smax_{objects}\}.$$

The proof given in [11] for the maximally parallel mode *max* again can be taken over for the other maximally parallel (set) derivation modes word by word, e.g., see [2].

5.3 Target Selection

In P systems with target selection, all objects on the right-hand side of a rule must have the same target, and in each derivation step, for each region a (multi)set of rules – non-empty if possible – having the same target is chosen. In [2] it was shown that for P systems with target selection in the derivation mode *smax* **no**

Theorem 6. *For any* $Y \in \{N, Ps\}$,

$$Y_{gen,smax}OP\,(ncoo, target\ selection) = YRE.$$

Theorem 7. *For any* $Y \in \{N, Ps\}$,

$$Y_{detacc,smax_{rules}}OP\,(ncoo, target\ selection) = YRE.$$

In contrast to all the other variants of P systems, P systems with target selection really take advantage of the membrane structure, no flattening is used or even reasonable. In that sense, this variant of P systems reflects the spirit of membrane systems with a non-trivial membrane structure in the best way.

6 The Strangeness of Minimal Parallelism

There is another derivation mode known from literature, which has two possible definitions, but these two variants unfortunately do not yield the same results.

Following the definition given in [13], for the minimally parallel derivation mode (*min*), we need an additional feature for the set of rules R used in the overall (tissue) P system, i.e., we consider a partitioning θ of R into disjoint subsets R_1 to R_h. Usually, this partitioning of R may coincide with a specific assignment of the rules to the membranes or cells.

There are now several possible interpretations of this minimally parallel derivation mode which in an informal way can be described as applying multisets such that from every set R_j, $1 \leq j \leq h$, at least one rule – if possible – has to be used (e.g., see [4]). Yet this *if possible* allows for two possible interpretations:

Minimal parallelism as a restriction of asyn

As defined in [13], we start with a multiset R' of rules from $Appl(\Pi, C, asyn)$ and only take it if it cannot be extended to a multiset R' of rules from $Appl(\Pi, C, asyn)$ by some rule from a set R_j from which so far no rule is in R'.

Minimal parallelism as an extension of smax

We start with a set R' of rules from $Appl(\Pi, C, smax_\theta)$, where the notion $smax_\theta$ indicates that we are using *smax* with respect to the partitioning of R into the subsets R_1 to R_h, and then possibly extend it to a multiset R'' of rules from $Appl(\Pi, C, asyn)$ which contains R'. This definition finally was used in [18] without using the notion *smax*, because at the moment when this handbook was written the notion of maximally parallel set derivation modes had not been invented yet. Moreover, the use of the notion *smax* so far was restricted to the discrete topology

Example 1. Consider the one-membrane P system working in the *min*-mode

$$\Pi = (V = \{a, b\}, T = \{b\}, w = aa, R = R_1 \cup R_2, \Longrightarrow_{\Pi,min})$$

with $R_1 = \{a \to bb\}$ and $R_2 = \{a \to bbb\}$.

Starting from *smax*, we get only one set of rules, i.e., $R' = \{a \to bb, a \to bbb\}$, whose application yields the result b^5.

In the case of starting with *asyn*, we may use one of the two rules twice, thus also getting the results b^4 and b^6.

Hence, when two rules are competing for the same objects, the results obtained with the two different definitions may be different, where the set of results obtained when using the first definition will always include the results obtained by the second definition.

The condition that the sets R_j, $1 \leq j \leq h$, have to be disjoint may be alleviated, for example, see [3].

A special variant of the minimally parallel derivation mode, with the sets R_j, $1 \leq j \leq h$, not being required to be disjoint, is the mode min_1, which in fact means that we stay with $smax_\theta$. Now let $smax_{\theta,k}$ denote a partioning θ with k sets of rules. As an interesting result we then get the interpretation of a purely catalytic P system using *max* as a P system using min_1 with the partitioning R_j, $1 \leq j \leq k$, where R_j is the set of non-cooperative rules $a \to u$ representing the corresponding catalytic rules $c_j a \to c_j u$. Denoting a partitioning in k sets of rules by θ_k, we obtain the following result:

Theorem 8. *For any* $d \geq 1$ *and any* $k \geq d + 3$,

$$Ps_{acc,min_1}OP\,(ncoo, \theta_k) = Ps_{gen,min_1}OP\,(ncoo, \theta_3) = N^dRE.$$

7 Conclusion

In this paper the effects of using different derivation modes on the generative and accepting power of many variants of hierarchical P systems and tissue P systems have been illustrated. Especially some differences between the maximally parallel derivation modes and the maximally parallel set derivation modes have been exhibited.

We have also given an overview on some control mechanisms used for (tissue) P systems.

Many more relations between derivation modes could have been discussed, but this would have gone much beyond a conference paper.

Acknowledgements

some previous years, and they are based on many discussions with Artiom Alhazov, Sergiu Ivanov, and Sergey Verlan, but also other colleagues from the P community, especially with Gheorghe Păun.

References

1. Alhazov, A., Freund, R., Heikenwalder, H., Oswald, M., Rogozhin, Yu., Verlan, S.: Sequential P systems with regular control. In: Csuhaj-Varjú, E., Gheorghe, M., Rozenberg, G., Salomaa, A., Vaszil, Gy. (eds.) Membrane Computing – 13th International Conference, CMC 2012, Budapest, Hungary, August 28-31, 2012, Revised Selected Papers, Lecture Notes in Computer Science, vol. 7762, pp. 112-127. Springer (2013). https://doi.org/10.1007/978-3-642-36751-9 9
2. Alhazov, A., Freund, R., Verlan, S.: P systems working in maximal variants of the set derivation mode. In: Leporati, A., Rozenberg, G., Salomaa, A., Zandron, C. (eds.) Membrane Computing - 17th International Conference, CMC 2016, Milan, Italy, July 25-29, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10105, pp. 83-102. Springer (2017). https://doi.org/10.1007/978-3-319-54072-6 6
3. Alhazov, A., Oswald, M., Freund, R., Verlan, S.: Partial halting and minimal parallelism based on arbitrary rule partitions. Fundam. Inform. 91(1), 17-34 (2009). https://doi.org/10.3233/FI-2009-0031
4. Ciobanu, G., Pan, L., Păun, Gh.., Pérez-Jiménez, M.: P systems with minimal parallelism. Theoretical Computer Science 378(1), 117-130 (2007). https://doi.org/10.1016/j.tcs.2007.03.044
5. Dassow, J., Păun, Gh.: Regulated Rewriting in Formal Language Theory. Springer (1989), https://www.springer.com/de/book/9783642749346
6. Freund, R.: P systems working in the sequential mode on arrays and strings. Int. J. Found. Comput. Sci. 16(4), 663-682 (2005). https://doi.org/10.1142/S0129054105003224
7. Freund, R.: Purely catalytic P systems: Two catalysts can be suffcient for computational completeness. In: Alhazov, A., Cojocaru, S., Gheorghe, M., Rogozhin, Yu. (eds.) CMC14 Proceedings – The 14th International Conference on Membrane Computing, Chşinău, August 20-23, 2013. pp. 153-166. Institute of Mathematics and Computer Science, Academy of Sciences of Moldova (2013), http://www.math.md/cmc14/CMC14 Proceedings.pdf
8. Freund, R., Kari, L., Oswald, M., Sosík, P.: Computationally universal P systems without priorities: two catalysts are suffcient. Theoretical Computer Science 330(2), 251-266 (2005). https://doi.org/10.1016/j.tcs.2004.06.029
9. Freund, R., Leporati, A., Mauri, G., Porreca, A.E., Verlan, S., Zandron, C.: Flattening in (tissue) P systems. In: Alhazov, A., Cojocaru, S., Gheorghe, M., Rogozhin, Yu., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing, Lecture Notes in Computer Science, vol. 8340, pp. 173-188. Springer (2014). https://doi.org/10.1007/978-3-642-54239-8 13
10. Freund, R., Oswald, M.: Catalytic and purely catalytic P automata: control mechanisms for obtaining computational completeness. In: Bensch, S., Drewes, F., Freund,

11. Freund, R., Păun, Gh.: How to obtain computational completeness in P systems with one catalyst. In: Neary, T., Cook, M. (eds.) Proceedings Machines, Computations and Universality 2013, MCU 2013, Zürich, Switzerland, September 9-11, 2013. EPTCS, vol. 128, pp. 47-61 (2013). https://doi.org/10.4204/EPTCS.128.13
12. Freund, R., Sosík, P.: On the power of catalytic P systems with one catalyst. In: Rozenberg, G., Salomaa, A., Sempere, J.M., Zandron, C. (eds.) Membrane Computing - 16th International Conference, CMC 2015, Valencia, Spain, August 17-21, 2015, Revised Selected Papers. Lecture Notes in Computer Science, vol. 9504, pp. 137-152. Springer (2015). https://doi.org/10.1007/978-3-319-28475-0 10
13. Freund, R., Verlan, S.: A formal framework for static (tissue) P systems. In: Eleftherakis, G., Kefalas, P., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Membrane Computing, Lecture Notes in Computer Science, vol. 4860, pp. 271-284. Springer (2007). https://doi.org/10.1007/978-3-540-77312-2 17
14. Krithivasan, K., Păun, Gh., Ramanujan, A.: On controlled P systems. Fundam. Inform. 131(3-4), 451-464 (2014). https://doi.org/10.3233/FI-2014-1025
15. Minsky, M.L.: Computation. Finite and In
nite Machines. Prentice Hall, Englewood Cliffs, NJ (1967)
16. Păun, Gh.: Computing with membranes. Journal of Computer and System Sciences 61(1), 108-143 (2000). https://doi.org/10.1006/jcss.1999.1693
17. Păun, Gh.: Membrane Computing: An Introduction. Springer (2002). https://doi.org/10.1007/978-3-642-56196-2
18. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press (2010)
19. Rozenberg, G., Salomaa, A. (eds.): Handbook of Formal Languages. Springer (1997). https://doi.org/10.1007/978-3-642-59136-5
20. The P Systems Website. http://ppage.psystems.eu/